



[Home](#) [Conference](#)
[Program and activities](#)
[Proceedings](#) [Logo projects](#)
[About us & Sponsors](#)

[Home](#) ▶ [Proceedings](#) ▶ [Index of authors](#)

Index of authors

[A](#) | [B](#) | [C](#) | [D](#) | [E](#) | [F](#) | [G](#) | [H](#) | [I](#) | [J](#) | [K](#) | [L](#) | [M](#) | [N](#) | [O](#) | [P](#) | [R](#) | [S](#) | [T](#) | [U](#) | [V](#) | [W](#) | [Y](#)

A

- Sueli de Abreu
[Constructing a space: computational environment applied to a psychopedagogic evaluation proces](#) (Paper)
- Efi Alexopoulou
[Changing a half-baked 3d navigational game](#) (Paper)
- Dimitris Alimisis
[Robotics & Constructivism in Education: the TERECOP project](#) (Paper)
[Teacher Education to Promote Constructivist Use of ICT: Study of a Logo-based Project](#) (Paper)
- Javier Arlegui
[Robotics & Constructivism in Education: the TERECOP project](#) (Paper)
- Noboru Ashida
[NeGAS: Authoring System for 3DCG using extended turtle metaphor](#) (Paper)

B

- Maria Vittoria Bedin
[Turtles with Roles: an application of an Object Oriented Pattern in JxLogo](#) (Paper)
- Meurig Beynon
[Dependency by definition in Imagine-d Logo: applications and implications](#) (Paper)
- Pavel Boytchev
[Design and Implementation of a Logo-based Computer Graphics Course](#) (Plenary Presentation)
[Words are silver, mouse-clicks are gold?](#) (Plenary Presentation)

C

- Enís Castellanos
[The use of the transcendental method for helping students to learn with Logo](#) (Paper)
- Toni Chehlarova
[Words are silver, mouse-clicks are gold?](#) (Plenary Presentation)
- SookKyoung Choi
[Comprehension of OOP Concepts using Dolittle in elementary school](#) (Paper)
- James Edward Clayson
[Radical bricolage: making the liberal arts coherent](#) (Plenary Presentation)
- Paulo A. Condado
[Learning Logo at a high school: Constructionism versus Objectivism](#) (Paper)
- Verónica Contreras-Valencia
[A Logo-based didactic sequence for the learning of numerical systems by high-school](#)



[students](#) (Paper)
Secundino Correia
[Natural Visual Programming Languages](#) (Paper)
Tiago Correia
[Natural Visual Programming Languages](#) (Paper)

D



Tünde Dancsó
[The ICT literacy in the OECD countries](#) (Paper)
G. Barbara Demo
[Contributing to the Development of Linguistic and Logical Abilities through Robotics](#) (Paper)
Mícheál O Dúill
[Logo: an object to think with](#) (Paper)

E



Tina Ebey
[Knowledge vs. Creativity: A False Dichotomy](#) (Panel Discussion)
Rocío Escobedo
[Designing Logo Interactive Activities for the Mathematics Programs of the Mexican School System](#) (Plenary Presentation)
Elizabeth Esparza-Cruz
[Stimulation of the development of spatial Euclidean relations through Logo programming activities](#) (Paper)

F



Károly Farkas
[Magic Imagine](#) (Paper)
Wallace Feurzeig
[Knowledge vs. Creativity: A False Dichotomy](#) (Panel Discussion)
[Toward a Culture of Creativity: A Personal Perspective on Logo's Early Years, Legacy, and Ongoing Potential](#) (Keynote Presentation)
Izabella Foltynowicz
[Cycloids and limaçons in the turtle graphics](#) (Paper)
[Cycloids and limaçons in the turtle graphics](#) (Workshop)
Stassini Frangou
[Robotics & Constructivism in Education: the TERECoP project](#) (Paper)
Gerald Futschek
[Knowledge vs. Creativity: A False Dichotomy](#) (Panel Discussion)
[Logo-like Learning of Basic Concepts of Algorithms - Having Fun with Algorithms](#) (Paper)

G



Katerina Glezou
[A novel didactical approach of the decision structure for novice programmers](#) (Paper)
Paul Goldenberg
[Knowledge vs. Creativity: A False Dichotomy](#) (Panel Discussion)
Sophia Gorlitskaya
[Practical Logo Team Competitions](#) (Paper)
Maria Grigoriadou
[A novel didactical approach of the decision structure for novice programmers](#) (Paper)
Paulo C. M. Guerreiro
[Disguised Programming as a Teaching Aid for Students with Special Needs](#) (Paper)

Ján Guniš

[PALMA junior – programming competition in Imagine](#) (Paper)

Valentína Gunišová

[PALMA junior – programming competition in Imagine](#) (Paper)

Alessio Gutiérrez-Gómez

[Painless Trigonometry: a tool-complementary school mathematics project](#) (Paper)

H



Brian Harvey

[How Not to Use Computers to Teach Kids](#) (Plenary Presentation)

[Knowledge vs. Creativity: A False Dichotomy](#) (Panel Discussion)

Celia Hoyles

[Knowledge vs. Creativity: A False Dichotomy](#) (Panel Discussion)

[Policy and Practice: a Logo Vision](#) (Keynote Presentation)

Andrea Hricová

[PALMA junior – programming competition in Imagine](#) (Paper)

Andrea Hrušecká

[Imagine Logo based magazine](#) (Paper)

[Logo in Lifelong Learning](#) (Paper)

Jana Hruskova

[Multimedia application in teaching computer skills](#) (Paper)

I



Vessela Ilieva

[ICT competition for kids](#) (Paper)

Ivailo Ivanov

[ICT competition for kids](#) (Paper)

J



JeongA Jang

[Comprehension of OOP Concepts using DOLITTLE in elementary school](#) (Paper)

Jesús Jiménez-Molotla

[Painless Trigonometry: a tool-complementary school mathematics project](#) (Paper)

Wanda Jochemczyk

[Logo in Polish Schools - cases](#) (Paper)

[Turtle and children on Moodle e-learning platform](#) (Paper)

Ian Jones

[A Logo-based Task for Arithmetical Activity](#) (Paper)

K



Martina Kabátová

[LEGO Robots for Classroom Experiments](#) (Workshop)

Ken Kahn

[Should LOGO Keep Going FORWARD 1?](#) (Plenary Presentation)

[The BehaviourComposer: A way for students to build computer programs without first learning to program](#) (Workshop)

Ivan Kalaš

[Editorial](#)

[Knowledge vs. Creativity: A False Dichotomy](#) (Panel Discussion)

Toshiyuki Kamada

[The Effect of Learning Programming with Autonomous Robots for Elementary School Students](#) (Paper)

Susumu Kanemune

[The Effect of Learning Programming with Autonomous Robots for Elementary School Students](#) (Paper)

Sofia Kasola
[Using Microworlds-Pro to support Effectively the educational Process](#) (Paper)

HanSung Kim
[Comprehension of OOP Concepts using Dolittle in elementary school](#) (Paper)

JongHye Kim
[Comprehension of OOP Concepts using Dolittle in elementary school](#) (Paper)

Eric Klopfer
[Knowledge vs. Creativity: A False Dichotomy](#) (Panel Discussion)
[StarLogo TNG – Making game and simulation development accessible to students and teachers](#) (Keynote Presentation)

Márta Kőrös-Mikis
[A Glance at the Foyer of the Future](#) (Paper)

Witold Kranas
[Logo in Polish Schools - cases](#) (Paper)

Yasushi Kuno
[The Effect of Learning Programming with Autonomous Robots for Elementary School Students](#) (Paper)

Shuji Kurebayashi
[The Effect of Learning Programming with Autonomous Robots for Elementary School Students](#) (Paper)

Irina Kuznetsova
[Practical Logo Team Competitions](#) (Paper)

Chronis Kynigos
[Changing a half-baked 3d navigational game](#) (Paper)
[Half-baked Logo microworlds as boundary objects in integrated design](#) (Keynote Presentation)
[Turtle's navigation and manipulation of geometrical figures constructed by variable processes in a 3d simulated space](#) (Paper)

L

Maria Latsi
[Turtle's navigation and manipulation of geometrical figures constructed by variable processes in a 3d simulated space](#) (Paper)

WonGyu Lee
[Comprehension of OOP Concepts using Dolittle in elementary school](#) (Paper)

Daniela Lehotská
[Imagine Logo based magazine](#) (Paper)

Fernando G. Lobo
[Disguised Programming as a Teaching Aid for Students with Special Needs](#) (Paper)
[Learning Logo at a high school: Constructionism versus Objectivism](#) (Paper)

Gabriela Lovászová
[Introduction to Nondeterminism](#) (Paper)

M

Giovanni Marcianó
[Contributing to the Development of Linguistic and Logical Abilities through Robotics](#) (Paper)

Christos Markopoulos
[Changing a half-baked 3d navigational game](#) (Paper)

Sylvia Martinez
[Student-Centered Support Systems to Sustain Logo-like Learning](#) (Paper)

Michele Moro
[Robotics & Constructivism in Education: the TERECOP project](#) (Paper)
[Turtles with Roles: an application of an Object Oriented Pattern in JxLogo](#) (Paper)



[Department of Informatics Education](#)

2005
2003
2001
1999
1997
1995
1993
1991
1989
1987



N



Ingrid Nagyova
[Multimedia application in teaching computer skills](#) (Paper)

Mitaro Namiki
[Musical Method in Programming Education](#) (Paper)

Erich Neuwirth
[Musical abstractions and programming paradigms](#) (Plenary Presentation)

Stéphane Norte
[Learning Logo at a high school: Constructionism versus Objectivism](#) (Paper)

Richard Noss
[Another look back and forward](#) (Keynote Presentation)

O



Katarzyna Olędzka
[Logo in Polish Schools - cases](#) (Paper)
[Turtle and children on Moodle e-learning platform](#) (Paper)

P



Attila Paksi
[Logo practice: from “turtling” to interactivity](#) (Plenary Presentation)

Viera Palmárová
[Introduction to Nondeterminism](#) (Paper)

Chris Panagiotakopoulos
[Using Microworlds-Pro to support Effectively the educational Process](#) (Paper)

Kyparissia Papanikolaou
[Robotics & Constructivism in Education: the TERECOP project](#) (Paper)

Janka Pekárová
[LEGO Robots for Classroom Experiments](#) (Workshop)
[Logo in Lifelong Learning](#) (Paper)

Andrej Petráš
[Process of educational software development](#) (Paper)

Pavel Petrovič
[Program your NXT robot with Imagine](#) (Paper)
[Programming Robots in Logo](#) (Workshop)

Alfredo Pina
[Robotics & Constructivism in Education: the TERECOP project](#) (Paper)

Panagiotis Pintelas
[Using Microworlds-Pro to support Effectively the educational Process](#) (Paper)

R



Brigitta Réthey-Prikkel
[Design & evaluation of maths related programs for special education](#) (Paper)

Chris Roe
[Dependency by definition in Imagine-d Logo: applications and implications](#) (Paper)

S



Ana Isabel Sacristán
[A Logo-based didactic sequence for the learning of numerical systems by high-school students](#) (Paper)
[Designing Logo Interactive Activities for the Mathematics Programs of the Mexican School System](#) (Plenary Presentation)

[Painless Trigonometry: a tool-complementary school mathematics project](#) (Paper)
[Stimulation of the development of spatial Euclidean relations through Logo programming activities](#) (Paper)

Evgenia Sendova
[Knowledge vs. Creativity: A False Dichotomy](#) (Panel Discussion)
[Words are silver, mouse-clicks are gold?](#) (Plenary Presentation)

Simonetta Siega
[LOGO and NOCbaby – programming Microworlds and Robots according to Logo Philosophy](#) (Workshop)

Lubomír Šnajder
[PALMA junior – programming competition in Imagine](#) (Paper)

Sergei Soprunov
[Turtle competitions in MicroWorlds EX](#) (Paper)

Gary S. Stager
[Towards the Construction of a Language for Describing the Learning Potential of Computing Activities](#) (Plenary Presentation)

Zuzana Szabóová
[PALMA junior – programming competition in Imagine](#) (Paper)

T



Ildikó Tasnádi
[Magic Imagine](#) (Paper)

Takeo Tatsumi
[Musical Method in Programming Education](#) (Paper)

Nicolás Tlachy
[Designing Logo Interactive Activities for the Mathematics Programs of the Mexican School System](#) (Plenary Presentation)

Peter Tomcsányi
[Synchronising processes in Imagine Logo: Why and How](#) (Paper)

Monika Tomcsányiová
[Transforming activities from workbooks for preschoolers into computer games](#) (Paper)

Adriana Sobreira Torres
[Creating games in Imagine for pre school](#) (Paper)

Dana Tržilová
[Procedural building-up of geometrical object in concepts and strategies of primary school pupils](#) (Paper)

Katsuhide Tsushima
[NeGAS: Authoring System for 3DCG using extended turtle metaphor](#) (Paper)

Márta Turcsányi-Szabó
[Design and evaluation of Maths related programs for special education](#) (Paper)
[Knowledge vs. Creativity: A False Dichotomy](#) (Panel Discussion)
[Logo practice: from “turtling” to interactivity](#) (Plenary Presentation)

U



Masayuki Ueno
[NeGAS: Authoring System for 3DCG using extended turtle metaphor](#) (Paper)

V



José Armando Valente
[ICT Appropriation by Digitally Excluded Communities: the role of the learning context](#) (Plenary Presentation)

Jiří Vaníček
[Procedural building-up of geometrical object in concepts and strategies of primary school pupils](#) (Paper)

Vítor M. M. Vieira

[Disguised Programming as a Teaching Aid for Students with Special Needs](#) (Paper)

Alla A.Vitukhnovskaya

[MicroWorlds Potential for Creating Educational Software for School](#) (Paper)

W



Shinjiro Wada

[NeGAS: Authoring System for 3DCG using extended turtle metaphor](#) (Paper)

Michael Weigend

[Logo Nanoworlds](#) (Paper)

Ronald Weiss

[Programming Robots in Logo](#) (Workshop)

Y



Elena Yakovleva

[Turtle competitions in MicroWorlds EX](#) (Paper)

Constructing a space: computational environment applied to a psychopedagogic evaluation process

Sueli de Abreu, sabreu@cnotinfor.com.br
NCE/UFRJ, CNOTINFOR BRASIL

Abstract

Works performed by different authors pointed out the possibility of understanding people's way of thinking through the use of a computer since you can follow their behaviour, and also see the final result of their representations on screen.

Psychopedagogic clinic needs this kind of tools – Kid Pix Deluxe® and Storybook Weaver Deluxe® - for allowing readings and recordings of situations or actions occurring during the evaluation process.

Constructing a space was developed based on the evaluation of a software and the need for technological resources of psychopedagogic clinic, offering means for the clinical diagnostic process of children from 7 to 11 years-old. Its aim is to provide the psychopedagogue with data about the child in process of knowledge construction in several dimensions: rational, desiderative and relational (Silva, 1998) or cognitive, emotional, pedagogic and social (Weiss, 1992). This software intends to create an environment where the dimensions aforementioned can be observed and recorded in an integrated way. Thus, it should be open enough for allowing the psychopedagogue to propose several activities.

This work is built upon a master degree dissertation** and presents the software *Constructing a space* (developed in Imagine Logo), its development and substantiation, and a brief study with psychopedagogues with the aim of their evaluation.

** Dissertation submitted to “Núcleo de Computação Eletrônica da Universidade Federal do Rio de Janeiro” – UFRJ, as required to obtain Master Degree in Informatics, in 2004.

Keywords

evaluation process; psychopedagogic diagnostic; psychopedagogic environment; LOGO microworlds, primary education

The psychopedagogic clinic

Psychopedagogy was born from the necessity of a better understanding of the learning process and it became a specific area of study concerned to knowledge in different fields and, also, to create its own object of study (Bossa, 2000).

The psychopedagogic clinic corresponds to one of its fields of performance, which objectives are to diagnosis and to deal with the emergent symptoms in the learning process. The psychopedagogic diagnoses investigate and analyze which are the obstacles that prevent children from learning, learning with great difficulty and/or learning in a slow pace. It also clarifies a complaint of the children themselves, their family or the school. (Weiss apud Scoz, 1991).

Psychopedagogy in Brazil has been developing - for over thirty years - its own theoretical background. "It is a new area of knowledge, which brings in itself the origins and contradictions of a performance to interdisciplinary, needing much theoretical reflection and searches" (Bossa, op.cit, p.13).

The psychopedagogy is occupied of the human learning, what happened from a demand - the learning problem, placing in a territory little explored, situated beyond the limits of Psychology and the proper Pedagogy - and evolves due the existence of resources, to take care of this demand thus, consisting, in one practical. As it is worried about the learning problem, it must occupy initially of the learning process. Therefore we see that psychopedagogy studies the characteristics of human learning: how we learn, how this learning varies in time and is conditional for some factors, how the alterations in learning are produced, how to recognize them, to treat them and prevent them. This object of study, that is a person to be studied by another person, acquires specific characteristics depending on the clinical or preventive work (Bosssa, op.cit. P. 21).

The distinction between the clinical and the preventive work is basic. The former aims at searching for the obstacles and the causes for the already installed problem of learning; and the latter aims at studying the developing conditions of learning, pointing ways to develop a more efficient learning process. Considering the definition of Bossa (Idem, p.21) on the two fields of performance of psychopedagogy:

The clinical work is defined in the relation between a person with its personal history and its modality of learning, searching to understand the message of another person, implicit in not-learning. In this process, where investigator and object-person of study interact constantly, any alteration becomes an issue for study for Psychopedagogy. This means that, in this modality of work, the professional must understand what the person learns, how it learns and why, beyond perceiving the dimension of the relation between psychopedagogue and person in order to favor learning. In the preventive approach the institution, while physical and psychic space of the learning is object of study of Psychopedagogy, once that is evaluated the didactic and methodology processes and the institutional dynamics that intervene with the learning process.

In the clinical exercise, the psycopedagogue must recognize children's process of learning, their limits and their abilities - mainly the intrapersonal and the interpersonal ones. Therefore, its object of study is the other person, being essential for the psycopedagogic practice the knowledge and the possibility of differentiation of what is pertinent to one another. "This interrelation of persons, where one looks forward to know the other in what it is the obstacle to its learning, implies a very complex thematic" (Bosssa, op.cit. P. 23). Psycopedagogues have as their duties to identify the structure of the person, the children' structural transformations over time, the influences of children's environment in these transformations and the relationship of these transformations with learning. This knowledge demands from the psycopedagogue the better understanding of the learning process and their interrelations with other factors that may influence it, such as the emotional, social, pedagogical and organic ones. Knowing the beddings of Psycopedagogy implies reflection upon its theoretical origins and to understand how different areas of knowledge are transformed into a very particular and new theoretical framework.

Psychology and Pedagogy are the core areas of psychopedagogy, but they are not enough to build up a solid base for this new field of human knowledge. For this reason, it was necessary to appeal to other areas, such as Philosophy, Neurology, Sociology, Psycholinguistics and Psychoanalysis, to reach a multifaceted understanding of the learning process.

The areas of study rose from the observation of different dimensions of the learning process which are organic, cognitive, emotional, social and pedagogic. According to Weiss (1992:22), "the interrelation of these aspects will help us build a gestaltic view of pluricasuality of the phenomenon, making possible a global approach of the subject and their multiple facets."

The emotional dimension is connected to the affective development and its relationship with the knowledge construction of the children and their expressions via graphic and written productions. Psychoanalysis is the area which gives theoretical support to this dimension, dealing with the unconscious aspects involved in the learning process, allowing psychopedagogues to take into account the desirous face of children. The social dimension is related to the society's own perspectives where family, the social group and the school are part of it. Social Psychology is the area responsible for providing theoretical base of this dimension. The cognitive dimension is connected to the cognitive structures of children applied to different situations. The Pedagogic dimension has to do with the content, methodology, classroom dynamic, educational techniques and evaluations children have to perform throughout their formal education at school. The Organic dimension deals with the physiobiological constitution of the learning children. The Linguistics is the area of study which pervases all the other dimensions previously mentioned.

However, none of the area mentioned came up to answer to specific questions about the human learning process. Though, all of them provide means to reflect critically upon the answers that emerge from questioning and it is also a valuable background for operating in the field of Psychopedagogy.

Figure [1] below, extracted from Weiss (*idem*, p.25) illustrates the relationship among the different areas that build up the theoretical background of Clinical Pedagogy; i.e, its delimitation.

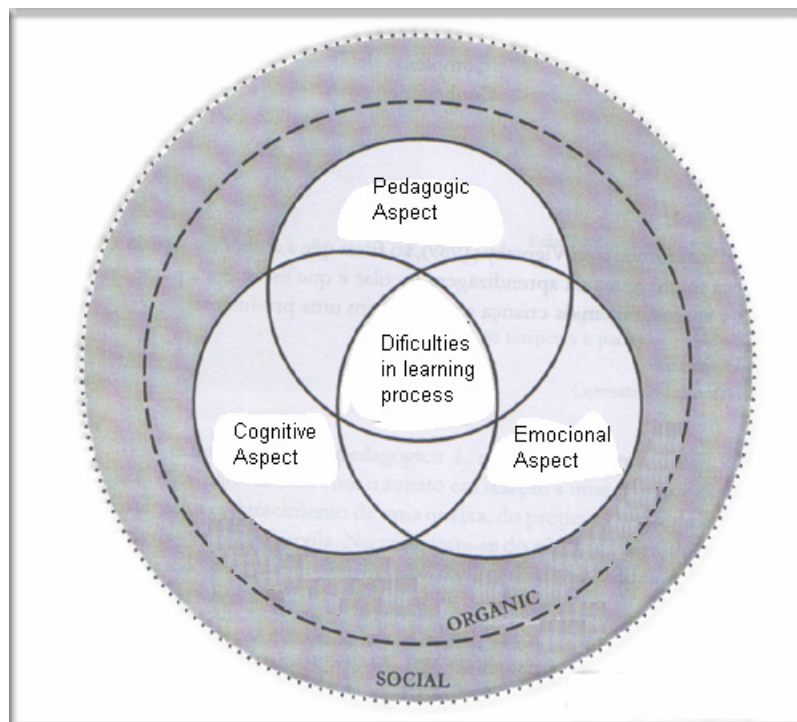


Figure 1 – Aspects related to the learning process.

The psychopedagogic diagnosis

The diagnosis is, per se, an investigation that follows fixed parameters defined by psychopedagogues who are in search of the causes of a complaint made by the children, their family and their school. The focus of the diagnosis is the obstacle within the learning process. A difficulty observed by the children, their school, their family and their social group is the symptom.

After acknowledging the obstacle in the children's learning process, a question arises: is it a diversion compared to what? This parameterization seems to be fundamental for the diagnosis process because it will define both the quality and importance of this diversion to formal development. After careful analysis, it is possible to come up with a plan to diagnose the problem.

To set in move the diagnosis process, the therapist needs to consider two main axes of analysis which should interact between themselves in a dialogic way: a) the horizontal axis – the a-historical perspective and b) the vertical axis – the historical perspective. In the horizontal axis, the therapist explores the present situation, looking for the existing causes as symptoms. In the vertical axis, the aim is to understand the general construction of the children a certain context.

According to Weiss (1992), the main objective of a psychopedagogic diagnosis is to identify the diversions and obstacles in the children's learning model¹ which are impeding them to develop themselves as socially expected. To understand a child's learning model, the psychopedagogue needs to gather data from the family, the school and the child under therapy, taking into account both a-historical and historical perspectives.

From the integration of the data collected arises the prognostic and material necessary for formulating the final hypothesis, which will be used during the diagnostic feedback interview.

The computational environment in the psychopedagogic clinic

Is it possible to diagnose children through their interaction with the computer? How can this be done?

Woks performed by different authors pointed out the possibility of understanding people's way of thinking through the use of a computer since you can follow their behaviour, and also see the final result of their representations on screen. The learning process becomes visible to the psychopedagogue via representational schemas' observation.

Our own experience as psychopedagogue using technology (since 1992) and the literature review, allowed us to conclude that the use of Informational Technology makes possible for children deal with their own thoughts in a different way. This new way of dealing with thoughts and reasoning – that flows differently from the paper-and-pencil school routines – may bring a new dimension to the clinic psychopedagogic.

The Kid Pix Deluxe® and the Storybook Weaver Deluxe® are tools that offer multiple resources while they may be presented as blank pages from which it is possible to create. From objects – their own construction blocks - children create new things. In this construction process, children start to reveal important aspects of their learning processes, as it happens in metacognition, planning, error observation and error correction.

¹ Learning Model means a dynamic group of structures that frame children's knowledge, their learning styles, their mobility, the cognitive performance, the acquired habits, their present motivations, their anxiety, and their conflicts and their defenses related to learning, their bonds with general knowledge and the object of formal knowledge, in particular, the meaning given to the formal education by children, their family and school " (Weiss, 2000, p.32).

Softwares like Kid Pix Deluxe® and Storybook Weaver Deluxe® are used in psychopedagogic clinic, during the diagnosis period, and are pointed by psychopedagogues (Oliveira, 1996; and Weiss, 2000) as one of the most used resources because of the resources they present.

However, from a psychopedagogic point of view, there is not a specific structured environment in these softwares aiming the diagnostic process with resources for registering complex and dynamic actions occurring during the process itself. The psychopedagogue should create a structure based on some available tools in the software and record /store the children's interaction process with themselves, using different means.

In the psychopedagogic diagnosis, we need to make use of different tools – in addition to the ones presented in Kid Pix Deluxe® and Storybook Weaver Deluxe® - to reach a more accurate diagnostic in a shorter period of time.

A structured computational environment that allows free expression will help children revealing specific aspects of their learning process, offering a valuable material for psychopedagogues. For us, a structure computational environment is a space ready to create specific situations where observations about children can be made. Both the topic and the logic structure follow a composition that lead children to exposing – via actions – how they articulate the rational, emotional and relational dimensions involved in their own learning process under specific circumstances.

Evaluating the softwares and the need of technological resources for pedagogic clinic, the software *Constructing a space* was developed with the goal of offering a resource for clinic diagnosis process of 7 to 11- years old children.

The objective was creating a propitious environment to evaluate the obstacles found in the children's learning process, focusing on both logic and projective activities which allow the construction of more consistent hypothesis that may be confirmed by applying different diagnostic resources, in a shorter period of time.

Project to be developed

When developing a software for children, it is necessary to take into account two aspects: the psychopedagogic requirements and the available tools that may be used to engage the child in a specific task. The more intuitive the interface (i.e, designed to allow the children to concentrate on the efforts to accomplish the tasks), the more friendlier the software will be (Sampaio, 1996:72).

From the observation and analysis of software in use in the psychopedagogic diagnosis process, we present the functionalities that support the development of the software *Constructing a space*.

- Its tools allow the choice of a variety of scenarios for the computer screen, objects, characters and their appearance, size, literalities, besides the possibility to delete, replace and drag them.
- A graphic bank of scenarios, objects and characters that allow a higher number of possible combinations among themselves to be applied on the screen.
- A specific space for written expression relating it to images and sounds.
- Flexibility in the navigation; i.e, getting in and out at any time.
- Possibility of registering the files and the process of thinking.
- Multimedia possibilities of inserting sounds and images through the use of files already recorded or recording their owns.
- Iconic language to facilitate the interaction for users (children from 8-11)
- Easy interaction and intuitive use for adults and children.
- Pleasant and attractive for the age group (children from 8-11)

During the psychopedagogic diagnostic process, the functionalities listed above seem to be essential to a software. It is important to work with a software open enough to allow users to make their own choices, revealing the strategies, decisions taken, correct decisions and answers and mistakes made. So, it is possible to observe the mechanisms employed by users in their process of learning.

The software *Constructing a space* was developed based on the criteria previously listed. It was also necessary to add a tool for registering and observing children's process of learning. This tool has the function of recording, loading and executing files in which the children's actions and thoughts are recorded during the diagnostic session. It makes possible for both therapist and child to verify whenever needed, the representational system of the user of the software. A software equipped with all these features becomes an environment where children should act and make choices, projecting their desires and wishes; therefore, it offers psychopedagogues an unique material that will enable them giving a more accurate diagnostic.

In a psychopedagogic diagnostic process of children (7-11 years old), it is important to encourage children to project themselves and, as consequence, enable them to deal with their daily routines. For this reason, the topic adopted in *Constructing a space* is home and its semantic area.

The target group can be categorized in two levels: primary and secondary. The primary public is the psychopedagogue who will use the software for collecting data and analyse the data gathered after each session. The secondary target group is children from 7 to 11 years old who will be using the software. The delimitation of the secondary group was influenced by the fact that this age group is the one who most seek psychopegagogic evaluation due to different sorts of school problems.

Graphic Project

The graphic project has an essential part in this software as the interface user-system is based on iconic knowledge. We also opted for colored icons which are organized in toolbars in the place of inscribed menu. The aim is make the use simple and intuitive. When there is any sort of written material, there are the options of hearing or visualizing it. The main idea is to focus on the task, avoiding that the child gets distracted by commands or any other kind of artificial operations which are exterior to the environment itself (Sampaio, 1996:73).

The gathered data also has an important part in this software, because all the activities proposed need to be done based on the data gathered and stored. It is from the combination and application of images in the computational environment that the logical, emotional and pedagogical relations can be established. Launches of projective mechanisms are dependent of the graphic universe – what it offers and the relationships that may be established from it. For this reason, we tried to create a bank of images which are part of children's universe, helping them to identify themselves, their families and friends. The graphics are logically grouped by their use in directories, making the identification and manipulation easier.

As the graphic data, the size and proportion of the images also matter because they are used to build up the graphic environment. All the images should be in an approximated scale but they should differ in proportion from one another. For each non-symmetric image, it was generated a correspondent mirror image.

The position of the images are in the same perspective as the background. Children are encouraged to look for a proportion relationship among the objects while placing them inside the rooms. They should also choose the appropriate side of the room and the correct perspective according to their own organizational plans.

The relationship among the distinct spatial representations is another point to be considered. The visualization of a ground plan of a house – for selecting a room –and the furniture – in perspective – help children to establish a series of complex spatial relations that can be

observed and analyzed during the diagnostic process. Due to the environment's presentation (in perspective), the images had to be adapted to different perspectives within a set limit ; i.e, not all the pictures can be placed on all the walls.

The software *Constructing a space* was conceived and developed in the platform IMAGE – a new Logo language program object-oriented. We selected this platform because it is the programming language mastered by the author of this article. Furthermore, this platform brings functionalities that may be inherited. It corresponds to the defined criteria for developing the software *Constructing a space*, as well.

Platform of development: IMAGINE LOGO

"Imagine" is a new generation of Logo, with object-oriented structure merged into traditional Logo philosophy, with empowered animation, open hierarchy of graphic screens and panes, hierarchy of objects and behaviors, parallel independent processes, direct painting tools, extended direct manipulation interface, tools for publishing for the Web, rich Logo language and other characteristics.

Imagine Logo was developed by a group of professors of the University Comenius, Bratislava, Slovak Republic, translated for the Brazilian Portuguese version by Cnotinfor Brasil.

"When developing this educational environment we had in mind a learner who wants to access a very broad palette of activities, from painting and animating to Web authoring, doing traditional Logo, creating multimedia, using speech input and output, modelling, constructing domain-specific learning frameworks, communicating ideas, building presentations, developing projects and micro worlds for numeracy, literacy or science, working with data...Our goal has been to provide students, teachers and developers with a challenging general tool for learning." KALAS, I; BLAHO, A.; Imagine... a new generation of Logo: Programmable pictures; Comenius University; Bratislava, Slovak Republic; Proceedings of the IFIP WCC 2000

Turtle Geometry

The Main Window of Imagine Logo can enclose several Pages which are shown one by one. In each Page it is possible to find a amount of small graphic objects named Turtles². In addition, each object Page may contain several Panel objects with a number of Turtle objects inserted in them.

In *Constructing a space*, when we develop the proceedings fro creating the recording function and the function for presenting all the moves done by children while using the program, turtle geometry is applied.

Object-oriented, Class and Object

One of the main characteristics of all object-oriented system is the creation of classes which specify similar objects.

Imagine Logo environment consists of individual objects and instructions, that may be sent and performed by the objects, and classes. Each individual object – an Instance – is part of a class and each class specifies the behavior of similar objects; i.e., their own status and procedures. These procedures make possible that the objects react to instructions.

In *Constructing a space*, the classes used from Imagine Logo are: Button, Panel, Page, Text Box, Toolbar and Turtles. From the Button class, we created the Four Class. From the Turtle class, the Pict class.

² The first version of the LOGO language appeared in 1968, as an enclosed part of a list processing. Then, the freestanding Turtle was designed: obeying commands such as walk and turn, a small robot walks on a piece of paper, leaving a pen-trace-like mark on the surface. For moving so slow, it was called Turtle. This icon is a representation of a mechanic turtle and it was idealized to bypass precision problems in the drawings made by children.

Instance

In *Constructing a space*, it is possible to create Instances of basic classes of Imagine Logo and classes designed from super-classes, as previously shown in Figure 2.

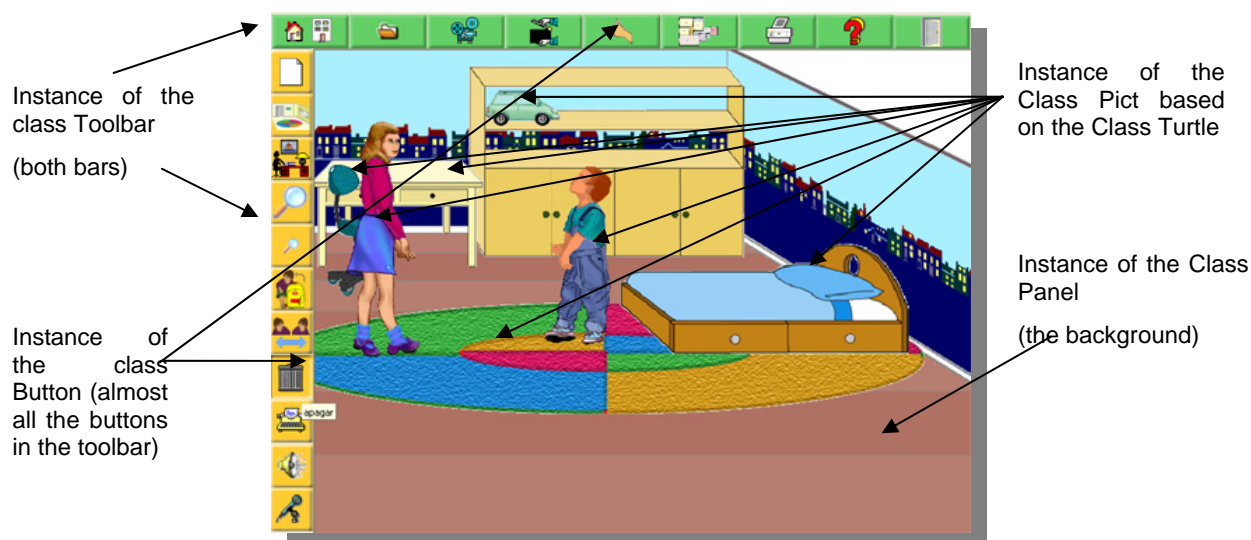


Figure 2

Characteristics and Functions of Constructing a space

Constructing a space is a software for building and simulating the home atmosphere – suitable for being applied to diagnostic process of children from 7 to 11 years old.

The central Idea is to give enough freedom to children so that they externalize their actions – from a minimum body of rules. Through this process, we allow psychopedagogues to build up inferences about the children's model of learning.

Brief description

At the beginning, the child is invited to simulate someone who is looking for a place to live. This character should choose a residence (a house or an apartment) for providing it with furniture and humanizing it. As soon as the choice is made, we give children four options of ground-plans: from 1 to 4 bedrooms. It is from the selection of the ground-plan that the children will choose which room they want to furnish first. They select the background – walls and floor, first –and, then, the objects, people, plants and animals that will be part of the environment. It is possible to furnish and humanize only one room or many rooms at a time. It is a free option in the software; i.e, it is not linked to any other predetermined task.

For this first version, it was not possible, due to lack of time, to create all the graphic structures for all the rooms of a house. So, we opted for enabling the bedrooms, since it is here that we find higher chances of a child develops projecting mechanisms. It was not possible to conclude the system for saving and opening LOG files.

Desktop

The desktop presents a central area where the activities happen and it can be visualized in two different toolbars: a) the main toolbar (at the top of the screen) and b) the side toolbar. All the functions presented in *Constructing a space* are activated via these two toolbars.

Toolbar: Main and Side

The *Main Bar* is enabled during the execution of the program, being hidden only when it executes the Help function. It comprises nine buttons which functions can be classified as selection, executable and filing.

The group of buttons as *selection tools* are those which grant access to selections of the ground-plans in the program, as shown in Figure 3, under numbers 1, 5 and 6.

The group named *executable tools* are responsible for carrying out the same defined action in all the modules of the program, as shown in Figure 3, under numbers 7,8 and 9.

The three-button group named filing tools – save, open and execute – are shown in Figure 3, under numbers 2,3 and 4.



Figure 3 – Main Bar

To set in motion the function of each button, just single-click on the button. If you single-click again, it is going back to its original status.

We will present all the tools presented in the Main Bar, their correspondent interface and the functions performed by them.

The tool *House* or *Apartment* may be set in motion, at any time, allowing the user to change the option previously made.

The tool *Open* opens the file with the actions recorded through the tool button open file. In accordance with the file chosen in the dialog box, the mode should be recognized (ground-plan or room interior) in Open function, letting it ready for execution.

The *Recorder* tool records in the file “executed actions” through a dialog box. In accordance with the file recorded, the mode for its execution should be recognized.

The “*Finalization*” tool indicates that a child confirms the end of the organization and humanization process in an environment and that they may move to another room. When children execute this function, the software will go back to the ground-plan previously selected by the child. Then, all the activities done in the environment will be recorded.

Ground-plan tool opens a window with four panels to be selected by the children with the number of rooms desirable. The children should click on the ground plan selected to set it in motion.

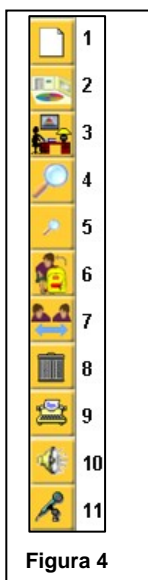


Figura 4

At the beginning of the activities with *Constructing a space*, when children choose between house or apartment, only the ground plans for the option selected is enabled.

The Print tool opens a dialog box for printing the screen area without the toolbars. At any time, children may print the ground plan selected.

The Help tool allows children to enter in the Help section. In this module, the text goes together with audio for a better understanding of the content.

The Exit tool closes the software and save the activity.

The Side bar (Figure 4) becomes activated only when the child selects a room in the chosen ground plan for furnishing and humanizing it. The buttons that comprise the side bar are: the selection, manipulation and utilities.

The selection tool (numbers 2, 3, 10 in Figure 4) has the function of selecting a background, objects, animals and characters and sounds that will be applied to the room in evidence in the desktop.

The manipulation tools (numbers 4, 5, 6, 7 in Figure 4) are used for re-arrange, resize, change the laterality and remove objects and/or characters. This group of tools has the

same operation mode; i.e, when the user single-click on them, they get activated. When we single-click on them again, they get inactive.

The utility tools (numbers 9, 11 in Figure 4) execute specific complementary functions to the program.

The Clean tool (numbers 1, 8 in Figure 4) deletes all the background, objects and/or characters.

The background opens a dialog box for the selection of the background for the room chosen. The environment may be visualized, one by one, when the user single-clicks on the name of the file. Selecting the background to an environment, this will appear on the center of the screen.

The room selected in ground plan will be present in perspective cut.

The tool Objects and Characters selects the objects and characters to be inserted in the room. When this tool is activated, a dialog box pops up presenting a list of directories organized into distinct categories: animals, people, toys and fun, adornment, electronics, and so on. Opening the directory of each category allows the user to visualize the files for selection.

When selecting the objects/characters in the dialog box, the indicator changes to points out the function of locating an object. After being applied to the screen, the object/character may be dragged within the limits of the central screen.

The Resize tool has the function of re-dimensioning objects/ characters inserted in the background area.

The Ordering tool allows the organization of objects/ characters on the desktop. The inserted objects and characters in the desktop go to the foreground in a click.

The Laterality tool has the function to alter the position of objects and characters inserted in the desktop.

The Delete tool removes objects and characters inserted in the desktop.

The Write tool enables a module for writing, opening a window with a dialog box in the desktop. Through this text box it is possible to write a text (without length limit), save it and open a text file in txt format.

The Sound tool has the function of selecting a file – wav, mp3 or midi – to be executed. The chosen file can be heard before it is set in motion.

The Recorder tool executes the recorder of Microsoft Windows, allowing users to record sounds in the wave format. These sounds may be executed via the same tool.

Evaluation

Up to which point, does the software *Constructing a space* correspond to the evaluation needs of the psychopedagogue?

Once the environment was designed and built, it was evaluated by a group of psychopedagogues. The evaluation allowed us to find ways to optimize the software and to consider some suggestions for future application and implementation.

Does the psychopedagogue recognise, in the environment, a tool for diagnostic evaluation?

During the first interview, in the software evaluation, we collected data about what makes a software appropriate for being applied to the diagnostic process. In the next stages of observation and post-observation interview, we matched these qualifications to the observations and comments of the participants while using the software.

From the analysis of the data that emerged from the answers to question posed above, we concluded that psychopedagogues acknowledge *Constructing a space* as a tool for observing

projective and cognitive aspects of children from 7-11 years old. In addition, psychopedagogues considered the software attractive and easily navigable.

Does the software help the psychopedagogue to evaluate children's structures of thought (rational dimension)?

It is possible to conclude that the resources, related to cognitive aspects, presented in *Constructing a space* are noticed by psychopedagogues; however, it is not so clear as they perceive the resources related to the emotional aspects. The therapists highlighted that, from the use of such tools, children may interact with their own thoughts allowing therapists to observe their models of learning.

The rational dimension of the learning process is absorbed by the subjects as a second function of the software, which is integrated to the evaluation of children's desiderative dimension.

Some subjects presented certain doubts related to the correspondence between the level of the thinking frame and the tasks involving logic and infralogic operations, and topologic, projective and "Euclidian" relationships.

These understandings led us to formulate two hypotheses: a) the tools available in the software are not clearly presented functions for this kind of evaluation, demanding from therapists more observation and better understanding of the software and; b) psychopedagogues have little knowledge about the way children construct their space and the logic and infralogic structures that are part of it. According to Piaget (1993), we believe that the age group delimitation (7-11) is adequate. However, we cannot take conclusions about this topic because it would be necessary to carry out a deeper study of the subjects while interacting with the software.

The way buttons work would facilitate the diagnosis of children's reasoning modality, making it easier - or not - from the correspondence between the buttons and the cognitive functioning.

Does the software facilitate the evaluation process of emotional aspects (desiderative dimension) connected to children's learning process?

The evaluation of the desiderative dimension, from the use of *Constructing a space* in a psychopedagogic diagnostic, is clearly detected for all the subjects.

We may conclude through the accounts of the evaluations done that *Constructing a space* is a software which helps in the process of emotional aspects involved in the learning process. In addition, the software is classified by the subjects as a projective software, being able of replacing formal tests that have the same purpose.

Does the software facilitate the evaluation of the articulation among all the dimensions involved in the process of learning?

Analyzing the interviews (post-observation phase), the data signal that the subjects, from the evaluation group, perceive the application of the software as a resource for noticing the process as complex; i.e., in all its dimensions.

We highlight the use of the filing tools as one of the most important resource presented in this software. The subjects emphasized that from the use of these tools children may interact with their own thoughts and the therapist may observe the learning style applied. This process can be recorded by the software, what makes it unique.

Evaluation of Tools and Suggestions for future modifications

Taking into account children's observations, we conclude that it would be necessary to implement some modifications in the design, interface and navigation of the software, such as:

- Delete, from the side toolbar, the *manipulation tool*. The idea is to make the actions performed by it in a direct mechanism of manipulation of the image on screen, becoming more flexible in relation to its on angulations and projection. The functions incorporated to

manipulation tools would be accessed by single-clicking on the mouse right button, opening a menu for selection of functions.

- Alter the images of the tool group Filing in a way to make them more intuitive.
- Make the colors used to discriminate the rooms in the ground-plans more vivid, the ground-plans clearer and the proportions more adequate.
- Create identification mechanisms for navigation in the rooms using the ground-plans so that children may visualize their own routes in the environment. This mechanism may present the objects and people from a top-bottom perspective.
- Modify the dialog Box to choose objects and characters, making it becomes iconic, categorizing the files using icons which go together with written material.
- Create mechanisms to Record the last actions automatically executed, avoiding missing files that contain the performed actions by children.

These modifications will make the software more efficient because the problems found would be repaired. Therefore, the tools become optimized and children's evaluation, more accurate.

The software *Constructing a space* was submitted to a first evaluation with psychopedagogues, presenting results that allowed us to conclude that, after the adjustments mentioned previously in this chapter, its application would be valid because its goals may be reached when used with children.

Conclusions and Future works

Constructing a space was developed aiming at the observation of complex and dynamic processes and their registers. It is a computational resource which intends to clearly present, to the psychopegagogue, the way in which the rational, emotional and social dimensions interact among themselves while children perform a complex and dynamic activity.

From the software's assessment by psychopedagogues, we could confirm some of the hypotheses about the use of the software, for instance, the software is appropriate to what is proposed. However, we could also see that some modifications in the interface are necessary and improvements should be done in relation to the graphic data bank.

Due to lack of time, we could not test the software with children. Though, its application with children is essential for validating this computational resource. Therefore, we suggest, as future works, a deeper study with psychopedagogues and children from 8-11 years old while interacting with the software, exploring it in the course of a psychopedagogic diagnostic process.

References

- BOSSA, N. A. A psicopedagogia no Brasil. Porto Alegre: Artmed, 2000.
- KALAS, I; BLAHO, A.; Imagine... a new generation of Logo: Programmable pictures; Comenius University; Bratislava, Slovak Republic; Proceedings of the IFIP WCC 2000
- OLIVEIRA, V. B. (org.). Informática em Psicopedagogia. São Paulo: Senac, 1996.
- PIAGET, J.; Inhelder, B. A representação do mundo na criança. Porto Alegre: Artes Médicas, 1993
- SAMPAIO, F. F. Linkit, Design, development and testing of a semi-qualitative computer modeling tool. Tese doutorado. Londres: Universidade de Londres/ Departamento de Ciência e Tecnologia do Instituto de Educação, 1996.
- WEISS, M. L. L Psicopedagogia clínica: uma visão diagnóstica. Porto Alegre: Artes Médicas, 1992.
- WEISS, M. L. L Psicopedagogia clínica: uma visão diagnóstica. Rio de Janeiro: DPA, 2000.

Changing a half-baked 3d navigational game

Efi Alexopoulou, efialex@ppp.uoa.gr
Educational Technology Lab, University of Athens

Chronis Kynigos, kynigos@ppp.uoa.gr
Educational Technology Lab, University of Athens and CTI

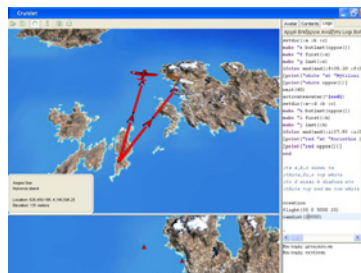
Christos Markopoulos, cmarkopl@upatras.gr
Educational Technology Lab, University of Athens

Abstract

A number of research studies focus on the ways digital games can provide a rich context for students to experiment, explore and engage in meaningful formalism. These researches suggest that games can help students develop valuable skills in the transitional stage between intuitions (informal) and formal mathematics. This can be done either by playing, designing games, or reflecting on game's rules (Kafai, 1996). In this paper we endeavour to combine these powerful approaches on learning with games, using half-baked games, an idea based on that of half-baked microworlds (Kynigos, 2001). These are games that incorporate an interesting game idea, but they are incomplete *by design*. Students while playing the game reflect on rules and change them in order to create a new game of their own choice. Thus they engage both in the process of instrumentation while using the game and in the process of instrumentalisation while changing the game resulting in a variety of artefacts (Guin and Trouche, 1999).

In present research study we used a half-baked game built on the Cruislet environment, a 3d digital medium that incorporates both GIS and Logo programming. It is designed for mathematically driven navigations in virtual 3d geographical spaces and thereby enables the user to explore navigation in 3d geographical space, spatial visualization and mathematical concepts embedded in a realistic context. Navigation with Cruislet is feasible by defining objects' (airplanes) displacements, either by using an iconic interface or using Logo. The Logo editor provides opportunities to run and edit programs and thus create multiple behaviours of objects, or game rules.

In this small scale research, two 17th year old students worked collaboratively and engaged in processes of playing the game, reflecting on its rules, de-coding the rule and intervening into the Logo code in order to change the game and create something of their own choice. Findings indicate that the half-baked game and the representations embedded in the Cruislet environment provided a meaningful context in which students engage in processes of mathematisation of geographical space.



Keywords

3d environment, half-baked games, navigation in geographical space, Logo

Introduction

Lately there is a growing interest about the ways in which game-based learning environments facilitate new ways of learning (Gee, 2003). The key feature of this approach is that playing games can provide a context for the development of valuable skills (Kirriemuir and McFarlane et al., 2004) in the transitional stage between intuitions (informal) and formal mathematics. Using games with an appropriate set of tasks and pedagogy, students can be engaged in exploration, problem solving, rule-based thinking and other forms of mathematical thinking (Goldstein et al, 2001). Although game-play could provide a rich context for students to experiment, explore and engage in mathematical expression, usually students adopt the role of passive consumer of the game, in contrast to the active role of learning when students construct their knowledge. Through the process of designing games and reflecting on game rules, students adopt an active role and engage in meaningful expressions (Kafai, 1996).

In order to combine these two approaches, we use the idea of half – baked games, an idea taken from microworld design (Kynigos, 2001). These are games that incorporate an interesting game idea, but they are incomplete *by design* in order to encourage students to change their rules. Students play *and* change them and thus adopt both roles of player and designer of the game (Kafai, 2006). In order to address our design rationale and our study of half-baked game use by students, we found it useful to employ an idea coming from cognitive ergonomics which distinguishes the object game from the mental construct created by each student through the process of using it. In articulating this idea, Rabardel (1995) uses the terms ‘artefact’ to denote the tangible game and ‘instrument’ to denote this construct which he called ‘schemes of use of the artefact’. Later, Guin and Trouche (1999) went further in using the term ‘instrumentalisation’ to denote the process by which students transform the artefact for specific uses. From this point of view, the process of designing game change and play activities can be seen as the design of the terrain within which the instrumentation/ instrumentalisation processes may take place by students’ interactions with the game and domain specific rules and concepts embedded in it. To support the construction of such games, we use an environment where rules are built within Logo procedures and thereby students are able to intervene into the structure of games’ rules and change them. Thus, Logo provides opportunities to express ideas in meaningful ways and in this way it can be seen as a medium in the transitional stage between intuitions and meaningful formalism (Dubinsky, 2000).

In this particular study, children’s engagement with a number of tasks focuses on the exploration of the concept of navigation in 3d geographical space, a concept which we consider as complicated; as it is strongly interrelated with several mathematical concepts and it is also depended on the situations that the concept appears (Vergnaud, 1991). Espousing Vergnaud’s view that a mathematical concept is not isolated from other concepts, we consider navigation as a *conceptual field* strongly related to concepts such as geographical coordinates, spatial visualization, covariation and the interplay between them. Using this approach to navigation, we endeavour to study the mathematical concepts that are integrated with geo-spatial representations and information, providing opportunities for processes of mathematisation of geographical space. The mathematical concepts involved in the particular tasks are those underlying the use of issues regarding 3d space, including linear functions, as the displacement of objects (in this case, airplanes) could be conceived as the co-variation or ‘rate of change’ of geographical coordinates. As Tall (1996) suggests enactive sensations of moving objects may give a sense that “continuous” change implies the existence of a “rate of change”.

The Cruislet environment

The 'Cruislet' environment is a digital medium based on GIS (Geographic Information Systems) technology that incorporates a Logo programming language. It is designed¹ for mathematically driven navigations in virtual 3d geographical spaces and is comprised of two interdependent representational systems for defining a displacement in 3d space, a polar coordinate and a geographical coordinate system. In the present study we'll focus on the geographical coordinate system and the ways this is used to navigate objects (airplanes) in 3d space.

The environment enables the user to explore spatial visualization and mathematical concepts by controlling and measuring the behaviours of objects. The objects are airplanes and their displacement is represented by a vector. The user is able to navigate airplanes either by using the Avatar tab or through the use of a Logo editor Tab (see Figure 1). In using the Avatar Tab, the user can create airplanes and define their position in either of the following ways: a) by determining the latitude, the longitude and the height in which the airplane will be placed or b) by determining the vector of displacement, i.e. the angles r and f of two vertical planes and the length r of the displacement, i.e. the length of the vector. When a parameter from one of the ways of displacement is changed the parameters in the other are dynamically changed as well.. In the Logo Tab the user can actually edit and run Logo programs and thus create multiple or relative displacement rules in 3d space. The Logo programmability is considered necessary as it provides users with the option to actually anticipate the result of their action and engage in expression of mathematical ideas through meaningful formalism by means of programming. In this sense, Cruislet can be conceived as a constructionist medium (Kafai & Resnick, 1996) in that the user can construct flights and build dependency between flights.

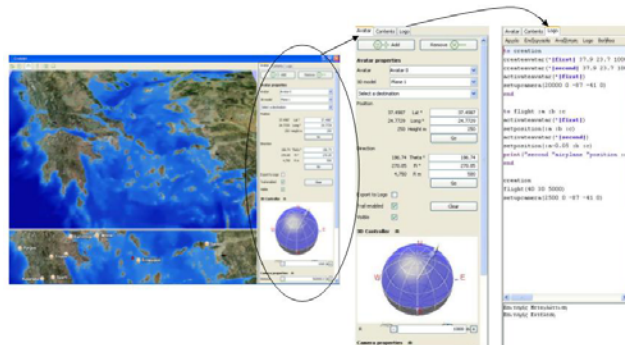


Figure 1. Cruislet environment – Avatar Tab – Logo Tab

Our initial aim when designing Cruislet was to create a realistic context such as a 3d geographical map where geo-coded data will be integrated with mathematic representations, in which students would be able to explore, experiment and construct virtual reality microworlds. Students' constructions in these microworlds are airplanes' trips as well as rules of airplanes' displacements that are defined in Logo procedures. In that sense, navigations in virtual 3d geographical spaces within Cruislet could be conceived as game play simulations, as users manipulate objects in a 3d game-like environment according to rules built on this which are defined by the designer of each microworld.

Task design

Students carried out three tasks, each one of which aimed to capture different aspects of interaction with the environment. We use the term interaction to define the ways students utilize

¹ 'Remath' – Representing Mathematics with Digital Media FP6, IST-4, STREP 026751 (2005 – 2008), <http://remath.cti.gr>

the environment either by using the representations provided, playing the game built on Cruislet or changing the game. In each of these tasks, the artefact becomes an instrument (Verillon and Rabardel, 1995) through the process of instrumentation (Guin & Trouche, 1999) as students work and play with Cruislet. According to these, in the first task which was considered introductory to the environment, we asked students to create an airplane and make a flight to Athens or any other city of their choice, by defining the geographical coordinates of each city. The task had two phases, where students had to do this using the avatar tab or edit Logo commands to carry out the same flight. This activity aimed at introducing students to Cruislet and the provided representations as well as editing basic Logo commands

Our approach in designing the second and third tasks promoted investigation through the design of activities that offer a research framework to investigate purposeful ways that allow children to appreciate the utility of mathematical ideas (Ainley et al., 2006). In the second task we created game rules built in Logo procedures, in order to engage students with Logo programming and simultaneously to embed mathematical concepts in game rules. The task was based on the idea of the "Guess my function" game, in order to provoke children to discuss, compare and experiment with dependence relations such as linear functions. In particular, students were asked to study the existence of a rate of change between the displacements of two airplanes which are defined in the geographical coordinate system.

The second airplane's position depends in a certain way on the first airplane's position, according to the 'distance rule', as the second airplane (spy or escort) follows the first one. Students must find the dependence relation by moving the first airplane and seeing what happens to the other one in order to decode the rule. The object of the game is to place the first airplane in a position where the spy will be placed at Thessaloniki (city on the north of Greece). A Logo procedure (named flight) determines the function, which is hidden to students and they have to guess it based on repeated displacements of the first airplane and observations of the relative positions and displacements of the first and the second airplane (spy). Thus defining the first's airplane's position using this procedure, the environment gives them feedback about second airplane's position. The procedure that defines the rule is:

```
to flight :a :b :c
  activateavatar("first")
  setposition(:a :b :c) → define first plane's position (Lat Long Height)
  activateavatar("second")
  setposition(:a-0.05 :b+0.05 :c-1000)
  print("second "airplane :a-0.05 :b+0.05 :c-1000) → print second plane's position
end
```

For example, if students write:

flight(37 28 10000) → define first plane's position (Lat Long Height) and the displacement of the first airplane is executed as well as the displacement of the second airplane and they will get the answer:

Second airplane (36.95 28.95 9000)

With this task, students get involved with the concept of function as a dynamic process of covariation using the geographical coordinates as a system of reference. In particular, students are actually asked to study the existence of a rate of change of relative displacements of points on the space as moving the first airplane between successive positions has as a result the displacement of the second plane between corresponding successive values (Confrey and Smith, 1995).

In third task we wanted to provoke students' interest, in a way that this could give them a motive to engage in the task. Based on Ainley, Pratt and Hansen work (2006, p. 35) we took into account three critical aspects involved in the tasks, in order to stress purpose and utility: i) we wanted an explicit end product created by students, ii) we told students to create something for another audience and iii) we created tasks in order to give opportunities to students to make meaningful decisions regarding mathematical concepts involved in the tasks. Thus, in the third task students have to formulate another rule and change the functional relations of plane

coordinates by intervening in the Logo code and by creating through this process another game to challenge another team. The process of finding another rule, alternate the game and challenge others could engage students with mathematical and geographical concepts as well as with Logo while experimenting with the environment. While students reflect on rules, engage in the process of instrumentalisation (Guin & Trouche, 1999), since displacement rules can be questioned and re-defined by them resulting in a variety of artefacts, which are actually the different games they create. We should mention here, that although the game is relying on mathematics and is very different to digital games students are used to play with, in this small scale study we endeavoured to create a rule within the activity, that could be considered by students as a game. Additionally we wanted to engage them in the activity of creating other rules and through this procedure, create a game.

Research setting and methodology

In our research we used a design-based research method (Cobb et al., 2003) which entailed the 'engineering' of tools and task, as well as the systematic study of the forms of learning that took place within the specific context as defined by the means of supporting it. Our approach is also based in iterative design (diSessa, 1989) and initially our aim was to carry out a small scale study and observe students using and modifying the environment as this would give as feedback into the next iteration of Cruislet development and task design.

The research reported herein took place in the computer lab of a public vocational school in Athens. Two 17th year old students, used the Cruislet environment and worked collaboratively in a set of activities which lasted 6 hours in total over 2 days. Two researchers participated in the experiment, espousing the role of naïve and participant observer allowing for the data to structure the results and to pilot their analysis. For data collection we used a camera and a tape-recorder in order to capture students' dialogues and their interaction with the environment. Background data were collected, such as students' notes on papers and one researcher was collecting observational notes as they occurred. All audio-recordings were analyzed verbatim. In analyzing the data we focused on two different processes in relation to students' actions, through which students interacted with game's rules: a) the process of instrumentation, and b) the process of instrumentalisation of the artefact. In the first case we looked for instances where meanings related to navigation and geospatial representations were expressed by students while playing the game. In the second case we tried to identify critical episodes during their experimentation, where students by reflecting on the rules constructed meanings of the relative displacements of the airplanes and actually changed the game constructing their own rules.

Playing the game

While students were playing the game several meanings emerged from their interaction with Cruislet environment regarding the inter-dependent relationship between geo-spatial representations and mathematics. Here, we focus on meanings where mathematical issues are integrated with geo-spatial representations and information, providing opportunities to students for processes of mathematisation of geographical space.

The students participated in our experiment were using their intuitions to anticipate the relation between airplanes and thus to find the linear function. In order to decode the rule, they experimented by giving various values to coordinates (Lat, Long, Height) that define the position of the first plane, they communicated their observations about the position of the second plane and they formed conjectures about the relationship between planes' positions. Initially they have started thinking of this relationship, visualizing the result of airplane's position (figure 2).

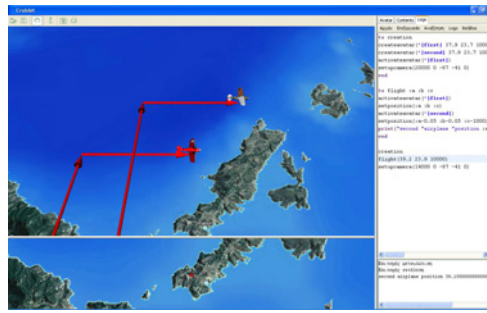


Figure 2. Airplane's displacement in Cruislet environment.

Although the feedback from the environment was mathematically represented, that is the 3 coordinates of each airplane's position, they preferred to think of it as a relationship between two objects in 3d space rather than a function between coordinates. Thus, when they were experimenting giving several coordinates in the first airplane and observing the second airplane's position, they translated addition and subtraction in functions as front, behind, left and right depending on airplanes' positions. Using direction to specify airplanes' relationship was easier for them, rather than using mathematical symbols, like plus or minus. In the following instances of an episode, the researcher asked them to explain how they correlate direction with airplanes position and coordinates.

- R1 Can we understand if it's back, front, right or left, looking at these? (refers to airplanes' positions)
- S1 If someone knows where the coordinates are, that is how these are defined from our point of view. How these are defined? I mean if it goes right with plus or minus or if it goes left with plus or minus.
- R1 So, how do you know if it's right or left? How do you think plus and minus are related to front or back, left or right?
- S1 Plus, if we see lat (refers to latitude), minus is behind and plus is in front.
- R1 Yes...
- S1 According to our position. In long (refers to longitude), plus is right and minus is left.
- ...
- S1 It is 5 meters in the right and behind us. It's 0.05 in latitude, 0.05 in longitude when we use integers. And it's 1000 feet underneath us.

The interesting point here is that students preferred to use an experiential egocentric system of reference in order to find the relationship rather than using mathematical formalism, although they were accustomed in using mathematical symbols and working with functions. The way they specified a position in 3d space, that is their system of reference, is related to the existence of a stationary point (first airplane) and a sense of direction (right, left, etc). However, because they were flying towards north of Greece, "behind" for them meant minus in the function, although this isn't the case. We think that this indicates the strong sense of the ego-centric system of reference that students have. The episode also indicates that students have realized that the system of geographical coordinates is an absolute fixed one, as they say that in order to correlate direction with signs "someone has to know where the coordinates are", implying that there is a convention in the geographical system of reference.

The Cruislet environment provided opportunities to students to associate issues regarding both navigation in geographical space and mathematical concepts underlying 3d space. Within this scope they explored concepts regarding 3d space and especially the third dimension, height. In the following episode students are trying to fly upon Greece and land in a city, but there are constraints that don't allow the flight. Students managed to overcome the problem that occurred,

following their intuitions and using the representational issues (the 3d map of Greece) provided by the environment.

- S2 The information we gave, with these coordinates, the airplane can't go there. Do we have to give...
- R Is there another reason that the airplane can't fly?
- S1 No, that's the reason why. Because we had it at low height, we had it at low meters. The airplane was at very low height.
- R And?
- S2 It couldn't. The program warned us that there are mountains in front of us, so the airplane couldn't fly. It has to do with the airplane's course. This is nice, ha? In case you are a pilot.
- R And what did you do?
- S1 We just raised meters. We had it in 1000 meters and we raised it at 5000 meters, which was the initial height.

Despite the fact that the environment didn't give a hint about the problem occurred, students found at once that the constraints had to do with height. We think that this is important as geographical information provide a context to explore representational issues regarding 3d space. Although in this case the third coordinate was manipulated effectively, students many times forgot to edit height or they thought of it as unnecessary when they were using the Logo editor Tab and had to specify a specific place in space. The following episode indicates that students got confused with the third coordinate.

- S1 And the other one is 22.9367. Close the bracket. (after editing lat and long)
- S2 What about height?
- S1 Height is fixed at 12000. Change it?
- S2 I don't know. No. Let it. Or we should edit height? (he is asking the researcher)
- R Do you want to change height?
- S2 No. Do we have to edit it or it remains fixed?

Although students knew that in order to specify a position in space they had to edit three coordinates, in this case they didn't want to change height, as they considered it as unnecessary. We could say that may be this is because they were not familiar with the environment and thought that the environment 'remembers' previous positions or coordinates. But this is not the case as such confusion occurred only with height and not with other coordinates, even if one of them remained stable. A possible interpretation about this confusion is that students are accustomed to 2d representations where they manipulate only two magnitudes and this is the reason why they usually preferred to fly at a fixed height. On the other hand if we accept the view of Dalgarno et al. (2002) that we understand 3D models through multiple 2D representations, maybe students had focused subconsciously on a simplified 2D way of visualising the displacements of the airplanes. We consider our findings compatible with another research conducted by Kynigos and Latsi (2005) in relation to 3D representation of vectors that have also shown that children have focused in a simplified 2D representation.

Changing the game

Our initial aim was to ask students to reflect on the game and change it if there was something they didn't like, in order to create another game for the next couple of students. During the experiment we experienced a pleasant surprise when students suggested that they would like to change something on the game and see what happens. With this pretext we encouraged them to reflect on the rules and thus make their own game and challenge other students. In this paper we focus on students' modifications of the game either at the imaginative level where students

reflect on rules and use their imagination to create new ones, or at the creative level, where students edit their rules using Logo and through this process create a new game. The two levels of game design that we categorized our findings are similar to Goldstein et al (2001) distinction between 'Game Inside' and 'Game Formal' respectively, as in the first case we refer to students' ideas while in the second to rules that students actually created. These two games differ as 'the expression and application of rules are mediated by the tools available' (Hoyles, et al, 2001) and as students were not familiar with Logo, the implementation of their ideas was limited.

Reflecting on rules

We refer to students' reflection on the rules as it is considered indicative of their way of thinking. Here we report students' ideas while they were thinking of alternative rules in order to change the game.

- Easy function

While playing the game students considered it as an easy one and the rules built upon airplanes' displacements as easy to decode, so they reflected on the rules in order to make a more realistic and difficult game for others to play. The first thing that considered as really easy was function's numbers.

S2 We would use more difficult numbers for the other to find. Instead of 5 we would use a decimal number. That is 56, the difference not to be easy.

R1 You mean the variation to be a difficult number?

S2 Yes.

This excerpt indicates that students were thinking that the function was easy because it involved easy numbers. This is interesting as they didn't thought for another non-linear function, but a linear function with complex numbers.

- Easy rule

Before playing the game students found that the way the second airplane followed the first one was not realistic as the second one should try to hide from the first, in order to avoid get caught by the first airplane. Thus, they thought that it would be realistic if the second one was moving not according to a specific plan or he was moving according to a complicated rule.

S1 If the spy (means: the second airplane) changed his place, it would be difficult to find the rule.

R1 I don't understand what you 're saying. Do you want to explain it?

S1 He (the spy) was in a position wherever we went; he was in a fixed position behind or in front of us, wherever we went. If his position was moving it wouldn't be easy for someone to find.

R1 How the position would be moving? Do you want to say to me?

S1 For example, if it goes right, left, or near us or far away form us, according to the place we go.

- Insert boundaries

Students wanted to limit the way the first airplane was flying. Specifically they wanted the first airplane to fly only above Greece and limit its possible displacements. With this rule students were thinking about how they could limit the coordinates that would be permissible and by this they were referring to the domain of numbers that the function could get.

Create the rules

After reflecting on rules, students focused on a specific rule and tried to edit the rule and extend the predefined Logo procedure. Their idea was to divide Greece into two segments where while the first airplane would fly upon one, the spy would follow him according to a function, and while the first would fly upon the other segment, the spy would follow him according to another

function. We think that students chose to implement only this rule because they were not experienced with Logo and this constrained the implementation of their ideas and the 'translation' of them in Logo commands.

During the process of making the rule, students were involved with the process of mathematisation of the geographical space as they wanted to divide geographical coordinates in two segments. Thus they searched for a city in the middle and cities in the north and south end of Greece and found their coordinates. In the following experts students are trying to explain the rule.

S2 I'm saying that from Heraklio (south) to Athens (middle)...

R Yes, nice.

S2 The second airplane has a distance 'a' from us.

R Ok.

S2 And from Athens to Drama (north), it has a different distance from the previous one.

...

S1 We have divided Greece in two segments and we have determined the boundaries. With some coordinates. That there...that in these coordinates are our boundaries. Up to there, the second one can be that close and in the rest of Greece the second will be in a longer distance.

R In the rest of Greece. You're saying that you have divided into two segments.

S1 In two segments.

S2 In two segments. Namely, what he looks (meaning the second airplane) is the length, its a (meaning the latitude).

The episode is interesting as it reflects students' thoughts about the rule involving both geographical concepts such as finding cities that are in the north, middle and south of Greece and mathematical concepts such as coordinates and functions. Here, geographical coordinates of Greece, could be seen as the domain of the function, where the function changes according to the domain it is applied and additionally it is limited by specific boundaries.

In order to specify and implement the rule, students had to extend the initial Logo code that was a black box to them when they were playing the game. It was really surprising to us that although students were not experienced with Logo programming, they successfully managed to decode the Logo code and reflect upon its structure. A possible reason for this is that Logo commands are based upon everyday language, resulting in an easy way of exploring the operation of each of them as well as of learning to use them.

The rule that students selected to implement was comprised of two conditions and the corresponding actions. Initially they were confused about the way they should express the rule in Logo as they didn't know how to edit Logo commands. Researchers told them to express the rule in a way independently of Logo and that they would help them with Logo. Surprisingly students chose the 'if' command, as they knew it from Excel and edited the rule, thus it was easy to experiment and finally edit the Logo code. The following procedure in Logo is the final product that students created. (Their contribution in the initial Logo procedure we gave them is marked in bold.)

```
to flight :a :b :c
activateavatar("first)
setposition(:a :b :c)
if :a<38.0551
[activateavatar("second)
setposition(:a-0.05 :b+0.05 :c-1000)
print("second "airplane "position :a-0.05 :b+0.05 :c-1000)]
if :a>38.0551
```



```
[activateavatar("second")
setposition(:a-0.07 :b-0.07 :c-1500)
print("second "airplane "position :a-0.07 :b-0.07 :c-1500)]
end
```

Another interesting idea that students had, was to divide Greece in 'four quadrants' as they said, so the second airplane would have to check for the longitude of the first airplane as well. When the researchers told them to write this idea in their Logo procedure, they were puzzled as they didn't know if they had to use more 'if' in the procedure or not. Finally they agreed that they had to use 2 additional 'if' in their procedure, as the segments they wanted to divide Greece were four. A possible explanation about students' difficulties to extend the rule could be their inexperience in Logo that they had to use in order to express the conditions and actions of the rule. Nevertheless, they seemed to have developed their own rule regarding the relative displacements of the two airplanes as co-variation within the frame of a function. Moreover, they tried to make more complicated the rate of change and the domain of the function, probably aiming to construct a more difficult rule, less recognisable by their classmates.

Conclusions

Students' engagement with the game built on Cruislet environment seemed to promote their thinking about a variety of concepts related to the concept of navigation, such as geographical coordinates and mathematical concepts regarding 3d space and linear functions. However, our aim was not to study a specific mathematical or geographical concept and student's way of thinking about this. We rather tried to focus on processes where students engage in meaningful game-play that provides opportunities for meaning making.

Initially, student's tried to find out and understand the rule of relative displacements of the two airplanes. Cruislet environment provided opportunities to students to associate issues regarding both navigation in geographical space and mathematical concepts. They experimented by altering the geographical coordinates of the first airplane aiming to correlate the displacements of the two airplanes. They also explored the functionalities of the microworld and identified the rule of the game by relating it with the rate of change of the geographical coordinates of the two airplanes.

Having explored and internalized the functionalities of the microworld and consequently the rule of the game students started to reflect on the microworld and actually building their own games within the mathematical context of geographical coordinates, 3d space and function. Through the utilization of the simulation, information handling, geographical systems and Logo programming students gradually developed their own game play environment. They built functional relationships between the relative displacements of the two airplanes initially by varying the rate of change and later by altering the domain of the function.

It seems that the half-baked game built on Cruislet environment provided a meaningful context in which students explored several meanings regarding both mathematical and geospatial concepts embedded in the rules of the game and in the representation provided by the environment. The key feature of the game was that it combined an interesting game idea with a mathematical and geographical domain and inviting at the same time changes to both. These changes were feasible using Logo, which is considered as an essential feature of the environment as it allows users to intervene into the structure of the game, find and decode its rules and change them.

References

- Ainley, J., Pratt, D., and Hansen, A. (2006) *Connecting engagement and focus in pedagogical task design*. British Educational Research Journal, Vol. 32, No.1, February 2006, pp.23-38.
- Cobb P., Confrey J., DiSessa A., Lehrer R., and Schauble L. (2003). *Design Experiments in Educational Research*. Educational Researcher, Vol. 32, No.1, 9–13.

- Confrey J. and Smith, E. (1995) *Splitting, covariation, and their role in the development of exponential functions*. Journal for Research in Mathematics Education, Vol. 26, No 1, pp. 66–86.
- Dalgarno, B., Hedberg, J. and Harper B., (2002) *The Contribution of 3D Environments to Conceptual Understanding*. In Proceedings of ASCILITE 2002 Conference., Auckland, New Zealand, pp. 44–54.
- Dubinsky, E. (2000) *Meaning and formalism in Mathematics*. International Journal of Computers for Mathematical Learning 5, pp. 211–240.
- diSessa, A.A. (1989) *A Child's Science of Motion: Overview and First Results*. U. Leron & N. Krumholtz (Eds.), Proceedings of The Fourth International Conference for Logo and Mathematics Education, Jerusalem.
- Gee, J. (2003) *What Video Games Have to Teach Us About Learning and Literacy*. Palgrave Macmillan
- Goldstein, R., Kalas, I., Noss, R., and Pratt, D. (2001) *Building rules*. In Proceeding of the 4th International Conference of Cognitive Technology. Edited by Beynon, M., Nehavin, L. and Dautenhahn, K., pp. 267–281.
- Guin, D. and Trouche, L. (1999) *The complex process of converting tools into mathematical instruments, the ceas of calculators*. International Journal of Computers for Mathematical Learning, Vol.3, pp. 195–227.
- Hoyles, C., Noss, R., Adamson, R., and Lowe S. (2001) *Programming rules: what do children understand?* In Proceedings of the 25th Annual Conference of the International Group for the Psychology of Mathematics Education (PME), Edited by M. vanden Heuvel-Panhuizen, Freudenthal Institute, The Netherlands, Vol.3, pp. 169–176.
- Kafai, Y. (1995) *Games in Play: Computer Game Design As a Context for Children's Learning*, Lawrence.
- Kafai, Y. (2006) *Playing and Making Games for Learning: Instructionist and Constructionist Perspectives for Game Studies*. Games and Culture, 1, pp. 36–40.
- Kafai, Y., and Resnick, M. (Eds.) (1996). *Constructionism in Practice: Designing, Thinking and Learning in the Digital World*, Lawrence
- Kirriemuir J., and McFarlane, A. (2004) *Literature Review in Games and Learning*. NESTA FutureLab, Vol. 2004, http://www.nestafuturelab.org/research/reviews/08_01.htm
- Kynigos, C. (2001) *E-slate logo as a basis for constructing microworlds with mathematics teachers*. In Proceedings of Sixth EuroLogo Conference. Edited by G. Futschek. Budapest, Hungary. pp. 104 – 112.
- Kynigos, C. and Latsi, M. (2006) *Vectors in use in a 3d Juggling Game Simulation*. International Journal for Technology in Mathematics Education, Vol. 13, No 1.
- Rabardel, P. (1995) *Les Hommes et les Technologies. Approche cognitive des instruments contemporains*. Paris: A. Colin.
- Tall, D. (1996) *Functions and Calculus*. In A. J. Bishop, K. Clements, C. Keitel, J. Kilpatrick, & C. Laborde (Eds.), International Handbook of Mathematics Education. Dordrecht, The Netherlands: Kluwer Academic. pp. 469–501.
- Verillon, P. and Rabardel, P. (1995) *Cognition and artifacts: a contribution to the study of thought in relation to instrumented activity*. European Journal of Psychology of Education, Vol. 10, pp. 77–100.
- Vergnaud, G. (1991) *La théorie des champs conceptuels*. Recherches en didactique des mathématiques. Vol. 10(2/3), pp. 133–169.

Robotics & Constructivism in Education: the TERECoP project

Dimitris Alimisis, *alimisis@otenet.gr*

Department of Education, School of Pedagogical & Technological Education, Patras, Greece

Michele Moro, *mike@dei.unipd.it*

Department of Information Engineering, University of Padova, Padova, Italy

Javier Arlegui, Alfredo Pina, *{arlegui, pina}@unavarra.es*

Public University of Navarra, Pamplona, Spain

Stassini Frangou, *stassini.frangou@sch.gr*

Secondary Education (Science Teacher), Athens, Greece

Kyparissia Papanikolaou, *spap@di.uoa.gr*

Department of Education, School of Pedagogical & Technological Education, Athens, Greece

Abstract

This paper presents the European project “Teacher Education on Robotics-Enhanced Constructivist Pedagogical Methods” (TERECoP). A first premise of this project concerns the implementation of constructivist – constructionist methods not only in classroom, but in teacher education as well. A second premise is referred to the technology-enhanced learning as occurred in the implementation of different kinds of curriculum innovation in the classrooms. A third is related with the emerging need for a teaching as a research-based profession and for the creation of a culture in which researchers and teachers can create a shared body of knowledge.

Although the role of teacher is crucial for the successful introduction of robotics in classrooms, only few projects have been undertaken to train school teachers in using this, completely new for them, technology. TERECoP project's aim and ambition is to contribute to fill in this gap suggesting a constructivist model of teacher training in these new technologies. Learning theories, modelling, technology and languages are the main aspects we will have to deal with. The main questions we are currently trying to answer (probably in this order) are: what is “Robotics” at School? Which methodology should we use to apply “Robotics” at school and teacher education? How can we design educational activities (within students' curricula and teacher training courses) once we have answered to the two previous questions?

Our work within the TERECoP project tries to give some answers to these questions. The paper describes the starting point of this project, focusing on the context, on the aims of the project and on the different partners' countries experiences, and outlines the different stages that are going to be developed to implement the project giving a description of every one. Finally some preliminary conclusions are presented.

Keywords

Logo, Lego, Constructivism, Constructionism, Teacher Education, Robotics

Introduction

Research in science and technology education has made possible the development of learning strategies and materials that attempt to meet students' needs and address their learning difficulties, such as computer-based learning environments and microcomputer-based laboratory tools (Niederer, et al., 2003). Nowadays, increasing attention is paid to computer-based robotic activities considered to be a valuable learning tool that contributes to the enhancement of learning and the development of student thinking. Taking into consideration that students have a better understanding when they express themselves through invention and creation (Piaget, 1974), teachers need to provide students with the opportunity to design, build and program their own models. Programming as a general model-building and toolmaking learning environment has been shown to support constructionist learning across the curriculum (Papert, 1992). The LEGO robot, an outgrowth of Papert's LOGO programming language created in the 1960's, partners technology with constructionist ideas.

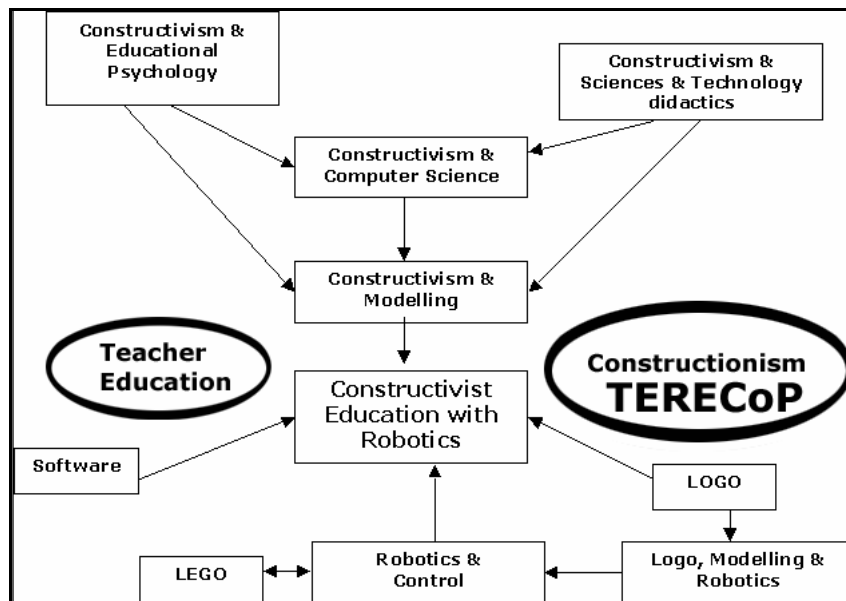


Figure 1 Related fields of constructivist robotics-based education

Logo in its various implementations was for years the main framework where applying the constructionist "way of thinking", and it still remains the natural environment from which to start to develop a constructionist approach of facing new learning challenges. Its turtle-robot works on a virtual environment, where the turtle behaviour is an iconic representation of real behaviours. So, the use of LOGO is well suited to learn by solving environmental problems (e.g. the relation between speed, time and space in a uniform movement), because, in this case, the robot has no side effects in the solving problem process and it behaves as a "virtual perfect robot". But the LOGO environment offers a very poor 'robotic architecture' and some clear limitations, like a 2D scenario, to perform complex tasks (from a classical robotics approach). The introduction of robotic elementary experiences with LOGO is a natural evolution towards a real environment with a real robot where the presence of physical constraints and new input stimuli (real sensors) offers a new learning scenario.

Under this framework, programmable robotic constructions have recently been proposed promising to enhance students' learning science and technology concepts. The LEGO Mindstorms system ([1]) provides a flexible medium for constructionist learning, offering opportunities for design and construction with limited time and small funds. It is comprised of building materials (regular blocks, gears, pulleys and axels) and programming software with an effective graphical interface for developing robotic applications based on LEGO robots.

In the previous versions the available tools were RCX Code and Robolab ([2]), whereas the current NXT version offers a more powerful tool ([3]) based on a customisation of LabVIEW, a well-known controlling and simulation environment developed by National Instruments. All these tools provide draggable icons to represent every programmable robotic element of the kit (motors and sensors) together with simple control structures. Programming a robot results in juxtaposing a sequence of iconised actions, possibly related on events and/or states produced by the applied sensors. The parameterisation of these actions is easily done through the graphical interface. Moreover LEGO NXT offers the opportunity to exploit alternative approaches: its firmware is 'open source', the host-robot communication protocol is well-documented and the descriptions of several different experimentations on controlling/programming the robot are already available. Some of them are based on specific programming languages (e.g. NBC, [4]) whose complexity can be calibrated on the basis of the pupils' level. More recently the Microsoft Robotic Studio initiative ([5]) has produced a first release of the environment which already supports LEGO NXT.

Currently the market offers an increasing variety of robotic proposals that we intend to investigate in the context of our project according to the requirements of different levels of learning and stimulating disciplines. For example you can find already constructed and very simple programmable units, like Bee-Bot ([6]) and the Parallax Scribbler ([7]); kits designed for making artistic creations, like PicoCricket ([8]) which similarly as LEGO Mindstorms comes from the MIT Media Lab researches; much more complex humanoid architectures like Robotis ([9]). More or less all these different options and approaches show that the programmable constructions make possible new types of science experiments, in which children investigate everyday phenomena in their lives both in and out of the classroom (Resnick et al., 1996).

The educational meaning of Robotics in school education, the methodology that should be used to introduce Robotics in school and teacher education and the design of robotics-based educational activities within a teacher training curriculum are among the main problems that the TERECOP project ("Teacher Education on Robotics-Enhanced Constructivist Pedagogical Methods") intends to copy with. Figure 1 summarizes the relation between these open problems, constructivism, constructionism, Logo & Lego.

This paper describes our current, and planned for the future, work within the TERECOP project (October 2006-September 2009). The starting point of this project is described in the next sections focusing on the context, on the aims of the project and on the different partners' countries experiences. The different stages that are going to be developed to implement the project are outlined and a short description of every one is given. Finally some preliminary conclusions are presented.

Starting point for TERECOP project

Context of the project

Our project is inspired from

- the constructivist theories of Jean Piaget arguing that human learning is not the result of a transmission of knowledge, but an active process of knowledge construction based on experiences gained from the real world and linked to personal, unique pre-knowledge (Piaget 1972).
- the constructionist educational philosophy of S. Papert adding that the construction of new knowledge is more effective when the learners are engaged in constructing products that are personally meaningful to them. Constructionism (Papert, 1992) is a natural extension of constructivism and emphasizes the hands-on aspect. The learners in a constructionist environment build something on their own, preferably a tangible object that they can both touch and find meaningful. The goal of constructionism is giving children good things to do so that they can learn by doing much better than they could before (Papert, 1980).

In this theoretical frame a socio-constructivist view is adopted, where learning is not an individual, but a particularly social and societal activity that means that learning always takes place in a social context. Under such a framework the use of educational technology could contribute to the realisation of

- meaningful learning based on students' own team work with teaching materials;
- authentic learning using learning resources of real-life, occupational situations, or simulations of the every day phenomena;
- social learning: technology supports the process of joint knowledge development; the available e-learning environments can support collaboration between fellow students, who can be at different schools, at home or abroad;
- active-reflective learning: students work on experiments or problem-solving using available resources selectively according to their own interests, search and learning strategies;
- problem-based learning: a method that challenges students to "learn to learn"; student groups are seeking solutions to real world problems, which are based on a technology-based framework used to engage students' curiosity and initiate motivation, leading so to critical and analytical thinking;

So, a first premise of this project concerns the implementation of constructivist – constructionist methods not only in classroom, but in teacher education as well. A second premise is referred to the technology-enhanced learning as occurred in the implementation of different kinds of curriculum innovation in the classrooms. A third is related with the emerging need for a teaching as a research-based profession and for the creation of a culture in which researchers and teachers can create a shared body of knowledge.

Goals of the project

The implementation of robotics-enhanced constructivist teaching and learning practices in science and technology classes, however, demands that teachers play a new role. This means that opportunities, like exposure to a number of critical examples and experience in designing computer-based robotic activities and integrating them in their classroom practice in constructivist ways, are of great priority. The goal of our project is teachers to be convinced by their own personal experience for the potentiality of robotic technology as a learning tool. Based on this principle, we intend to develop and implement a teacher-training course to support teachers' professional development. Course participants, who will be practicing or in-service teachers, will be provided with opportunities to examine how robotic technologies can be used to promote a constructivist approach to learning under a co-operative and collaborative frame of work.

So the overall aim of the project is to develop a framework for teacher education courses in order to enable teachers to implement the robotics-enhanced constructivist learning in school, and reflect on their experiences from the implementation of this framework. More specifically our objectives are:

- To develop a methodology of innovative collaborative strategies supporting social constructivist teaching and learning, applied both in the teacher courses and in students' teaching and learning.
- To select and organize a repertoire of appropriate robotics-based learning environments that can support robotic activities and produce a set of critical examples for using in a constructivist way with teachers of secondary level in science and technology subjects.
- To test and evaluate the practical implementation of the selected tools both in training courses and in real classrooms situation (by the trainees).
- To create a community of practice between educators and teachers for facilitating and sustaining teachers' professional development in using robotic tools to support their students' learning by active exploration and social construction of new knowledge.

In the pre-mentioned frame of objectives key issues to be addressed during the project are:

- The integration of technological, cognitive, pedagogical, and social aspects in order to design and develop learner-centred technology-enhanced learning environments with an emphasis on technology as a cognitive partner in learning.
- The design of robotics-enhanced activities where learners learn with technology and accomplish cognitive tasks beyond their reach.

In this project we regard that technology alone cannot affect minds. Our curriculum design will follow an innovative constructivist perspective with an emphasis on aligning computer and robotic technology with learning objectives and learners' needs for the purpose of constructing meaning in social learning environments. In such learning environments the focus is not on the individual but on "interactive environments" that include individuals interacting with each other, instructional materials, subject matter, and tools. Computer-based robotics is an innovative technology that can create or support a rich interactive environment encouraging constructivist learning.

The joint cognitive partnership between technology and learners depends on mindful engagement and interaction. Consequently, to engineer a desirable effect with or of a technology requires more than just introducing the technology. Therefore, in this research project we will apply constructivist pedagogy and a learner-centred didactical approach taking into consideration learner's characteristics for an effective technology-enhanced learning design.

Striving for a collaborative learning environment is based on the belief that the inherent dynamics of a necessary mutual process are likely to be more conducive to meaningful transformation, carrying so a sense of greater potential for development. This is highly supported by the development of e-learning communities.

The target groups of the project include

- a) student-teachers expected to be educated in a way that robotic technology-based learning will play an important aspect of their future work as teachers or professional educators
- b) in-service teachers expected to become aware of the robotic technology-based learning and of different classroom uses and activities for improving their students' learning in science and technology
- c) teacher educators expected to be informed for providing similar courses in local level and
- d) educational authorities expected to undertake future action on teacher technology-based education and further training

Some elements describing the current situation in some of the partners' countries

Robotics is not included in the official curriculum of **Greek** school education. Some occasional implications are mentioned in literature mainly for research reasons. There have also been a few examples of use of Robotic activities with Lego Mindstorms in private schools as extra curriculum activities (Ekpaidefthiria Douka [10], Phychiko College [11]). Some evening private schools (frontistiria) also use these technologies to teach computer skills to young students (e.g. Interactive Learning [12]).

Nevertheless, educational Robotics seem to be very popular in higher education and especially in Engineering and Computer Science departments, as part of the curriculum or as a subject for extended coursework e.g. at the National Technical University of Athens, National Technical University of Patras, University of Macedonia, University of Crete. Moreover, several research projects in this field have been developed focusing on the use of educational Robotics in primary and secondary education. Frangou and Kynigos (2000) used Lego Robotics with secondary students (13-15 years old) in order to investigate educational aspects of these technologies. They found that through Robotics students can acquire hands on experience on variety of science concepts, develop problem solving skills and progress in constructing physical and computer models. The project "Technical school students design and develop robotic gear-based constructions for the transmission of motion" developed by the School of Pedagogical and Technological Education in Patras investigated how programmable robotic constructions can be

effectively used in Technical and Vocational Schools (student age 16-20). In that project students were invited to design, develop and program a robotic construction using the Technological Inventions LEGO Mindstorms Package. The project provided very promising indications that students learn important mathematical and scientific concepts through their own design and programming activities (Alimisis et al., 2005, Karatrantou et al., 2005). Another project that investigated the potential of educational Robotics in teaching programming in secondary education stressed the importance of the interaction between the construction and the algorithm of the software in understanding basic programming structures (Kagkani et al., 2005). Finally, two more research projects focused on primary education. They stressed the cooperative character (Dimitriou & Xatzikraniotis 2003) and the experimental aspect (Karatrantou et al 2006) of robotic activities.

Though Robotics is not officially included in the **Italian** primary and secondary educational system, the interest on educational robotics is rapidly increasing. Apart from the contributions of isolated experiences and advanced laboratories in technical secondary schools and universities, some relevant recent projects, involving both school teachers and experts, are giving impulse to the subject. Among others: *Usò didattico della Robotica* (educational use of robotics) at IRRE Piemonte ([13]); *Costruiamo un Robot* (let us build a robot) ([14]); *La bottega dei robot* (the robot shop), The National Science and Technology Museum of Milan ([15]); *Robot@Scuola*, a school network involved in educational robotics ([16]); *EduRobot*, The Institute for Educational Technology of Italian National Research Council ([17]); *AmicoRobot*, a school network in Milan ([18]). Most of these projects are related to the Lego Mindstorms robotic architecture.

The **Spanish** situation is similar to the previous ones, and the use of robotics in primary and secondary education is very limited and not official at all. In general there are a lot of activities in the field of Robotics in Spain mainly in research or industry ([19]) and there are also a few robot competitions organized ([20]). In some of these competitions the participants are secondary level students. It is also a fact that the different educational institutions (national and regional) are aware of introducing and using computer science & technology at schools ([21] in Catalonia, [22] in Madrid, [23] in Navarra or [24] as the national reference in Educational Computer Science & technology). Nevertheless it is quite difficult to find deep and complete experiences in Robotics & Education. Some of the relevant experiences are the use of LOGO (the approach is similar to ours) at school ([25]), some teachers' initiatives like the project RESS (secondary level experience with LEGO done in 2003: [26]) or personal ones like the web page and materials from the "freelancer" Koldo Olaskoaga ([27]). We found two experiences close to our project; one in Educational Robotics done in Primary school level by Alfredo Rodríguez Rebollo (Director of the public college "San Francisco de Cifuentes", Guadalajara, Spain [28]) with an important effort in integrating these activities within the curricula; the other one carried out by the University of Alicante group called TEDDI, which works (among other research areas) in finding didactic applications of robotics at different levels in school ([29]).

Implementing the project

The project TERECoP started in October 2006 in the frame of the European Programme Socrates/Comenius/Action 2.1 (Training of School Education Staff) and its total duration will be 3 years. 8 institutions from 6 different European countries participate in the project: School of Pedagogical and Technological Education (GR, coordinator), Institut Universitaire de Formation des Maîtres d'Aix-Marseille (FR), Department of Information Engineering – University of Padova (IT), University of Pitești (RO), IT+Robotics srl (IT), Museo Civico di Rovereto (IT), Charles University Prague, Faculty of Education (CZ), Public University of Navarre (ES). During the 1st year a methodology for designing robotic technology-enhanced constructivist learning will be developed and teacher education courses will be designed. During the 2nd year a pilot and a final teacher education course will be implemented including testing of trainees' teaching activities in school classes. Finally during the 3rd year the evaluation of the courses and the development of dissemination activities will take place.

Expected outputs of the project

An **e-community** will be created to offer for both educators (from the beginning of the project) and teacher-students (during the 2nd and 3rd year) a communication platform including:

- a public space available for all the members of the community (educators and student-teachers) to post their messages and to upload their files;
- a forum to develop discussions on selected topics related to the project subject;
- synchronous and asynchronous communication through bulletin boards, chat and e-mail services.

The e-community intends to support the development of a learning community engaging the teacher-learners in social learning, supporting meaningful conversations among learners and between educators and learners, promoting new perspectives and helping them to construct knowledge in a collaborative way.

A **project web site** (see figure 2, <http://www.terecop.eu>) presents the whole work done in the frame of the project and connects the project with the broad educational community.

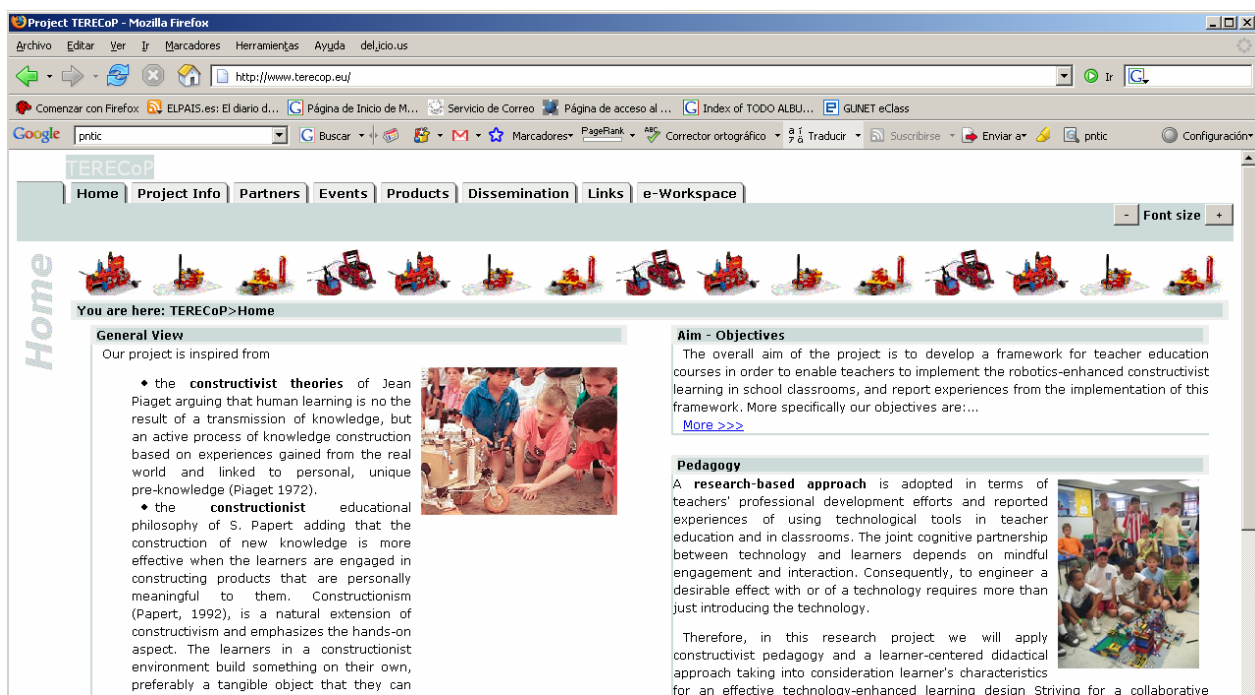


Figure 2. The TERECOP Web site

The partners are currently working to develop a **methodology** for designing computer-based robotics-enhanced constructivist learning, applied both in the teacher courses and in students' teaching and learning. The methodology, incorporating results from the relevant research literature (books on subject, educational journals, proceedings of educational conferences, web resources and educational software tools), will outline basic principles, learning objectives and strategies, appropriate technology-based environments and learning activities and some critical examples of robotics-enhanced constructivist learning.

The design of the teacher education courses will be based on the methodology developed in the beginning of the project; this will permit us to design a pilot course curriculum. The design of the course curriculum will include **learning materials** and **evaluation tools**. Emphasis will be placed:

- on the development of innovative collaborative strategies between educators and teachers
- on the selection of expressive or exploratory learning activities that can support social constructivist teaching and learning.
- on the practical use of the selected tools in a real classroom context.

Pilot courses

The pre-mentioned course design will be implemented and evaluated with student-teachers in three different countries by the corresponding partners. From the beginning of the face to face course student-teachers will be invited to participate in an e-community and will have access to e-learning materials. In these courses student-teachers will elaborate on the development of robotics-based constructivist teaching activities and materials for their students. They will be encouraged to create and present joint projects regarding constructivist teaching activities planned to be implemented with school students, and to argue for their choices. The student-teachers will also be encouraged to implement their projects in real school classes, where it is possible, and to evaluate them in cooperation with their tutors. The projects and the evaluation results will be published and discussed in the e-community where educators and teachers will have the opportunity to share and reflect on their experiences.

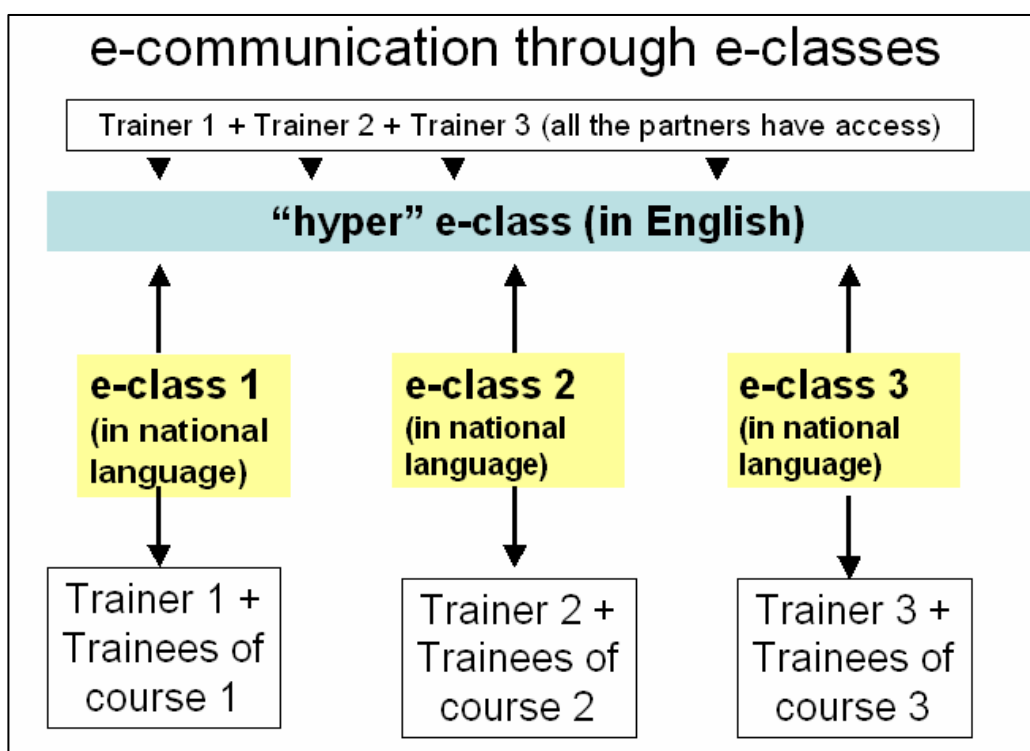


Figure 3. E-classes organization for the pilot courses

Evaluation

An evaluation report on the pilot courses will be presented to all the partners, and using it as a feedback, a revised curriculum of the courses based on the evaluation results will be developed. After that, 3 teacher training final courses will be organized using the revised curriculum and learning materials and combining again the face to face course with an e-class community. The evaluation of the courses will be carried out using the same (revised if necessary) evaluation instruments developed for the needs of this project based on data collected in the courses and on data collected from the implementation of students' projects in a real school class. In the end of this process a final revised curriculum based on the results and findings coming from the evaluation process are expected to be obtained.

Courses in the future through the Comenius Catalogue

The final course, as it will have been refined in the end of the project, will be offered for in-service training of secondary science and technology teachers from the whole European educational community through the Comenius Catalogue following the same constructivist

pedagogy and applying the learner-centred approach developed during this project. Innovative collaborative strategies supported by the development of e-learning communities will make possible the cooperation between educators and teachers during and after the end of the face-to-face courses.

Finally, the project results and all the experience, that we expect to gain in the project, will be used for the long-term improvement and renewal of the education methods implemented by our institutions regarding the technology-enhanced learning. The planned dissemination activities are also expected to contribute to a long-term exploitation of the project results by educational institutions, authorities, policy-making bodies, unions and networks that will become aware of them for integrating computer-based robotics in teaching and learning and transforming their teaching/learning environment towards constructivist learning.

Conclusions

Robotics is a growing field that has the potential to significantly impact the nature of technology and science education at all levels, from primary to graduate school.

- So far robotics has been introduced mainly in departments of engineering at university level.
- Last few years several attempts have been made in international level (our countries included more or less) to introduce robotics in secondary school education mostly in science and technology subjects.

Although the role of teacher is crucial for the successful introduction of robotics in classrooms, only few projects have been undertaken to train school teachers in using this, completely new for them, technology.

TERECOP project's aim and ambition is to contribute to fill in this gap suggesting a constructivist model of teacher training in these new technologies. So, TERECOP project is expected to be a beneficial one for teachers both at national and European level enabling them to introduce robotics in their classrooms in a constructivist framework. We also hope that its outcomes will constitute a significant educational advantage for students (end-users), for teachers and for the science and technology education in general.

Acknowledgements

This paper was based on work done in the frame of the project "Teacher Education on Robotics-Enhanced Constructivist Pedagogical Methods" (TERECOP) funded by the European Programme Socrates/Comenius/Action 2.1, Agreement No 128959-CP-1-2006-1-GR-COMENIUS-C21 2006 – 2518 / 001 – 001 SO2.

References

- Alimisis, D., Karatrantou, A., Tachos, N. (2005) *Technical school students design and develop robotic gear-based constructions for the transmission of motion*, Eurologo 2005, Digital Tools for Lifelong Learning, Proceedings, Warsaw, Poland, pp. 76-86
- Dimitriou A., Xatzikraniotis E (2003) *Educational robotics as a tool for skills development*, In Proceedings of the 2nd national teachers' conference on "ICT in Education", Syros, May 2003, pp. 146-157 (in Greek).
- Kagkani K., Dagdilelis V., Satratzemi M., Evangelidis G., (2005) *A case study of teaching programming in secondary education with Lego Mindstorms*. In on- line Proceedings of the 3d National Conference "Teaching Computer Science", University of Peloponnese, Korinthos, 7-9 October 2005, http://www.etpe.gr/uploads1/paper_s53.pdf (in Greek).
- Karatrantou A., Panagiotakopoulos X., Pierri E. (2006) *Robotic constructions of Lego Mindstorms and science understanding in primary education: a case study*. In Proceedings of the 5th National

Conference on ICT in Education, University of Thessaloniki, 5-8 October 2006, pp. 310-317 (in Greek).

Karatrantou A, Tachos N., Alimisis D., (2005), *Introduction in basic principles and programming structures using the robotic constructions LEGO Mindstorms*. In on- line Proceedings of the 3d National Conference "Teaching Computer Science", University of Peloponnese, Korinthos, 7-9 October 2005 http://www.etpe.gr/uploads1/paper_s81 (in Greek).

Kynigos C. and Frangou S. (2000), *Aspects of pedagogical use of control technology in school class*, In Proceedings of the 2nd national conference on "ICT in education", University of Patras, pp. 83-91 (in Greek).

Niederer, H., Sander, F., Goldberg, F., Otero, V., Jorde, D., Slotta, J., Stromme, A., Fischer, H., Lorenz, H., Tiberghien, A., Vince, J. (2003). *Research about the use of information Technology in science education*. In Dimitris Psillos et al. (Eds.) *Science education research in the knowledge- based society*, Kluwer, pp. 309-321.

Papert, S. (1980). *Mindstorms: Children, Computers, and Powerful Ideas*. N.Y.: Basic Books.

Papert, S. (1992). *The Children's Machine*. N.Y.: Basic Books.

Piaget, J. (1972). *The Principles of Genetic Epistemology*. N. Y.: Basic Books.

Piaget, J. (1974), *To understand is to invent*. N.Y.: Basic Books.

Resnick, M., Martin, F. G., Sargent, R., & Silverman, B. (1996). *Programmable bricks: Toys to think with*. IBM Systems Journal, 35(3&4), pp. 443-452.

[1] <http://www.legomindstorms.com>

Lego Mindstorms Web site

[2] <http://www.cceo.tufts.edu/>

Robolab

[3] <http://www.ni.com/academic/mindstorms/>

NXT-G iconic language

[4] <http://bricxcc.sourceforge.net/nbc/>

NBC language

[5] <http://msdn.microsoft.com/robotics>

Microsoft Robotic Studio

[6] <http://www.bee-bot.co.uk>

Bee-bot robot

[7] http://www.parallax.com/html_pages/robotics

Scribbler robot

[8] <http://www.picocricket.com>

Pico Cricket robot

[9] <http://www.robotis.com>

Robotis humanoid

[10] <http://www.doukas.gr/tp/tp112.htm>

Ekpaidefthiria Douka

[11] <http://www.haef.gr>

Phychiko College

[12] <http://www.interactive.gr>

Interactive Learning

[13] <http://robotica.irrepiemonte.it/robotica/index.htm>

IRRE Piemonte

[14] <http://www5.indire.it:8080/set/microrobotica/default.htm>

Costruiamo un Robot

[15] http://www.museoscienza.org/est/museo/robot_0.asp
Technology Museum of Milan

The National Science and

[16] <http://www.scuoladirobotica.it/retemiur/>

Robot@Scuola

[17] http://www.itd.cnr.it/Progetti_Rispo1.php?PROGETTO=93

EduRobot

[18] <http://www.amicorobot.net/>

Amico Robot

[19] <http://www.cea-ifac.es/wwwgrupos/robotica/index.html>
<http://www.robocity2030.org/>

Robotics in Spain

- [20] <http://complubot.educa.madrid.org/>
<http://www.roboteca.org> Robot competitions in Spain
- [21] <http://www.xtec.es/> Education in Catalonia
- [22] <http://www.educa.madrid.org> Education in Madrid
- [23] <http://www.pnte.cfnavarra.es/> Education in Navarra
- [24] <http://www.cnice.mecd.es/> Education in Spain
- [25] <http://roble.cnice.mecd.es/~apantoja> Logo in Spain
- [26] http://www.cnice.mec.es/pamc/pamc_2003/2003_proyecto_ress/
RESS project
- [27] <http://www.euskalnet.net/kolaskoaga/es/>
<http://robotikas.blogspot.com/> Koldo Olaskoaga
- [28] <http://www.educa.jccm.es/educa-jccm/cm/revistaldea>
college San Francisco de Cifuentes
- [29] <http://www.teddi.ua.es/> TEDDI

Teacher Education to Promote Constructivist Use of ICT: Study of a Logo-based Project

Dimitris Alimisis, *alimisis@otenet.gr*

Department of Education, School of Pedagogical & Technological Education, Patras, Greece

Abstract

In the frame of a training course for future teachers, an ICT-enhanced project, based on a Logo learning environment and inspired from a constructivist education philosophy, was undertaken aiming at familiarizing students with the use of computer as a tool that can trigger constructivist learning and helping them to adopt an exploratory and constructivist teaching practice.

The training methodology applied in the training course focused on the trainees' self-activity and active engagement in lab exploratory activities. The common feature of the training activities was not the programming language but the so-called "Logo spirit", a constructivist spirit of action and learning. Through such a process, it was expected that student-teachers would gain personal experiences of exploratory and constructivist learning and would be able to inspire into their future students the same educational spirit.

Student-teachers were asked to give instructions to their turtle to draw regular polygons and a circle using movement commands. They were encouraged, before they give instructions to the Logo turtle to draw a geometrical figure, to analyse the problem and to think about the instructions. Very often student work resulted in unexpected wrong figures on the screen. So students had the opportunity to recognise their errors and to try again. This process was repeated several times until students reached the expected result. They were encouraged to try several solutions and keep trying until they succeed, to work with self-action and autonomy and finally to be questioned about the educational value of those activities.

- The evaluation of the project was based on
- Teacher-students' achievement on the tasks as it was recorded on their work sheets.
- A 5-point Likert scale measuring the degree to which students agreed or disagreed with statements regarding the interest and the educational usefulness of the learning experience they had working on the tasks.
- Answers and explanations on open questions given in written form asking them to detect and mention any positive educational issues they had found in that method and to think of and report similar methods for teaching their school subjects
- Two case-studies: two teacher-students were encouraged to implement the same activities in their school class with their students and report results and experiences.

The evaluation results indicated a positive impact on the students in terms of familiarization with the spirit of the proposed methodology and of understanding the potential of ICT for constructivist learning. The evidence coming from the two course participants, who applied and evaluated the methodology in a school class, provided encouraging indications that the students became capable of applying the methodology they had learnt in real classroom settings.

Keywords

constructivist pedagogy; teacher education; Logo; problem solving

Introduction

This study was based on two premises. The first concerns the implementation of the ICT-enhanced constructivist learning today in classroom. The second refers to the emerging need for the appropriate teacher education and professional development as a presupposition for the implementation of constructivist innovation in classrooms.

Computer technology and the constructivism paradigm

There is wide consensus in education that learning is no longer seen simply as the result of a transmission of knowledge. Nowadays pedagogical strategies employed in the current ICT-based learning are linked to *constructivism* paradigm. According to constructivism, knowledge is considered to be socially and individually constructed; learning is the acquisition of meaningful competences in a realistic context; learning is advanced through interactive and authentic experiences that dovetail with the interests of the student and through active learning. So the focus is on the development of a suitable environment for constructing knowledge rather than for its transfer.

In such an environment the use of ICT can trigger constructivist innovation in the classroom contributing to the realisation of meaningful authentic, active-reflective and problem-based learning, a method that challenges students to "learn how to learn"; students seek solutions to real world problems, which, based on an ICT framework, are used to engage their curiosity and initiate learning, leading so to critical and analytical thinking.

The constructivist education philosophy aims at a school where students learn how to learn, in a learner-centered environment with emphasis on learning through discovery and exploration and on experiences in the development of problem-solving strategies (diSessa et al 1995).

The emerging need for teacher education in ICT-enhanced constructivist learning

ICT-enhanced constructivist classroom practices, however, demand that teachers play a new role. This means that opportunities, like exposure to a number of critical examples and experience in designing ICT-based activities and integrating them in their classroom practice in constructivist ways are of great priority. The aim is to convince teachers for the potentiality of ICT as constructivist learning tool through their own personal experience. For this reason the development and implementation of appropriate courses is very important for the teachers' professional development and crucial for the success of innovative approaches using ICT.

Teachers need to go beyond traditional approaches (Bhattacharya and Richards, 2001) and become acquainted with new methods in order to get a clear understanding of the educational functionality of technological tools in their educational practices. The approaches to staff training include the need for awareness of the advantages and possible difficulties of the proposed methods for school learning and usage of settings and tools for training similar to those expected to be used in classrooms in the sense of learning by doing and applying the new knowledge in real learning contexts.

Designing ICT-enhanced constructivist learning

Under the pre-mentioned theoretical framework, appropriate ICT-enhanced activities, critical examples and strategies that can be applied both in teacher training courses and in students' teaching to trigger constructivist innovation, have been integrated in the curriculum of the courses offered by the Educational Technology Lab at the School of Pedagogical and Technological Education (ASPETE) in Patras (Greece). Course participants, who are future teachers, are provided opportunities to examine how ICT can be used to promote a constructivist, learner-centered approach to learning.

Using a Logo environment as tool for constructivist learning

Logo is one of the ICT tools used among others in the training activities. Logo was selected because it is not only a dynamic programming language but also a valuable problem solving tool and a general mind tool. It is adjustable to student cognitive structures and offers micro worlds, simulations, open problem solving environment, diagnosis of student own errors, control and autonomy of their learning, spontaneous reflection, and development of self-knowledge (Papert 1980, Noss 1987, Hoyles and Sutherland 1989).

As the students programme the computer, they teach the machine to think and through this process they also discover the patterns of their own personal thinking (Papert 1980). Logo in general supports constructivist learning and the use of computer as a tool for the development of intellectual skills. So, Logo is used as a tool that can trigger constructivist learning in classroom. The objective of training activities with Logo is mainly to help teachers to recognise its educational value and to adopt corresponding teaching methods in their future classroom.

The training methodology

Basic constructivist conceptions are presented and discussed with the trainees before and concurrently with the lab activities focusing on Papert's theoretical work and especially on *Mindstorms* (Papert 1980), which is given to the trainees among other resources. The training methodology applied in our training course focuses on the trainees' self-activity and active engagement in lab exploratory activities. The common feature of the training activities is not the programming language but the so-called "Logo spirit", a constructivist spirit of action and learning (Papert 1980). Through such a process, it is expected that student-teachers will gain personal experiences of exploratory and constructivist learning and will be able to inspire into their future students the same educational spirit.

Student-teachers are asked to work on problem solving with self-action and autonomy and to be questioned about the educational value of those activities. They are encouraged, before they give instructions to the Logo turtle to draw a geometrical figure, to analyse the problem and to think about the instructions. Very often student work results in unexpected wrong situations on the screen. So students have the opportunity to recognise their errors and to try again. This process is repeated several times until students reach the expected result. This way, learning turns out to be a personal "adventure" of knowledge construction.

The study

In the frame of the evaluation of our courses, a study was undertaken during the 2nd semester of academic year 2005 to examine their impact on student-teachers. A part of that study which monitored and evaluated some representative Logo-based training tasks is reported in this paper including evaluation data and results.

The sample

57 student-teachers participated in the training activities. They had already finished their studies in a university (most of them in technical disciplines) and were attending a pedagogical course of two semesters in the School of Pedagogical and Technological Education (Patras, Greece) in order to qualify for teaching in the corresponding discipline in secondary education. The training activities were developed during the 2nd semester (2005). So, students had already been taught, among other pedagogical subjects, theories of learning, including constructivism. None of them had any previous experience in using Logo but they were comparatively ICT-literate.

The training tasks

First of all, students had the opportunity during one teaching session to familiarise themselves with the command centre of MicroworldsPro (Logo Computers Systems Inc) and the basic Logo movement commands. In a next session a worksheet that presented the tasks assigned to the

students with the necessary instructions was handed over to students. There was no other help given to them except technical advice when it was needed.

In those tasks students were asked to give instructions to their turtle to draw regular (meaning identical sided and identical cornered) polygons using four movement commands: *forward*, *back*, *right*, *left*. They could try several solutions, and they were encouraged to keep trying until they succeed. The exact tasks were the following:

Task 1: write instructions for the turtle to draw a regular square.

Task 2: continue with a regular triangle

Task 3: can you invent a command to help you make the previous polygons easier and quicker to draw?

After the end of task 3 the command *repeat* was introduced and we showed to the students how to use it:

- `repeatspacen[Commmand1spaceArgument1spaceCommmand2spaceArgument2space etc. ...]`

They were asked to try this new command on the triangle and the square and to write their commands in a table.

Task 4: Continue for drawing the next regular polygons (Pentagon, Hexagon, Octagon, Nonagon, Decagon) using the command *repeat*. Write your commands in a table.

Task 5: Now try a circle. Notice and use the previous “sides-angle” pattern.

Task 6: Look at your whole work. Can you see a relationship between the *Sides* and the *Angle* numbers? Can you write a theory that says how many degrees need to be turned to complete the polygon so that your turtle returns to its original heading?

After the end of the tasks a discussion was organised in the class where the expected solutions and answers were presented and the entire student work was commented.

The evaluation instruments

The evaluation of the project was based on

- Teacher-students’ achievement on the tasks as it was recorded on their work sheets.
- A 5-point Likert scale measuring the degree to which students agreed or disagreed with 4 statements (2 positive, 2 negative) indicated in table 7 regarding the interest and the educational usefulness of the learning experience they had working on the tasks.
- Answers and explanations on open questions given in written form asking them to detect and mention any positive educational issues they had found in this method and to think of and report similar methods for teaching their school subjects

Two case-studies: two teacher-students were encouraged to implement the same activities in their school class with their students and report results and experiences.

Results and discussion

- The evaluation data collected with the instruments mentioned above are presented on the following tables and commented shortly.

Students’ achievement in the tasks

- The first task appeared a rather easy one for the students. Almost all of them after only few trials gave the right instructions to the Logo turtle to draw a square with dimensions of their own choice using mostly the set of commands *forward 100 right 90* (4 times).

<i>results</i>	Frequency	Percent %
failure	02	03.5
Success	55	96.5
Total	57	100.0

Table 1. Students' achievement in task 1: drawing a square

- The 2nd task proved more difficult compared to the first one. The students tested several methods before they were able to write a set of commands to draw a regular triangle. The main difficulty they encountered was identified in the estimation of the right angle. Most of them, knowing that in the regular triangle each angle is 60 degrees, started trying this angle. This choice, unexpectedly for them, didn't result in a triangle. They needed to repeat their effort several times in order to find that the angle of the turtle's turn should be different from that of the triangle.

<i>results</i>	Frequency	Percent
failure	9	15.8
success	48	84.2
Total	57	100.0

Table 2. Students' achievement in task 2: drawing a triangle

- Some of them were observed to put themselves in the turtle's position trying to find the right turn (120 degrees). The lesson learned from this unexpected initial failure was that all of us (including children), need to concretise a problematic situation in order to reach a solution. The computer screen proved such a tool for the concretisation of abstract notions. What the students watched on the screen was just a visualisation of their own thought. So they had the opportunity to re-assess their own initial thinking and to correct their mistakes.

<i>answer</i>	Frequency	Percent
"repeat"	09	15.8
"square-triangle"	08	14.0
No clear answer	07	12.3
other	05	08.8
No answer	28	49.1
Total	57	100.0

Table 3. Students' answers in task 3: inventing a command

- The invention of a command that would help them to make the triangle and the square easier and quicker to draw was proved a difficult task. Only a few students found the expected command *repeat*, although almost all of them had already used the repetition of movement commands in the previous two tasks. 8 students answered suggesting the names of the wanted polygons ("square-triangle"). These answers are interesting because they might be interpreted as suggestions for relevant routines drawing immediately the corresponding polygons and could be exploited as a good opportunity for an educator to introduce the Logo processes.

<i>results</i>	Frequency	Percent
failure	01	01.8
success	40	70.2
partial success	16	28.1
Total	57	100.0

Table 4. Students' achievement in task 4: drawing the next polygons

- The students continued with the next polygons using the new command *repeat* and writing their commands in a table indicating the number of sides and the angle of turtle's turn for each polygon. The majority succeeded in drawing the polygons but there was a significant percentage of those who drew some polygons successfully but failed in others. Once again the main difficulty was encountered in the angle. The same instance mentioned above of students' testing different methods was observed in this task too.

<i>results</i>	Frequency	Percent
failure	04	07.0
success	37	64.9
A new polygon with more sides	11	19.3
More than one circle	01	01.8
missing	04	07.0
Total	57	100.0

Table 5. Students' achievement in task 5: drawing a circle

- The final drawing asked by the students was the circle. The great majority of students continued to draw polygons with a continuously increasing number of sides and a decreasing angle. These efforts were resulting in polygons gradually approaching the circle in the term of a polygon with an unlimited number of sides. Some efforts stopped in a new polygon having simply more sides than the previous ones failing in seeing the "sides and angle" pattern indicated on the table where the number of sides and angles had been recorded. But the majority of the students could exploit that table and identified the relationship between the numbers of sides and angle. This relationship guided them to draw the circle using a command like the following: *Repeat 360 [forward [step] right 1]*.
- Finally students were asked to look at their whole work and write a theory saying how many degrees need to be turned by their turtle in order to complete the polygon and to return to its original heading. Not surprisingly, the number of students who could identify the wanted theory was identical to that of students who succeeded in drawing the circle. It is obvious that the identification of the "sides and angle" pattern was crucial for the successful fulfilment of the previous circle-drawing task.

<i>results</i>	Frequency	Percent
No answer	11	19.3
Angle x sides = 360 degrees	37	64.9
Faulty	5	08.8

missing	4	07.0
total	57	100.0

Table 6. Students' answers in task 6: concluding a theory

- The observation of student work and behaviour during the tasks showed that the tasks attracted the attention and interest of the great majority and in some cases students worked enthusiastically. Scenes of celebrations were observed in some cases in which students had achieved, after several unsuccessful trials, to instruct their turtle to draw the expected polygon (and especially the circle) on their screen. We also noticed the insistence of some others on continuing to experiment on a task even after the end of the given session.

Students' opinions on the interest and the educational usefulness of the tasks.

<i>In my opinion these activities with Logo were...</i>	Agree	Rather agree	No opinion	Rather disagree	Disagree
An interesting learning experience	90.6	09.4	00.0	00.0	00.0
Boring	02.2	00.0	00.0	11.1	86.7
No useful	08.9	08.9	11.1	22.2	48.9
Helped me to think of similar activities for my teaching subjects	52.1	20.8	14.6	06.3	06.2

Table 7. Percentage % of students' opinions on the interest and usefulness of the tasks (N=57)

Students were also asked to evaluate in writing the entire project regarding the interest and the educational usefulness of the learning experience they had working on the tasks. The evaluation results (table 7) indicate that almost all the students agreed with the statement that the entire project was an interesting and far from boring learning experience. The great majority (a total of 71.1%) agreed or rather agreed that the Logo activities were useful and helpful regarding their possible transfer to other school subjects (a total of 72.9%). There were a low percentage of students, who although they found the activities interesting, were not convinced of their usefulness in an educational context, possibly due to the subject of the activities that is the construction of geometrical drawings, which perhaps made it difficult for some students coming from different disciplines to transfer the educational meaning into their school subjects.

Answers and explanations to open questions

- In order to examine in more depth the impact that our project had on students, we asked them, if they had agreed (or rather agreed) on the 1st and 4th statements of the table 7 (concerning the interest and usefulness of the tasks) to think about and answer in writing the following two open questions:
- Question 1: What positive educational issues did you find, if any, in this method? Explain your opinion shortly.
- Question 2: Can you think of similar methods for teaching your school subjects? If yes, please describe shortly.

Categories of answers (This method ...)	frequency
offers creative learning	6
is attractive and interesting for students	5
promotes learning through discovery	5
facilitates the understanding of concepts concerned	5
offers active role for students	4
exploits students' mistakes	2
develops students' imagination	2
cultivates intellectual skills	1
provides motives for learning	1
concretises abstract notions	1
is useful as exercise but only after teacher's presentation of concepts concerned	3
no clear answer	5

Table 8. Positive educational issues identified by students (open question 1) (N=22)

- 22 students answered each of the two open questions. The answers to question 1 were analysed and categorised qualitatively according to the educational issue mentioned by students. The resulted categories are presented shortly on the table 8 (most of the students mentioned features belonging to more than one category). The majority of them (14/22) could identify and mention several constructivist issues belonging to or approaching constructivist ideas ("offers creative learning", "active role for students", "exploits student faults" etc). They seemed to have been influenced and understood the educational meaning of the activities. This was more obvious in the following characteristic answers and explanations:
 - "...the student is involved in a process that requires deep thought and use of his intelligence in order to reach a solution".
 - "...The student observes his/her errors and is able to correct them".
 - "...In this way student constructs the knowledge through personal work instead of simply receiving information from a teacher"
 - "...Children can approach geometrical concepts in a more free, playful way"
 - "...Students don't simply watch the computer screen, they are active, they use their imagination and cultivate their creativity"
 - "... It helps students to put themselves in the turtle's position"
- Five students failed to mention any precise educational features and answered in very general and unclear terms. Three others re-confirmed that the method was useful and interesting but they would prefer the traditional method of lecture before the application of the method. As one of them explained "...we should present the subject on the blackboard and then apply this method for better understanding".

In the words of another one from the same group:

- "First, I would present the theory to the students and the formula $\text{angle} = 360 / \text{sides}$ to help them understand the rationale of this method and then

I would ask them to apply it... this work would be done faster if there was a library with ready drawings on the screen and students were asked to do only changes in parameters”

- This kind of answers indicates, in our opinion, a misunderstanding of the pedagogic rationale of our project and implies persistence with the traditional model of the teacher-guided instruction of the new concepts instead of the proposed knowledge construction by students themselves.

<i>Categories of answers</i>	<i>frequency</i>
Creation and use of simulations	7
Engineering drawings	7
Other programming techniques	2
Not applied in my subject	4
No clear answer	2

Table 9. Similar methods for teaching school subjects mentioned by students (open question 2) (N=22)

- Answering to the 2nd open question, the majority of students (16/22) suggested a method that could be implemented in their own school subject and described it more or less sufficiently. Students’ answers were categorised according to the method suggested and they were heavily influenced, as it was expected, by their discipline. So, the engineering students suggested different engineering drawings and the informatics students other programming techniques. Seven other students suggested different kinds of simulations to be created by school-students in a way very similar to that they had followed in our project. There were four students (economics and sociology graduates) who answered that they couldn’t imagine any applications of the method in their school lessons. As one of them put it “in social sciences this method does not help the transfer of concepts to pupils”.

Evaluation of two case studies

- Two teacher-students were encouraged to apply the same activities in their school class with their students and to report in writing evaluation results and experiences.
- **Case study 1:** The first teacher-student (teacher of informatics at secondary school level) was initially very sceptical about the implementation of Logo activities in his school class. He was used to teaching informatics in a traditional teacher-centered way with emphasis on lectures, blackboard presentations, and drill and practice exercises on computers. He accepted to test the logo-based activities (the same ones he had already done) with his school-students at a lower secondary school, where he was working on a temporary base. 16 students from grade 3 (age 14) followed the activities.
- According to the report he wrote after the end of those lessons (Stoyannopoulos, 2005) he needed one teaching period (45 minutes) to familiarise students with the Logo command centre and the movement commands mentioned above. Students, introduced to programming for the first time, learned how to use the movement commands fast, although they encountered difficulties with the arguments of the commands. A difficulty appeared in the beginning of the tasks: “there was a student anxiety focusing more on finding the correct solution than on working freely and creatively, due to a tendency to seek high marks”. This point indicates that the teacher-student seems to have realised the necessity for the so-called “Logo spirit” to be present in the Logo activities, a spirit very different from the traditional mark-oriented one that is dominant in the class (a well known situation in Greek secondary education).

- Gradually the logo activities succeeded to attract the students' interest and most of the students through a test and error practice achieved sufficient results. As he mentions "the sides and angle model played a significant role for the successful drawing of the circle by some student groups". He concludes that the overall activity acted very positively for student learning and that "the traditional teaching approaches cannot help students in the development of an algorithmic way of thinking... It is very important for students to be engaged in problem solving in a creative, not mechanistic way working collaboratively in the class..." These last points imply a remarkable shift from his initial scepticism to a more positive attitude to the "Logo spirit" indicating a positive effect that the previous training had on him.
- **Case study 2:** The second student-teacher who accepted to implement the Logo activities was also an informatics teacher in a higher secondary school (student age 16). He had shown great interest in the training sessions and accepted enthusiastically to work with his students on the same tasks. According to his report (Theodorides, 2005), his students after an initial frustration, showed great interest in that work and great commitment to their tasks. Most of them reached sufficiently good results. He explains the initial student frustration noting that "they were engaged in an exploratory way of learning very different from what they were used to so far".
- Evaluating his students' work he notes that "students participated actively in the whole learning process", "it was very productive for students to watch their errors on the screen after a wrong instruction to the turtle... it allowed them to retry another set of instructions and so on... it was like playing with the turtle". He observed that "what students liked more in these lessons was the opportunity they had to work with autonomy, to make decisions and act on them without having to listen passively to lectures". He concludes that "it was a self-regulating process for students facilitating learning through exploration and discovery".
- Similarly to the first one, this case study indicates that our student-teacher succeeded in implementing in his classroom the same activities he had followed during his training in a constructivist way. He seems through his report to recognize in his students' reactions and behaviour in the classroom some of the expected outcomes of the learning approach (very similar to those observed in the training class). He also seems to appreciate this kind of learning as exploratory, self-regulating and different from the teaching methods he used to follow till then.

Conclusions

- The inferences drawn from the data reported in this paper indicate that the inclusion of ICT-based problem solving in the teacher training curricula in a constructivist way helped student-teachers to recognise its pedagogical potential for school learning. Although there were some failures in the students' work and misunderstandings were noticed in a few cases, the whole project seems to have had a significant impact on the majority. This impact included familiarisation of the students with the constructivist learning spirit and concrete personal experience of an exploratory and constructivist learning that can inspire their future teaching methodology and convinced them to use computer as a constructivist learning tool.
- The added educational value of this project concerns not only the advancement of the teachers' technological culture but also (and more importantly) the development of a constructivist pedagogy that can be applied with computers as learning tools in real classrooms. The evidence coming from the two course participants, who applied and evaluated the methodology in classroom with positive results, provided encouraging indications that students became capable of applying the methodology they had learnt in real classroom settings. The monitoring of a long-term impact on student-teachers' future teaching praxis could be subject of further investigation.

Finally, the evidence offered by this project argues that the integration of ICT in teacher education can play an important role to introduce constructivist innovation in teaching and learning. We concur with Becker and Riel (2000) that if we want our schools to offer creative problem-solving and constructivist, independent thinking, the most effective way to achieve these goals may be to design a teacher education system where teachers are encouraged to be creative problem solvers in the design of learning environments for their students.

References

- Becker, H.J. and Riel, M.M. (2000) *Teacher professional engagement and constructivist-compatible computer use*. Available online at http://www.crito.uci.edu/tlc/findings/report_7/text.html (accessed 10 October 2006).
- Bhattacharya, M. and Richards, C. (2001) *Innovative Course Design as Action Research: Instructional Technology for Teacher Education*. In Proceedings of Society for Information Technology and Teacher Education International Conference (Chesapeake, VA: AACE). Edited by C. Crawford et al. pp.1052-1057
- diSessa, A., Hoyles, C., Noss, R. and Edwards, L. (1995) *Computers and Exploratory Learning: setting the scene*. In Computers and Exploratory Learning, A.A.diSessa, C.Hoyles and R.Noss (eds.), NATO ASI Series, 146, 1-14
- Hoyles, C. and Sutherland, R. (1989) *Logo Mathematics in the Classroom*, Routledge, London.
- Noss, R. (1987) *How do Children do Mathematics with Logo?* Journal of Computer Assisted Learning, 1, 2-12
- Papert, S. (1980) *Mindstorms*. Basic Books, New York.
- Stoyannopoulos S. (2005) *Development of basic programming skills from C grade students using MicroworldsPro environment*. Dissertation, School of Pedagogical and Technological Education, Patras.
- Theodorides, A. (2005) *Logo as a tool to understand processes and variables in programming*, Dissertation, School of Pedagogical and Technological Education, Patras.

NeGAS: Authoring System for 3DCG using extended turtle metaphor

Katsuhide Tsushima, ktsushima@sannet.ne.jp

Dept of Digital Games, Osaka Electro-Communication University, Osaka, Japan

Shinjiro Wada, wada@zb3.so-net.ne.jp

Dept of Secretarial Studies, Poole Gakuin College, Osaka, Japan

Masayuki Ueno, ueno.masayuki@gmail.com

Dept of Digital Art and Animation Osaka Electro-Communication University, Osaka, Japan

Noboru Ashida, ashida@fukui-nct.ac.jp

Fukui National College of Technology, Fukui, Japan

Abstract

A new type of Authoring System NeGAS for 3DCG animation is developed in which a user can easily manipulate three dimensional objects in 3DCG space on the computer.

This system is described using an object oriented three dimensional Logo language called o3logo developed by us. The user can manipulate objects comfortably with the help of the turtle metaphor by using o3logo programming, programming palette and hardware interface called "Kame (Turtle Controller)" developed by us. Objects, cameras, lights and turtle in 3DCG space can be manipulated by using turtle commands in our system, so DMI (Direct Manipulation Interface) for 3DCG animation is realized in our NeGAS (Fig. 1).

We have developed a visual action palette on o3logo, in which an user can manipulate them by cricking buttons on the palette instead of programming, to manipulate objects, cameras, lights and turtle in 3DCG space. We extend this visual palette to "o3Art" system by the adding several functions to form three dimensional body in 3DCG space to generate and form artful three dimensional object (Fig.3). About 600 users are using o3logo, o3Art and Negas in several Japanese universities, so we can refine the usability of our system.

This research has been supported by the Japanese national research project called Intelligent Cluster Creation Project.

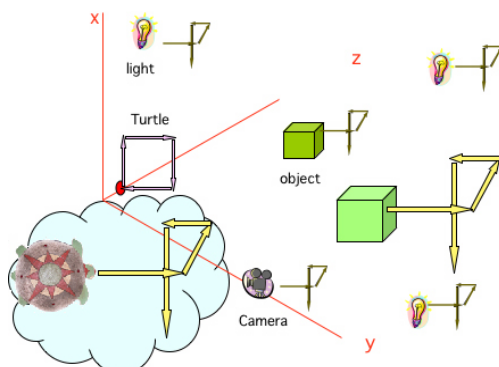


Figure 1. The Turtle Metaphor

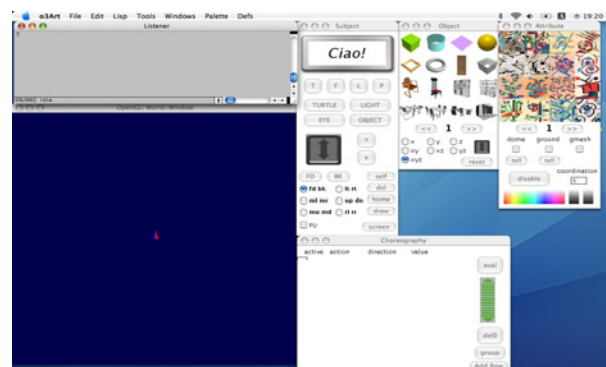


Figure 3. The whole aspect of o3Art system

Keywords

3DCG Authoring System, Object oriented 3D LOGO Language, The Extended Turtle Metaphor

1. Introduction

We propose a new type computer language and authoring system for 3DCG in this paper. There are many software packages for 3DCG creation, but they are different from each other. So, a user cannot obtain a unified concept and skill in their manipulation of these 3DCG software packages. That is one of the reasons why a confusion arise in game and amusement education concerning 3DCG software packages. (Tsushima, 2005) We deeply consider the origin of the defect of the ordinary 3DCG authoring system in Chap.4 and propose an object oriented Logo language o³logo as the base of a 3DCG authoring system called the Next Generation Authoring System (NeGAS) developed by us. (Tsushima et al., 2006)

We want to manipulate objects, cameras, lights by using the unified approach in 3DCG space (Fig.1). The Extended Turtle Metaphor proposed by us in this research may be promising to give a learner a unified approach for 3DCG.

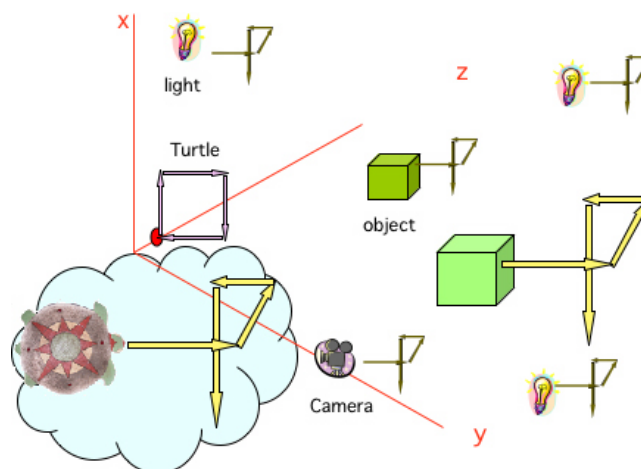


Figure 1. The Turtle Metaphor

At first, we developed an object oriented three dimensional LOGO language called o³logo on Macintosh common Lisp.

In order to create more sophisticated content, script is used frequently. But, in the ordinary 3DCG system the scriptive language does not have a kinematical nature but logical nature, so it is indirect for a learner who wants to move and manipulate objects in 3DCG space. We chose a Graphic Package Open GL for o³logo.

We can manipulate objects, cameras, lights and turtles by using turtle commands in our system, so the unified approach to manipulate them is possible in our o³logo.

As a result, the use of the Direct Manipulation Interface (DMI) for 3DCG creation was realized in our NeGAS. A user can use programming, programming palette to manipulate objects comfortably in 3DCG space with extended turtle metaphor in our o³logo. (Bowman et al., 2004), (Harvey 1997), (Lindsay and Norman, 1977)

This research project has been in progress five years as a national project called Intelligent Cluster Creation Project in Japan. (Tsushima et al., 2007)

2. Development of an object oriented Three dimensional LOGO language: o³logo

Two dimensional LOGO is frequently used in primary and secondary education in the world. It is known as an easy computer language to learn because of intuitive and interactive interface based on turtle movement. (Papert, 1980) Further, we can draw two dimensional figures and

three dimensional bodies on the computer by moving a turtle in 3DCG space using an ordinary three dimensional LOGO language such as 3D-LOGO. (Uni-Bynus 1989)

But we cannot move or manipulate a drawn figure in 3DCG space simulated on the computer. We developed an object oriented three dimensional LOGO language called o3logo (object oriented OECU logo) in which drawn figures and three dimensional bodies can be moved and manipulated like a turtle using turtle commands.

o3logo language was developed by us on LISP language at first. Parser of o3logo developed by us was implemented on CLOS of Macintosh Common LISP. (Tsushima et al., 2005)

The graphic function is very important to raise the capability of drawing and description of a visual event in o3logo. We chose Graphic Package OpenGL for o3logo after our deliberate discussion.

FD, RT, DN, MR, MU and **RR** are essential for turtle movement in three dimensional space in three dimensional LOGO language.(Fig.2)

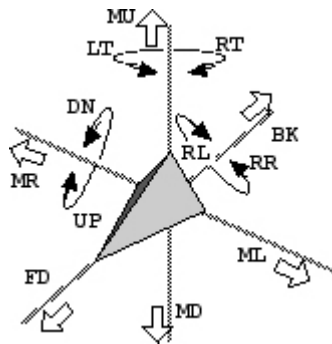


Figure 2. Turtle Commands of o³logo

2.1 creation of object

Any locus drawn by a turtle movement can be defined as an object by using **hold** <object name> command in o3logo. If we want to manipulate the defined object, we use **tell** <object name> command. Then this object can be manipulated by using turtle command.

2.2 grouping of objects

Many objects can be defined as a group by using **make-group** command. Any number of objects can be declared as a group. When this group is switched on using the **tell** command, we can manipulate this group like a turtle by using ordinary turtle commands. Hierarchical definition of group is possible in o3logo. About 120 commands are prepared in o3logo. An outline of commands of o3logo is shown in App.1

We can manipulate turtle, objects, lights and cameras using turtle commands supported by turtle metaphor in our mind. Eventually DMI for three dimensional animation can be obtained.

3. Visual Action Palette

Turtle metaphor strongly helps a user to move and manipulate objects in 3DCG space. On the other hand, we must create the form of a three dimensional body in 3DCG system. This creation is tried using the section prepared by a user or by the system in almost all the 3DCG system.

We want to develop a 3DCG authoring system helped by single metaphor; the extended turtle metaphor, to give our user DMI for 3DCG authoring. A user can produce three dimensional form using not only a section diagram but movement of object.

We realized this easily by using duplicate commands in o3logo programming.

As is well known, programming takes time and some creators do not like programming. So, we developed a palette system called the Visual Action Palette on o3logo. A user can quickly and easily make figures and three dimensional forms without programming but using our Visual Action Palette.

We developed a new palette system called Visual Action Palette on o3logo to form three dimensional bodies in 3DCG space. (Uni-Bynus 1989), (Tsushima et al., 2005) A user can make forms using successive applications of actions instead of programming in this palette system.

The Visual Action Palette has the following four palettes; Object, Choreography, Attributes and Subject (Fig.3)

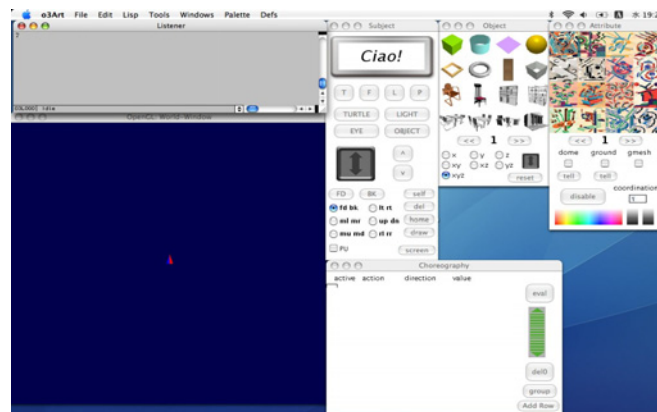


Figure 3. The whole aspect of o3Art system

3.1 Object Palette

There are about 40 objects in the Object Palette which can be used as an object in 3DCG space. (Fig.4) When an object is clicked on Object palette, it is generated on the World-Window as an object. The size and form of each generated object can be changed continuously by using a control arrow with a mouse. The rate of scaling for each direction can be changed continuously with a mouse by selecting appropriate radio buttons. Several 3D characters are prepared to create the content for daily life.

3.2 Attribute Palette

By selecting an appropriate registered pattern in the Attribute Palette, a user can paste it on the surface of the selected object. (Fig.5) As there are several hundred patterns and photos in the Attribute Palette, we can decorate an object and a background in 3DCG space very easily by using the above patterns and photos. And there is a color sub-palette to give selected colors to an object. Pattern and color can be overlapped, so we can change a color of patterns easily by using the color sub-palette.

3.3 Control Palette

We can move objects, lights, cameras and turtles in 3DCG space by manipulating the buttons on the Subject Palette. (Fig.6) When there are many objects in 3DCG space, we can select them by clicking the corresponding upward or downward buttons.

When we click an object button, the present objects are selected in cyclic sequence. We can confirm the present object by clicking a self button, because the figure of the corresponding object blinks in the window.

There are twelve turtle commands to manipulate objects in 3DCG space. We can manipulate an object by clicking the corresponding command button on Control Palette, then the object moves

with a fixed step defined beforehand. We can manipulate an object continuously with an arrow button by moving a mouse.

As quick response is obtained by using these buttons on the Control Palette to manipulate an object, a user can try several ideas on the computer very quickly. As a result, more effective and satisfactory results can be obtained.

We can change the views quickly by clicking T, F, L, P buttons. Therefore we can see the generated results from different view points in 3DCG system. Such a confirmation is frequently used in three dimensional space.

And the present scene from World-Window can be saved in disk by clicking a save button. This figure can be loaded by clicking a load button.



Figure 4. The Object Palette

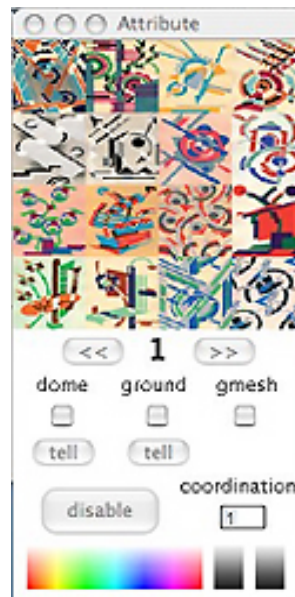


Figure 5. The Attribute Palette



Figure 6. The Control Palette

3.4 Choreography Palette

We can manipulate an object in 3DCG space by using o3logo language. We developed the Control Palette to give a comfortable and quick manipulation method for a user. This palette is comfortable and convenient to move single object in 3DCG space. But, it can not be used in the following manipulation process.

- 1) Parallel complex movements of an object
- 2) Movement accompanied with modification of form, change of object parameter such as color and transparency
- 3) Parallel movement of object, light and camera

At first, we struggled with these problems using only o3logo language.

It takes a long time to create content by trial and error using programming. In these cases, step-by-step confirmation of scene using programming is troublesome. We wanted to prepare a quick and comfortable method for our user.

So, we developed the Choreography palette to improve this situation.

Even an expert of o3logo programming feels this palette is comfortable in his visual creation and strengthen his creation power.

We can manipulate an object according to parallel sequence of manipulation action by varying parameters in the Choreography Palette. In this case, the introduced decorated sphere moves forward and turns right leaving its copy behind. As a result, a bent cylinder is generated in this space. (Fig.7)

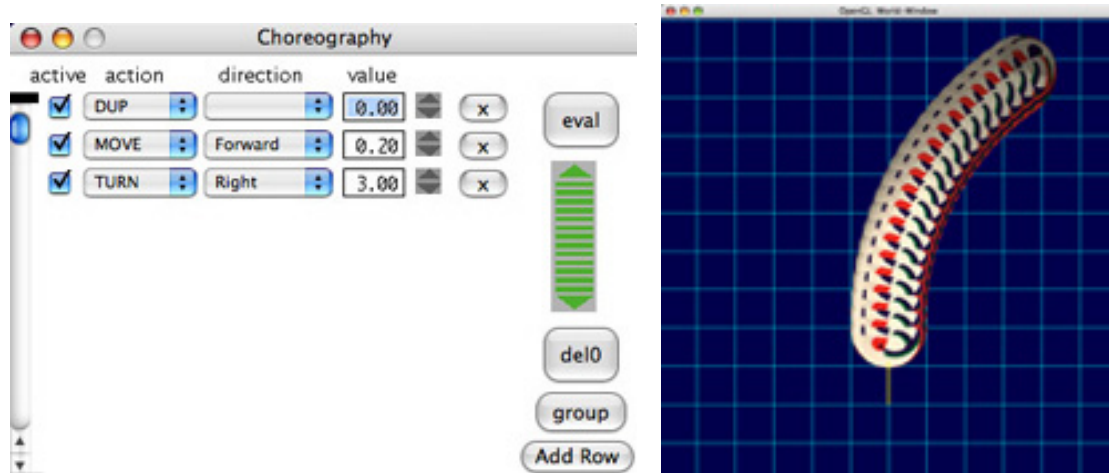


Figure 7. The example of Chereography Pallete

7 actions are prepared and several sub categories can be selected with appropriate value of parameters in the Choreography Palette. (Tab.2) Successive modification is possible by selecting appropriate ratio parameters. The generated body can be treated as a new object when a group button is clicked. Then we manipulate it as another meta-object in this Choreography Palette. So we can produce intuitively a complicated three dimensional body using this hierarchical function. (Fig.8)

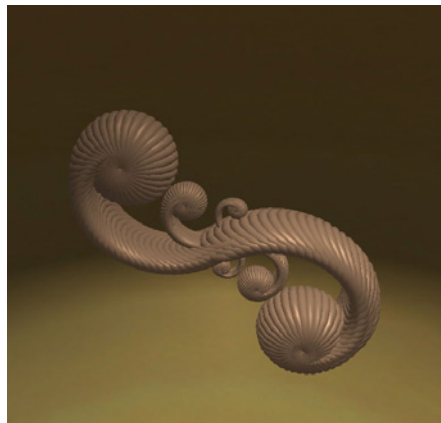


Figure 8. Content

We are aware that this Choreography Palette is really applicable to various visual creations.

At first, we can use a series of commands to move objects. So we added the following several functions to this palette.

- 1) Parallel description capability
- 2) Continuous changing parameter mechanism
- 3) To support movement , modification of the form, color, transparency

4) Simple assignment without commands

We can produce the three dimensional body of an object using this Choreography palette to move and duplicate original objects very easily and quickly. And we can assign systematic movement of objects to a check box without using the commands of o3logo language.

4. Educational Plan for 3DCG Creation using NeGAS

There is appropriate example of DMI for making figure with help of visual metaphor on the computer. It is LOGO interpreter developed by Papert.

The difference between the ordinary LOGO language and o3logo language developed by us is discussed in detail in the following.

Make (draw), see, recognize are important factors in manipulating the turtle to solve the problem in the ordinary 2D LOGO. They may be operators in human mind to solve problems concerning making content. (Tab. 1)

Ordinary 2D LOGO	3D LOGO	objective 3D LOGO
make	make	make
see	see	see
	peep	peep
		lighten
		ditriute
		move
		leave
recognize	recognize	recognize

Table 1. Problem Solving Operator in Humanmind

In 2D space, all the produced figures can be easily seen by a user without thinking of the position of camera (eye), because the camera is fixed in the indefinite point. So, a user can easily recognize what is produced by himself by seeing the figure made on the CRT manipulating command. It is the origin of comfortability of the ordinary 2D LOGO language.

In a 3D LOGO language like 3D-LOGO, the figure produced by a user varies according to the position of the camera. So, a user must give care to the position of the camera in his manipulation. Therefore, harmony between make, see, recognize is destroyed, so peeping at the object is necessary to see the produced figure in the 3D space.

Therefore, a new strategy, such as move and peep the figure in order to recognize the meaning of produced figure.

In LOGO language and 3D-LOGO language, the drawn figure is fixed on the plane and space, and we cannot move it. If a user can move the drawn figure in 3DCG space, he can get an exciting experience in his creation. So, we developed the object oriented three dimensional language called o3logo in which the drawn figure can be moved.

We can manipulate objects, lights and cameras in o3logo language. And we can manipulate 3D turtle to draw figures in it.

A user must consider the effect of light and the position of camera to see the produced object in 3DCG space. And further, a user can move object in 3DCG space. And further, he can distribute

objects using turtle commands and duplicate command in 3DCG space systematically shown in Fig.7 and Fig.8.

So, users have many strategies to make objects and scenes using o3logo. In ordinary 3DCG software package, a user uses various strategies in a haphazard way.

To decrease mental load of a user in his manipulation, we must give a user a mental metaphor which allow him manipulate light, camera, object and drawing turtle in the 3DCG space in a unified approach. So, we extend the applicability of turtle commands to light, camera, object.

We call this concept "extended turtle metaphor". Because this control method drastically reduces the mental load of a user in 3DCG space, we give our user DMI for 3DCG with help of extended turtle metaphor as a result.

5. Conclusion and View

An authoring system for 3DCG called NeGAS is developed. It gives us comfortable manipulation and programming environment based on the movement of a turtle in 3DCG space. The form of three dimensional object can be generated using the movement of basic simple objects with help of turtle commands. So a unified approach is obtained for movement and making the forms of three dimensional objects with help of the extended turtle metaphor.

We developed other manipulation methods using a visual palette and hardware such as controller and electric pen for object in this research. The details are shown in reference 1, 2, 6 and 9. This is a special feature of our NeGAS in comparison with other advanced LOGO system such as FMSLogo.

But, It is not so easy to generate the motion of a object which has many internal freedom of motion even if o3logo programming is used. We developed a physical engine on o3logo and implemented it in NeGAS using coupled differential equations. Refinement of this engine is our future problem.

The extension of the Choreography Palette for many objects will give us the ultimate art tool for 3DCG and animation.

We have tested o3logo and Visual Action Palette in the actual education in our university for several hundreds of students. The student who does not have the knowledge of o3logo language can easily make many 3DCG contents using the Visual Action Palette.

We are encouraged by this fact to propose a national educational plan for 3DCG for students from elementary school to university student.

This research is supported financially by Japanese National Project called Intelligent Cluster Creation project.

Appendix 1. Commands of o³logo

o³logo has many command. The gross structures of command are shown in Table below. command about control of object is important to understand the feature of o³logo, typical example of object control command are shown in Figure 2.

Control	3
Turtle Command	18
Paint	8
Turtle Control	8
Pen Control	8
Object	6
Object Turtle	15
Parallel Execution	2
Grouping of Object	2
GLU Object	9
GL Object	7
Texture	5
Background	9
Material	5
Light Control	2
Animation	3
Time	9
Keyboard	26
etc.	45

Table 2. Commands of o³logo

Appendix 2. Action and subcommand in Choreography Palette

The subcommands of each Action are automatically appeared cyclically by cricking each Action button.

Action	subcommand
dup	
move	forward backward rightturn leftturn upword downword
turn	up down right left
roll	right left
grow	x y z xy yx zx xyz
color	Hue Saturation Brightness

Table 3. Commands in Choreography Palette

References

- Tsushima, K. (2005) Progress towards the Science of Game and Amusement. (Invited) Proc. of Active Media Technology 2005,254-259.
- Tsushima, K., Ueno, M., and Tanida, K. (2006) The Authoring System for 3DCG with help of 3D Turtle Metaphor. Proc. of 9th Generative Art conference 2006, 281-291.
- Bowman, D.A. Kruijff, E. LaViola, J.J. and Poupyrev, I. (2004) 3D User Interfaces. Addison-Wesley Professional.
- Harvey, B. (1997) Computer Science Logo Style 2nd edition. The MIT Press.
- Lindsay, P.H., Norman, D.A., Human Information Processing : An Introduction to Psychology. Academic Press, 1977.
- Tsushima et al. (2007) Report of Proj.7 in Human L-Cube Project, Human L-Cube Tsushima Project, 2007.
- Papert, S. (1980) Mindstorm: Children, Computers and Powerful Ideas. Basic Books, 1980.
- UNI BYNUS. (1989) 3D-LOGO is a product of UNI BYNUS Co. in Japan.
- Tsushima, K. Ueno, M. Nishiki, T. Otsuka, T. and Nomura, N. (2005) Art tool and Objective Three Dimensional LOGO. Proc. of ED-MEDIA 2005. 4409-4415.

Turtles with Roles: an application of an Object Oriented Pattern in JxLogo

Maria Vittoria Bedin, *bedinmar@gmail.com*

Bentec S.p.A. (Benetton Group), Via Villa Minelli 1, 31050 Ponzano Veneto, Treviso, Italy.

Michele Moro, *mike@dei.unipd.it*

Dept of Information Engineering, University of Padova, Via Gradenigo 6/B, 35131 Padova, Italy

Abstract

JxLogo is an undergraduate student project developed at Padua University and entirely realized in Java. It adds an object oriented extension to the Logo language and it integrates in the environment an effective graphical interface and useful tools. The introduction of this extension makes it arise some general questions about its educational motivations, as pointed out in the introduction. Then the paper discusses how the object oriented component can effectively support Logo users of different ages in learning complex concepts. The power of the proposed approach is showed through the introduction of the *Role* abstraction as a form of dynamic inheritance, in contrast with the traditional static inheritance based on derivation between classes. Such an abstraction is introduced through the known Role Object Pattern: two implementations of this pattern are presented, one that uses the *JxLogo* language constructs, and another based on the *TurtleRole* class that is provided at low level. Both levels of implementation permit to add transitory behaviours and attributes to the *Turtle* basic class. Other relevant elements of the environment are also presented, specifically how events are currently supported. Finally some didactic motivations try to answer to the questions aforementioned and to justify the introduction of advanced programming issues as accessible learning options.



Figure 1. The JxLogo online help

Keywords

Logo, JxLogo, Object Oriented, Role Object Pattern, Dynamic Inheritance

Introduction

Logo is a well known learning tool (Papert, 1980) (Papert, 1999) where relatively simple graphical actions (the turtle geometry) coexist with more complex concepts like recursion, interactivity, multithreading. The procedural approach of its language has been proved sufficiently expressive to support pupils during their exploration of new concepts. With respect to the past, recently developed Logo environments pay more attention to the graphical interface adding features such as icons, animated elements, control components etc., but in the Logo history the general architecture and the operative modes have remained more or less the same as the original ones. Actually only in a few cases significant extensions to the language have been introduced to empower its expressiveness: the objective is to allow skilled users to develop projects with new structured, complex abstractions. One of these extensions is the object oriented (OO) component (Boychev, 1999) (Tomcsanyi, 2003) (Kanemune and Kuno, 2005) (Lehotska, 2005) which is the validated paradigm that characterizes most of the widespread general purpose languages such as Smalltalk, C++, C# and Java. This paradigm introduces a relevant number of benefits in terms of software robustness and reliability, software reusing, and a more precise resembling of the real world.

Now some general questions arise: is the OO approach too complex for pupils involved in Logo projects? Does it obscure the clearness and the straightforwardness of the Logo procedural structure? What kind of didactic motivations should the introduction of the OO component in Logo suggest? Finally, at what levels should OO programming be exploited?

As mentioned above, the OO approach can more precisely describe entities expressing characteristics of the real world where 'objects' act according to their properties, with some exposed and some hidden attributes, with some described and some not known behaviours, objects which can be activated through well defined interfaces. In a Logo system the turtle, which is its main entity, may be seen as an object and its model (class) may be used to create different instances in a multiturtle scenario. In this sense a proof consisting in a complete hierarchy of classes describing elementary geometric shapes and their dependencies was proposed in (Moro, 2005). Moreover, the integration of the OO component must be realised preserving the native procedural approach of the Logo language, in order to provide a smooth progression conducting the user to fully exploit the OO features like inheritance and polymorphism.

JxLogo was already presented in previous Eurologo conferences (Moro, 2001): it is an undergraduate student project developed at Padua University and entirely realized in the Java language. Its goal is twofold: it is intended on the one hand, to encourage experimentations and usage of Java technologies by university students, on the other, to implement a complete and innovative environment to support advanced Logo programming. Starting from a strict compliance to MSWLogo for the name and meaning of the basic primitives, JxLogo includes the OO component as an actual extension of the traditional Logo language. The hoped progression in learning complex structures derives from the attention paid during the design of the language and partly from the interpreted nature of the language.

This paper, after a brief description of the main actual aspects of the JxLogo environment, presents some recent improvements that can enforce the justification of introducing a OO programming style in a Logo environment. Then the concept of *Role* is explained as a form of dynamic inheritance which can be used in all those cases when temporary capabilities have to be added to already created objects. JxLogo has sufficient expressiveness to easily implement this concept but some reasons to introduce the same mechanism at a lower level for the important turtle object are also presented. A section deals with events which are other recent empowering elements giving the turtle more reactivity to external stimuli. The last section contains some final remarks and ideas for future work.

The JxLogo environment

In order to give classes a more incremental development, in JxLogo the class construction is divided into two separate steps. In the first one we declare the attributes and the single constructor: the initialization code is quite similar to a usual Logo procedure body. In the second step methods are separately defined and compiled so that they can be added to the class one-by-one in a way similar to the adding of new procedures during a Logo session. Some syntactic extensions have been defined to represent hierarchy aspects, visibility rules, assignments and references to the class attributes and methods (for more details and explanations regarding the syntax that JxLogo uses to manage objects refer to (Moro, 2001) and (Moro, 2005).

The following explanatory example defines a specialised turtle with an overridden `forward` method and the specific method called `square` (the added comments help for an intuitive understanding):

```
class newTurtle [:x 100][:y 100] isa Turtle[:x :y]
  ; [100 100] is the default position of the turtle
  ; :x and :y are passed to the superclass constructor
  ; own attributes: globally visible only for reading
  own [initialX :x
        initialY :y
        distance ^]
  ; set: the keyword to assign values to attributes
  set 'distance sqrt sum :x*:x :y*:y
end

to newTurtle'forward :m    ; forward :m*20 pixels drawing a dash line
  repeat :m [
    old'forward(10) ; it refers to the inherited version of forward
    'penup()
    old'forward(10)
    'pendown()]
end

to newTurtle'square :m    ; old-style square
  repeat 4 [old'forward(:m) 'rt(90)]
end
```

To create an instance of the class the programmer has to call its constructor through the name of the class giving the required arguments enclosed between square brackets (some of them can be optional and therefore defaulted); the reference of the created instance can be assigned to a variable with the usual `make` command:

```
make "t1 newTurtle[150 150]
```

The class construct leads to a static description of derivable data types, whereas the dynamic approach of the Logo language is maintained with the on-the-fly declaration of methods. JxLogo introduces another kind of 'dynamic' extension of a class in form of the extension of an already created object. In the following example:

```
make "t2 (:t1[hidden [step 1]])
  ; t2 is derived from t1 with a new hidden attribute (step)
```

```
to :t2'setStep :s      ; a method defined on t2
  set 'step :s
end

to :t2'forward :m      ; another method defined on t2
  old'forward(:m * 'step)
end
```

an anonymous class is derived from `newTurtle`, the class of the `t1` instance, with the new hidden attribute `step` (visible only within the new class) and a single instance of this new class is created and represented by the `t2` variable. The state of the inherited attributes is copied from the original `t1` object to the new `t2` object. It is also possible to use a similar construct directly referring to the superclass:

```
make "t2 (newTurtle[200][hidden [step 1]])
```

In this case the declaration can include some or all the constructing arguments.

To transform JxLogo into a suitable Logo environment, able to support and help the user during the entire work session, two kinds of editing tools have been defined together with a online help system. In fact JxLogo recognizes two different types of users:

- children and very young students using Logo as a learning tool, who need an easy, handy environment suitable to their limited abilities and focused interests;
- advanced older users ready to face the object oriented programming challenge, in order to experiment relative complex projects and to improve their programming skills.

For the first type of users an elementary editor to code simple Logo procedures is almost enough, together with a command line where it is possible to invoke a list of instructions. The second type of users are invited to take advantage of the full potentiality of the system that includes an integrated editor/debugger to code and test procedures, classes and methods.

Currently JxLogo provides a total of more than 250 primitives and several built-in classes whose interface and semantics must be known by the user for an effective programming. The need of a complete online documentation is therefore mandatory. We decided to implement an ad-hoc tool, using the Sun Microsystem JavaHelp: an XML structure for the help files has been defined in order to separate the content management from the contents themselves (Figure 1).

Dealing with Roles

In object oriented programming abstractions like specialization and adaptation may be easily implemented using the class derivation paradigm. This is a static declarative construct by which hierarchically organized models may be defined and instantiated. In this scenario each object is univocally associated with one generating class during all its life.

Basically Logo tends to encourage users to incrementally define new elements (variables and procedures) that can be coded and tested on-the-fly exploiting the interpreted nature of the language. In JxLogo this characteristic was improved when the object oriented component was defined. In fact, even if the class header includes data elements and initializing code, we saw that methods can be incrementally added to the class. When a new method is defined, a new class version is compiled and stored in memory: thus objects created before this modification are associated with the old version of the class whereas objects created hereafter refer to the new one. Even when a new anonymous class is created and compiled, the object 'extending' the old one is an instance of the new class. These three approaches to extend a class (usual class derivation, method adding, anonymous class) are more or less static constructs which lead to permanent models which do not allow removing of previously added components and maintain the strict relation between an object and its class.

Notwithstanding, there are situations when a greater dynamism is required: new components and behaviours should be temporarily added to an object in order to satisfy specific runtime requirements. Well-known and immediately understandable examples are all those situations in which an object describes the attributes of a person (or an equivalent entity) who enters an environment where new capabilities are assigned and/or required. These capabilities represent the concept of *Role* and specific architectures are proposed in literature to support effectively this concept (Fowler, 1997) (Baumer, 1997) (Kendall, 2000). A role summarizes capabilities used temporarily and under conditions, and it must express a kind of dynamic inheritance, i.e. a part or all the behaviours of the original object can be assumed by the object even when it plays one or more specific roles. New or modified behaviours may be related to its current active roles and, generally speaking, they could depend on the context in which they operate.

Roles implemented in the JxLogo language

In order to show that JxLogo is sufficiently expressive to support one of the most flexible patterns for roles, the so called *Role Object Pattern*, proposed by Fowler, we present a demonstrative example referring to a turtle object which has well-known basic behaviours in Logo. In this pattern the original object (and its class) acts as a *core object* which includes some general functions to manage roles defined as separate classes. A core object may be dynamically linked to a set of separate role objects: each one, as explained above, describes specific capabilities. To implement the required form of inheritance, all the functions, which are available in the core object and are chosen as functions to be inherited by its possible roles, must be applicable to each role object. This is obtained applying a simple delegation, i.e. a role class must implement each 'inherited' method calling the homonymous method in the core object, apart from a possibly added specific code.

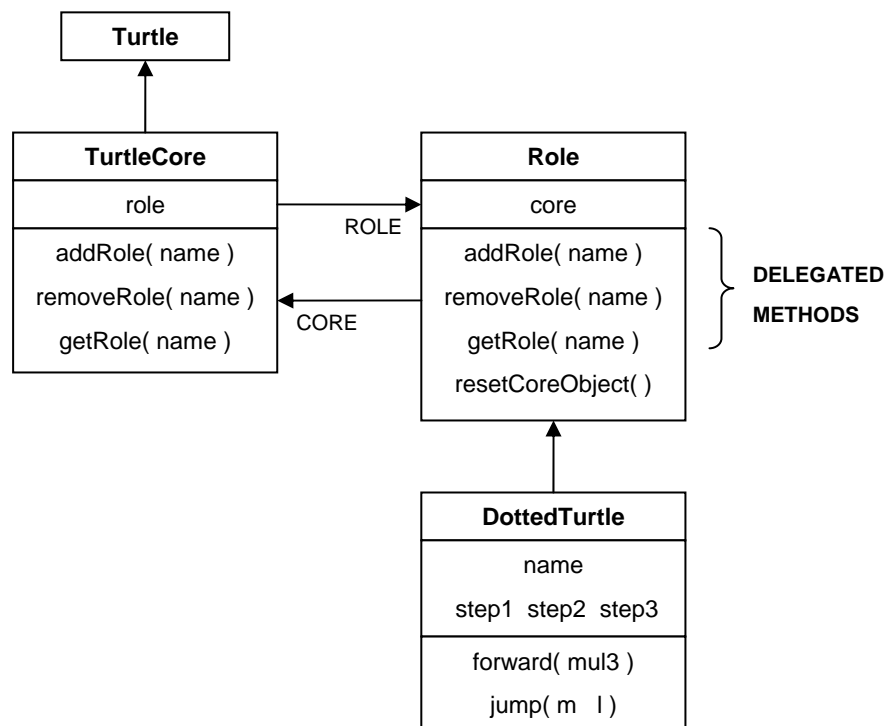


Figure 2. The role example structure

Considering that in a Logo environment the turtle is the main graphical component with proper pre-defined functions, it is reasonable to think to transforming a turtle object into a core object and to implement some special movement behaviours as instances of different classes. More in detail, in the following example the class `TurtleCore` extends the base built-in `Turtle` class,

and defines itself as a *core class* including three general managing methods for roles (`addRole`, `getRole` and `removeRole`) of straightforward meaning (Figure 2). Without loss of generality, instead of a list of roles, the class includes a reference to a single role object. Moreover, for the sake of simplicity, the `addRole` method manages directly all the allowable roles instead of relying on a suitable data structure where to register them in a configuration phase (notice that classname symbols, like `DottedTurtle`, may not be passed as parameters because they must be translated at compile time, thus the role name is provided to the method as a string parameter). Delegated methods in each *role class* should be the same three managing methods (e.g. adding on a new role to a previous role means adding it to the associated core object) plus other ‘inherited’ methods, in this example, the `forward` method. Considering that the role managing methods are completely general, they are included in a base `Role` class which must be extended by all the specific role classes. The `DottedTurtle` class defines a different `forward` method drawing a dotted (dash) line instead of the usual straight line.

```
class TurtleCore :x :y isa Turtle[:x :y] ; compulsory coordinates
  own [role "null"] ; the 'list' of roles (actually at most one role)
  (show "The "current "role "is 'role)
end
to TurtleCore'addRole :name ; the role name
  if (equalp "DottedTurtle :name)      ; code to recognize every assumable role
    [set 'role DottedTurtle['this]]
    . . . ; similar for other roles
end
to TurtleCore'removeRole
  'role'resetCoreObject() ; if necessary
  set 'role "null ; remove association with the role
end
to TurtleCore'getRole
  output 'role
end
```

To support the delegation to the associated core object, the `Role` class contains a reference to the core object, which is cleared when the role is removed.

```
class Role :core ; the associated core object
  hidden [corRef :core]
end

      ; delegated methods
to Role'addRole :name
  'corRef'addRole(:name)
end
to Role'removeRole
  'corRef'removeRole()
```

```

end
to Role'getRole
  'corRef'getRole()
end
to Role'getCore
  output 'corRef
end

      ; specific methods
to Role'resetCoreObject
  set 'corRef "null
end

class DottedTurtle :core isa Role[:core]      ; a specific role
  own [name "DottedTurtle
    step1 10
    step2 5
    step3 2]
  (show "Added "the "role 'name)
end

      ; delegated methods
to DottedTurtle'forward :mul3 ; triplicate the step?
  if (not equalp :mul3 0)[set 'step3 3*'step3]
  repeat 10 [ ; ten separated segments of step3 units
    'getCore()'forward('step3)
    'getCore()'penup()
    'getCore()'forward('step3)
    'getCore()'pendown()]
end

      ; specific methods
to DottedTurtle'jump :m :l ; new position
  'getCore ()'penup()
  'getCore ()'setx(:m)
  'getCore ()'sety(:l)
  'getCore ()'pendown()
end

```

The following list of instructions:

```
make "t1 TurtleCore[10 20] ; a new turtle in [10 20]
```



```
. . .
:t1'addRole("DottedTurtle) ; add a 'dotted' behaviour to t1
make "r1 :t1'getRole()
:r1'forward(0) ; dotted line with step = 2
:r1'forward(1) ; dotted line with step = 6
:r1'jump(40 -100) ; jump to a new position
:r1'removeRole() ; the role is unlink from its core
:r1'forward(0) ; run-time error: the delegated method is no longer applicable
```

creates a TurtleCore instance and, in different moments, add a role, use that role for delegated and specific activities, and eventually remove it from the core object.

The example shows that the role becomes a sort of second skin that empowers the original object during the period in which the role is active: once the object gives up the role, it gives up everything new or different the role represents. The method delegation is the price to pay to 'simulate' the inheritance of the 'superclass' (core) methods.

Roles implemented within JxLogo

Other Logo environments allow a form of dynamic association of specific behaviours with the fundamental turtle object (Tomcsanyi, 2003). The role pattern reveals a greater power because several roles can be added to an object and singularly referenced by means of a unique identification (e.g. the role class name). Therefore, even if the JxLogo language provides the sufficient expressive power to implement the Role Object Pattern on generic objects, there has been the convenience of implementing a built-in structure to effectively define roles for a turtle. Such a structure is based on the availability in the system of the built-in TurtleRole class which includes all the role managing methods, present also in the built-in Turtle class, and the basic turtle methods (forward, back, penup, etc.). The user can easily derive its own role classes from TurtleRole specifying new turtle behaviours and/or modifying its default behaviours. This structure seems to positively support both multiturtle interactions in complex systems and the typical enrichment of a turtle 'agent' when the mobility from one machine to another will be provided.

Named again DottedTurtle the role class that defines the new version of the forward method for drawing dotted lines, the following instructions, using the built-in class, create a turtle, add this kind of role, use it and then remove it (Figure 3). Notice that, when a first instance of the class derived from TurtleRole is created, the system registers a 'prototype' that can be successively used to add this role to core objects by means of its symbolic name string.

```
class DottedTurtle isa TurtleRole[]
. . . ; same definitions as in the previous example
. . .
make "t1 Turtle[100 100]
:t1'forward(90)
:t1'right(90)
make "r1 DottedTurtle[] ; a DottedTurtle prototype is registered
:t1'addRole("DottedTurtle) ; the role is added
make "r1 :t1'getRole("DottedTurtle) ; set a reference to the role
:r1'forward(90) ; 'dotted' forward
:r1'left(90) ; equivalent to :t1'getRole("DottedTurtle)'lt(90)
:r1'forward(90)
```

```
:r1 'removeRole( "DottedTurtle")
:t1 'left(90)
:t1 'forward(90) ; normal forward
```

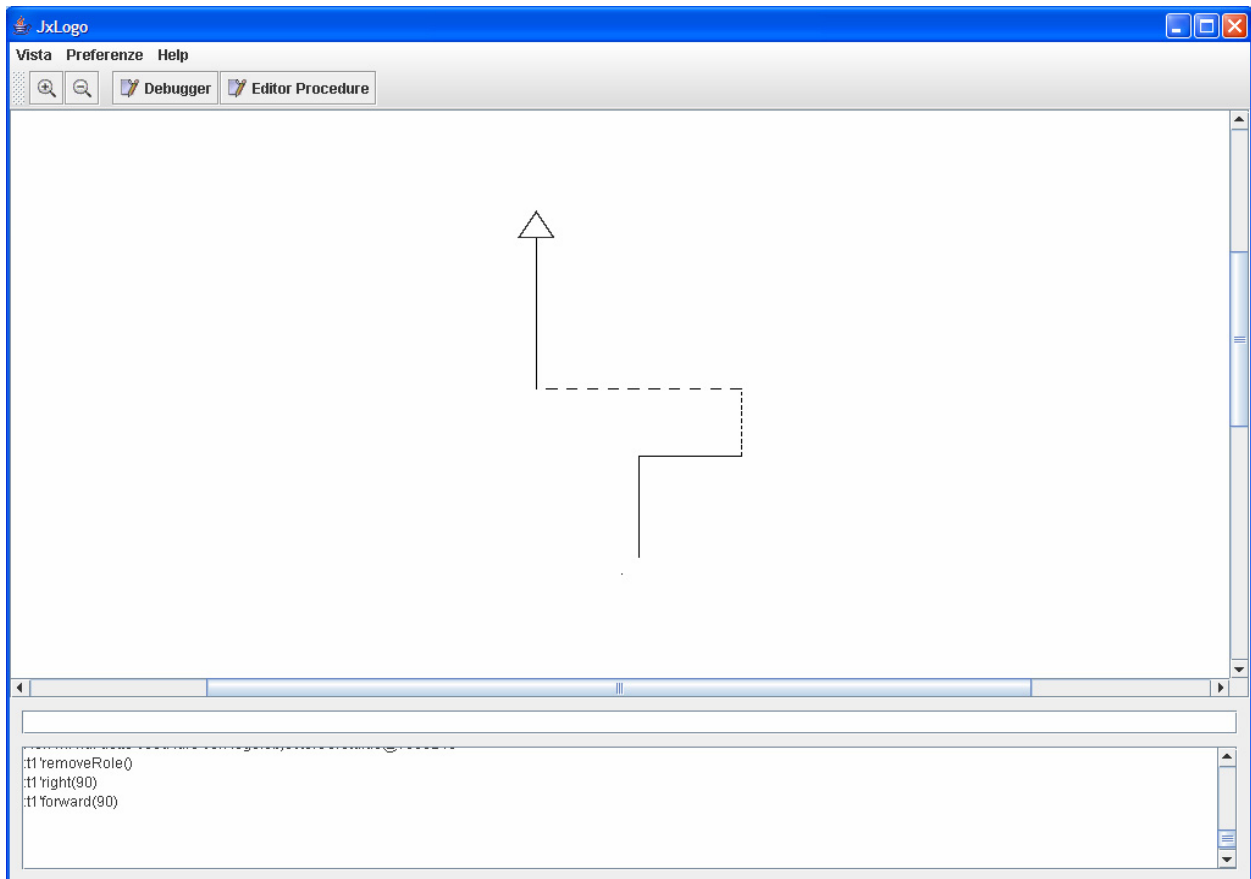


Figure 3. Use of a role: during the section in which the turtle is associated with the 'dotted' role, its behaviour is specific.

Events

In JxLogo the graphical interface is a relevant key aspect of the environment: the major part of the user's attention is devoted to graphical objects and interactions. The turtle is the most important active element with a lot of commands related to the drawing facilities of the system. The essence of the turtle as an active object is guaranteed by the programmability of its preordered actions, whereas the responsiveness to asynchronous events (the *reactive* turtle) may be delegated to the definition of *callback* actions to be performed when the user interacts with the graphical environment during the execution of the program. A third vision of the turtle as a *proactive* entity is connected with the notion of agent and it will be subject for future studies. For example, in the realization of a game application, the user could either point and click with the mouse on one of the defined turtles or press an arrow key on the keyboard to force special behaviours while those turtles are moving and drawing within their operational area. These considerations led to the introduction of *event management* that, together with the role support, provides a powerful tool for behavioural control.

The management of events is limited to the turtle because this is the only recognizable object for which the meaning of events is immediately understandable. It is realized using the low-level Java events: the turtle becomes the source of the event, the callback function is implemented as a method of a dedicated object (listener). Such a method has a predefined interface: when an event takes place, only if the specific callback has already been registered within the graphical

object that represents the turtle, the foreseen reaction is executed; otherwise nothing happens. The declaration of a callback function is actually done in the form of a list of instructions that is to be executed when the specified event is received by the associated turtle. For example, if `t1` is a turtle instance, after the following command is executed:

```
:t1'setEvent("MouseClicked  
    [t1'penup() :t1'fd(40) t1'pendown()])
```

any time the user clicks with the mouse on the graphical representation of that turtle it jumps 40 units ahead.

The list of the event names currently registered within a certain turtle can be obtained with the built-in Turtle method `events()`. Another Turtle method, called `sendEvent("eventName")`, may be used on the command line to test the action registered for the *eventName* event.

The association of the callback function is not permanent and it may be successively removed or redefined:

```
:t1'setEvent("MouseClicked [t1'rt(45)])  
.  
.  
.  
:t1'removeEvent("MouseClicked")
```

Reasonably the model enables the user to operate with predefined events (some of the events supported by Java) but a similar approach may be easily extended to user-defined events seen as a form of very simple asynchronous interactions between turtles. In other words, usually a method applied to a turtle is executed by the only thread which has the responsibility to control that turtle on the basis of the defined program. Instead the `sendEvent()` method may be executed by a different thread that wants to force the controlling thread to execute the associated callback action. Accordingly *sendEvent* may be used in the command line to test also actions for user-defined events.

Discussion and final remarks

The objectives reached by the current version of the JxLogo environment and the experimentations carried out until now reveal some agreeable answers to the questions posed in the introduction of the paper. The complexity is maintained under control proposing simplified declarative structures but at the same time allowing users to exploit all the advanced features of the OO programming. The teacher can choose at what level making the experimentations (basic Logo, anonymous extensions, traditional class inheritance). The basic Logo level is in any case available with some facilities regarding the compatibility with other systems and a national language support. Therefore it is the starting point to explore more advanced concepts under both programming and application point of view. The OO programming style may improve the correspondence between the application and the real world (or the real personal experience) through a progressive learning path that guides young users to a more professional approach to programming. Consequently its level depends on the user's age, on the kind of projects, and on the teacher's didactic aims.

Roles underline the usefulness of the OO component in a Logo environment even in the case of a complex goal like defining data types in a dynamic hierarchical structure. At the same time the built-in support of roles for turtles allows a user to manage 'dynamic classes' without been bored with technicalities regarding the role pattern. Moreover roles represent a substrate to support an agent-based view of the turtle where interaction-related issues are developed within roles whereas algorithmic-related actions are included in the core objects (Cabri et al., 2005).

Future developments should therefore regard the transformation of turtles from simple drawing atoms to movable, proactive entities able to interact in a controlled way both with other turtles in the same machine and with other turtles or other Logo objects in different machines where the original turtle has been moved. In this sense some experimentations have already been carried

out in terms of multiturtle interactions and of using standard protocols to support the turtle migration, which may be improved in the near future. The JxLogo experience has also been offered as a experimental platform for educational robotics in the scope of the recently started TERECoP (Teacher Education on Robotics-Enhanced Constructivist Pedagogical Methods) project, a Socrates Programme, Comenius 2.1 Action (Training of School Education Staff).

References

- Papert, S. (1980) *Mindstorm, Children, Computers and Powerful Ideas*. Basic Books, New York.
- Bäumer, D., Riehle, D., Sibersky, W., Wulf, M.: *Role Object*. In: Proceedings of the 4th Annual Conference on the Pattern Languages of Programs, Monticello, Illinois (1997).
- Fowler, M. (1997) *Dealing with Roles*. In Collected papers from the PLoP '97 and EuroPLoP '97 Conference, Technical Report wucs-97-34, Washington University Department of Computer Science (<http://st-www.cs.uiuc.edu/users/hanmer/PLoP-97/>).
- Baumer, D. and Riehle, D. and Siberski, W. and Wolf, M. (1997) *Role Object*. In Proceedings of the 4th Annual Conference on the Pattern Languages of Programs, Monticello, Illinois.
- Boychev, P. (1999) *Elica Logo and Objects*. In Proceedings of the 7th European Logo conference, Sofia, pp. 160 - 168.
- Papert, S. (1999) *Introduction: What is Logo? And Who Needs It?* In Logo Philosophy and Implementation, LCSl, Montreal, pp. V-XVI.
- Kendall, E. A. (2000) *Role Modelling for Agent Systems Analysis, Design and Implementation*. IEEE Concurrency, 8(2), pp. 34 - 41
- Moro, M. (2001) *JxLogo: A new Integrated Java-based Programming Environment for Logo*. In: Proceedings of the 8th European Logo conference. Edited by Futschek G., Linz, pp. 209 - 218.
- Tomcsanyi, P. (2003) *Implementing object dependencies in Imagine Logo*. In Proceedings of the 9th European Logo conference. Edited by Cnotinfor, Porto, pp. 127 - 140.
- Lehotska, D. (2005) *Advanced programming classes in Imagine*. In Proceedings of the 10th European Logo conference. Edited by Gregorczyk G. and oth., Warsaw, pp. 154 – 163.
- Kanemune, S. and Kuno, Y. (2005) *Dolittle: an object-oriented language for K12 education*. In Proceedings of the 10th European Logo conference. Edited by Gregorczyk G. and oth., Warsaw, pp. 144 - 153.
- Cabri, G., Ferrari, L., Leonardi, L. (2005) *Injecting roles in Java agents through runtime bytecode manipulation*. IBM System Journal, Vol. 44, no. 1, pp. 185 - 208.
- Moro, M. (2005) *Object Oriented programming and development in JxLogo*. In Proceedings of the 10th European Logo conference. Edited by Gregorczyk G. and oth., Warsaw, pp. 132 - 143.

Dependency by definition in Imagine-d Logo: applications and implications

Chris Roe, croe@dcs.warwick.ac.uk

Centre for New Technologies Research in Education, The University of Warwick, Coventry, UK

Meurig Beynon, wmb@dcs.warwick.ac.uk

Dept of Computer Science, The University of Warwick, Coventry, UK

Abstract

Dependency is a concept whose importance for learning is strongly suggested by – amongst other things – the wide range of applications that spreadsheets have found in education. Given the prominent role that Logo has played in research into computer support for constructionist learning, there is a natural motivation for studying the relationship between spreadsheets and Logo programming. The technical issues surrounding the implementation of dependency in Imagine Logo (Kalas & Blaho, 2000) have been addressed in previous work of Peter Tomcsanyi (2003). Tomcsanyi's research gives useful insight to the Imagine Logo programmer, but does not indicate how dependency might be exploited by end-users the high-level way that it has been in spreadsheets. This paper introduces a prototype extension of Imagine Logo, informally called **Imagine-d Logo**, that allows dependencies to be specified using spreadsheet-style definitions.

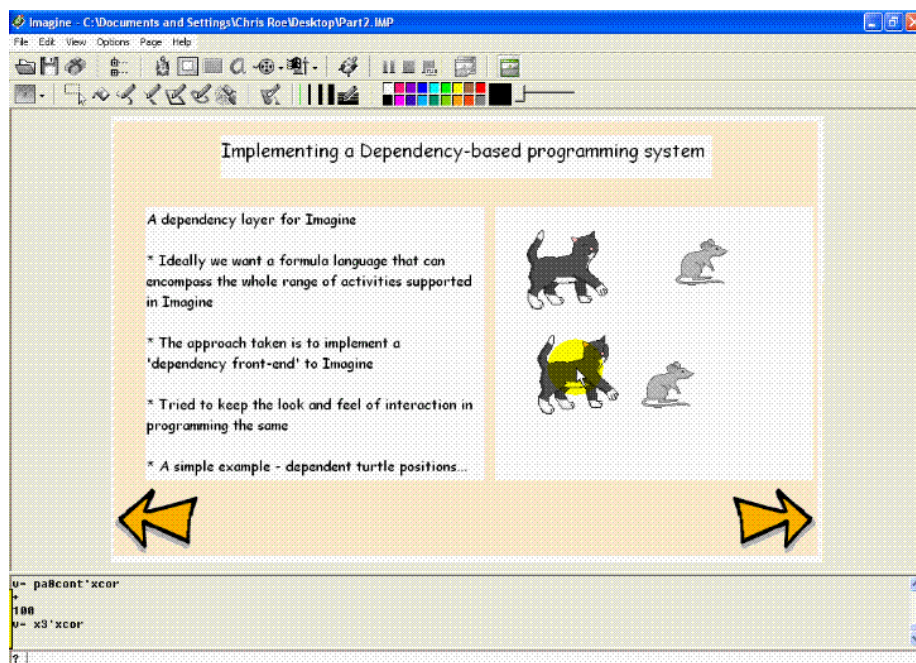


Figure 1. A slide from a presentation developed using Roe's **Imagine-d Logo** prototype

The current scope and potential for Imagine-d Logo are discussed and illustrated with reference to practical issues encountered in using Imagine Logo to program a microworld, to implement spreadsheets and to give computer support for presentations. The broader implications of introducing *dependency by definition* into a Logo environment in this manner are briefly discussed from the perspectives of education and computing.

Keywords

Imagine Logo, dependency, spreadsheets, Empirical Modelling, programming paradigms

Introduction

Spreadsheets and Logo – in all its rich and numerous variants – have been amongst the most influential applications where computing support for education is concerned. By way of background, Baker and Sugden (2003) describes the wide range of educational applications for spreadsheets, and *What is Logo?* (Logo Foundation) gives a good account of the impact of Logo on education. It is natural to seek to integrate these two approaches to educational technology. Such an integration raises challenging issues from several different perspectives:

- **Practical programming:** What motivating advantages might there be in introducing spreadsheet principles into Logo and how is this best accomplished technically?
- **Pedagogy:** What is the significance of spreadsheets and Logo as vehicles for learning, and what is the relationship between them?
- **Computing science:** How can we make unified sense of the two different computational paradigms represented in spreadsheets and Logo?

Our initial and primary emphasis here is on the practical programming perspective. We introduce a prototype extension of Imagine Logo, developed by the first author, that was motivated by issues encountered in programming microworlds based on the throwing events of the Olympic Games (Roe et al, 2005). This prototype – to be referred to as *Imagine-d Logo* – builds on previous work by Tomcsanyi (2003), but more closely resembles Geomland (Sendov & Dicheva, 1988) in its exploitation of dependency. Another implicit influence stems from both authors' work on the principles and tools of Empirical Modelling [EM], a general approach to computing that can be viewed as rooted upon a methodology for modelling with dependency (EM website; Beynon, 2007). The later sections of the paper discuss the Imagine-d Logo prototype from the broader pedagogical and computational perspectives associated with EM thinking, and conclude by considering the potential implications for the future development of Logo-like languages.

Adding spreadsheet-style capabilities to Imagine Logo

The spreadsheet is the archetypal example of an application that exploits *dependency maintenance*, the mechanism whereby changing one value propagates changes to other values in a predictable way. The idea of dependency is often informally invoked by programmers in a much more general context. In implementing object dependencies in Imagine Logo, Tomcsanyi (2003) is concerned with quite general situations in which an object's attributes depend on the attributes of another.

Tomcsanyi's approach to implementing dependencies within an object-oriented programming framework is similar in spirit to that of Perry (2001). Dependencies are introduced by the programmer through embellishing the underlying classes in ways that demand considerable technical skill. By contrast, making computer models using spreadsheets is an activity that can be carried out by non-specialists, and can be viewed as a form of end-user programming (see e.g. Nardi, 1993). In order to introduce dependency into Imagine Logo in a way that gives similar scope for the non-expert programmer, certain key requirements must be met. The programmer:

- should not be concerned with the way in which the underlying system maintains the dependencies between objects in the program;
- should be able to ascertain the definition of an attribute/object/variable by inspection.

To achieve the kind of expressive power that Tomcsanyi (2003) realises in his implementation of object-dependencies, it is also desirable to support a wider range of dependency relations than is supported by a traditional spreadsheet. For instance, whereas the spreadsheet is best suited to maintaining scalar relationships recorded in a tabular format, it is natural to consider maintaining relationships between geometric objects in the Imagine Logo environment. By way of illustration, Figure 2 displays what Tomcsanyi (2003) identifies as "a kind of canonical construction of dynamic geometry" – a triangle together with the common point of intersection of the perpendiculars dropped from its vertices onto the opposite side.

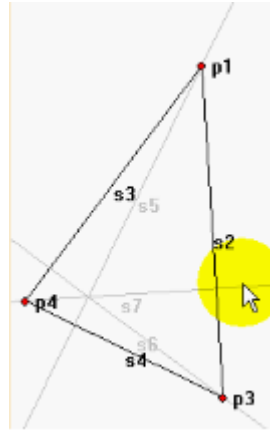


Figure 2. A simple dynamic geometry construction

In this construction, the geometric relationships are maintained by dependency when the position of any of the red vertices p1, p3 and p4 is changed.

Neither of the above requirements is met in Tomcsanyi's implementation of dependency in Imagine Logo. The Imagine-d Logo prototype addresses these requirements by making it possible to specify and revise the dependencies between attributes of objects in much the same way that this is done in a spreadsheet: by formulating and editing explicit definitions for attributes. This has the advantage of making dependency relationships much easier to understand and to trace.

From a technical programming perspective, Roe's extension of Imagine Logo is similar to that described by Tomcsanyi (2003). The same underlying principles are used to maintain dependency, but the details of the implementation are hidden from the programmer. This potentially means that end-user programmers can make effective use of dependency. It also promotes qualities that can be found in modelling with spreadsheets, making models simpler to specify, quicker to build, and more readily open to extension.

A dependency layer for Imagine Logo

The Imagine-d Logo prototype was developed without access to the Imagine Logo source code: it accordingly adds a dependency layer at the dialogue-box level to Imagine Logo. This layer supplies a 'front-end' in which attributes can be given values using definitions that can be specified and edited dynamically in much the same way as the cells of spreadsheet. This involves developing a language within which to specify dependencies between many different kinds of attributes – ideally, a language sufficiently rich to support all aspects of programming activity afforded by Imagine Logo. A further important consideration is that the extension of the interface should respect the look and feel of the Imagine Logo programming environment.

The full technical details of the implementation are beyond the scope of this paper. The features of the prototype will be motivated and illustrated with reference to practical programming issues that were encountered in the process of building the Olympic Games microworlds.

A simple example of the kind of dependency to be specified is that between the positions of two turtles that is discussed in Tomcsanyi (2003). The relationship between the position of the cat and the mouse in Figure 1 illustrates the general idea. In the pane on the right-hand side of this figure dragging movements of the cat result in left-right movements of the mouse that maintain a fixed horizontal distance between the two. The way in which the model-builder specifies this dependency differs between the top and bottom cat-mouse pairs, though the effect is the same.

The specification of the dependency between the positions of the cat and the mouse in the top cat-mouse pair in Figure 1 typifies the way in which a relationship between attributes is expressed in an Imagine Logo program. As displayed in Figure 3, the specification takes the

form of a triggered action – located in the Events field – that is invoked whenever the cat (here implemented by the turtle x1) is dragged about the screen. The effect of this triggered action is to update the x coordinate of the mouse (here represented by the turtle x2).

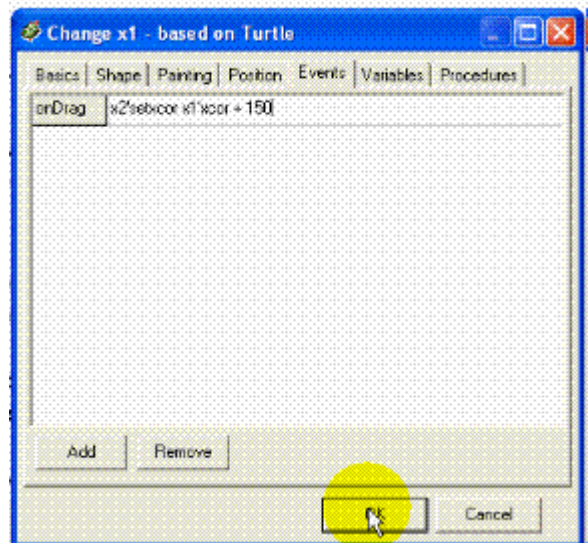


Figure 3. A triggered action associated with “dragging the cat”

In contrast, the specification of the dependency between the positions of the cat and the mouse in the bottom cat-mouse pair in Figure 1 exploits a spreadsheet-style definition feature that is implemented in Imagine-d Logo. For this purpose, the Basics field in the specification for the mouse at the bottom right (here implemented by the turtle x4) is adapted – without any significant change to its layout and format – as displayed in Figure 4. The x and y coordinates of the turtle x4 can be specified in this context not only as explicit values but *by definitions*, so that the x coordinate of turtle x4 is defined to be the x coordinate of turtle x3 incremented by a constant number of pixels. This definition, like a spreadsheet definition, can be directly edited by the model-builder. Figure 1 shows the result of changing the value of the constant increment from 150 to 100 pixels.

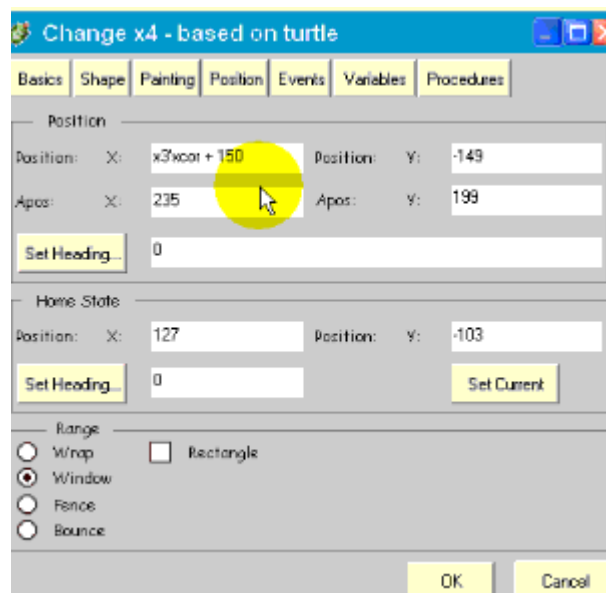


Figure 4. Defining the x position of the mouse via the modified interface

The Imagine-d Logo prototype introduces a conceptually new feature into the programming environment. Where the event-driven paradigm of Imagine Logo requires that relationships are specified by means of triggered actions, the definition of an attribute does not readily admit a simple interpretation as a state-changing action. The implications of this are complex and subtle. It is significant for instance, that defining the x coordinate of turtle x4 with reference to the coordinate of turtle x3 ensures that the value of x4'xcor changes whenever the value of x3'xcor changes, no matter what the nature of the agency that is operating. This contrasts with the discriminating way in which a position update may be associated with a specific event, as in dragging the cat around the screen (cf. Figure 3).

In general, it may be said that definitions afford clearer and more explicit descriptions of relationships though – a superficial analysis suggests – at the expense of some operational simplification. Event-driven sequences of actions can support cyclic patterns of update for instance, whereas the value of an attribute cannot be meaningfully defined in terms of itself. The potential benefits of expressing a dependency through a definition are best appreciated where more complicated patterns of dependency are present. Where a value is defined in terms of many different values, each of which is subject to change through a range of different agencies, the task of analysing the implicit relationships amongst families of attributes can become exceedingly complex, and specifying and maintaining the program structure becomes accordingly more difficult.

Further illustrations

Some additional examples serve to illustrate some of the key issues regarding the introduction of spreadsheet-style definitions to the Imagine Logo environment.

The distinction between the event-driven and definition-based approaches to specifying dependencies is relevant even to simple scenarios, such as maintaining the relationship between the value in a box and the position of a slider (cf. Figure 5). The principal advantage of using a definition is that the model-builder need only inspect the specification of the box object in order to determine why it has its current value. This may not pose a serious problem where the dependency that determines the value in the box relates to the position of a single slider and is kept up to date by a single triggered action. It can become much more problematic when the dependency relations are complex.

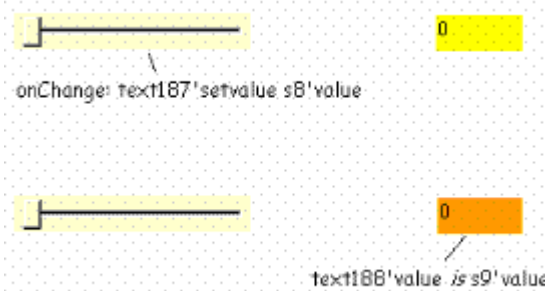


Figure 5. Alternative ways to link the value of a box to a slider

Figure 6 shows a control mechanism drawn from the Shotput microworld (Roe et al, 2005). In this figure, the angle of elevation of the green arrow is determined by the lower slider, and the angle of elevation of the purple arrow relative to the green arrow by the upper slider. The textbox values and the headings of the turtles are then dependent on a combination of the positions of the two sliders. Within the Imagine-d Logo prototype, the heading for the purple arrow can be defined directly by the formula “angleslider'value + attackslider'value”. When the dependency is implemented in the traditional way, this formula appears on the right hand side of assignments in two separate triggering events associated with the two sliders. To understand how the heading

of the purple arrow is determined the model-builder has to search for the events that might have an impact upon them. If it becomes necessary to update the relationship, this formula has to be updated in identical ways in two places.

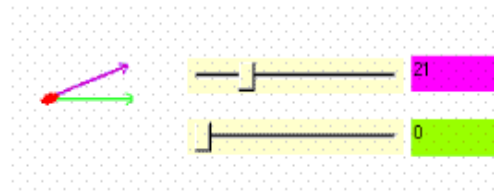


Figure 6. A more complex dependency relationship involving two sliders

Another feature of the Shotput microworld in which complex dependencies are involved is the dynamic graph used to display the x and y coordinates of the shotput trajectory. Because of the wide variety of possible trajectories, it is important to be able to adjust the range of values displayed on the axes, and the granularity of the xy-grid dynamically (see Figure 7). This involves setting up dependencies in which multiple turtles and several textboxes are involved. In this example, dragging the purple point triggers the recalculation of the maximum x and y coordinates that need to be displayed. These positions of the grid lines are dependent on these maximum values, and grid lines are only displayed if they are within the range of the graph. The dependency relationships required to specify the positions and visibility of grid lines are here expressed by definitions.

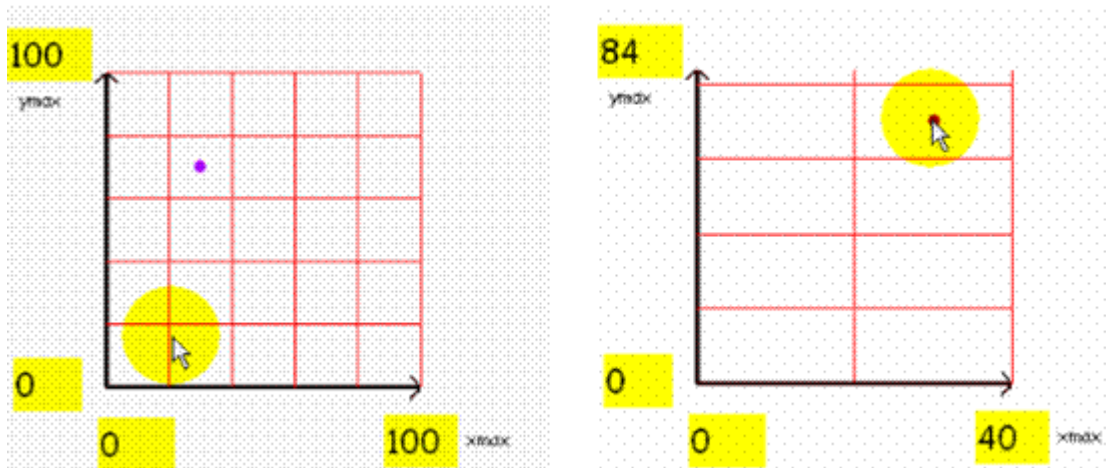


Figure 7. Dynamic graphs implemented using dependencies expressed by definitions

A very rich family of relationships can be programmed by introducing generic dependency maintenance to the Imagine Logo environment. This is evident from the range of applications that can be readily implemented using the Imagine-d Logo prototype. Figure 8 displays another slide from the same presentation from which Figure 1 was drawn. Dependencies framed as definitions have been used in this presentation in several ways:

- To implement a generalised spreadsheet: A simple spreadsheet has been embedded within the displayed slide. In addition to the normal dependencies between values, there is scope to make the presentation of data depend on its content, to exploit slider positions as if they were cell values and to attach names to "floating cells" that do not have to be located within the grid.
- To manage the screen layout: The dimensions, organisation and attributes of windows on the screen can be determined using dependency relationships. The position of bullets in the text can be determined relative to a reference point.
- To animate the presentation: By dragging the dog icon at the foot of the slide from left to right across the screen different right hand panes can be slotted into the slide to illustrate

each of the bullet points in turn. This animation effect is obtained by creating a series of frames that display a different variant of the dog icon according to its horizontal position, and defining the choice and location of the right hand pane in terms of this position.

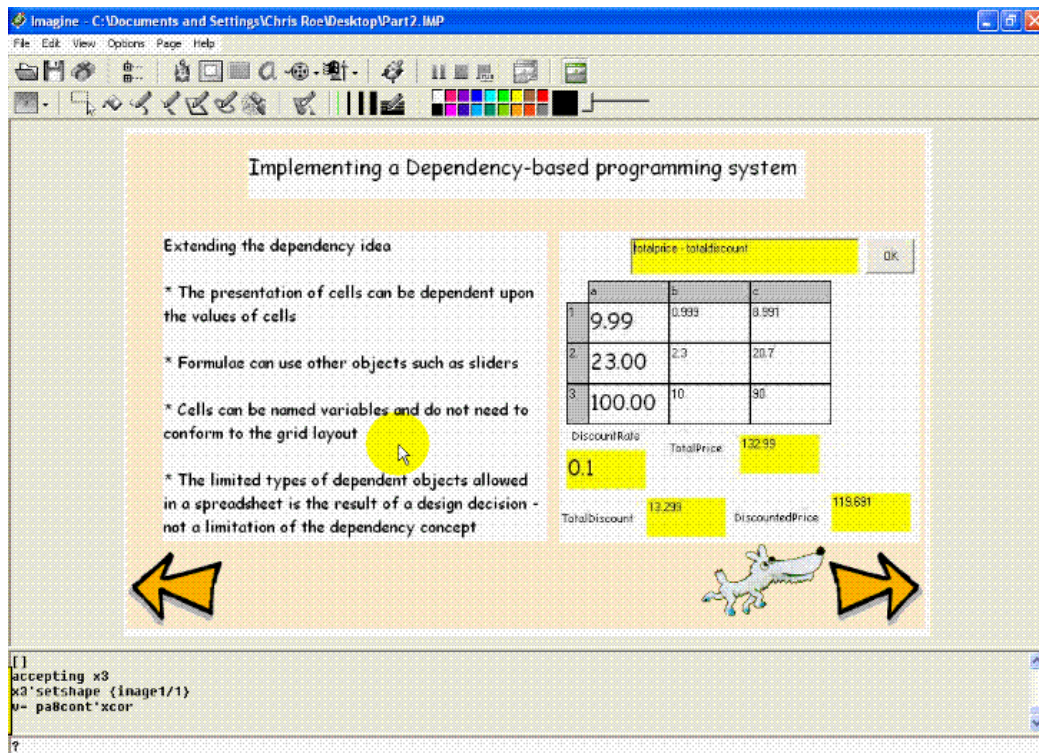


Figure 8. Illustrating dependency in specifying slide content, layout and animation

Discussion

Introducing dependency-by-definition to Imagine Logo has a significant practical impact on model construction. In implementing features such as dynamic graphs, the convenience of being able to formulate dependencies as definitions without needing to explicitly set up the underlying maintenance mechanisms is liberating, and promotes the speed and flexibility of development that is associated with the use of spreadsheets. Extending Imagine Logo in this way seems to be particularly topical when we consider the prominent role that dependency plays in many successful educational products, such as *Cabri Geometry* and *Visual Fractions*. However, in this context, it is worth noting (Kalas, personal communication, 2005) that the designers of Imagine Logo have long been aware of the expressive qualities of dependencies, yet have been uncertain about how these can be effectively and coherently exploited. This is consistent with the fact that understanding the relationship between modelling with dependency and programming has been a central theme of the Empirical Modelling project (EM Web) since its inception more than twenty years ago. Over this period, research on this theme has generated:

- practical tools and models to demonstrate that modelling *purely* with definitions in principle has expressive power and range encompassing what has been illustrated using Imagine-d Logo above (see for instance the presentation graphicspresHarfield2007 (EM Web), in which definitions support geometric modelling and interface management, and implement notations for formulating definitions that are themselves defined within the modelling tool).
- a methodology for modelling with dependency that is associated with a radical reappraisal of thinking about computing (Beynon, 2007; King, 2007).

Relevant issues can be exposed by reviewing "programming with dependency" from each of the three perspectives identified in the introduction: programming practice, pedagogy, and computing science.

Programming practice

It is a conceptual mistake to suppose that adding features to a programming language that contribute to ease-of-development and ease-of-use necessarily thereby enhances its value as a vehicle for constructivist learning. Learning and 'optimisation to achieve a specific programming goal' are activities that engage the mind in quite different ways, and develop in entirely divergent directions. In learning, it is vital to pay as much attention to the environment as possible and to explore every discrepancy between expectation and observation, whereas optimisation involves specialising the task and engineering the context so as to minimise the information we need to know about the environment and eliminate all observation and action that can be made superfluous. The distinction drawn between virtuosity and musicality in performance is one indication that the speed with which we can accomplish specific tasks is not a good measure of how much we have learnt about a domain.

It is also well-recognised that adding features to a programming environment does not necessarily lead to a coherent conceptual enhancement. The challenge to conceptual integrity in Imagine-d Logo can be gauged from comparing the code developed by Tomcsanyi (2003) to implement the dynamic geometry depicted in Figure 2 with the following listing, which specifies the same configuration using the EDEN interpreter – the principal tool for Empirical Modelling:

```
line AB, BC, AC
point A, B, C
AB=[A,B]           // the line joining points A and B
BC=[B,C]
AC=[C,A]
A = {20, 180}       // the point p4
B = {260, 490}      // the point p1
C = {285, 50}       // the point p3
line perpA, perpB, perpC
perpC = perpend(C, AB) // the perpendicular dropped from C on to AB
perpB = perpend(B, AC)
perpA = perpend(A, BC)
point D
D = intersect(perpA, perpB)
```

In both EDEN and the Imagine-d Logo prototype, the programmer can *either* specify the explicit procedural actions that maintain the geometric dependencies in Figure 2, *or* embody them in a set of definitions. But by what criterion should one approach be adopted rather than the other, and is the distinction between these two approaches more than a matter of individual taste?

Pedagogy

In comparing Logo with Boxer, diSessa (1997) stresses the importance of criteria other than abstract expressive power in relation to learning. His concern is as much with the physical form that programs take as with the underlying programming constructs. This is consistent with his observation (diSessa, 2000) that "*make it experiential* is perhaps the single most powerful educational heuristic I know". diSessa also seeks to challenge the assumption that educational software "comes in large units that are so slick and complex as to require many person-years of effort to create them, usually by highly technically competent software engineers".

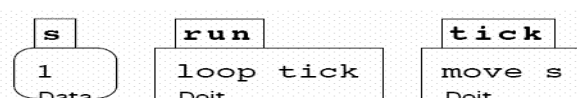


Figure 9. The **tick model** that defines uniform motion (diSessa, 2000)

diSessa's "tick model" of uniform motion, as depicted in Figure 9, reflects his desire to shift the emphasis from programs as complex bodies of code for professional analysis towards visual artefacts that are intelligible to the learner. It also highlights his concern for a close correspondence between the interpretation of the program in the domain and the components of a visual representation. Similar concerns for *experiential* rather than symbolic representations of programs were influential in the design of Imagine Logo (Kalas et al, 2000).

Beynon and Roe (2006) raise a deeper concern about the suitability of programming languages as a means to realise constructionist ideals. The traditional and fundamental understanding of *what a program is* casts the programmer in the role of a specialist with recondite knowledge whose job is to ensure that a program carries out specific functional tasks that conform to explicit user requirements. Under this interpretation, neither the programmer nor the user interacts with the program as an artefact in a way that is well-matched to the agenda of the active learner. The kind of conflation of pupil, teacher, developer roles that constructionist learning ideally demands is much better mapped on to the collaborative moulding of a spreadsheet model to situation by human agents with different levels of domain knowledge and computational literacy discussed by Nardi (1993). Empirical Modelling endorses this idea, privileging the use of sets of definitions to express the latent dependency in situations over the use of conventional procedural constructs and data abstractions to describe processes and behaviours.

Computing Science

The besetting problems of large-scale complex software development are closely connected with the need to develop domain knowledge in an incremental and systematic fashion whilst making computer models in a constructivist spirit. This highlights a strong connection between pedagogical aspirations for software and fundamental concerns in computing science. Empirical Modelling identifies the primitive learning that must underpin software development with a particular way of gaining familiarity and understanding in a domain. This activity has a primary orientation closer to Papert's *bricolage* than conventional programming (cf. Roe (2003), pages 109-116), in that its emphasis is upon modelling concrete situations rather than prescribing abstract behaviours. As discussed in Beynon (2007), this is achieved through making artefacts that are appropriately characterised as *construals* rather than programs. A construal offers interactive experience to the model-maker in which empirically validated patterns of observation, dependency and agency in the domain are faithfully reflected.

The notion of giving computer-support to making construals fits in well with diSessa's conception of *material intelligence* as "an intelligence achieved cooperatively with external materials". The close correspondence between the components of the Boxer program in Figure 9 and their physical counterparts in the domain is well-matched to an analysis of domain observables, dependencies and agents. Generalising such a correspondence to typical computer programs is altogether more problematic. The fundamental historical rationale for the program concept has been that of automating human agency out-of-the-loop. The sharp mathematically defined abstractions that pervade the theory of programming are its legacy. The difficulty of maintaining a credible correspondence between objects as experienced in the domain and objects as they emerge in the design of programs is but one indication of the tension between automated action and reflective thinking. It is clear for instance that the object-oriented programming activity associated with implementing the dynamic geometry of Figure 6 in Tomcsanyi (2003) does not admit the same direct interpretation in geometric terms as the family of EDEN definitions above.

Conclusion

The above discussion highlights the crucial importance in constructionist use of the computer of the experiential aspects of the modeller's interaction. For learning to be effective, there must be a close and – in some sense – directly perceptible relationship between situations, objects and concepts in the domain and the interactive experience of the model-making. In establishing this relationship, what is – or comes to be – hidden and implicit in the interaction is quite as important as what is explicit. Arguably, much learning is associated with acquiring such familiarity with

relationships between significant observables that the procedures for identifying these relationships that at first must be laboriously followed become routine and internalised. But where the polished products of professional software development strive to anticipate what kind of functionality and familiarisation is appropriate and attainable and optimise the user's interaction accordingly, the essence of educational software is that it should expose the raw activities that are associated with identifying functionality and gaining familiarity with procedures.

Dependencies are archetypal examples of relationships that underpin learning. The merits of spreadsheets for educational applications can plausibly be attributed to the fact that – unlike conventional programs – they carry out dependency maintenance activity in a way that is both hidden and intentionally *uninterpreted*. Programming environments such as Imagine-d Logo and EDEN in principle support similar sense-making conventions that allow some “program code” to be explicitly interpreted and some to be as-if-internalised by the model-maker. The problematic issue is that such a policy of blended interpretation and uninterpretation is not consistent with the orthodox abstract conception of computational semantics.

Modern variants of Logo, such as Imagine Logo and Boxer, have taken significant steps towards the concretisation that is essential in supporting a new practice of computing better attuned to the demands of constructionist learning. The sense-making theories needed to enfranchise such practice and to promote its mature use have yet to be identified. As Albirini, citing Gärdenfors and Johansson, remarks: “educational technology still lacks a unifying theory of learning that would serve as a foundation for its use in the classroom” (Albirini, 2007). The motivating thesis behind Empirical Modelling is that such a theory of learning has to be developed in conjunction with a radically new conception of computing (Beynon, 2007).

As diSessa's “tick model” of uniform motion illustrates (see Figure 9), there are contexts in which there is an excellent match between experience of a domain and of its representation in a computer-based artefact. It is evident that the quality of the “tick model” is linked to the fact that the “turtle” metaphor for change is ideally suited to representing the dynamics of physical motion and time. One of the problematic aspects of our current conception of computing is that the professional programmer aspires to a world-view that is equally well-attuned to the very same turtle metaphor. Under this view, software should ideally fit into a world where the processes of change are effected through the continuous action of tamed agents operating under the stable gaze of a benign objective external observer. In this world, constructing software is uncovering the theory that governs the mind as machine and all change as ultimately explicable with reference to eternal absolute physical laws.

Papert's vision for constructionist learning is not well-served by this limited understanding of computation. Such an objectified conception of change needs to make room for an altogether wilder perception of the perspectives and vectors that are associated with learning: their subjectivity, potential incoherence, and the absence of well-rehearsed and practiced trajectories. The primary emphasis in computing for this purpose is not on processes and behaviours, but on situations in which human interpretation operates in flux, uncertainty and potential confusion, and on construals rather than programs.

The above reflections suggest two different ways in which the Imagine-d Logo prototype might contribute to future research. With a sufficiently robust and comprehensive implementation, the dependency front-end could become a standard feature of the Imagine Logo environment, to be used in a discretionary manner as users see fit. The examples in this paper have illustrated the potential for using such a front-end in several different ways: to link the contents of a screen display to internal values, to set up dependencies between turtle positions, or to define simple dynamic geometry configurations. There is scope for school-based empirical testing to explore the merits of such a mixed-paradigm approach. A more radical research direction would involve trying to blend the fine qualities of tools such as Imagine Logo and Boxer where accessibility is concerned with existing Empirical Modelling tools such as the EDEN interpreter (as discussed in King, 2007). The extensive Empirical Modelling project archive (EM web) already demonstrates proof-of-concept for dependency-by-definition in a wide range of applications, but as yet has had

little exposure where users without specialist computer science knowledge are concerned. Given the Jamesian philosophical stance of Empirical Modelling, a better metaphor for mental activity in that context – rather than a turtle in motion – might be that proposed by Naur (2001): *an octopus jumping in a pile of rags*.

References

- Albirini, A. (2007) *The Crisis in Educational Technology, and the Prospect of Reinventing Education*, Educational Technology & Society. 10(1), 227-236
- Baker, J.E. & Sugden, S.J. (2003) *Spreadsheets in Education - the First 25 Years*. e-Journal: Spreadsheets in Education 1(1): 18-43
- Beynon, M. (2007). *Computing technology for learning - in need of a radical new conception*. Educational Technology & Society. 10(1), 94-106
- Beynon, W.M., Roe, C.P. (2006) *Enriching Computer Support for Constructionism*. In Alkhalifa, E. (ed.) *Cognitively Informed Systems*, 209-233
- Cabri geometry: <http://www.cabri.com/v2/pages/en/index.php> (27th March 2007)
- diSessa, A.A. (1997) *Twenty reasons why you should use Boxer (instead of Logo)* In M. Turcsányi-Szabó (Ed.), *Learning & Exploring with Logo: Proceedings of the Sixth European Logo Conference*. Budapest Hungary, 7-27
- diSessa, A.A. (2000) *Changing Minds: Computers, Learning and Literacy*. Boston: MIT Press
- EM Web: <http://www.dcs.warwick.ac.uk/modelling> (28th March 2007)
- Kalas, I. & Blaho, A. (2000) *Imagine...New generation of Logo: Programmable pictures*. In *Proceedings of WCC2000*, 427-430
- King, K. (2007). *Uncovering Empirical Modelling*, MSc, Computer Science, Warwick University
- Logo Foundation. *What is Logo?* <http://el.media.mit.edu/logo-foundation/logo/index.html> (25th March 2007)
- Nardi, B. A. (1993). *A small matter of programming*. MIT Press
- Naur, P. (2001) *Anti-philosophical Dictionary*. naur.com publishing
- Perry, M. (2001) *Automate Dependency Tracking*, Parts 1, 2 & 3, JavaWorld.com, Aug-Oct 2001, <http://www.javaworld.com/javaworld/jw-08-2001/jw-0817-automatic.html> (26th March 2007)
- Roe, C.P. (2003). *Computers for Learning: An Empirical Modelling perspective*. PhD thesis, Computer Science, The University of Warwick
- Roe, C.P., Pratt, D., Jones, I. (2005) *Putting the **learning** back into e-learning*. In *Proceedings of CERME4*, Sant Felix de Guixols, Spain
- Sendov B., Dicheva D. (1988). *Mathematics Laboratory in Logo Style*, In Fr. Lovis & E. Tagg (eds.), *Computers in Education*, IFIP, ECCE'88, North Holland, 213-217
- Tomcsanyi, P. (2003) *Implementing object dependencies in Imagine Logo*, In *Proceedings of EuroLogo 2003*, 27-30 August, Porto, Portugal
- Visual Fractions: <http://www.logo.com/cat/view/logotron-visual-fractions.html> (27th March 2007)

Design and Implementation of a Logo-based Computer Graphics Course

Pavel Boytchev, boytchev@fmi.uni-sofia.bg

Dept of Information Technologies, Faculty of Mathematics and Informatics, Sofia University

Abstract

Two years ago the Faculty of Mathematics and Informatics at Sofia University makes a decision to design a new series of Logo-based courses which make use of the modern technology. The pedagogical component of the challenge is to design a multidisciplinary course suitable for students with different skills and interests. From a development perspective the challenge is to build an entirely new one. And finally the course must be attractive regardless of the seriousness and complexity of the topics included in it.

The paper discusses the structure of the course including the final weeks when topics emerging from students' course projects are taught. Each lesson from the course is based on sets of sample programs representing the general lifecycle of software development. This includes designing, coding and debugging. Samples are created on-the-fly, thus different instances of the course results in different final projects. Lessons are interactive and students may interfere with the direction of demonstrated software development.

Three lessons from the course are sketched in the paper. The first one is taught in week 4 and spans over Computer Science, Calculus, Analytical Geometry; and Applied Statistics and Probability. The lesson in week 6 is focused on composition of complex movements and their synchronization. It uses elements from Computer Science, Geometry, Physics, and Trigonometrics. The third lesson is about relative transformational geometry and its application in the form of Turtle Graphics. It uses elements from Physics, Robotics, Biology and Art. Snapshots from the projects are shown in Figure 1.

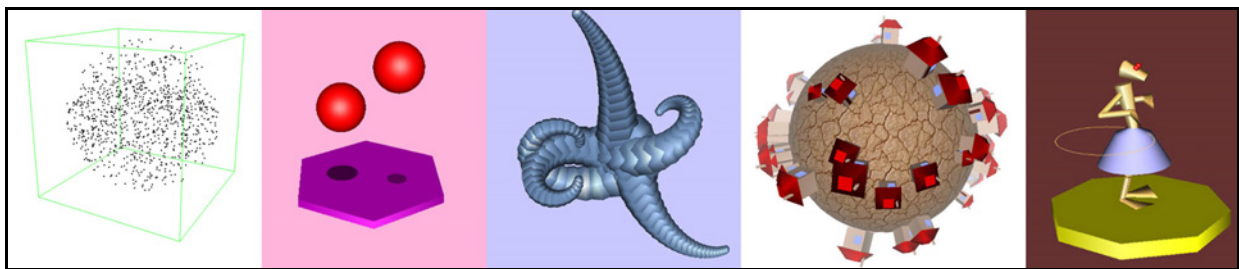


Figure 1 Snapshots of projects developed through the course

A few of the students' course projects are also presented in the paper – an animated 3D model of the Solar system, a transformational geometry impression called "United Colours of Elica", and a 3D model of the Faculty building.

The presented Logo-based Computer Graphics course 'conquers' educational territories from the dominating C and C++. It is taught since 2005 and is well accepted by students. They find it both interesting and useful for their education. The future of the course is very promising. A textbook is planned, as well as extension of the teacher-student interaction beyond the time frame of the course.

Keywords

Logo; Computer Graphics; course; Elica

Preface

The Logo programming language has been traditionally used in the classroom to describe, to explain and to explore the fundamental principles of Computer Science. The properties of this language make it an advantageous choice for a first programming language. It is not only the simplicity of the syntax that makes it beneficial to students, to their teachers and to the overall learning process. Another factor is the immediate access to drawing functionality via Turtle Graphics. The widespread use of Turtle Graphics is a unique phenomenon which has some negative effect on the public opinion about Logo. Namely, Logo might be considered as just a system for doing graphics with a turtle. The opposite opinion is also still widespread – some turtle graphics environments and libraries are considered to be Logo.

The postponed negative effect of these opinions is that some teachers and parents think of Logo as of a childish language, not appropriate for doing serious stuff. Apparently what is considered as serious, is merely everything which has direct positive impact on entry and exit exam.

The Faculty of Mathematics and Informatics at Sofia University, has been using Logo for some decades in various courses – from teacher training courses to e-Learning and Technology Enhanced Learning (TEL) courses (Nikolova and Sendova, 1995). The faculty members were not only using Logo in their classes, but also they were developing new Logo versions. The first one being Plane Geometry System, later renamed to Geomland, released more than 20 years ago, to the latest Elica Logo, which is still under active development (Elica, 2007).

The Challenge

In the spring of 2005 the Department of Information Technology makes a decision to provide greater support to the development of Logo, as well as to revive its use in the courses. Thus the main challenge is formulated as to *design and implement a new series of Logo-based courses* which make use of the modern technological achievements.

Pedagogical challenges

The creation of a new stream of courses would have greater academical value if it is adaptable to the specific needs of the teacher and the course. For example, the Logo-based Computer Graphics course could be taught to Bachelors and to Master's programmes. It could be taught to students in Informatics, Mathematics, Applied mathematics, Mechanics, etc. Many of the students attending the courses are supposed to be familiar with some of the basic concepts of computer science – algorithms, procedural and functional programming, OOP, etc. However, the courses should be accessible to students which have no significant (or in some cases any) experience in these areas either because they are freshmen, or their specialty does not require the full extend of the programming skills. Examples of such students are those who are studying for becoming teachers in Mathematics or Informatics (Vitukhnovskaya, 2005).

The initiative for designing new courses clearly stated that they should cover all previous courses, so the educational plan of the Faculty is not invalidated. Additionally, the courses should provide enough complexity and robustness so that new students from other programs could also join. For example students from the Computer Graphics Master's Programme are also supposed to visit these courses.

Development challenges

The Logo implementations used by the Faculty in the past are getting more outdated, and the new versions of these implementations are not available due to various reasons. This poses the challenge what programming environment to use for the courses (Laucius, 2006). It is decided to use Elica Logo as a software backbone for the course and to design new courses based on the advanced features of this Logo dialect (Boytchev, 2005). Switching to a new Logo dialect with a wider but different range of features makes most of the already existing teaching materials obsolete. The design of new courseware would lead to major changes in the lessons' content

and will affect the actual teaching procedures. This poses a unique challenge as to how to design and implement new courses in a way that the overall advantage is much bigger than the overall disadvantage.

Psychological challenges

Charm is one of the most undervalued factors for modern courses. Teaching Computer Science and Computer Graphics at university level is treated as a serious endeavor. The matter is heavy, sophisticated and knotty. Whether it is attractive to students or not is not as important as the content of the course. The new courses which are to be developed need this charm in order to win students' hearts. Initially the courses start as selective, so it is important to design the course in such a way that students are willing to enroll not only to gain some credits, but to learn something useful. The so called Nintendo generation (West, 1995) poses new requirements for courses, especially those related to computer graphics. Providing adequate level of charm is essential factor to meet these requirements.

Course structure

A typical course spans over 15 academic weeks and includes 30 lecture hours plus 30-60 lab hours. Traditionally students are evaluated several times during a course and eventually they have an exam, which comprises the biggest part of their final score. The final exam usually has two components – practical and theoretical.

The Logo-based courses could follow the same settled structure, but instead, it is decided that the way to measure student's efforts, knowledge and imagination is to provoke their creativity in the creation of a Logo-based course projects. As a result there is no examination synopsis of Elica, and students are focused on their projects long before the examination session.

Introductory section

The course is divided into three sections – see Figure 2. The first section takes three weeks (i.e. 6 academic hours). It is dedicated to the introduction of Logo and Elica. During the first week, while students get accustomed to Elica, they learn the basics of the Logo language, including data types, all reserved words and some functions for words and lists processing.

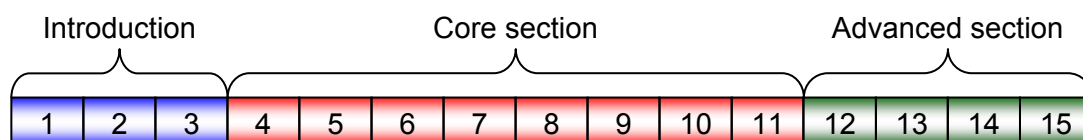


Figure 2 Course sections – weekly structure

The second week is left for iteration, recursion and custom operators. The third introductory week is dedicated to OOP features of Elica. Students learn various methods for class definition, object using and manipulation of OOP entities.

The core section

The core part of the course takes roughly 8 weeks and covers topics directly related to computer graphics, 3D modeling and animation. The lessons also utilize implicitly knowledge from the conventional computer science subjects, like algorithms and optimizations. It is not possible to make a precise topic-week relationship, because the course is continuously adapting itself to the level of the students. Follows a list of the basic topics in the core section:

- 1D/2D objects: points, lines, segments, rays, circles, ellipses, squares, rectangles.
- 3D objects: cubes, parallelograms, spheres, cones, cylinders, spline surfaces.
- 3D spaces: coordinate systems, transformations (translation, scaling, rotation).

- Custom user-defined objects, custom methods for object rendering, support for nested graphical objects, local transformations.
- Color, RGB color space, lighting, object materials, fog effects, textures.
- Turtle graphics in 3D space, hierarchical joint models, creating complex android models.
- View points, flying through space, perspective and orthogonal projections.
- Animation, smooth movement, techniques for achieving realism – resistance, inertia. Simulation of physical movements, composition of various movements.
- Building graphical user interfaces, event handlers, capturing mouse activity, responding to user interaction, building interactive environments.

Advanced section

The last few weeks of the course are left for topics which are not taught during the basic course, but are related to the course projects selected by the students. During the second half of the core section students determine the topic of their projects. If the projects use specific objects or techniques, then they are included in the advanced section.

As a result of this layout, the advanced sections for different semesters are always different. They depend entirely on what students chose to do. Here are some examples of advanced topics:

- Simulation of water waves.
- Run-time modification of classes and objects.
- Building Logo libraries and software packages.

The exam

About the middle of the course, weeks 8 to 10, students are encouraged to propose their projects. If accepted they can start working on them immediately. Some proposals might not be accepted either because they are too complex or are too easy. Complex proposal are simplified to a level which will make them doable in a reasonable period of time. Projects that are too simple or too easy to implement in Elica are loaded with some extra features.

Students may request additional information about issues related to their projects. In this way they determine the topics for the third section of the course (Stoyanova, 1999). When they think they are ready with the projects they submit them electronically for evaluation. If submission is done in advance, students get reply with the current rating and suggestions how to improve it. If students have time and are willing to get a higher rating they can adjust their projects and resubmit them. This project-evaluation scheme is repeated several times.

For the last 4 semesters this scheme proves to be successful for students. Most students resubmit their projects twice. Very few of them send three or four versions. The possibility to have their projects reviewed and resubmitted makes students more comfortable with the course and lets them display their creativity and imagination.

Class structure

The structure of each individual class is a small copy of the whole course structure. It starts with an easy to understand basic concepts, and then it goes to details and advanced ideas, and ends with explanations-responses to students' questions.

Classes are extremely interactive. The topic is presented as series of programs which are typed and executed in real time. The image from the teacher's computer is projected on a big screen and students can follow the process of programming from the beginning to the end.

Usually the first sample program is just few lines long. Next samples just add some features to it thus building more complex programs. One of the co-ideas of the course is to demonstrate the process of real-world programming. The selected technique of erecting a full-functional program from scratch is a perfect live scenario for students (Dagienė, 1999).

One of the most enjoyable by the students moments is when the program does not behave correctly. Such situations are handled by on-the-fly debugging and modification of the code. The teacher's thinking is vocalized and projected on the screen without interruptions, so students learn various techniques to resolve bugs.

Quite often students are asked to provide ideas how to solve a problem. Most of the cases they suggest interesting solutions which are immediately tested. There are also cases when the suggestions do not solve the problem. Such ideas are also tested, because they are an excellent opportunity to practice important problem-solving and pitfall-avoiding skills.

All samples during each class are archived and provided to the students, so they can replay the whole lesson, and do further explorations with the code. Some are getting the sources right after the lesson; others are downloading them from an online repository (Elica Repository, 2007). At the beginning of the course many students use a paper notebook where they try to copy by hand all sample programs. However they soon find that it is impossible to cope with the dynamics of a live coding. The sample programs are constantly changing while testing different ideas and debugging, so students realize that it is much more valuable to grasp the ideas and the techniques, rather than to memorize the exact programs.

The selected structure of the classes is well accepted by students. However, it imposes additional requirements for the teacher. The most obvious one is that it is unavoidable to write only correct programs. A single class may have 20 to 30 program samples, and each of them is a potentially dangerous place for the teacher to make a mistake. Some of the mistakes are intentional, but others are not. The latter makes teaching very demanding, because the teacher should extract educational value not only from good examples, but also from bad. On the other hand unintentional errors place the teacher in a stress situation, because s/he has to provide a solution with an adequate reasoning in a limited time frame.

Multidisciplinarity

Each lesson from the course has a main topic about features and techniques from the area of programming with Elica. Additionally, each lesson is enriched with many smaller fragments from other subjects, mainly from Mathematics and Physics. Some lessons 'rent' ideas from Biology, Psychology, Astronomy, Geography, Statistics and even Arts. Multidisciplinarity¹ is an intentional feature of the course which makes classes more interesting and demonstrates the application of software in various scientific and artistic domains (Sendova, 2006). To demonstrate the multidisciplinarity of the lessons, three of them are briefly described below.

Points (week 4)

The first graphical lesson is scheduled for week 4 of the course. It starts with the introduction of the most primitive graphical object in Elica – the point. A point is defined by 3 numbers which stand for its coordinates. The initial examples used in this lesson are used to describe how the graphical system is initialized, how to visualize the coordinate system, and how to create a point at some position. During the lesson the students face a short challenge – to create 1000 random points within a virtual cube. The second challenge is more complex: to create the points in a sphere, not in a cube – Figure 3. Some of the students do not know how to do this, others suggest using the formula of a sphere $x^2+y^2+z^2=r^2$.

¹ Although the course uses elements from other subjects the main focus is always on the graphics with Logo. Topics from non-programming subjects are used mainly to give ideas for projects and to illustrate the interconnection of sciences.

The most often suggested solution is to create random points in a cube, but keep only those which are internal to the sphere. This is one of the milestones in the lesson, because students realize that the created points are less than 1000.

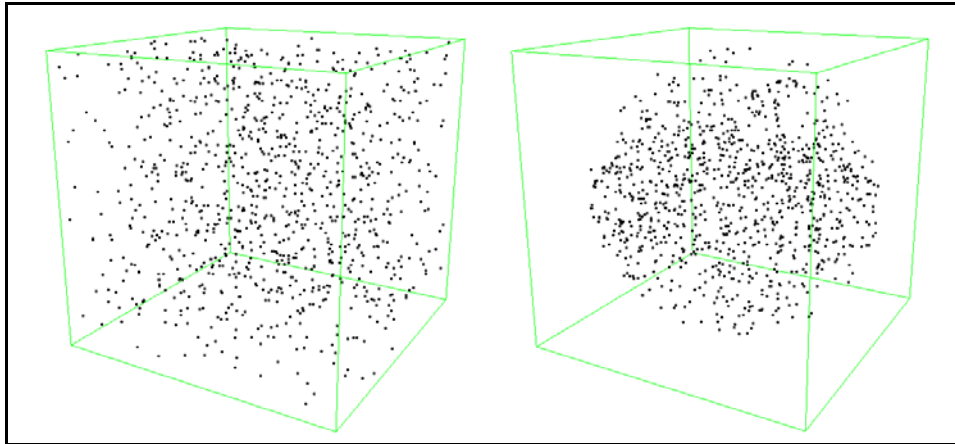


Figure 3 Random points in a cube and in a sphere

The lesson continues in the direction of how to fix this. Several solutions are suggested and tested. One of them is to keep track of the number of internal points and terminate the loop when 1000 is reached. Another solution is to check for validity before the creation of a point. Throughout a series of tests and modifications students learn to be doubtful about a program that looks correct.

Once they calm down thinking they've done all possible solutions, they are asked to find a way to directly create a point inside the sphere without the need to test for inclusion. Some students try to give up, others test teacher's sense of humor by making all the points at (0, 0, 0). With help from the teacher students find two more solutions – one based on Cartesian coordinates, and another based on polar coordinates.

At the end of the lesson students are directed back to the wrong solution of 1000 points in a cube some of which are also in a sphere. This wrong solution serves as a basis to solve a new problem – how to find the approximate value of π exploiting the bug in the program. It takes some time to conclude that the number of points in the sphere and in the cube depends on the ratio of the sphere's and cube's volumes which is $\pi/6$. Table 1 shows the approximate values for π calculated for cases with 10, 100, 1000, and 10000 points.

	10	100	1000	10000
Test 1	1.8	3.12	3.22	3.15
Test 2	3.0	3.66	3.05	3.16
Test 3	2.4	2.82	3.23	3.09

Table 1 Calculated approximation of π

This first graphical lesson spans over a few subjects studied in the faculty. Students learn some basic concepts from Computer Science (various loops, conditional execution, counters), Calculus (function composition), Analytical Geometry (Cartesian and polar coordinates, equations for cubes and spheres, volumes); and finally – Applied Statistics and Probability.

Bouncing balls (week 6)

The bouncing balls lesson is taught at week 6. It is the first time students learn how to simulate physical properties of objects through their movement. The final goal of the lesson is to show how to make two balls bouncing on a flexible plate – Figure 4. The balls have different bounce

periods. The plate starts to vibrate whenever it is hit by any of the balls. The vibration gradually fades if no hit is encountered for some time. Both balls have shadows casted on the plate. While balls go away from the plate, shadows become smaller and lighter.

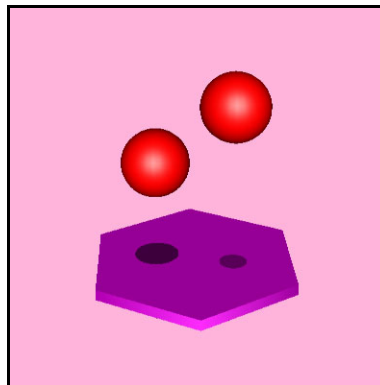


Figure 4 Bouncing balls

The series of programs in this lesson starts with the simple case of a ball going linearly up and down. Students realize immediately that such movement is not visually acceptable because it is physically incorrect. The next step is to replace the linear movement with sine function². Now the movement of the balls near their highest positions is acceptable, but they do not bounce off sharply from the plate. This gives the clue to use the absolute value of sine – Figure 5.

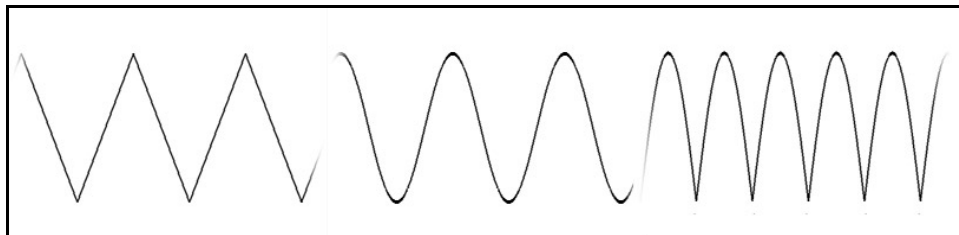


Figure 5 Linear, sine and absolute-sine bouncing

By using $|\sin|$ students learn that very often physical movements in an animation are not based on the physically correct formulae. Because of performance issues some calculations are replaced by faster one. That is why $|\sin|$ can effectively replace the calculation of a parabola.

Sine is heavily used throughout the whole lesson. It is 'responsible' not only for the bounce, but for another 3 actions – the vibration of the plate based on variation of $\sin(x)/x$, the rotation of the plate which smoothly alternates between clockwise and counter-clockwise, and the flying of the view point around the scene.

An interesting problem with the bouncing balls is the synchronization of various movements. The bouncing of the balls is independent on the vibration of the plate. The program must explicitly synchronize both motions in order to trick the viewer to think that vibration is caused by the hit.

The "Bouncing balls" lesson is another example of a multidisciplinary. It practices skills in several subjects – Computer Science, Geometry, Physics, and Trigonometrics.

² By describing the properties of the bouncing the students are asked to think of a function with similar properties. Some students come up with the idea of sine or cosine quite quickly, others need more hints.

Relative Transformational Geometry and Turtle graphics (week 8 and 9)

Transformational Geometry deals with affine transformation of objects. It is heavily used in many Elica lessons, because the original shapes of all objects are the canonical solids (like cube $1 \times 1 \times 1$, or sphere with radius 1). All other objects are generated by transforming the canonicals. Relativity in geometry is when each object has its local coordinate system, and other objects bound to it are defined in terms of its local system.

The usage of relative transformational geometry is based on transformational matrices. One interesting application is how to stack transformations for chained objects. Because of the flexible structure of each lesson, the same lesson taught in two different years are different. The main topic is the same, but the set of samples and especially the final programs becomes quite different. For example, “Dandelions” from Figure 6 are the final program of the lesson from December 2005, while “Octopus” is from the same lesson two semesters later.

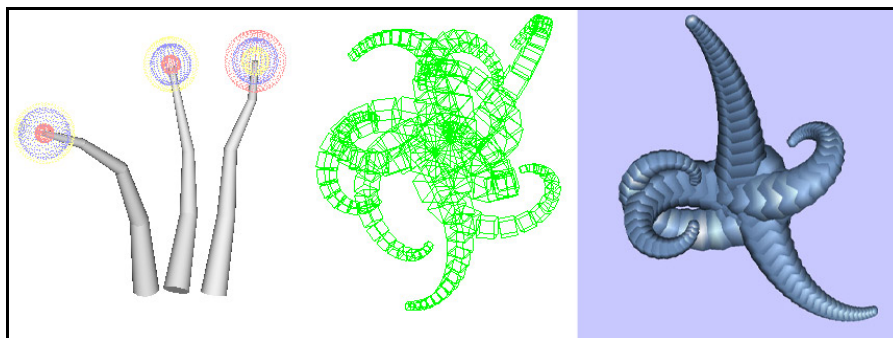


Figure 6 “Dandelions” (left), “Octopus” (middle), and “Lernaean Pentapus” (right)

“Octopus” gives birth to other program which is included in the Online Elica Museum (Elica Museum, 2007). The third image in Figure 6 is a snapshot from “Lernaean Pentapus” exhibit from the museum. Using Relative Transformational Geometry prepares students to accept easily the benefits of Turtle Graphics. Relativity is the nature of turtle’s motion and thus it has common grounds with Differential Geometry.

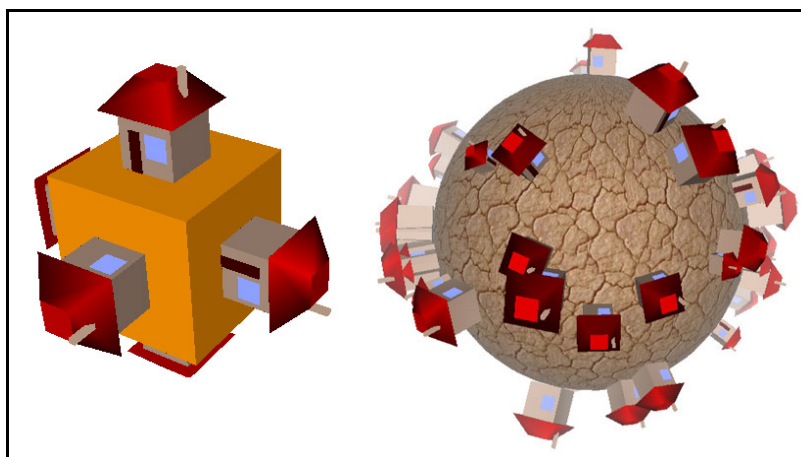


Figure 7 Houses created by turtle crawling on the surface of a cube and a sphere.

The first 5-10 programs in the lesson are used for introduction to 2D and 3D turtle graphics. A typical example of this introduction is to make a turtle crawl on the surface of a cube and build various objects on each face, or to crawl on a sphere – see Figure 7.

The more advanced usage of Turtle Graphics is to create the objects drawn earlier with relative transformational geometry, but this time using turtles. Students see how the reimplementations become shorter and easier to understand, because turtles and relative transformations have a common mathematical background. The rest of the lesson is dedicated to step-by-step building and animation of a humanoid body – legs, hands, torso, and head, traversed by an invisible turtle. The animation is done by synchronous changes in the joints – every joint has a local coordinate system and is a place where the body parts have some level of motion freedom.

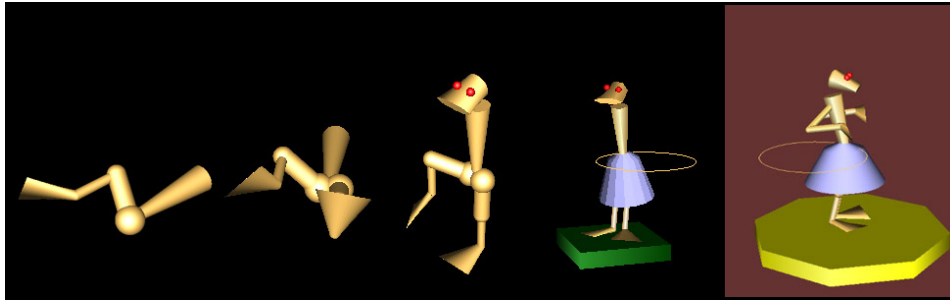


Figure 8 Four phases of humanoid building (left) and the “Circus Dancer” (right)

Most often the created body is of a dancer playing with a hoop – Figure 8. The right-most image is from the “Circus Dancer” exhibition from the Online Elica Museum.

This lesson teaches students how to program more complex systems. It makes use of various elements from Physics, Robotics, Biology, and Arts.

Course projects show-cases

The Online Elica Museum is a section from the Elica site containing a collection of Elica programs, mostly related to animation of 3D objects. The programs are freely available as source code for everyone. Some of the course projects developed by students are so successful that they are included in the museum as independent exhibits. Of course, the original source code is polished to make it more representative and clearer, so that other students can learn from it. This section of the paper presents snapshots of several students’ course projects – some are published in the museum, others are not.

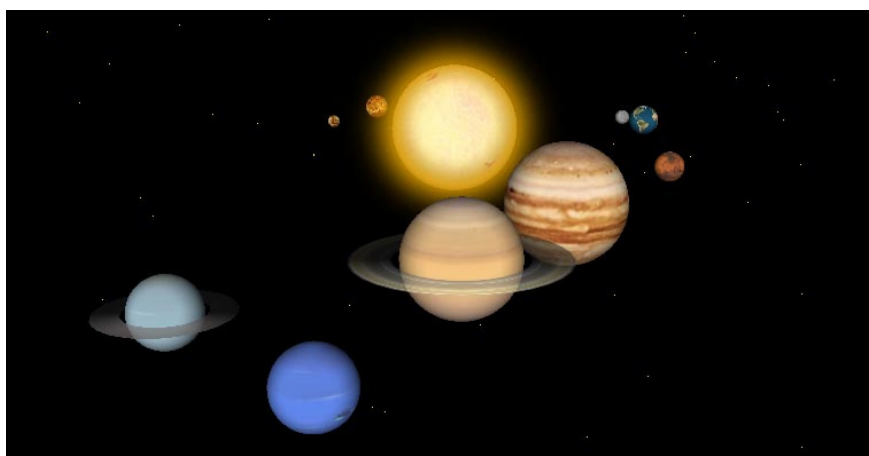


Figure 9 Project “Solar System”

The project “Solar System” creates the Sun and the planets as 3D objects, and then animates the system by rotating the planets around the Sun – Figure 9. Each planet is dressed in a texture

taken from real astronomical images. Some details are added to make the animation more realistic – there are stars at the background, the planets spin around their axes too, Saturn and Uranus have semitransparent rings, and the Moon rotates around the Earth. The making of this project requires understanding of how textures are used and some math skill to define the orbits and spinning of the planets as well as composition of several non-linear movements.

The second project – see Figure 10 – is named by the student who wrote it “United Colours of Elica”. It features a series of animations of cube transformation. Each face of the cube is implemented as a spline surface which control points are defined by an invisible 3D turtle.



Figure 10 Project “United Colours of Elica’

Another student’s project is to make a virtual 3D building of the Faculty of Mathematics and Informatics. A snapshot of this building is shown in Figure 11.

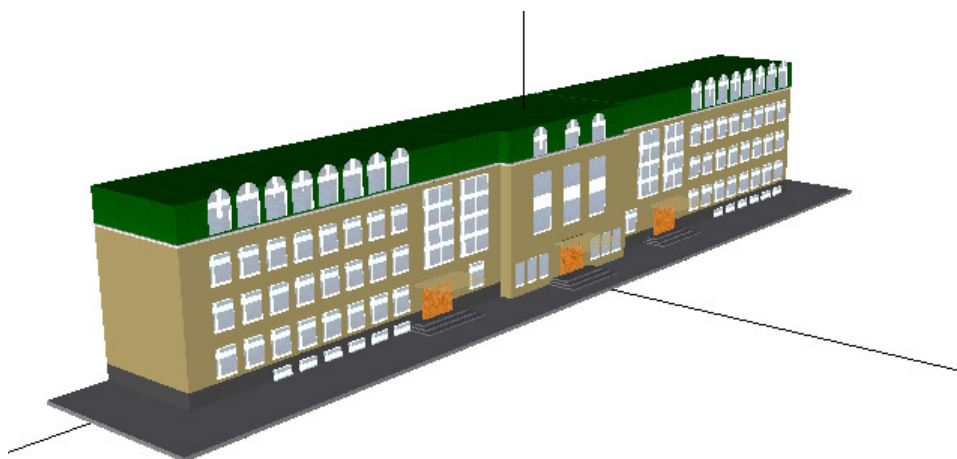


Figure 11 Project “The Building of Faculty of Mathematics and Informatics”

Conclusion

Logo is used in primary and secondary school curricula, but it can be used in various courses at a university level too. This applies not only for light-weight courses, but also to traditionally heavy subjects like Computer Graphics where C and C++ are dominant programming languages. The course described here has been active for 4 semesters. The number of enrolled students is getting bigger, popularity rises. Anonymous questionnaires show that students definitely prefer interactive lessons. One of the most liked features is the life demonstration of programming which includes both coding and debugging. The project-based scenario of each lesson serves as a prototype of the lifecycle of a typical software development.

Another feature of the course admired by students is the possibility to unfold their imagination by working on a course project related to their personal interests. Multidisciplinarity builds bridges between programming and the highly varied skills of students – although they come from different specialities, they learn something new and interesting. The way how the topics in the course are designed, especially the custom-defined ones during the last few weeks, is a thing which helps students build awareness of and confidence in their own skills. This is due to the fact the students feel the course as if it is personally oriented towards them.

The future of the Logo-based course is very promising. There are several ideas for its further development. One of them is the writing of a textbook, which is a challenge of its own, because of the dynamism of the course topics. The first step is to collect enough variations for each topic.

Another idea is to move the Elica Repository to a more interactive online system – Moodle. This will add many new possibilities for student-teacher off-lesson interaction.

Since January 1, 2007, Bulgaria joined European Union. This had a significant impact on the educational system. Many Universities are now revising their Programmes because of expected increase of foreign students. One of the possible future plans of the Logo-based Computer Graphics course is to offer it to English-speaking audience.

References

Laucius, R. (2006) *Issues of Selecting a Programming Environment or a Programming Curriculum in General Education*. ISSEP 2006, LNCS 4226, Springer-Verlag Berlin Heidelberg 2006, pp. 169-178.

Sendova, E. (2006) *Handling the Diversity of Learners' Interests by Putting Informatics Content in Various Contexts*. ISSEP 2006, LNCS 4226, Springer-Verlag Berlin Heidelberg 2006, pp. 71-82.

Boychev, P. (2005) *Using Logo to Model and Animate*. In Proceedings of Eurologo 2005. Edited by Gr. Gregorczyk et al, August, pp 66-75.

Vitukhnovskaya, A. (2005) *Logo for the Would-be Teachers of the Computer Science Elementary Course*. In Proceedings of Eurologo 2005. Edited by Gr. Gregorczyk et al, August, pp 245-256.

Dagienė, V. (1999) *Logo and Changes in Learning: Project-based Methodology*. In Proceedings of Eurologo 1999. Edited by R. Nikolov et al, August, pp 179-186.

Stoyanova, N. (1999) *The Students - Authors of Tasks*. In Proceedings of Eurologo 1999. Edited by R. Nikolov et al, August, pp 334-339.

Nikolova, I. and Sendova E (1995) *Logo in the Curriculum for Future Teachers: A Project-Based Approach*. In Proceedings of Eurologo 1995. Edited by M. Dúill, July, pp 7-12.

West, I. (1995) *Logo: Forward 1 to Freedom*. In Proceedings of Eurologo 1995. Edited by M. Dúill, July, pp 87-92.

Online References

Elica (2007), Elica Home Page, <http://www.elica.net>

Elica Museum (2007), Elica Online Museum, <http://www.elica.net/site/museum/museum.html>

Elica Repository (2007), Elica Samples Repository, <http://www.fmi.uni-sofia.bg/Members/boychev/elika/>

Words are silver, mouse-clicks are gold?

(or how to optimize the level of language formalization of young students in a Logo-based cubics world)

*One should NOT aim at being possible to understand,
but at being IMPOSSIBLE to misunderstand.*
Quintilian, circa 100 AD

Evgenia Sendova, jsendova@mit.edu

Institute of Mathematics and Informatics, Bulgarian Academy of Sciences

Toni Chehlarova, tchehlarova@mail.bg

Faculty of Mathematics and Informatics, Plovdiv University, Bulgaria

Pavel Boytchev, Boytchev@fmi.uni-sofia.bg

Faculty of Mathematics and Informatics, Sofia University, Bulgaria

Abstract

How do we teach children to express and communicate ideas in a formal and informal mode? What type of language do they need in a concrete context? How should they determine a proper level of formalization of their descriptions? In an attempt to explore these issues we carried out an experiment with 5th graders from three Bulgarian schools during which the students experienced the whole process of generating a *good description* – becoming aware of the ambiguity, producing counterexamples, reducing the ambiguity, eliminating the redundancy. The educational scenarios and the *Cubix Editor* (an Elica-Logo application) used in the experimental activities were developed in the frames of the DALEST European project.

The first impressions confirm our belief that the language is playing significant role in the learning experiences of the students, that *the relationship between thoughts and words involves back and forth reshaping process*. While constructing and describing cubical structures they articulated their own ideas, developed concepts collaboratively with others, moved between everyday and mathematical terms, between procedural and declarative style, exploring the boundaries of understanding. Such interplay with the step-wise refinement of their descriptions of cubical structures would hopefully enhance students' skills for working with mathematical definitions, on one hand, and prepare them for writing, debugging and explaining programs, on the other.



Figure 1. Describing a 2D representation of Cubix construction



Figure 2. Describing a 3D virtual representation of Cubix construction

Keywords

Language, formalization, definition, precision, virtual world, 3D visualization, mathematics, Logo, teaching and learning

The role of a language when learning math in a computer environment

Learning to express and communicate ideas – an educational challenge

Students find it difficult to express themselves. To express one's thoughts clearly and precisely involves significant cognitive processes that pass through different levels of formalization. Learning a language, to use John Austin celebrated phrase, is learning how to do things with words – not simply what to say, but how, where, to whom, and under what circumstances (Bruner, 1990).

It has been suggested (Truss 1999, Khait, 2005) that mathematics is an essentially linguistic activity characterized by association of words with precise meanings. The classical use of definitions in mathematics is to introduce new concepts (Vinner, 1991). The information society has added a new aspect - a competent professional has to construct new definitions of various computer objects and understand definitions constructed by colleagues. Furthermore, mathematical literacy can be interpreted (Kent, Noss, 2002) as an ability to communicate ideas, based on an understanding of the ways in which the ideas can be expressed. Experts in mathematics and informatics education find that the most important component of the mathematical way of thinking for workers in ICT is the ability to express their ideas in a rigorous language, where each word and expression has an unambiguous meaning (Khait, 2003).

The work of the citizen of the information society could be seen as ongoing transition between formal (human-computer) and informal (human-human) communications. Thus the education should take care of enhancing the abilities of the students to translate their intuition to a formal language, to create definitions of new abstract objects, to write comprehensible descriptions of abstract and real structures and to understand those created by others.

The main problem we have been interested for years is how the young students “grasp the significance” of situations in a way that can help them master the lexicon that fits the situation? Or more concretely, how to develop the ability of students to articulate their ideas in an ICT enhanced environment. Since learning how to push the frontier of what we can express with words (Papert, 1980) has always been a very important and appealing educational goal.

Looking back

In the early days of launching computers in Bulgarian education Logo was introduced as a part of an integrated subject for 5th and 6th graders – *Language and Mathematics* taught in the frames of a Bulgarian educational experiment (Nikolov, Sendova, 1991). The intention was to show the intersection of the language study with mathematical thinking in the context of informatics. Differences and similarities between the natural languages and formal languages (mathematics and Logo) have been discussed with the students. In addition to being a formal language, Logo was appreciated as an executable language, giving an immediate feedback of one's reasoning. The experimental textbooks included problems on translation from a natural language into an artificial language and vice versa, algorithmic descriptions of grammar rules, etc. A further step in that direction was the development of integrated textbooks in mathematics and informatics (Dicheva et al, 1997) in which the traditional mathematics definitions (e.g. about symmetry) were accompanied by Logo procedures.

An important component of the educational experiment involved development of software tools for mathematical explorations based on a linguistic structure (Sendov, Sendova, 1993). Thanks to appropriately designed Logo-based environments such as *Geomland* (Sendov, Dicheva, 1988, Filimonov, Sendov 1989) students were able to make their mathematics definitions workable and even to correct some definitions published in the math textbooks (Sendova, 1992).

Developer's perspective on languages - a glance BK

Of course, programming languages are special type of tools for expressing ideas, not only those of users, but also of developers. Most of the programming languages are settled into a rut. Changes occur very rarely and after a long period of redesign. The Logo language is quite different. It helps users to enrich the language with their own definitions, but it also inspires people to make their own dialects of Logo. It is hard to implement a programming language but the educational philosophy behind Logo is so appealing that many people got the courage to make their own versions. For the first 40 years of its existing Logo had more than 170 different implementations. Some of them are made by teams of professionals, others - by researchers and students at Universities, and yet a third part of Logos - by parents. The common feature of most Logo developers is that they did it for fun.

Five years ago *The Logo Tree Project* was launched (Boychev, 2007). The goal of the project was to build a genealogical tree of all known new and old Logo implementations. This tree was expected to demonstrate the evolution, the diversity and the vitality of Logo as a programming language.

The project has three phases. The first phase, *Data Collection*, has started in September 2002 and is aiming at collecting some basic data about each Logo implementation – name, versions, dates, platform, inspirators. The second phase, *Data Analysis*, is meant to get clusters of historically and evolutionary related Logos and to analyze the internal relationships among Logo implementations. The last phase is the *Data Visualization* focusing on the interactive visual representation of the Logo Tree.

Clicking vs. speaking

There are still ongoing discussions of whether we need a programming language when using computers for educational purposes, and if so, which languages are the most appropriate ones. Even enthusiastic supporters of Logo as educational philosophy and culture claim that *this language is only an indication of what might be offered in the future. But the very fact that it is indication is significant* (Noss, 1993). Those who find programming languages to be a great obstacle for teachers advocate the direct manipulation tool kits: *If the exploratory spirit in school could be achieved by tools with "ease-of-use features" why should we insist on working with languages?* The answer is not to contrast the two approaches – *the software designers should develop continuous media providing the whole range, from easy means for making sketches up to means enabling you to create the tools you need* (Sendov, Sendova, 1995)

What the level of precision of speech should be?

Among the multiple versions of the famous joke about the way different scientists (a philosopher, a physicist, a mathematician and a programmer) described a herd of cows, we particularly like the following – the philosopher stated: *all the cows are black*; his physicist friend corrected him: *all the cows in that herd are black*, the mathematician said: *all the cows in that herd visible from here are black* - and the programmer added: *all the cows in that herd visible from here are black at least from their visible side*.

Such a gradation of the precision of human speech reflects well certain inadequacy of the way we sometimes express ourselves influenced by our profession. If the goal of math educators is that students learn to formalize their thinking and to understand at least part of the mathematical language, the professional mathematicians are going to the other extremity - some seem to take *greater pride in how few people could read their work, than in how many people had read their work*. According to the poetic expression of Ó Dúill Logo's level of formalization is intermediate between mathematics and natural language – between Gödel and Goethe. Thus, for us, the Logo fans, it was natural to look for a way to make the students verbalize their thinking even when they are clicking! And further on – to develop their skills in choosing a reasonable level of formalization depending on the context.

The experiment with cubical constructions

Our rationale behind the experiment was that the mastery of mathematics definitions (language) can come from participation in formal language as an instrument for communication. For the purpose we designed the following scenario:

The scenario

- Writing description of various cubical constructions by 2D representations
- Building constructions of fixed number of cubes by means of a Logo application and describing them to a peer so that s/he could reproduce the construction.
- Joint explorations of descriptions produced by the students
- Back and forth process of reshaping the descriptions. Discussion
- Free style constructions
- Identifying the right level of formal description

The participants

The *official* participants were students from 5th grade attending five schools from Sofia and Plovdiv of slightly different nature – two mathematics school with intensive study of mathematics and three others - with traditional curriculum, with extracurricular activities in mathematics, and with high specialization in sports, respectively. Occasionally we would ask friends and colleagues, university students, in-service and pre-service teachers to describe a construction or suggest strategies for giving “the best” description of structures of the kind.

The construction tool

The main tool for creating the cubical constructions we used was *Cubix Editor* - a specially designed Elica-Logo application (Boytchev, 2007). It was developed in the frames of the DALEST project (<http://www.ucy.ac.cy/dalest/>) whose goal is to enhance middle school students' 3D geometry understanding and spatial visualization skills by working with dynamic visualization images (Boytchev, Chehlarova, Sendova, 2007). The theoretical background behind the design of the DALEST software has been considered in details in (Christou et al. 2007)]. *Cubix Editor* allows the construction of 3D unit-sized cube structures by clicking. The students can make a library of various constructions (*Figure 3*). A very useful characteristic of the application is the rotation of the platform, which enables dynamic visualizations of the front, side and top view.



Figure 3. *Cubix Editor* – some constructions of its library

Episode 1: A simple construction – multiple descriptions



Figure 4. Three versions of a 4-cube construction for students to describe

The first problem was to describe three versions of a construction (Figure 4).

The idea was to check if the students would take into account the colors and relate the position of the construction to a board. The coordinate system has still not been introduced.

Which mathematical terms can we use? Are you going to grade us? May I explain it without writing? Are you going to check my spelling? – these were the typical questions asked. In order to predispose kids to write the way they believe was the most suitable for a peer to reconstruct the figure we told them the philosopher-vs.-programmer joke quoted above. Little we knew...

The first class from Sofia Math School had no experience with the *Cubix Editor*. It was interesting for us to see to what extent would the students be influenced by the “silent contract” of implementing the topics learned most recently - volume and surface of solids. (Students’ names are fictitious since they were concerned with possible grammar and spelling mistakes. We told them to sign their sheets only if they wanted.)

Damian: *Figure 4-a consists of 4 cubes of the same color. Its surface is 18.*

Maria: *Figure 4-a consists of 4 cubes of the same color. 2 of them are joint in an upright **rectangular cuboid**, and the rest 2 form a flatwise **rectangular cuboid**. The two cuboids are joint so that the bottom cube of the **upright cuboid** is glued to the back cube of the flatwise one.*

Lea: *The Figure 4-a has 4 cubes with **18 sides** all being grey. It consists of two cuboids – the **base** of the first one is its smaller side, and that of the second – its bigger side.*

Peter: *The figure 4-a consists of 4 cubes and its **volume is 4 cm³**. They form two **cuboids** – one of **altitude 1 cm**, and the other of **altitude 2 cm**.*

Others included the board in their descriptions as a construction of smaller cubes:

Jenny: *There are **44 cubes in a square form**. There are 25 bigger cubes inside completing the square. There is a building on the whole square: two cubes one on top of the other and next to them glued aside - another such little “tower”.*

The most unexpected for us was their literal reflecting the joke we told them at the beginning so as to illustrate the different levels of speech precision:

Veronica: *On Figure 4-a we see 4 cubes - next to an upright cuboid formed by 2 cubes another 2-cube cuboid is flatwise. **All cubes are grey from their visible side.***

Denitsa: *There are 4 cubes on Figure 4-b. Under the first one there is another cube. There is a cube to the right of the second one **and all its visible sides are dark. The visible sides of the rest of the cubes are white.***

The students from Plovdiv who already had been working with the *Cubix Editor* used the board to locate the cubes similarly to a chess board:

Koya: *The board is 5x5. I denote the columns A, B, C, D, E and the rows – 1, 2, 3, 4, 5. Then the Figure 4-a could be described as follows: a cube on C3. A cube on D3. On D4 there is a cube and on top of it – another one.*

Let us note that when describing Figure 4-b Koya thought of adding the color as a third element of the information needed. This would be a smooth transition for Koya to learn about the classical way of presenting the position of each unit cube in the space by means of coordinate systems. For Figure 4-a possible representations of the kind are:

- (2; 1; 1); (2; 2; 1); (1; 2; 1); (1; 2; 2) – an imaginary relative coordinate system fitting the size of the construction (2x2x2), or
- (3; 2; 1); (3; 3; 1); (2; 3; 1); (2; 3; 2) – an absolute coordinate system associated with the board. For Figure 4-b the color should be added as a fourth coordinate.

The richness of possible descriptions even with such a simple construction was a matter of hot discussions among the students. Some kids gave several descriptions of the same figure:

Neda (6th grade): *There is an angle of 3 cubes. On top of the one which is not the corner, there is another cube, or I could say:*

*There is a square on the board of 4 cubes. One **of the cubes is lifted and put on** a cube next to it.*

We encouraged them to formulate their own criteria about what would be a “better description” – *more understandable, more clear, shorter*. Standard mathematics notation turned out to be helpful but it would often be mixed with everyday terms:

Maria: *Take 4 cubes and label them **a, b, c, d**. Arrange them **in the shape of a chair of your taste**. They have 12 sides and 8 vertices. **a** is on top of **b**, and **c** is next to **d**.*

Vesko: *The solid consists of 4 cubes, 3 of which form a right angle. One of the cubes is on top of one of the ending cubes.*

Boyan: *3 cuboids are of size $(2a \times a \times a)$. Every two cuboids have a common vertex.*

Svetla: *Put 2 cubes next to each other. Put a cube on top of the **left** and in front of the **right** one.*

Although many of the students had used the terms *left cube* and *right cube* there was a student from the Plovdiv Athletic School (having worked with *Cubix editor*) who immediately said: *If I look from behind, the description will be different*. By the way the problems of explaining the notion of “left” to someone who doesn’t see what we see was discussed in (Feynman, Leighton, Sands, 1971): *If we tell a Martian that our heart is on the left side, he would ask: “Duhhh – the left side?” Now our problem is to describe to him which side the heart goes on without his ever seeing anything that we see, and without our ever sending any sample to him of what we mean by “right” – no standard right-handed object. Can we do it?*

Epizode 2: Constructing and describing simple compositions

The second task was to build constructions of fixed number of cubes (5 and 10) by means of *Cubix Editor* and to describe them to a peer so that s/he could reproduce the construction.



Figure 5. Alija's original construction and Georgi's materialised description of it

Even in the case of 5-cubes figures it was unexpected for the students to see that their descriptions were ambiguous.

Alija: *5 cubes are put on the board as follows: 2 sideward, starting from the last – one more forward, next to it one more, and starting from the last cube one more forward.*

When constructing his composition after Alija's description (Figure 5) Georgi showed how he would describe his own construction to eliminate the ambiguity:

Georgi: *A pink cube is put on the 3d row of the 2nd column of a 5x5 board. Next to it on its right (again in the 3d row) there is a green cube. On the top of it – a red cube, next to it in the air is glued a yellow one.*

Most descriptions were in a declarative style – this was expected since the students will learn their first programming language, Logo, in 6th grade:

Emmy: *4 cubes are touching each other in such a way that if looked from above they form a square. On the top of one of the corners there is a cube.*

Still there were a few in a procedural style:

Pavlina: *On the 5x5 board put green cubes as follows: put 3 cubes on the second row on the middle 3 squares. Repeat the same on the 3^d and the 4th row. Put another cube on the top of the central cube of those already constructed.*

Some students coined their own terms such as empty square – a square having its middle cubes removed. Still there were some rather fuzzy (although imaginative) descriptions, e.g. 10 cubes in the shape of T (sort of); “Г” is formed of 4 cubes; a “plus” flatwise; a 3-hump snake, a gun lying down, stairs, castle, wall, tower.

The interaction of language and thought has been described by Vygotski (1934) by means of a dynamic model: Thought is not merely expressed in words; it comes into existence through them, this relationship between thoughts and words involving back and forth reshaping process.

Episode 3: Back and forth between clicking and speaking

Such a reshaping process becomes very natural when having *an object to think with* – a virtual manipulable construction. Making consecutive refinements of a description was in the core of the learning scenario for the next session. The students were expected:

- to start constructing after a description of a peer (chosen by the teacher)
- to demonstrate that the proposed description is not complete
- to improve the description written on the board
- to construct and describe their own figure

The description of a construction given by one of the kids reads: 12 cubes are put in groups of 3 so that every 3 form a right angle. Each group touches the other one by a cube. They are colored. The description together with different solutions according to it is shown in Figure 6. Note that the idea of coloring the separate groups in different colors was generated by the kids without this being explicitly stated in the description. Some students showed different solutions on a single board.

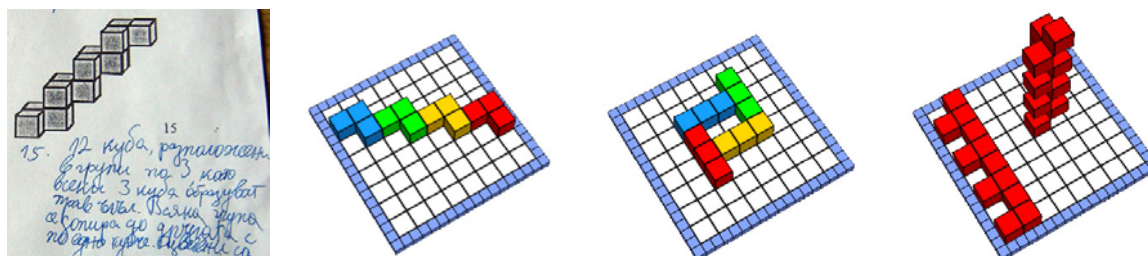


Figure 6. Constructions based on a description of a classmate

When the students looked around and saw so many different constructions fitting the description they were eager to see the original. Then their reactions differed as well:

Ivan: I got it right, you didn't!

Iliana: I can construct another one fitting this description.

Boris: *Here is how you can improve the description* - three cubes are forming an angle. We put on top of one of the ending cubes again a 3-cube angle. **And so on, until** we have 4 angles.

Similar was the process of materializing the description of another student of the same class: There are 11 cubes. 4 are on a horizontal line, 2 are on a line perpendicular to it, and from these two cubes three new are rising. Out of the three cubes two are sticking out. The original and different solutions of the students are shown in Figure 7.

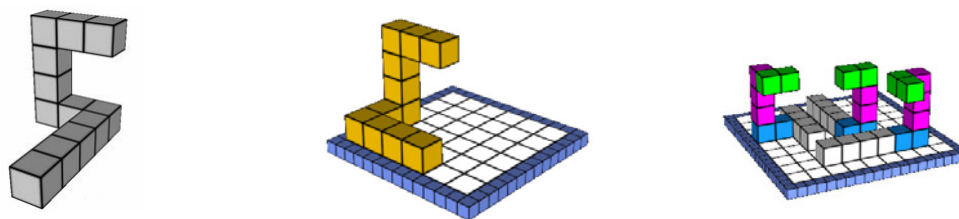


Figure 7. Constructions based on a classmate's description of a figure

This time the constructors were aware of the variety of solutions and even started calculating the possibilities. Unlike the previous situation when many of them thought that the *right solution* would be to guess the original construction this time they were proud to show what we agreed to call “counter examples” of the claim that the description is not ambiguous. To facilitate the checking they would use different colors for the consecutive steps in the description. When the teacher drew on the board the original construction the author of the description exclaimed: *I don't believe I have written this...*

The next task for all was to improve the descriptions so as to eliminate the ambiguity. In most of the cases the descriptions were much more precise:

Yavor – Build a row of 4 cubes. Put two cubes on the left side of the last one. Put 3 additional cubes above the leftmost cube. Put two more cubes on the right of the topmost cube.

Evgenia and Veronika: A row of 4 cubes. When looking from above – a row of 2 cubes is stuck to the left side of the forth cube. On top of the cube not touching the row a column of 3 cubes is built. Next to the topmost cube two cubes have been stuck towards the row of 4 cubes.

Nikola: Let us look the board from above and denote the 4 directions East, West, North, and South. There is a 4-cube row from West to East. The fourth cube is the one most to eastwards, next to the northern side of the 4th cube a row of 2 cubes is glued spreading to the North. The second cube is the one most to the North. On the top of it there are 3 cubes one over the other. On the southern side of the topmost cube of this column 2 cubes are spread from North to South, the second cube being most to the Southwards.

Although the ambiguity has been reduced this time there was a redundancy in most of the descriptions – thus the next task was to delete what was not necessary. Such interplay with the step-wise refinement of the descriptions would hopefully cultivate their skills for working with mathematical definitions so as to appreciate their compactness and precision. Furthermore, describing constructions and building after descriptions would prepare them better for programming in Logo.

Episode 4: Free-style constructions – session 1

The last task of the session involved a lot of imagination – the students were given the freedom of making their own constructions related to the real world. After that they had to describe them for their peers. At this point all kind of interesting creatures started appearing on the screens – robots, flowers, jumping dogs, fish in aquariums, shooting canons, etc. (Figure 8).

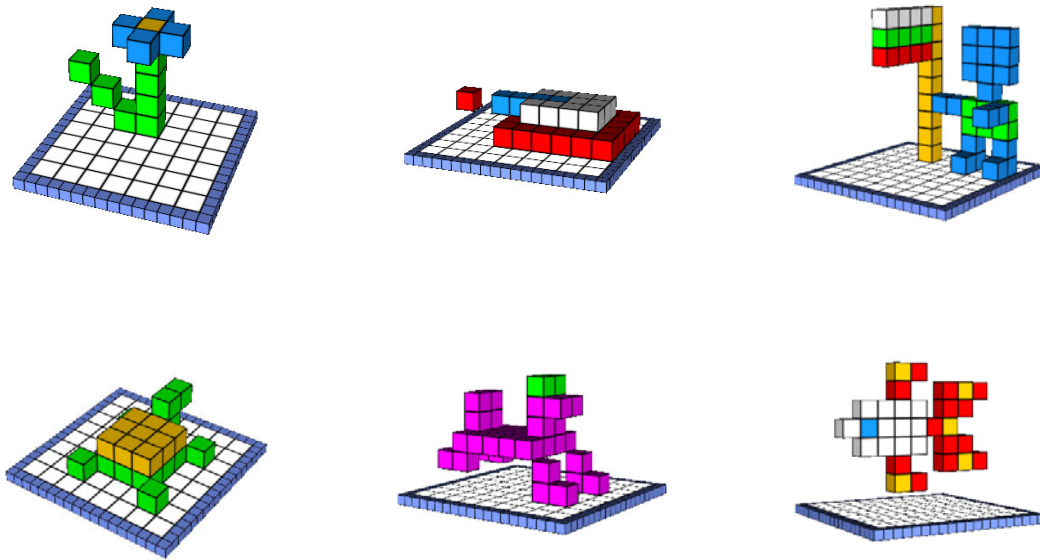


Figure 8. Constructions in a free style

Easy and pleasant to construct but so difficult to describe – especially when the construction had gaps... *How could we explain this? If it was in the plane, it is clear – we could use the notation of the chess board, but here?* Then there came the insight: *May I add one more line, something like a 3D chess and use figures, Latin letters and Cyrillic letters?* No sooner said than done (Figure 9, left). This was a great generalization provided that they had no experience with coordinates other than the chess board and the cross words.

Other questions were related to the proper naming (a problem we often face with the Logo procedures): *Can I say that this is a flower (a dog, a fish) and use a name for my construction? I will name the colored parts of my Ninja Turtle and this would help the builder to reconstruct it* (Figure 9, right)... It made sense – they were talking to a peer. Thus the idea of organizing a contest in the style of *Guess my object named so and so* was born – the criteria would be to evaluate both the esthetics of the construction and the correctness of the description.

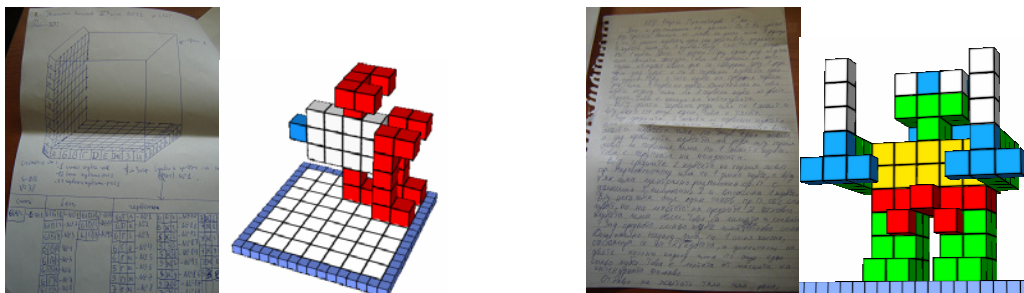


Figure 9. Constructions in a free style with their descriptions

Episode 5: Constructing in a free-style or under constraints?

The next task was for the students to create and then describe a composition under the limitation of exactly 25 cubes. Even within such a constraint some of the constructions were imaginative and rather complex (Figure 10).

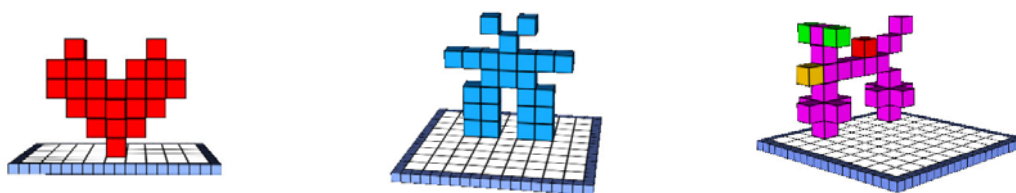


Figure 10. Twenty-five cube constructions

It was interesting to observe that when left to work in a free style this time (unlike the previous session) some students would choose simpler constructions so as to facilitate the process of describing them. As seen in Figure 11 the constructions included not only *architectural pearls* but objects relatively easier to describe.

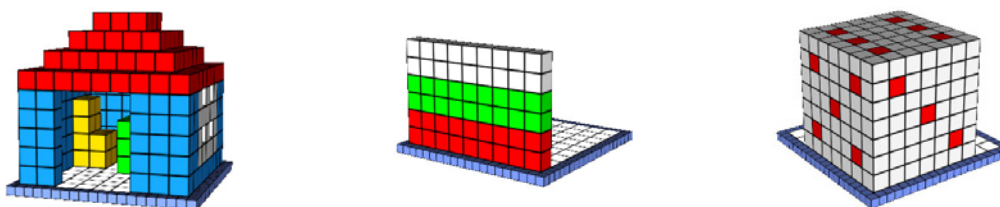


Figure 11. Free style constructions – second session

Four descriptions were chosen by random and written on the board to be materialized – this time most of the figures coincided with the original but we still discussed with the students if this was a matter of chance or of the qualities of the description.

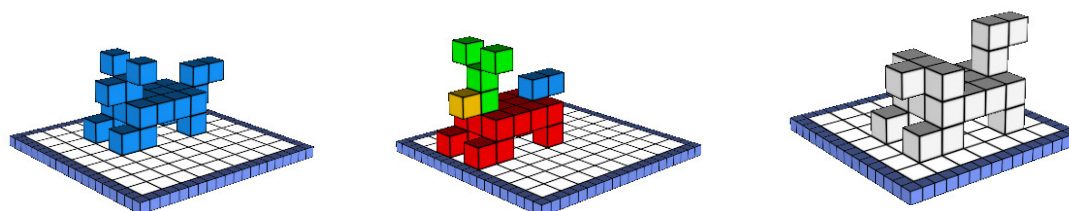


Figure 12. The original dog and its variations according to the description

Naming the compositions with reference to real world objects helped to a great extent. Figures 12 and 13 show the original constructions named correspondingly *My dog* and *Trojan horse* and some of their materialized descriptions. In the case of the *Trojan horse* it was interesting for us to observe that some students decided not to follow strictly the description although it was quite clear and included information about the surface, the volume and the number of cubes: *How could the legs be only 3-cube tall, why is the neck so short – it wouldn't look like a real Trojan horse?..* It was obvious that when the artist in you prevails you do not follow instructions...

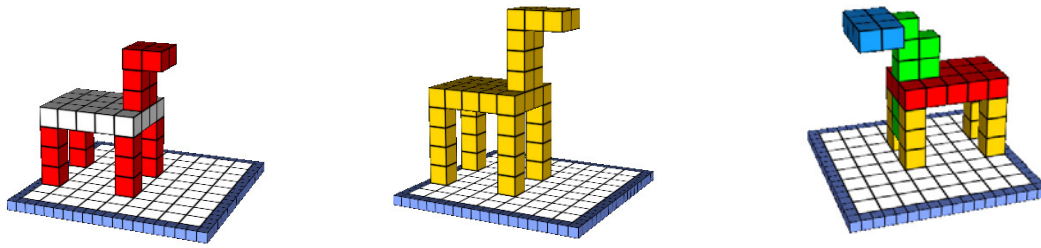


Figure 13. The Trojan horse and some variations

To work within different types of constraints – a fixed number of cubes and one's own endeavor to produce a non-ambiguous definition of a construction, turned out to be very interesting for the kids and for the teachers alike. The endeavor to produce not only an interesting construction but to make sure that their peers would reproduce it made many of the kids work hard at home. Next session they came with new descriptions, typed on a nice sheet of paper with a drawing, or a picture of their compositions attached on a separate sheet.

Episode 6: Developing a Logo-like vocabulary for describing cubics

When working with relatively big groups of students it was difficult for us to follow and *debug* the refinement-of-descriptions process for each individual. Therefore we were curious to experience this process in the style of a case study. Below is the dialog of a 10-year old girl (E) working with the third author (P) in a face-to-face mode:

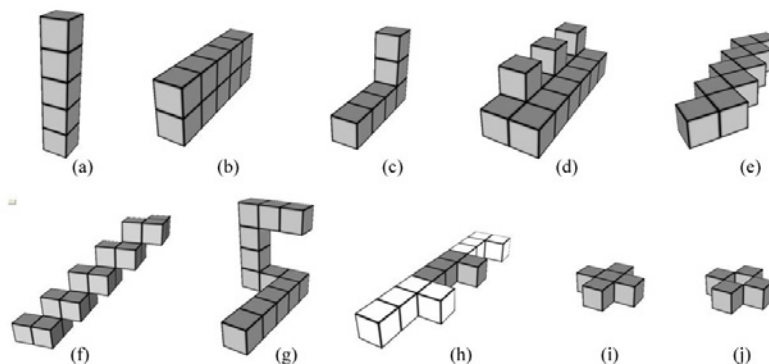


Figure 14. A set of constructions to be described

P: Please describe the constructions in Figure 14.

E: (a) is 5 cubes one over another. (b) is 5 cubes in a row and another row above them. (c) is 4 cubes and 2 more cubes over the last one. (d) is ... huh ... I cannot say, but (e) is zigzag and (f) is ... (as an orchestra conductor she 'draws' with hands in the air the shape of the figure)

P: **Imagine you are in a cube** and describe how you would go through all cubes. You can turn left or right, up or down if needed. Start with figure (c).

E: Step, step, step, step, turn up, step, step.

P: Now try figure (e).

E: Right, step, left, step, right, step, left, step, and so on.

P: And now try (f).

E: Step, step, left, step, turn up... Huh... I cannot do it.

P: OK, then. Try an easier figure. Try (g).

E: *I'll start from the top-right cube.* (She starts quickly) *step, step, step, down, step, step, step ... huh ... huh* (here she tilts her head trying to figure out directions in an upside-down position) ... *huh...* (continues very slowly)... *down, step, ... step... left.... Is it left?* (looks for a hint).

P: You see, you can describe the figures in this way, but what does make it still hard?

E: *It's the turning* (and she again tilts her head trying to express the feeling of disorientation with hands pointing left and right).

P: I see. Try to describe without turning. When you say 'left' it is as if you step to the left without turning. Try now (g) starting from the front cube.

E: *Step, step, step, step, step, left, left, up, up, up, right, right...* (she smiles after describing the figure without pauses for thinking)

P: Go back to figure (f).

E: *I will start from the* (She thinks how to express the step towards us)... *nearer, left, down,* (then continues quickly) *nearer, left, down, nearer, left, down ... It is easy.* (She even does not continue to describe the figure till the end, after realizing she got the pattern right)

P: And now try (h).

E: (waiting for a second thinking how to describe colors) *The first one is white, then white* (looking for another word for 'step') ... *forward, white forward, white right, left, gray forward, gray forward ...*

P: Do you need to say 'white' and 'gray' for each cube?

E: (replaying silently figure (h) again) *No, only when the color changes.*

P: Let's now do figure (i).

E: *Forward, right, left, left, right, forward.*

P: Oh-key... and now the last figure.

E: (Facing a new challenge with the gaps in the structure) *cube, step forward without cube* (secretly looking to see any approval or rejection of her innovation... and then continues boldly), *step right with cube, step left without cube, step left with cube, step right with cube, forward with a cube.*

P: Great! Now you see that it is easy to describe all figures once you have a suitable way for describing.

E: Yes, *really* (and she runs in her room to play a game of her own).

Episode 7: Through the eyes of a university student

We decided to compare the above reasoning with what a student in mathematics and informatics at Plovdiv University wrote when describing Figure 14-(i):

Let us make the cube a die – then each numbered side could correspond to a movement in one of 6 directions: 1- step forward; 6- step back ; 2-step to the right, 5- step to the left, 4-step up, 3- step down. We have two states of the die (corresponding to PD and PU in Logo) and the command CUBE generates a cube in the current position of the die.

PD CUBE PU

step forward step left PD CUBE PU

step right step right PD CUBE PU

step left step forward PD CUBE

Conclusions

Logo Community has proved on many occasions that its members *feel connected, feel familiar, feel at ease, feel friends with ideas* – all these being key factors to learning (from a letter by R. Noss, published on the occasion of the 70th birthday of S. Papert).

Feeling comfortable when expressing ideas, however, doesn't come naturally. Learning a formal language (be it Logo or mathematics) is in many ways like learning a natural language – new vocabulary, syntax, semantics. Both types of languages require a lot of practice to make perfect but the formal languages lack ambiguity and vagueness.

At the previous Logo conference a couple of very interesting educational issues were raised: *How do we best help children to learn the new skill of writing algorithms?* (Ó Dúill, 2005). *How to we teach students to explain their programs to others so that they understand the program?* (Futschek, 2005). We explored these issues with the idea that writing descriptions/algorithms and executing them are important skills, essential for work in a computer environment and a good ground for the next stage of using Logo as a language.

The episodes with the students during DALEST experimental activities described above suggest that the language is playing significant role in the learning experiences of the students. Articulating their own ideas, developing concepts collaboratively with others, moving between everyday and mathematical terms, between procedural and declarative style, exploring the boundaries of understanding, were all phenomena we enjoyed observing. The descriptions the students wrote have not been fully analyzed yet. Still they could be grouped according to some basic features: labeling the cubes; taking into account the initial conditions (number of cubes, size); using metaphors, mathematical objects, similarity to letters, using projections; taking into account the symmetry and the repetition of elements; using relative movement; using coordinates.

The main satisfaction for us was that the students experienced the whole process of generating a *good definition*: becoming aware of the ambiguity, producing counterexamples, reducing the ambiguity, reducing the redundancy. In many situations their examples could be a good reference point to the specifics of the mathematical definitions. At the same time many of their observations regarding the cube structures such as symmetry, modularity, repetition, etc. introduced in a natural way the use of terms and phrases typical for the programming jargon. Such terms as *repeat, repeat-until, forward, back left, draw a cube* prepared the ground for introducing Logo (even in its 3D version) in a natural way. When describing the cubic structures the students were working top-down (from a general description to smaller and smaller details) or bottom-up (developing progressively more complex structures starting from components of the structure being described). Hopefully, at a later stage students would appreciate that techniques that were good for describing cubical structure are good for structured programming, as well. The refinement process itself was very much in the style of debugging a program rather than starting from scratch. We noticed that although the declarative descriptions prevailed there were also procedural ones or such in a mixed style. Just as in Logo *the turtle drawing could be represented procedurally and declaratively* (Blaho, Kalas, Matusova, 1994). So which style is more *natural*? When we asked a university student in mathematics what type of a description he would use for the cubic constructions, declarative or procedural, he answered:

That would depend on to whom I'm talking: If I am speaking to a carpenter, it would be procedural - how to build it out of wood. If I am speaking to a mathematician, it would be procedural -- how to build it out of elementary functions. To a sculptor, declarative. To most people, declarative referring to common objects such as animals.

This was a perfect answer to our endeavor - to teach kids how to reach the right level of formalization depending on the context, the level between *Gödel and Goethe*.

Instead of P.S.

While staying in line for ordering some lunch at the institute canteen the day before submitting our paper, we were deeply involved in the philosophical conversation about the right level of precision of language in a given context. When the turn of the third author (the Elica-Logo developer) arrived he uttered laconically: "A grill, please". The cook, a big peasant type of woman, smiled and answered: *Sorry, Sir, it is too heavy for me to bring it for you...*

Acknowledgments

The work on this paper has been supported by the DALEST project co-funded by the European Union, under MINERVA action.

The authors express their gratitude to the students and teachers participating in the experiment for their enthusiasm and dedication. Special thanks are due to Irina Sharkova and Lilia Ekimova (Sofia Mathematics School) who organized the students and facilitated the analysis of students' work.

References

- Blaho, A., Kalas, I., Matusova, M (1994) *Symbolic Computations and Logo*, Bratislava
- Boytchev, P (2007), *Cubix Editor*, <http://www.elica.net/site/download/dalest/ce.html>
- Boytchev, P.(2007), *Elica*, <http://www.elica.net>
- Boytchev, P. (2007) Logo Tree Project, <http://www.elica.net/download/papers/LogoTreeProject.pdf>
- Boytchev P, Chehlarova T, Sendova E. (2007) *Enhancing Spatial Imagination of Young Students by Activities in 3D Elica Applications*, Proc. of the 36th Spring Conference of the Union of Bulgarian Mathematicians, 2007, Varna, Bulgaria, pp. 109-119
- Bruner, J. (1990), *Acts of Meaning*, Harvard University Press, p. 71
- Christou, C. et al (2007), *Developing Student Spatial Ability with 3D applications*. Submitted to 5th Conference of the European Society for Research in Mathematics Education. Larnaca: Cyprus
- DALEST project: <http://www.ucy.ac.cy/dalest/>
- Dicheva, D. et al (1997) *School informatics in Logo style: a textbook facing the new challenges of the Bulgarian informatics curriculum*, in M. Turcsanyi-Szabo (Ed.) *Learning and Exploring with Logo*, Proceedings of the Sixth European Logo Conference Eurologo'97, Budapest, Hungary, 20-23 August, 1997, pp. 234-239
- Feynman, R., Leighton, R., Sands M. (1971) *The Feynman Lectures on Physics*, Vol. 1, Addison Wesley (1971)52-4
- Filimonov, R., Sendov, B. (1989) *Drawing Logo closer to the Curriculum*, in G. Schuyten & M. Valcke (eds.) *Proceedings of the Second European Logo Conference*, Gent, Belgium, 1989, pp. 363-373
- Gutschek, G. (2005) *Explaining and Understanding LOFO Programs, a Discipline of learning Computer Programming*, in Gregorczyk, G et al (Eds.) *Proceedings, EUROLOGO'2005*, pp. 327-333
- Kent, P. and Noss, R. (2002) *The mathematical components of engineering expertise: the relationship between doing and understanding mathematics*, I.E.E. Second Annual Symposium on Engineering Education, London <http://www.ioe.ac.uk/rnoss/MCEE/Kent-Noss-EE2002-preprint.pdf>

- Khait, A. (2003) *Goal orientation in mathematics education*, International Journal of Mathematical Education in Science and Technology, Volume 34, Number 6, November-December 2003, pp. 847-858(12)
- Khait, A. (2005) *The Definition of Mathematics: Philosophical and Pedagogical Aspects*, Science & Education, vol. 14, Issue 2, p.137-159
- Noss, R. (1993) *The Politics of Logo*, in P.Georgiadis, G. Gyftodimos, Y. Kotsanis, C. Kynigos (eds.) *Logo-like Learning Environments: Reflection&Prospects* - Proceedings of the Fourth European Logo Conference, Athens, Greece
- Nikolov, R. Sendova, E. (1991) *Informatics for All School Ages*, in E. Calabrese (ed.) Proceedings of the Third European Logo Conference, Parma, Italy, pp. 83 - 96
- Ó Dúill, M. (2005), *2 Intelligences: a bricolage*, in Gregorczyk, G et al (Eds.) *Proceedings, EUROLOGO'2005*, pp. 113-122]
- Papert, S. (1980) *Mindstorms*, The Harvester Press, p. 96
- Sendov, B, Dicheva, D. (1988) *A Mathematical laboratory Logo Style*, in Lovis, F and Tagg E. D. (eds.) *Computers in Education – Proceedings of the IFIP TC3 European Conference on Computers in Education (ECCE'88)*, Lausanne, North Holland
- Sendov, B., Sendova, E. (1993) *Learning to Speak Mathematically and Speaking Mathematically to Learn in the Logo-based Environment "Geomland"*, in P.Georgiadis, G. Gyftodimos, Y. Kotsanis, C. Kynigos (eds.) *Logo-like Learning Environments: Reflection&Prospects* - Proceedings of the Fourth European Logo Conference, Athens, Greece, pp. 281 – 287
- Sendov, B., Sendova, E. (1995) *East or West - GEOMLAND is BEST, or Does the Answer Depend on the Angle?*, in A.A. diSessa, C. Hoyles, R. Noss (Eds.) *Computers and Exploratory Learning*, NATO ASI Series, Series F; Computer and Systems Sciences, Vol. 146, Berlin: Springer - Verlag, pp. 59 – 79
- Sendova, E. (1992) *Enhancing the Scientist into the Pupil: A Computer Environment supporting Discoveries in the Classroom*, in Education and Society / R. Aiken (Editor), Information Processing 92, vol. 2, Elsevier Science Publishers B.V. (North-Holland), 1992, IFIP pp 174-180, ISBN:0-444-89750-X
- Sendova, E. (1998) *Identifying Computer Environments and Educational Strategies to Support Creativity and Exploratory Learning*, in Davies, G. (ed.) *Teleteaching'98 Distance Learning, Training and Education*, Proceedings of the XV IFIP World Computer Congress, Vienna/Austria and Budapest/Hungary
- Truss, J. (1999) *Discrete Mathematics for Computer Scientists*, 2nd ed, Addison-Wesley
- Vinner, S. (1991). *The role of definitions in teaching and learning mathematics*, in Advanced mathematical thinking, D. Tall (ed.) Kluwer Academic publishers, pp. 65-81
- Vygotski, L.S. (1934). *Language and Thought*, Gosizdat, Moscow (translation from Psycholinguistics, Holt, Rinechart and Winston, NY, 1961, p. 509)

The use of the transcendental method for helping students to learn with Logo

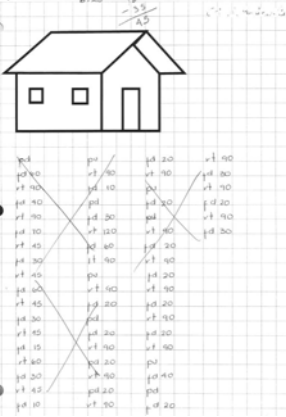
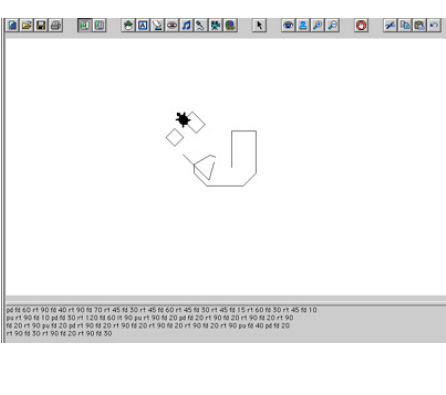
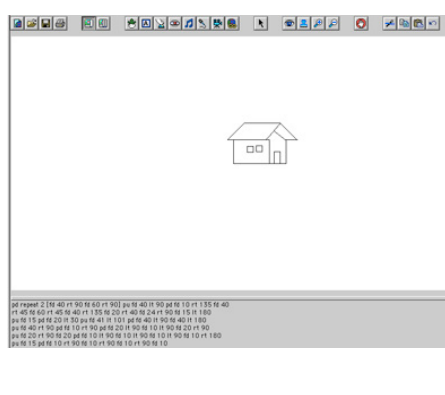
Enís Castellanos, *eniscast@prodigy.net.mx*

Departamento de Arte, Diseño, y Arquitectura. Universidad Iberoamericana-Puebla. Km 3.5 Carr. Fed. Puebla-Atlixco Puebla,Pue., MEXICO

Abstract

Planning and implementing previously designed activities can support the development a specific designed microworld. These activities must endure certain mental operations in the students toward a more meaningful apprenticeship. In order to reach this goal, in this paper is proposed the use of a method —such as the transcendental method discovered by B. Lonergan— to help students find the new knowledge more meaningful. This way the students might use this new knowledge in different areas.

In this paper, the transcendental method is used as a tool to identify the mental operations made by the student. It is illustrated with a simple example where a student must draw on paper a shape, must write the Logo code that might graphic the shape. Later the student is asked to type the Logo instructions to see if the shape was described properly. He is given time to review his instructions and find which were the errors. Finally he is asked to fix the errors and graphic the same shape.

		
<p>The original shape in paper</p>	<p>Resulted shape of her programming</p>	<p>Resulted shape after debugging her programming</p>

In order to find the errors in her programming and to construct the shape properly, the student used her finger as the turtle:



Keywords

Transcendental Method, Collaborative Learning, Meaningful Apprenticeship, Microworlds, Logo

Introduction

In this paper is presented how the transcendental method discovered by Bernard Lonergan can help structure the activities to be performed inside a microworld to help turn the knowledge into meaningful apprenticeship.

It starts with the explanation of the transcendental method and the importance on the planning and design of the activities that are going to be resolved inside the microworld.

Afterwards is suggested a “kind of questions” made for each level and some hints to develop better collaborative groups.

Brief explanation of the transcendental method of Bernard Lonergan

Beyond doubt, when a student is exposed to a teaching-learning situation there is a series of operations going on inside the student's mind. This mental operations embrace from very simple operations —sensory— to more complex mental operations that helps the student makes a responsible decision. All this mental operations are focus toward the idea of an object and are performed by the student. He must be conscious of these operations making an “insight”, where the content is objective and conscious (Grace, 1996).

These operations occur in different conscious levels and have various intentions. It is important to distinguish each level (Grace, ibid p.2).

1. The empiric level is the level of the senses.
2. The intellectual level is the level of the investigation, the understanding, and the expression.
3. The rational level is the level where the student thinks and judges rationally the accuracy or inaccuracy of the information provided and makes the considerations according to his principles.
4. The responsible action level is the stage where the student applies what he knows to make a decision more accurate with what he is.

These levels are also known as of “self transcendence” what suggests that there is a group of mental operations through which the individual can transcend from the lonely being that interacts with the world beyond him to more concerning and caring being (Dunne, 2006).

These conscious operations are summarized as follows:

1. To experiment through senses.
2. To understand through assimilating.
3. To judge through thought based on rational and moral judgments.
4. To decide the ongoing action and perform it.

Also, these conscious and intellectual operations can be identified through an analysis of the objects constructed by the students.

It is important to mention that the individual transcend from one knowledge level to another; this means he cannot skip any level.

- Next it is explained how this transcendental method can support the activities designed to be used inside a microworld.

The use of the transcendental method as a support on the design of the activities created to be used inside a microworld

The constructivism is an epistemological theory that emerges from the basis that knowledge is the development of mental structures in the individual. This structure regenerates as the individual obtains information and relates it to the knowledge he already had through interaction, this way the knowledge is not discovered but constructed from the individual perspective, his particular way of thinking and his interpretation of the gotten information. This implies to acknowledge that each person has his very unique way of learning.

In order to these cognitive structures be connected with other areas of knowledge, it is necessary to make them meaningful. The knowledge of the world is constructed and reconstructed constantly through the personal experience. The knowledge is not to be given, decoded, retained and reapplied but it is a personal experience that is progressively constructed and transformed according to the experienced events.

Microworlds, according to the point of view from Hoyles and Noss (1996), can express ideas (mental objects) in a concrete way (like visible objects) allowing the computer to become as an expressive media —so characteristic in the constructionist learning environments. According to Kent (2000) Hoyles and Noss use the term “auto-expressive” to emphasize the way in which the computational environment can turn into a place where mathematical actions can be developed and thought. When these environments are carefully design —for working in a collaborative way— it is easier to create a webbing activity among different knowledge areas. To realize if the webbing is taking place, certain methods can be used to evidence the different mental actions made by the individual during his constructionist learning process. These actions (see, hear, think, calculate, etc.) are known in the transcendental method as mental operations.

The teacher can guide mental operations through previously designed activities.

These activities can also help develop a learning collaborative environment (so necessary in a microworld) working as tasks to generate a *positive interdependence*.

Teacher must visualize the moment in which these activities will be applied to students and the duration for each one.

Each activity must include questions that will help the student to transcend from one conscious knowledge level to the next one. The teacher has to consider how much the student is paying attention during his exposition where he gives the relevant information that will help the student to have the insight where the information is analyzed and understood. Once he comprehends (and builds) the knowledge he can make rational judgments upon that information and use it on his convenience. The student will decide how and where to use it. In this moment he makes principle judgments that are inherent to the human being and that can be reflected into a better collaborative group.

According to Vygotsky (1978), students are capable to carry out on higher intellectual levels when they are asked to work in collaborative situations. The diversity in terms of knowledge and experience contributes positively to the apprenticeship process. They feel more responsible of their own apprenticeship and the apprenticeship of their partners.

The number of participants for each group depends on the personality, age and interest of each participant. It is suggested that the students are motivated to work at least in pairs. Even though each student has his own materials it has to be asked to deliver only one answer sheet to develop a positive interdependence. The positive interdependence refers on having a tangible support object that forces the group to have perspective where the only way to succeed is by working as a team (Hill & Hill, 1990).

To limit the resources given is a way to create positive interdependence —the dependence among each other— because it forces the students to work together to finish the task.

The educative practice fosters this working process and the exchange of ideas in the classroom, teaching during the process, values that are considered educative by the humanity. If these values were or weren't taught is something that only the student can tell at long term and depends on attitude of the student. The collaborative learning is a situation that provokes an adequate environment. It is neither a method nor a mechanism and will only be successful when working as a whole (Castellanos & Sacristán, 2005).

As reported in the papers presented during Eurologo 2003, and 2005 as a result of previous investigations, the activities must be short (Castellanos & Sacristán 2003, 2005). It seems that when the activity lasts long, the students felt tired and disappointed, but even though the same activity was presented as a group of activities, they felt as a success when they finish each one.

The different levels of the transcendental method in the learning through Logo

The activities can be designed embracing each of the levels proposed in the transcendental method. In this part I am proposing general questions that each teacher can adjust according to the specific needs of the microworld that he or she is developing. These questions ponder the mental operations made by the student. These mental operations evolve into complexity. For a better understanding, some of these mental operations developed in each level are mentioned, such as the questions that might help to transcend each stage and the kind of answer it will have according to Rugarcía, de la Chausse and Diosdado (2005).

In order to illustrate the way a student might transcend through this method while working with Logo, in the following example each level is identified, with the mental operations.



The activity consisted in several stages: during the first stage, the student has to draw with pencil in a paper a little house. In the second stage they have to write down the Logo code that he (or she) thought that will graphic the little house without the use of the computer. During the third stage the student typed the code of the second stage. On the fourth stage the student has to fix that programming to make it graphic the little house of the first stage.

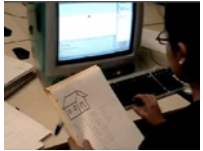
The goal is to make the student translate the description of an abstract object into Logo code, fixing it until they decided that the graphic they obtained is the one they draw at the beginning.

Sensorial level:

This level is when the student is "paying attention". It responds to the data input through the senses. The mental operations that the student makes are seeing, tasting, touching, smelling, remembering, fillings developed towards the data perceived —of repulsiveness or agreeableness— among others. The questions should be around the data. The answer will be the data that was captured by the student.


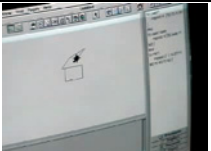
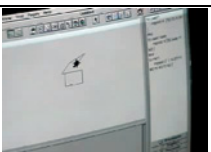
In this stage the exercises to be resolved are to guarantee that the data was input through the senses and that will be remembered.

Sensorial level		
Reads		1. The student took out the material that she prepared as homework.
Recognizes		2. The student recognizes the figure to be reproduced: the little drawing, she made, of the house.

Types		3. She typed the Logo code she thought would construct the graphic of the little house. That code was written as part of her homework, without the use of a computer.
-------	---	---

Intellectual level:

This level is the stage that goes from the understanding to the comprehension of the data. The mental operations that the student makes are asking, imagining, understanding, conceiving, formulating, knowing what can be done with the data, understanding what is it good for, or for what it is not that good, among others. The questions should be: what is it? How is it? Why is it? Can it help with something? It must have exercises, which results show that the student understood how to apply the data acquired in the sensorial level and comprehends (in a more profound way) when to use it. In order to have this comprehension, the student makes an insight of the data, that allows him (or her) to translate it on his or her own words, being allowed to explaining it and knowing in which situations can it be applied and in which situations it is not convenient. And the answer will be given through the insight of the data; it will illustrate what the student understands.




Intellectual level		
Compares		1. She recognizes her mistake: the graphic obtained has nothing to do with the graphic she has draw.
Analyses Decides		2. She analyzed her programming and decided to start all over again without using what she had already programmed. Also she decided to start the figure and starting in the corner of the base of the main rectangle. 2b. She started programming step by step.
Programs Executes Debugs Programs Executes		3. Whenever she made a mistake, she immediately corrected it until she finished.

This level maybe the stage in which the educators have put more attention to, but as the transcendental method establishes, in order to get to this level of consciousness of knowledge, the student must have had transcended in a proper way from the sensorial level. And it seems that we have sometimes forgotten about what comes next with that information, where the judgment rules the actions to be done.

Rational and principle judgment level:

This level is the stage where the student judges the data. When the student arrives to this stage, he or she has already the information, knows how to use it, but has to decide in a conscious way if that new knowledge is not only useful but if he is capable to use it in a good way. For example, that the student is able to compare one process to obtain an answer upon other or if the student already has an answer, to make (him or her) look for a different way to obtain it. Finally, the student will be able to decide which proposal is better. The mental operations that the student makes are thinking, reuniting proves, pondering, evidencing, and judging all the work that can be done with the data among others. The questions should be: is this that I understood the way it should be understood? Is it true? Is it not true? And the answer will be given through the rational and principle judgment of the student and will demonstrate if something was or not understood properly and if something of good can be made with it.

In order to be able to answer these questions, it is necessary to include in the activities, an exercise where it is asked for different ways to get the same answer. It is necessary to develop “debugging” as a common practice. To make the student define the facts that makes one answer better than another. These facts will help establish principle judgments that will help to make a better decision (e.g. how many lines are programmed in the procedure, how accurate it is, or if the “game” that is being programmed by the students has a negative or a positive connotation).

Rational level (of rational and moral judgments)		
Debugs Programs Executes		1. It was necessary for her to go back to the previous level.
Compares	<div>   </div> <div> Graphic made with Logo Graphic drawn in paper </div>	2. She recognizes the figure as “the same” that she was looking for (the little house drawn by herself).
Recognizes	<pre>fd 40 rt 90 fd 60 rt 90 fd 40 rt 90 fd 60 rt 90 pu fd 40 lt 90 pd fd 10 rt 135 fd 40 rt 45 fd 60 rt 45 fd 40 rt 135 fd 20 rt 40 fd 24 rt 90 fd 15 lt 180 pu fd 15 pd fd 20 lt 30 pu fd 41 lt 101 pd fd 40 lt 90 fd 40 lt 180 pu fd 40 rt 90 pd fd 10 rt 90 pd fd 20 lt 90 fd 10 lt 90 fd 20 rt 90 pu fd 20 rt 90 fd 20 pd fd 10 lt 90 fd 10 lt 90 fd 10 lt 90 fd 10 rt 180 pu fd 15 pd fd 10 rt 90 fd 10 rt 90 fd 10 rt 90 fd 10</pre>	9. She recognizes some repetitions in her programming.

Action level:

The student acts in consequences of the judgments made on the previous level, for it will be a responsible action. In this stage the student has the arguments to decide which proposal is better, and must argue with his partners, inside the collaborative group, the most important facts that made him consider one proposal upon another.

Action level		
Efficient	<pre>repeat 2 [fd 40 rt 90 fd 60 rt 90] pu fd 40 lt 90 pd fd 10 rt 135 fd 40 rt 45 fd 60 rt 45 fd 40 rt 135 fd 20 rt 40 fd 24 rt 90 fd 15 lt 180 pu fd 15 pd fd 20 lt 30 pu fd 41 lt 101 pd fd 40 lt 90 fd 40 lt 180</pre>	1. In the previous level she recognizes the repetitions and in this level she decides to debug her programming.

<pre> pu fd 40 rt 90 pd fd 10 rt 90 fd 20 lt 90 fd 10 lt 90 fd 20 rt 90 pu fd 20 rt 90 fd 20 pd fd 10 lt 90 fd 10 lt 90 fd 10 lt 90 fd 10 rt 180 pu fd 15 pd fd 10 rt 90 fd 10 rt 90 fd 10 rt 90 fd 10 </pre>	
---	--

Although the student didn't make the exact shape, she decided that she has reached the goal. There are clear differences in the windows, the roof and the door. These differences were not important to her. She consciously didn't fix them. Her action was responsible.

Conclusions

There are two important facts to be consider for having success: the content must be presented as meaningful knowledge to the student (meaningful apprenticeship), and to endeavour that the knowledge is built by the student himself, that he establish the connections among the new and the previous knowledge to facilitate the transference of the new concepts to other areas of knowledge.

When applying the activities, it is important to establish the initial conditions (to work individually, in pairs or in small groups). To have control upon the data that will be provided to the students. Establish the rules for discussions and watch over all the interactions inside the microworld.

The activities must be short and designed according to any method that can provide a guide to make the student turn information into meaningful apprenticeship. The transcendental method discovered by Lonergan can do so. For each conscious knowledge level the teacher can create questions that will help the student transcend through each step. This way the activities may support with accuracy the microworld. In the next table, the mental operations were identified in each level of the transcendental method.

Level	Mental Operations
Sensorial	Read
	Recognize (the shape)
	Type
Intellectual	Compares
	Recognizes (mistakes)
	Decides
Rational judgment	Fix (the programming)
	Executes
	Recognizes (the success or failure, and thinks of the activity as finished)
	Recognizes (some repetitions and mistakes that were not determinant)
Moral Judgment	Decides (to keep or not debugging the programming)
Responsible Action	Fix and debug

References

Castellanos, E. & Sacristán, A. (2003). *Logo as a tool for introducing Fractal Geometry to students of Graphic Design*. In Proceedings of EuroLogo 2003. Edited by Cnotinfor, Lda. pp. 306 – 316.

Castellanos, E. & Sacristán, A. (2005). *A Fractal Geometry Logo based microworld for Graphic Design graduate students*. In Proceedings of EuroLogo 2005. Edited by G. Gregorczyk, A. Walat, W. Kranas & M. Borowiecki OEliZC. pp. 23 – 32

Dunne, T. (2006). *The Internet Encyclopaedia of Philosophy: Bernard Lonergan (1904-1984)*. Consulted on August 12, 2006 at: <http://www.iep.utm.edu/l/lonergan.htm#H2>

Grace, R. J. (1996). *The Transcendental Method of Bernard Lonergan*. Consulted on August 23, 2003 at: <http://www.lonergan.on.ca/reprints/grace-method.htm>

Hill, S. & Hill, T. (1990). *The Collaborative Classroom: A guide to co-operative learning*. Heinemann. Nueva Hampshire, Australia

Kent, P. (2000). *Some Notes on Expressiveness*. Consulted on October 5, 2003 at: <http://www.lkl.ac.uk/came/events/weizmann/Expressiveness-notes.pdf>

Noss, R. & Hoyles, C. (1996). *Windows on Mathematical Meanings. Learning cultures and computers*. Kluwer Academic Press. UK

Rugarcía, A., de la Chausse, M.E. & Diosdado, B. (2005). *Programa de Desarrollo Integral*. Ed. UIA-P, Puebla, México

Vygotsky, L. (1978). *Mind in Society: The development of higher psychological processes*. Harvard University Press. Cambridge, EE.UU.

Comprehension of OOP Concepts using Dolittle in elementary school

JongHye Kim, *jonghye.kim@inc.korea.ac.kr*

Graduate School of Computer Science Education, Korea University, South Korea

SookKyoung Choi, *sookkyoung.choi@inc.korea.ac.kr*

Graduate School of Computer Science Education, Korea University, South Korea

HanSung Kim, *hansung.kim@inc.korea.ac.kr*

Graduate School of Computer Science Education, Korea University, South Korea

JeongA Jang, *jeonga.jang@inc.korea.ac.kr*

Graduate School of Computer Science Education, Korea University, South Korea

WonGyu Lee, *lee@comedu.korea.ac.kr*

Professor, Dept of Computer Science Education, Korea University, South Korea

Abstract

This article describes the research on Object-Oriented Programming (OOP) learning of the elementary students. The elementary school students were exposed to OOP through Dolittle, which is an educational programming language. Teaching concepts of OOP to elementary school students need high cognitive states because most students have concrete operational period. Our teaching approach focused on OOP concepts and it also featured programming-language oriented thinking method. Evaluation methods were both qualitative and quantitative. At the end of tutoring, most elementary students understood the basic principles of OOP and they naturally developed procedural thinking ability. The two main contributions of this research are: (1) Dolittle is appropriate educational programming language for teaching OOP concepts to elementary school students, and (2) students who are in the concrete operational period can achieve abstract thinking ability through learning OOP.

Keywords

OOP (Object-Oriented Programming), Dolittle, Educational object-oriented programming language, Procedural thinking ability, GALT (Group Assessment of Logical Thinking), Elementary school student

1. Introduction

In December 2005, the "Guidelines for Information and Communications Technology Training in Elementary and Secondary Schools" was published in Korea (Ministry of Education and Human Resources Development, 2005). The biggest change among revised ICT education curriculum was that it focused on learning computer science principles. There exists a virtual consensus that teaching fundamentals of computer science should be focused on scientific principles, problem-solving and project development skills, rather than on specific artefacts such as programming languages and operating systems (Gal-Ezer, Beer, Harel and Yehudai, 1995). One of the education curriculum goals from Grade 3 to Grade 4 is improving problem-solving methods. Students in Grade 5 start to learn programming using programming languages. Abstraction dimension, such as mapping from the problem domain to the programming domain, is important for beginners (Detienne, 2001). However, teaching concepts of OOP (Object-Oriented Programming) to beginners need high cognitive states (Hadjerrouit, 1998). Understanding abstraction dimensions for elementary school students is not easy because they are in the period of concrete operations which is one of the stages of cognitive development formulated by Piaget (Wadsworth, 1989). When elementary school students learn programming with high-level programming language such as Java, they spend a lot of time learning syntaxes instead of focusing on improving problem-solving skills (Dan Aharoni, 2000). Therefore, this study focuses on teaching the concept of OOP to elementary school students and examines procedural thinking ability by using Dolittle, which is an educational OOP language with syntax minimization (Susumu Kanemune et al., 2004). In addition, it examines whether elementary school students can understand concepts of OOP concepts and apply these concepts in real-life problems.

2. Related work

Piaget stated that the concrete operations stage of cognitive development can be found in children ranging from 7 to 11 years old. Although this age range is not fixed, children around these ages start developing logical operations (Wadsworth, 1989). However, logical operation in the period of concrete operations is not sufficiently developed such that children who are in this period can deal with only concrete objects. In other words, children can not understand abstract concepts and they can not solve language problems. This study focused on elementary school students and it examined stages of cognitive development by evaluating students' logical minds before teaching programming. Instructors agree that there is a need for a different pedagogical approach for teaching OOP (Bishop, 1997). In introducing objects, the following concepts have to be treated: the object state, changing the object state, and the way objects relate to each other (Woodman, M. & Holland, S, 1996). It is better to hold off the study of control flow, complex statements and advanced OOP topics such as inheritance (Stein, 1997). Some authors gave a detailed list of students' misconceptions regarding OOP concepts. For example, students only view programming as a computational tool but not as a changing attributes; confusion between an object and an attribute (Holland, S. Griffiths, R & Woodman, M, 1997). To account for these misconceptions, educational programming software called Dolittle will be used in this study to evaluate how appropriate the software is for elementary school students when they learn OOP concepts.

Dolittle – Educational Object-Oriented Programming Language

In order to select an appropriate programming language based on elementary school students' cognitive development level, Dolittle was tested. Dolittle is an educational OOP language and a prototype-based language which copies objects without the concept of class. It is a text-oriented

language and it consists of simple structure as well as easy error treatment. Figure 1 shows a sample Dolittle program that is written in both Korean and English.

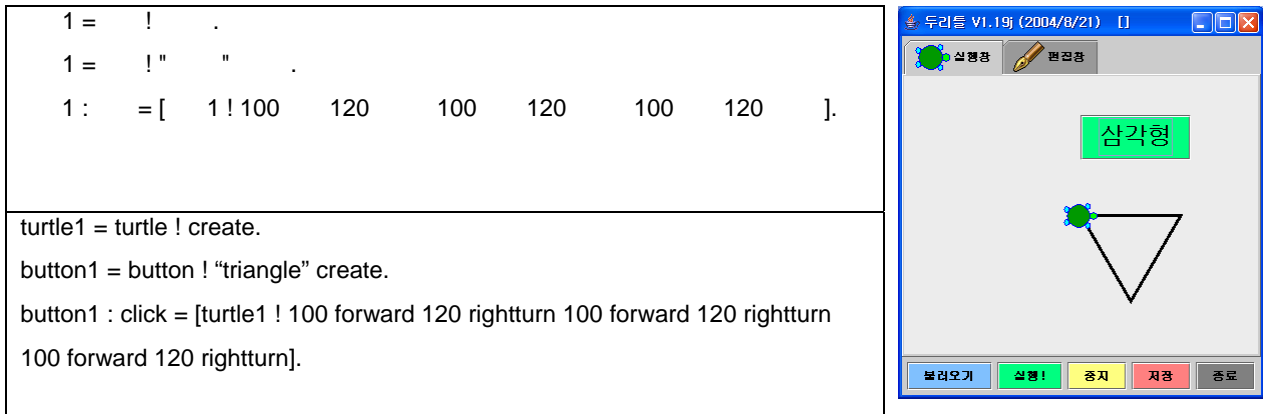


Figure 1 A sample Dolittle program and its execution

Dolittle is known for its syntax minimization. As an example, ‘turtle1=turtle ! create.’ can be seen from Figure 1. Dolittle programs call on objects using “!”. This simple statement sends a “create” message to the prototype object “turtle” to let it clone. The period “.” indicates the end of the statement.

In programming, creating an object using Dolittle is different from other programming languages such as Java. In Java, users create objects using classes while users in Dolittle simply copy an object instead of creating them. For example, ‘button1 = button ! “triangle” create. button1 : click = [turtle1 ! 100 forward 120 rightturn 100 forward 120 rightturn 100 forward 120 rightturn].’ can be seen in Figure 1. This statement creates a button object called “button1”. Next statement assigns a block to the “click” attribute of the “button1” object. The “click” is the method executed when the button is clicked. When you click the button, the execution message is sent to the “turtle1”, and the “turtle1” executes the block. As a result, the screen shows the triangle.

Dolittle is easy to elementary and middle school students because it is in their first(mother) language (Susumu Kanemune et al., 2004). Five under-achieved students who were in Grade 7 got interested in programming after learning programming with Dolittle (HeeKang, 2005). 5 under-achieved students who were in Grade 9 improved their programming skills after learning programming with Dolittle (HeyMinGil, 2004). In other study, 64 Grade 11 students were divided to 2 classes and were taught about logical circuit. One class was taught with an aid of Dolittle, while the other did not. A class that learned logical circuit with Dolittle showed high interaction and improved their problem-solving skills and long-term memories (SuKyungYoo, 2005). These results proved that Dolittle can reach its full effectiveness in secondary education in short period time.

3. Method

Phenomenographic research methodology describes the variation of understanding of a particular phenomenon found in a group of people. The unit of phenomenographic research is a way of experiencing something, and the purpose of the research is the variation in ways of experiencing phenomena (Anna Eckerdal, Anders Berglund, 2005).

Lessons

This research used a phenomenographic research methodology (Anna Eckerdal, Anders Berglund, 2005). Table 1 shows the lesson plan for teaching OOP concepts.

Lessons	Subject	Contents	OOP Concepts
1,2	Understanding Objects	<ul style="list-style-type: none"> Pre-interview Introduction of Dolittle Object creation and Modification 	Object Method
3,4	Understanding Method and Instance	<ul style="list-style-type: none"> Understanding Method Understanding Instance 	Method Instance
5	GUI programming	<ul style="list-style-type: none"> Figures creation and modification Button, Label, Field creation and modification 	Encapsulation
6,7	Game programming	<ul style="list-style-type: none"> Timer creation and modification Collision method modification 	Modularity
8	Final project	<ul style="list-style-type: none"> Final programming Evaluation(OOP concepts and Procedural thinking ability) 	

Table 1 Lesson Plan for teaching OOP concepts from Grade 3 to Grade 6

At the beginning of the 1st class, a pre-survey which asks about programming using experiences was given. After the survey, participating students used Dolittle and moved objects inside the program. They learned how to operate and correct errors in the program. During the 3rd and 4th class, the students learned methods and instances by operating the object. In 5th class, the students made figures and used methods for moving those figures. Additionally, the students learned event programming which relates objects inside programming with GUIs such as buttons. The next 2 classes taught the students how to program games using timer and collision method. For the final class, the students were asked to build programs based on what they have learned from previous classes.

A total of 8 lessons were held with 12 male students ranging from Grade 3 to Grade 6. Dolittle was used for 1 hour in 8 different lessons. Final project consisted of game programming in order to develop an interest for programming. During the lessons, the teacher told students the name of "Object" but did not tell students concepts of OOP. The outcomes of each process were then uploaded to the website and these results were shared among students. During the final lesson, students were asked to create their own programs based on the training they had received. This survey used the rating scales and contained evaluation process after each lesson. Formative assessment was made through quizzes at the end of each lesson.

Evaluation

The tests included qualitative and quantitative evaluation. Evaluation consisted of pre-tests, process tests, and post-tests. Pre-tests used Group Assessment of Logical Thinking (GALT) test and surveys (Shayer and Adey, 1981). GALT test evaluates student's cognitive development level. Survey evaluates whether elementary students learned programming and/or OOP concepts. Also, it evaluates whether they know the meaning of flow-chart. During Dolittle lessons, phenomenographic research was used to evaluate student's satisfaction. Post-tests evaluate OOP concepts and procedural thinking ability. Table 2 shows the evaluation criteria, which are

OOP concepts that are likely hard for novices to understand (Noa Ragonis & Mordechai Ben-Ari, 2005).

Problems relating to OOP concepts should be used in real-life problems, not just relating to coding or computer system. Problems relating to procedural thinking ability consist of “Elevator’s Operation Sequence” and “Vending Machine Operation”. The evaluation used in “Elevator’s Operation Sequence” was objective, while “Vending Machine Operation” was subjective. In addition, “Vending Machine Operation” used flow-chart as an additional method.

Domain	Evaluation Criteria
OOP Concepts	Object Creation
	Identification of Objects
	General understanding of Method
	General understanding of Instance
	Understanding Encapsulation
	Understanding Modularity
Procedural thinking ability	Elevator’s operation sequence, Vending machine operation

Table 2. Evaluation Criteria about OOP concepts and procedural thinking ability

4. Results and Discussion

Pre-tests

The result of pre-tests shows that most students belong to concrete operational period and there is no one who understands the OOP concepts. The results of tests for checking students' initiative learning ability can be divided into GALT and questionnaires. GALT measures 6 kinds of logical thoughts such as conservation logic, combinational logic, proportional logic, probabilistic logic, correlation logic, and controlling variables logic. Table 3 shows the result of cognitive development type of the students.

	concrete operational period	transitional period	formal thinking period
Number of students	8	3	1
	Grade 3 - 5	Grade 5,6	Grade 6

Table 3 Type of cognitive development after GALT test

As shown in Table 3, most of the students belong to concrete operational period in which their abstract conception is not formed completely yet. Only one student was in the period of formal thinking.

As a result of questionnaires, students never heard of OOP concepts such as objects, encapsulation, and modularity. It was also found that students' never heard of the term 'flow-chart'.

Process tests

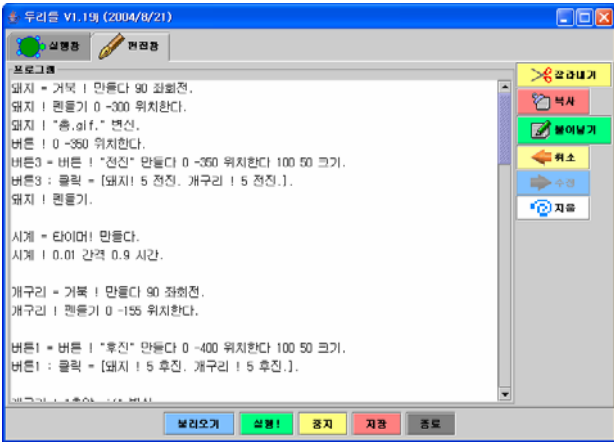
After analyzing process tests, it indicated that the procedural thinking ability and abstract thinking ability of some concrete operational period students were improving.

According to many researches, Dolittle generally raises students interest in programming (HeyMinGil, 2004, SuKyungYoo, 2005). However, many students experienced difficulties from Lesson 6, which goal was to devise various problem-solving methods while they learn programming with Dolittle.

For example, Grade 3 students lost interest when they encountered "timer" object. Confusion arose from the fact that the "timer" was not a visible object. The results showed that Grade 3 students were no longer interested in OOP when they were required to 'think outside the box'.

In case of Grade 4 students, not all of them lost interest when taking process tests. Dolittle was an appropriate educational programming language tool for Grade 4 students without cognitive load. Program which was made by Grade 4 students showed that they improved procedural thinking ability as well as abstract thinking ability.

Figure 2 is about an example of game programming made by Grade 6 students.



```

Pig = turtle ! create 90 leftturn.
Pig ! penup 0 -300 position.
Pig ! "gun.gif" looks.

Frog=turtle ! create 90 leftturn.
Frog ! penup 0 -155 position.

Button ! 0 -350 position.
Button3=button ! "forward" create 0 -350 position 100 50 size.
Button3 : click =[pig ! 5 forward. Frog ! 5 forward.].
Pig ! penup.

Time = timer ! create.
Time ! 0.01 period 0.9 time.

Button1=Button ! "backward" create 0 -400 position 100 50 size.
Button1 : click = [Pig ! 5 backward. Frog ! 5 backward.].
        
```

Figure 2 A sample Dolittle program written by Grade 6 student

The transitional period level student made a monster game as shown in Figure 2. The goal of the game is to shoot the flying monsters. The program used 6 "turtle" objects and 8 "button" objects. In the program, the monster can change its movement when the monster was shot. Moreover, the objects were copied in order to make more monsters and each was assigned a different time value in the timer. Also, the students understood the procedural process of the game programming and gained more confidence when they completed their own programming.

Post-tests

The post-tests results indicate that most students understand OOP concepts as well as procedural thinking ability using Dolittle. A test about the OOP concepts comprehension was conducted. Figure 3 shows the quantitative evaluation result of students' understanding of OOP concepts. After evaluating the results, most students were able to comprehend OOP concepts.

Apart from 2 students, all students were able to solve the problem that was connected to real life objects, methods, instances, encapsulation, and modularity. The 2 students only understood the concept of objects, methods, and instances. These results show the limitation that is connected to real life and abstract thinking ability of computing. Figure 3 shows the students' average score of OOP concepts problems. Most students could distinguish instances and methods and learn the major concepts of OOP.

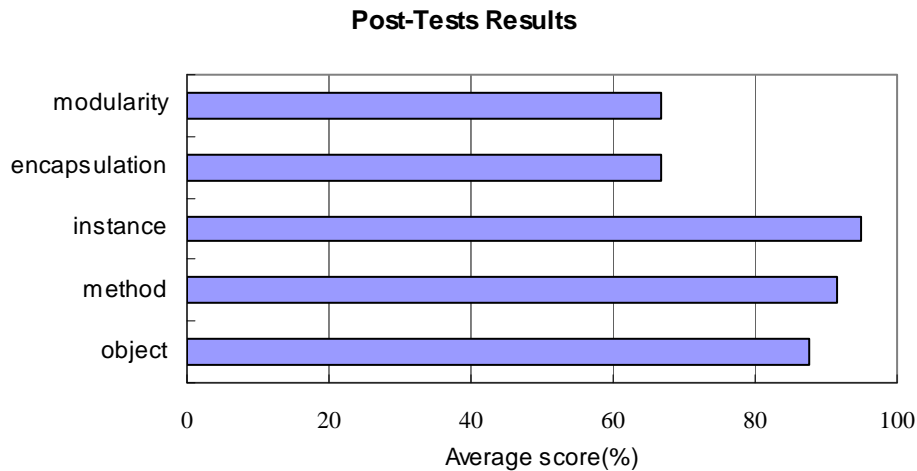


Figure 3 The result of post-tests is that most students understand OOP concepts.

After conducting OOP concepts test, procedural thinking ability test was given to the students. The evaluation of procedural thinking ability test shows that most students understood the method of problem solving. During the lessons, they did not learn programming education using flow-chart. Although students did not cognize the concept of flow-chart, it is worthy of notice that most students could solve using flow-chart. The students naturally understood the processes of the conditional and repetition though they could not understand the diagram.

Conclusions

This paper researched that learning computer programming with Dolittle improves the procedural thinking ability of students and helps them understand the concepts of OOP. Additionally, it showed that the elementary students gained confidence upon completion of their own computer program using Dolittle. Despite the fact that Grade 4 students are in concrete operational period, it was found that Dolittle could help elementary students understand OOP concepts and develop procedural thinking ability as early as Grade 4. The post-tests results provided further evidence towards the fact that abstract thinking ability and procedural thinking ability can be developed at early stage of education. However, the experiment in this paper was only based on 12 students. In order to further support aforementioned results, experiments with more participants are recommended for future study.

References

- Anna Eckerdal, Anders Berglund(2005), *What Does It Take to Learn 'Programming Thinking'?*. ICER'05.
- Bishop, J.M.(1997) *A Philosophy of teaching Java as a first teaching language*. ACM SIGCSE Bulletin, 29(1),pp140-142.
- Dan Aharoni(2000) *Cogito, Ergo Sum! Cognitive Processes of students dealing with data structures*. SIGCSE.
- Detienne, F(2001) *Software design-Cognitive Aspects*. London:Springer.
- Hadjerrouit, S.(1998) *Java as first programming language : A critical evaluation*. ACM SIGCSE Bulletin, 30(2), 43-47.
- HeeKang(2005), *Learning Motivation induction of under-achievement student using Dolittle*. Korea University.
- HeyMinGil(2004) *Application and Evaluation of Object-Oriented Educational Programming Language 'Dolittle'*. Korea University.
- Holland, S. Griffiths, R & Woodman, M(1997), *Avoiding object misconceptions*. ACM SIGCSE Bulletin, 29(10), pp131-134.
- Leron, U(1987) *Abstraction barriers in mathematics and computer science*. In J.Hilel(Ed.), *Proceedings of the third International Conference for Logo and mathematics Education*. Monteval.
- Ministry of Education and Human Resources Development(2005) *Guidelines for Information and Communications Technology Training in Elementary and Secondary Schools*. Ministry of Education and Human Resources Development.
- Noa Ragonis & Mordechai Ben-Ari(2005) *A Long-Term investigation of the Comprehension of OOP concepts by Novices*. Computer Science Education, pp203-221.
- Shayer, M. and P, Adey(1981) *Toward a science of science teaching : cognitive development and curriculum demand*. London : Heiemann educational Books.
- Susumu Kanemune, Shingo Fukui, Yasushi Kuno, Takako Nakatani, Rie Mitarai((2004) *Dolittle-Experiences in Teaching Programming at K12 Schools*.
- Stein, L.A.(1997), *Beyond objects*. Educations Symposiun, Conference on OOP Systems, Languages, and Applications, Atlanta, Georgia. Retrived October 30 2004 from <http://222.ai.mit.edu/projects/cs101/beond-objects.ps>
- SuKyung Yoo(2005), *Analysis on Learning Achivemnet, Instructional Design and development on 'Digital Logic Circuit' using Dolittle*. Korea University.
- Tony Jenkins(2002) *On the difficulty of Learning to Program*. 3rd Annual LTSN-ICS Conference, pp.53-58.
- Wadsworth, B. J(1989) *Piaget's Theory of Cognitive and Affective Development(4th ed.)*. N. U: Longman Inc.
- Woodman, M. & Holland, S(1996) *Form software user to software author:An initial pedagogy for introductory object-oriented computing*. ACM SIGCSE Bullen, 28(SI), pp60-62.

Radical bricolage: making the liberal arts coherent

James Edward Clayson, james@clayson.org

Department of Mathematics and Computer Science, American University of Paris

Abstract

Because of the very broad and fragmented nature of undergraduate general education requirements there is a need to help students find unity in this diversity. The search for coherence has led my institution, the American University in Paris, to introduce a series of freshmen courses called First Bridge that deliberately pairs professors from different disciplines to develop and teach a common course that explores the linkages between each professor's area of special interest. This paper describes my own experience teaching a First Bridge course called *Visual Thinking and Artful Seeing*, in which I represented the math and computer science department while my co-teacher, a painter, came from the art department. It was our intention to explore how different ways of seeing, the very act of seeing and the art of talking about seeing can help each of us begin to discover the commonalities that often lie behind seemingly different disciplines and their methodologies. My part of the bargain requires using Imagine Logo to gain access to different levels of seeing by building computer models to examine a range of visual artefacts and their inherent structures.

My co-teacher's role was to develop, through drawing exercises, the student's ability to cultivate a heightened sense of line, texture, light, colour, volume, position in space, movement, shape and how these affect the viewer. We agreed to adopt an approach that we call "radical bricolage". Bricolage is a French term denoting the art of constructing things from whatever is at hand, using all the skills and tools available. My partner and I broadened the idea to turn our students into "bricoleurs" who would bring their own resources to bear in whatever form was helpful towards progressing into the task at hand. Using an assignment from my section of the course I will show how student J used the notion of radical bricolage to investigate the major visual themes in three paintings. Two of these were accessible by deconstructing their geometrical shapes. The third painting presented a different challenge because it was a dynamic and fluid action painting and, therefore, did not lend itself to this kind of deconstruction. When he first began, student J had no idea how to approach his tasks. But as we walked, talked, sketched, modelled, exchanged ideas and feelings in class, certain avenues began to emerge. By bringing all his senses to bear, by practising the art of radical bricolage, J found links he did not know existed. Thus he had extended and enhanced his ability not only to see and read works of art but also, by extension, the greater world around him.

Keywords

liberal arts, general education, visual thinking, visual modelling, Logo

Constructionism ... shares constructivism's connotation of learning as "building knowledge structures" irrespective of the circumstances of the learning. It then adds the idea that this happens especially felicitously in a context where the learner is consciously engaged in constructing a public entity... **Seymour Papert**

[The] 'bricoleur' ... derives his poetry from the fact that he does not confine himself to accomplishment and execution: he 'speaks' not only with things ... but also through the medium of things **Claude Lévi-Strauss**

To know what I mean, I must see what I say. **James Blachowicz**

Constructivist faith

Papert's quote above is a kind of constructionist manifesto – rich in interpretive potential and rich in its ambiguity. I do believe it, but as a teacher in a liberal arts environment, I don't find this quote immediately useful. We need to know more about how that model construction happens. We know it happens, but where is it happening? Can we influence it? Can we speed it up; make it more sophisticated; more innovative? We need to construct an environment in which we can both watch our own modelling and that of others because their approach might be useful to us. We need to build visual models of our modelling process, of the modelling of ourselves, so that we/it can be seen and commented upon by people with whom we have emotional and intellectual affinities.

Over the last five years I have experimented with a new course, called *Visual Thinking and Artful Seeing*, that encourages students to do just this and to talk about doing it. I've had to jumble many of the traditional undergraduate approaches to make them far more multidisciplinary. I've had to challenge some of the sacred cows of liberal arts education (like the primacy of writing) in my university to achieve this. I like to label my approach: "radical bricolage" and, as the animator of this activity, I am the "radical bricoleur".

The nature of the bricoleur

The French word "bricoleur" is often translated into English as a do-it-yourself (DIY) person but I don't think this comes close to catching the full French meaning of the term. The bricoleur does not shop at DIY centres and doesn't generally work from plans. He uses *whatever is on hand for a task*, and while he remembers how things worked together in the past he also likes putting things into new configurations. He seems to generate original thought as much from physical play with objects as from his memory and inside-the-head cognition. In fact, he probably doesn't view these as separate activities.

The bricoleur has no shame, he is not coy, nor is he hesitant. He gets on with whatever task is at hand unburdened by a need for "perfection", knowing, too, that to do one thing may leave other things less finished. The work of the bricoleur is never closed, never finalized, everything is work in progress.

Is bricolage just thoughtful trial and error? Is there no place for theory or planning? For the bricoleur, planning is an emergent characteristic of his tinkering and iterative approach. In bricolage, the notions of trial and error and planning are not two ends of some continuum, they are not separate activities: they are two qualities of his craft.

Might a bricoleur's craft, his approach to his work, give us insight into how to go about modelling/exploring concrete situations? That is, does the way he works give us a medium in which to record his methods? I agree with Levi-Strauss: "the bricoleur speaks not only with things but through the medium of things." But how can we observe this medium of things? I suggest that a first step would be to radically extend our concept of "tools" to include all the bits and pieces the bricoleur touches and the methods he uses to arrange and manipulate them. We must watch and listen as well.

What is radical bricolage?

I have talked about the need to encourage students to tinker with problems computationally because of evidence that this can encourage them to build their own models of their world. But how do they do this, this personal work? How do they know how they do this? Can they see it being done, being constructed? Are they conscious of it and, if so, can they control it? Can they watch their sense-making, can they steer it? Can they, can we, catch ourselves in the act of meaning-making so that we can watch it happen? In short, can we verify our belief in personal model construction? Isn't this radical bricolage?

Radical bricolage, liberal arts and the American University of Paris (AUP)

I teach applied mathematics and computer science at the American University in Paris (AUP), a small, private, US-style, liberal arts institution in Paris. AUP has several small graduate programs but its major activity is its undergraduate programs of about 1000 students. All undergraduates must fulfil general education requirements in languages, science, mathematics, humanities and social sciences before they go on to specialize in one or more majors.

Six years ago a committee was formed to evaluate AUP's general education scheme and to suggest changes. The committee decided that the current method of general education was giving out the wrong message. AUP's wide range of diverse courses had the effect of fragmenting knowledge, whereas a liberal arts agenda should illustrate the complementary nature of all disciplines.

To reinforce this perceived need for more coherence, the committee suggested that AUP require entering freshmen to select one option from a list of linked courses, to be called First Bridge. Each of the two courses would be from different disciplines, say one from history another from anthropology, but assignments would stress the complementarity of the two methodological approaches to explore a set theme. The two instructors would visit each others classes and time would be set aside to reflect on the links between the courses.

I was eager to participate in the First Bridge experiment and found a colleague, Ralph Petty, in the AUP Fine Arts department who wanted to join me in building a common course to be called: Visual Thinking and Artful Seeing.

Visual Thinking and Artful Seeing

My syllabus description was the following:

This course explores computing skills and applications in a non-traditional way. Using the computer languages Logo we will construct computational environments to explore how we look, think and feel. We will move quickly into such complex modelling areas as: tile design, simulation of themes found in famous works of art, design of artificial life forms, alternative spatial systems, visual data analysis and concrete poetry.

The course exercises will be tightly scheduled to keep pace with the aesthetic and design work being done in the linked course, Beginning Drawing I. The visual vocabulary and skills learned in Drawing will enhance the model-building activity; and what is done in the computational arena will amplify the effects of working in traditional studio arts.

Course outcomes

At the end of this two-part course students should have:

- *increased their visual vocabulary and design skills*
- *designed and built relatively complicated visual models for a variety of applications*
- *learned to describe and explain in words how these models were conceived and built*
- *acquired problem-solving techniques suitable for many disciplines*
- *mastered the basic notions of programming so they are equipped to learn other computer languages and software packages.*

The real agenda

All of the First Bridge courses are cross-disciplinary yet based on reading texts, writing and research. Traditional assignments and exams are given and a fairly large research paper is required supported by library and internet research methods. My teaching partner and I, while agreeing that effective writing is a major skill, shared the belief that writing education, at least in text analysis and research paper mode, is overly privileged at the expense of speaking, listening, calculating and manipulating something non verbal, like images. We felt that a combination course of visual modelling with Logo and traditional drawing could encourage a more even and rich mix and would be very much in the radical bricolage vein. Drawing, especially, has been a neglected contributor to liberal arts.

Lessons from the drawing studio

Just what are some of the features from art studios that can benefit computer modelling?

First: drawing from the nude model is a standard exercise in drawing classes. The model sits or stands on a raised platform surrounded by chairs. Beginning students have a tendency to pick a spot and sit down in one place, draw from that one position and go on to produce one “finished” drawing. It doesn’t take them long, however, to realize the advantage of drawing sketches from several different vantage points. If the viewer changes position, so too do the lines, textures and shapes of the subject. Students come to sense that few drawings are ever finished; they are works in progress and means of interacting in a complex visual and emotional space. The important thing is to produce many sketches from different points of view.

Second: drawing from the model is hard for most liberal arts students because they get hung up on whether or not they are able to “accurately” reproduce what is in front of them. Using a technique called blind drawing, on the other hand, forces a student to focus on the subject – the nude – and not on the drawing. In blind drawing the drawers must not look at the sketchpad while they are drawing. Students are totally surprised to discover the results. Not only do they recognize the subject in what they have drawn, but they also are shocked to discover that their drawing has an emotional content. The lesson here is that if allowed to, we draw naturally.

But, what do we draw naturally? We draw both how we feel as well as the image. We, our bodies, in fact, are drawing the image that we see and feel. This is an enormously important insight. It shows that our conscious mind can get in the way of fresh seeing; we see only what we have seen and cartoons of noses are easier to draw than making sense of that new thing out there. Whether we attempt to draw the shape of flesh or rely on our cartoon image of the body we are drawing us seeing. We are catching ourselves in the act of seeing.

Third: drawings recapitulate the stages of drawing. In a way, they are drawings of the act of drawing; we watch them grow in front of us, flowing out of our drawing pens. Nothing is lost, if nothing is erased: drawings recapitulate the whole experience.

A sample assignment from our course on Visual Thinking and Artful Seeing

In this exercise I directed students’ computational eyes at abstract paintings and suggested a number of painters whose use of geometrical forms can be computationally explored. I asked that they explore three paintings: two of which were by the same painter. I also asked that they pick a painter whose geometrical objects were easy to see and to manipulate and another whose paintings were difficult to deconstruct into parts.

The goal of this exercise was NOT to produce a painting that is really like, say, a Malevich or a Kandinsky. Rather, the goal is to identify the themes that we see in these paintings: first by describing them in words and sketches and then by transforming these into computer models. This activity should enable us to see more clearly both how we look at a specific painting, and the painting itself. The gap between our simple modelling play and a real Malevich holds the mystery that attracts us. More often than not, it is in this first modelling phase when students discover that seeing is an iterative process and that verbalizing can facilitate their ability to see. Each informs the other.

For example, one of my students, J., began by selecting the following painting from Kasimir Malevich.



Figure 1. Eight Red Rectangles

Relying on the radical bricolage approach which brings all our senses to bear on this image, here is what he had to say about key themes and their emotional and visual impact on him.

J's word description of the painting

"Red rectangles float in a rectangle of textured space of ambiguous dimension: flat but with painty depth. This thick, off-white background space bleeds up through the objects. Are they porous, translucent? And where are they in this space? On the surface, or in it? The red objects are organized: the do not overlap, they come close but do not touch, they are oriented along the bottom-left to upper-right diagonal, smaller objects hang below the larger. The organization is taut – tight and controlled – but inherently unstable: the rightmost element touches the right edge, pushing against it, trying to extend the space it is in, two other rectangles "push" it ... but I feel that the assemblage must slip, might collapse"

J. translates his words into computer models

Just as the American Philosopher James Blachowicz has observed, "To know what I mean I must see what I say", J. started with seeing what he meant by saying "rectangles floating in a rectangular frame." The form of the procedures he wrote is not important, but what he chose to explore visually is. Here are some examples. Note that these are illustrations of ideas and themes described above. This exploration took J into areas that he found very interesting since he had a flair for math and could implement algorithms for placing rectangles of different sizes within a rectangle; first allowing them to overlap and then prohibiting this overlapping.

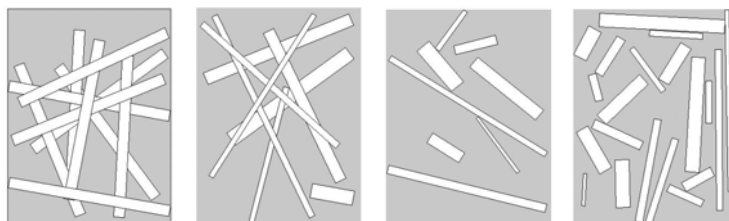


Figure 2. Some of J.'s visual explorations: Overlapping rectangles of the same size; Overlapping rectangles of different sizes; Eight non-overlapping rectos of different sizes; Twenty non-overlapping rectos of different sizes.

Computational explorations of themes J. saw in Eight Red Rectangles

Here is a sequence of non-overlapping rectangles inspired by the Malevich original, playing with similar colour, orientation and spatial placement



Figure 3. J explores the theme he saw in *Eight Red Triangles*

Comparison between one of J's images and the target image

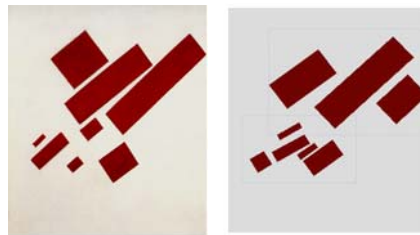


Figure 4. Malevich and J

How well did J do?

What lies between J's image and that of Malevich? When he compared these two images, J. suddenly realized that he hadn't noticed the artist's attention to texture as well as geometry. On the other hand, he was really thrilled with how he had modelled the idea of floating objects in space, on space, and how that "floatiness" could give flat canvas space a quality of depth. As a result, he was eager to explore other paintings by Malevich and he selected the one below.



Figure 5. *Supremetism*

J's word description for his second painting choice

"I see a strong, upside down, L-shape form, placed diagonally on the canvas. The two L legs hold parallel layers of randomly-sized rectangles. As in *Eight Red Rectangles*, the space is not really flat; there are several layers of objects floating in it. The bright blue stick seems to slide between two layers; and the small red and yellow rectangles are the top layer. There is also a strong V-shape in the painting against which the blue stick is slightly at odds: a blue blade slicing through the image. The whole design seems overly-tight and about to slide apart ... Finally, there is an almost-invisible rectangle, just above the lower salmon-coloured object, in the same shade as the background".

To know what I mean I must see what I say

The complexity of this painting forced J. to use his sketchbook much more this time. He began by sketching the entire painting and then decomposing the whole into themes and their placement. The first motif he isolated is a rectangular box that holds three other rectangles; the second motif is another box of rectangles that is at right angles to the first box. He realized that these two boxes form a kind of tilted, upside-down L-shape. The third motif is a blue rectangle that slides between the first two rectangular box motifs. Motif four shows the V-form of the tilting

L-shape. There are two other small objects in the painting: a small red square (seen in J's drawing) and a small yellow rectangle (not in J's sketch)

Entries from J's sketchbook

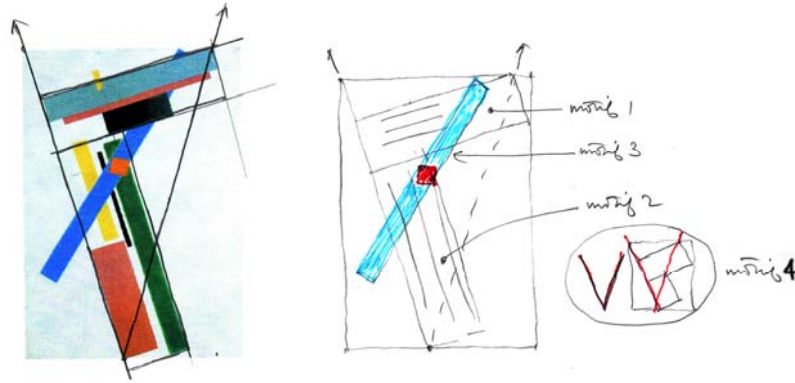


Figure 6. Overall plan and motifs as decomposed into four elements

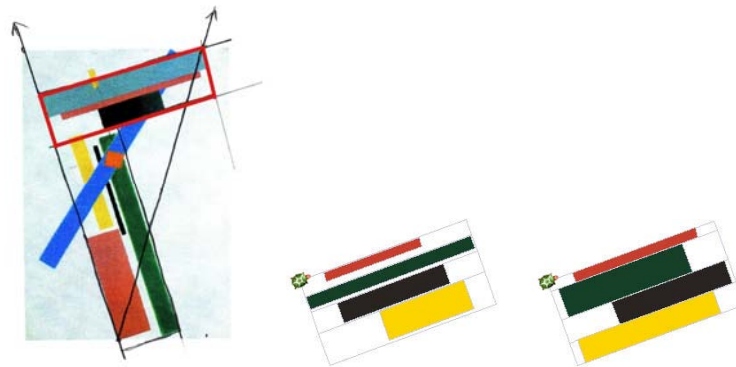


Figure 7. Motif one - the top rectangular box of rectangles. Outlined in red

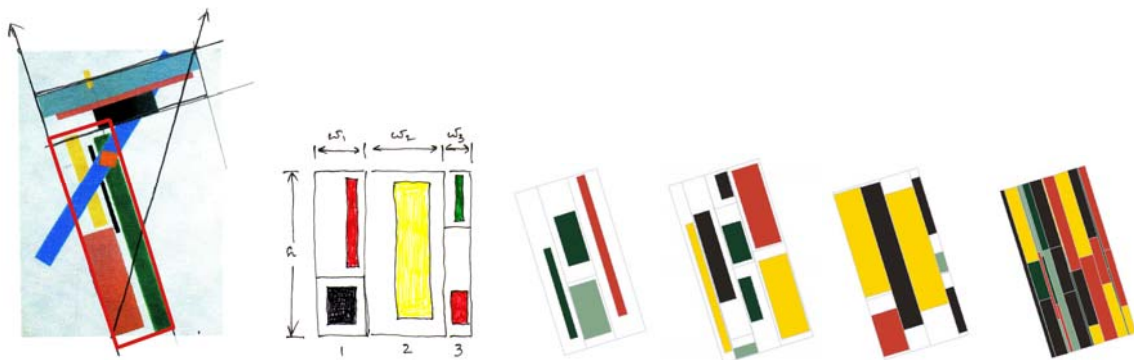


Figure 8. Motif two - the left side rectangular box of rectangles. Outlined in red



Figure 9. Motif three – the blue stick

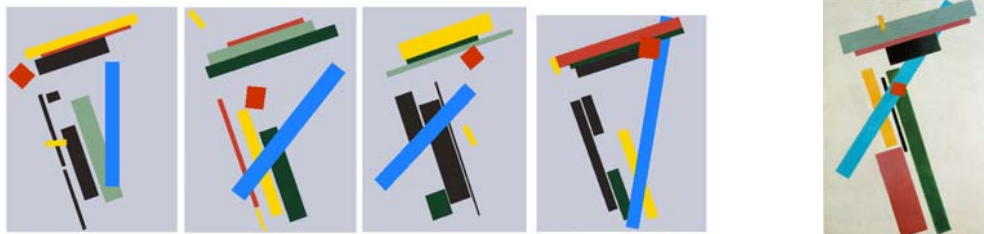


Figure 10. Computational explorations of themes J. saw in Malevich's Suprematism

Example 3

The last example is by a contemporary American painter, Brice Marden, whose themes are very different from those of Malevich and more difficult to deconstruct into distinct parts. The computer modelling is therefore more technically challenging. So how did J. proceed?



Figure 11. Tang Dancer

J's word description

I see a kind of quiet curving activity inside a rectangular frame. Black, dark green, yellow and red ribbons or trails mark paths that fill rectangular space. I very much like how the paths are not caught or stopped by the edges but move gently along the edges. I see soft sinuous waves: at ease in their curvings: exploring, playing with the rectangular.

The act of pathing, rather than a specific path, constructs this picture space. A kind of flat, wave space... equally foreground and background. Curvature ground. But, Marden called his painting: Tang Dancer and these curves do have a slow dancerly motion about them.

The theme, therefore, that I want to explore computationally, is not just this specific painting. The theme that I want to explore is a pathing process that will grow from my description of elements from this specific Marden work.

Sketching the theme



Figure 12. One loop from Tang Dancer

"I decided to select one of the paths, the red one, and sketch it independently of the others. I picked the red one, since it seemed to me to be the simplest of the four curves since it did not interact with itself. It was a sinuous curve that did not overlap. Too, it looked dancerly; the curvedancer was dancing in the space while filling it. The first idea, then, that I would like to explore is: generate a non-overlapping sinuous curve that fits happily inside a rectangular confine.

To know what I mean I must see what I say...

"First, I'll pick a point on each of the edges of the canvas randomly, then n random points placed within the canvas. Second, I'll connect these points in the order that they were generated by line segments then, third, I'll use this ordering for a spline shape. I don't know how my Logo implementation actually calculates splines so I'll explore them within my built-context. Fourth, I'll start with the first point randomly generated and then find the path based on the 'go to the next closest spot' heuristic. So here are the four scenes of these four notions based on one random selection of points."

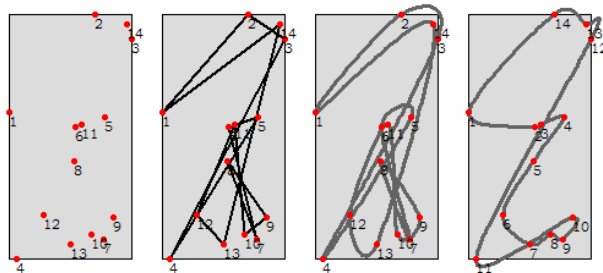


Figure 13. Random points and closest neighbour links

"I see that my nearest neighbour curve often has loops. I originally thought that maybe this curve would not overlap itself but I see that often it does and definitely so the more points I generate. So I'll explore the nature of this loopiness by generating a matrix of images:"

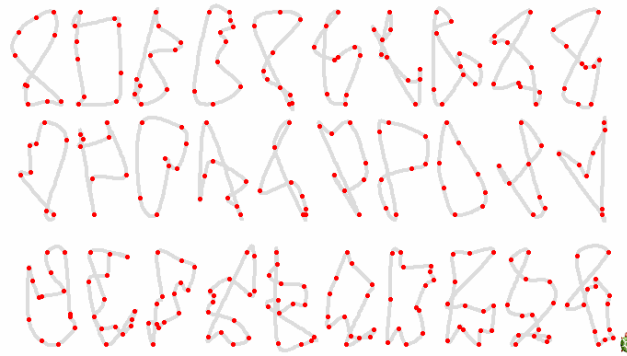


Figure 14. A suite nearest neighbour loops using ten points

J sketches and talks about unlooping

“Well, there must be some easy way to unloop a curve so I’ll sketch a concrete case with one loop and see if I could come up with a way to unloop that. I drew the picture below and my doodling suggested to me a method to try. If it works in a simple case I’ll extend the idea to curves with many loops

OK: find any two line segments that cross and exchange their end points like in my drawing. I’ll have to reverse the order of some of the points, the circled ones, and output a new order of points with the one loop removed. I’ll apply this method to multiple-looped curves, one loop at a time, to watch how it works. Sometimes my method fails and I’m not sure why. But I’ll work on that later ...”

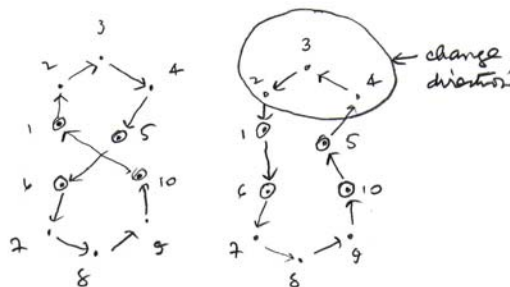


Figure 15. Sketch to think about unlooping one loop.

“Here are two examples of my method that unloops one loop at a time.”



Figure 16. Two examples of unlooping

“Then I tried more points ... ”

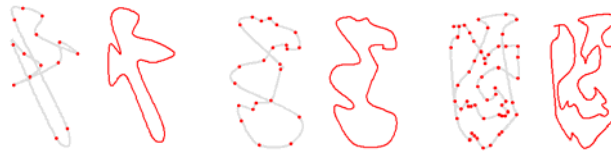


Figure 17. Three examples of unlooping with more points

“Then I decided to put my ideas into a graphic as I had done previously. I add my unlooping technique to the ones above and make sure the thing fits into the frame. The last image fills my delooped curve with colour.”

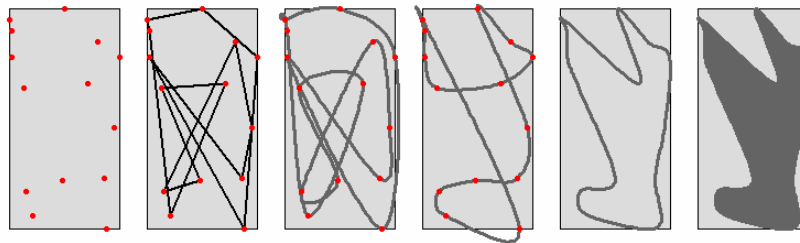


Figure 18. Unlooping with 10 points

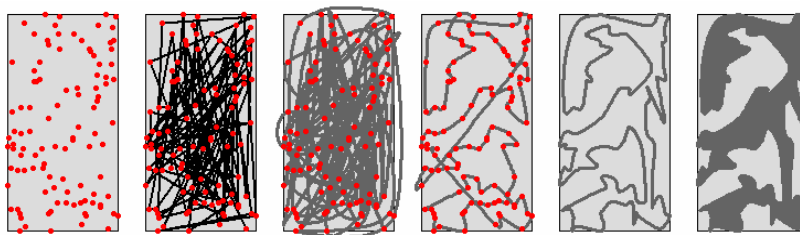


Figure 19. Unlooping with 100 points

Computational explorations of themes J saw in Tang Dancer

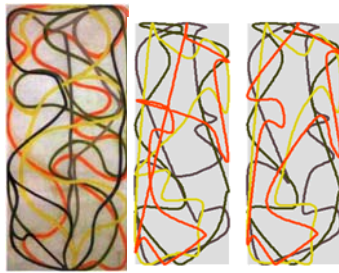


Figure 20. Marden painting and 2 of J's

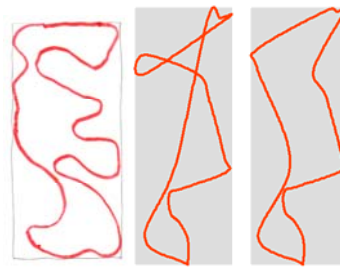


Figure 21. Marden Loop and 2 of J's

Theme extensions

"But what really struck my fancy were the odd images that formed when I filled my unlooped or partially unlooped shapes of many points with colour. That struck me as a kind of writing: *Tang* writing, perhaps."



Summary

What we have seen in the three preceding examples is an illustration of what I call "radical bricolage". In this approach, the student tinkers his way – using words, sketches, guided trial and error and computer modelling into an iterative exploration of complex visual tasks. Although liberal arts education has traditionally valued the student's acquaintance with a wide variety of disciplines and methodologies, it has often failed to re-integrate these diverse offerings within the local classroom environment. My First Bridge class at AUP, Visual Thinking and Artful Seeing, has tried to overcome this fragmentation. It requires that students bring all senses, skills and available tools to bear in constructing their own learning systems.

Further reading

- Albers, J. (1975) *Interaction of Colors*, Yale University Press, New Haven.
- Arnheim, R. (2004) *Visual Thinking*. University of California Press, Berkeley and Los Angeles.
- Blachowicz, J. (1998) *Of Two Minds: the Nature of Inquiry*, State University of New York Press, Albany.
- Clayson, J. (1988) *Visual Modelling with Logo: a Structured Approach to Seeing*, MIT Press, Cambridge.
- Clayson, J. (forthcoming) *The Computational Eye: Personal Excursions into Visual Thinking*.
- Garrels, G. (2006) *Plane Image: A Brice Marden Retrospective*, The Museum of Modern Art, New York.
- Gombrich, E.H. (1994) *The Image and the Eye: Further Studies in the Psychology of Pictorial Representation*, Phaidon Press, London.

Levi-Strauss, C. (1966) *The Savage Mind*, University of Chicago Press, Chicago.

Milner, J. (1996) *Kazimir Malevich and the Art of Geometry*, Yale University Press, New Haven.

Polkinghorne, D. E. (1988) *Narrative Knowing and the Human Sciences*, State University of New York Press, Albany.

Learning Logo at a high school: Constructionism versus Objectivism

Stéphane Norte, *snorte@ualg.pt*

UALg Informatics Lab, DEEI-FCT, University of Algarve, Campus de Gambelas 8000-117 Faro, Portugal.

Paulo A. Condado, *pcondado@ualg.pt*

UALg Informatics Lab, DEEI-FCT, University of Algarve, Campus de Gambelas 8000-117 Faro, Portugal.

Fernando G. Lobo, *flobo@ualg.pt*

UALg Informatics Lab, DEEI-FCT, University of Algarve, Campus de Gambelas 8000-117 Faro, Portugal.

Abstract

In the world of education, the traditional teacher-centric pedagogy is often supported by the traditional lecture format. This methodology however, may not be the best way to assist students to learn in a computer science classroom. We have made different activities in high school to understand students' difficulties to learn the Logo programming language. This paper describes the activities made in two classes which belong to a multimedia course with a specially adapted education syllabus and the results obtained.

The population of the study consists of students from a high school in the south of Portugal who never learned a programming language before, and that have exhibited many learning difficulties and behavioural problems in the past. Some of the students are quite old with respect to their school year. Under normal circumstances, a 10th grade student should be about 15-16 years old. The fact that some of them are much older than that is due precisely to their learning difficulties and behaviour problems, resulting in a large fraction of student failures at the end of the school year. Those students are required to repeat the school year once more. Indeed, a lot of students have failed several times.

Each group of students was exposed to Logo programming using different pedagogical models. We employ the traditional teacher-centric pedagogy (objectivism) in a class and apply the constructionist approach to another class. The study was made throughout 5 months with 180 minutes per week.

It is important that educational systems worldwide progress towards a better future with new generations and new philosophies of education. The research described in this paper supports the notion that the role of the teacher in the classroom needs to change, and that the possibility to learn by exploration must be given to students. Specifically, it encourages prospective teachers to adopt Logo under a constructionist learning environment. The teacher is essential in the learning process to provide motivation and to aid the students in the exploration of new skills.

Keywords

Logo, constructionism, objectivism, learning, education

Introduction

Education and learning in today's computer science classroom is usually based in two philosophical paradigms: objectivism and constructionism (Wulf, 2005).

Objectivism has dominated the field of education for several years. An objectivist educator believes that learning generally occurs when students listen to a teacher's explanation (Fosnot, 1996; Skinner, 1953). This traditional paradigm often manifests itself in teacher-centered and teacher-controlled classrooms. Under this pedagogical approach, the instructor serves as a subject matter expert who is the principal source of knowledge. Learning is seen as an information transfer procedure in which the instructor imparts his or her knowledge to the students. Constructionists, on the other hand, focus more on the experiences of the student. The basic and most fundamental assumption of constructionism is that knowledge does not exist independent of the learner, knowledge is constructed by experimental exploration, by making things (Kafai and Resnick, 1996; Papert, 1993; Papert, 1996).

With the development of new technologies in schools, teachers can follow a less traditional paradigm and employ a more constructionist approach (Gorp and Grissom, 2001). Unfortunately, some schools are not technologically and pedagogically prepared to encourage teachers to use these technologies on a frequent and sustained basis to enhance student learning. The challenge should be about using new approaches to help students to learn concepts and skills embedded in computer science education (Watt, 1988). It is not so much about powerpointing, spreadsheeting or word processing. The focus should be on teaching and learning strategies that make a difference in daily practice, on activities translating into stronger student performance.

This paper describes a research study done at a high school about learning computer programming by different pedagogical paradigms. Our work focuses on Logo because it is an excellent tool for learning introductory programming. Using Logo programming language is also an excellent introduction into the world of computer science education. We decided to employ the traditional teacher-centric pedagogy (objectivism) in a class and apply the constructionist approach to another class.

The paper is organized as follows. The next section describes the population that participated in our study. Then, we present our experience in teaching Logo, both under an objectivist and a constructionist environment. Following that, two sections, one on student feedback and another on student evaluation are presented. The paper ends with a summary and the major conclusions drawn from this work.

Research setting

The population of the study consisted of two classes from the 10th grade of a high school in Algarve, Portugal. Students from both groups never learned a programming language before and are known in the school for having learning difficulties, lack of concentration, and bad behaviour inside the classroom.

The two classes are referred to as the M class and the N class. Both belong to a multimedia course with a specially adapted syllabus. The M class is constituted by 19 students between the ages 16 and 20, with 7 girls and 12 boys. The N class is constituted by 21 students between the ages 16 and 23, with 6 girls and 15 boys. Notice that some of the students are quite old for being still at the 10th grade. Under normal circumstances, a 10th grade student should be about 15-16 years old. The fact that some of them are much older than that is due precisely to their learning difficulties and behaviour problems. The result is that a large fraction of the students fail at the end of the school year and have to repeat it again the following year. Some of the students have failed already several times.

Each group had lessons of Logo programming using different paradigms. In the M class we decided to use a traditional educational approach and in the N class we used the constructionist approach. The study was made throughout 5 months with 180 minutes per week.

Traditional environment

The M class began the introduction to Logo programming by listening to the instructor's explanation and responding to external motivation. We used screen capture video and narrated power point presentations to introduce the Logo turtle motion. The Logo interpreter employed in the beginning of the course was MicroWorlds free demo. We explained to the students how to use the MicroWorlds interface and we spent a lot of lessons describing all features.

After the explanation, each student tried the MicroWorlds environment with the aid of a teacher. Once they have learned the software we began to introduce the Logo vocabulary. With a digital projector we presented some vocabulary and simple procedures to draw geometric shapes. The students drew simple shapes with one defined by us with the turtles using simple Logo vocabulary (see Figure 1).

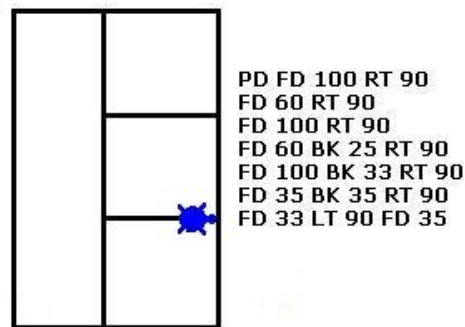


Figure 1 – First shape drawn by the students

When they had defined a few shapes on their own, students were challenged to create some common shapes such as square, triangle, pentagon, hexagon, heptagon and octagon. The students demonstrated a lot of difficulties to create the hexagon, heptagon and octagon. Only two students were able to make the shapes without any help. Figure 2 shows the shapes created by them.

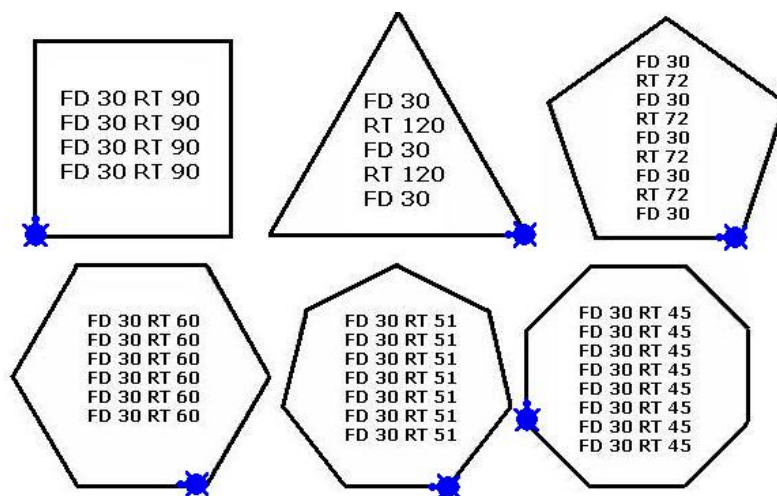


Figure 2 – Shapes created by the students on the M class

Once they had created some shapes, we introduced the basic idea of Logo procedures. This activity allows students to understand some essential geometric ideas and to see the potential of procedures to build shapes. Throughout the lessons, we realized that the activities of the students were only focused on doing precisely what we said and none of them had the curiosity to explore the Logo programming language a little more. The questions were very poor and we detected that the students memorized the Logo commands and procedures instead of understanding the Logo language.

The next step we made was to introduce the “repeat” command. We spent some lessons showing examples of the “repeat” command and asked the students to use it to create shapes. But after the explanation we realized that the students did not understand the functionality of the repeat command. We decided to change the style of the lessons and created more practical activities (Watt, 1988). Throughout the lessons, with a lot of exercises, the students learned the “repeat” concept and we observed improvements in their performance.

The next activity was to explore the creation of a house using procedures, repeat commands and other commands learned in the course. The major problem felt by the students were associated with mathematical concepts. While drawing the house roof, the students had a lot of problems to place the roof in the correct position in accordance with the remaining portion of the house. To help the students with this problem we explained the Pythagoras' theorem. The goal of this activity was to observe students' skills in building a house with the concepts learned in class. But we realized the students only use simple commands and never the “repeat” command.

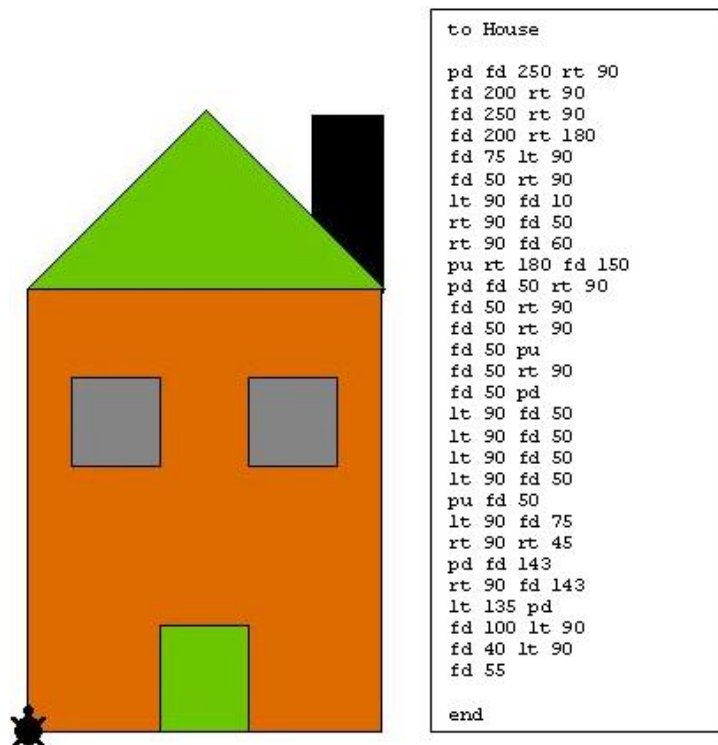


Figure 3 – House drawn by the students with Logo

We asked to the students why they had not used the “repeat” command and the answer was “The Teacher did not say that we needed to use the repeat command in this activity”.

The last activity was to analyse syntactically and logically buggy code and write the modified code in MicroWorlds.

Constructionist environment

In our society, schools focus more in traditional pedagogy instead of using different strategies of learning. We decided to develop a course centered in a pedagogy typified by experimental discovery learning through exploration (Papert, 1993; Papert, 1996). The only material given to the students in the classroom was the computers with MicroWorlds. In the beginning of the course the only help we gave was the explanation of the help options inside MicroWorlds. The students were forced to explore the programming concepts on their own. To motivate the students we showed some works made by other pupils and we explained the potential of Logo programming. We realized then that the students manipulated and learned various Logo vocabularies faster than the other class. We were surprised to see the students using advanced vocabulary such as: *setshape*, *touching?*, *forever*, *waituntil*, *repeat*, and so on.

The students learned the Logo turtle motion and the MicroWorlds interface in a fast way. We challenged them to create shapes with the minimum of programming vocabulary. Almost all of them used the “*repeat*” command to draw the shapes. The next activity given was the construction of the house. Some students did not make it because they said that it was very easy and they preferred to create Logo animations. While some of the students made drawings of a house using simple and advanced Logo vocabulary, others created animations with horses, motor-cycles and plants (see Figures 4 and 5).



Figure 4 – Drawing made by the students with horse, motor-cycle and plants animations



Figure 5 – The move horse procedure

During a lesson a student asked if it was possible to create a game in Logo. We said of course it is possible. The entire class exchanged a lot of ideas and all the groups said: “We want to create

a labyrinth game with a turtle inside". The idea of the students was to create 3 levels: an easy, a medium, and a hard labyrinth. But inside the labyrinth some rules were created. The turtle can never touch the wall. If that happens the turtle moves to the beginning of the labyrinth. When a turtle finds the end of the labyrinth it automatically changes to another level. To move the turtle inside the labyrinth it is required to use the buttons. When a turtle wins the last levels, a picture appears.

Throughout the course the students worked a lot on this project and all the groups showed a strong motivation. Figure 6, shows two labyrinths created by the students.

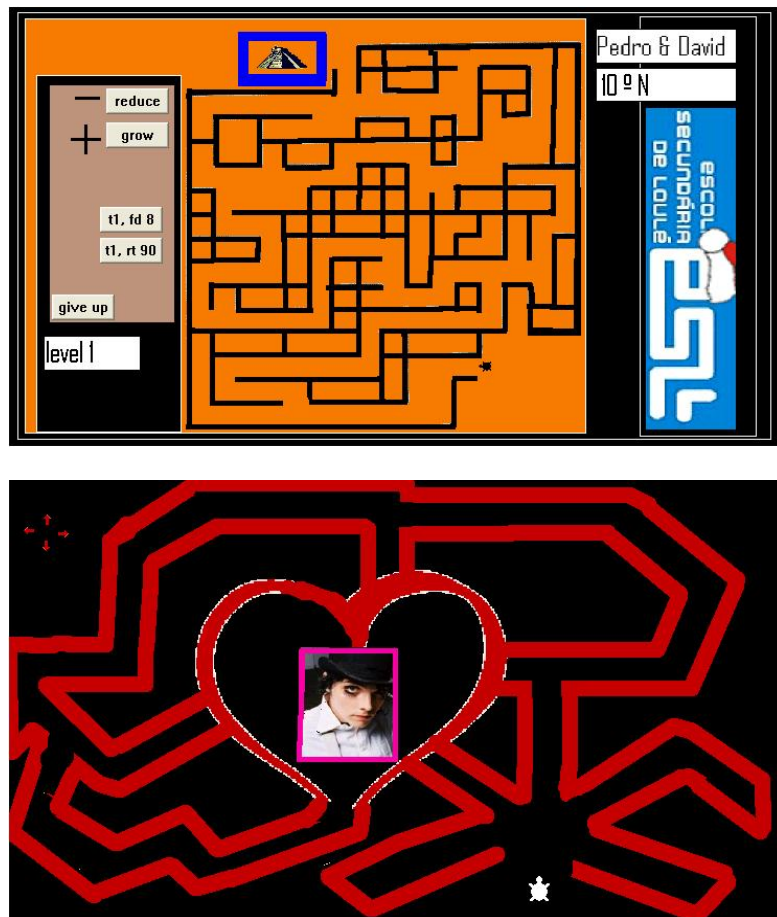


Figure 6 – Labyrinth game made by the students

The students created procedures to build the labyrinth. The most important procedures made were the walls and the transition to another level. Some groups made different procedures (see Figure 7). A group used the "touching?" command to change the level and other groups used the "colorunder" command. Both groups use the "colorunder" command to disallow the turtle to cross the walls.

<pre>To Walls Turtle1, forever [if colorunder = 15 [setpos[-23 -104]]] end</pre>	<pre>To ChangeLevels t1, forever [if touching? "t1 "End1 [setpos[-340 84] level2]] end</pre>	<pre>To Levels turt1, forever [if colorunder = 105 [setpos[-340 184] level2]] end</pre>
--	---	--

Figure 7 – Procedures made by students in the labyrinth game

The goal of the students was to build a hard labyrinth to exchange the game between them and see who would be capable of arriving at the end of the game. After the creation of the labyrinth the students created an HTML template to play the game on the web. The whole work was made by themselves with some suggestions of the teachers.

Student feedback

At the beginning of the course, the students of the N class did not understand the methodology used in the classroom. They asked a lot of questions such as: "But how are we going to learn the Logo programming language alone by ourselves?", "This is like a big project!". All the questions had a traditional style.

On the second lesson a positive sign grew in the classroom. The students interacted with each other and constructed a good environment in the class. Many students discussed the motion of the turtle and others talked about the MicroWorlds interface. The motivation of the students grew as the lessons went by, and we can even say that they also became instructors since the environment in the classroom encouraged them to help their colleagues. The students' feedback to the constructionist environment was very positive and they understood the concepts of learning by doing. We observed a change of mentalities inside the classroom.

The behavior of the M class contrasted sharply with the one observed in the N class. The students made only a few questions and the interaction in the classroom was very limited. Their activities were only focused on doing exactly what we said and they showed little signs of motivation and enthusiasm.

Student evaluation

For both classes, we finished the course with a final exam. Truth be told, we were reluctant to give a final exam to the N class. After all, the idea of a final exam with a set of questions with known answers is something that goes against the spirit of constructionism. Nevertheless, we had to give the exam because it was a formal requirement of the school. A final grade had to be given to each student and, according to the school regulations, it could not be given based on a subjective evaluation.

The exam had three questions limited to 35 minutes and was given to both classes at the same time. The exam dealt with the creation of some shapes using the Logo vocabulary learned in the course, as well as with building simple animations.

During the exam, the students of the M class showed some difficulties in programming the shapes and were worried about the time. The exam produced a stress reaction in them. Their ability to program in Logo was more theoretical and less focused in the discovery process. The students were not stimulated to learn alone and explore the Logo programming vocabulary. In this class we perceived a bad reaction during the exam. When a student did not obtain a positive result the first reaction was to quit. The N class showed a more relaxed behaviour. The students were more calm to accomplish the exam and did not complained.

The final grades describe the perspective we expected. The N class had better grades. The average grade for the N class was 12,4 and for the M class was only 10,1. Note that in Portugal, a 20-point grading scale is used, in which 20 is the highest grade and 0 is the lowest.

We would like to emphasize that, in our opinion, there was no need of doing a final exam in order to conclude that the students of the N class had a better learning experience (and performance). By looking at the activities conducted on both classes, it became evident that the N class was learning much more, was having more fun, and were even doing things that we thought they were not capable of doing. The results of the exam just confirmed what we already felt. A possible explanation for the better performance of the N class during the exam might be due to the enthusiasm towards the subject matter. By the end of the course, the N class really enjoyed programming in Logo while the M class showed no particular interest in it.

Summary and Conclusions

This paper presented two different pedagogical paradigms used in an introductory computer science classroom at a high school in Portugal. The study supports the notion that constructionist activities can aid learning better than a traditional environment.

There is a great interest in the exploration of constructionism in learning environments. In the near future, we intend to create more experiments in high school with students with special educational programs. We also intend to disseminate the Logo programming for other teachers to use in their classrooms.

It is important to try to adopt the constructionist approach in order to open new mentalities in schools. It is important that the educational system progresses for a better future with new generations and new philosophies of education. With this research we believe that the role of the teacher in the classroom needs to change, and the possibility to learn by exploration must be given to students. Any person possesses the ability to learn and discover. The teacher is essential in the learning process to provide motivation and to aid the students in the exploration of new skills.

Acknowledgments

We sincerely thank the contributions of the students who made this study possible. This work was sponsored by the Portuguese Foundation for Science and Technology (FCT/MCES) under grant POCI/CED/62497/2004 and by Foundation Caloust Gulbenkian under grant Proc. 65538.

References

- Fosnot, C.T. (Ed.). (1996). *Constructivism: Theory, perspectives, and practice*. New York, NY: Teachers College Press.
- Gorp, M. J. and Grissom S. (2001) *An Empirical Evaluation of Using Constructive Classroom Activities to Teach Introductory Programming*. Computer Science Education, Vol. 11, No. 3, pages 247-260.
- Papert S. (1993), *Mindstorms: Children, computers, and powerful ideas* (2nd ed.), New York: Basic Books.
- Papert S. (1996), *The connected family: Bridging the digital generation gap*, Longstreet Press, Inc.
- Kafai, Y. and Resnick, M. (eds) (1996): *Constructionism in Practice: Designing, Thinking, and Learning in a Digital World*. Mahwah, New Jersey, Lawrence Erlbaum.
- Skinner, B.F. (1953). *Science and human behavior*. New York, NY: Free Press.
- Watt, D. H. (1988) *The Computer as Microworld and Microworld Maker*. In SIGCUE Outlook, ACM press, New York, USA, Vol. 20, No. 1, pages 81-89.
- Wulf, T. (2005) *Constructivist approaches for Teaching Computer Programming*. In SIGITE '05: Proceedings of the 6th conference on Information Technology Education, ACM press, October, New Jersey, USA, pages 245-248.

A Logo-based didactic sequence for the learning of numerical systems by high-school students

Verónica Contreras-Valencia, *vero_c_v@yahoo.com.mx*

Dept. of Mathematics Education, Center for Research & Advanced Studies (Cinvestav), Mexico

Ana Isabel Sacristán, *asacrist@cinvestav.mx*

Dept. of Mathematics Education, Center for Research & Advanced Studies (Cinvestav), Mexico

Abstract

In this paper we present a didactic sequence that is part of a research that looked into ways of helping high-school students understand the structure of numerical systems and the methods of conversion between systems, specifically positional number systems using a base. We have observed that high-school students have difficulties in understanding how different numerical systems are formed and how to convert numbers between systems. Specifically, students tend to have difficulties in a) the understanding the concept of base, of b) how a base can be structured using a set of symbols and c) how to read and construct a number in a base other than the decimal one.

However, because of the use of binary and hexadecimal numerical systems in computer science, we consider it important for students of this discipline to be able to understand numerical systems and the general methods for converting numbers between systems.

Thus, in an attempt to minimize the difficulties presented above, we designed a computer-mediated didactic sequence (using the Logo programming language) to help students understand how numbers in a base-system are formed, and how to convert between bases. The sequence was designed following Papert's constructionist paradigm, so that many of the tasks of the sequence involved writing or modifying Logo programs.

Specifically, the didactic sequence combined paper-and-pencil activities, with Logo-programming ones, for converting numbers between different base-systems. Ultimately, students had to write two general conversion computer programs: the conversion of numbers given in base 10, to any other base; and the conversion of numbers given in any base, to base 10.

In a study with computer science high-schools students in Mexico, we observed that the didactic sequence was indeed very helpful for helping students understand the structure of numerical base-systems, the conversion methods between bases and in how to read numbers in bases other than the decimal one.

Keywords

Numerical systems; conversion algorithms; computer-based didactic sequence; Logo; high-school students

Introduction

In this paper we present part of a research that looked into ways of helping high-school students understand the structure of numerical systems and the methods of conversion between systems, specifically positional number systems using a base. Our interest in this study derived from the difficulties, which we have observed, that high-school students have in understanding how different numerical systems are formed and how to convert numbers between systems. Because of the use of binary and hexadecimal numerical systems in computer science, we consider it important for students of this discipline to be able to understand numerical systems and the general methods for converting numbers between systems.

Many types of numerical systems have been developed throughout history, most of them using additive and/or positional techniques. Most systems also use a *base*; that is, a set (with a fixed number of elements) of symbols necessary for representing a quantity. The decimal and binary systems we use today, are positional numerical base-systems. Castillo et al. (2002, p. 21) explain the following about these base systems:

In a numerical system of base b , there should exist exactly b different symbols for representing all of its numbers. If $b < 10$ (is less than) 10, the symbols can be simply the digits 0, 1, 2, ..., b ; but if $b \geq 10$ (is bigger than or equal to) 10, then it is necessary to introduce new symbols for representing the numbers that in the decimal system are represented by 10, 11, 12, etc.¹

In the current national secondary school mathematics curriculum of Mexico, the topic of numerical systems consists of a presentation of ancient number systems, such as the Babylonian, the Roman and the Mayan, and the positional number systems in bases 10 and 2. However, in the last years of high-school, this topic is only covered in computer science workshops, where students are given equivalence tables between bases 10, 2 and 16, and shown a few examples and tasks of conversions between these three systems: decimal, binary and hexadecimal. And even though all base b positional numerical systems are ruled by the same principles, rarely are other systems covered nor the general conversion methods analysed, either in class or in the textbooks used (e. g. Tocci, 1993). Furthermore, existing tools, such as calculators, that allow students to convert numbers between systems do not make explicit the conversion algorithms and can have other limitations (e. g. limited numerical systems).

We consider that students of this level can work with other numerical systems of this type (base b positional) and be able to understand the general conversion methods. But being aware of the difficulties that this topic represents, we wanted to research ways that would promote learning. First, from the cognitive perspective, we discarded the traditional “instructionist” teaching approaches, in favour of a constructivist pedagogical approach, that would promote the construction of the knowledge related to the topic we were concerned with. In particular we chose to follow the “constructionist” paradigm (Papert, 1993): that is, we wanted students to engage in activities where students would have to construct means to convert numbers between different numerical systems.

For this, we also considered it important, to include a computer-mediated learning approach. We considered that computers have a great potential for education because they can provide students with expressive (as well as exploratory) power. Thus, our approach was to have students construct programs for converting numbers between the decimal (base 10) numerical system and systems in any other base, and vice versa; as well as having them define the symbols and structure of numerical systems of other bases. The idea was that by programming the computer –i.e. teaching the computer to think— students can engage in an exploration of how they themselves think (Papert, 1980).

¹ Translated from the original in Spanish.

We chose Logo as the programming language for our project for several reasons:

- First, Logo is an ideal tool for learning (Papert 1980); it facilitates investigations, explorations and constructions.
- From a computational perspective: it has very simple control structures; and
- it has the possibility to define new commands, by creating new procedures (Segarra & Gayan, 1985).
- Modularity facilitates the construction of complex programs through simple building “bricks”.
- It has powerful recursion characteristics that allow recursive programs to be expressed simply in very few lines (this was important, because the conversion algorithms between base-number systems are iterative and thus require the use of recursion).
- It is easy to operate with lists (list-processing) (Segarra & Gayan, 1985).
- Finally, Logo can facilitate the construction of a general algorithm from the construction of specific cases.

Thus, we designed a sequence of paper-and-pencil and Logo-based activities that centred on the construction (programming) of algorithms for converting numbers from one numerical system to another.

A didactic sequence for the learning of the conversion of numbers between different bases

We wanted students to construct and explore different numerical systems in a base b , and to be able to convert numbers between bases. Ultimately, students had to write two general conversion computer programs: the conversion of numbers in base 10, to any other base; and the conversion of numbers in any base, to base 10.

Methodology

As explained above, for the didactic sequence, we combined paper-and-pencil activities, with Logo-programming ones, for converting numbers between different base-systems.

For the computer-based activities, we used MSWLogo as the version of the Logo programming language.

The paper-and-pencil activities served to create a basis that could be built upon, through the constructive programming activities. The activities were structured through a series of worksheets. Each activity had several worksheets: some of them containing examples, and others, specific exercises, reflection questions, and programming tasks.

In each activity, the first example or task was presented and explained by the teacher; for all other tasks, students solved the problems on their own.

This sequence was tried out with senior high-school students in Mexico (17-18 yr-olds) taking a course in computer science; and we carried out a study to assess the learning that can take place with the sequence we designed. That study (Contreras-Valencia, 2006) took place in two phases: a pilot phase with 9 students, and a main phase with 10 students. In order to assess students' learning, in addition to field observations, those students were given a diagnostic questionnaire prior and after the study; they were also interviewed. The results are reported in Contreras-Valencia (2006).

During the didactic sequence, students worked in pairs with one computer per pair of students. According to Sacristán (2000), working in pairs can promote collaborative work, discussion and enrich students' initiative for exploration and construction.

Students worked for a total of 6 hours in two sessions on the didactic sequence. Prior to this, the students had had 10 hours of general Logo programming experience, which included recursive programming and list-processing.

The structure of the didactic sequence

The didactic sequence included the following activities:

- 1.- Conversion of numbers from any base x to base 10, using paper and pencil
- 2.- Conversion of numbers from bases 2, 8, 4 and 5 to base 10, using Logo
- 3.- Conversion of numbers from a system in any base x to base 10, using Logo
- 4.- Conversion of numbers from base 10 to bases 2, 8, 4 and 5, using paper and pencil
- 5.- Conversion of numbers from base 10 to bases 2, 8, and 4, using Logo
- 6.- Conversion of numbers from base 10 to any base x , using Logo.

Below, we present an overview of the activities from the study.

Activity 1: Paper-and-pencil conversion of numbers from any base x to base 10

The purpose of this activity was to familiarize students with the method for converting numbers from any base x to base 10.

This activity began by giving students worksheets with five examples of conversion of numbers from bases 2, 8, 4, 5 and 16 to base 10 (see Figure 1), that were also explained and presented by the teacher. Students then had to convert numbers from bases 2, 8, 4, and 5 to base 10, using paper and pencil (see Figure 2).

To convert a number $(110)_2$ given in **base two**, to base ten:

- 1) Take apart the digits from the number given in base x

1	1	0
---	---	---
- 2) Number (give a positioning value) to each of the digits, from left to right

1	1	0
2	1	0
- 3) Raise the base x (in this case, $x = 2$) in which the number is given, to the power given by each positioning value

1	1	0
2^2	2^1	2^0
- 4) Multiply each of the digits that were taken apart in step 1, by each of the powers from step 3, according to their respective position; then add them.

$$\boxed{1} * \boxed{2^2} + \boxed{1} * \boxed{2^1} + \boxed{0} * \boxed{2^0} = \left(\boxed{6} \right)_{10}$$

Figure 1. Worksheet example of the conversion of a number from base 2 to base 10

At the end of the activity, students were given an algorithm, see below, for converting a number from base x to base 10. Although the teacher verbally explained this algorithm, students were asked to try to analyse and understand it by relating it to the method given in the previous examples and tasks. Additionally, the algorithm was clarified by relating each of its steps as explained in natural common language, to those in a computer-programming structure, and to possible corresponding partial Logo instructions (in a similar way to that presented in Figure 5).

The conversion algorithm of a number from base x to base 10

1. Read number
2. `total_conversion = 0`
3. `position = 1`

4. $nb_items = size(number)$
5. $item = number[position]$
6. if $position > nb_items$ then
 - 6.1.1. print conversion
 - 6.1.2. go to step 7
- if not:
 - 6.2.1. $total_conversion = total_conversion + item * 2^{nb_items - position}$
 - 6.2.2. $position = position + 1$
 - 6.2.3. go to step 5
7. end

The purpose of presenting students with this algorithm was to give them a means to relate the method given in this Activity 1, to an algorithm in a computer program as explored in the following activity.

Convert the number $(310)_5$ given in **base 5**, to base 10.
Don't worry if you have left-over blank spaces.

1)

--	--	--	--	--

2)

--	--	--	--	--

3)

--	--	--	--	--

4)

	*		+		*		+		*		+		*		+		*		=	<div style="border: 1px solid black; width: 20px; height: 20px;"></div>)	10
--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	---	---	----

Figure 2. Example from the paper-and-pencil Activity 1, of a conversion task of a number given in base 5 to base 10

Activity 2: Exploring and constructing Logo programs for converting numbers from specific bases (2, 8, 4 and 5) to base 10

This activity began by giving students a Logo program, for converting a number from base 2 to base 10, which would serve as a starting point for writing a more general conversion program. Students had to analyse this program by playing with it and going through the tasks structured in the worksheets. The initial program that students were given was as follows:

Logo program for converting numbers given in base 2 (a binary system) to base 10

```
to convert_bin_ten :number
  process_ten :number (count :number) 1 0
end

to process_ten :number :nb_digits :pos :total
  if :pos > (count :number) [print :total stop]
  process_ten :number (:nb_digits - 1) (:pos + 1) (:total +
    ((item :pos :number) * (power 2 :nb_digits - 1)))
end
```

Examples of the results produced by the *convert_bin_ten* procedure are given below:

```

convert_bin_ten 110101
53
convert_bin_ten 1111000010
962
convert_bin_ten 111111100
508

```

Students were then asked to modify this program, to create programs for converting numbers from specific bases (from bases 8, 4, 5 and any two others that the students chose) that they tried out and verified through tasks given in the activity's worksheets.

Activity 3: Generalising the Logo program for converting numbers from any base x to base 10

In the next activity (Activity 3), the students had to generalise the Logo programs create in Activity 2, to write a general one for converting any number from a system in any base x to base 10. Students were given the *convert_x_ten* procedure (given below) and the title line for the sub-procedure *process_ten*, and asked to complete the latter, in order to complete the program.

After a period of exploration, all of the student pairs were able to produce a correct procedure for converting numbers in bases lower than 10, to base 10. In the worksheets, students were asked to observe what happened when they tried their procedures to convert a couple of hexadecimal numbers (12C and 13) to base 10, and to think how they would modify them to accept as inputs numbers in bases larger than 10. The additional difficulty here, in comparison to the specific cases that the students had already programmed, was that numbers in bases higher than ten, are written using symbols other than digits; therefore, students were given the *transform* and *transform1* procedures below, for transforming a string of characters. After trying these procedures as suggested in the worksheets, students were asked to use them to modify the *convert_x_ten* program (by modifying its sub-procedure *process_ten*), so that *convert_x_ten* could be used to accept strings of characters of numbers in bases larger than 10 (up to base 36).

The final program was one such as the following:

General Logo program for converting numbers from any base (up to 36), to base 10

```

to convert_x_ten :number :base
  process_ten :number (count :number) 1 0 :base
end

to process_ten :number :nb_digits :pos :total :base
  if :pos > (count :number) [print :total stop]
  process_ten :number :nb_digits -1 :pos + 1 :total + ((transform
    :number :pos) * (power :base :nb_digits - 1)) :base
end

to transform :string :position
  output transform1 (item :position :string)
end

to transform1 :symbol
  ifelse (numberp :symbol) [output :symbol]
    [output (ascii :symbol) - 55]
end

```

Examples of the results produced by the *convert_x_ten* procedure are given below:

```

convert_x_ten 110101 2

```



```

53
convert_x_ten  AA2  16
2722
convert_x_ten  745  8
1351
  
```

Activity 4: Paper-and-pencil conversion of base 10 numbers to a base x

The following activity was paper-and-pencil based and had as aim to familiarise students with the method for converting base 10 numbers to any other base x. As in previous activities, students were first presented with examples (see Figure 3) illustrating the method for converting base 10 numbers to other bases. The examples involved converting numbers from base 10 to bases 2, 8, 4 and 5.

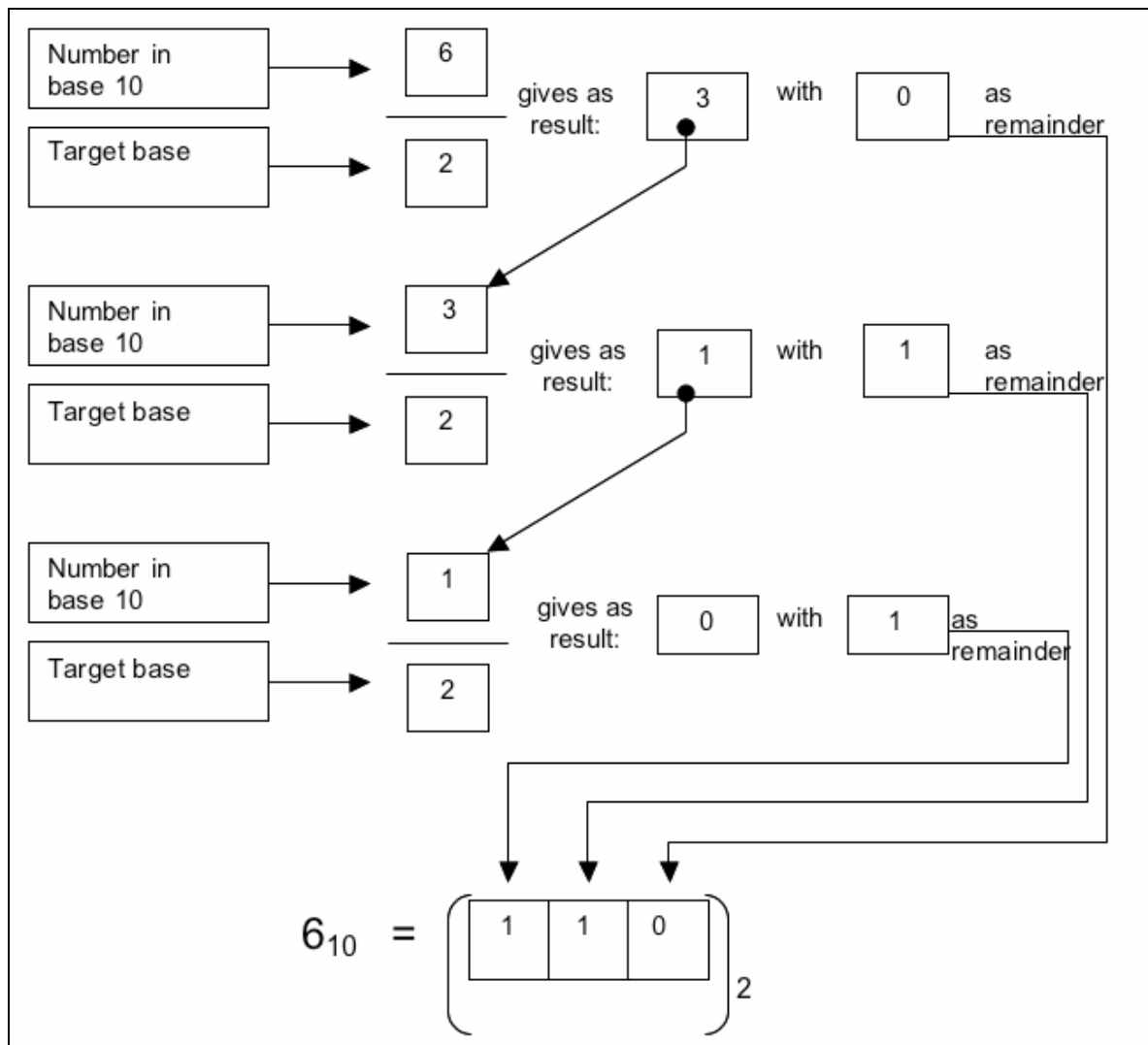


Figure 3. Worksheet's example of the conversion method from base 10 to base 2

After the presentation of the examples, in the second part of the activity, students had to use the method they had been presented with, to convert specific numbers from base 10 to other bases –2, 4, 8 and 5— (see Figure 4).

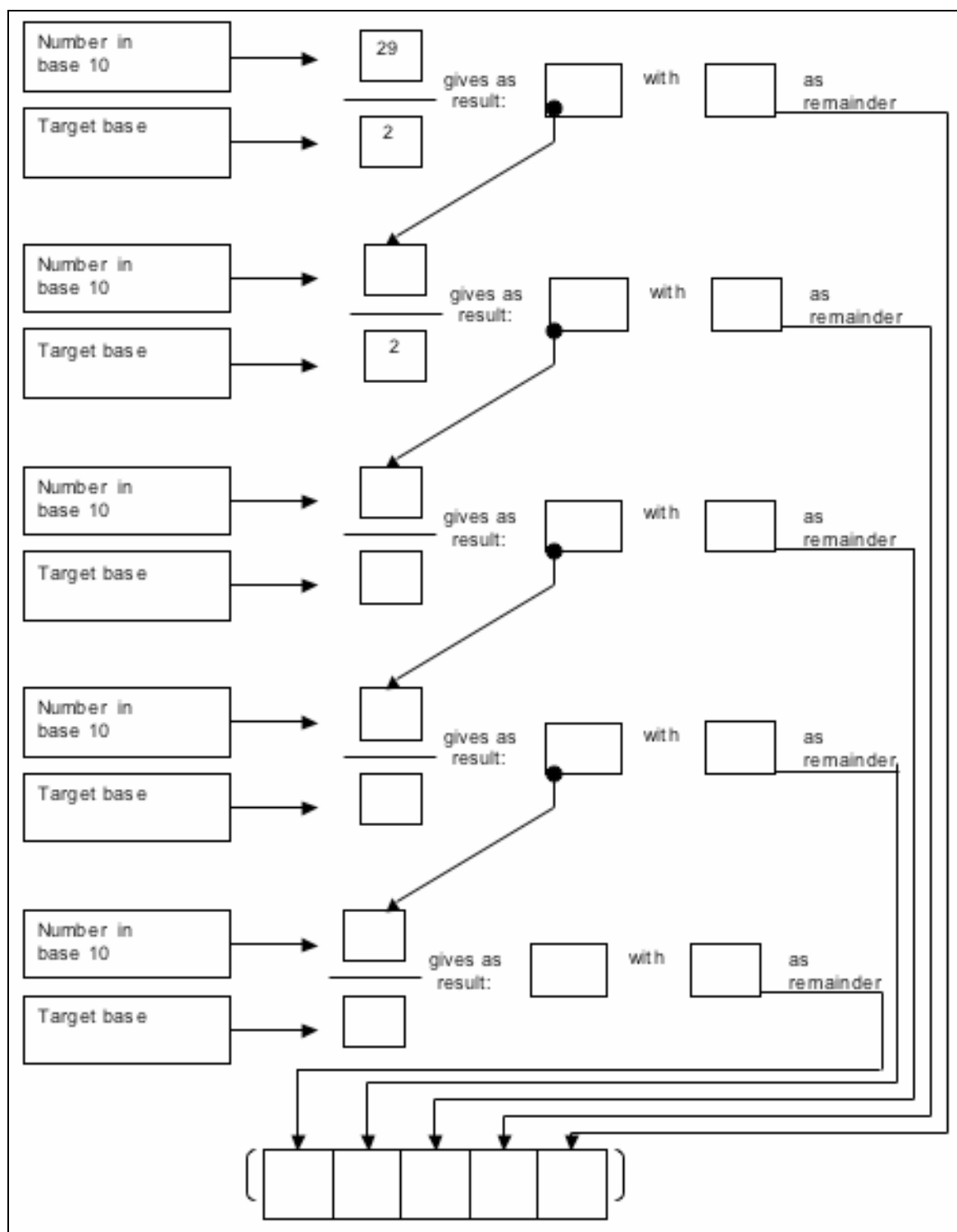


Figure 4. Example from the paper-and-pencil Activity 4, of a conversion task of a number given in base 10 to base 2

At the end of the activity, students were asked to try to create an outline for an algorithm for converting base 10 numbers to any base x . The activity concluded by analysing the algorithm for converting a number in base 10 to another base, using the method explored in the previous activity. The algorithm was clarified by relating each of its steps explained in natural common language, to those in a computer-programming structure, and to possible corresponding partial Logo instructions, as shown in the table given in Figure 5.

Algorithm in natural language	Algorithm as a computational structure	Partial Logo instructions
	The target (converted) number is defined as a variable that is initially an empty set (list): result = []	make "result []
Take the given number in base 10	A number in base 10, is assigned to a variable: number	:number
and apply the following algorithm: Divide the number by the target base		process :number
which gives an integer part	integer (number / base)	int :number/:base
and a remainder	remainder (number / base)	remainder :number :base
The remainder is part of the representation of the number in the target base, positioning each new remainder to the representation from right to left.	result is now defined as the concatenation to the left of the remainder of the previous result: result := [remainder (number/base) result]	make "result [fput remainder :number :base :result]
If the integer part is zero, the process ends, and the number is represented by the concatenation of the remainders	If integer (number/base) = 0 , stop (the process is ended)	if :number = 0 [stop]
	and output the value of result up to that point	output :result
Otherwise, the previous steps are applied on the integer part, in an iterative way (until the integer part is zero).	number is assigned the value of the integer part resulting from the division of the original number and the target base number := integer (number / base) And the previous steps are repeated for this new number	process (int :number/:base)

Figure 5. Explanatory table for the algorithm to convert a given number in base 10 to any other base

Activity 5: Exploring and constructing Logo programs for converting numbers from base 10 to specific bases (2, 8, and 4)

In the next activity, students were first given a Logo program (with main procedure *convert_to_bin*, shown below) for converting a number from base 10 to base 2, which would serve as a starting point for writing a more general conversion program.

Logo program for converting numbers given in base 10 to base 2 (to a binary system)

```

to convert_to_bin :number
  make "string [ ]
  processbin :number
  output :string
end

to processbin :number
  if :number = 0 [stop]
  make "string (fput remainder :number 2 :string)
  processbin (int :number / 2)
end

```

Examples of the results produced by the *convert_to_bin* procedure are given below:

```

convert_to_bin 53
110101
convert_to_bin 962
1111000010
convert_to_bin 508
111111100

```

As in previous activities, through a series of worksheets and tasks, students had to modify this program to create programs for converting numbers from base 10 to specific bases (bases 8 and 4, and any others that the students chose).

Activity 6: Generalising the Logo program for converting base 10 numbers to other bases

In the final activity, the task was to complete a program for converting numbers given in base 10, to any other base-system. Students were first asked to try to modify the programs from Activity 5, to make a general program (they were given the main procedure *convert_ten_x* –shown below— and asked to complete the subprocedure *process* for which only the title line was given). Almost all of the students were able to successfully write a program to convert base 10 numbers to bases lower than 10.

They were then asked to reflect on how to modify the program to be able to convert numbers to bases larger than 10. For this they were given a useful procedure *convascii* (see below) for converting a numerical value to a string of ascii characters; using this additional sub-procedure, they were asked to modify the program to convert base 10 numbers to any base (up to base 36). The finished program was something like this:

Logo program for converting numbers given in base 10 to any base (up to 36)

```
to convert_ten_x :number :base
  make "string []
  process :number :base
  output :string
end

to process :number :base
  if :number = 0 [stop]
  make "string (fput convascii remainder :number :base :string)
  process (int :number / :base) :base
end

to convascii :num
  if :number < 10 [output :number]
  output char (:number + 55)
end
```

Examples of the results produced by the *convert_ten_x* procedure are given below:

```
convert_ten_x 53 2
110101
convert_ten_x 2722 16
AA2
convert_ten_x 1351 8
2507
```

Concluding remarks

Initially, the students we worked with, were only familiar with the decimal and binary systems, and they had difficulties in a) the understanding of the concept of base, of b) how a base can be structured using a set of symbols and c) how to read and construct a number in a base other than the decimal one; but as the didactic sequence progressed, we observed great improvements in all three areas of difficulty. By the end of the didactic sequence, when students had written general programs for converting numbers from a base *b* to base 10, and viceversa, they had developed a more general view of numerical base-systems and how these are structured; specifically they showed a much better understanding of the relationship between a

base and the symbols that can form that base. They also understood how to apply the general conversion methods between bases. (For further details, see Contreras-Valencia, 2006.)

The role of Logo was important in this, because, thanks to its accessibility, its recursive power and the ease of modular programming, the programs constructed could be very short, which facilitated their analysis and the understanding of the algorithm described in those programs.

In a future research, we want to explore how to develop a better understanding of operations of numbers in bases other than the decimal one.

References

- Castillo, E., Forero, J. and Rodríguez, J. C. (2002) *Matemáticas e Informática: Programa Educativo Visual. Enciclopedia Autodidáctica Millennium*. Editorial Norma, Mexico.
- Contreras Valencia, V. (2006) *Una secuencia de actividades con computadora para el aprendizaje de los sistemas de numeración en bachillerato*. M. Sc. thesis, Cinvestav-IPN, Mexico.
- Papert, S. (1980) *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, New York.
- Papert, S. (1993) *The Children's Machine*, Basic Books, New York.
- Sacristán, A. I. (2000) *Investigación del aprendizaje matemático mediante micromundos computacionales*. In Vinculación y retos ante la dinámica del entorno: Primer encuentro interdisciplinario de investigación. Universidad Iberoamericana Laguna, Torreón, Coahuila, Mexico. Pp. 11-18.
- Segarra, M. y Gayan, J. (1985) *Logo para maestros. El ordenador en la escuela: propuesta de uso*. Gustavo Gili, Barcelona.
- Tocci, R. J. (1993) *Sistemas digitales: principios y aplicaciones*. [Digital systems: principles and applications]. Prentice Hall, Mexico.

Natural Visual Programming Languages

Tiago Correia, *tiago@cnotinfor.pt*

Department of Computer Science, University of Coimbra, Coimbra

Secundino Correia, *secundino@cnotinfor.pt*

Cnotinfor, Coimbra

Abstract

Children born in an age of knowledge, have access to much more information than we thought. Even without the capacity of reading or writing they understand information and in most cases create new information, build new knowledge. We face a challenge on how to create environments and tools that allow them to develop skills and create new knowledge, specially the ability to program in a natural way, instead of being stuck in the complexity of the environment.

Integrated Learning Environment (ILE) have been studied and implemented along the last decades. They have been defined (Correia et al.) as an ecosystem whose components are: learners, learning organizers, a system or environment where learners interact, technological structured resources including learning objects, methods and strategies for teaching and learning and a set of values to be clarified on action.

We can consider a Natural Visual Programming Language (NVPL) as an ILE providing at least the following five issues: Stimulus provoking the user to take some meaningful actions; Coaching on what makes sense (or not) on the context of programming; Feedback about all steps; Formalization sets; Generalization.

To achieve these goals concepts like the Flow theory (Csikszentmihalyi), Mini-languages (Brusilovsky et al.) and Edutainment (Resnick), should be taken into account. NVLP should allow the users to build computer programs in a natural fluid way and should lead, those who are really interested, to learn the principles, structures and algorithms of programming.

Keywords

Programming; games; visual programming; learning context; integrated learning environment

Introduction

For designing and developing successful products “companies need a development philosophy that targets the human user, not the technology” (Norman, 1999).

In our days technology is everywhere. Children since they born, deal with technology. There are so many devices in our daily life with technology embedded, that we don't even realise how important they are. TV, mobile phones, microwaves, stoves, clocks, consoles, computers and cars are just some examples of artefacts that we use without even thinking about all the technology that it is behind them. For a child is completely normal to have a mobile phone to talk with friends, or even use the Internet to do it, either by using instant messages or VoIP.

Children born in this new age of knowledge, have access to much more information than we thought. Even without the capacity of reading or writing they understand information and in most cases create new information, build new knowledge.

We assist to a general and deeper trend of computer based technologies imbibed on every day objects by disguising and getting almost invisible. This drift implies new artefacts with imbibed technology and a new functional design much more centred on human being than technology. On this sense we may be talking about disappearing technology, because technologies go behind like electricity that we only notice when it fails. This implies the emergence of new artefacts centred on individual's daily needs, artefacts with almost hidden computational and information processing power, natural leading to the disappearing of computers like we know them nowadays (Norman, 1999).

What steps do we need to take in order to design and develop such type of environments to allow children to create new knowledge, specially the ability to program in a natural way, so they are not stuck by the complexity of the environment?

Before trying to answer the previous question, let's first clarify some related overlapping concepts like Constructionism, Integrated Learning Environments and Learning Contexts.

Constructionism is a theory of learning and education based on two different senses of "construction." First, constructionism is grounded in the idea that people learn by actively constructing new knowledge rather than by having information "poured" into their heads. Secondly, constructionism asserts that people learn with particular effectiveness when they are engaged in constructing personally meaningful artefacts (such as computer programs, animations, or robots). Social constructivism adds to this the important role of pairs on this process, by sharing, interacting and joining efforts for building a solution. Metacognition is also enforced by the need of explain our vision to the others.

A close concept linked to constructionism is the concept of microworld forged by Marvin Minsky and Seymour Papert. For them Microworld implies an environment of discovery, where the learners can operate, manipulate and create objects by testing the effects they produce each other. Usually these kinds of environments inlay some form of mathematical processing, allowing users to observe how there decisions affect the organization and balance of the environment.

A microworld was initially more like a learning context or a framework where learners could make theirs constructs, by opposition to digital contents or learning objects. During the nineties and first decade of twenty one century a big emphasis was put on creating and indexing digital contents and learning objects by opposition to microworlds and mind tools for learners to rebuild there on knowledge by doing and interacting.

Nowadays it begins to be consensual the need of balance between learning contexts and learning contents or learning objects. Neither it is viable a pure learning context without contents and coaching, nor it's effective a pure contents repository. In fact, contents need a context and strong behind models to become valuable and flexible and contexts need content to get meaningful.

In our view an idea that seems to integrate all these concepts is the idea of Integrated Learning Environment (ILE). We can define an ILE as an ecosystem whose components are: learners, learning organizers, a system or environments where learners interact, technological structured resources including learning objects, methods and strategies for teaching and learning and a set of values to be clarified on action.

ICT easily fall on the trap of existing educational structures by a process of domestication where the learning construction process glided to a contents transmission process, although very well packed in wonderful multimedia packages or web based contents repositories. Biodiversity, contexts, values clarification and the empowerment of learners (both students and teachers) subtly glide to unilateral control and one-dimensional views. Networks, learning communities, communities of practice, collaborative contexts easily glide to supervised webs that confine creativity, change and critical thinking.

The challenge of constructing ILE with a strong technological soundness mainly consists on creating learning contexts. By them, learners use materials, concepts and imbibed methodologies to build, destroy and rebuild knowledge synapses in a collaborative, creative and critical way, promoting reflection about the cognitive, emotional and setting values processes in the relation between learners/ groups/ contexts/ contents and containers. In this vision teachers are essentially learning contexts and contents organizers. They should be leaders capable to create values based shared visions. Teachers should be strategists capable to identify opportunities and building bridges, energizing individuals and groups, focusing them on the design of the future. In fact the best way to prepare and predict the future, as Alan Kay says, is to invent it. This is the central role of schools: to invent the future.

We can see a Natural Visual Programming Language a special case of Integrated Learning Environment providing at least the following issues:

- Stimulus provoking the user to take some meaningful actions; these stimulus could be visual and audible. The possibility of kinetic stimulus should also be considered.
- Guidance on what makes sense and makes not sense on a particular context – programming; This should be like a coach or a tutor that gives contextualized guidance on what to do next.
- Feedback about all steps so the learner knows what makes and makes not sense when building a particular computer program (construct).
- Formalization sets, this means the possibility to have an overview of all the steps and the way they interconnect each other, and why the all construct makes sense on this particular context.
- Generalization, which means the possibility to try a similar solution on a different computer program (construct).

Programming

What it is programming? Wikipedia presents a nice definition of programming “Computer programming (often shortened to programming or coding) is the process of writing, testing, and maintaining the source code of computer programs.”

But computer programs are not only what is normally known as application software, like OpenOffice, FireFox or Skype. Computers programs are combinations of instructions to perform a task by the computer. Using different combinations of the same instructions, might lead to different tasks and different results. These instructions are the building blocks of any computer program. Instructions are usually text. An example in C programming language:

```
while (i < 10)
    printf("%d", i++);
```

Figure 1. C programming language example

The instructions presented might not be easily understood by most of the adults that didn't have any contact with programming languages.

For a long time programming was a skill that not everyone was able to master or even learn. The programming languages were very tied with the hardware and were very difficult to learn. Computers kept evolving and so it did the software, that become more user centred. Programming language started to be more abstract from hardware. Many programming languages appeared and new paradigms of programming also emerged. Even so, programming is still a problem for most computer science education institutions.

Papert (1980) described the design of the LOGO programming language as taking the best ideas in computer science about programming language design and "child engineering". It seems that LOGO provided the jump to bring programming to the ordinary user. But now computers have evolved and software is becoming easy to use. Programming languages have also grown to a new level of possibilities and facility, but still are difficult to learn and master (Kahn, 1996).

There have been many efforts to develop programming languages to support the initial steps in the world of programming (Mendelson et al., 1990). Let us to have an overview.

Mini-languages

Mini-languages are a type of microworlds that are designed to be small and simple programming languages for students to learn programming by usually controlling an actor, either virtual or physical device (Brusilovsky et al., 1997). The objective of mini-languages is to provide a solid background for computer science learning by using a context designed for this purpose. Their syntax and semantics are small and simple, the student can spend less time on learning the language and more time in solving algorithms.

The turtle geometry subset from the LOGO programming language provides a mini-language where the student controls a turtle in a virtual world. Usually this subset is as the starting point in learning programming using LOGO. But programming develops general thinking skills as Papert and others (1980) observed, like problem decomposition, component composition, explicit representation, abstraction, debugging and thinking about thinking. To learn to program is not just a simple skill, but an important one that can boost many others. "It is the acquisition of algorithmic thinking as an explicit, familiar and powerful tool" (Brusilovsky et al., 1997). Many of these skills provide the basis for logical and abstract reasoning that is fundamental to the learning process. By these reasons, may be programming should be explored since the early stages of education.

Mini-languages provide an adequate learning context for programming. They are visually intuitive, simple and a powerful way to introduce students to programming. But mini-languages can be used for many others purposes then programming. Like the LOGO subset could be used to learn geometry. Another example is SOLO programming language (Eisenstadt, 1983) that was used in the context of psychology. It is not very visual appealing for our current standards, but for the time it was developed it was quite evolved. It even had a help/tutoring system to help the students.

For very young children mini-languages can be provided in a different way by using a tangible object. The Robot Roamer is an artefact that uses a subset of LOGO language to control the robot. The interface is a concept keyboard on the shell of the robot with commands to move the robot and create music. The extended version can do more complex tasks as controlling motors, lights and sensors. The context in which the robot is used is very different from learning to

programming, even if the student develops programming skills by using it. The student can change the look of the robot to a context where he feels comfortable.

Robot Roamer is a nice example how mini-languages can be used to learn programming and develop many others skills like creativity, music, electronics, manual dexterity, spatial orientation, geometry, artistic expression, social skills, collaborative work and thinking about thinking.

Games and programming

If we take a look at the Entertainment Software Association information about 2006, 69% of American heads of households play computer games. Games are part of our live and most of the students have already played a game. An extensive research about games with children and adults in anthropology, psychology and education shows that games are important mediators for learning and socialization during life (Ribier, 1996). Computer games have been used for learning for quite a long time, for many different purposes and in many different contexts (Mitchell and Savill-Smith, 2004).

To use them to promote the learning process of programming has also been tried by making the students creating their own games (Leutenegger and Edgington, 2007) using ActionScript programming language as a starting point. One of the reasons to use Flash is that it provides instant visual feedback for the student which helps him to see if the program is correct. Flash development environment is very general and it has not by default a proper context for games development, but by providing snippet code for the student it provides a better learning context for programming games.

There are many environments for programming games, but most of them are not easy to use or are very easy but don't allow building more complex games. They usually lack of the simplicity and smallness of the mini-languages.

But games have many features that make them appealing. Understanding these features and implementing them on software for learning to programming gives birth to a new generation of software. ToonTalk is an example, that such type of software is possible to build (Kahn, 1999).

Flow theory

Mihaly Csikszentmihalyi in an effort to explain happiness introduced the concept of Flow, which has since become fundamental to the field of positive psychology. Flow represents the feeling of complete and energized focus in an activity, with a high level of enjoyment and fulfillment.

During the Flow experience, we lose track of time and worries, which is very common on players. They are completely focused in the performance and pleasure of the game. But it can also be associated with professional athletes and artists, or every human being (Chen, 2007).

Csikszentmihályi research identified eight major components of Flow:

- A challenging activity requiring skill;
- A merging of action and awareness;
- Clear goals;
- Direct, immediate feedback;
- Concentration on the task at hand;
- A sense of control;
- A loss of self-consciousness; and
- An altered sense of time.

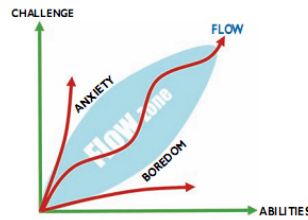


Figure 3. Flow zone factors (Chen, 2007)

A good game keeps the player in the flow zone. This is the challenge of the game designers. Resnick (2004) states that often designers, teachers and parents try to make things “easy” for gamers, learners and children, thinking that they are attracted easyneses. But Mihaly Csikszentmihályi has found that people become most deeply engaged in activities that are challenging, but not overwhelming. Similarly, Seymour Papert has found that learners become deeply engaged by having “hard fun”. In other words, learners don’t mind activities that are hard as long as the activities connect deeply with their interests and passions.

Edutainment

By combining games and learn, the term edutainment nurture in our head. Edutainment attempts to merge both concepts: education and entertainment. As previously said, content should not be provided without a learning context, but the entertainment industry is providing a large amount of content as edutainment. Knowledge can’t be provided as content. More sceptically knowledge can’t be provided anyway; We only can provide contents, hopefully on a real and correct learning context.

Resnick (2004) distinguish edutainment from “playful learning” since the focus is on the play and on the learning, and not on the entertainment and education. We also agree with Resnick that such combinations of games and programming languages should provide new software where the player/student have an active role playing and learning simultaneously.

NVPL offer the right contexts to reach formal thinking, using a transparent and natural environment.

Visual programming languages

Currently there are many visual programming environments. Wikipedia presents a good definition of Visual Programming Languages (VPL), “A Visual programming language (VPL) is any programming language that lets users specify programs by manipulating program elements graphically rather than by specifying them textually.” Using graphics to represent programming blocks provides a better efficiency and simplicity on the interaction between the user and the computer.

Designing a VPL for creating any type of computer program, is a very difficult task. Visual programming is a good way to start to learn to program, to create models with some level of abstraction or to design mock-ups of applications. It would be very hard to program the kernel of an operating system using a VPL, since this type of programming is very tight with the hardware.

Some examples of VPLs are StarLogo TNG and Scratch . Both environments have many similarities, but are designed to be used in different contexts. StarLogo TNG focus on the learning of programming using mostly LOGO syntax and programming rules, while Scratch is on expression of music, dance and arts, but it can also be used in the same context as StarLogo TNG. A computer program is not written in these environments. Instead blocks with different shapes must be combined with each other in a jigsaw way. It is not possible to have syntax errors which are very common in the textual programming language (a missing semi comma, misspelling a command). Many common programming errors of mismatch type variables don’t

happen on these environments, since the block of different types have different shapes and it would not fit on wrong blocks.

Many environments are in fact VPL, but they are just the same instructions that exist on regular programming languages wrapped in images with different shapes and colours. Even so they have the advantages stated before, so they are a preferred media to learn to program. But, do the shapes and colours have a meaningful sense for the user? That is another question, which is not going to be addressed on this paper.

Natural Visual Programming Languages

VPL in most cases can not be considered mini-languages, but as most of the mini-languages, despite the fairly intuitive interface, they can not be used in a natural way. The *reactTable* is a very good example on how electronic music can be built naturally, with a very small learning curve. It is an interface that requires zero manual, zero instruction. Any user can create electronic music (Jordà, 2003). The *reactTable* is sonically challenging and interesting, learnable, suitable for complete novices and for advanced electronic musicians and totally controllable (Jordà, 2003). The same concept could be applied to NVPL, making the use of such environments natural for the user.

Natural Visual Programming Languages (NVPL) would still require time to learn to program, as in the *reactTable* it would require time to learn to build *real* music. But the initial learning curve to use the *reactTable* needs no tutors, tutorials or manuals. The interface is so clear and self-explained that the user easily grasp how to use it. The visual response of the application to the user is so intuitive that can be used by any user naturally. The feedback (visual and sound) provided completely guides the user on how it should proceed (Jordà, 2003). Also, when an object is added to the table, there is an immediate visual and sound feedback of the consequences of adding it.

By using such an environment does not means that the user will be able to build music, but that can do it much more easily. Since the usage of the environment is not any more an obstacle, users can be focused by the learning context to learn musical meaningful contents, by doing. In Mindstorms, Papert (1980) point that children learn a lot on their first years of live, without being able to read or write. They do many mistakes, but if they don't experiment, they don't learn.

In ours days we tend to think that someone unable to read and/or to write can not learn most of the knowledge available, which is not true. Literacy does provide an enormous boost in the possibilities of learning, since we don't depend on the help of others, but there is much more knowledge besides the ones provided by text. Our body has 5 senses, and if all of them can be stimulated, different perspectives of the same information are provided to us, allowing us to create a much more accurate and quickly knowledge. Learning styles and multiple intelligences hypothesis by Gardner also reminds us that using only logical and linguistic reasoning is not the best trend for everyone to learn.

NVPL should become environments where anyone can build computer programs by combining graphical elements, either by using mouse, gesture, voice or any other device, or even by using tangible objects. The main difference is that formal literacy should not be essentially required to begin programming in these NVPLs. In an ideal environment even users with disabilities or autism should be able to use them. Not only visual interfaces should be provided to "read" information, but sound and touch should also be included to stimulate a richer experience to the user. "Good animation and sound effects help greatly in making an environment self-revealing" (Kahn, 2004). The Flow theory by Csikszentmihalyi give us an idea about how this environments should challenge the user with clear goals, but without loosing the sense of control and by getting immediate feedback on every action (or even while no action is taken).

Kahn (2004) states it is very difficult to create a completely self-revealing environment. So to help creating a self-revealing environment the user must be coached on the right way (natural way) and in context; more possibilities to fulfil it purposes would emerge on the context of

programming. Indeed it's impossible to learn without stimulus and without guidance. If we are kept on a empty room with no doors neither windows, our learning will be very limited. As the feral children case studies show us the right kind of early interaction is essential for normal development and learning (Candland, 1993). As much richer and contextualized are the stimulus more possibilities we have for learning. This richness and significant context can be mediated by our parents, teachers, companions, but can also be a book, a virtual companion or a screen computer that makes sense for us and provokes a learning path where we build some meaningful construct for us. This means that we can connect what we are learning with ours previous learning experiences and that this make sense for us.

There are not yet NVPLs softwares. Dragões e Companhia and ToonTalk are some examples of uncompleted NVPL. And they are not prepared natively to be used by people with disabilities. Also the learning process to master these environments is not short. Even thou, these environments use a very different approach to VPL, so we will make an overview of both next.

Dragões e Companhia

Dragões e Companhia is the second generation of the software Floresta Mágica produced by Cnotinfor. The new generation follows the same principles of the previous, but with enhanced engine and sleek graphics.

The main idea for programming is to create rules for objects that are created on a page. The objects are always visual objects (frog, donkey, warrior, car, apple ...). Each object has it own rules which are created by a condition (event) and a sequence of actions that should be performed. There are special building blocks (control stones) that can be put alone in a rule to making the object be controlled by a device like a mouse, keyboard and/or joystick.

The metaphor used for the building blocks is that they are stones that should be put together in a rule, which is a papyrus. An object can have many rules – papyrus. It is an odd metaphor, but the children that used it never thought it has a strange one (Andrade, 2002).

This environment provides tools for building simple and complex games, stories, simulations and animations. The building blocks are 14 events/conditions, 28 actions and 5 controls stones. It seems a small number, but this low number provides a mini-language for systematic problem solving. Since it was designed for very young children (4 to 14 years old) the interval of the parameters of the stones is limited.

The design idea for the engine of Dragões e Companhia is that there are events occurring, and the user must create actions to respond to these events, that can trigger other events.

A nice example of a simulation that can be programmed is two kids (objects) playing with each other and they throw a ball to each other. This same simulation, as almost any computer programs can be programmed in many different ways. A possible solution is to create a rule to each kid (object) that when a red ball it him, throw a red ball.

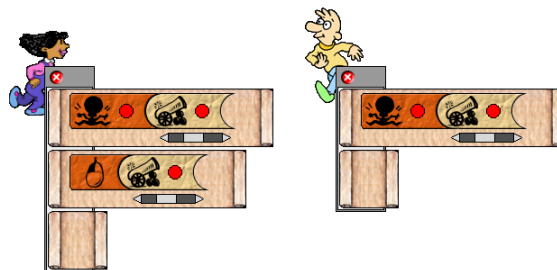


Figure 4. Rules of objects in Dragões e Companhia

There is an extra rule on the girl, when the girl is clicked throw a red ball. This rule is needed to start sending the ball between the two kids.

One of the goods things in favour to become an NVPL is the fact of providing challenging starting points, but without a tutor it is not easy for most of children to engage and explore the starting points in a productive way. Also it provides a very concrete way of representing the constructs that allows the user to think about the solutions created to solve a problem.

Dragões e Companhia has some problems to be NVPL. The interface is not intuitive, and in the firsts approaches it takes some time to get used to the method of creating computers programs. It also lacks of some visual and audible responses on the user interaction. When user first enter in the environment it is not clear what he can do. A blank page is shown and naturally he asks: what can I do? The only thing possible is trying the handles. But by clicking on the handles it's not clear what it is possible to do. Dragões e Companhia should have a mechanism of contextualized coaching in order to approaches it from the NVPL concept. It should offer more stimulus and guidance for building meaningful constructs. When we do things that have no sense, the environment should give advice and feedback pointing to other possibilities that make sense on the context.

It also lacks of the possibility to create new building blocks, which in many cases would help the children to decompose problems in smaller problems. The possibility to extend the current language is one of the major features of programming languages, but Dragões e Companhia lacks of it. So to have a small number of blocks, and not being able to extend them, it is good for a quick learning, but not good in the long run, since children will have more difficulty to express them selves, since only a small amount of blocks are available. Only children more motivated and with a better mathematical knowledge will learn to use them beyond the trivial purpose (Papert, 1980). But once learned it allows the user to build very quickly and easy, many graphically appealing computer programs.

ToonTalk

ToonTalk provides a complete different metaphor for programming from what computer scientists are used to. The programming is done by demonstration in a virtual animated interactive world. All environment objects are familiar to the user. Birds, scales, trucks, robots, vacuum, notebooks, and others. One of the most interesting features is that the source code of the computer program is animated. For very young children this is great, since the objects behave just like ordinary objects. For example, if the user gives an object to a bird it will take it to his nest.

All the objects in ToonTalk have a corresponding computational concretization (Kahn, 1996):

Computational

Computation

agent (or actor or process or object)

methods (or clauses or program fragments)

method preconditions

method actions

tuples (or arrays or vectors or messages)

comparison tests

agent spawning

agent termination

constants

channel transmit capabilities

ToonTalk

city

house

robots (with thought bubbles)

contents of thought bubble

actions taught to robot inside thought bubble

cubbies

scales

loaded trucks

bombs

number pads, text pads, pictures

birds

channel receive capabilities

nests

program storage

notebooks

ToonTalk has a tutorial mode called Puzzle, that teaches many features and methods of usage of ToonTalk, by solving a series of problems that are part of a story. Each problem presents a new feature/tool of ToonTalk. By presenting the tools/features gradually in a predefined order, make it possible to master ToonTalk. Each tool is presented, as it is needed by the user to solve the problem. Similar and very good examples of the success of such approach are computer games like Lemmings or The Incredible Machines. In such games each new tool and ways of solving problems are presented gradually, and only when the user is able to use them, new ones are presented to him. In this way ToonTalk presents most of the features to the user in a contextualized environment (adventure game) and in a gradual form.

During the normal usage of ToonTalk, Marty provides tips and tricks to help the user to use ToonTalk. Also when something is not possible to do, there is a feedback either visual, audible and/or sensorial (if a force feedback joystick is connected to the computer).

ToonTalk is not an environment that a user can naturally use without tutoring, although it has a character called Marty that is always present (if the user wants it), to give feedback about the tools and objects. It also gives some hints on the beginning on how to use every object and tool.

ToonTalk provides modes that could with some evolution, convert it into a more natural environment. The Puzzle mode and the demos are very important modes for learning programming languages, since they provide ways for the user to learn how to use them. But demos must be carefully produced since the current standards of videos are the ones presented today in TV and cinema, with good voice, script and graphically seductive (Kahn, 2004).

One very important characteristic of ToonTalk is that it was designed having social learning in mind. It fulfils this purpose by several ways, but the more interesting ones are: working with a long viewing distance (great to use in classroom contexts), multiple players support and network collaboration without losing its metaphor (long distance birds) (Kahn, 2004).

Conclusion

Natural Visual Programming Languages (NVPL) is still a concept and there is a long path to walk to develop such environments. ToonTalk it seems one of the softwares that more resembles with NVPL.

Creating environments that combine so many features of so many different software types is the next step. ToonTalk might provide the correct framework to build this type of software, but the programming paradigm is very different from what is usually learned and used. Combining the engine of Dragões e Companhia with the learning contexts provided by ToonTalk might be another option to explore.

We believe that this kind of software will revolutionise the way programming languages are thought and that the learning context around this subject would change. The challenge now is to find feasible solutions for each characteristic of NVPL that can be combined in one single solution.

References

- Andrade, M. (2002) *Floresta Mágica: o contributo dos parceiros na construção do Playground*. Dissertação de mestrado. Lisbon: Universidade Aberta.
- Brusilovsky, P., Calabrese, E., Hvorecky, J., Kouchnirenko, A. and Miller, P. (1997) *Mini-languages: a way to learn programming principles*. Education and Information Technologies (2), 65-83.

- Candland, D. K. (1993) *Feral children and clever animals: reflections on human nature*. Oxford University Press. Relates the story of Kamala and Amala.
- Chen, J. (2007) *Flow in Games (and Everything Else)*. Communications of the ACM 50 (4), 31-34.
- Correia, S. et al. (2004) *Micromundos AIA*. Coimbra: Cnotinfor.
- Csikszentmihalyi, M. (1990) *Flow: The Psychology of Optimal Experience*. London: Harper Perennial.
- Jordà, S. (2003) *Interactive Music Systems for Everyone - Exploring Visual Feedback as a way for creating more intuitive, efficient and learnable instruments*. Stockholm Music Acoustics Conference (SMAC 03), Stockholm (Sweden).
- Kahn, K. (1996) *ToonTalk - An Animated Programming Environment for Children*. In *The Journal of Visual Languages and Computing*, Volume 7, number 2, June.
- Kahn, K. (1999) *A Computer Game to Teach Programming*. National Educational Computing Conference 1999.
- Kahn, K. (2004) *ToonTalk - Steps Towards Ideal Computer-Based Learning Environments*. In Tokoro, M. & Steels, L. *A Learning Zone of One's Own: Sharing Representations and Flow in Collaborative Learning Environments*. Ios Pr Inc.
- Leutenegger, S. and Edgington, J. (2007) *A Games First Approach to Teaching Introductory Programming*. SIGCSE'07, 115-118. Covington, USA
- Mendelson, P., Green, T. and Brns, P. (1990) *Programming languages in education: the search for an easy start*. In Hoc, J., Green, T., Gilmore, D. & Samway, R. (eds) *Psychology of Programming*, 175-200, London:, Academic Press.
- Mitchel, A. and Savill-Smith, C. (2004). *The use of computer and video games for learning*. London: Learning and Skills Development Agency.
- Norman, D. (1999) *The Invisible Computer*. London: MIT Press
- Papert, S. (1980) *Mindstorms: Children, Computers, and Powerful Ideas*. New York: Basic Books.
- Piaget, J. (2001) *Studies in Reflecting Abstraction*. Hove, UK: Psychology Press.
- Resnick, M. (2004) *Edutainment? No thanks. I prefer playful learning*. Associazione Civita Report on Edutainment.
- Rieber, L. P. (1996) *Seriously Considering Play: Designing Interactive Learning Environments Based on the Blending of Microworlds, Simulations, and Games*. Educational Technology Research & Development. Georgia: The University of Georgia.
- Wikipedia. (2007) *Computer programming*. Retrieved March 10, 2007 from <http://en.wikipedia.org/wiki/Programming>.
- Wikipedia. (2007) *Constructionist learning*. Retrieved April 2, 2007 from http://en.wikipedia.org/wiki/Constructionist_learning.
- Wikipedia. (2007) *Programming language*. Retrieved March 10, 2007 from http://en.wikipedia.org/wiki/Programming_language.
- Wikipedia. (2007) *Visual programming language*. Retrieved March 10, 2007 from http://en.wikipedia.org/wiki/Visual_programming.

The ICT literacy in the OECD countries

Tünde Dancsó, *dtunde@bp.kodolanyi.hu, dancsot@enternet.hu*

Kodolányi János College – Informatics Department,

H – 1139 Budapest, Frangepán u. 50-56. Hungary

Graduate School of Educational Sciences

H – 6722 Szeged, Petőfi sgt. 30-34. Hungary

Abstract

The usage of ICT is becoming increasingly important in the world of work, so computer skills development plays a key role in education as well. Every young person will need to use ICT in their lives for a full scale participation in modern society. Based on the objective assessments education can be developed more efficiently, this way the assessment of knowledge will have an even bigger role in education. In the last years several international researches were organised about the assessment of informatics skills.

The most significant data were provided by the result of the ICT questionnaire made by the OECD (*Organization for Economic Co-operation and Development*) in 2003. The questionnaire revealed the 15-year-old students' attitudes towards computers, access to computers and internet, self-confidence in routine, internet and high-level ICT tasks and learning methods. In Hungary such assessments have been made, which indicate the informatics skills and the effectiveness of learning methods. This study presents the international and domestic results of informatics assessment.

The results of the PISA assessment provide good grounds for comparisons with results of other IT assessments. Besides the tests based on students' self reports surveys are also needed which measure students' knowledge, skills and abilities.

In Hungary the Center for Research on Learning and Instruction University of Szeged conducted a survey for the 14 and 18 year-old students in 2007 which measured students' knowledge acquired at school and in everyday life. The aim of the IT evaluation is to prove the efficiency of informatics teaching and the objective measurement of students' knowledge.

Keywords

ICT; ICT literacy; PISA; assessment

Introduction

Surveys, researches by means of ICT have been playing an even bigger role in labour market and education as well. The OECD (*Organisation for Economic Co-operation and Development*) is an organization of 30 member countries. Their common aim is to answer economic, social and environmental challenges of globalization. Participation is important in the international assessment because they develop their assessment according to the methodology elaborated for the education of the most developed countries (Csapó, 2005). The results provide opportunity for the identification of problems and planning the objective development strategies. The PISA (*Programme for International Student Assessment*) evaluates the 15-year-old students' performance according to the *index of Economic Social and Cultural Status (ESCS index)*.

The PISA 2003 index of *economic, social and cultural status* is based on three variables:

- highest occupational status of parents (HISEI),
- educational level of parents (PARED),
- and home possessions (HOMEPOS).

The PISA 2003 survey measured students' literacy in mathematics, reading, science, problem solving and the effect of background variables on performance. 30 OECD countries and 11 partner countries participated in the survey. 127 165 students, among them 4765 Hungarians answered the questionnaire. Students in the 31 countries (containing 24 OECD countries) answered an ICT questionnaire, which concerned their access to and familiarity with ICT.

Students provided information on the following:

- ICT was available to them at home, at school or in other places,
- or how often they used ICT,
- how self-confident they felt performing certain tasks on a computer,
- their general attitudes to using computers,
- how they learned to use computers and the Internet and who they turn to for help.

According to the questionnaire indexes were made describing ICT skills. The indexes were divided into quarters based on the ESCS index. Using the indexes the achieved results of Hungarian school system can be revealed and certain elements of the development strategy can be indicated (Dancsó, 2005).

Frequency use of ICT

Students were asked how frequently they performed various Internet and entertainment tasks using ICT. The index of ICT Internet/entertainment use (INTUSE) and the index of ICT program/software use (PRGUSE) were derived from students' responses. Altogether 12 items measure the frequency of different type ICT use.

Positive values on this index indicate high frequencies and negative values indicate lower frequencies of ICT use.

The students had to choose from 5 options below:

- almost every day,
- a few times each week,
- between once a week and once a month,
- less than once a month,
- never.

The frequency index describes the total results, the indexes were standardized. The mean of the OECD countries is 0, the standard deviation is 1.

We can measure with indexes

- to what extent the national indexes differ compared with the international average,
- to what extent the quarter indexes differ compared with the same quarter indexes in the OECD countries,
- what skills our educational system is able to develop independently of the social, cultural and economic background.

Frequency use of ICT for Internet and entertainment (INTUSE)

On average across the OECD countries, 56% of students often use computer for e-mail or chat, 55% of them use the internet to look up information, 53% of them play games, but only 31% of them use the internet to collaborate with a group (1. Table).

How often do you use ...	OECD average (%)	Males (%)	Females (%)
A computer for electronic communication (e.g. e-mail or "chat rooms")?	56	56	55
The Internet to look up information about people, things or ideas?	55	59	50
Games on a computer?	53	70	35
The Internet to download music?	49	56	40
The Internet to download software (including games)?	38	51	25
The Internet to collaborate with a group or team?	31	36	27

1. Table. Students' use of ICT for Internet and entertainment.
(Source: OECD, Figure 3.3, Table 3.3)

The results indicate that the students

- should do exercises in groupwork on the Internet more often,
- should download useful softwares and music from the Internet more often,
- should search for information on the Internet more often.

In case of downloading softwares and playing games there is a significant difference between genders. Twice as many boys play on computer as girls and twice as many boys download softwares from the Internet as girls.

There isn't any difference between genders considering electronic communication.

The INTUSE index in the OECD countries is between -0,91 and 0,63.

The lower indexes indicate that students don't benefit from the information gathering and communication supporting opportunities of Internet efficiently.

	Bottom quarter	Second quarter	Third quarter	Top quarter
OECD	-1,05	-0,32	0,16	1,22

2. Table. Index of ICT use for the Internet and entertainment.
(Source: OECD, Table 3.2)

Frequency of use of ICT for program and software (PRGUSE)

Students use word processing by the highest percentage. 30% of students frequently use graphics programs and 30% use the computer to learn school material. 23% of them use computer for programming, 21% use spreadsheets and only 13% use educational software frequently (3. Table).

How often do you use ...	OECD average (%)	Males (%)	Females (%)
Word processing (e.g. Word)?	48	48	49
Drawing, painting or graphics programs on a computer?	30	34	26
The computer to help you learn school material?	30	31	29
The computer for programming?	23	30	16
Spreadsheets (e.g. Excel)	21	24	18
Educational software such as mathematics programs?	13	15	11

3. Table. Students' use of ICT for programs and software.
(Source: OECD, Figure 3.4, Table 3.5)

The frequencies indicate that students

- rarely use programs for learning
- and rarely use computer for programming (for example creating Logo or Basic programs).

Boys use computer for programming, working with spreadsheets and drawings more often. There isn't any difference between genders for example in using word processing and computer use for learning.

In Hungary the use of word processing, spreadsheets and graphics programs are compulsory so the

- 53% of Hungarian students often use word processing,
- 31% of them often use spreadsheets,
- 30% of them often use graphics programs.

Although the Hungarian students rarely use programs for learning but in the near future a lot of them will use the Sulinet Digitális Tudásbázis (SDT), which is an LCMS e-learning system. They use computers for learning more often because teachers also give them homework more often, which requires computer programs.

The PRGUSE index in the OECD countries is between -0,35 and 0,33 which indicates that the students of the countries participating in the survey apply similar programs in education.

	Bottom quarter	Second quarter	Third quarter	Top quarter
OECD	-1,15	-0,22	0,28	1,14

4. Table. Index of ICT use for programs and software.
(Source: OECD, Table 3.4)

Students' confidence in using ICT

Data were collected about the confidence level of usage with 23 questions. These tasks fell into three categories: routine tasks (such as opening, saving, deleting, moving, copying, creating, editing files, or playing computer games), internet tasks (such as downloading files or music from the internet, writing and sending e-mails, attaching a file to an e-mail message) and high-level tasks (such as creating presentations, using spreadsheets to plot a graph, using database to produce list of addresses, constructing web page, finding computer virus or creating computer programs).

The students had to choose from 4 options below:

- I can do this very well by myself.
- I can do this with help from someone else.
- I know what this means but I cannot do it.
- I don't know what this means.

Based on the results 3 indexes were created:

- confidence in routine ICT tasks (ROUTCONF);
- confidence in Internet ICT tasks (INTCONF);
- confidence in high-level ICT tasks (HIGHCONF).

The indexes were standardized.

The mean of the OECD countries is 0, the standard deviation is 1.

Confidence in routine ICT tasks (ROUTCONF)

The students are confident in doing simple routine tasks, for example opening files, starting games, creating drawings, editing and saving documents, scrolling a document up and down a screen. However they admitted to expecting more help with printing, moving, copying and deleting files (5. Table).

Routine tasks	I can do this very well by myself (%)	I can do this with help from someone (%)	I know what this means but I cannot do it (%)	I don't know what this means (%)
Open a file	90	7	2	1
Play computer games	90	7	2	1
Start a computer game	86	10	3	1
Save a computer document or file	88	8	3	2
Delete a computer document or file	88	8	3	2
Draw pictures using a mouse	85	10	3	1
Print a computer document or file	86	9	3	2
Scroll a document up and down a screen	87	8	3	3
Create/edit a document	80	13	4	2
Move files from one place to another on a computer	76	17	6	2
Copy a file from a floppy disk	75	16	7	3

5. Table. Percentage of students who are confident performing routine ICT tasks.
(Source: OECD, Table 3.9, Box 3.3)

The ROUTCONF index in the OECD countries is between -0,80 and 0,39.

	Bottom quarter	Second quarter	Third quarter	Top quarter
OECD	-1,34	-0,11	0,61	0,81

6. Table. Index of confidence in routine ICT tasks.
(Source: OECD, Table 3.8)

Confidence in Internet ICT tasks (INTCONF)

The INTCONF index in the OECD countries is between -0,81 and 0,77, that also indicates big difference in internet use between students having favourable or unfavourable economic, social and cultural background.

Internet tasks	I can do this very well by myself (%)	I can do this with help from someone (%)	I know what this means but I cannot do it (%)	I don't know what this means (%)
Get onto the internet	89	7	3	1
Copy or download files from the Internet	70	19	8	3
Attach a file to an e-mail message	58	24	13	5
Download music from the Internet	66	20	11	3
Write and send e-mails	79	12	6	2

7. Table. Percentage of students who are confident performing Internet tasks.
(Source: OECD, Table 3.11, Box 3.3)

	Bottom quarter	Second quarter	Third quarter	Top quarter
OECD	-1,23	-0,17	0,51	0,87

8. Table. Index of confidence in ICT Internet tasks.
(Source: OECD, Table 3.10)

Confidence in high-level ICT tasks (HIGHCONF)

The HIGHCONF index in the OECD countries is between -0,71 and 0,43.

High-level tasks	I can do this very well by myself (%)	I can do this with help from someone (%)	I know what this means but I cannot do it (%)	I don't know what this means (%)
Use a database to produce a list of addresses	52	30	11	7
Create a presentation	47	27	15	10
Use a spreadsheet to plot a graph	43	31	17	9
Use software to find and get rid of computer viruses	38	29	26	7
Create a multimedia	35	35	23	7

presentation

Construct a web page	28	39	28	6
Create a computer program	20	34	32	14

9. Table. Percentage of students who are confident performing high-level tasks on computers.
(Source: OECD, Table 3.12, Box 3.3)

	Bottom quarter	Second quarter	Third quarter	Top quarter
OECD	-1,14	-0,31	0,22	1,25

10. Table. Index of confidence in high level ICT tasks.
(Source: OECD, Table 3.12)

Attitudes towards ICT (ATTCOMP)

The ICT questionnaire surveyed the students' attitude to computers. The degree of attitude was defined qualifying statements about ICT activities.

These statements included

- whether students think computer use is important,
- whether they enjoy using them,
- whether they are motivated by an interest in computers and
- whether they lose track of time when they use a computer.

The students had to choose from 4 options below:

- strongly agree
- agree
- disagree
- strongly disagree.

The ATTCOMP index in the OECD countries is between -0,41 and 0,31. The negative score on this index indicates less positive attitude than the average of students in OECD countries.

Of course students from less well-off backgrounds have lower index and well-off students have higher index.

	Bottom quarter	Second quarter	Third quarter	Top quarter
OECD	-1,24	-0,35	0,37	1,22

11. Table. Index of attitudes towards computers.
(Source: OECD, Table 3.6)

The assessment of Hungarian students' knowledge

The results of the PISA assessment provide good grounds for comparisons with results of other IT assessments. Besides the tests based on students' self reports surveys are also needed which measure students' knowledge, skills and abilities.

In Hungary the Center for Research on Learning and Instruction University of Szeged conducted a survey for the 14 and 18 year-old students in 2007 which measured students' knowledge acquired at school and in everyday life. The aim of the IT evaluation is to prove the efficiency of informatics teaching and the objective measurement of students' knowledge.

The survey provides opportunity for evaluating knowledge, skills and abilities in operating systems, word processing, using spreadsheets, making presentations, familiarity with database topics as well as identifying the level of self-confidence in doing IT operations. The questionnaire contains exercises which students meet at school or in everyday life so during evaluation we will be able to compare the level of knowledge at school and in everyday life as well.

According to our hypothesis with this assessment we can reveal

- the difference between genders in certain fields
- the level of development of certain IT skills and abilities
- the level of development of IT means used in other subjects
- the difference between age groups
- the degree of development between the ages of 14 and 18.

Using the data, global and detailed analysis can be conducted and we can create an exact picture of students' knowledge. With these results improvements based on objective data can be planned in education.

References

Balázsi Ildikó, Szabó Vilmos és Szalay Balázs (2005): A matematikaoktatás minősége, hatékonysága és az esélyegyenlőség. A PISA 2003 nemzetközi tudásmérés magyar eredményei. *Új Pedagógiai Szemle*, 11. sz. 3-21.

Csapó Benő (2005): A komplex problémamegoldás a PISA 2003 vizsgálatban. *Új Pedagógiai Szemle*, 3. sz. 43-52.

Dancsó Tünde (2005): New Trends of Using Information and Communication Technology in Hungarian Educational Strategies. *Új Pedagógiai Szemle*, 11. sz. 36-48.

OECD (2005): Are Students Ready for a Technology-Rich World? What PISA Studies Tell us? Online: <http://www.oecd.org/dataoecd/28/4/35995145.pdf>.

Contributing to the Development of Linguistic and Logical Abilities through Robotics

G. Barbara Demo, *barbara@di.unito.it*
Dept of Informatics, Turin University – Italy

Giovanni Marcianó, *marciano@irrepiemonte.it – margi@bmm.it*
National Agency for Development of Schools - Piedmont, Ministry of Public Education - Italy

Abstract

In primary schools robot programming is fun and may therefore represent an excellent tool both for introducing to ICT and for helping the development of logical and linguistic abilities of schoolchildren. Core of our project is NQCBaby, a Logo-like robot programming language, which – in the tradition of Logo – is child-oriented rather than robot-oriented like NQC. In the early 90's S. Papert had already suggested the educational use of "small robots programmed in Logo" [Papert 1993]. We had robots, of course we had Logo, but we had no Logo for robots. NQCBaby is an Italian version of a Logo-like interface to NQC, so that children used to deal with the Turtle can transfer and adapt their implicit abilities to robots, also discussing the differences. In the kind of activities we consider, schoolchildren start by using a very simple language, the first level of NQCBaby, to interact with the simplest robots RCX; later, as their robot-assembly experience grows, they move on to successively richer versions of the language. As a matter of fact, the constructive pedagogical methodology (and consequently our tools) structures the learning of NQCBaby in several steps, starting with NQCBaby01 up to NQCBaby05, as new hardware components are introduced to build new more sophisticated kinds of robots. In the meanwhile children are also introduced to NQC, the "real" robot language, by looking at how their descriptions of robot behaviors are translated into NQC "in order to be understood by robots". This step-by-step activity of schoolchildren is coordinated with the parallel learning of the basic linguistic abilities in their native language. A software environment, based on a precompiler from NQCBaby to NQC, is currently being developed for supporting the project principles, in a sequence of levels of increasing complexity and abstraction from NQCBaby01 to NQCBaby05.

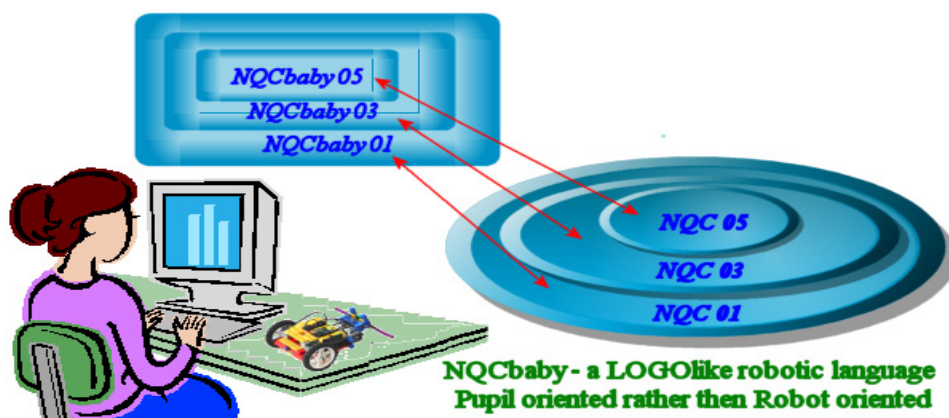


Figure 1. Children assemble a robot, decide behaviour, describe it in NQCBabyh, and look at the resulting specification in NQC; then they enter the next step to learn the next level NQCBabyh+1

Keywords

Constructive learning, robotics, Logo, precompiler, RCX, NQC

1 Introduction

In primary schools robot programming is fun and may therefore represent an excellent tool for both introducing to ICT and helping the development of logical and linguistic abilities of children in their first years of instruction. Robotics experiences have been carried out in Italian schools since 2000-2001 when the very first project “SeT: Let us build a robot together” was proposed; its description can be found at <http://www5.indire.it:8080/set/microrobotica/>. These activities with children convinced us that teaching digital technologies in primary schools should be concerned with the language used for interacting with robots rather than be limited to making robots move as we want to. The project here described introduces children to programming the small Mindstorms RCX robot through different linguistic steps where the programming language evolves with the knowledge the pupils acquire of the robot components. When pupils, at the beginning of their robot experience, build the simplest robots they also use a simple LOGO-like language, called NQCBaby, to interact with them. As children knowledge evolves also the interaction with robots evolves: pupils learn that not only new words but also new way of structuring sentences are needed when different robot components are introduced and different behaviours may be designed and specified.

Thus, for primary school pupils learning robots programming also becomes an opportunity to develop their linguistic and logical capabilities within a context where pedagogical aspects are focused rather than, or before, introducing technological terminology. Also, children can, at the same time, use similar LOGO languages to make turtles move in virtual environments (or microworlds) and to manipulate LEGO RCX robots which are real objects.

At the beginning of the project, we considered using the textual language NQC, with its Bricx Command Center (BCC) integrated development environment, rather than iconic languages because the use of the same format allows most immediate influence exchange between linguistic competences pupils are collecting in their native language and during the robot programming activities [BricxCC, Marcianó 2005]. Also native language translations of most used robot programming languages were considered because they are obviously nearer to the pupils. But analyzed proposals resulted unsatisfactory since we consider mandatory that languages are designed to be pupil oriented rather than robot oriented, and, for example, concepts from the child world must be used to describe a robot behavior. Already in [Marcianó 2004] we pointed out that LEGO system, available from 2002 with Mindstorm (RCX) kits, is too far from the LOGO philosophy because affected by a technological approach. Similarly, from the first experiences of programming the RCX robot with NQC, using the Bricx Command Center (BCC) integrated development environment, we were convinced that pedagogical aspects were sacrificed to technological components.

The LOGO-like language presented by Marcianó in [Marcianó 2006a] is specifically developed for pupils using Lego RCX based kits. In fact, the present paper collects the outcomes resulting from several experiences based on the theoretical concepts inspiring that paper.

NQCBaby, here described, is a proposal toward a language more pupils oriented. During their first experiences, pupils are given robot-assembling boxes containing different sets of RCX components. Of course pupils build, under the guide of the teacher, different robots with boxes having different sets of components, usually robots with an increasing number of abilities as the number of components increases. As an example given a light sensor it obviously becomes possible to distinguish different levels of lights thus becomes possible for a robot to follow a path on the floor marked by a black ribbon. In our approach the language evolves with the pupils knowledge of the environment. The following Section 2. describes the language NQCBaby by means of different steps as it is introduced to the children in our experiences together. Examples of robot programs using different levels of the NQCBaby language are given in Section 3. After the language has been introduced we come back to motivations why we consider it more suitable to children (Section 4).

In Section 5 a short description is given of the software solutions we considered to integrate NQCBaby with NQC and then with Bricx Command Center until the precompiler-postcompiler solution of the language currently under development.

2 NQCBaby by steps

In this section different robots are considered with language elements needed to direct them. During the first meeting each pupils' group is given a box only containing components needed to assemble the simplest robot: it is the so called *chariot* having two wheels moved by motors A and C and no sensors neither lights nor pincers. To interact with this simple robot a simple language is needed, shown in the left column of Table 1. Each interaction begins by saying hallo to the robot called by name, Robby, and finishes saying ciao (bye). The translation of each NQCBaby instruction into the corresponding NQC sequence of instructions is shown in the right column of Tables that follow.

Robby >	task main() {
avanti(v) >	OnFwd(OUT_A+OUT_C); Wait(v);
indietro(w) >	OnRev(OUT_A+OUT_C); Wait(w);
destra(t) >	OnFwd(OUT_A);OnRev(OUT_C); Wait(t);
sinistra(u)>	OnFwd(OUT_C);OnRev(OUT_A);Wait(u);
veloce(x) >	SetPower(OUT_A+OUT_C, x);
avantisempre>	while(true){OnFwd(OUT_A+OUT_C); };
ripetisempre>	while(true) {
ripeti(z) >	repeat (z) {
fine >	Off(OUT_A+OUT_C); }
ciao>	Off(OUT_A+OUT_C); }

Table 1. NQCBaby01 Language.

Examples of different behaviours specifications using the above primitives are the following (from // the translation of Italian instructions is given):

1. Robby
avanti(100) indietro(100) // forward(100), back(100)
ciao
2. Robby
ripeti(4) avanti(100) destra(90) fine // repeat(4) forward(100), right(90) end
ciao

Behaviour 1. is one of the first try a pupil makes for testing if his/her robot moves forward for a while (avanti(100)) and then comes back to its starting point (indietro(100)).

Behaviour 2. is soon produced by children who already have used some Logo, which is often the case in schools concerned with their pupil's introduction to technology. Children try to give the robot the same sequences of commands already used with the Logo turtle for example the one in 2. is for designing a square. Obviously from this code sequence it turns out that right(90) does not mean to the robot to turn right 90 degrees as to the turtle. Discussions after this try are quite valuable for learning differences between the two environments.

Code sequence 2. also shows the use of the primitive `fine` (`end`) to terminate a repeat sequence of commands rather than having the sequence between `{}` brackets not present on an Italian keyboard.

As a second step, we expand `NQCBaby01` with instructions shown in Table 2 concerning light and touch sensors. Usually the touch sensor is already introduced during the second meeting with a class. One new word (`sensor_touch`) appears in the language for the new component and some other words appear to say what to do with the new component: `se_tocca` i.e. `if_touches`.

<code>sensore1_tocco ></code>	<code>SetSensor(SENSOR_1, SENSOR_TOUCH);</code>
<code>se_tocca ></code>	<code>if (SENSOR_1 == 1) {</code>
<code>Accendiluce ></code>	<code>On(OUT_B);</code>
<code>Spegniluce ></code>	<code>Off(OUT_B);</code>
<code>aspetta(z) ></code>	<code>wait(z);</code>

Table 2. Primitives for `NQCBaby02`

`Accendiluce` and `Spegniluce` are primitives for switching on and off the lamp on B actuator. Having motors on A and C, as decided above, the lamp goes on the actuator B. Expressions in Table 2 allow to write programs like the following where the robot goes forward until something is hit. Such hitting causes switching on the light, moving to the right and switching off the light. Again the translation follows the `//`.

```

Robby
sensore1_tocco      // touch sensor goes to sensor number 1 of the robot
  Ripetisempre      // repeat always
    avanti(10)      // forward(100)
    se_tocca        // if hits
      accendiluce    // switch on the light
      indietro(100)  // back(100)
      destra(50)     // right(50)
      spegniluce     // switch off the light
    fine            // end
  fine
ciao

```

In the third step we introduce a second touch sensor to control the rear of the robot (where sensor number 2 is located). When we bring out of the robot kit the light sensor, we have to settle the value we consider the border value for light (`chiaro`) and dark (`scuro`). In our experiences it is reasonable to have 48 as this border value. Light sensor goes on robot sensor position number 3 that is under the chariot on the front.

Expressions in Table 3 are added to the language:

<code>sensore2_tocco ></code>	<code>SetSensor(SENSOR_2, SENSOR_TOUCH);</code>
<code>se_toccadietro ></code>	<code>if (SENSOR_2 == 1) {</code>
<code>senti3_luce</code>	<code>SetSensor(SENSOR_3, SENSOR_LIGHT);</code>
<code>se_chiaro</code>	<code>if (SENSOR_3 > 48) {</code>
<code>se_scuro</code>	<code>if (SENSOR_3 <= 48) {</code>

Table 3. Primitives for `NQCBaby03`

On a fourth step, the language is enriched with primitives in Table 4, that provide new controls directed to the light sensor and for the execution of an alternative sequence of commands, by using the `else` component of the language

<code>sinoache_chiaro</code>	<code>until (SENSOR_3 > 48);</code>
<code>sinoache_scuro</code>	<code>until (SENSOR_3 < 48);</code>

altrimenti	else
acaso(n)	Random(n)

Table 4. Primitives for NQCBaby04

Notice the introduction of the powerful and fascinating primitive *acaso(n)* (*Random(n)*) to explore the randomness world.

On the final step, instructions to control two procedures named *prima* (*first*) and *seconda* (*second*) are introduced as shown in Table 5.

uno	task prima() {
fai_uno	start prima;
stop_uno	stop prima;
due	task seconda() {
fai_due	start seconda;
stop_due	stop seconda;
fine_fai	}

Table 5. Primitives for NQCBaby05

3 Some examples of robots programs using the above primitives

Some examples of robots programs are shown using the above primitives.

```

Robby
veloce(3) avanti(100) veloce(7) indietro(100) // speed(3), forward(100), speed(7), back(100)
ripeti(3) destra(90) sinistra(90) fine // repeat(3) right(90) left(90) end
ripeti(2) indietro(10) avanti(20) fine // repeat(2) back(10) forward(20) end
ciao // hallo

```

A pupil in a second class of a primary school, seven years old, has developed this very simple program based on **NQCBaby01**.

```

Robby
senti1_tocco
ripetisempre
    avanti(10)
    se_tocca
        indietro (100) destra(100)
    fine
fine
ciao

```

During a stage a group of teachers has developed the above program based on **NQCbaby02** defining a robot that goes around the classroom, avoiding obstacles.


```


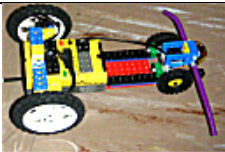
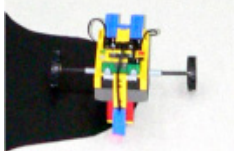
Robby sensore3_luce
  ripetisempre
    se_scufo avanti(10) accendiluce destra(10) spegniluce fine
    avanti(10) sinistra(10)
  fine
ciao

```

A group of pupils in their fourth year of a primary school (nine years old) has developed the above program based on **NQCBaby03**. It defines a robot able to go forward along a border between a black (on its left side) and a white area (at its right side).

4 Why NQCBaby is useful

While introduced to NQCBaby as described above, pupils in first years of primary schools progress in the development of their linguistic and logical abilities because of the use of their native language but also, and more relevant, because they use a **higher level language** wrt NQC language. Higher level because of its *compactness* and because of the *use of children well known concepts rather than using the names of robot components*. As for compactness, we can see from the Tables in the previous Section that several NQCBaby commands correspond to more than one command in NQC. In NQCBaby *concepts well known* to pupils are used such as dietro (on-the-back) for the primitive se_toccadietro (if touches on the back) to distinguish it from the simple se_tocca (if touches) used for if (SENSOR_1 == 1){, introduced in the language when only one sensor_touch was given to the pupils while building the robot. If, on the one side, we lose in flexibility because, as an example, in NQCBaby sensor-2 is devoted to a touch sensor and sensor-3 is devoted to a light sensor, on the other side in their first steps children find more natural describing robot behaviour in NQCBaby. They are more involved in the activities and develop a lot of different programs for their small robots during a single lesson, comparing their jobs and improving their programs.

Robby ciao veloce(x) avantisempre avanti(v) indietro(w) destra(t) sinistra(u) ripeti(z) ripetisempre fine		NQCBaby01 " " " " " " " " "				
accendiluce spegniluce aspetta(z) sensore1_tocco se_tocca			NQCBaby02 " " "			
sensore2_tocco se_toccadietro sensore3_luce se_chiario se_scufo				NQCBaby03 " " " "		
sinoache_chiario sinoache_scufo altrimenti					NQCBaby04 " "	

acaso(n)		“	
uno		NQCBaby05	
fai_uno		“	
stop_uno		“	
due		“	
fai_due		“	
stop_due		“	
fine_fai		“	

Table 6 – Comprehensive view of NQCBaby language

Also, syntactical components as { } , ; from NQC are not present in NQCBaby: the children learn they exists when they begin looking at their code translation in NQC and are given motivations for having them when they ask. After NQCBaby05 they will use NQC for describing their robot behaviour, from the beginning and with the whole expressiveness of the NQC language: but, at that point, they will be familiar with many robot-only language components.

In Table 6 commands of NQCBaby language are listed all together in the first column. From this inclusive view one can see how NQCBaby is in many instructions syntactically near to Papert's Logo which is indeed our constant reference. Logo is popular in Italian primary schools and a number of teachers have developed activities syntonc with Logo philosophy [Papert99]. When beginning an experience of robot programming in a new environment teachers have reported of quite different reactions of pupils depending on whether NQCBaby or NQC are proposed to begin with. Pupils are obviously more confident finding commands having the same names of Logo commands though commands are different because, as an example, for robot we have speed and time while for turtles we have distances and degrees as parameters in some instructions.

Moreover, conceived for third (or even second) year school children which are 7 years old, NQCBaby goal is to encourage pupils to play with concepts such as avanti (forward) / indietro (backwards), sinistra (left) / destra (right) related to the robot position. Also playing with time, space, speed to control robot movements, allows children to manipulate complex concepts that cannot be explored in a different way.

After this “manipulation step”, children are required to program their robots for executing specific tasks. Through these experiences they learn to be precise, because if they do not follow language rules their programs are not understood, also they learn precision in a pleasant and constructive way (the robots do not behave as they would like to). This is common to learning how to program in general but it is a good result when teachers find it out in their first years class, after some programming experiences, having seen actively involved most of their pupils. On this concern Simonetta Siega, teacher in the primary school of Baveno, presents her experiences in the workshop titled “**Seven years old pupils programming turtles and robots using Logo-like language: NQCbaby**” While experimenting, using the first macro version of NQCBaby dealt with in next Section, we had several hints to improve the language toward being more-child-oriented directly from pupils!

5 From NQCBaby to NQC: how

Proposed beginning 2005 in a very simple form to verify its usefulness we are currently devoting to NQCBaby a typically technological activity of implementing a precompiler converting NQCBaby texts into NQC programs.

Macro defined language: quick but not coherent

At the very beginning NQCBaby has been proposed by Marcianó as a macro defined language [Marciano2005] where every primitive *P* having a NQC-translation *Q* was defined in a line of code as follows:

#define P Q

This is of course a very simple yet powerful technique for experimenting the different language layers on top of NQC. Obviously it does not support coherence of the answers from the compiler to the pupils. Indeed syntactical errors are signalled on NQC primitives that pupils, at least at the beginning, do not know. Thus our current effort is to implement 1) a precompiler to signal NQCBaby syntax errors on NQCBaby primitives and 2) a postcompiler to translate NQC compiler errors messages on NQCBaby error messages.

The precompiler output can be shown to the children so they can learn what their code looks like in NQC because, as a last step shown in Figure 1., children program in NQC in order to have the entire flexibility of this language in dealing with any assembly of a robot they can envisage: even those that children call “crazy”.

NQCBaby Precompiler

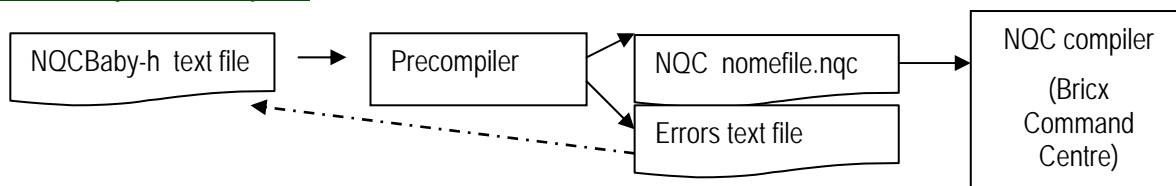


Figure 2. Children assemble a robot, compose its behaviour in a NQCBaby-h text file, go back to the beginning if errors are found or look at the translation in a NQC file to be compiled and sent to the robot.

In this paragraph we briefly describe how NQCBaby precompiler does currently work, as shown in Figure 2. Children define the behaviour of their robot according to an NQCBaby any level language in a text file. This is given in input to the precompiler which either returns as a result a syntax error notice, obviously concerning the NQCBaby code, or returns a text file containing an NQC code sequence as from Tables above. The latter is then given in input to the NQC compiler in BrixCC for the translation into an executable code sequence.

Our present activity is toward implementing a tool easier to use for children and looking similar to the Bricx Command Center where they work for the final translation from NQC and for sending the executable code to the robot.

Indeed the tool under development will have two work modes. First use the mode NQCBaby-only mode, where only NQCBaby is considered consistently with the environment provided by the BCC (Bricx Command Center) environment. Thus an editor of NQCBaby supports syntax highlighting and code completion for the NQCBaby language.

The second work-mode is called NQCmode because it adds to the NQCBaby only mode the translation of each NQCBaby primitive into NQC commands. Thus pupils can see does it look like the other language the robot can understand: it is a first step toward learning the NQC language.

The postcompiler integrates with the precompiler to rewrite errors found in NQC code by NQC compiler, into syntax errors in NQCBaby code listed at the end of the NQCBaby code page. Clicking on the error message BCC (Bricx Command Center) highlights the line where the error has been found: a professional function realized in a friendly way.

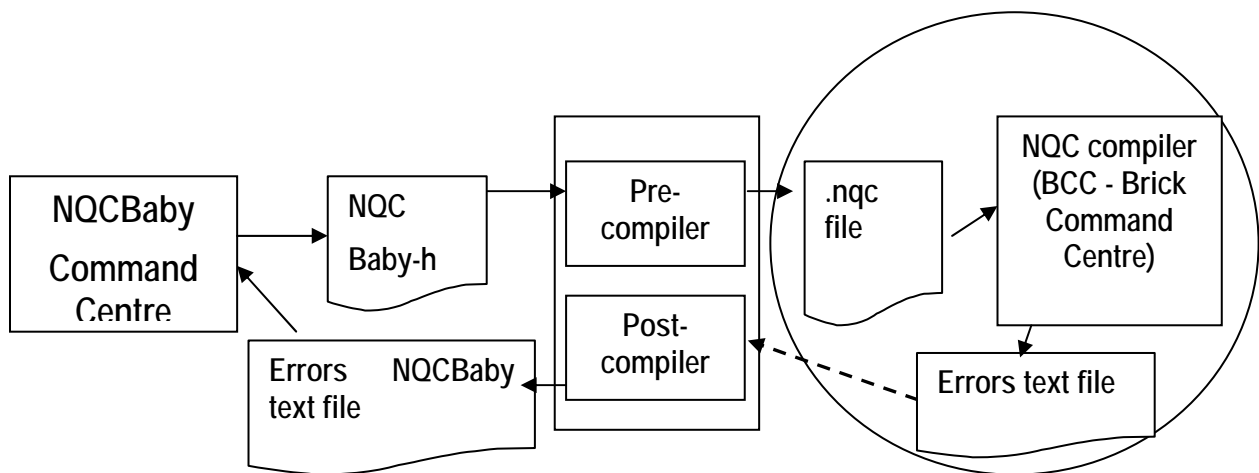


Figure 3. The new component Postcompiler with errors listing on NQCBaby.

NQCBaby Postcompiler

Next development in our systems is shown in Figure 3. where the Postcompiler component is the relevant new entry.

Errors found by the BCC (Bricx Command Center) during compiling NQC into the executable code are captured and translated by the Postcompiler, using some Precompiler data, into NQCBaby errors. Thus the Postcompiler component allows the translation of errors found in NQC code by NQC compiler, into syntax errors in NQCBaby code. Errors are then listed at the end of the NQCBaby code page. Thus when used in NQCBaby-only mode the system will coherently signals errors on the NQCBaby code: essential for pupils during their first experiences of robots programming when they don't yet use NQC.

6 Final remarks and acknowledgments

Proposed for five years, the ideas described in this paper have been tested by many colleagues and their pupils. Simonetta Siega and her colleagues deserve a special thank-you for suggestions and comments on children reactions during their several years of robot experiences in primary schools of Baveno, Tortona and other schools in Piedmont. Images and reports with comments on these activities can be seen on the web at: <http://robotica.irrepiemonte.it/robotica/index.htm>

Our goal with NQCBaby is to develop a methodology rather than a language. Indeed, our next activity will concern moving the project to languages for programming other small robots: we think this can be done adding to the methodology some other components. Currently we are working on defining an analogous by-step language and set of tools for LEGO NXT programming (NBC language referred). With NXT a number of different robots can be built: thus we are defining a language where robot programs contain two distinct components: the robot *configuration description* and the robot *behaviour description, that is the program*. This is an evolution in programming of small robots that makes it converge toward the fundamentals of computer programming defined as *Algorithms+ DataStructures = Programs* by Wirth in his work about Pascal around the same years when Papert and his group were working on Logo [Wirth 1971, Wirth 1976, Papert 1980].

References

BCC - *Bricx Command Center site*, at: <http://bricxcc.sourceforge.net/>

Marcianó, G. (2004) *Linguaggi robotici a scuola*, at

http://margi.bmm.it/old/robotica/Linguaggi_robotici_a_scuola.pdf

Marcianó, G. and Siega, S. (2005) *Linguaggi robotici per la scuola*. In Proceedings of DIDAMATICA 2005, Edited by A. Andronico, N. Cavallo, A. De Michele, M. Fasano, Potenza, pp. 185 - 196.

Marcianó, G. (2006a) *Informatica come Linguaggio*. In Proceedings of DIDAMATICA 2006, Edited by A. Andronico, F. Aymerich, G. Fenu, Cagliari, May 2006. pp. 185 - 196.

Marcianó, G. (2006b) *Robotica project site*, at

http://robotica.irrepiemonte.it/robotica/linguaggi/doc/linguaggi_robotici_scuola_2006x.pdf

Miglino, O.- Lund H. H. – Cardaci, M. (1999) *Robotics as an Educational Tool* in JI. of Interactive Learning Research (1999) 10(1), 25-47

Papert S. (1980), *Mindstorms: Children Computers and Powerful Ideas*, Basic Books, 1980.

Papert, S. (1993). *The children's machine: rethinking schools in the age of the computer*, Basic Books, 1993.

Papert S. (1999), *Logo Philosophy and Implementation*, LCSI, Canada, 1999

Wirth, N (1971) *The Programming Language Pascal*. Acta Informatica, 1, (Jun 1971) pp. 35-63

Wirth, N. (1976) *Algorithms+Data structures=Programs*, Prentice-Hall Series in Automatic Computing. Eaglewood, N.J.: Prentice-Hall Inc. 1976.

Logo: an object to think with

Mícheál Ó Dúill, mikedoyle@logios.org; mike.doyle@bollingspecialschool.org
Bolling Special School, BD4 7SY UK; 37 Bright Street, Skipton, BD23 1QQ UK

Abstract

When Logo was first developed it had no Turtle and was seen as an aid to learning mathematics and a means of introducing adults to computing. Seymour Papert failed to see why children shouldn't program computers (rather than being programmed by them) Logo came to school.

In this piece I aim to turn turtle on the Turtle and treat Logo, or rather what happened in school with LOGO, as an object through which to think about human learning and creativity. But most of all, given that we have no conception of how it is that humans can draw, I want to address the question concerning graphics.

My Turtle came to school in 1983 without Logo but with a BASIC implementation of Abelson & diSessa's Turtle Graphic commands. I liked neither the Turtle, reduced as it was to a graphics plotter nor the computer – speechless with the typewriter keyboard that school had previously rejected. Working with the kids in my class we soon had a set-up that helped them learn. This was achieved by creatively changing the technology (fig A1). It didn't catch on.



Figure A1. Topless Turtle

After a number of years, it became very clear that LOGO in school was all about drawing with the Turtle. This was followed by graphics in MicroWorlds, Imagine or MSW Logo. Why graphics?

It is very difficult to notice something that is not there. So, it was a number of years before I fully appreciated the import of the absence of knowledge about drawing. It was a trifle difficult to research a field in which there is not any knowledge. It was only by researching language and its evolution that I began to understand why drawing was so important. All the artefacts that steadily advanced technologically after we evolved are assembled from simple geometric components: Hence the question concerning graphics.

When I enquired into how children drew; the question arose: From whence comes the data?

The search for data and its processing led me to Darwinian evolution, feature neurons, the silent prefrontal cortex, the nature of children's drawings and why a square becomes a diamond when rotated. The outcome is a notion of how humans are creative and why technology is intellect.

Keywords

LOGO, Logo, Turtle Graphics, drawing, evolution, language, brain, learning, creativity, school

Introduction

In England, there is a national shortage of teachers who can take the lead in ICT in the classroom. This is despite, or perhaps because of, a quarter century of Government investment in the use of computers in school. A part of the problem is that ICT is skills-based, with little or no intellectual content. Given the importance of the Turing machine for developing our understanding of matters as fundamental as the origins of the universe and the nature of mathematics, this seems perverse.

Unfortunately, the programming of computers – writing for a Turing machine – is not considered a mental skill like mathematics or language literacy. The curriculum of our schools emphasises capability: the practical skills that 'Capability' Brown celebrated in landscape gardening rather than the scientific enquiry into things natural that was the foundation for Darwin's speculations.

The source of this approach to ICT is the character of learning and teaching. Education is the exercise of craft skills and technique within a well established, print-oriented, knowledge-based, conceptual framework. Here, the computer, as evaluable writing, has no place. Its role is as a magic cupboard brimming with classroom resources. The interactive whiteboard is the apogee of this gadget oriented, class teaching, approach to ICT in School.

The present approach to ICT contrasts with the child-oriented group-work classroom into which Logo was first introduced. The change in ethos and organisation was initiated in England by the British Government in the form of the English National Curriculum (Graham 1993); and then consolidated by a target-driven approach to educational outcomes. Logo should have been, and could yet be, the counterpoint, but aspects of the early promotion of Logo brought it into concert.

I would like to rehearse some of the early history of Logo (in the UK) and then take a step back to reflect upon some initial findings of the pioneers. Thereafter I ask a rather different question about how human beings learn, one more in tune with technological fluency and construction.

A little history

As Margaret Thatcher came to power in the UK, Seymour Papert published *Mindstorms*. The two had rather different ideas about education. Margaret Thatcher imposed the English National Curriculum. Papert was less dogmatic: he wrote of the LOGO Turtle as an object to think with (1980:11). He likened it to the physical gears of his childhood play, which helped him visualise his maths. This gave him the notion of a mathetic (1989:39), learning, microworld. Papert's ideas were attractive to the English primary education tradition, which at the time, influenced by Piaget, was child-centred and revolved around group-work within an integrated school day. The manner in which "LOGO" entered the classroom was nicely described by Anderson (1986), who brought out the two major strands: Turtle Graphics and Control.

Control evolved into LEGO®/LOGO and the RX brick. It forms a separate strand, both in Logo and school-subject terms: mechanics as opposed to mathematics. Its intellectual home was MIT.

The route to "LOGO" in school began with a programmable toy: the Milton Bradley tank *Big Trak*; followed by the Floor Turtle; and finishing up as an arrow the screen. This remains basically so today. The Valiant *Roamer* and *Pip* and *Pixie* from Swallow Systems have replaced Big Trak and Floor Turtle. Computer generation children find a screen turtle, now turtle shaped, easier.

Logo vTurtling

The most notable aspect of Anderson's (op cit) survey was the discussion of the relative merits of a full (sic) Logo over a Turtle Graphics package. This was not trivial, because it went to the heart of School's approach to the computer. The Logo that schools were offered was very different from the Logo originally devised by Wally Feurzeig (undated) at Bolt, Beranek & Newman or McArthur (1980) at the Edinburgh University DAI the late 1960s and early 1970s.

This Logo is to be found in Harvey's (1985, 1986) three volume *Computer Science Logo Style* Goldenberg & Feurzeig's (1987) *Exploring Language with Logo*; and somewhat in *LogoWriter*.

Logo, as it came to school, was about the aspects of shape and space that the new *Turtle Geometry* (Abelson & diSessa 1980) explored. As these authors explained, Turtle Graphics packages could be written in any computer language – they noted Pascal and appended an implementation of their Turtle Procedure Notation in BASIC, whilst remarking that their notation was very close to Logo. Papert's (1980) seminal work *Mindstorms*, with a Turtle on the dust jacket (fig.1) and the title "Turtle Geometry" for Chapter 3, set the scene for School.



Figure 1. The illustration on the dust jacket of *Mindstorms*

Personal history

This is how I, as a teacher of children with learning difficulties, was introduced to Logo. I already had a BBC Microcomputer equipped with a Star Microterminals *Concept* (overlay) keyboard and a Votrax *TypeNTalk* (text-to-speech synthesiser) because I didn't want my kids to fight with a typewriter keyboard and text-to-speech could help with beginning reading – but that is another story. So, my Edinburgh Turtle came with *DART*, at Turtle Graphics package, modelled upon Abelson & diSessa (op cit) and written in BASIC. This was presented to me as "LOGO". I had great fun making my Turtle talk and designing overlay control panels for the kids to work it with.

I was not entirely new to the notion of the Turtle. In 1969 I had been introduced to William Grey-Walter (1961:113), the designer of its precursor, a cybernetic tortoise called *Machina speculatrix*. Hence, it is unsurprising that my approach to Turtling as a teacher of children with special learning difficulties focused both on its cybernetic character and communication possibilities.

Out of the box, my 1983 Edinburgh Turtle was little more than a mobile graphics plotter, which was a disappointment. Furthermore, unlike the structural apparatus I was used to, it gave little help to the children. Why, I mused, were they to be restricted to the geometers 360° notation when I was teaching points of the compass, the time, and fractions? Why were children expected to 'discover' what numeral sequences related to which distance and turn when I had protractors and rulers to hand? And for children who were uncertain about their left and right, why was the colour coding strategy I used not encouraged. I decided that maths teachers were a very odd bunch. So, I cut the top off the turtle's dome and replaced it with a platform on which to mount a protractor and pointer. The children drove it around a model village using an informative overlay for control with the computer 'speaking' the commands entered so that the child guiding it could monitor the instructions it had been given by his or her partner (fig.2a).

Interested in combining Control with Turtling, I devised a much modified Edinburgh Turtle (fig.2b) that had 'senses' of sight, sound and touch, and which knew which way to turn and by how much when the pointer was rotated to its new heading. All great fun! Annaturtle had her début at the first (proceedingless) Eurologo Conference in Dublin, where she blew up a BBC Microcomputer. (We only use it for word-processing.) There is a short description of both in Denis (1992) and a longer one in Doyle (1886). None of this work used Logo, just a hacked-about *DART* package.



Figure 2a. Two of my 1984 class driving our Topless Turtle

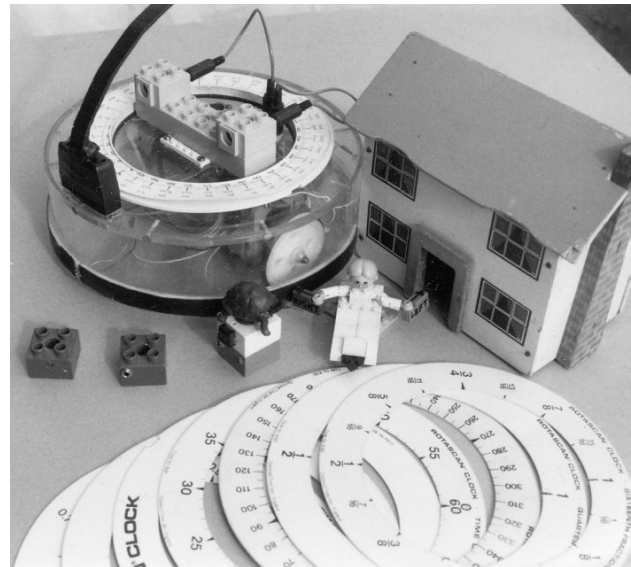


Figure 2b. The Turtle in its Dublin form

Logo and the Turtle were as disjunct in school as the authors of *Turtle Geometry* had asserted. It was only when I came to use *LogoWriter Robotics* that I, and the children I was teaching, had real need of Logo's LISP heritage. Even then, I found an overlay keyboard (LCSI interfaced a Concept keyboard) a far more informative interface to *LogoWriter* projects than the computer keyboard – a problem later solved by the advent of mouse and on-screen graphic environments.

Did the Turtle do for Logo?

The Turtle and Logo are disjunct¹. When you look at the research undertaken on Logo (Hoyles & Sutherland 1984) and the materials produced for schools (Blythe 1990, The Advisory Unit 1990, SMP 1990), it is obvious that the Turtle with its drawing capability had educationalists fascinated. But it is more than the Turtle: the Turtle is told, is programmed, to draw. The Turtle is an object to be taught to draw geometric shapes and childish drawings: in the name of mathematics.

The same has been true within the Eurologo community. At least four conferences featured the Turtle on the cover of the Proceedings; and Geomland, an honourable exception, is graphics-oriented. Of course the Turtle is a totem, a symbol; as the flyleaf photograph of Papert holding a live one demonstrates. But of what so humanly important and unique is it a symbol?

We have been held in thrall by this ancient shelled metaphor. And it has distracted us from the original notion of Logo as a notation for helping children 'teach' the computer rather than being drilled by it. The proof that we have let 'writing for the computer' fall by the wayside is the growth of graphic devices, point-and-click, and dialogue boxes relative to the effort put into making the editor helpful; LogoWriter an honourable exception. Comenius Logo, and Imagine, embraced the

¹ Such was the confusion over just what 'Logo' might be – confusion compounded by another rather less than Classical Greek usage of 'logo' to denote a graphic device or emblem – that Turtle Graphics packages were marketed as Logo without qualification. Consequently, in England, the term "Logo" may now legally be applied to any Turtle Graphics package (NYCC 1993) written in any computer language

graphics of the screen by including shapes (rather their names) in its list processing capabilities and then facilitating the direct manipulation of lists of graphics in the LogoMotion shape editor. So, what is it with graphics?

The question concerning graphics

Before I try to answer this question, I will take a short detour into language. Papert, immersed in an Artificial Intelligence culture that was highly engaged in language processing, saw computer programming as “talking to your drawing” by using TURTLE TALK (1980:56). The best way, therefore, to learn Logo was the immersion method used to teach a second natural language. And this, undirected, structured by the microworld, approach was that adopted in schools. All the research cited assigned great importance to the quality of discussion engendered by Turtle Graphics. But a computer language is not 'speech' and nor is mathematics: both are graphics.

The evolution of speech

Our capacity for speech is a Darwinian adaptation that evolved over the two to three million years it took our species to evolve (Pinker 1995). The suite of physical adaptations include finer control over breathing and the oral cavity, and a descended larynx; plus finger and eye pointing. Two parts of the brain, Broca's and Wernike's areas are associated with productive and receptive language respectively (Carter 2000:21). Palaeontological evidence, taken together, situates the emergence of a modern language capability with the common ancestor of our species and the Neanderthals, about half a million years ago (Dunbar 2004:122, Leakey 1993).

Dunbar (1996) suggested that speech evolved from primate grooming to provide for community cohesion in larger groups. This rather begs the question. For what reason might speech be a successful adaptation? Trivers (1977, 2002) proposed an evolutionary driver: reciprocal altruism. This individual strategy can only be successful in a defensive culture of cooperation; otherwise defecting free-riders become ascendant. Such a culture makes high cognitive demands: good memory, recognition of self and others, and sanctions on defectors. Communication is the co-operator's defence and defector's weapon. This leads to a linguistic arms race (Nettle 1999).

Language evolved for gossip and negotiation. Linguists (Burling 2005) note that hunter-gatherer conversations revolve around debts and relationships. Practical instruction in tool use is restricted to “Do it like that”. Language is spoken and heard. We carry the necessary equipment around in our evolved bodies. Where the language function fails properly to develop, as in certain autistic children, much of what we consider human vanishes with it.

The language trap

In concert with Papert, and most academics and philosophers, I fell into the language trap.

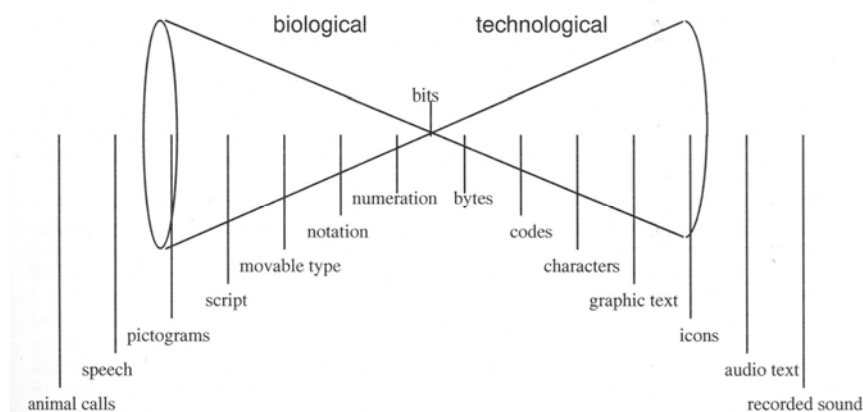


Figure 3. The Athens Language Cone

My Athens Eurologo2003 paper included what I called the “language cone” (fig.3). Looked at today, it clearly has little to do with language and more to do with what we use language to talk about. I mistook the medium for the message.

My error became glaringly apparent, as I read more about the evolution of our species. The very earliest date for the appearance of modern technology: component-built tools to flexible designs using material to hand was a quarter of a million years ago (McBrearty & Brooks 2000). Before this artefacts were a phenotype extension constructed to an inbuilt template, like a bird's nest.

Graphics is unique to our species: our evolutionary adaptation. The archaeological and genetic evidence converge on a central date of about one hundred and fifty thousand years ago in Africa or its, and our, emergence. All the elements I placed in the Language Cone are graphic-based.

The Bugged House

Both Mindstorms and Turtle Geometry open with a discussion of bugs, errors in thinking. Papert (1980:14) emphasises the Bugged House (reproduced as fig.4a), whilst Abelson & diSessa (1986:7) choose to give the Triangle Bug (reproduced as fig.4b) pride of place.

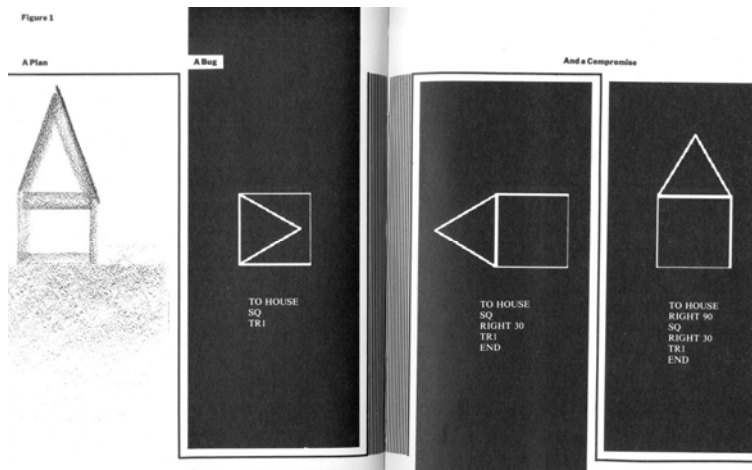


Figure 4a Papert's Bugged house

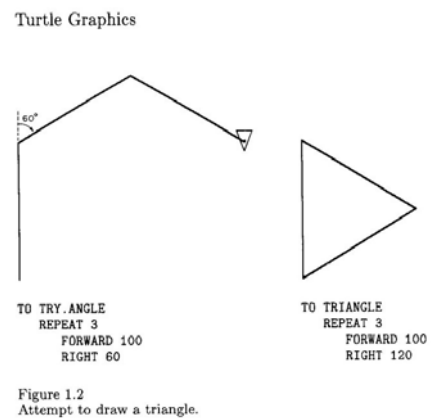


Figure 4b. Abelson & diSessa's Triangle Bug

In the case of the Triangle Bug, the authors considered transfer knowledge of internal angle from classic geometry to be the cause: the Turtle turns through the external angle. For the House, the bug is a failure to keep in mind the position of the turtle and its heading. Two solutions are proposed. Walking and talking Turtle (body geometry); and encouraging children to understand the need for an 'interface' procedure. Thus was any conflict between a Turtle Geometry of paths in space and a Euclidean Geometry of Platonic forms worked around.

When Sendov introduced the Plane Geometry System (Filimonov & Sendov 1989), later called Geomland, the psychology of the two systems of representation could be considered in a Logo context. Sendov's point, line, and arc were mechanical objects, the products of a metaphorical compass and ruler: live things that could be animated into mechanisms, not just a dead trail.

Children and shapes

Children begin by stacking blocks, cubic blocks. How less natural a form than a cube? Houses are triangular blocks on top of square ones. Children assemble components: bricks, wheels, gears, motors, and sensors. The 'interface' – picking up a suitable brick and transporting it to the appropriate location in an appropriate orientation – is not a part of the mental plan. The 'named' procedure of Turtle Graphics denotes the form. Procedural arrangement is cognitively different

The Triangle bug occurs only where a shape is to be drawn and the apex is associated with “60”. The bug never happened in my set-up where the pointer on the turtle was physically turned to

the new heading. The context influenced the response, as has been noted with Piagetian tasks. The findings from Gestalt psychology, and Mme. Montessori, all point to there being something very fundamental about Platonic. The very earliest 'modern' component-built technology entailed the replacement of flakes off a core by retouched geometrics (McBrearty and Brooks op cit).

Squares and Diamonds

Below is a 'bug' known to all mathematics teachers. In Turtle Talk: 'sq' and 'left 45 sq' (fig.5).

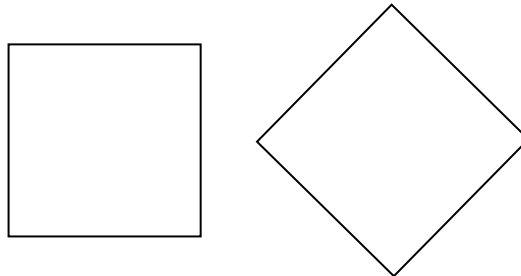


Figure 5. Two shapes.

You can carry out the experiment very easily – a restaurant table napkin is a good object. Hold it up and rotate it. All will agree that it is still called a napkin – our perceptual capacity to maintain object constancy – whatever its orientation. If you hold it with the edges horizontal and vertical, all will confidently announce that the shape is a square. Rotate it one eighth of a turn, however, and suddenly it has another name: diamond. This phenomenon is not peculiar to English. Humans see a different shape when a square is rotated. (One said “triangle” for the diamond.) The conundrum is why.

The silence of Psychology

We have no idea how human beings draw. The science of human behaviour, classical, e.g. Gross (2001) or evolutionary, e.g. Barrett et al (2002), is silent. Gregory (1998) demonstrates our susceptibility to illusion with a plethora of illustrations, the genesis of which is unknown.

The failure of psychology to consider the behaviour that makes technology possible is made all the more poignant by the decision of graphic designers to use prehistoric cave paintings on the covers of Gaulin & McBurney (2001) and Dunbar (2004). The latter cites the work of Lewis-Williams on shamanistic rock art. He found that trance-induced hallucinations can be similar to geometrics like those on the walls of the caves at Lascaux. This does not provide enlightenment on the evolutionary 'how' question. On the other hand, it is interesting in the light of the failure to teach a language-using chimpanzee the art of joining dot-to-dot (Inversen & Matsuzawa 2001), which we may compare with their skill in direct manipulation and use of stone tools to crack nuts. But in general a look at the index of any introductory psychology text will confirm that drawing appears to have been forgotten since the Harris-Goodenough “draw-a-man” test fell into disuse.

We have some idea how graphic capability develops in childhood (Cox 1992 Anning & Ring 2004), but this tends to be from a representational art perspective. We also know that our drawing adaptation develops after speech. It is profusely displayed on the wall of elementary classrooms. Below (fig.6) is a collection from the three to four year-old age range.

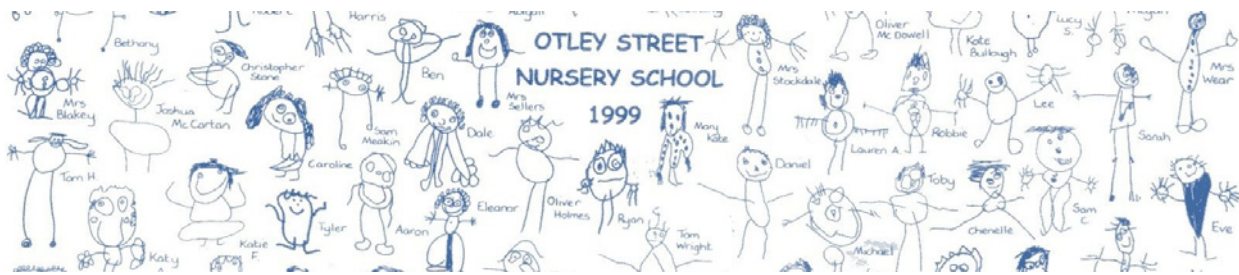


Figure 6. Children's early drawings.

This form of drawing lasts into adulthood, as a pioneering engineer's sketch (fig.7) shows.

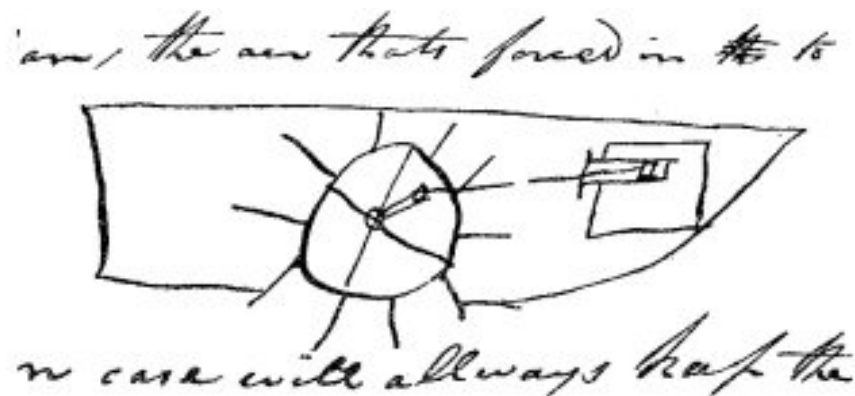


Figure 7. Richard Trevithick's 1806 sketch for a steam boat

The lower illustration is not a representation but a concept. What of the children's drawings? From whence comes the data for these figures? The image at our eyes is nothing like these drawings. The natural world is all irregularity. So why are the drawings composed of geometrics?

Data

The question is the classical philosophy question about the origin of Platonic forms and the redness of red. The experiment with rotated squares points to a source of data in the brain. Our brains are mammalian, of the primate variety (Carter 2000). Compared to reptiles, mammals have additional nervous tissue which confers a greater capacity to recognise the environment, learn from experience and communicate. The brain decomposes sense data and stores it.

The way incoming data is analysed into discrete features was determined by Hubel & Weisel (Hubel 1995). Lines (edges) of different inclination excite specific neurones in the primate cortex. Thus, our square in its two orientations will produce separate and distinctive neurological activity. That we name these two sensations differently tells us that we somehow know this.

The most significant difference between our species and *H erectus* and *H neanderthalensis* is that we have a much larger prefrontal cortex (Leakey 1993 Stringer & Andrews 2005).

The 'silent' brain

Highly connected to older parts of the brain, which it dominates (Deacon 1998:265), activity in the prefrontal lobe appears unspecific. The prefrontal neurones seem to 'live off' the data from the older parts of the brain. This was adaptive in a species with a reciprocally altruistic lifestyle.

The square/diamond experiment suggests that the 'new' brain has access to the earliest stages on visual processing. The evolution of the prefrontal cortex appears to have given access to a 'knowledge of line' that is embedded in the nervous system. A consequence is that we can draw. The very character of children's drawings, once infant's limited fine motor control is allowed for,

is geometrical: points, curves and straight lines assembled to represent the major engineering features of the human figure. The same is true of children's house drawings – and Trevithick's boat. Writing is also a collection of geometric forms, later run together in adult scrawl.

Feature neurones extend beyond simple shape. They abstract colour and sound: the atoms of art and music; and they also abstract movement. With movement, the sense data goes beyond the expressive capability of speech. For all that an infinity of utterances is possible, language is only able to 'name' entities. Movement can only be spoken of. It takes a Turtle to express movement and the Turtle is a machine, as is the computer within which it resides as data.

Data alone is not sufficient, it has to be processed. It appears that our prefrontal lobes may function as a creative workspace where atomic features can be combined to produce entirely novel data. Exported to the natural world, this would alter the sense data input, and thereby expand the database. The process is recursively creative leading to a capability to improve and redesign artefacts and transform natural materials in a manner radically different from primates.

Education

With this source of internal atomic data in mind, we can begin to look at what education does. The first step, however, is to take as read the prior evolution of the human social lifestyle and language. Schools, as institutions, place much store by language development, socialisation, and equity. The latter reflect the lifestyle of our species, with an ever present tendency to defect; language is the capability we deploy to negotiate within the cooperative/selfish social milieu.

Default to speech

Because we arrive in school speaking, it is an easy option to redirect the graphic elements that emerge in our earliest drawings into graphemes. Deeply embedded in our social structure is the ethics of reciprocity; the wariness of strangers; the fear of defection; and our inherited primate troupe structure. The gossip that serves us in these settings, once creatively recorded, becomes the literature and stories that hold us in thrall. Writing stories is the easy option in school. The effect selects for amenable aspects of mathematics: computation is the most obvious. Learning 'number' by oral methods fits the default-to-speech mould. Speech offers School an entrée. It offered an entrée for Turtle Graphics on the grounds that Turtling encouraged group discussion.

Education, it seems, is at the same cultural stage as philosophy: Sassure's 'langue' our crowning achievement as a species, the while begging the question of how we are able to speak about it.

Default to speech, "speaking computer," has led to intellectual problems elsewhere. An area close to my heart is text-to-speech synthesis. Production of intelligible synthetic speech from text by grapheme-phoneme mapping is straightforward. But our Pygmalion tendency made a natural sound the goal. When computing prosodic features proved impossible: the work-around was a large speech database. Speech engines like those for Imagine obscure the graphics of text by introducing the schwa and elisions to mimic speech. We lost an educational entrée to writing.

Graphicity

Although the neural mechanism I have hypothesised can access all sensory areas – hence we can colour and make music – it is the graphic capability that education, and our success as a species, makes most use of. From a social viewpoint, a mark has the merit of setting in stone the spoken contract. But more importantly graphicity is the entrée to technicity. Once we are able creatively to generate non-extant images, we are able to plan construction.

The entities we call 'symbols' are constructions. We can assemble systems of graphic creations to perform many tasks. The 'words' of speech may be recorded using grapheme-phoneme mapping, as in Korean. They may also be represented independently of any dialect, as in Chinese – which may be 'read' in English. Numbers and their manipulation may be represented mechanically, as on an abacus, or graphically as in the decimal system. Construction can be

carried out by bricolage manipulating the materials directly, or graphics can be used to plan. A drawing may be a geometrical exercise or a blueprint for a construction, static or mechanical.

From the Logo perspective, the atoms of graphicity accessible within our brain led, after about one hundred and fifty thousand years of recursive constructive refinement and elaboration, to the electronic stored program digital computer and its mathematical counterpart, the Turing machine. This has taken the graphic medium into a new phase where the graphics may move. It is for this reason, I believe, that the Turtle held in thrall a profession that could not understand.

School's dilemma

The schooling versus learning debate in education is more than a philosophical one. The former capitalises on our superb memory and language; adaptations that evolved in precursor species. The latter accesses internal data and creatively constructs; and is founded on our own peculiar adaptation. The former guarantees a goodly supply of 'internal data' but mainly in linguistic form. The latter runs into trouble with differential access to data, its availability and validity. Thus, our social heritage of speech and superb memory is counterpoised a recent constructional creativity.

The academic, language-based approach is institutionally more reliable and standards can be set and performance assessed using written essays. The constructional approach demands projects and portfolios, is idiosyncratic and difficult to assess to common standards. However, in an increasingly technological world, verbal fluency is not adequate we need to be technologically fluent, as Papert put it (LCSI 1999). Our species cannot 'write' itself out of global warming.

Discussion

Logo, as promoted by Papert, never was just computer programming for kids, although it is that. It carried a philosophy derived from Piaget and formed in the image of MIT. This philosophy carried a view of learners that went beyond concern for equity to self-realisation and autonomy. This approach resonates with the concerns of a teacher of children with learning difficulties.

However, from the same perspective, philosophy is inadequate: it is pre-scientific. Philosophers weave words in an attempt to understand through language. Evolutionary evidence suggests that language does not have the intellectual power required: it evolved for other ends. We need good science behind our educational practice (Piaget 1971) rather than a craft and tradition from pre-computer eras. A 'philosophy' (Logo or otherwise) is not the way forward.

In this piece Logo has been taken for what it is: an object to think with. The history of the Logo project tells us that the computational is not understood in education. Logo projects, exciting at germination, have not flourished: the institutional ground was too stony. It is impossible to cause a system change without good scientific evidence. This the constructionist agenda has lacked. In looking at Logo as an object and asking the question concerning graphics we can begin to build the necessary scientific base. But in so doing we need to recognise that the intellectual power of science is in the technology that we create to extend the capabilities of our gene's phenotype.

The Darwinian perspective deployed here was founded in graphic data. This perspective, by the observations and conclusions it has guided, has produced a body of evidence that enables us firstly to challenge the primacy of language; and secondly – given the economic cost of the silent brain – to suggest a mechanism for creativity and argue its fierce intellectual power. This notion is not directly supported by evidence. It was developed from the problem of explaining how we are able to draw, why children draw as they do, and why a square becomes a diamond. The part of the brain that enlarged hugely in our species alone, has access to the database of the older primate, mammalian, brain. Given that we are able to isolate atomic perceptual data, the idea that Darwinian processes drove the adaptation of creativity into technology does not run counter to our understanding of evolution. All that is needed is the realisation that the genetic event that caused the step change from reptilian to mammalian brain structure – a step that proved adaptive – could occur again to further increase data processing capacity; if it proved adaptive.

There is a new balance to be struck in school. Present practice elevates elegant language above other 'subjects' whilst it relegates technology to the utilitarian or to decorative arts. Science, a subject that bridges language and technology does have a reasonable status in school, third place to language and mathematics. But this is a recent phenomenon, a product of the pressures of the process of industrialisation. But neither science nor stories would exist if we did not have an evolutionary adaptation that enables us to create and then refine our creation.

References

- Abelson H & di Sessa A (1986) *Turtle Geometry* Cambridge Mass: The MIT Press
- Advisory Unit (1990) *Primary Logo Pack* Hatfield: Advisory Unit Microtechnology in Education
- Anderson B (1986) *Learning with LOGO* Loughborough: Techmedia
- Anning A & Ring K (2004) *Making Sense of Children's Drawings* Maidenhead: Open University
- Barrett L Dunbar R & Lycett J (2002) *Human Evolutionary Psychology* Basingstoke: Palgrave
- Burling R (2005) *The Talking Ape* Oxford: Oxford University Press
- Blythe K (1990) *Children Learning with LOGO* Warwick: NCET
- Carter R (2000) *Mapping the Mind* London: Phoenix
- Cox M (1992) *Children's Drawings* London: Penguin Books
- Deacon T (1998) *The Symbolic Species* London: Penguin Books
- Denis B (ed) (1993) *Control Technology in Elementary Education* Berlin: Springer Verlag
- Doyle M P 1986 *The Microcomputer as an Educational Medium: with specific reference to special needs children* Unpublished M Phil thesis Manchester University
- Dunbar R (2004) *The Human Story* London: Faber & Faber Ltd
- Dunbar R (1996) *Grooming, Gossip and the Evolution of Language* London: Faber & Faber Ltd
- Feurzeig W (C1979) *The Early Days of Logo* Unpublished typescript
- Filimonov R & Sendov B (1989) *Drawing Logo Closer to the Curriculum* In G Schuyten & M Valke (eds) *Teaching and Learning in Logo-based Environments* Amsterdam: IOS
- Gaulin S J & McBurney D H (2001) *Psychology: An Evolutionary Approach* London: Prentice Hall International
- Goldenberg EP & Fuerzeig W (1987) *Exploring Language with Logo* Cambridge Mass: The MIT Press
- Graham D with Tyler D (1993) *A Lesson for Us All* London: Routledge
- Gregory R L (1998) *Eye and Brain* 5th edn Oxford: Oxford University Press
- Grey-Walter W (1961) *The Living Brain* London: Penguin
- Gross R (2001) *Psychology: The Science of Mind and Behaviour* London: Hodder & Stoughton
- Harvey B (1985, 1986) *Computer Science Logo Style Vols 1-3* Cambridge Mass: The MIT Press
- Hoyles C & Sutherland R (1984) *The Logo Maths Project* London: Institute of Education
- Hubel D (1995) *Eye, Brain and Vision* New York: Scientific American Library
- Inversen I H & Matsuzawa T (2001) *Establishing Line Tracing on a touch Monitor as a Basic Drawing Skill in Chimpanzees (Pan Troglodytes)* In T Matsuzawa ed 2001 *Primate Origins of Human Cognition and Behaviour* Tokyo: Springer Verlag
- LCSI (1999) *Logo Philosophy and Implementation* Quebec: Logo Computer Systems Inc

- Leakey R & Lewin R (1993) *Origins Reconsidered* London: Abacus
- McArthur (1980) *A12 LOGO User's Guide and Reference Manual* DAI Occasional Paper 14
- McBrearty S & Brooks A S (2000) *The Revolution that Wasn't* J Human Evolution 39 5 453-563
- Nettle D (1999) *Linguistic Diversity* New York: Oxford University Press
- NYCC (1993) Letter to Logo User Group from North Yorkshire Trading Standards Department
- Papert S (1980) *Mindstorms* Brighton: Harvester Press
- Piaget J (1971) *Science of Education and the Psychology of the Child* London: Longman
- Pinker S (1995) *The Language Instinct* London: Penguin
- SMP (1990) *Maths with a Micro USING LOGO* Cambridge: Cambridge University Press
- Stringer C & Andrews P (2005) *The Complete World of Human Evolution* London: Thames & Hudson
- Trivers R L (2002) *Natural Selection and Social Theory: selected papers of R L Trivers* Oxford: Oxford University Press

Knowledge vs. Creativity: A False Dichotomy

Evgenia Sendova, jsendova@mit.edu

Institute of Mathematics and Informatics, BAS, Sofia 1113, Bulgaria

with

Celia Hoyles, c.hoyles@ioe.ac.uk

London Knowledge Lab, Institute of Education, University of London

Márta Turcsányi-Szabó, turcsanyine@ludens.elte.hu

Dept. of Media & Educational Technology, ELTE University, Hungary

Gerald Futschek, futschek@ifs.tuwien.ac.at

Institute of Software Technology and Interactive Systems, Vienna University of Technology

Ivan Kalaš, kalas@fmph.uniba.sk

Department of Informatics Education, Comenius University

Wallace Feurzeig, feurzeig@bbn.com

Principal Scientist, Learning Systems, BBNT, Cambridge, Massachusetts, USA

Brian Harvey, bh@eecs.berkeley.edu

Computer Science Division University of California, Berkeley

Eric Klopfer, klopfer@mit.edu

Associate Professor, Director, MIT Scheller Teacher Education Program, MIT Media Lab

Paul Goldenberg – virtually, pgoldenberg@edc.org

Education Development Center, Inc., Division of Mathematics Learning and Teaching, MA

Tina Ebey, ebey@aol.com

Pacific Research Center, Inc., East Palo Alto, CA

Abstract

- “Knowledge Society” or “Creative Society”—which society are we preparing our children for?
- Are computers used at school as expressive tools for knowledge construction?
- How do we enhance learning to design, invent, and express oneself creatively by means of ICT?
- What is the wrong way to use computers to teach kids?
- When and how should we introduce young children to programming?

The panel presenters will try to find answers to these questions together with the audience based on their personal experience. As an excellent appetizer and challenge we present here short notes of **Paul Goldenberg** to the questions listed above:

- For any society, rural or urban, rich or poor, success depends on both knowledge and creativity. Some knowledge is “just” broadening: No matter what one’s circumstances, if one knows only about those circumstances (the practical knowledge for living in that way at that time), one’s mind is limited. I don’t “use” knowledge of Beethoven, but it makes my limited world richer. But even *within* my limited world, extending knowledge beyond what my parents had gives me choices to improve that world. The enormous change that Fábio Rosa wrought through rural electrification in Brazil – helping rice farmers *remain* rice farmers but also survive – depended on more than knowledge, of course. He could not have succeeded without extreme determination, creative problem-solving, a willingness to challenge, and so

on. But neither could he have succeeded without knowledge. Knowledge without creativity is too static; no change can be expected. Creativity without knowledge is too weak; no effect can be expected.

- Some people argue that there's so much knowledge to be had that the most important thing is to "learn how to learn" and all the details can then just be looked up when one needs them. I don't believe that. So much discovery (medical discovery is a prime example) depends on serendipity – I'm studying the effects of various drugs on one disease process and notice that one of them reduces some particular protein level associated with a completely different disease.... We encounter so much data, so many things we could take note of, that we routinely (and necessarily) dismiss things that seem irrelevant. It takes knowledge *in one's head* to detect what is and isn't "relevant."
- We (at least some of us) used to think that computers in schools could help to change schools toward a realization that knowledge is constructed. We wound up (probably not because of computers) with a bifurcation: knowledge is handed down (the old idea); knowledge is "invented" (a perversion, or at least a misunderstanding, of "constructed") by children. The latter seemed to suggest that nothing need be passed from generation to generation; whatever ideas the child invented were good enough. Both are anti-intellectual. The first expects the child to do no thinking and just absorb; the second requires no grounding in the reality of a growing body of culture and knowledge. Before computers can be used as expressive tools for knowledge construction, we have to get past this one-or-the-other split – either "knowledge" or "construction." Historians do literally (re)construct history, but not oblivious to the data they get. Even as children, we can do some of the creative inventing of theories about what might actually have happened, but we need, as the professional historians need, lots of data to work from. We can't simply make it up. Personal knowledge is constructed (that is, I can't put my ideas in your head; only you have access to your head); but not just made up (or all our realities would bear no relation to each other).
- Alas, one thing that's wrong with the way computers are used to teach kids is that they – like books, manipulative materials, science experiments, and teachers – can only be used the way we already conceive of teaching, and the most widespread views of teaching/learning are stuck on that binary split of knowledge and creativity. The U.S. has "solved" the problem by terrifying teachers/schools into focusing back on a list of the 40,000 things to know. Computers are used extensively (almost, though not quite, exclusively) for some version of test-prep. This is far from the Logo dream we had.
- When we should introduce young children to programming presupposes that we ever will! In the U.S., we don't do it at all any more. My 1999 EuroLogo (Bulgaria) paper on the dwindling of formal language – the remarkable coincidence of the reduced interest in programming (no longer a battle about whether Logo or BASIC or Pascal was best for "thinking," but a complete loss of interest in programming except as a job skill) with reduced emphasis on algebra and grammatical analysis and phonics – presented my take on the reasons for this: again profound anti-intellectualism.

Unless we can address the anti-intellectualism – an issue in society *outside* the schools – interventions within the schools (technological or otherwise) are unlikely to take root.

Designing Logo Interactive Activities for the Mathematics Programs of the Mexican School System

Ana Isabel Sacristán, asacrist@cinvestav.mx

Dept. of Mathematics Education, Center for Research & Advanced Studies (Cinvestav), Mexico

Nicolás Tlachy, ntlachy@yahoo.com

EFIT-EMAT, Instituto Latinoamericano de la Comunicación Educativa (ILCE), Mexico

Rocío Escobedo, rocio_tierra@hotmail.com

EFIT-EMAT, Instituto Latinoamericano de la Comunicación Educativa (ILCE), Mexico

Abstract

In the past decade, the Mexican Ministry of Education has been making intense efforts for incorporating digital technologies. One of such efforts is the Enciclomedia programme which provides teachers with a system of computer interactive resources and activities designed to be used mainly on electronic interactive whiteboards. We were asked to design Logo interactive activities for such a system, to be used in the lower secondary mathematics programs of Mexico. This presented many technical and didactic challenges, the foremost of which was the difficulty of preserving the spirit of Logo, and its benefits as a programming and constructive environment, in a situation where the interactive activities have to be used as self-contained “instructional” presentation tools by the teacher on an interactive whiteboard, with limited typing possibilities. The didactic design was thus crucial. We give an overview of the interactive activities we designed (such as the one in Figure 1), and exemplify the didactic design through the detailed description of one of the activities.

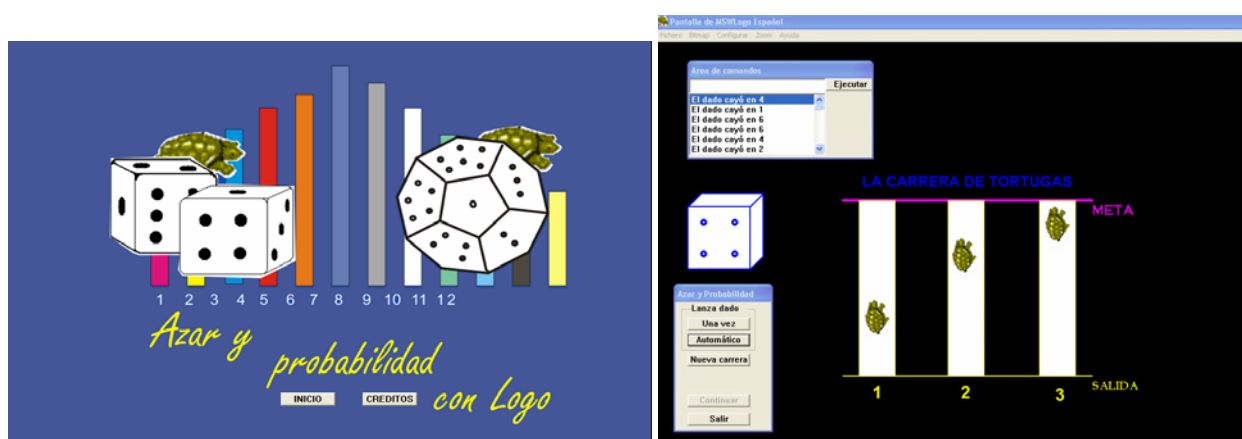


Figure 1. Scenes from the Logo interactive activity “Randomness and Probability” which includes a turtle race: each turtle goes forward depending on what number is generated by a dice. But the race is unfair, and users have to predict the rules that make each turtle go forward, and modify the rules to make the race fair.

Keywords

Mathematics; interactive activities design; Logo; interactive whiteboard; Enciclomedia

Introduction: Technology in the Mexican School System

Since 1997, the Mexican Ministry of Education has been making intense continuing efforts for incorporating digital technologies into the classrooms of the basic education system (primary and lower secondary levels)¹. Some of the largest projects in this effort are the *Teaching Mathematics with Technology* (EMAT) and *Teaching Physics with Technology* (EFIT) programmes for lower secondary schools (children 12-15 year-olds), which began in 1997; and the *Enciclomedia* programme for primary schools, which began in 2003. The EMAT programme provides activities and a constructivist pedagogical model for incorporating the use of technological tools in classrooms in order to enrich the teaching and learning of mathematics (Ursini & Rojano, 2000). The EMAT model promotes student-centred exploratory and collaborative activities in computer laboratories (or with graphing calculators). The main tools currently used in EMAT are Spreadsheets, Dynamic Geometry, Logo and CAS activities with the TI-92 calculator. On the other hand, Enciclomedia –which has been massively implemented in all public primary schools² in Mexico in the past two years— aims to help teachers by providing resources, computer interactive activities and strategies –designed to be used mainly on electronic interactive whiteboards— through links in an enhanced digital version of the mandatory textbooks (Lozano et al., 2006).

In 2005-06, an extension of the Enciclomedia model to the lower secondary level was considered; and particularly, incorporating the model to the “Tele-Secondary” (*Telesecundaria*) School programme. The latter programme –which began four decades ago, in the late sixties, as a very innovative project (Castro et al. 1999)— is an educational model of the Ministry of Education that aims to reach the wider community (e.g. in rural areas) that may not have access to regular lower secondary schools: in a Telesecundaria school, learning has traditionally been structured through three types of educational materials: learning guides, content guides, and television programs; with one teacher-promoter for all subjects. Despite its successes, the Telesecundaria programme –because of, for instance, fixed transmission schedules of the television programs— didn’t allow for many opportunities for students to express, interchange and develop ideas. The Telesecundaria model has recently been renewed: the vision has been to design learning activities that promote discussion, collaboration and critical analysis through the use of a variety of resources and didactic materials, with the teacher acting as a link between the students and the knowledge. In this renewed model, information and computing technologies (ICT) are seen as the potential agent for change, not only to enrich the teaching and information resources and forms of representation, but also to create situations that promote discussion and communication practices in the classroom and in which the student can have a more active role in his learning. At a first level, Telesecundaria schools are being equipped with the Enciclomedia hardware: that is, a computer for use in the classroom with projection equipment for multimedia material (e.g. on an interactive whiteboard) thus giving the possibility to carry out interactive activities. At a second level, the vision is that these schools will have media labs with computers and/or other tools like graphing calculators and sensors.

Thus, in the past two years, intense efforts have been made to develop activities for both the possible lower secondary Enciclomedia programme, and the new model of Telesecundaria. In the case of mathematics, since the use of the EMAT tools and materials is expanding in the country (and a great proportion of lower secondary schools already use them), and it is hoped that eventually the Telesecundaria schools will also have the possibility to use them, it was

¹ According to the official statistics of the Mexican Ministry of Education (<http://www.sep.gob.mx/work/appsite/nacional/index.htm>, retrieved 30 March, 2007) in the academic year 2005-2006, there were over 14.5 million students registered in primary schools; and almost 6 million students in lower secondary schools, out of which 1.2 million (20%) were in Telesecundaria schools.

² Mexico has over 90,000 public primary schools.

decided to try to develop some activities that would create a bridge between the EMAT activities, and the interactive activities for the Telesecundaria and lower secondary Enciclomedia programmes. In particular, since Logo is one of the EMAT tools, we were asked to design interactive activities using Logo for the mathematics programme of Telesecundaria (and possibly of the regular lower secondary schools) to be used with the Enciclomedia system.

The Challenge: Teacher-centred Logo Interactive Activities

The didactic challenges: preserving the spirit of Logo

The interactive activities using Logo had to be designed to be used mainly by the teacher in an interactive whiteboard environment. We were thus faced with a great challenge. The Logo philosophy promotes student-centred exploration and construction; a constructionist approach to learning where “programming itself is a key element of this culture” (Papert, 1999; p. xv).

The Logo EMAT activities, while trying to comply with a set school curriculum (see Sacristán, 2003), adhered to the Logo philosophy principles: For the incorporation of Logo into that project, we placed emphasis on a constructionist approach (Harel & Papert, 1991) where mathematical learning could be derived from student-centred programming activities.

With the interactive whiteboard we had to design activities to be used by the teacher that still provided some of the advantages of Logo. If we were going to use Logo, we needed to preserve the spirit and benefits of this tool, and not use it simply to present something that could be done just as well (or better) with another piece of software.

In fact, some early activity proposals –made before we were asked to join the design team—used Logo only as an underlying platform on which to create animations for some instructive presentation. The criticism was that there was no difference between those Logo-based proposals and equivalent ones using other animation software like Flash; so, why use Logo? Again, if Logo was to be used, its didactic advantages had to be exploited. But how?

The didactic design was thus crucial, and so was the choice of themes.

The first thing was to try follow, in general, the Logo philosophy. Thus, for the didactic design we tried to have activities that would engage the whole classroom in collaborative activities of exploration and, as far as possible, *construction*. And related to the latter, we were also concerned with not “betraying” the potential of Logo as a programming language. We will try to illustrate, in a later section, how we tried to include this in the didactic design, through the example of one of the activities.

Also, we had to take into account that the teachers would not be familiar with Logo, and that in the case of Telesecundaria they may also not be mathematically proficient (since in that system the “teachers” are guides, and not necessarily well-trained mathematics teachers). This meant that the interactive activities had to be self-contained, and self-explanatory (more on this later). But we wanted to use this “limitation” to our advantage, since, in Papert’s words, “a crucial aspect of the Logo spirit is fostering situations which the teacher has never seen before and so has to join the students as an authentic co-learner”; that is create a “relationship of apprenticeship in learning”, where “the student should encounter the teacher-as-learner and share the act of learning” (Papert, 1999; p.ix).

In addition to the didactic design, we needed to find mathematical themes that would provide “powerful ideas” (Papert, 1980) and benefit from a Logo-based presentation; but that we could also incorporate easily into teacher-centred interactive activities. One obvious choice for us, was the inclusion of activities on fractals, because fractals are so easy to construct and describe using Logo, due it simultaneously providing an accessible language and recursive capabilities. In fact, we were specifically requested to design interactive activities on fractals and infinity, so that these topics, that are normally not part of the school curriculum, could also be explored in the lower secondary schools. For other activities, we looked through the EMAT activities (Sacristán

& Esparza, 2006) and chose some of the richest activities mathematically speaking. We give a detailed description of the themes and activities we have chosen and designed, further below.

The technical challenges: the version of Logo, and the system limitations

The choice of themes and the didactic design that preserved some of the spirit of Logo, were not the only challenges. We have also been faced with other technical challenges.

One of these, is the problem of the version of Logo to be used for the interactive activities. The EMAT programme uses MSWLogo; one of the main reasons for that choice was that MSWLogo is an open source version, available in a Spanish, that we could upgrade and make it freely available (see Sacristán, 2003, for further details). For the new interactive activities we were again restricted to the use of some freeware version of Logo and preferably available in Spanish. That meant that we couldn't use commercial versions of Logo with integrated easy-to-use design and multimedia capabilities like Imagine, or Microworlds Logo. We looked into several freeware versions of Logo, but, in the end, it was decided that if EMAT used MSWLogo, and if one of the aims was to link the new activities with some of the EMAT ones, we should use the same version as EMAT, despite the many technical restrictions it imposed.

However, after almost two years of working in the design, we have realized that, in some ways, the choice of using MSWLogo has also affected the didactic design. One of the arguments for using MSWLogo in EMAT, besides the main reasons explained earlier, was that the simple interface could make students focus more on the programming aspect through the writing and debugging of procedures and that we did not need, or even want advanced features, nor to develop sophisticated microworlds or environments (Sacristán, 2003). But in our current situation, the latter is exactly what we are doing: we *are* creating sophisticated interactive environments and we have been restricted by the possibilities of the software. Nevertheless, we have made the best of the software we are working with, and know that the use of the same version for the interactive activities and EMAT will be easier for the users in the cases when both models are used in a school.

Another challenge is that the interactive activities have to be "inserted" within the Enciclomedia system. That means that, not only are the activities opened through links in the Enciclomedia system, but the entire activity is executed within a frame of the system (with the Windows platform actually hidden from the users). This is why most of the non-Logo interactive activities that have been designed for this project, are Java applets. In our case, we had the problem that the MSWLogo software had to be loaded unto the machine that will be using the activities; we sorted this out, by adding the installation file of MSWLogo to the installation program of the Enciclomedia system.

We also faced many other technical requirements, such as using a maximum screen resolution of 800 X 600, the use of specific colours, and the need to comply, as much as was possible, with the established look-and-feel for all the interactive (non-Logo or Logo) activities being designed.

Other challenges had to do with the use of the interface, since the activities would be presented on an interactive whiteboard. This meant that we needed to restrict the amount of typing to a minimum. This was something that we considered mainly in the didactic design, but also tackled from the technical perspective:

To give commands to Logo's turtle, we created a button command window (see Figure 2) with the basic graphic primitives (forward, right, repeat, penup, etc.). (Note: the commands are given without abbreviations, so that they can be self-explanatory.) When the 'repeat' button is pressed, a window appears asking what commands the user wants to repeat; these commands can be inserted into the window using the other command buttons. Also, whenever any of the buttons are used, we made sure the corresponding commands appear in Logo's text screen. We were also requested that we add an 'undo' [*deshacer*] button; we didn't like the idea much (after all, we want the users to reflect on what the turtle had done to deduce what needs to be typed to correct a path), so this button only undoes the last command by giving the turtle the inverse instruction to the prior one.



Figure 2. The button graphic commands window.

Although we tried to restrict typing as much as possible, the need to type commands at some point was inevitable (and, after all, we still wanted to have some programming activities). At first we only considered restricting typing from the perspective that we wanted the activities to flow easily in the teacher-in-front-of-classroom situation, and we didn't think that some typing would be a problem. Then we realized that we also had to consider how to input the characters. We knew that interactive whiteboards have a virtual keyboard, and we thought that would be sufficient. But when we tested the activities with a real interactive whiteboard, the use of the whiteboard's virtual keyboard was extremely difficult. Two possible solutions for this are: (i) To recommend to the teachers that they use the computer's physical keyboard; this is the easiest but at the same time often impractical solution (e.g. the computer may be far away from the interactive whiteboard, etc.); although a wireless keyboard could in theory be used. (ii) The second solution we considered was to program an MSWLogo virtual keyboard. "Fortunately" for us, the problems with the interactive whiteboard's virtual keyboard was a generalized problem not exclusive to the Logo activities, so the Enciclomedia system developers are now adding to the Enciclomedia toolbox a link to open and use Windows XP's On-Screen Keyboard. We have not yet tried this out with an interactive whiteboard, but we hope it will work better.

Finally, we had the problem of how the Text/Command window of MSWLogo normally appears, as it includes several buttons which were confusing for non-trained teachers and users (such as Pause, Trace, etc.). We also had a problem of an overlapping and an overcrowding of windows. For these reasons, we modified the Text/Command window of MSWLogo to make it smaller and so it would only include the "Run" [Ejecutar] button. The new Text/Command window can be seen in the upper left corner of Figure 5.

The interactive activities



Figure 3. The opening screen of the "Fractals" interactive activity

In the following sections we will try to illustrate the didactic design of the interactive activities.

The general design of the activities

Each activity is opened through a link in the Enciclomedia system and begins with an opening screen (see Figure 3). After pressing the "Begin" [Inicio] button, the activity begins.

At the bottom of the screen there is a bar with buttons for Instructions, Purpose (or Objectives), Didactic suggestions, Activities and Exit (see Figure 4). All of the interactive activities (Logo and non-Logo) have to have these required links, because the activities have to be self-contained; these instructions and objectives relate to the entire activity.

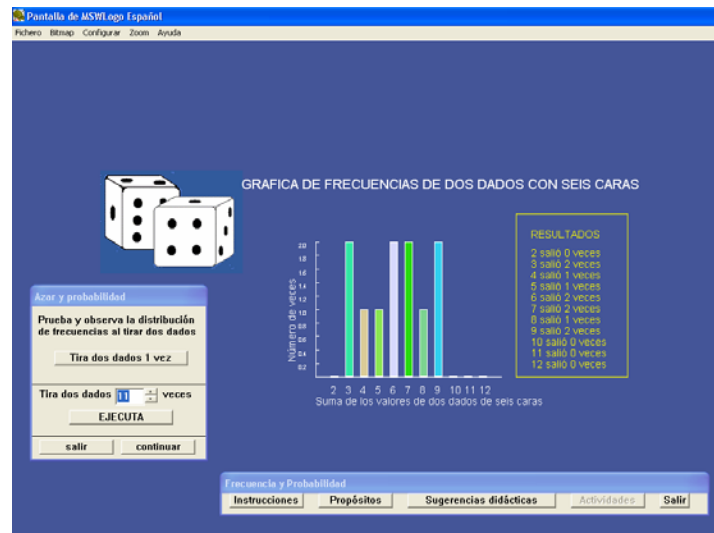


Figure 4. A scene from the interactive activity “Frequencies and probability”. Notice the button bar at the bottom with the Instructions, Purpose, Didactic suggestions, Activities and Exit links.

Each interactive activity is composed of sequence of “scenes” or sub-activities; by pressing the “Activities” button, the user can see which are the sub-activities and jump to a different scene; if its not appropriate to be able to jump to another scene (e.g. if a previous sub-activity needs to be completed) then this button appears dimmed. If the sequence of sub-activities is followed sequentially, then a “Continue” button takes the user from one scene to next.

The different interactive activities

Earlier we mentioned that one of the challenges for the interactive activities, was the choice of themes. So far, we have developed activities in the following themes, mostly based on activities from the Logo EMAT materials (Sacristán & Esparza, 2006):

- The construction and properties of regular polygons: This activity is illustrated in a section below.
- Randomness and probability: This activity (see Figure 1) has as aim to introduce the concept of randomness, and to analyse probabilities. It includes a turtle race where each turtle goes forward depending on what number is generated by a dice. But the race is unfair, and users have to predict the rules that make each turtle go forward, calculate the probabilities that each turtle has for going forward, and modify the rules so that the three turtles have equal probability to advance; i.e. to make the race fair.
- Probability and frequency distribution: This activity (see Figure 4) has as aim to analyse the frequency of times that each number “falls” when a dice is thrown, and calculate each number’s probability. The frequencies and probabilities are analysed for a 6-sided dice, for a 12-sided dice, and the latter compared with those of two 6-sided dice. To help in the analysis, frequency graphs and percentage tables are used.
- Ratio and proportion: In this activity, users are given a procedure to produce an “L” letter. They are then asked to draw similar “L” letters of different sizes, and then to modify the procedure so that it creates an “L” of another size. Then the users go through the same sub-activities for producing the drawing of a house. In the final scenes, the aim is to modify the “LetterL” and “House” procedures using variables so that they become general procedures.

- Sequences and recursion: This activity aims to introduce recursion. It has sub-activities presenting rotating figures and others of sequences of numbers.
- Geometric sequences: The aim of this activity is to study simple geometric sequences through their graphic representations and their Logo instructions that define them (such as bar graphs and spirals), and reflect on “what happens at infinity”. The activity ends with the example of how a fractal tree can be produced.
- Fractals: This activity (see Figure 3) presents different examples of fractals. It also includes sub-activities where the perimeter and area of Koch’s snowflake is analysed.

(The last three activities, although also included in the EMAT materials –Sacristán & Esparza, 2006— are based on the work by Sacristán, 1997).

The “Regular Polygons” activity

Below, we present a detailed description of the activity on regular polygons. This was the first activity we developed and we based the design of all the other activities on what we learned from designing this one. Its didactic design should illustrate how we tried to find solutions to the technical and didactic challenges presented earlier.

The regular polygons activity consists of six “scenes” or sub-activities. We will present each of them in turn.

Scene 1: Examples of regular polygons

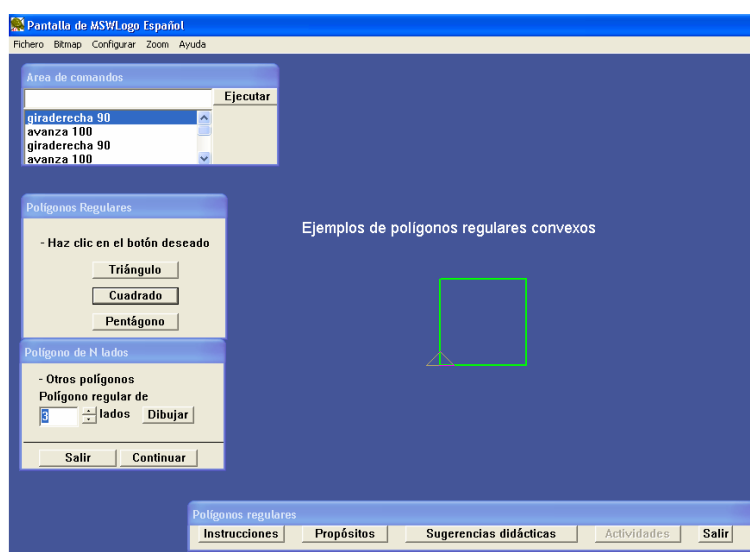


Figure 5. The first scene or sub-activity of the “Regular Polygons” activity

Since this has to be a self-contained, instructional activity, after the opening screen (not shown), the first scene (see Figure 5) gives examples of regular polygons, with buttons for the most basic ones (triangle, square, pentagon), and a space for specifying the number of sides of any other polygon that the users wish to draw. Whenever the turtle draws a polygon through the buttons of this scene, the commands that the turtle is executing (e.g. *forward 100*) are written in the text/command window, so that the students can see how the polygon is being created.

Scene 2: Construction of regular polygons

In the second scene (see Figure 6), the users have to draw a randomly generated polygon –in direct mode using the command buttons or directly typing the commands— over a dotted line; besides the dotted line polygon, only the size of the side is given, but not the angles which the users have to figure out.

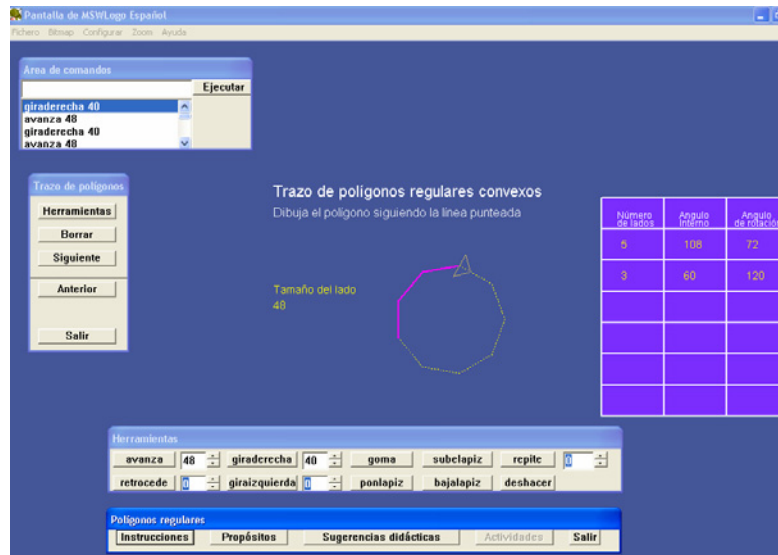


Figure 6. The second scene or sub-activity of the “Regular Polygons” activity

When the polygon is finished, three consecutive pop-up windows appear asking for the number of sides that the polygon had, the internal angle, and the rotation angle; if the inserted values are correct, they are recorded in a table, as shown at the right of Figure 6.

After drawing and analysing at least three polygons in this way, the user can press the “Continue” button. Before going on to the next scene, a “Discussion” window (see Figure 7) is displayed, that asks to reflect on the relationship between the internal angle of the polygon and the rotation angle used. We consider these discussion windows to be an important pedagogical tool that induce classroom debate and reflection on the mathematical relationships.

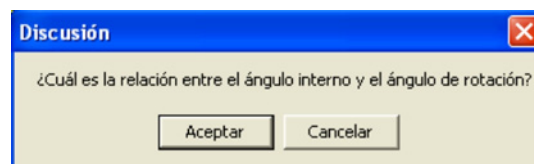


Figure 7. Sample “Discussion” window, asking a question for classroom analysis and debate.

Scene 3: Construction of regular polygons 2

The aim of the third scene (see Figure 8) is to use a full line of Logo instructions to draw a polygon. Users are asked to fill-in the values for the number of sides of the polygon –as the input to the ‘repeat’ instruction— and the rotation angle –the input to the ‘right’ [*giraderecha*] instruction, in the given window with the instruction line: repeat <> [forward 50 right <>]³. Above the space for each input, we have added short labels describing each: e.g. “Number of sides”, “Rotation angle”, so that the meaning of the command inputs is self-explanatory. After the instructions are filled-in, by pressing a “Draw” button the instructions are run and the figure drawn.

If the resulting figure is a correct closed polygon, with the turtle returning to the initial position, then a pop-up window –as is shown in the upper left-corner of Figure 8— asks for the value of the rotation angle, and a follow-up table to the one from the previous scene is filled in (i.e. the values of the number of sides and the rotation angle for each of the polygons investigated so far, are contained in the table so that it can be analysed).

³ In Spanish Logo, the window shows: *repite <> [avanza 50 giraderecha <>]*.

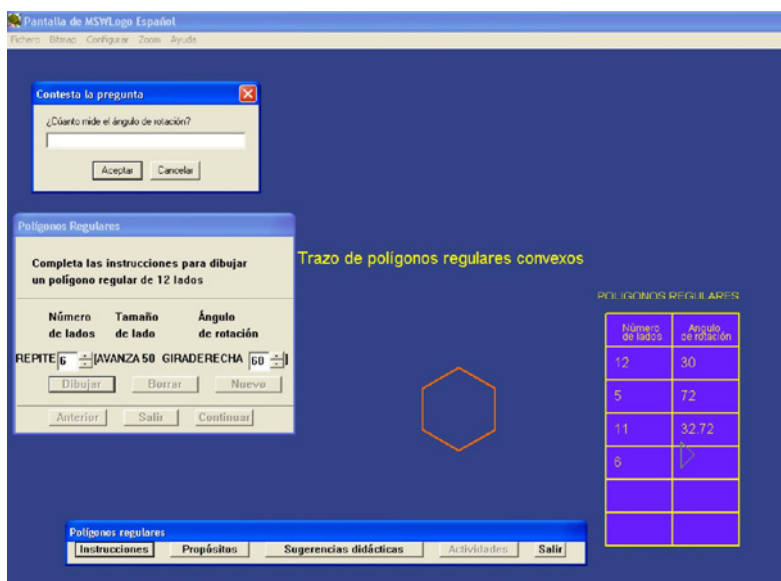


Figure 8. The third scene or sub-activity of the “Regular Polygons” activity

If the figure generated is not a correct closed polygon, then a window asks to observe that fact and to reflect on why that is the case.

When the “Continue” button is finally pressed, another “Discussion” window is displayed, that asks to reflect on the relationship between the number of sides of the polygon and the rotation angle used.

Scene 4: Predict what regular polygon will be produced

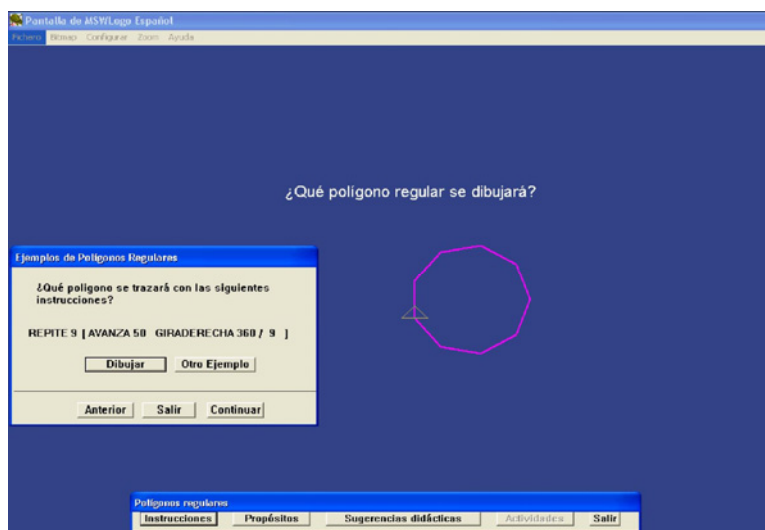


Figure 9. The fourth scene or sub-activity of the “Regular Polygons” activity

In the following scene (see Figure 9), the purpose is to give users explicit examples of the relationship between the number of sides and the rotation angle needed to draw a regular polygon. Thus, filled-in instructions are given to draw a polygon, where the input to the turning angle (the ‘right’ [*giraderecha*] command) is a function of the number of sides (i.e. as a relationship to the full turn of 360). That is, the instructions are given in the form of the following example:

repeat 6 [forward 50 right 360 / 6]

Users are then asked to predict what polygon the turtle will draw. They can verify their prediction by pressing the “Draw” [*Dibujar*] button. They can look at as many examples as they wish.

Scene 5: Draw any polygon

The following scene is very similar to third scene, with a window where users can fill in the inputs to the instruction line: repeat <> [forward <> right <>]. The aim of this scene is to allow users to freely play and draw examples of as many regular polygons as they wish, after hopefully having observed, in the previous scenes, how the number of sides of a regular polygon is related to turning angle. In this way, they can apply that knowledge.

Scene 6: Generalising the construction of regular polygons

The last scene (Figure 10) has as aim to lead users to create a general procedure for a regular polygon. It also introduces how to create a Logo procedure, the editor and, very importantly from an algebraic perspective, the use of variables.

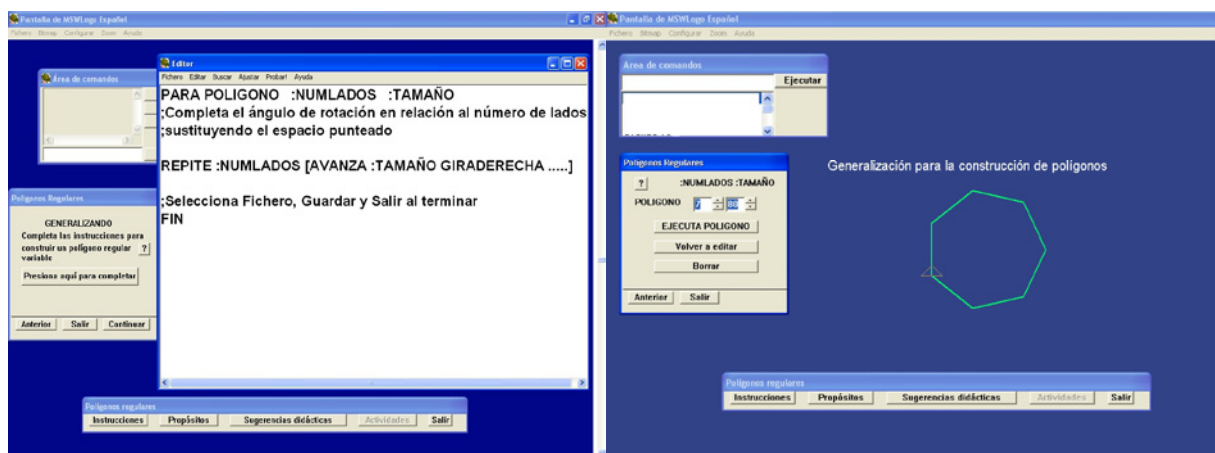


Figure 10. The final scene or sub-activity of the “Regular Polygons” activity. In the screen capture on the left (a) we see the editor open for completing the general procedure of a regular polygon. In the screen capture to the right (b), we see how the Polygon [POLIGONO] procedure can be run.

When the scene first opens, a window with the following text appears: “Generalizing: Complete the instructions to build a variable regular polygon” with a button that says “Press here to complete them”. This button opens the editor (see Figure 10a) where an incomplete general procedure for a regular polygon appears, with comments to help the user:

```
TO POLYGON :NUMSIDES :SIZE

;Complete the instructions by substituting the dotted lines with
the input for the rotation angle in terms of the number of sides

REPEAT :NUMSIDES [FORWARD :SIZE RIGHT ..... ]

;Press Save and Exit when you are finished

END
```

As further assistance, a help button (marked as “?”) explains that variables are written preceded without spaces by a colon (:).

When the editor is closed, a new window appears (see Figure 10b) with the instruction line POLYGON followed by blank spaces for each of its variable inputs, and a button “Run Polygon”. There is also a help “?” button; when this button is pressed, labels with the names of the variables of the inputs of Polygon appear above the blank spaces. In this way, the users can try out their polygon procedure. If needed, there is a button to go back to edit the procedure.

This is the last scene, and the hope is that users will then play by modifying in different ways the procedure, perhaps also creating stars.

Final remarks

With this paper we tried to illustrate how we tried to preserve Logo's philosophy and spirit in a design situation that presents many technical and didactic challenges. In particular, with the "Regular Polygons" activity, we tried to exemplify how we tried to find solutions to those challenges:

- from the challenge of an activity that needs a design as a self-contained "instructional" tool by the teacher (and that has technical limitations such as the typing restrictions);
- to the difficulties of using Logo with no previous introduction to the language or environment;
- to our attempts to provide users (the teacher and his/her students) with opportunities to explore, play and try out instructions; to learn together (in a relationship of "apprenticeship in learning"); and to reflect on mathematical relationships (access to "powerful ideas") and discuss them collaboratively;
- to how we tried to lead users to define a mathematical relationship and show them how this could be done;
- to present users with an opportunity to generalize the mathematical relationship through the use of variables, and finally give them the possibility to complete a "half-baked" general Logo procedure and thus learn to program in Logo and complete a constructive process.

References

- Castro, C. de M. ; Wolff, L. and García, N. (1999) *Mexico's Telesecundaria: Bringing Education by Television to Rural Areas*. In TechKnowLogia: International Journal of Technologies for the Advancement of Knowledge and Learning, 1-1, 29-33.
- Harel, I., & Papert, S., (eds.) (1991). *Constructionism*. Norwood: Ablex.
- Lozano, M. & Sandoval, I. & Trigueros, M. (2006). *Investigating mathematics learning with the use of computer programmes in primary schools*. In Proceedings 30th Conference of the International Group for the Psychology of Mathematics Education, J. Novotná, H. Moraová, M. Krátká & N. Stehlíková (Eds.), Prague: PME. Vol. 4, pp. 89-96.
- Papert, S. (1980). *Mindstorms: Children, Computers, and Powerful Ideas*, New York: Basic Books.
- Papert, S. (1999) *What is Logo? Who Needs It?* In Logo Philosophy and Implementation, Logo Computer Systems Inc. pp. iv-xvi.
- Sacristán, A. I. (1997), *Windows on the Infinite: Constructing Meanings in a Logo-based Microworld*. Ph.D. Dissertation, University of London, Institute of Education, London, England.
- Sacristán, A. I. (2003) *Mathematical Learning with Logo in Mexican Schools*. In Eurologo'2003 Proceedings: Re-inventing technology on education. Coimbra, Portugal: Cnotinfor, Lda., pp. 230-240.
- Sacristán, A.I. and Esparza, E. (2006). *Programación computacional para matemáticas de secundaria: Libro para el alumno*. Mexico: SEP.
- Ursini, S. & Rojano, T. (2000) *Guía para Integrar los Talleres de Capacitación EMAT*. Mexico: SEP-ILCE.

Acknowledgements

The materials presented here belong to Mexican Ministry of Education (SEP), in partnership with the *Instituto Latinoamericano de la Comunicación Educativa* (ILCE).

Euclidean relationships

This study looked into the genesis of the Euclidean conceptions, as defined by Jean Piaget (Piaget & Inhelder, 1948), in his theory of the genetic construction of space. In the Euclidean space, objects are referenced through systems of reference (e.g. length, width, height, etc.) that develop metric ideas in the child (Guzmán, 1998). Thus, the Euclidean notions—which belong to the level of concrete operations—, relate to metric properties of objects (angles, parallelism and distance), and involve *similarities*, the *coordinate system*, *lengths*, and *measure*. The Euclidean relationships, like other spatial relations, are developed through movement and the senses, and are formed at the final stage of the representation of space, after the topological and projective relations. We give a summary of what Piaget and his colleague Inhelder explain for each of them.

Similarity and proportion: For Piaget & Inhelder (1948), a similarity is a transformation that preserves parallel lines and angles. They argue that before the child is able to think of “similar figures”, he/she needs to perceive directly if figures of different sizes have similar relationships. The perceptual recognition that two figures are similar or proportional comes from the idea of transposition. Holloway (1982, p. 85) explains this concept with the example of the square: the transposition of a small square into a bigger one implies considering the square as a whole (the square itself), the size of the angles (that continue to be right angles) and the lengths relative to sides or diagonals (that remain proportionally the same).

The coordinate system: This is a system of reference. Piaget & Inhelder (1948) explain that on a first level, the coordinate system is simply a vast lattice that links all the objects and consists of order relationships that are simultaneously applied to each object in the three dimensions: left-right, up-down and forward-backward. The links in the three directions are made along parallel lines in each dimension, and intersected perpendicularly with those of the other two dimensions. Holloway (1982; p. 95) further explains that the reference point is an Euclidean space relatively independent of the moving objects inside of it.

Length and distance: Piaget and his colleagues were concerned with the perception of conservation of lengths. Inhelder (1975) explains that the problems of conservation of lengths imply problems of change of “order” (i.e. changes in the start and end points) due to judgements such as: “that one is longer because it ends further” or “those have the same length, because they end at the same point” that are very difficult to overcome and show an absence of differentiation in the order of the start and end points.

Measure: Piaget (1975/1950) defines measure as the result of a fusion between the operations of partition and relocation, where partition is a decomposition—e.g. of B into A and A’— (or a recomposition—recomposing B by uniting A and A’—) and relocation is placing one object before, after, over, or underneath another. Thus, for measure, repetition or a repositioning of a unit is necessary over the object to be measured. Also, the conservation of lengths and areas is a necessary condition for every measurement operation. Guzmán (1998) adds that the notions related to measure are constructed through actions on object in space, first through motor-sensory actions that are later internalized.

The use of programming activities to stimulate the development of the Euclidean relationships

The study we present here had as aim to stimulate the development of Euclidean relationships through programming activities using Logo. We considered programming with Logo, for the following reasons:

- Programming a computer means communicating with it, in a language that both the machine and the user can understand (Papert, 1980).

- When using Logo, children have to develop their awareness on how they themselves move in space and be able to communicate that knowledge for writing commands for Logo's turtle to move. Thus Logo should help develop their representations of space.
- We believe in Papert's *constructionist* pedagogical model: that is, in the idea that the construction of knowledge happens more felicitously if the learner is engaged in the construction of something concrete and shareable, that can be perceived by the senses (Papert, 1991).
- With Logo, children can construct, explore and discover. In the words of Segarra & Gayan (1985), Logo is a means for transforming the classroom into a research centre.
- That is why we wanted Logo programming activities, where the students can write, analyse and modify programs and procedures, while simultaneously engaging and developing the Euclidean notions and relationships.

The research was carried out with 26 sixth-grade 11 yr-old primary school children in Mexico. (In previous stages of our research, we carried out trials with two pilot groups with the same characteristics.) In his psychogenetic theory, Piaget (1975) explains that the Euclidean operations are only formed after the projective relations are established, which can only happen when the topological relations are fully developed and operationally grouped. According to this theory, we consider that in students of the sixth grade of primary school (11 yr-olds) the topological (order and partition) and projective (perception) relationships have been established. Thus, these children are at the stage of developing the Euclidean relationships (those related to movement and measure).

The main part of our study was constituted by a didactic sequence of programming activities. This was complemented with pre- and post-tests (i. e. diagnostic questionnaires), that included Piagetian-style tasks, designed to diagnose the development stage of the Euclidean notions; as well as interviews of the children. In addition to that, we provided the children with a preliminary workshop for introducing them to the basic concepts and ideas of programming with Logo.

During the computer activities, the children worked in pairs, with one computer per team. This meant that there were 13 pairs of children during the programming stage.

The diagnostic questionnaire

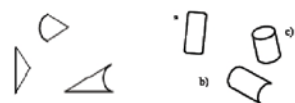


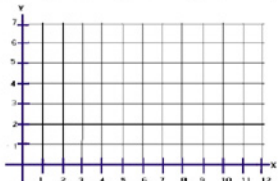

<p>CUESTIONARIO EXPLORATORIO</p> <p>INSTRUCCIONES: Encierra la respuesta correcta a cada pregunta.</p> <p>1. Si se unen las tres piezas de la izquierda ¿qué figura se formará?</p>  <p>Explica tu respuesta: _____</p> <p>2. De los siguientes caminos, ¿hay uno que sea más largo que los otros dos?</p>  <p>Cómo lo supiste: _____</p>	<p>3. ¿Cuántas veces cabe el cuadrado dentro del rectángulo?</p>  <p>Explica tu respuesta: _____</p> <p>4. En el plano cartesiano, localiza los siguientes puntos: A)(6,2) B)(10,4) C)(5,6) D)(2,5) E)(7,3)</p> 	<p>5. Localiza en el plano cartesiano los siguientes tres puntos que forman un triángulo.</p> <p>Para vértice A: (1,7) Para vértice B: (2,4) Para vértice C: (4,6).</p> <p>Escribe el nombre correspondiente a cada vértice.</p>  <p>Traslada el triángulo para que el vértice A esté en la coordenada (6,3), en el nuevo triángulo: El vértice B está en el punto: _____ El vértice C está en el punto: _____</p>
--	--	---

Figure 2. The diagnostic questionnaire

Previous to the programming activities, we asked the children to solve a written questionnaire (see Figure 2) –based on Piagetian tasks— in order to diagnose the state of development of their Euclidean relationships (for further details, see Esparza, 2005). In this way we observed that, previous to the research experiment, the subjects of the study were in fact in the process of developing their Euclidean relationships:

In terms of the notion of *similarity*, most of the children had difficulties in recognising similarities between parts of a figure and the whole, and we felt that they still needed to develop abilities for identifying the invariant characteristics in a figure.

For the notion of *length*, in many cases the children were unable to recognise the permanence of length, in a task where line-segments placed in different positions, and/or with different headings, were presented.

On the other hand, most children adequately used the idea of *measure* in a comparison task between the areas of a large and a small figures.

And in terms of their knowledge of the *coordinate system*, the majority of the subjects were able to locate the five points that the task required.

The workshop for introducing Logo programming

Previous to the didactic sequence, we held a workshop for introducing the children to the basic programming concepts and commands of Logo: The primitive commands we introduced in this workshop were *fd*, *bk*, *rt*, *lt*, *home*, *cs*, *pu*, *pd*, *penup*, as well as *repeat*; we also presented the concept of modular programming (see Figure 3). This workshop was composed of five activities based on ones by Sacristán & Esparza (2006); the activities were presented in an activity booklet.

The children, who worked in teams of two, were very enthusiastic about using computers and they were very creative although they also demanded a lot of attention.

At the beginning, they had some difficulties in learning how to move Logo's turtle, particularly giving the inverse turning commands. They also often skipped necessary commands. But by the end of the workshop, they had overcome these initial problems.

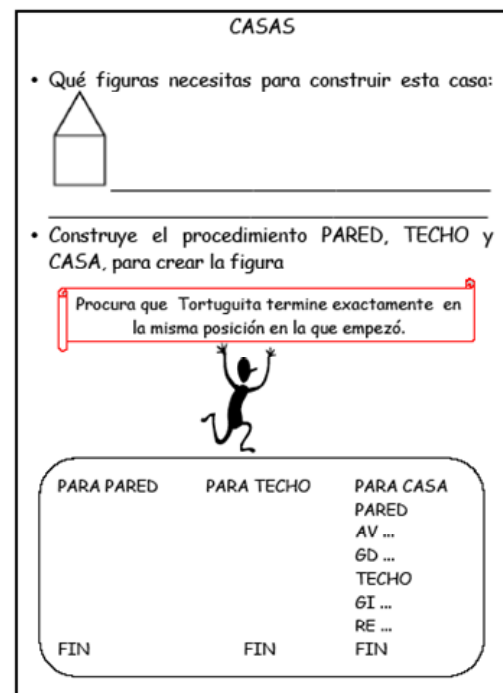


Figure 3. Sample activity (on modularity) from the workshop

In general, with this workshop, the children were able to learn:

- how to use the keyboard
- to understand and anticipate the turtle's movements
- the use of the basic primitive programming commands
- the use of the command *repeat*
- how to use the editor
- how to modify and write Logo procedures
- the use of modularity in writing programs (i. e. to construct programs as sets of procedures)

The sequence of programming activities for developing the Euclidean relationships and the results

The didactic sequence that we developed was integrated by eight programming activities with Logo, where we set up situations –structured through worksheets— that put into play the elements that compose the Euclidean relationships: the coordinate system, similarity, length and measurement. Even though almost all the activities relate to *all* the Euclidean relations, we designed two activities for predominantly putting into play each particular Euclidean relation.

For the coordinate system, we had two activities that had as aim to promote the recognition of the invariance of a figure when its position (or heading, i.e. when the figure is rotated) is changed; that is, to recognise that the paths –movements in the coordinate system— to construct the figure (or main parts of it), remain invariant. The two activities were: “The Labyrinth” and “Stars (*Estrellas*)”. The first is described below. The “Stars” activity (see Figure 1) asked students to draw a star, using a given procedure for a triangle; the purpose of this is for students to be able to recognise that the star is formed by four triangles in different positions, and that, when rotated, the triangles don’t lose their geometric properties.

For the notion of similarity, two activities had as aim to promote the identification of geometric invariants when scaling up or down a figure (i.e. when constructing proportional figures). The two activities were: “The same constructions” and “Letters”. The first is described below. The second had as task to write a procedure to draw a letter “E”, then to modify it to draw a bigger “E” and a smaller “E”.

For the conservation of lengths and distances, the aim of the activities was to promote the recognition that lengths (and distances) can be the same when they begin and end in different places (i.e. when they have different trajectories). The two activities were “Staircases” and “Flags”. The first is described below. The “Flags” activity began with a task where children had to write a couple of procedures: one that drew a flag pointing to the right, and another that drew an “inverse” flag pointing to the left. They were then given a procedure (AXES) that drew coordinate axes with the origin on the Home position of the turtle. Using the previous three procedures, children were asked to write a program to copy a drawing that had a flag in each of the four quadrants of the coordinate plane (with right-pointing flags in the right quadrants; and left-pointing ones, in the left quadrants, but all of them with the bottom of the flag post at the same distance from the centre, even though they *seem* to be placed at different distances). They were then asked to write down the length that the turtle walked from the Home position (the origin) to each flag, and to say and explain which flag was further away.

For the idea of measure, the aim was to promote the idea of iteration and displacement of unit of measure, in the construction of figures. The two activities were “Bricks” and “Frames”. The first is described below. In the “Frames” activities, a drawing, was presented with three nested frames (rectangles), each labelled with the lengths of their sides, and each frame at the same distance from the next one. Children are asked to write a program for this drawing, with the implicit task being that they have to find the distance between one frame and another.

We present below the first four of the eight activities, in the order they were given to the children.

Activity 1: The Labyrinth:

This activity was related mainly to the coordinate system. The specific purpose was to promote the recognition of the permanence of a path –movement in the coordinate system— in a figure when its position (heading) is changed (see Figure 4). In this activity children were presented with a maze (the labyrinth) on paper and on the Logo screen, and they would have to move the turtle to cross it and get out of it on the other side, then write down the sequence of commands they typed. They would then need to get out of a second maze (a rotation of the first) –see Figure 5—, and asked if the same sequence of commands would allow them to cross both mazes and why.

Ten of the 13 pairs of children recognised the permanence of the properties of the labyrinth when its position and direction was changed: that is, they recognised that the first maze was rotated 180° (they all wrote comments like: “it’s the same but its upside down”), and they therefore saw that the same path was valid for both mazes. In fact, for both mazes they used the same sequence of commands:

```
FD 30 RT 90 FD 120 RT 90
FD 30 LT 90 FD 120 LT 90
FD 50 RT 90 FD 65 LT 90
FD 135
```

However, one of the 13 pairs of students gave no indication, neither through their programming, nor in their written work, that they recognised that the object, after changing its position (or in this case being rotated) preserved its geometric characteristics.

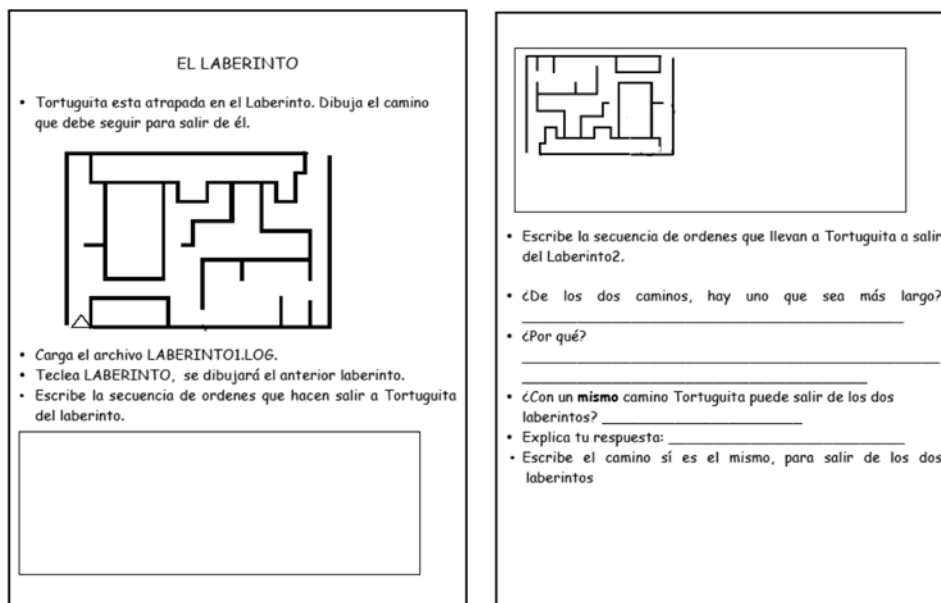


Figure 4. The Labyrinth activity for working with coordinates (and similarity)



Figure 5. The two mazes of the Labyrinth activity. The second maze is simply a rotation of the first. Notice also that the position of the turtle (represented by the small triangle) is the same relative to each maze.

In the second activity for the coordinate system (the “Stars” activity), one more team, than for “The labyrinth” completed the task satisfactorily (i.e. 11 of the 13 teams were successful). In that activity, children also used repetitions and modularity, and were better able to anticipate results.

Activity 2: “The Same Constructions”

This activity was related to the conception of similarity. Its aim was to promote the identification of the invariant elements in a figure, when scaling up (or down) its size. The worksheet (see Figure 6) presented three figures; children were asked to choose one and copy it with a pencil below, then to write a Logo procedure for it. They then were asked which instructions they

LAS MISMAS CONSTRUCCIONES

Escoge una de estas figuras y dibújla con lápiz en el espacio de abajo

- Ahora, escribe un procedimiento Logo que la construya:

Para

fin

- ¿Qué instrucciones necesitas cambiar para que tu figura sea más grande?

- Comprueba tu respuesta en Logo y escribe el procedimiento modificado.

- ¿Qué instrucciones no cambiaron?

- Explica, ¿Porqué algunos comandos si cambian y otros no?

Nine of the 13 student pairs, identified that when scaling up the figure, there was conservation of angles, while the lengths changed. In programming, they changed the inputs to the forward (fd) commands in their procedures (representing the lengths), while the inputs of the rotations –left (lt) and right (rt) commands— remained invariant (see the FIGURE and BIGGERFIGURE procedures below). They were also able to express this recognition in writing.

```

TO BIGGERFIGURE
LT 90 FD 60
RT 90 FD 100
RT 90 FD 300
LT 90 FD 100
LT 90 FD 60
END

```

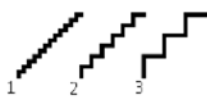
However, in the second activity for similarity (“Letters”), all of the teams successfully completed the task, and in the interviews, children who had been unsuccessful in the first “Same constructions” activity, explained that during the second activity they had realized that the angles remained invariant in scaling transformations.

Activity 3: The Staircases (*Escaleras*):

This activity was related to the conservation of lengths. Its purpose was to promote the recognition of equal lengths in different headings. The task involved the comparison of three staircases (see Figure 7) where children were asked which was the longest, and why, and then asked to write Logo procedures to draw each of them (i.e. one with 12 steps of size 5; another with 6 steps of size 10; and a third one with 3 steps of size 20), and asked what distance Logo's turtle walked. Then they were given a general pre-written procedure (*Escalera*) that had two variables for the number of steps and the width/height of each step, respectively. With this procedure they had to explore given cases of staircases of different number and size of steps but that had the same total length (320).

ESCALERAS




De las siguientes escaleras:



¿Cuál es la escalera más larga?

Explica tu respuesta

- Escribe un procedimiento para cada escalera:
(Toma en cuenta el número de escalones y la medida de cada escalón).

PARA ESCALERA1	 12 escalones de tamaño 5
FIN	
PARA ESCALERA2	 6 escalones de tamaño 10
FIN	
PARA ESCALERA3	 3 escalones de tamaño 20
FIN	

- ¿ En total cuánto avanzó Tortuguita en cada escalera? :
 ESCALERA1: _____
 ESCALERA2: _____
 ESCALERA3: _____
- ¿Cuál es la escalera más larga? _____
- Carga el archivo ESCALERA.LOG.
- Prueba en la ventana de comandos:

Escalera 32 5

Escalera 16 10

Escalera 8 20
- Llena la siguiente tabla:

Tipo de Escalera	# de Escalones	Ancho de cada Escalón	Altura de cada Escalón	Distancia recorrida horizontalmente	Distancia recorrida verticalmente	Distancia Total Recorrida
Escalera 32 5	32	5	5	160	160	320
Escalera 16 10						
Escalera 8 20						
Escalera 16 10				160	160	320
Escalera 8 20						

- ¿Por qué en todas las escaleras Tortuguita recorre la misma distancia? _____
- ¿Todas las escaleras tienen la misma longitud? _____

Figure 7. The Stairs activity for working with lengths

For the first part of the activity, nine of the 13 pairs of students, showed, partially, that they recognised that the three staircases had the same length, independently of the number and size of the steps. Most of the children wrote procedures¹ of the following type for *each* staircase:

```
TO STAIRCASE
REPEAT <number of steps> [FD <size> RT 90 FD <size> LT 90]
END
```

and some of them noticed that the input for the *repeat* multiplied by the input of the forward (*fd*) command, in all three cases gave 60, as could be seen by their written responses to the questions on the worksheets: "How much did the turtle walk in each staircase?" and "Which is the longest staircase?":

- They all measure the same because the distance walked is the same
- They all measure the same because $12 * 5$ is 60 and $6 * 10$ is 60 y $3 * 20$ is 60
- They all are the same distance and in all the distance walked is 60

¹ For the benefit of the reader, we have translated all the Logo procedures and commands in the paper, from the original Spanish into English.

Notice that although they recognise the conservation of length, in their explanations they are observing the distance walked (the length) either horizontally or vertically, not the total one.

And even though the majority of the students were successful in this activity, we are also aware that for each staircase the start and end points of each staircase were placed in the same position relative to one another. This could have had a positive influence on the results.

However, in one particular case, the activity led a pair of students to reach the wrong conclusion: at first they had predicted that the staircases were the same (perhaps because of the relative placement of the start and end points, as discussed in the previous paragraph), but when they programmed them, they used different methods for each (e.g. for the first they did not use *repeat*). they had difficulty finding the actual length and then concluded and wrote that the staircases had different lengths.

In the second activity “Flags” for the Euclidean relation of conservation of lengths and distances, even though all the children were able to write the required programs correctly, four of the teams were not able to make explicit the descriptions contained in their programs and answered the questions incorrectly. Thus only 9 out of 13 teams completed the task successfully; but interestingly, they were not the same children as the successful ones for the “Staircases” activity. Looking together at the results of both activities, we deduce that this relationship is still not fully consolidated.

Activity 4: The Bricks (*Ladrillos*):

This activity was related to measurement. The main purpose was to promote, through the construction of figures, the technique of iteration of a unit of measurement. In this activity, students were given a procedure *LADRILLO* (BRICK) and were asked to use it to create a program for drawing a specified wall (*pared*) –see Figure 8. The procedure *PARED* (WALL) was to be constructed first freely, then children were asked to use the command *repeat* (if they hadn’t already), then asked to create and use a subprocedure *FILA* (ROW) that would create each row (*fila*) of bricks of the wall. Finally, the task was to create a bigger wall (*pared grande*) with the previous procedures.




<p style="text-align: center;">LADRILLOS</p> <p>Edita y prueba el procedimiento LADRILLO (No olvides editar los subprocedimientos: PREPARA y COLOREAR)</p> <div style="border: 1px solid black; padding: 5px; margin: 5px;"> <p>para LADRILLO prepara repite 4 [av 30 gd 90] colorear fin</p> </div> <div style="border: 1px solid black; padding: 5px; margin: 5px;"> <p>para PREPARA poncl 0 poncolorrelleno 4 fin</p> </div> <div style="border: 1px solid black; padding: 5px; margin: 5px;"> <p>para COLOREAR sl gd 45 av 5 relleno re 5 gi 45 bl fin</p> </div> <p>• Utiliza LADRILLO para hacer la siguiente PARED:</p> <div style="border: 1px solid black; padding: 10px; text-align: center;">  </div> <div style="border: 1px solid black; padding: 5px; margin-top: 5px;"> <p>Para PARED</p> </div>	<p>• Si no lo has hecho, edita un procedimiento para PARED usando REPITE.</p> <div style="border: 1px solid black; padding: 10px; text-align: center;"> <p>Para PARED</p> <p>Repite []</p> </div> <p>• Edita un procedimiento para PARED usando subprocedimientos:</p> <div style="border: 1px solid black; padding: 10px; text-align: center;"> <p>Para PARED</p> <p>Repite [FILA]</p> </div> <div style="border: 1px solid black; padding: 10px; text-align: center; margin-top: 10px;"> <p>Para FILA</p> </div>	<p>• Qué tienes que cambiar en PARED para hacer PAREDGRANDE:</p> <div style="border: 1px solid black; padding: 10px; text-align: center;"> <p>PARA PAREDGRANDE</p> <p>REPITE []</p> </div>
---	---	---

Figura 8. The Bricks activity for working with measurement

The other seven pairs of students, built their procedures by putting together the bricks without following a lineal order. This made it difficult for them to draw rows of bricks; and they were unable to synthesise the movement and placement methods that they had used for creating a procedure for a row, or the one for the bigger wall. This disorderly method in the displacements indicates, according to Piaget, the presence of greater intuitive difficulties in the conception of a

continuum formed by the reiteration of one of its own parts, when this part is not previously delimited by a perceptual cut (Piaget, 1975/1950; p. 202).

Only 6 of the 13 pairs of students, followed a lineal technique in the construction of the wall; that is, they repeated the procedure for the brick, by relocating, to build either vertical or horizontal columns or rows. This methodical technique allowed them to be able to synthesise it, using *repeat* into a procedure for the row; then, by putting together four rows of four bricks each, they were able to draw the specified wall. This orderly method also allowed them to easily increase the size of the wall. This was an indication of adequate development of their measurement conceptions: according to Piaget, measure is a result of the fusion between the operations of partition and displacement (Piaget, 1975/1950; p. 200). The procedures they created were like the following:

```
TO BRICK
REPEAT 4 [FD 30 RT 90] COLOR_IT
END
```

```
TO WALL2
REPEAT 4 [BRICK FD 30
  BRICK FD 30 BRICK FD 30
  BRICK FD 30 BK 120 RT 90
  FD 30 LT 90]
END
```

```
TO ROW
REPEAT 4 [BRICK FD 30 ]
END
```

```
TO WALL3
REPEAT 4 [ROW BK 120 RT 90 FD 30 LT 90 ]
END
```

```
TO BIGGERWALL
REPEAT 6 [ROW BK 90]
END
```

Although less than half of the teams were successful in the “Bricks” activity, in the second measurement activity “Frames” almost all (11 of the 13 teams) were successful, working in a much more methodical manner than for the “Bricks” activity: they were able to identify the similarity of the rectangular figures and used repetition and modularity in their programs, most of them constructing a *DISTANCE* subprocedure (*lt 90 fd 25 rt 90*) which indicates their recognition that the three figures were placed at equal distances. Undoubtedly, the experience in programming helped them very much in analysing the task and for developing a methodical strategy and program in the “Frames” activity, the last one of the sequence.

Concluding remarks

Even though the development of Euclidean relationships is part of the genetic development, we found that most students did show some progress in their development after the programming activities. This was shown with the use of a post-activity questionnaire (very similar to the diagnostic questionnaire used as the pre-test), by comparing its results with those of the pre-test, and through interviews. We found that in most cases, although the ways in which students worked implicitly (i.e. how they expressed relationships in their Logo programmes and procedures) wasn’t necessarily explicit for them, there was improvement in their Euclidean relationships: the greatest improvement was in for the notion of *similarity*, as well as for the *integrated* use of all the relations (see Table 1). The Euclidean notions related to the *coordinate system* and *measurement* seemed fairly well established for most students in the pre-test, although for the latter we found difficulties for this relation during the programming activities (particularly in the “Bricks” activity). In terms of the tasks related to *length and distance*, a third of the students had difficulties in the programming and we did not see improvement during the post-test.

But the most important result is that we saw a significant improvement in the integration of all the relations (where the number of success increased from 9 in the pre-test to 20 in the post-test). We attribute this to the fact that during the programming tasks (including those of the introductory workshop) the children were engaging, in an implicit way, all of the Euclidean

relations (in each of the activities). Thus, the programming activities offered students the opportunity to put into play the Euclidean relationships, and in some cases to stimulate them, particularly in the case of the geometric invariants (the Euclidean relation of similarity). Furthermore, we observed, during the programming activities, that the children developed their abilities to anticipate results and to analyse the problems, important skills for the further development of the Euclidean relations.

EUCLIDEAN RELATIONS	Programming activities: Number of teams that successfully completed the task (out of 13)		Pre- and post-test: Number of students that successfully completed the task (out of 26)	
	First Activity	Second activity	Pre-test	Post-test
Coordinate system	Act. 1: The labyrinth 10	Act. 5: Stars 11	22	22
Similarity	Act. 2: Same Constructions 9	Act. 6: Letters 13	10	16
Conservation of lengths and distances	Act. 3: Staircases 9	Act. 7: Flags 9	18	18
Measurement	Act. 4: Bricks 6	Act. 8: Frames 10	21	22
Integrated relationships			9	20

Table 1. Summary of the study results

With this study, we corroborated how enriching the incorporation of Logo programming can be for the teaching and learning of mathematics at the primary school level.

On the other hand, we also confirmed the importance of the role of the teacher: it is his/her tasks to make explicit the links that lead children to become aware of the meaning of the turtle's movements and to relate them to mathematical ideas in contexts other than that of programming.

References

- Esparza Cruz, E. (2005) *Estimulación de las relaciones euclidianas a través de actividades de programación Logo*. M. Sc. thesis, Cinvestav-IPN, Mexico.
- Guzmán, J. and Ramírez, M. F. (1998). *L'utilisation de la calculatrice TI-82 dans la resolution de problèmes verbaux impliquant des systèmes d'équations lineaires à deux inconnues*. In Actes du Colloque GDM-98, Concordia University, Montreal, Canada. Pp. 104-117.
- Holloway, G. E. T. (1982). *La concepción del espacio en el niño según Piaget*. España: Paidos.
- Inhelder, B. (1975). *Aprendizaje y estructuras del conocimiento*. Madrid: Morata
- Papert, S. (1980). *Mindstorms: Children, Computers, and Powerful Ideas*. New York: Basic Books.
- Papert, S. (1991). *The Children's Machine*. New York: Basic Books.
- Piaget, J. (1975 [1950]). *Introducción a la Epistemología Genética*. [Introduction to Genetic Epistemology]. Buenos Aires: Paidos.
- Piaget, J. [& Inhelder, B] (1948). *La representación del espacio en el niño* [The child's conception of space]. Argentina: Paidos.
- Sacristán, A.I. and Esparza, E. (2006). *Programación computacional para matemáticas de secundaria: Libro para el alumno*. Mexico: SEP.
- Segarra, M. y Gayan, J. (1985). *Logo para maestros*. El ordenador en la escuela: propuesta de uso. Barcelona: Gustavo Gili.

Magic Imagine

Károly Farkas, farkas.karoly@nik.bmf.hu

John von Neumann Faculty of Informatics, Budapest Tech

Ildikó Tasnádi, tasnadi_ildiko@hotmail.com

Informatics teacher, Budapest

Abstract

Imagine is a new generation of the Logo-languages, which combines traditional Logo-technique with an object-oriented approach, in line with today's programming trends, providing an ideal environment for the playful method of teaching IT. At the same time, it offers a more sophisticated programming platform for advanced students, as well.

We can begin the playful introduction of the basics of object-oriented programming to children already experienced in the use of Logo-language. It is enough to start with some basic notions, such as: class, entity, parent, descendant. Explanations can be derived from their micro-worlds, using their vocabulary and lots of illustrative examples. Succession, for instance, can be demonstrated by means of family relationships, highlighting similar (inherited) and different (unique) features of the parent and the descendant. This would then be followed by abstraction of all of the above, starting from the specific example to then reach the general concept.

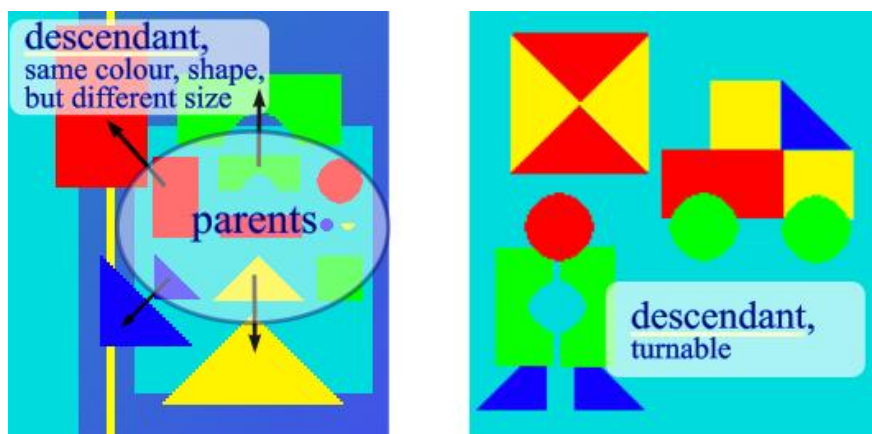


Figure 1. Similarities and differences between the parent and the descendant

Imagine has several **pre-defined primitive classes**, based on which we can easily define our own individual subclasses and instances.

First of all, we have to pre-define our expectations towards our class, in terms of the properties (conditions) and events (procedures) it should possess.

The class determines the behaviour of each turtle of this kind in various situations. Contrary to normal turtles, though it is invisible and it exists in the background only. Only after this can we create the specific, already visible entities.

Keywords

Imagine, object-oriented approach, object-oriented programming, playful IT – teaching

Recent possibilities in Logo: object-oriented programming playfully

Once the abstractions have been adequately prepared, by learning a few commands and expressions we can basically start our object-oriented program. To facilitate the understanding of these abstractions, situations need to be devised, which pupils can identify with.

To illustrate this exercise, we created a program called "BUILDING BLOCKS". In this program there are building blocks on the right side of the screen (these are the parent classes), from which we can choose by clicking on them, creating a copy that can be dragged (the descendant) which can be used for building.



Figure 2. The program

In the learning phase, classes should be generated using only already existing pre-defined classes (e.g. Turtle).

We can define a new class using the `newClass` command in the command-line.

```
? newClass "parent_name "myClass_name [setting-list]
```

Example:

```
? newClass "Turtle "ParentBuildingBlock []
```

This new turtle-class is invisible on the screen, but with the Project Manager (F4) it can be made "real and tangible" for children, as well.

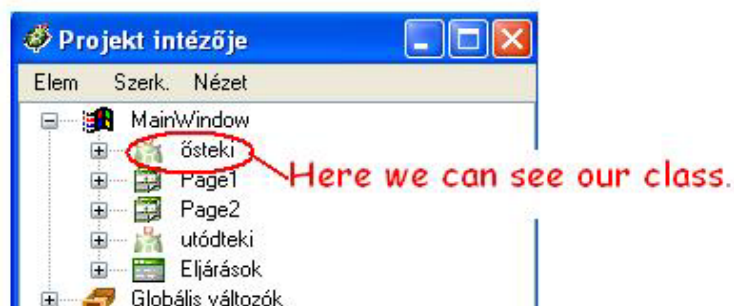


Figure 3. Our class is "real and tangible" for children.

Using the Change Me dialogue we can set and modify different settings in a quick and easy way, thus avoiding the command line.

We would like our new class to have the following properties:

- If we *press the right mouse-button*, the turtle should change its colour (red, blue, green, yellow). This is easiest done using frames in LogoMotion.
- If we *press the left mouse-button* then a new building block with the same colour and shape should be created, in the same position, but of double the size of the basic element (width, length * 2), and it should of course be draggable.

Implementation:

Open the Change Me dialogue of the ParentBuildingBlock.

1. Event → Add → onRightDown: setframe frame + 1

2. Event → Add → onLeftDown:

```
new "cube [pos (pos) shape (shape) frame (frame)
          shapescale (shapescale * 2) .name "block]
ask lastname [startdrag]
```

To ensure selection from among the colours, a command should be entered into the command line:

```
? setframemode "true
```

Prior to the development of the specific individual units, the turtle class called Cube should be defined, specifying the properties of the descendants.

```
? newClass "Turtle "Cube [autodrag true pen pu]
```

Later on the properties of the descendant turtle named Cube can be modified, as well, using the Change Me dialogue. (At this point it should be repeated, that the descendant class may have properties different from those of the parent class, in addition to the inherited properties, since offsprings may have features different from their parents.)

Open the Change Me dialogue of the Cube.

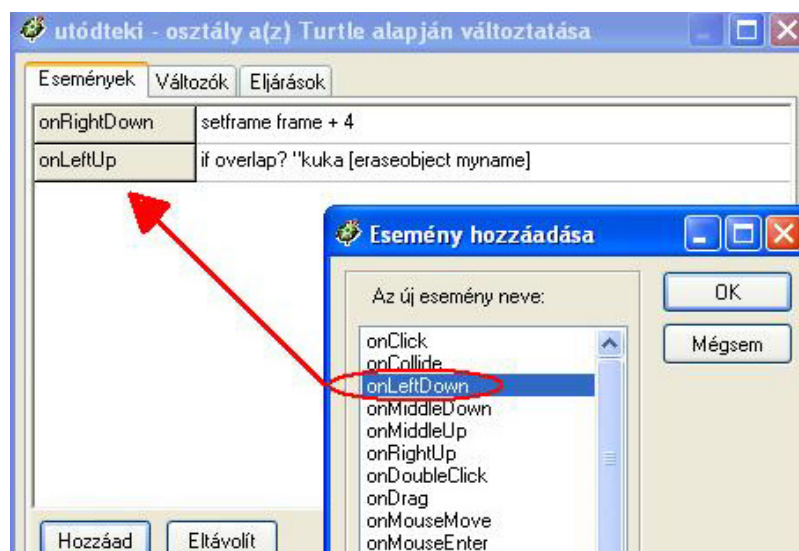


Figure 4. Change Me dialogue

Event → Add → onLeftDown : tofront

Subsequently, the turtles visible on screen, as well, are generated.

```
? new "BuildingBlock
```

A new turtle bearing the pre-defined BuildingBlock features appears on the screen. All there is left to do, is change its shape, to arrive at the first building block element. It can be given a new name, buildingblock1, for instance.

A few more turtles of different shapes are generated in the same way and we have the basic building blocks (e.g.: buildingblock1 – buildingblock10).

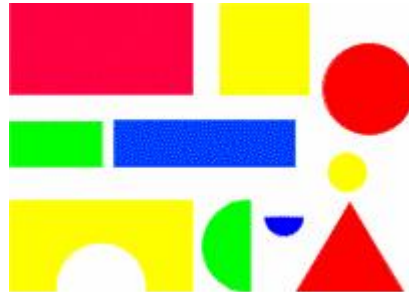


Figure 5. buildingblock1 – buildingblock10

A rectangular area should be defined as a drawing surface and the possibility to delete should be provided.

There are two ways for deletion: elements no longer useful can either be thrown into a waste bin or the entire drawing sheet can be wiped clear pressing a single button. For the one-by-one deletion of elements we need a simple turtle which can wear the shape of the bin and an event (onLeftUp) is added to the properties of the Cube class. Let us assign to it the following:

```
if overlap? "bin [erase_from_list eraseobject myname]
```

Erase_from_list is a procedure which can be written in Cube's Change Me dialogue.

```
to erase_from_list
  make "members butmember myname :members
end
```

In order to be able to erase from a list, we first need to create it. The easiest way to do that is to add it to the list when it is created. Thus extend ParentBuildigBlock's onLeftDown event to include a procedure that performs this operation.

```
new "cube[pos (pos) shape (shape) frame (frame)
  shapyscale(shapyscale * 2) .name "block]
ask lastname [startdrag put_into_list]

to put_into_list
  make "members lput lastname :members
end
```

But, first of all, a command should be entered into the command line:

```
? make "members []
```

The building we have thus created can be deleted in one step, as well, if the following procedure is entered to the onPush event of a button (X):

```
to erase_list
  if not empty? :members
  [
    make "effaceable first :members
    eraseobject :effaceable
    make "members butmember :effaceable :members
    erase_list
  ]
end
```

The result of this command is that all of our turtles bearing the name "blockn (all descendants) will be deleted from the memory and will thus cease to exist.

At the end, create a close button and an event (onPush) is added to the properties of it:

```
(bye "false).
```

Ideas for further development:

Let us modify the program in such a way that the descendant turtles can be rotated (by 30, 45, or 90 degrees) by clicking on them.

Summary

The future belongs to object-orientated programming. The earlier children become acquainted with this approach, the easier it will be for them to use it in their work. We can start teaching it already in primary school when children are open and ready to learn new things and ideas. In our paper we have attempted to illustrate this through a simple example.

References

A. Blaho, I. Kalas, L. Salanci, P. Tomcsanyi (2001) *Imagine Reference Guide*, Logotron

Toward a Culture of Creativity: A Personal Perspective on Logo's Early Years, Legacy, and Ongoing Potential

Wallace Feurzeig, *feurzeig@bbn.com*

Principal Scientist, Learning Systems, BBNT, Cambridge, Massachusetts, USA

Abstract

Logo was born in 1966, forty-one years ago this summer. It was designed very specifically to be a powerfully expressive yet readily accessible language for construction, exploration, and investigation of ideas and processes in math, science, language, and music—to give children a lively learning environment. This was a time of great expectations in the United States. Following the first moon landing, American children everywhere were counting backwards from 10 to zero, closing with a triumphant “Blastoff!” Everything seemed possible. Schools, sometimes among the most change-resistant of institutions, were often open to project-based inquiry, in marked contrast with older, lock-step instruction.

Computers are now nearly ubiquitous in schools throughout the U.S. They are used extensively for word processing and information retrieval. Instructional applications abound, often enhanced by visually rich graphics and animation. Because of the dramatic rate of development and application of computers, one might have predicted that the new learning experiences made possible by programming ideas and activities would be well-established throughout schools by now. But sadly, facilities for student design and invention are severely limited or absent. The use of high-level programming languages in schools, particularly during children's formative years, has almost vanished. Their powerful potential as expressive tools for knowledge construction has yet to be realized.

Education should aim to help children succeed in tasks that they have to work at, that take time, and that require a significant investment of thinking. Such learning experiences also teach children how to make a serious commitment to a task or subject area. Throughout its history, the Logo movement has played a significant role in fostering these goals, and it continues to do so, both within schools and outside. Logo may have gone underground in U.S. schools. But the influence of the ideas and philosophy underlying Logo remains powerful and pervasive, in the world and even in my country. I envisage a process of cultural change that will eventually produce a critical mass of young people who are comfortable and competent with the richness and variety of learning experiences made possible by constructive tools like Logo. These men and women will become, in turn, the creators and teachers of the next wave of learning technology.

Cycloids and limaçons in the turtle graphics

Izabella Foltynowicz, iza@rovib.amu.edu.pl

Dept of Chemistry, A. Mickiewicz University, ul. Grunwaldzka 6, 60-780 Poznań, Poland

Abstract

In my understanding the main purpose of education in science is to help our students with deeper understanding of the main ideas of mathematics and physics and teach them to find the proper tools and methods of discovering the rules, and as a result to encourage them to study on their own to continue the so-called lifelong learning process.

The ready demonstration programs prepared by the teacher or found in the internet are applied to check the way the results depend on the data initially introduced. If the student writes a demonstration program himself/ herself/ they get a chance of deeper understanding of the problem. One approach does not exclude the other or rather they are complementary.

We should study the problem from different points of view; to show different approaches to the same problem. The purpose of the paper is to give a proper tool rather than to give ready answers.

In this paper two points of view are contrasted and compared: the intrinsic (pen-coordinate system) and the extrinsic one (page coordinate system) and also Cartesian and polar plots of the sine family function. Uniform circular motion is a source of a sine wave, circular motion combined with translation gives regular cycloid and a combination of two circular motions produce Lissajous curves and when one circle rolls upon another circle the cycloids are created. The rotational motion is convenient to be realized in intrinsic way, when it is combined with translation or another rotation it is convenient to relate it to the extrinsic coordinate system but it is not necessary. Intrinsic realization of combined motions slows down the process of drawing. Then we can simulate the parallel processes using many turtles.

The following two articles have inspired me to do all this trials: (Armon, 1997) and (Farcas, 2003). Another reason for writing this work is my experience connected with quantum chemistry course when we are trying to imagine what the orbitals looks like. The source of our difficulties is the fact that the students have seen polar plots of the sine family function never before.

I start with the source definition of the sine function. Then I show the simplest way to create the Lissajous figures. Next I demonstrate how the circular motion generates a Cartesian plot and next the polar plot of the sine family functions (rhodonea, limaçons). (It was like discovering the original simplicity of the problem which we have already learnt to see as complicated.) At the next step the superposition of two waves is graphically presented in both coordinate systems. I also present the procedure for generation of a family of cycloidal curves (in a wide meaning) – hypo- and epi-cycloids are the special cases of this general procedure. Then I propose to investigate the Cartesian and polar plots of cycloidally modulated waves and also cycloidally modulated Lissajous figures. This paper presents only a few curves from a huge variety and number of possible ones that can be generated using the procedures proposed. At the end I mention the alternative way of creating polar plots and a beautiful way of creating cycloids and other curves defined by intrinsic curvature proposed by Armon (Armon, 1997).

I intend to show that Logo is convenient and fruitful tool for investigation of the Cartesian plots and polar plots of the curves from the families of limaçons, rhodoneas and cycloids created in the intrinsic style.

Keywords

circular motion, harmonic oscillator model, complex harmonic motion, Lissajous curves (figures), polar coordinates, sine function, cycloid, limaçon

Introduction – the definition of the sine function

The harmonic oscillator is the most powerful and most widely used single model in all of physical science. It is the basis of much of our understanding of the behaviour of molecules and electromagnetic radiation. In the harmonic oscillator model the sine function describes displacement, while the cosine – velocity, as a function of time.

The sine function and every superposition (linear combination) of different sinusoidal waves (Fourier series) are the solutions of the classical wave equation. The sine function is an important component of the solutions of quantum-mechanical wave equation (orbitals). What is the source definition of the sine function? Probably when hearing “the sine function” most people see a Cartesian plot of this function, a sinusoidal wave. But we know, that there is not one and only way to represent this function graphically. The Cartesian plot is one of many possible conventions. The source definition of the sine function is (Weisstein, 2006):

The sine function is the vertical coordinate of the arc endpoint of the unit circle rotating uniformly counter clockwise.

The circular motion is assumed to be at a constant speed (angular speed) or alternatively with a constant frequency measured in revolutions per unit of time. It means that the angle or time is the variable. The sine curve is created when the y coordinate of the constant circular motion becomes the y coordinate of the Cartesian plot, whereas the x coordinate originates from a translation along the x axis at a constant speed related (in time) to the speed of the circular motion. The relation is established by the frequency of the circular motion, i.e. by the number of revolutions made during translation in the x direction over a distance of two diameters of the circle.

We need four turtles with the same initial position and different shapes and pen colours to draw a sine curve. The first turtle (t4) draws a circle and transfers his y coordinate to another turtle (t3), the third one (t2) goes along the x axis (i.e. the propagation direction of the wave) with a proper constant step (hx) and the last one (t1) takes the y coordinate from t4 or t3 and x coordinate from t2 and draws a sine curve as a result. Turtles t2 and t3 are redundant for drawing sine curve: they only transfer coordinates between the turtle performing the circular motion (t4) and the turtle performing the sine curve (t1), but they visualize the whole process very well.

It is convenient to assume that the circle is performed by the statement:

```
repeat 360 [fd :s rt 1]
```

where :s parameter is the arc length step corresponding to Δs and $\Delta \phi$ is constant and equal to 1. Therefore $1/s$ is a measure of the intrinsic curvature of the circle and also the angular speed of the point moving over a circular orbit. Let the sine function be defined as:

$$y = a \sin (n x + p)$$

where a is the amplitude, n is the frequency and p is the phase. If $n = 1$, the distance of two diagonals of the circle must be completed during one revolution of a circle, for $n = 2$ – during two revolutions of a circle etc. The amplitude a is the radius of the circle. Let one step in translation in x direction be hx. Phase p is the initial position of the turtle drawing a circle (circular motion at a constant speed) measured in degrees. Phase 90 means that $y = a \cos (n x)$.

We can choose a, n, and p as an input for the procedure and calculate s and hx or we can start with s, n and p and calculate a and hx. The sine function is described by three independent variables. The procedure sinecurve1 creates a sine curve with four turtles, procedure sinecurve2 – with two turtles. What will happen if we change rt into lt in these two procedures?

```
to sinecurve1 :a :n :p
; y = a sin (n x + p)
make "s :a * 3.14 / 180 make "hx 2 * :s / :n / 3.14
```

```

;initial positions for turtles
cs ask [t1 t2 t3 t4][pu setpos [-300 0] pd] ask "t4 [repeat :p [fd :s rt 1]]
ask "t1 [pu setycor ask "t4 [ycor] pd]

repeat 360 * :n [ask [t4] [fd :s rt 1] ask [t1 t2] [setxcor xcor + :hx]
ask [t1 t3] [setycor ask "t4 [ycor]] wait 10]

end

? sinecurve1 100 2 90
? ask [t1 t2 t3 t4][stamp]

```

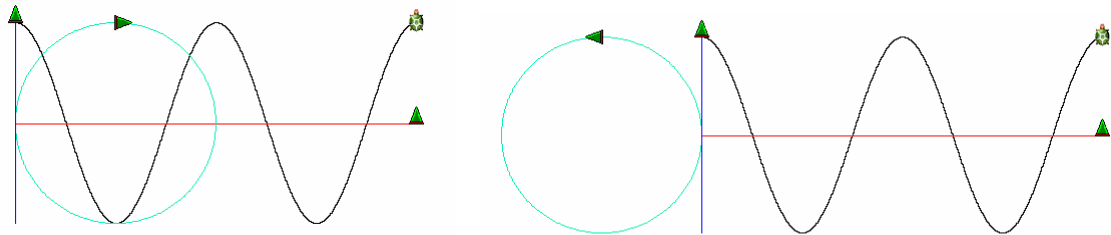


Fig.1 The cosine function with *rt* (left) and *lt* (right) created by procedure *sinecurve1*

In the procedure *sinecurve2* the turtles *t2* and *t3* are unemployed and the bolded part of the procedure *sinecurve1* is replaced by:

```

repeat 360 * :n [ask [t4] [fd :s rt 1] ask [t1 ] [setxcor xcor + :hx setycor ask "t4 [ycor]]
wait 10]

? sinecurve2 100 2 90
? ask [t1 t4][stamp]

```

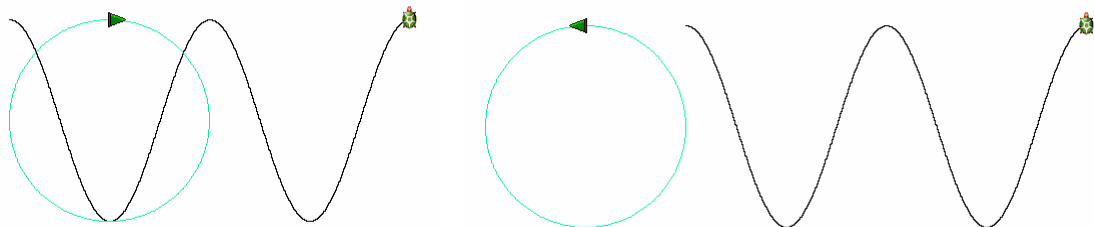


Fig.2 The cosine function with *rt* (left) and *lt* (right) created by procedure *sinecurve2*

Lissajous curves in the turtle graphics.

This family of curves was investigated by Nathaniel Bowditch in 1815, and later in more detail by Jules Antoine Lissajous in 1857.

Jules Antoine Lissajous (1822-1880) was a French mathematician and physicist. He was interested in waves and developed an optical method for studying vibrations. Lissajous curves have applications in physics, astronomy, and other sciences.

Some of these figures, and geometrical methods of constructing them, have been discovered independently, e.g. the Lemniscate of Gerono (eight curve) (Xah Lee, 2004).

The Lissajous figures are created by the system of parametric equations which describes two perpendicular harmonic vibrations:

$$\begin{cases} x = r_1 \sin(n_1 t + p) \\ y = r_2 \sin(n_2 t) \end{cases}$$

Do we need the sinusoid to generate the Lissajous curves, or could only two circular motions create these figures? If the latter was possible, it would be the simplest way to create the Lissajous figures.

The time variable t changes from the moment when the process starts to the moment when the process ends (after n_1 revolutions along first circle and n_2 revolutions along the second circle, circular motions are simultaneous). The turtle drawing the Lissajous curve takes the x coordinate from one turtle moving along the green circle and y coordinate from the turtle drawing the second, blue, circle. We can use three turtles or five turtles. In the last case these additional two turtles transfer the coordinates between those drawing the circles and the one drawing the Lissajous figure. We can also use 3 turtles and make two of them invisible: in that case we will see only one turtle drawing the Lissajous curve. The parameters of the procedure (data) could be the steps for circular motion (Δs_1 and Δs_2) or the radii (amplitudes r_1 and r_2), phase (p) and frequencies (n_1 and n_2). After some trials we will see that the shape of the L figures depends on the ratio of the frequencies of the two perpendicular waves. Dissimilar shapes can be selected by dividing the number of repetitions by the greatest common divisor of the two frequencies or by choosing only these values of frequencies whose greatest common divisor is 1.

The procedures are given for rt case only. The turtles must be created with proper attributes at the beginning.

to turtles

```
new "Turtle [penColour "green4] ask "t2 [setshape loadImage "|Triangle Turtle.lgf]
new "Turtle [penColour "blue] ask "t3 [setshape loadImage "|Triangle Turtle.lgf]
new "Turtle [penColour "green4] ask "t4 [setshape loadImage "|Triangle Turtle.lgf]
new "Turtle [penColour "paleRed penwidth 2] ask "t5 [setshape loadImage "|Turtle.lgf]
ask "t1 [setshape loadImage "|Triangle Turtle.lgf] setpencolour "blue]
end
```

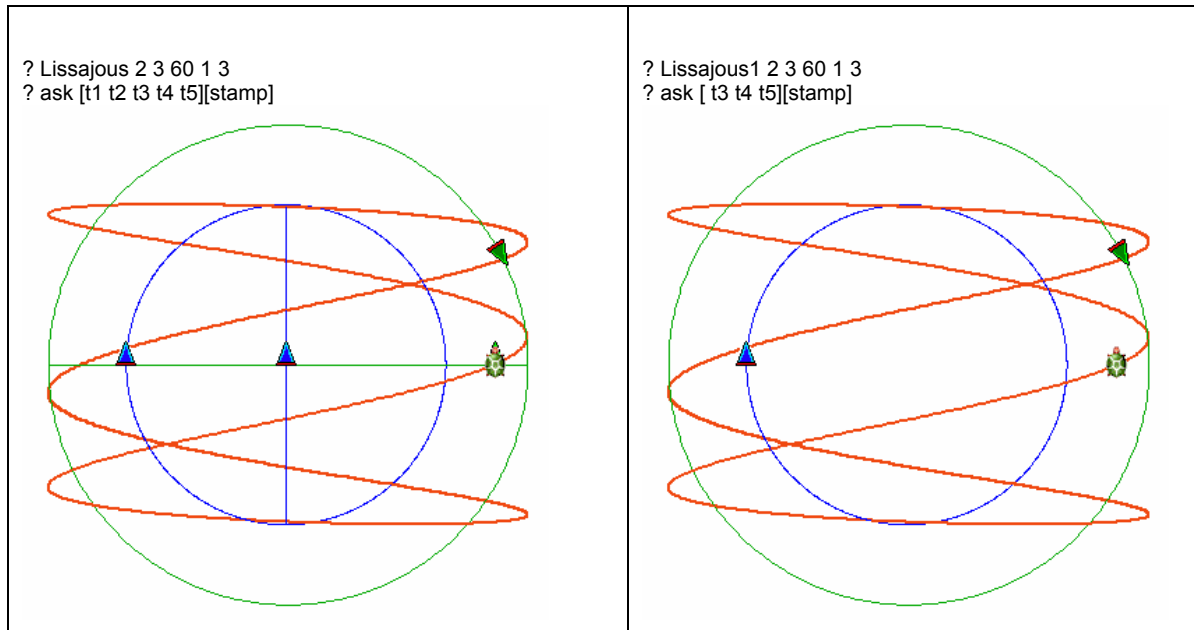
to GCD :a :b

```
while [:b <> 0] [let "r mod :a :b let "a :b let "b :r]
op :a
end
```

to hop :dx :dy

```
pu rt 90 fd :dx lt 90 fd :dy pd
end
```

<pre>to Lissajous :s1 :s2 :p :n1 :n2 cs ;amplitudes make "x 180 / 3.14 make "r1 :s1 * :x make "r2 :s2 * :x ;initial position for turtles ask "t3 [hop -:r1 0] ask "t4 [hop 0 :r2 rt 90 pu repeat :p [fd :s2 rt 1] pd] ask "t5 [pu setxcor ask "t4 [xcor] setycor ask "t3 [ycor] pd] repeat 360 * :n1 * :n2 / GCD :n1 :n2 [ask [t3] [fd :s1 / :n2 rt 1 / :n2] ask [t1 t5] [setycor ask "t3 [ycor]] ask [t4] [fd :s2 / :n1 rt 1 / :n1] ask [t2 t5] [setxcor ask "t4 [xcor]] wait 10] end</pre>	<pre>to Lissajous1 :s1 :s2 :p :n1 :n2 cs ;without t1 and t2 ;amplitudes make "x 180 / 3.14 make "r1 :s1 * :x make "r2 :s2 * :x ;initial position for turtles ask "t3 [hop -:r1 0] ask "t4 [hop 0 :r2 rt 90 pu repeat :p [fd :s2 rt 1] pd] ask "t5 [pu setxcor ask "t4 [xcor] setycor ask "t3 [ycor] pd] repeat 360 * :n1 * :n2 / GCD :n1 :n2 [ask [t3] [fd :s1 / :n2 rt 1 / :n2] ask [t4] [fd :s2 / :n1 rt 1 / :n1] ask [t5] [setxcor ask "t4 [xcor] setycor ask "t3 [ycor]] wait 10] end</pre>
--	--



In the following demo the statement “/ GCD :n1 :n2” in the Lissajous procedure may be removed.

to demoLissajous

```
for "s1 [2 3] [for "n1 [1 5] [for "n2 (se :n1 5) [for "p [0 360 15] [if GCD :n1 :n2 = 1
[(print [phase=] :p) (print [amplitude1=] :s1) (print [amplitude2=] 3)
(print [frequency1=] :n1) (print [frequency2=] :n2)
Lissajous :s1 3 :p :n1 :n2] wait 100] wait 1000]]]
end
```

The bolded part of the Lissajous procedure could have another different shape – it gives the same result but works quicker and less accurate.

```
repeat 360 [ask [t3] [dot fd :s1 * :n1 rt :n1] ask [t4] [dot fd :s2 * :n2 rt :n2]
ask [t5] [setxcor ask "t4 [xcor] setycor ask "t3 [ycor] dot]]
```

Polar coordinates and polar plot of the sine functions

What determines the choice of the coordinate system in a given situation? Why the Cartesian coordinates are not used to identify the position of our town on the globe? For the same reason we do not use the Cartesian coordinates to determine the position of an electron with respect to the atomic nucleus. The reason is symmetry. When the coordinate system suits the symmetry of the object under consideration, the equations describing it become simpler, and the calculations easier. For example: the equation of a circle of radius r centred at the pole is $r^2 = x^2 + y^2$ in the Cartesian coordinates, while in the polar coordinates it is: $r = \text{const}$. I suspect that in everyday life we use polar coordinates and their three dimensional counterpart, i.e. spherical coordinates (in geography), more frequently than the Cartesian coordinates. However, we usually do not know how simple mathematical functions look in the polar coordinates. What benefit could come from drawing them in polar coordinates? Could we understand better for example the shapes of the orbitals (one electron wave functions which are the solutions of time independent Schrödinger equation) in chemistry? Let us investigate only one family of sine functions, namely $r = \sin(n\phi)$. Is there anything the students/pupils can discover in this area? I ask my students to describe in a laboratory report their findings they found particularly interesting. One of the students answered: “Honestly speaking, every subsequent function was a huge discovery, great news and a great riddle for me.” His answer encouraged me to share my didactic experience with you.

For the definition of the polar coordinates see for example (Weisstein, 2002).

R is the radial distance from the origin (initial position of the turtle), φ is defined as the counter clockwise angle from the x axis, but in Logo it will be convenient to choose φ as the clockwise angle from the N-S axis (by the way: it is a good exercise to see what is the difference between the different possibilities of choosing the reference axis and the direction of counting the angle).

We offer a job to the fifth turtle (besides four introduced in the Introduction) who for subsequent values of angle φ (from 0 to 360 with the step 1) draws a line (vector) of length equal to the value of the sine function for this particular value of variable, namely equal to $\sin \varphi$. The value of $\sin \varphi$ is equal to the value of y coordinate of the turtle rounding along the circle (and turtle oscillating along the y axis). Besides of the usual "mathematical convention" we can try also the frequently used convention of drawing the radii (vectors r) corresponding to the negative values of the function as positive and in different colour. Let us call it the "two-colour convention" (unfortunately I use two colours in every convention).

to turtles

```
repeat 4 [new "Turtle []]
ask [t2 t3 t4] [setshape loadImage "|Triangle Turtle.lgf]
ask "t1 [setPW 2]
ask "t5 [setPW 2 setHomestate [[200 0]0] setshape loadImage "|Turtle.lgf]
end
```

Below the procedure `sinepolarplot` which is a good tool to demonstrate the shapes of the sine family functions in the polar coordinates ("mathematical convention").

```
to sinepolarplot :a :n :p
; y = a sin (n x + p)
let"s :a * 3.14 / 180 let "hx 2 * :s / :n / 3.14
; initial positions for turtles
cs ask [t1 t2 t3 t4][pu setpos [-300 0]pd] ask "t4 [repeat :p [fd :s rt 1]]
ask "t1 [pu setycor ask "t4 [ycor] pd] ask [t1 t5][setPW 2 pu]

repeat 360 * :n [let "y ask "t4 [ycor] ask [t4] [fd :s rt 1] ask [t1 t2] [setxcor xcor + :hx]
ask [t1 t3] [setycor :y] ask [t1] [ifelse :y < 0 [setPC 4] [setPC 2] dot]
ask [t5] [ifelse :y < 0 [setPC 4] [setPC 2] fd :y dot bk :y rt 1 / :n] wait 50]

end
```

? sinepolarplot 90 2 0

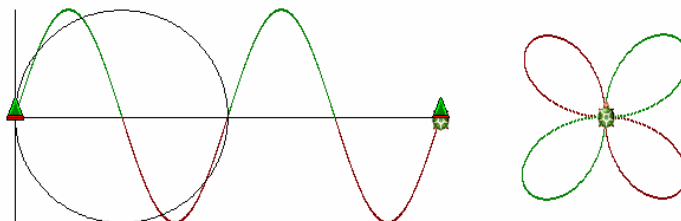


Fig. 3. The Cartesian plot of the function $y = \sin 2x$ and the polar plot of the function $r = \sin 2\theta$

On the Fig. 3 is one example of rhodonea (Wassenaar, 2005). As before, we can remove the redundant turtles `t2` and `t3`, as well as hide the motion of turtles `t4` and `t1` if we would like to see the polar plot only. To obtain the functions from the limaçon (Weistein, 2003) family we must only add the constant value to the `ycor` in the procedure `sinepolarplot` (Fig. 4 is the example).

The example result for fractional n and enlarged number of repetitions is shown on the Fig.5.

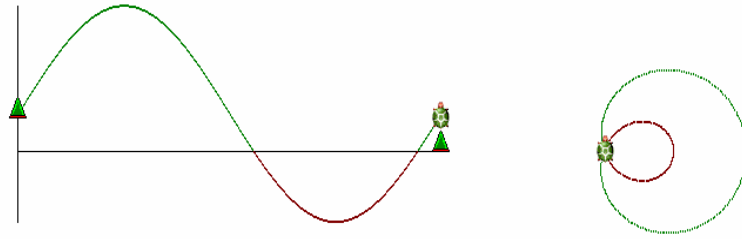


Fig. 4 The Cartesian plot of the function $y = 1/3 + \sin x$ and the polar plot of the function $r = 1/3 + \sin fi$

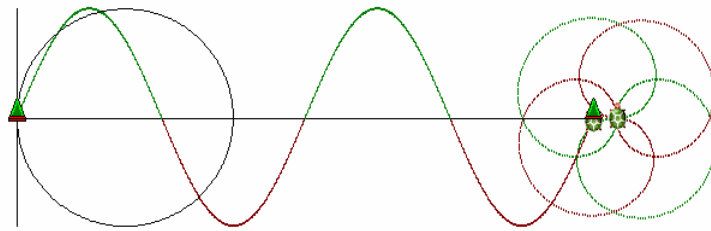


Fig. 5 The Cartesian plot of the function $y = \sin 2/3x$ and the polar plot of the function $r = \sin 2/3fi$

Program startij demonstrates the limaçon (Wassenaar, 2004) family functions $r = a + \sin(i/j * \phi)$ and shows how the shapes depend on the value of a for a given small integer values of i and j . The positive values are in red, negative in blue, the circle of radius a with black.

```
to sinpij :a :i :j
  cs repeat 360 * :i * :j / GCD :i :j [let "y 100 * (sin :i / :j * (repc - 1)) fd :a + :y
    ifelse :y < 0 [ setpc 1] [setpc 12] setpw 3 dot bk :y setpc 0 setpw 2 dot bk :a rt 1]
end

to startij
  pu ht
  for "i [1 5][ for "j [1 5][if GCD :i :j = 1 [(pr :i :j) for "a [100 0 -5] [sinpij :a :i :j wait 10] wait 1000]
    for "a [0 -100 -5] [sinpij :a :i :j wait 10] wait 1000]]
end
```

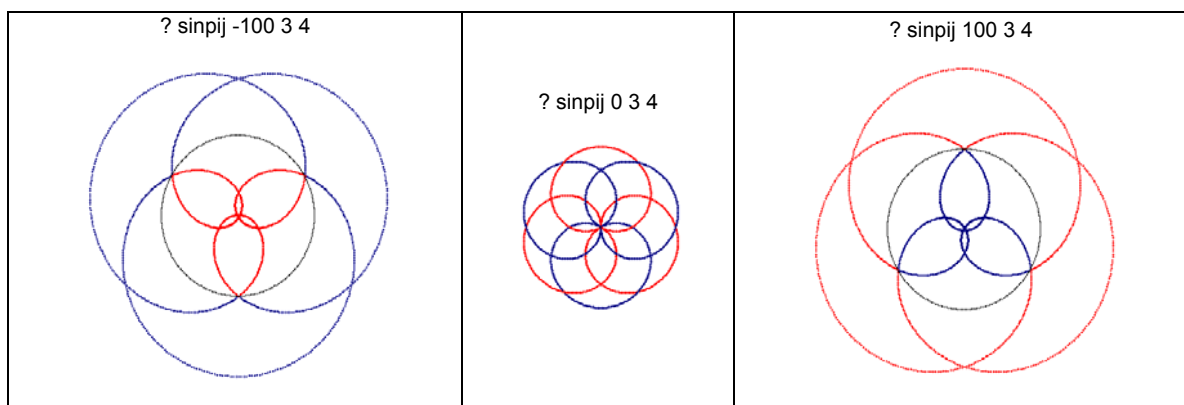


Fig. 6. The polar plots of the functions $r = -1 + \sin 3/4 fi$, $r = \sin 3/4 fi$ and $r = 1 + \sin 3/4 fi$.

Superposition of waves

Now we are ready to demonstrate the Cartesian and polar superposition (interference, beat) of any two waves, each of them characterized by three parameters: amplitude, frequency and phase. We could offer a job for 12 turtles: t4 and t6 are making circular motions, t3 and t9 demonstrates one-dimensional (y direction) harmonic motions, t2 and t8 are making translation in x direction, t1 and t7 create Cartesian plots, t5 and t10 – polar plots and t11 and t12 – Cartesian and polar sums of two given waves. For clarity we can make some of the plots invisible as well as we can hide some of the turtles to show step by step how the superposition of two waves is created and to compare clearly the components and the sum in the Cartesian and polar coordinates. We can also compare the polar plots in the “mathematic” and the “two-colour” conventions.

It is a certain inconsistency to avoid, but only in the Cartesian coordinates, not in the polar ones, when the amplitudes are not the same: one could have the impression that the wave of a small amplitude propagates with a lower velocity but it is a wrong conclusion (see Fig 10, the Cartesian components and the polar sum). We can easily improve the procedure below by scaling hx value by the values of two amplitudes; hx1 ought to be of the same value as hx2.

```
? sinepolarsumdot 90 45 3 6 0 90
```

```
? ask all [stamp]
```

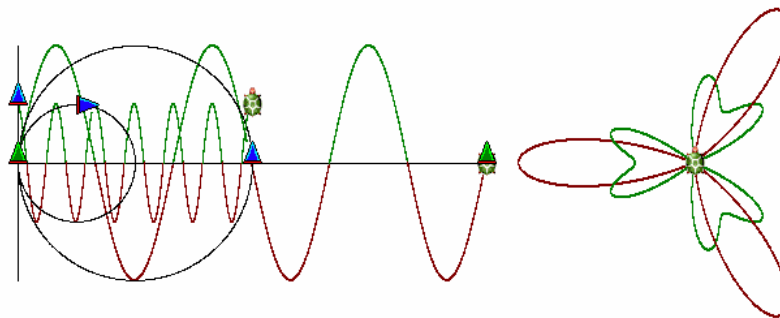


Fig. 7. The Cartesian components and the polar sum of two waves of different amplitudes

Below is one of many possible procedures, sinepolarsumdot:

```
to sinepolarsumdot3 :a1 :a2 :n1 :n2 :p1 :p2
; y1 = a1 sin (n1 x + p1), y2 = a2 sin (n2 x + p2)
let"s1 :a1 * 3.14 / 180 let"s2 :a2 * 3.14 / 180
let "hx1 2 * :s1 / (:n1 * :n2 * 3.14)
let "hx2 2 * :s2 / (:n2 * :n1 * 3.14)
ifelse :hx1 > :hx2 [let "hx :hx1][let "hx :hx2]
; initial positions for turtles
cs ask [t1 t2 t3 t4 t6 t7 t8 t9 t11][pu setpos [-300 0]pd] ask [t11 t12][st pu]
ask [t4 t6 t8 t9][ht pu] ask [t3 t2][ht] ask "t4 [repeat :p1 * :n2 [fd :s1 / :n2 rt 1 / :n2]]
ask "t6 [repeat :p2 * :n1 [fd :s2 / :n1 rt 1 / :n1]] ask "t1 [pu setycor ask "t4 [ycor] pd]
ask "t7 [pu setycor ask "t6 [ycor] pd] ask [t1 t5 t7 t10][ht pu]

repeat 360 * :n1 * :n2 [let "y1 ask "t4 [ycor] let "y2 ask "t6 [ycor] let "y :y1 + :y2
ask [t4] [fd :s1 / :n2 rt 1 / :n2] ask [t6] [fd :s2 / :n1 rt 1 / :n1]
ask [t1 t2] [setxcor xcor + :hx1] ask [t1 t3] [setycor :y1]
ask [t7 t8] [setxcor xcor + :hx2] ask [t7 t9] [setycor :y2]
ask [t11][setxcor xcor + :hx setycor :y] ask [t1] [ifelse :y1 < 0 [setPC 4] [setPC 2] dot]
ask [t7] [ifelse :y2 < 0 [setPC 12] [setPC 1] dot]
ask [t5] [ifelse :y1 < 0 [setPC 4] [setPC 2] fd :y1 dot bk :y1 rt 1 / :n1 / :n2]
```

```

ask [t10] [ifelse :y2 < 0 [setPC 12] [setPC 1] fd :y2 dot bk :y2 rt 1 / :n1 / :n2]
ask [t11] [ifelse :y < 0 [setPC 13] [setPC 11] dot]
; mathematical convention
ask [t12] [ifelse :y < 0 [setPC 13] [setPC 11] fd :y dot bk :y rt 1 / :n1 / :n2 ]
; two-colour convention
;
ask [t12] [ifelse :y < 0 [setPC 13 bk :y dot fd :y] [setPC 11 fd :y dot bk :y]
;
rt 1 / :n1 / :n2 ]
]
end

```

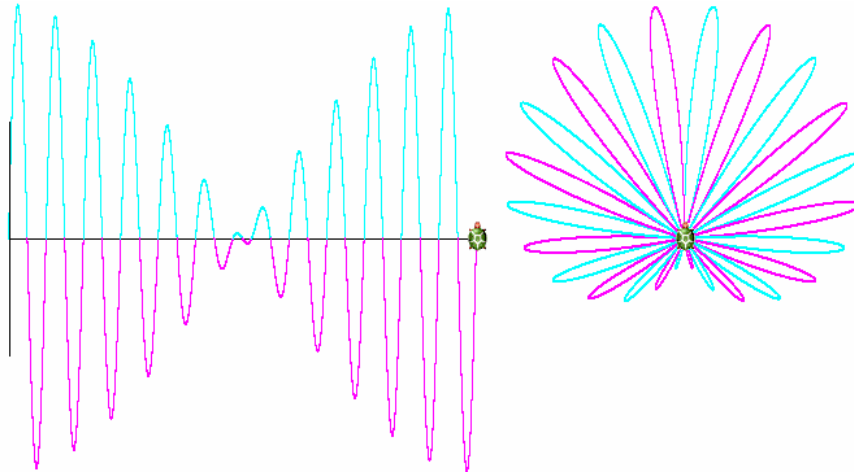


Fig. 8 The Cartesian and the polar (two-colour convention) plots of the superposition of two waves with parameters: 90 90 12 13 0 0 (beat)

Cycloids

A cycloid (Weisstein, 2003, 2004) is a plane curve that is the trajectory of a point lying on a circle that rolls without slipping along a straight line (regular cycloid) or upon another circle. A cycloid is called an epicycloids or hypocycloid, depending on whether the rolling circle has external or internal contact with the fixed circle. When the ratio of the radii of these two circles is rational, then the cycloidal curve is a closed algebraic curve.

In procedure `cycx` one turtle draws a regular cycloid combining two motions: circular motion (intrinsic equation of a circle, green colour) and translation in x direction (the extrinsic Cartesian coordinate system, blue colour).

```

to cycx :s
cs pu setxy -100 0 setheading 270
repeat 360 [dot fd :s rt 1 setxcor xcor + :s wait 10]
;circuference is 360 * s, radius is 180 * s / phi
end

```

The procedure `cyc` draws a regular cycloid on a wholly intrinsic way and may be simplified to the procedure `cyc1`:

```

to cyc
cs setPW 1 st setxy -100 0 setheading 270
repeat 360 [pd st setPC 12 fd 1 rt 1 wait 20
pu ht repeat repc [lt 1 bk 1]
pd st setPC 0 bk 1 wait 20
pu ht repeat repc [fd 1 rt 1]]
end

```

```

to cyc1 :s
cs pu setPW 1 ht setxy -100 0 setheading 270
repeat 360 [dot fd :s rt 1 repeat repc [lt 1] bk
:s repeat repc [rt 1]]
end

```

The main procedure for hypo- and –epicycloids could be as follows:

```
to hecyys :f :i :s
; i=-1 epi, i=1 hypo, f = 1, 2, 3, 4, ...
; dfl of fixed circle=const; s (scale) (from 1 to 2)
cs ht
repeat 360 [pd setPC 12 fd :s rt :i * :f
  pu repeat repc [lt :i * :f bk :s]
  pd setPC 0 lt 1 bk :s
  pu repeat repc [fd :s rt :i * :f]]
end
```

where parameter $f = n_1/n_2$ (ratio of frequencies).

Procedure hecyys could have two interesting modifications: hecyysrt in which the direction of the rolling circle is the same as the fixed one and hecyysm in which the centre of the rolling circle moves along the circumference of the fixed one.

```
to hecyysrt :f :i :s
cs ht
repeat 360 [pd setPC 12 fd :s rt :i * :f
  pu repeat repc [lt :i * :f bk :s]
  pd setPC 0 fd :s rt 1
  pu repeat repc [fd :s rt :i * :f]]
end
```

```
to hecyysm :f :i :s
cs ht
repeat 360 [pd setPC 12 lt 90 fd :s rt :i * :f
  pu repeat repc [lt :i * :f bk :s]
  pd setPC 0 rt 90 lt 1 bk :s lt 90
  pu repeat repc [fd :s rt :i * :f] rt 90]
end
```

We can create the whole family of cycloidal curves in quite a different way. In the procedures hypoepicyc and hypoepicyc1 the parameters s_1 and s_2 are the steps of arc lengths (these values are directly proportional to the radiuses) of the rolling and fixed circles, parameters n_1 and n_2 are the frequencies of these two circular motions. In each of these procedures for the same n_1 and n_2 , the ratio of frequencies is the same, but is obtained in a different way and in consequence the first procedure draws slower and more accurate than the second one.

```
to hypoepicyc :s1 :s2 :n1 :n2 :i
; hypo i = 1, epi i = -1
cs pu
repeat 360 * :n1 * :n2
[setPC 12 setPW 2 dot fd :s1 / :n2 rt :i / :n2
repeat repc [lt :i / :n2 bk :s1 / :n2]
setPC 0 setPW 1 dot lt 1 / :n1 bk :s2 / :n1
repeat repc [fd :s1 / :n2 rt :i / :n2]]
end
```

```
to hypoepicyc1 :s1 :s2 :n1 :n2 :i
; faster but with low quality
cs pu setPW 2
; additional with phase difference :p
; repeat :p / :n1 [fd :s1 * :n1 rt :i * :n1]
repeat 360 [setPC 12 dot fd :s1 * :n1 rt :i * :n1
repeat repc [lt :i * :n1 bk :s1 * :n1]
setPC 0 dot lt :n2 bk :s2 * :n2
repeat repc [fd :s1 * :n1 rt :i * :n1]]
end
```

If in the procedures hypoepicyc and hypoepicyc1 **lt :n2** change to **rt :n2** hypocycloid change into epicycloid with the same parameters and vice versa.

The “natural” cycloids created by the procedure hecyys could be obtained by generalized procedure hypoepicyc for the data fulfilled the basic cycloidal rule:

$$\frac{s_1}{s_2} = \frac{r_1}{r_2} = \frac{n_2}{n_1} = \frac{1}{f}$$

For example, the results of hecyys 3 1 1, hecyys 3 1 1.5, hypoepicyc 1 3 3 1 1 and hypoepicyc .5 1.5 3 3 1 1 differ only with the size.

Below is the version of the hypoepicyc procedure realized with three turtles: t1 draws the cycloid, t3 draws the fixed circle, t3 is invisible and cooperates with t1 and t2 in transferring the Cartesian coordinates and the turtle heading.

```
to turtles
repeat 2 [new "Turtle []]
ask [t1 t2 t3] [setHomestate [[-100 0]0]]
end

to hypoepicyc2 :s1 :s2 :n1 :n2 :i
cs ask [t1 t2 t3][pu] ask "t2 [ht]
ask "t1 [ pd setPW 2 setPC 12] ask "t3 [setPC 0 setPW 1]

repeat 360 * :n1 * :n2
[ask "t2 [setpos ask "t1 [pos] setheading ask "t1 [heading] repeat repc - 1 [lt :i / :n2 bk :s1 / :n2]]
ask "t3 [setpos ask "t2 [pos] pd lt 1 / :n1 bk :s2 / :n1 pu]
ask "t2 [setpos ask "t3 [pos] setheading ask "t3 [heading] repeat repc - 1 [fd :s1 / :n2 rt :i / :n2]]
ask "t1 [setpos ask "t2 [pos] setheading ask "t2 [heading] fd :s1 / :n2 rt :i / :n2]]
end
```

Let us see only one the most interesting modification: the fixed circle is additionally modified by the cycloid. Compare for example hypoepicyc2 1 3 3 1 1 with hypoepicyc3 1 2 2 1 1 and hypoepicyc3 2 1 1 2 1.

```
to hypoepicyc3 :s1 :s2 :n1 :n2 :i
cs ask [t1 t2 t3][pu] ask "t2 [ht]
ask "t1 [ pd setPW 2 setPC 12] ask "t3 [setPC 0 setPW 1]

repeat 360 * :n1 * :n2
[ask "t2 [setpos ask "t1 [pos] repeat repc - 1 [lt :i / :n2 bk :s1 / :n2]]
ask "t3 [setpos ask "t2 [pos] pd lt 1 / :n1 bk :s2 / :n1 pu]
;or pd fd :s2 / :n1 pu rt 1 / :n1]
ask "t2 [setpos ask "t3 [pos] repeat repc - 1 [fd :s1 / :n2 rt :i / :n2]]
ask "t1 [setpos ask "t2 [pos] fd :s1 / :n2 rt :i / :n2]]
end
```

Cartesian and polar plots of the cycloidally modulated waves

We can ask what are the shapes of the Cartesian and the polar functions created by cycloidal motion instead of circular one. Below is one of the many possible procedures; the cycloid is drawn by turtle t4.

```
to sinepolarplot3 :s1 :s2 :n1 :n2 :i
let "hx 2 * :s2 / :n1 / :n2 / 3.14
;initial positions for turtles
home ask [t1 t2 t3 t4][pu setpos [-300 0]pd]
;ask "t4 [repeat :p [fd :s rt 1]]
ask "t1 [pu setycor ask "t4 [ycor] pd] ask [t1 t5][setPW 2 pu] ask [t4] [ht pu] ask "t5 [ht]

repeat 360 * :n1 * :n2 [ask [t4] [setPC 1 dot lt 90 fd :s1 / :n2 rt :i / :n2
repeat repc [lt :i / :n2 bk :s1 / :n2]
setPC 0 dot rt 90 fd :s2 / :n1 rt 1 / :n1 lt 90
repeat repc [fd :s1 / :n2 rt :i / :n2] rt 90 ]
let "y ask "t4 [ycor]
ask [t1 t2] [setxcor xcor + :hx]
ask [t1 t3] [setycor :y]
ask [t1] [ifelse :y < 0 [setPC 4] [setPC 2] dot]
ask [t5] [ifelse :y < 0 [setPC 4] [setPC 2] rt 1 / :n1 / :n2 fd :y dot bk :y]]

end

? cs sinepolarplot3 1 2 3 1 1
? sinepolarplot3 1 2 5 1 -1
```

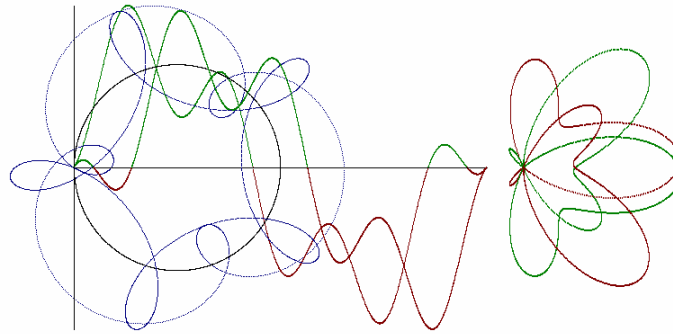



Fig. 9 Two Cartesian and polar functions created by two cycloidal motions (the centre of the rolling circle moves along the circumference of the fixed one).

Cycloidally modulated Lissajous curves

Also Lissajous curves could be modulated by cycloidal motion giving a wide variety of shapes. Below there is the procedure Lissajouscycloid, which is a modification of the procedure Lissajous1 – turtle t3 draws cycloid, turtle t4 usual circle. Turtle t5 takes xcor from t4 and ycor from t3 and draws cycloidally modified Lissajous curve.

```
to Lissajouscycloid :s1 :s2 :s3 :p :n1 :n2 :n3 :i
; s3, n3: rolling circle (t3), s1, n1: fixed circle (t3), s2, n2, p: usual circle (t4), hypo i = 1, epi i = -1
; amplitudes
cs make "r1 180 * :s1 / 3.14 make "r2 180 * :s2 / 3.14
; initial position for turtles
ask "t3 [ht hop -:r1 0 pu] ask "t4 [hop 0 :r2 rt 90 pu repeat :p [fd :s2 rt 1] pd]
ask "t5 [pu setxcor ask "t4 [xcor] setycor ask "t3 [ycor] pd setPW 2]
ask [t3 t4] [setPW 1] let "g GCD (GCD :n1 :n3) :n2
repeat 360 * :n1 * :n2 * :n3 / :g [ask [t3] [dot fd :s3 / :n1 / :n2 rt :i / :n1 / :n2
repeat repc [lt :i / :n1 / :n2 bk :s3 / :n1 / :n2]
fd :s1 / :n2 / :n3 rt 1 / :n2 / :n3
repeat repc [fd :s3 / :n1 / :n2 rt :i / :n1 / :n2]]
ask [t4] [fd :s2 / :n1 / :n3 rt 1 / :n1 / :n3]
ask [t5] [setxcor ask "t4 [xcor] setycor ask "t3 [ycor]]]
end
```

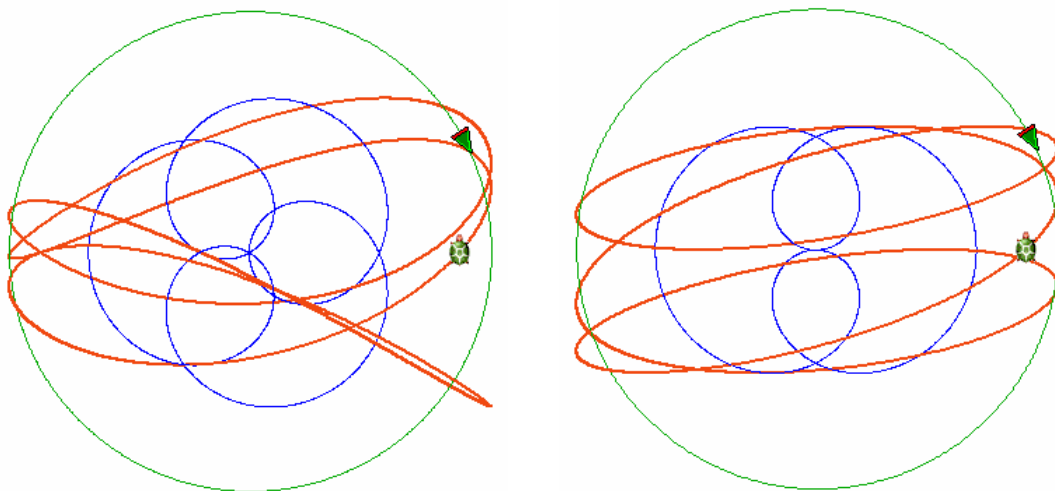


Fig. 10 Curve Lissajous 2 3 60 1 3 cycloidally modulated.

? Lissajouscycloid 2 3 1 60 1 3 3 1

? Lissajouscycloid 2 3 1 60 1 3 2 1

Comments

Playing with all these above presented programs we should not forget about two recurrent (or its iterative versions) procedures: the one creating a polar plot of a given function and the general procedure creating a curve of the intrinsic curvature described by a given function of ϕ variable (Armon, .1997).

```
? cs pu
to polarec :s :fi :kfi :fun :lfi
  if :fi > :lfi [stop]
  let "r :s * run :fun
  ifelse :r < 0 [setPC 4]
    [setPC 2] fd :r dot bk :r rt :kfi
  polarec :s :fi + :kfi :kfi :fun :lfi
end
```

```
? cs pd
to intr_graph :s :fi :kfi :fun :lfi
  if :fi > :lfi [stop]
  rt 1 let "ds :s * run :fun
  ifelse :ds < 0 [setPC 4]
    [setPC 2] fd :ds
  intr_graph :s :fi + :kfi :kfi :fun :lfi
end
```

In my opinion it is worth trying to create different curves in different ways, to compare them, to classify them in a new way, to generalize them, because there are several problems connected with the intrinsic representations of mathematical curves and also connected with the physical meaning of the concept of motion, that have not been solved yet or, at least, that are not well and simply described on the educational level.

References

Uzi Armon, (1997) An algorithm that translates intrinsic equations of curves into intrinsic procedures of these curves, <http://eurologo.web.elte.hu/lectures/intrins.html>

Károly Farkas, (2003) Logo and native language. Intrinsic procedures of some curves, in Proceedings of Eurologo 2003, Porto, Portugal, August, pp. 69-79.

Weisstein, Eric W.(2006) "Sine." From MathWorld--A Wolfram Web Resource.
<http://mathworld.wolfram.com/Sine.html>

Weisstein, Eric W. (2002) "Polar Coordinates." From MathWorld--A Wolfram Web Resource.
<http://mathworld.wolfram.com/PolarCoordinates.html>

Weisstein, Eric W. (2004) "Epicycloid." From MathWorld--A Wolfram Web Resource.
<http://mathworld.wolfram.com/Epicycloid.html>

Weisstein, Eric W. (2003) "Hypocycloid." From MathWorld--A Wolfram Web Resource.
<http://mathworld.wolfram.com/Hypocycloid.html>

Weisstein, Eric W. (2003)"Limaçon." From MathWorld--A Wolfram Web Resource.
<http://mathworld.wolfram.com/Limacon.html>

Jan Wassenaar (2004) "Limacon" <http://www.2dcurves.com/roulette/roulettel.html>

Jan Wassenaar (2005) "Rhodonea" <http://www.2dcurves.com/roulette/rouletter.html>

Xah Lee (2004) „Epicycloid and hypocycloid“
http://xahlee.org/SpecialPlaneCurves_dir/EpiHypocycloid_dir/epiHypocycloid.html

Xah Lee (1998) „Limacon of Pascal“
http://xahlee.org/SpecialPlaneCurves_dir/LimaconOfPascal_dir/limaconOfPascal.html

Xah Lee (2004) „Lemniscate of Gerono“
http://xahlee.org/SpecialPlaneCurves_dir/LemniscateOfGerono_dir/lemniscateOfGerono.html

Cycloids and limaçons in the turtle graphics

Izabella Foltynowicz, iza@rovib.amu.edu.pl

Dept of Chemistry, A. Mickiewicz University, ul. Grunwaldzka 6, 60-780 Poznań, Poland

Abstract

In my understanding the main purpose of education in science is to help our students with deeper understanding of the main ideas of mathematics and physics and teach them to find the proper tools and methods of discovering the rules, and as a result to encourage them to study on their own to continue the so-called lifelong learning process.

The ready demonstration programs prepared by the teacher or found in the internet are applied to check the way the results depend on the data initially introduced. If the student writes a demonstration program himself/ herself/ they get a chance of deeper understanding of the problem. One approach does not exclude the other or rather they are complementary. We should study the problem from different points of view; to show different approaches to the same problem. The purpose of the paper is to give a proper tool rather than to give ready answers.

In this paper two points of view are contrasted and compared: the intrinsic (pen-coordinate system) and the extrinsic one (page coordinate system) and also Cartesian and polar plots of the sine family function. Uniform circular motion is a source of a sine wave, circular motion combined with translation gives regular cycloid and a combination of two circular motions produce Lissajous curves and when one circle rolls upon another circle the cycloids are created. The rotational motion is convenient to be realized in intrinsic way, when it is combined with translation or another rotation it is convenient to relate it to the extrinsic coordinate system but it is not necessary. Intrinsic realization of combined motions slows down the process of drawing. Then we can simulate the parallel processes using many turtles.

The following two articles have inspired me to do all this trials: (Armon, 1997) and (Farcas, 2003). Another reason for writing this work is my experience connected with quantum chemistry course when we are trying to imagine what the orbitals looks like. The source of our difficulties is the fact that the students have seen polar plots of the sine family function never before.

I start with the source definition of the sine function. Then I show the simplest way to create the Lissajous figures (the conventional way of creating them from the system of parametric equation in Excel is shortly presented in Appendix 1). Next I demonstrate how the circular motion generates a Cartesian plot and next the polar plot of the sine family functions (rhodonea, limaçons). (It was like discovering the original simplicity of the problem which we have already learnt to see as complicated.) At the next step the superposition of two waves is graphically presented in both coordinate systems. I also present the procedure for generation of a family of cycloidal curves (in a wide meaning) – hypo- and epi-cycloids are the special cases of this general procedure. Then I propose to investigate the Cartesian and polar plots of cycloidally modulated waves and also cycloidally modulated Lissajous figures. This paper presents only a few curves from a huge variety and number of possible ones that can be generated using the procedures proposed. At the end I mention the alternative way of creating polar plots described in details in Appendix 2 (in the form of a laboratory exercise) and a beautiful way of creating cycloids and other curves defined by intrinsic curvature proposed by Armon (Armon, 1997).

I intend to show that Logo is convenient and fruitful tool for investigation of the Cartesian plots and polar plots of the curves from the families of limaçons, rhodoneas and cycloids created in the intrinsic style.

Keywords

circular motion, harmonic oscillator model, complex harmonic motion, Lissajous curves (figures), polar coordinates, sine function, cycloid, limaçon

Introduction – the definition of the sine function

How to show, how to explain the main ideas of mathematics and physics? How to avoid routine in thinking? One possibly answer is: by creating new routines (!) and using them in a trial and error approach.

The harmonic oscillator is the most powerful and most widely used single model in all of physical science. It is the basis of much of our understanding of the behaviour of molecules and electromagnetic radiation. In the harmonic oscillator model the sine function describes displacement, while the cosine – velocity, as a function of time,

The sine function and every superposition (linear combination) of different sinusoidal waves (Fourier series) are the solutions of the classical wave equation. The sine function is an important component of the solutions of quantum-mechanical wave equation (orbitals). What is the source definition of the sine function? Probably when hearing “the sine function” most people see a Cartesian plot of this function, a sinusoidal wave. But we know, that there is not one and only way to represent this function graphically. The Cartesian plot is one of many possible conventions. The source definition of the sine function is (Weisstein, 2006):

The sine function is the vertical coordinate of the arc endpoint of the unit circle rotating uniformly counter clockwise.

Does this rotation need to be counter clockwise? It depends on the initial position of the rotating point (e.g. (one could say) turtle) or, in other words we can say that the initial position of the arc endpoint (turtle drawing the unit circle) depends on the direction of the circular motion. How – see Fig.1, inspired by (Farkas, 2003)

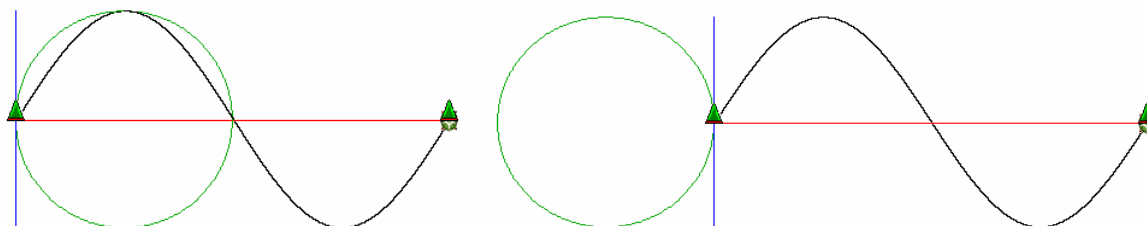


Fig.1 The plot of the sine function as a result of the origin definition: left for clockwise rotation, right – for counter clockwise one (inspired by Farkas, 2003)

The circular motion is assumed to be at a constant speed (angular speed) or alternatively with a constant frequency measured in revolutions per unit of time. It means that the angle or time is the variable. The sine curve is created when the y coordinate of the constant circular motion becomes the y coordinate of the Cartesian plot, whereas the x coordinate originates from a translation along the x axis at a constant speed related (in time) to the speed of the circular motion. The relation is established by the frequency of the circular motion, i.e. by the number of revolutions made during translation in the x direction over a distance of two diameters of the circle.

We need four turtles with the same initial position and different shapes and pen colours to draw a sine curve. The first turtle (t4) draws a circle and transfers his y coordinate to another turtle (t3), the third one (t2) goes along the x axis (i.e. the propagation direction of the wave) with a proper constant step (hx) and the last one (t1) takes the y coordinate from t4 or t3 and x coordinate from t2 and draws a sine curve as a result. Turtles t2 and t3 are redundant for drawing sine curve: they only transfer coordinates between the turtle performing the circular motion (t4) and the turtle performing the sine curve (t1), but they visualize the whole process very well.

It is convenient to assume that the circle is performed by the statement:

repeat 360 [fd :s rt 1]

where :s parameter is the arc length step corresponding to Δs and $\Delta\varphi$ is constant and equal to 1. Therefore $1/s$ is a measure of the intrinsic curvature of the circle and also the angular speed of the point moving over a circular orbit. Let the sine function be defined as:

$$y = a \sin (n x + p)$$

where a is the amplitude, n is the frequency and p is the phase. If $n = 1$, the distance of two diagonals of the circle must be completed during one revolution of a circle, for $n = 2$ – during two revolutions of a circle etc. The amplitude a is the radius of the circle. Let one step in translation in x direction be hx . Phase p is the initial position of the turtle drawing a circle (circular motion at a constant speed) measured in degrees. Phase 90 means that $y = a \cos (n x)$.

We can choose a , n , and p as an input for the procedure and calculate Δs and hx or we can start with Δs , n and p and calculate a and hx . The sine function is described by three independent variables. The table below shows the mutual relations between the parameters.

$a = \frac{180 \cdot \Delta s}{\pi}$	$\Delta s = \frac{\pi \cdot a}{180}$	$hx = \frac{2 \cdot \Delta s}{n \cdot \pi} = \frac{2 \cdot a}{n \cdot 180}$
90	$\frac{\pi}{2} = 1.57$	$\frac{1}{n}$
$\frac{180}{\pi} = 57.3$	1	$\frac{1}{n} \cdot \frac{2}{\pi} = 0.64$
100	$\frac{\pi \cdot 100}{180} = 1.74$	$\frac{1}{n} \cdot \frac{100}{90} = \frac{1}{n} \cdot 1.11$

Table 1. The relations between Δs , a , n and hx parameters of motion ($\Delta\varphi = 1$)

The procedure `sinecurve1` creates a sine curve with four turtles, procedure `sinecurve2` – with two turtles. What will happen if we change `rt` into `lt` in these two procedures?

```

to sinecurve1 :a :n :p
; y = a sin (n x + p)
make "s :a * 3.14 / 180 make "hx 2 * :s / :n / 3.14
; initial positions for turtles
cs ask [t1 t2 t3 t4][pu setpos [-300 0] pd]
ask "t4 [repeat :p [fd :s rt 1]]
ask "t1 [pu setycor ask "t4 [ycor] pd]

repeat 360 * :n [ask [t4] [fd :s rt 1] ask [t1 t2] [setxcor xcor + :hx]
ask [t1 t3] [setycor ask "t4 [ycor]] wait 10]

end

? sinecurve1 100 2 90
? ask [t1 t2 t3 t4][stamp]
```

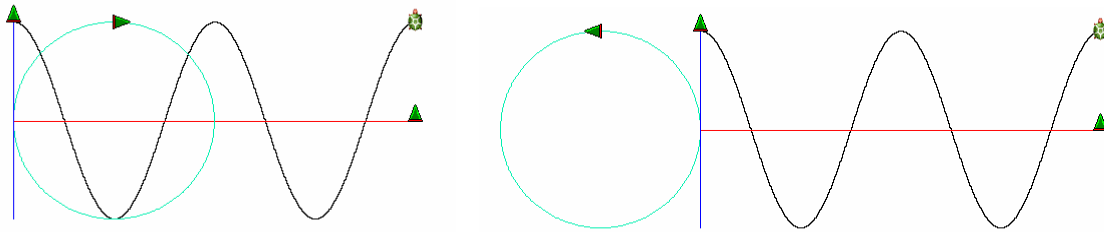


Fig.2 The cosine function with *rt* (left) and *lt* (right) created by procedure *sinecurve1*

```
to sinecurve2 :a :n :p
;without turtles t2 and t3
;y = a sin (n x + p)
make "s :a * 3.14 / 180 make "hx 2 * :s / :n / 3.14
;initial positions for turtles
cs ask [t1 t4][pu setpos [-300 0] pd]
ask "t4 [repeat :p [fd :s rt 1]]
ask "t1 [pu setycor ask "t4 [ycor] pd]

repeat 360 * :n [ask [t4] [fd :s rt 1] ask [t1 ] [setxcor xcor + :hx setycor ask "t4 [ycor]]
wait 10]

end

? sinecurve2 100 2 90
? ask [t1 t4][stamp]
```

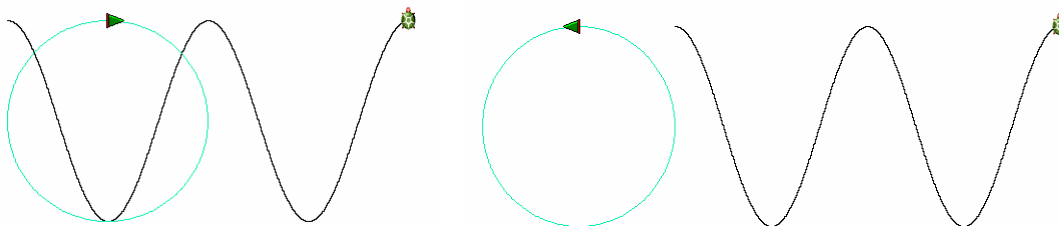


Fig.3 The cosine function with *rt* (left) and *lt* (right) created by procedure *sinecurve2*

Lissajous curves in the turtle graphics.

This family of curves was investigated by Nathaniel Bowditch in 1815, and later in more detail by Jules Antoine Lissajous in 1857.

Jules Antoine Lissajous (1822-1880) was a French mathematician and physicist. He was interested in waves and developed an optical method for studying vibrations. Lissajous curves have applications in physics, astronomy, and other sciences.

Some of these figures, and geometrical methods of constructing them, have been discovered independently, e.g. the Lemniscate of Gerono (eight curve) (Xah Lee, 2004).

The Lissajous figures are created by the system of parametric equations which describes two perpendicular harmonic vibrations:

$$\begin{cases} x = r_1 \sin(n_1 t + p) \\ y = r_2 \sin(n_2 t) \end{cases}$$

Appendix 1 shows how to create the Lissajous curves conventionally in Excel.

In the previous section we have checked that we understand really well how the family of sine curves could be generated from the circular motion. Do we need the sinusoid to generate the Lissajous curves, or could only two circular motions create these figures? If the latter was possible, it would be the simplest way to create the Lissajous figures.

The examples of the initial positions of the turtles are shown in Fig.4.

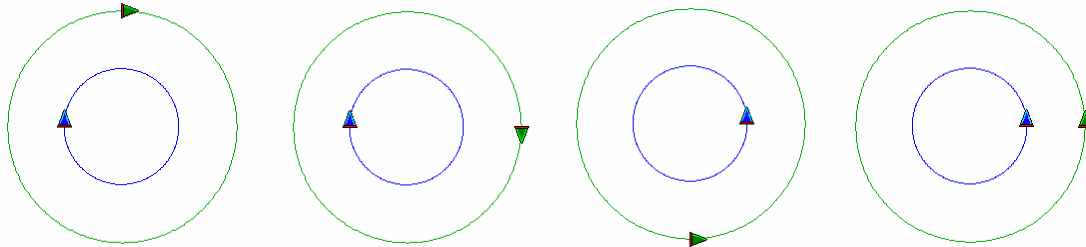


Fig. 4 The initial positions of the turtles for two concentric circles for the phase difference 0 and 90: two left pictures for rt , the next two for lt .

The time variable t , changes from the moment when the process starts to the moment when the process ends (after n_1 revolutions along first circle and n_2 revolutions along the second circle, circular motions are simultaneous). The turtle drawing the Lissajous curve takes the x coordinate from one turtle moving along the green circle and y coordinate from the turtle drawing the second, blue, circle. We can use three turtles or five turtles. In the last case these additional two turtles transfer the coordinates between those drawing the circles and the one drawing the Lissajous figure showing two perpendicular harmonic vibrations. We can also use 3 turtles and make two of them invisible: in that case we will see only one turtle drawing the Lissajous curve. The parameters of the procedure (data) could be the steps for circular motion (Δs_1 and Δs_2) or the radii (amplitudes r_1 and r_2), phase (p) and frequencies (n_1 and n_2). After some trials we will see that the shape of the L figures depends on the ratio of the frequencies of the two perpendicular waves. Dissimilar shapes can be selected by dividing the number of repetitions by the greatest common divisor of the two frequencies or by choosing only these values of frequencies whose greatest common divisor is 1.

The procedures are given for rt case only. The turtles must be created with proper attributes at the beginning.

to turtles

```
new "Turtle [penColour "green4]
ask "t2 [setshape loadImage "|Triangle Turtle.lgf]
new "Turtle [penColour "blue]
ask "t3 [setshape loadImage "|Triangle Turtleb.lgf]
new "Turtle [penColour "green4]
ask "t4 [setshape loadImage "|Triangle Turtle.lgf]
new "Turtle [penColour "paleRed penwidth 2]
ask "t5 [setshape loadImage "|Turtle.lgf]
ask "t1 [setshape loadImage "|Triangle Turtleb.lgf] setpencolour "blue]
end
```

to GCD :a :b

```
while [:b > 0] [let "r mod :a :b let "a :b let "b :r]
op :a
end
```

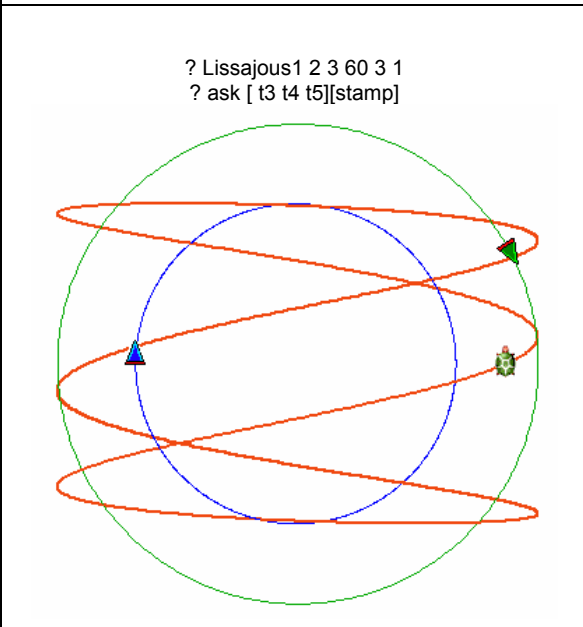
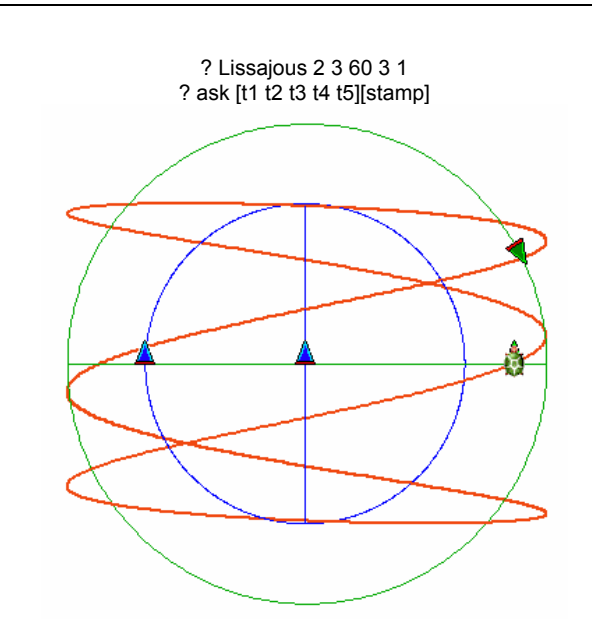
to hop :dx :dy

```
pu rt 90 fd :dx lt 90 fd :dy pd
end
```

```
to Lissajous :s1 :s2 :p :n1 :n2
cs
;amplitudes
make "x 180 / 3.14
make "r1 :s1 * :x make "r2 :s2 * :x
;initial position for turtles
ask "t3 [hop - :r1 0]
ask "t4 [hop 0 :r2 rt 90 pu repeat :p
[fd :s2 rt 1] pd ]
ask "t5 [pu setxcor ask "t4 [xcor]
setycor ask "t3 [ycor] pd]

repeat 360 * :n1 * :n2 / GCD :n1 :n2
[ask [t3] [fd :s1 / :n2 rt 1 / :n2]
ask [t1 t5] [setycor ask "t3 [ycor]]
ask [t4] [fd :s2 / :n1 rt 1 / :n1]
ask [t2 t5] [setxcor ask "t4 [xcor]]
wait 10]
end
```

```
to Lissajous1 :s1 :s2 :p :n1 :n2
cs
;without t1 and t2
;amplitudes
make "x 180 / 3.14
make "r1 :s1 * :x make "r2 :s2 * :x
;initial position for turtles
ask "t3 [hop - :r1 0]
ask "t4 [hop 0 :r2 rt 90 pu repeat :p
[fd :s2 rt 1] pd ]
ask "t5 [pu setxcor ask "t4 [xcor]
setycor ask "t3 [ycor] pd]
repeat 360 * :n1 * :n2 / GCD :n1 :n2
[ask [t3] [fd :s1 / :n2 rt 1 / :n2]
ask [t4] [fd :s2 / :n1 rt 1 / :n1]
ask [t5] [setxcor ask "t4 [xcor]
setycor ask "t3 [ycor]]
wait 10]
end
```



In the following demo the statement “/ GCD :n1 :n2” in the Lissajous procedure may be removed.

```
to demoLissajous
for "s1 [2 3] [for "n1 [1 5][for "n2 (se :n1 5) [for "p [0 360 15] [if GCD :n1 :n2 = 1
[(print [phase=] :p) (print [amplitude1=] :s1) (print [amplitude2=] 3)
(print [frequency1=] :n1) (print [frequency2=] :n2)
Lissajous :s1 3 :p :n1 :n2] wait 100] wait 1000]]]
end
```

The bolded part of the Lissajous procedure could have another different shape – it gives the same result but works quicker and less accurate.

```
repeat 360 [ask [t3] [dot fd :s1 * :n1 rt :n1] ask [t4] [dot fd :s2 * :n2 rt :n2]
ask [t5] [setxcor ask "t4 [xcor] setycor ask "t3 [ycor] dot]]
```

It can be improved and saved as a web project, but it is not our main purpose.

We have created a tool for investigation of the L curves and by doing it ourselves we have understood what L figures really are.

Polar coordinates and polar plot of the sine functions

What determines the choice of the coordinate system in a given situation? Why the Cartesian coordinates are not used to identify the position of our town on the globe? For the same reason we do not use the Cartesian coordinates to determine the position of an electron with respect to the atomic nucleus. The reason is symmetry. When the coordinate system suits the symmetry of the object under consideration, the equations describing it become simpler, and the calculations easier. For example: the equation of a circle of radius r centred at the pole is $r^2 = x^2 + y^2$ in the Cartesian coordinates, while in the polar coordinates it is: $r = \text{const.}$ I suspect that in everyday life we use polar coordinates and their three dimensional counterpart, i.e. spherical coordinates (in geography), more frequently than the Cartesian coordinates. However, we usually do not know how simple mathematical functions look in the polar coordinates. What benefit could come from drawing them in polar coordinates? Could we understand better for example the shapes of the orbitals (one electron wave functions which are the solutions of time independent Schrödinger equation) in chemistry? Let us investigate only one family of sine functions, namely $r = \sin(n\varphi)$. Is there anything the students/pupils can discover in this area? I ask my students to describe in a laboratory report their findings they found particularly interesting. One of the students answered: "Honestly speaking, every subsequent function was a huge discovery, great news and a great riddle for me." His answer encouraged me to share my didactic experience with you. Appendix 2 gives a description of this laboratory exercise, which after completion becomes the laboratory report.

For the definition of the polar coordinates see for example (Weisstein, 2002).

R is the radial distance from the origin (initial position of the turtle), φ is defined as the counter clockwise angle from the x axis, but in Logo it will be convenient to choose φ as the clockwise angle from the N-S axis (by the way: it is a good exercise to see what is the difference between the different possibilities of choosing the reference axis and the direction of counting the angle).

We offer a job to the fifth turtle (besides four introduced in the Introduction) who for subsequent values of angle φ (from 0 to 360 with the step 1) draws a line (vector) of length equal to the value of the sine function for this particular value of variable, namely equal to $\sin \varphi$. The value of $\sin \varphi$ is equal to the value of y coordinate of the turtle rounding along the circle (and turtle oscillating along the y axis). Besides of the usual "mathematical convention" we can try also the frequently used convention of drawing the radii (vectors r) corresponding to the negative values of the function as positive and in different colour. Let us call it the "two-colour convention" (unfortunately I use two colours in every convention).

to turtles

```
repeat 4 [new "Turtle []]
ask [t2 t3 t4] [setshape loadImage "|Triangle Turtle.lgf]
ask "t1 [setPW 2]
ask "t5 [setPW 2 setHomestate [[200 0]0] setshape loadImage "|Turtle.lgf]
end
```

Below the procedure `sinepolarplot` which is a good tool to demonstrate the shapes of the sine family functions in the polar coordinates in the "mathematical convention"

```
to sinepolarplot :a :n :p
; y = a sin (n x + p)
let"s :a * 3.14 / 180 let "hx 2 * :s / :n / 3.14
; initial positions for turtles
cs ask [t1 t2 t3 t4][pu setpos [-300 0]pd]
ask "t4 [repeat :p [fd :s rt 1]]
ask "t1 [pu setycor ask "t4 [ycor] pd]
ask [t1 t5][setPW 2 pu]

repeat 360 * :n [let "y ask "t4 [ycor]
```

```
ask [t4] [fd :s rt 1]
ask [t1 t2] [setxcor xcor + :hx]
ask [t1 t3] [setycor :y]
ask [t1] [ifelse :y < 0 [setPC 4] [setPC 2] dot]
ask [t5] [ifelse :y < 0 [setPC 4] [setPC 2] fd :y dot bk :y rt 1 / :n]
wait 50]
```

end

? sinepolar dot 90 2 0

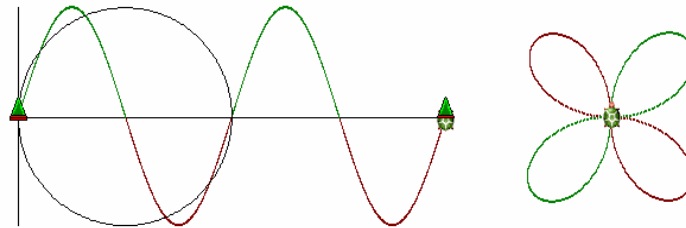


Fig. 5. The Cartesian plot of the function $y = \sin 2x$ and the polar plot of the function $r = \sin 2\theta$

On the Fig. 5 is one example of rhodonea (Wassenaar, 2005). As before, we can remove the redundant turtles t2 and t3, as well as hide the motion of turtles t4 and t1 if we would like to see the polar plot only. To obtain the functions from the limaçon (Weistein, 2003) family we must only add the constant value to the ycor in the procedure sinepolar dot (Fig. 6 and 7 are the examples).

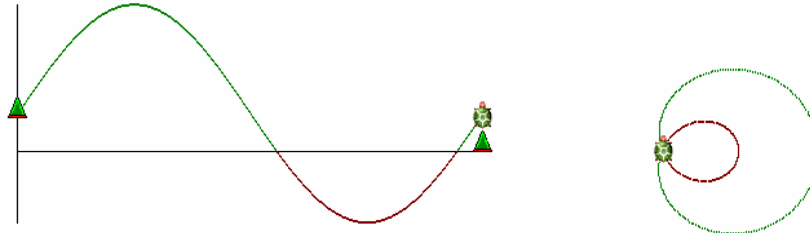


Fig. 6 The Cartesian plot of the function $y = 1/3 + \sin x$ and the polar plot of the function $r = 1/3 + \sin \theta$

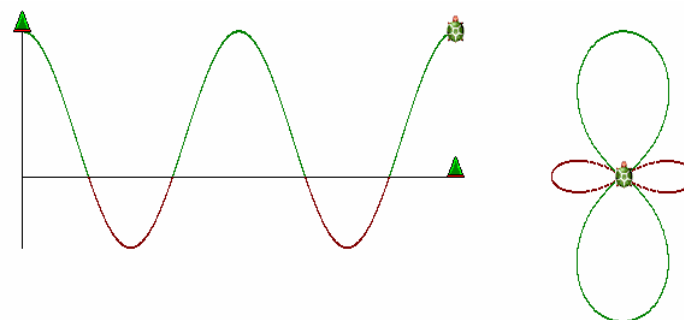


Fig. 7 The Cartesian plot of the function $y = 1/3 + \cos 2x$ and the polar plot of the function $r = 1/3 + \cos 2\theta$

The example result for fractional n and enlarged number of repetitions is shown on the Fig.8.

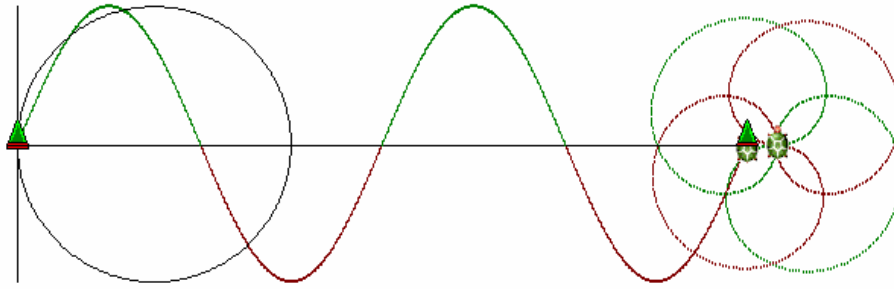


Fig. 8 The Cartesian plot of the function $y = \sin \frac{2}{3}x$ and the polar plot of the function $r = \sin \frac{2}{3}\phi$

Program startij demonstrates the limaçon (Wassenaar, 2004) family functions $r = a + \sin(i/j * \phi)$ and shows how the shapes depend on the value of a for a given small integer values of i and j . The positive values are in red, negative in blue, the circle of radius a with black.

```
to sinpij :a :i :j
; r = a + sin (i/j * phi)
cs
repeat 360 * :i * :j / GCD :i :j [let "y 100 * (sin :i / :j * (repc - 1)) fd :a + :y
ifelse :y < 0 [ setpc 1] [setpc 12] setpw 3 dot bk :y setpc 0 setpw 2 dot bk :a rt 1]
end

to startij
pu ht
for "i [1 5][ for "j [1 5][if GCD :i :j = 1 [(pr :i :j)
for "a [100 0 -5] [sinpij :a :i :j wait 10] wait 1000]
for "a [0 -100 -5] [sinpij :a :i :j wait 10] wait 1000]]
end
```

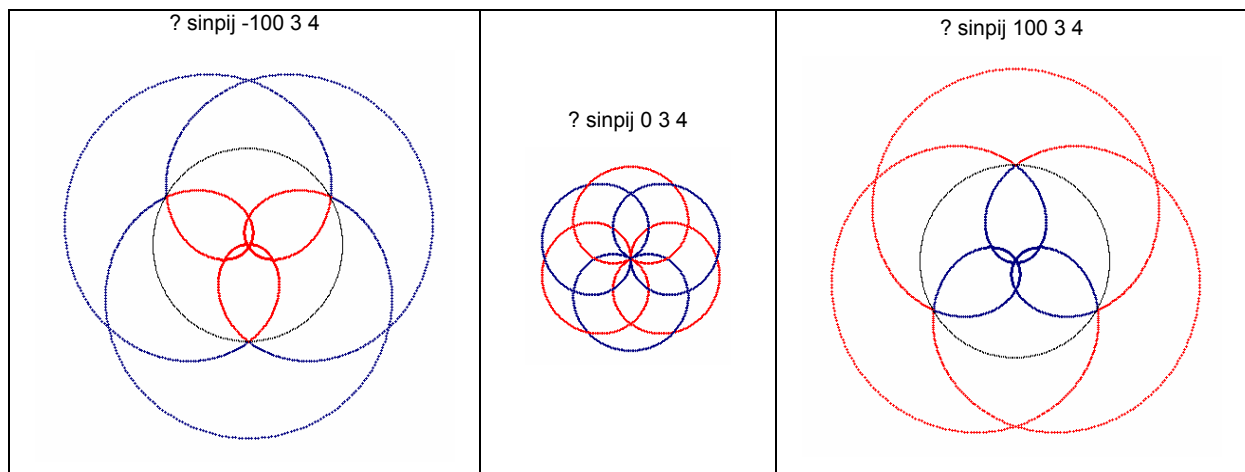


Fig. 9. The polar plots of the functions $r = -1 + \sin \frac{3}{4}\phi$, $r = \sin \frac{3}{4}\phi$ and $r = 1 + \sin \frac{3}{4}\phi$.

Superposition of waves

Now we are ready to demonstrate the Cartesian and polar superposition (interference, beat) of any two waves, each of them characterized by three parameters: amplitude, frequency and phase. We could offer a job for 12 turtles: t4 and t6 are making circular motions, t3 and t9 demonstrates one-dimensional (y direction) harmonic motions, t2 and t8 are making translation in x direction, t1 and t7 create Cartesian plots, t5 and t10 – polar plots and t11 and t12 – Cartesian and polar sums of two given waves. For clarity we can make some of the plots invisible as well as we can hide some of the turtles to show step by step how the superposition

of two waves is created and to compare clearly the components and the sum in the Cartesian and polar coordinates. We can also compare the polar plots in the “mathematic” and the “two-colour” conventions.

It is a certain inconsistency to avoid, but only in the Cartesian coordinates, not in the polar ones, when the amplitudes are not the same: one could have the impression that the wave of a small amplitude propagates with a lower velocity but it is a wrong conclusion (see Fig 10, the Cartesian components and the polar sum). We can easily improve the procedure below by scaling hx value by the values of two amplitudes; $hx1$ ought to be of the same value as $hx2$.

? sinpolarsumdot 90 45 3 6 0 90

? ask all [stamp]

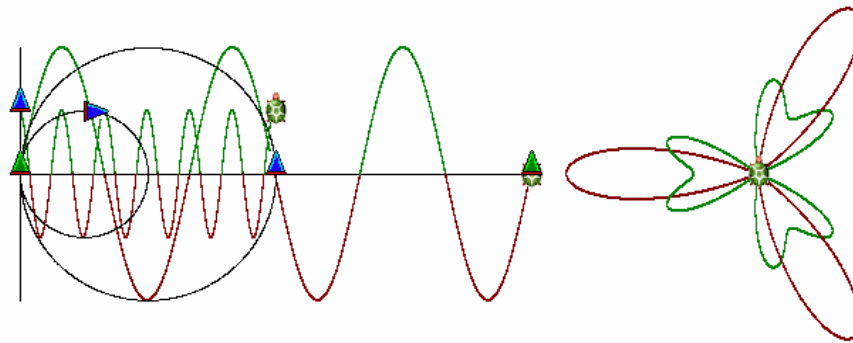


Fig. 10. The Cartesian components and the polar sum of two waves of different amplitudes

Below is one of many possible procedures, sinpolarsumdot:

```
to sinpolarsumdot3 :a1 :a2 :n1 :n2 :p1 :p2
; y1 = a1 sin (n1 x + p1), y2 = a2 sin (n2 x + p2)
let"s1 :a1 * 3.14 / 180 let"s2 :a2 * 3.14 / 180
let "hx1 2 * :s1 / (:n1 * :n2 * 3.14)
let "hx2 2 * :s2 / (:n2 * :n1 * 3.14)
ifelse :hx1 > :hx2 [let "hx :hx1][let "hx :hx2]
; initial positions for turtles
cs ask [t1 t2 t3 t4 t6 t7 t8 t9 t11][pu setpos [-300 0]pd]
ask [t1 t12][st pu]
ask [t4 t6 t8 t9][ht pu] ask [t3 t2][ht]
ask "t4 [repeat :p1 * :n2 [fd :s1 / :n2 rt 1 / :n2]]
ask "t6 [repeat :p2 * :n1 [fd :s2 / :n1 rt 1 / :n1]]
ask "t1 [pu setycor ask "t4 [ycor] pd]
ask "t7 [pu setycor ask "t6 [ycor] pd]
ask [t1 t5 t7 t10][ht pu]

repeat 360 * :n1 * :n2 [let "y1 ask "t4 [ycor]
let "y2 ask "t6 [ycor] let "y :y1 + :y2
ask [t4] [fd :s1 / :n2 rt 1 / :n2]
ask [t6] [fd :s2 / :n1 rt 1 / :n1]
ask [t1 t2] [setxcor xcor + :hx1]
ask [t1 t3] [setycor :y1]
ask [t7 t8] [setxcor xcor + :hx2]
ask [t7 t9] [setycor :y2]
ask [t1 t11][setxcor xcor + :hx setycor :y]
ask [t1] [ifelse :y1 < 0 [setPC 4] [setPC 2] dot]
ask [t7] [ifelse :y2 < 0 [setPC 12] [setPC 1] dot]
```



```

ask [t5] [ifelse :y1 < 0 [setPC 4] [setPC 2]
      fd :y1 dot bk :y1 rt 1 / :n1 / :n2]
ask [t10] [ifelse :y2 < 0 [setPC 12] [setPC 1]
      fd :y2 dot bk :y2 rt 1 / :n1 / :n2]
ask [t11] [ifelse :y < 0 [setPC 13] [setPC 11] dot]
; mathematical convention
ask [t12] [ifelse :y < 0 [setPC 13] [setPC 11]
      fd :y dot bk :y rt 1 / :n1 / :n2 ]
; two-colour convention
;
;      ask [t12] [ifelse :y < 0 [setPC 13 bk :y dot fd :y]
;                                [setPC 11 fd :y dot bk :y]
;                                rt 1 / :n1 / :n2 ]
;
end

```

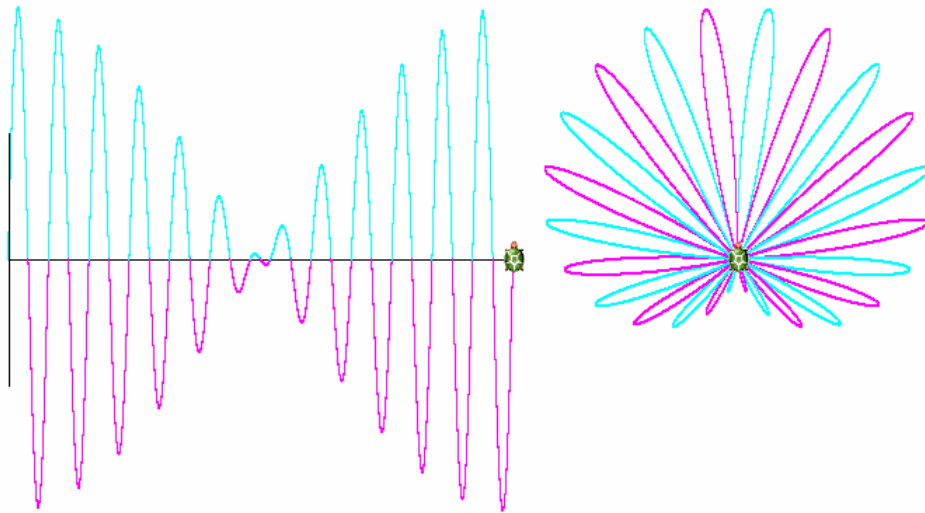


Fig. 11 The Cartesian and the polar (two-colour convention) plots of the superposition of two waves with parameters: 90 90 12 13 0 0 (beat)

Cycloids

A cycloid (Weisstein, 2003, 2004) is a plane curve that is the trajectory of a point lying on a circle that rolls without slipping along a straight line (regular cycloid) or upon another circle. A cycloid is called an epicycloids or hypocycloid, depending on whether the rolling circle has external or internal contact with the fixed circle. When the ratio of the radii of these two circles is rational, then the cycloidal curve is a closed algebraic curve.

In procedure `cycx` one turtle draws a regular cycloid combining two motions: circular motion (intrinsic equation of a circle, green colour) and translation in x direction (the extrinsic Cartesian coordinate system, blue colour).

```

to cycx :s
  cs pu setxy -100 0 setheading 270
  repeat 360 [dot fd :s rt 1 setxcor xcor + :s wait 10]
  ;circuference is 360 * s, radius is 180 * s / pi
end

```

The procedure `cyc` draws a regular cycloid on a wholly intrinsic way and may be simplified to the procedure `cyc1`:

```
to cyc
cs setPW 1 st setxy -100 0 setheading 270
repeat 360 [pd st setPC 12 fd 1 rt 1 wait 20
  pu ht repeat repc [lt 1 bk 1]
  pd st setPC 0 bk 1 wait 20
  pu ht repeat repc [fd 1 rt 1]]
end
```

```
to cyc1 :s
cs pu setPW 1 ht setxy -100 0 setheading 270
repeat 360 [dot fd :s rt 1 repeat repc [lt 1] bk
:s repeat repc [rt 1]]
end
```

The main procedure for hypo- and –epicycloids could be as follows:

```
to hecycs :f :i :s
; i = -1 epi, i = 1 hypo, f = 1, 2, 3, 4, ...
; dfi of fixed circle = const; s (scale) (from 1 to 2)
cs ht
repeat 360 [pd setPC 12 fd :s rt :i * :f
  pu repeat repc [lt :i * :f bk :s]
  pd setPC 0 lt 1 bk :s
  pu repeat repc [fd :s rt :i * :f]]
end
```

where parameter $f = n_1/n_2$ (ratio of frequencies).

Procedure hecycs could have two interesting modifications: hecycsrt in which the direction of the rolling circle is the same as the fixed one and hecycsm in which the centre of the rolling circle moves along the circumference of the fixed one.

```
to hecycsrt :f :i :s
cs ht
repeat 360 [pd setPC 12 fd :s rt :i * :f
  pu repeat repc [lt :i * :f bk :s]
  pd setPC 0 fd :s rt 1
  pu repeat repc [fd :s rt :i * :f]]
end
```

```
to hecycsm :f :i :s
cs ht
repeat 360 [pd setPC 12 lt 90 fd :s rt :i * :f
  pu repeat repc [lt :i * :f bk :s]
  pd setPC 0 rt 90 lt 1 bk :s lt 90
  pu repeat repc [fd :s rt :i * :f] rt 90]
end
```

We can create the whole family of cycloidal curves in quite a different way. In the procedures hypoepicyc and hypoepicyc1 the parameters s1 and s2 are the steps of arc lengths (these values are directly proportional to the radiuses) of the rolling and fixed circles, parameters n1 and n2 are the frequencies of these two circular motions. In each of these procedures for the same n1 and n2, the ratio of frequencies is the same, but is obtained in a different way and in consequence the first procedure draws slower and more accurate than the second one.

```
to hypoepicyc :s1 :s2 :n1 :n2 :i
; hypo i = 1, epi i = -1
cs pu
repeat 360 * :n1 * :n2
[setPC 12 setPW 2 dot fd :s1 / :n2 rt :i / :n2
repeat repc [lt :i / :n2 bk :s1 / :n2]
setPC 0 setPW 1 dot lt 1 / :n1 bk :s2 / :n1
repeat repc [fd :s1 / :n2 rt :i / :n2] ]
end
```

```
to hypoepicyc1 :s1 :s2 :n1 :n2 :i
; faster but with low quality
cs pu setPW 2
; additional with phase difference :p
; repeat :p / :n1 [fd :s1 * :n1 rt :i * :n1]
repeat 360 [setPC 12 dot fd :s1 * :n1 rt :i * :n1
repeat repc [lt :i * :n1 bk :s1 * :n1]
setPC 0 dot lt :n2 bk :s2 * :n2
repeat repc [fd :s1 * :n1 rt :i * :n1]]
end
```

If in the procedure hypoepicyc $lt\ 1 / :n1$ change to $rt\ 1 / :n1$ and in hypoepicyc1 $lt\ :n2$ change to $rt\ :n2$ hypocycloid change into epicycloid with the same parameters and vice versa.

The “natural” cycloids created by the procedure hecycs could be obtained by generalized procedure hypoepicyc for the data fulfilled the basic cycloidal rule:

$$\frac{s1}{s2} = \frac{r1}{r2} = \frac{n2}{n1} = \frac{1}{f}$$

For example, the results of hecyys 3 1 1, hecyys 3 1 1.5, hypoepicyc 1 3 3 1 1 and hypoepicyc .5 1.5 3 1 1 differ only with the size.

Below is the version of the hypoepicyc procedure realized with three turtles: t1 draws the cycloid, t3 draws the fixed circle, t2 is invisible and cooperates with t1 and t3 in transferring the Cartesian coordinates and the turtle heading.

to turtles

```
repeat 2 [new "Turtle []]
ask [t1 t2 t3] [setHomestate [[-100 0]0]]
end
```

to hypoepicyc2 :s1 :s2 :n1 :n2 :i

```
cs ask [t1 t2 t3][pu] ask "t2 [ht]
ask "t1 [ pd setPW 2 setPC 12] ask "t3 [setPC 0 setPW 1]
repeat 360 * :n1 * :n2
[ask "t2 [setpos ask "t1 [pos] setheading ask "t1 [heading] repeat repc - 1 [lt :i / :n2 bk :s1 / :n2]]
ask "t3 [setpos ask "t2 [pos] pd lt 1 / :n1 bk :s2 / :n1 pu]
ask "t2 [setpos ask "t3 [pos] setheading ask "t3 [heading] repeat repc - 1 [fd :s1 / :n2 rt :i / :n2]]
ask "t1 [setpos ask "t2 [pos] setheading ask "t2 [heading] fd :s1 / :n2 rt :i / :n2]]
end
```

Let us see only one the most interesting modification: the fixed circle is additionally modified by the cycloid. Compare for example hypoepicyc2 1 3 3 1 1 with hypoepicyc3 1 2 2 1 1 and hypoepicyc3 2 1 1 2 1.

to hypoepicyc3 :s1 :s2 :n1 :n2 :i

```
cs ask [t1 t2 t3][pu] ask "t2 [ht]
ask "t1 [ pd setPW 2 setPC 12] ask "t3 [setPC 0 setPW 1]
repeat 360 * :n1 * :n2
[ask "t2 [setpos ask "t1 [pos] repeat repc - 1 [lt :i / :n2 bk :s1 / :n2]]
ask "t3 [setpos ask "t2 [pos] pd lt 1 / :n1 bk :s2 / :n1 pu]
;or pd fd :s2 / :n1 pu rt 1 / :n1]
ask "t2 [setpos ask "t3 [pos] repeat repc - 1 [fd :s1 / :n2 rt :i / :n2]]
ask "t1 [setpos ask "t2 [pos] fd :s1 / :n2 rt :i / :n2]]
end
```

Cartesian and polar plots of the cycloidally modulated waves

We can ask what are the shapes of the Cartesian and the polar functions created by cycloidal motion instead of circular one. Below is one of the many possible procedures; the cycloid is drawn by turtle t4.

to sinepolarplot3 :s1 :s2 :n1 :n2 :i

```
let "hx 2 * :s2 / :n1 / :n2 / 3.14
;initial positions for turtles
home ask [t1 t2 t3 t4][pu setpos [-300 0]pd]
;ask "t4 [repeat :p [fd :s rt 1]]
ask "t1 [pu setycor ask "t4 [ycor] pd]
ask [t1 t5][setPW 2 pu]
ask [t4] [ht pu] ask "t5 [ht]
```

```
repeat 360 * :n1 * :n2 [ask [t4] [setPC 1 dot lt 90 fd :s1 / :n2 rt :i / :n2
repeat repc [lt :i / :n2 bk :s1 / :n2]
```

```

setPC 0 dot rt 90 fd :s2 / :n1 rt 1 / :n1 lt 90
repeat repc [fd :s1 / :n2 rt :i / :n2] rt 90 ]
let "y ask "t4 [ycor]
ask [t1 t2] [setxcor xcor + :hx]
ask [t1 t3] [setycor :y]
ask [t1] [ifelse :y < 0 [setPC 4] [setPC 2] dot]
ask [t5] [ifelse :y < 0 [setPC 4] [setPC 2] rt 1 / :n1 / :n2 fd :y dot bk :y]]
end

? cs sinepolardot3 1 2 3 1 1
? sinepolardot3 1 2 5 1 -1

```

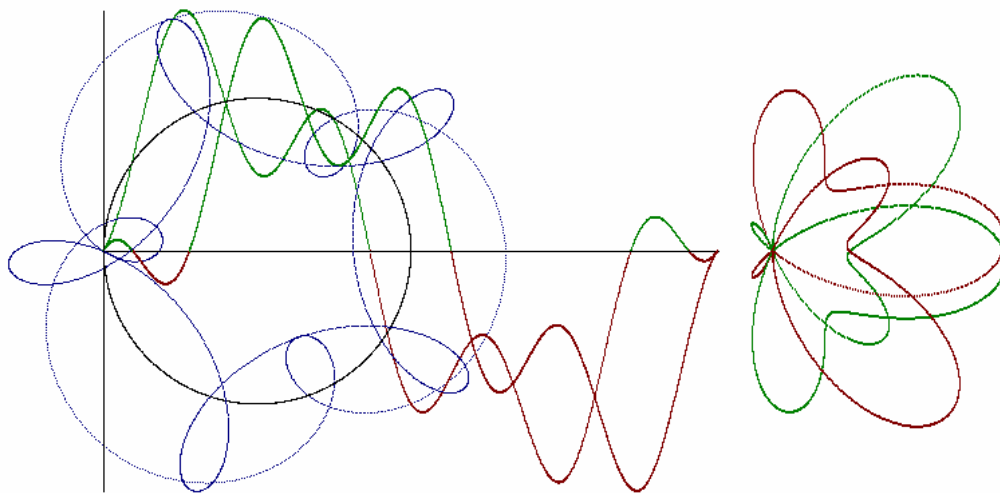


Fig. 12 Two Cartesian and polar functions created by two cycloidal motions (the centre of the rolling circle moves along the circumference of the fixed one).

Cycloidally modulated Lissajous curves

Also Lissajous curves could be modulated by cycloidal motion giving a wide variety of shapes. Below there is the procedure Lissajouscycloid, which is a modification of the procedure Lissajous1 – turtle t3 draws cycloid, turtle t4 usual circle. Turtle t5 takes xcor from t4 and ycor from t3 and draws cycloidally modified Lissajous curve.

```

to Lissajouscycloid :s1 :s2 :s3 :p :n1 :n2 :n3 :i
cs
;s3, n3: rolling circle (t3), s1, n1: fixed circle (t3), s2, n2, p: usual circle (t4)
;hypo i = 1, epi i = -1
;amplitudes
make "r1 180 * :s1 / 3.14 make "r2 180 * :s2 / 3.14
;initial position for turtles
ask "t3 [ht hop -:r1 0 pu] ask "t4 [hop 0 :r2 rt 90 pu repeat :p [fd :s2 rt 1] pd]
ask "t5 [pu setxcor ask "t4 [xcor] setycor ask "t3 [ycor] pd setPW 2]
ask [t3 t4] [setPW 1] let "g GCD (GCD :n1 :n3) :n2
repeat 360 * :n1 * :n2 * :n3 / :g [ask [t3] [dot fd :s3 / :n1 / :n2 rt :i / :n1 / :n2]
repeat repc [lt :i / :n1 / :n2 bk :s3 / :n1 / :n2]
dot fd :s1 / :n2 / :n3 rt 1 / :n2 / :n3
repeat repc [fd :s3 / :n1 / :n2 rt :i / :n1 / :n2]]
ask [t4] [fd :s2 / :n1 / :n3 rt 1 / :n1 / :n3]
ask [t5] [setxcor ask "t4 [xcor] setycor ask "t3 [ycor]]
end

```

? Lissajouscycloid 2 3 1 60 1 3 3 1

? Lissajouscycloid 2 3 2 60 1 3 2 1

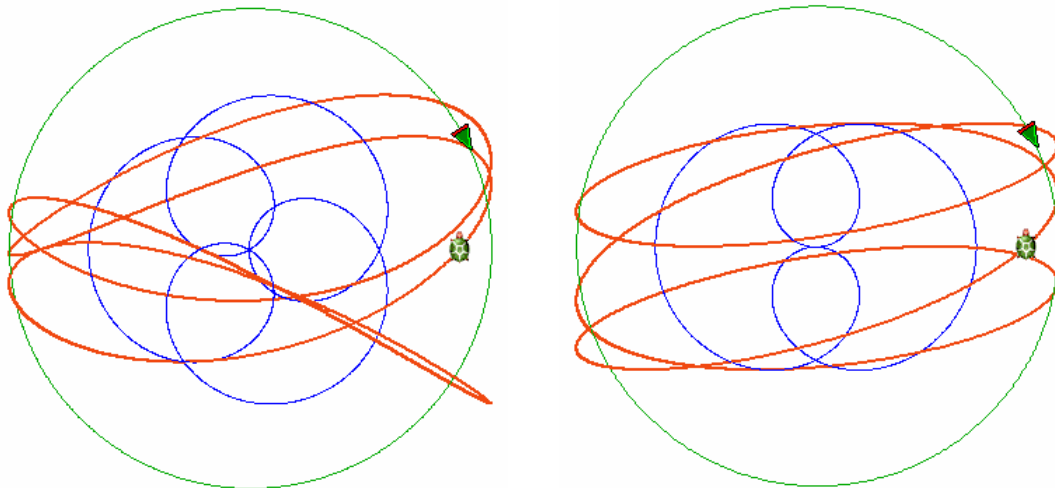


Fig. 13 Curve Lissajous 2 3 60 1 3 cycloidally modulated..

Comments

Playing with all these above presented programs we should not forget about two recurrent (or its iterative versions) procedures: the one creating a polar plot of a given function (see Appendix 2) and the general procedure creating a curve of the intrinsic curvature described by a given function of ϕ variable (Armon, 1997).

```
? cs pu
to polarec :s :fi :kfi :fun :lfi
  if :fi > :lfi [stop]
  let "x :s * run :fun
  ifelse :x < 0 [setPC 4]
    [setPC 2] fd :x dot bk :x rt :kfi
  polarec :s :fi + :kfi :kfi :fun :lfi
end
```

```
? cs pd
to intr_graph :s :fi :kfi :fun :lfi
  if :fi > :lfi [stop]
  rt 1 let "x :s * run :fun
  ifelse :x < 0 [setPC 4]
    [setPC 2] fd :x
  intr_graph :s :fi + :kfi :kfi :fun :lfi
end
```

Let us look at the program:

```
to startintr1
  cs ht pd
  for "a [-15 -1.1 .1] [cs intr_graph3 :a 0 1 wait 10]
  wait 30
  for "a [-1 1.1 .1] [cs intr_graph1 :a 0 1 wait 30]
  wait 30
  for "a [1.1 15 .1] [cs intr_graph2 :a 0 1 wait 10]
end
```

```
to intr_graph1 :a :fi :kfi
  if :fi > 1440 [stop]
  rt 1 let "x :a + (sin 3 / 4 * :fi)
  ifelse :x < 0 [setPC 4]
    [setPC 2] fd :x
  intr_graph1 :a :fi + :kfi :kfi
end
```

```
to intr_graph2 :a :fi :kfi
  if :fi > 1440 [stop]
  rt 1 let "x (:a + (sin 3 / 4 * :fi)) / :a
  ifelse :x < 0 [setPC 4]
    [setPC 2] fd :x
  intr_graph2 :a :fi + :kfi :kfi
end
```

```
to intr_graph3 :a :fi :kfi
  if :fi > 1440 [stop]
  rt 1 let "x (:a + (sin 3 / 4 * :fi)) / (-:a)
  ifelse :x < 0 [setPC 4]
    [setPC 2] fd :x
  intr_graph3 :a :fi + :kfi :kfi
end
```

and compare Fig. 14 with Fig.9.

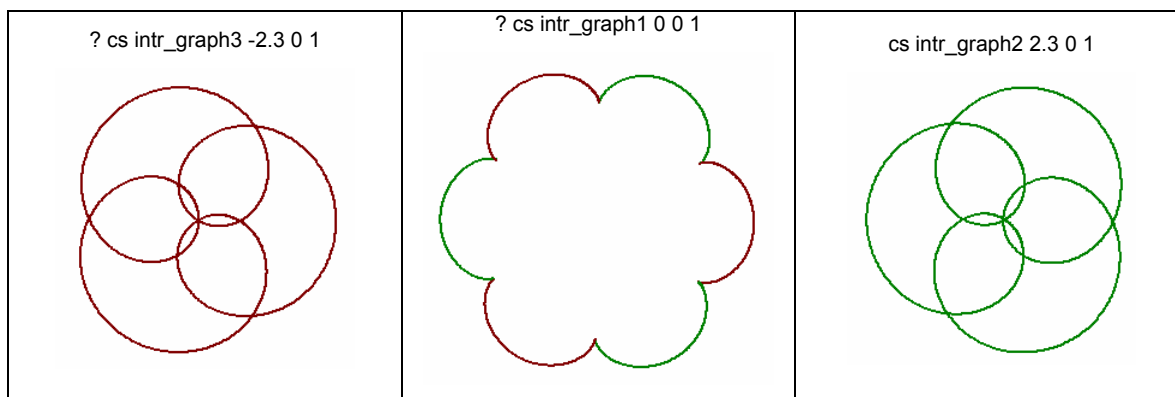


Fig. 14 The plots of the curves of the intrinsic curvature described by the functions: $-2.3 + \sin(3/4f)$, $\sin(3/4f)$ and $2.3 + \sin(3/4f)$

In my opinion it is worth trying to create different curves in different ways, to compare them, to classify them in a new way, to generalize them, because there are several problems connected with the intrinsic representations of mathematical curves and also connected with the physical meaning of the concept of motion, that have not been solved yet or, at least, that are not well and simply described on the educational level.

References

- Uzi Armon, (1997) An algorithm that translates intrinsic equations of curves into intrinsic procedures of these curves, <http://eurologo.web.elte.hu/lectures/intrins.html>
- Károly Farkas, (2003) Logo and native language. Intrinsic procedures of some curves, in Proceedings of Eurologo 2003, Porto, Portugal, August, pp. 69-79.
- Weisstein, Eric W. (2006) "Sine." From MathWorld--A Wolfram Web Resource. <http://mathworld.wolfram.com/Sine.html>
- http://www.hardycalculus.com/calciindex/IE_lissajous.htm
- <http://www-groups.dcs.st-and.ac.uk/~history/Biographies/Lissajous.html>
- Weisstein, Eric W. (2002) "Polar Coordinates." From MathWorld--A Wolfram Web Resource. <http://mathworld.wolfram.com/PolarCoordinates.html>
- Weisstein, Eric W. (2004) "Epicycloid." From MathWorld--A Wolfram Web Resource. <http://mathworld.wolfram.com/Epicycloid.html>
- Weisstein, Eric W. (2003) "Hypocycloid." From MathWorld--A Wolfram Web Resource. <http://mathworld.wolfram.com/Hypocycloid.html>
- Weisstein, Eric W. (2003) "Limaçon." From MathWorld--A Wolfram Web Resource. <http://mathworld.wolfram.com/Limacon.html>
- Jan Wassenaar (2004) "Limacon" <http://www.2dcurves.com/roulette/roulettel.html>
- Jan Wassenaar (2005) "Rhodonea" <http://www.2dcurves.com/roulette/rouletter.html>
- Xah Lee (2004) "Epicycloid and hypocycloid" http://xahlee.org/SpecialPlaneCurves_dir/EpiHypocycloid_dir/epiHypocycloid.html
- Xah Lee (1998) "Limacon of Pascal" http://xahlee.org/SpecialPlaneCurves_dir/LimaconOfPascal_dir/limaconOfPascal.html
- Xah Lee (2004) "Lemniscate of Gerono" http://xahlee.org/SpecialPlaneCurves_dir/LemniscateOfGerono_dir/lemniscateOfGerono.html

Appendix 1 Lissajous figures created conventionally in Excel

The tool: Excel

The definition: by the system of parametric equations which describes complex harmonic motion:

$$\begin{cases} x = A \sin(\alpha t + \delta) \\ y = B \sin(\beta t) \end{cases}$$

	A	B	C	D	E	F	G	H	I
1	t	Asin(alpha* t+D)	Bsin(beta* t)	tp =	-3,141593	A =	1		
2	-3,141593	-0,258819045	2,4503E-16	tk =	3,141593	alpha =	3	▲	
3	-3,078761	-0,435231099	0,125333234	ht =	0,062832	B =	1	▼	
4	-3,015929	-0,596224875	0,248689887			beta =	2		
5	-2,953097	-0,736097087	0,368124553			D =	0,262	15	▲
6	-2,890265	-0,849892693	0,481753674						
7	-2,827433	-0,933580426	0,587785252						
8	-2,764602	-0,984195608	0,684547106						
9	-2,70177	-0,999945169	0,770513243						
10	-2,638938	-0,980271175	0,844327926						

Table 1 Part of the table presenting numbers

	A	B	C	D	E	F	G	H	I
1	t	Asin(alpha* t+D)	Bsin(beta* t)	tp =	=-PI()	A =	1		
2	=tp	=A*SIN(alpha*\$A2 + D)	=B*SIN(betha*\$A2)	tk =	=PI()	alpha =	3	▲	
3	=A2+ht	=A*SIN(alpha*\$A3 + D)	=B*SIN(betha*\$A3)	ht =	=(E2-E1)/100	B =	1	▼	
4	=A3+ht	=A*SIN(alpha*\$A4 + D)	=B*SIN(betha*\$A4)			beta =	2		
5	=A4+ht	=A*SIN(alpha*\$A5 + D)	=B*SIN(betha*\$A5)			D =	=H5*PI()/180	15	▲
6	=A5+ht	=A*SIN(alpha*\$A6 + D)	=B*SIN(betha*\$A6)						
7	=A6+ht	=A*SIN(alpha*\$A7 + D)	=B*SIN(betha*\$A7)						
8	=A7+ht	=A*SIN(alpha*\$A8 + D)	=B*SIN(betha*\$A8)						
9	=A8+ht	=A*SIN(alpha*\$A9 + D)	=B*SIN(betha*\$A9)						
10	=A9+ht	=A*SIN(alpha*\$A10 + D)	=B*SIN(betha*\$A10)						

Table 2. Part of the table presenting formulas

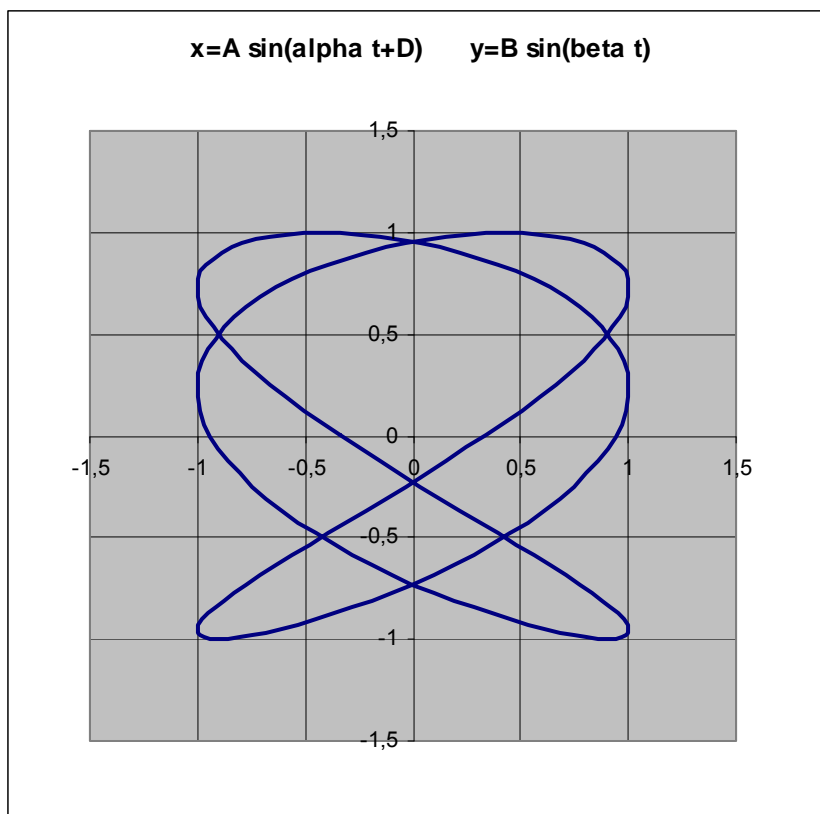
The worksheet data selected for the plot: B1:C102.

Spinner buttons: select View/Toolbars/Forms, create the Spinner button. Right-click the Spinner, then click Format Control and enter the following information: Minimum value, Maximum value, Incremental change and Cell link.

Spinner button A: 1 10 1 G2

Spinner button B: 1 10 1 G4

Spinner button D: 0 360 15 H5



Appendix 2 Polar coordinates and polar plots. Sine function family.

Full name:

Read carefully description of the exercise and then write down or paste answers for questions with red pen color.

1) Introduction

The exercise is meant as a step, for some students perhaps the first one, towards familiarisation with polar plots of mathematical functions. Understanding of the polar plots is necessary to grasp the meaning of orbitals in chemistry and to understand what the geographic coordinates are. The choice of the optimum coordination system to describe a given phenomenon is determined by symmetry. The polar and spherical plots (spherical plots are the three-dimensional correspondents of the polar ones) are more suitable to solve the problems of spherical symmetry (the atom, the globe) than the Cartesian ones. In the real life situations we more often use the polar than the Cartesian coordinates, whether we are aware of it or not. Unfortunately, in the high school curricula the Cartesian coordinates are prevalent, so familiarisation with the polar coordinates is much beneficial.

Let's begin with the simplest example of the equation of a circle of a given radius r in the polar coordinates; it is

$$r = \text{const.}$$

The most illustrative examples of polar functions are spirals of different types and trigonometric functions:

$$r = \sin(n\varphi)$$

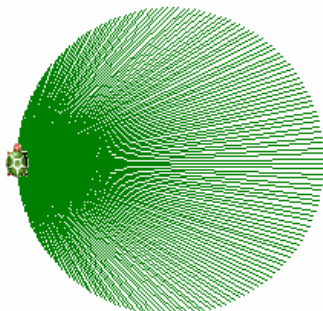
$$r = \cos(n\varphi)$$

for integer and fractional values of parameter n .

The polar plots are made by drawing a vector of a value equal the value of the function (r) calculated for the angle (φ) made by the vector with a distinguished direction. It is worth mentioning here that the distinguished direction can be that of the Cartesian axis y (analogy to the north-south direction N-S) or the direction of the x axis. Moreover, the direction with respect to which the angle φ increases must be specified. In LOGO it would be the most natural to distinguish the direction N-S – the initial direction of the turtle movement. We assume that the angle increases in the clockwise direction (sometimes the opposite convention is assumed, e.g. in some dialects of the Lisp language).

Even the simplest iteration program for drawing the polar plot of the function $r = \sin \varphi$ brings out the problems we will have to solve.

```
to sinp0
  cs st setPC 2
  repeat 360 [let „r 200 * sin repc fd :r bk :r rt 1]
end
? sinp0
```



The angle φ varies from 1 to 360 at a step 1; 200 pixels correspond to the function value of 1, so the number of 200 is used for scaling of the plot. The first remark is that it would be preferable to have the angle φ varying from 0 to 360. As the procedure parameters we would like to have: the scaling factor, step of changes in the angle φ (it does not have to be 1), the initial value of the angle and certainly the final value of angle φ . There is another problem – when the function assumes negative values the turtle moves back instead of forward (the sign of the vector r changes) and the fragment of the plot for the negative values of the function $r = \sin \varphi$ overlaps the fragment for the positive values of the function, which gives a misleading impression that these two fragments are the same. Let's call this usually used convention as the “mathematical convention” To deal with this problem we can accept the frequently used convention of drawing the radii (vectors r) corresponding to the negative values of the function as positive and in different colour. Let us call it the “two-colour convention” (unfortunately we are using two colours in every convention!). The above example in the “two-coloured convention”, still without parameters but for the angle φ varying from 0 to 360 at a step 0.1 looks like this:

```
to sinp
  cs st
  repeat 3601 [let "r 200 * sin 0.1 * (repc - 1) ifelse :r < 0 [setPC 4 bk :r fd :r ]
                                                         [setPC 2 fd :r bk :r ]
                                                         rt 0.1]
end
```

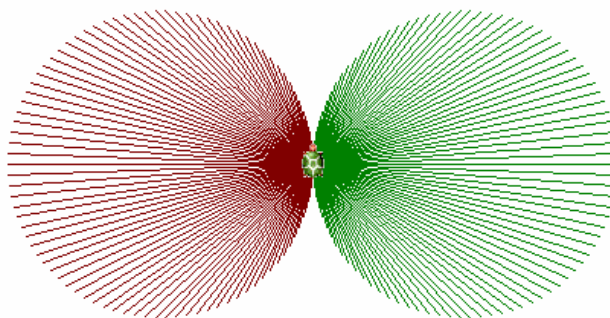
end

The same but with the parameters and assuming that the function and the variation range of the angle φ are the same and only the step of its changes is different:

```
to sinp1 :s :kfi
  cs st
  repeat 360 / :kfi + 1 [let „r :s * sin :kfi * (repc - 1)
                        ifelse :r < 0 [setPC 4 bk :r fd :r ]
                        [setPC 2 fd :r bk :r ]
                        rt :kfi]
end
```

end

? sinp1 200 2



Additionally with the initial value of φ and the function as a parameters:

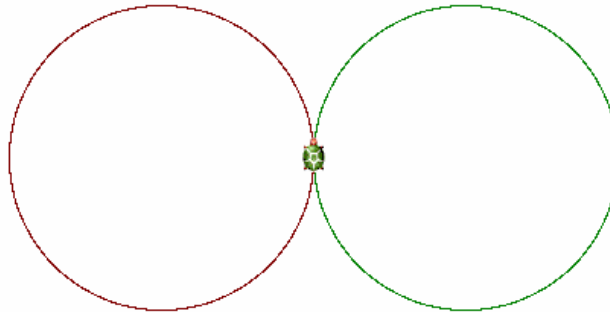
```
to polar1 :s :fi :kfi :fun
  cs st pd
  repeat 360 / :kfi + 1 [let "r :s * run :fun
                        ifelse :r < 0 [setPC 4 bk :r fd :r ] [setPC 2 fd :r bk :r ]
                        rt :kfi let "fi :fi + :kfi]
end
```

end

? polar1 200 0 1 [sin :fi]

As follows from the above, we have achieved a high degree of generality of the procedure.

- 2) Now, it would be more convenient to use the loop **while** instead of the **repeat** and let it be the first task for students. Call the procedure **polar2**. We assume that the angle *fi* always varies from 0 to 360 degrees. Paste the working procedure.
- 3) Modify the last procedure so that only the curves were drawn, not the radius-vectors to particular points of the curve. In Logo there is the primary procedure **dot** working when pen is up (**pu**). Paste the working procedure.



Below there is the recurrent way of drawing a circle in the polar coordinates.

```

to polarcircle :r :fi :kfi
  if :fi > 360 [stop]
  fd :r bk :r rt :kfi
  polarcircle :r :fi + :kfi :kfi
end
  
```

```
? cs polarcircle 200 0 1
```

The procedure permits drawing a circle of the radius of 200 pixels for the angle *fi* varying from 0 to 360 at a step 1.

- 4) Write in Imagine the recurrent version of the procedure **polar2** (**polarek**). Tested procedure paste below.
- 5) Using the recurrent program you wrote draw the polar plots of the function $r = \sin(n_1/n_2 \phi)$ for different integer values of n_1 and n_2 (for $n = n_1/n_2 > 1$ as well as $n < 1$). In the second case it is convenient to assume the final value of the angle being a multiple of 360 degree. Introduce additional parameters if needed. Paste the commands with an explanation saying which function is drawn, in which scale, what is the range of the argument and at which step it changes. An example of such a description:

```
? polar2 200 0 0.1 [sin :fi ]
draws the function sin fi, the unit value of the function corresponds to 200 pixels, the function is drawn for the angle
varying from 0 to 360 degrees at a step 0.1
```

Compare the polar plots with the Cartesian ones (Excel).

Please write down these of yours discoveries, which turned to be especially interesting for you.

Logo-like Learning of Basic Concepts of Algorithms - Having Fun with Algorithms

Gerald Futschek, futschek@ifs.tuwien.ac.at

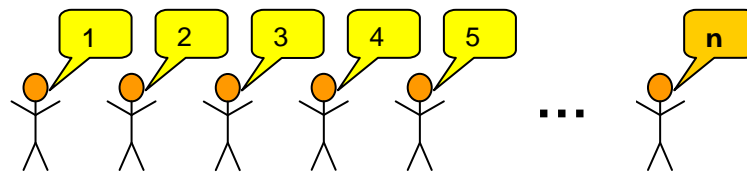
Institute of Software Technology and Interactive Systems, Vienna University of Technology

Abstract

The ability of algorithmic thinking is one of the major goals of informatics education. But learning algorithms is very often hard and boring for many students.

We show in this article that learning algorithms can be made easier and can be a lot of fun. The more the students are actively involved in the design and application of algorithms the more fun they have. We use constructivistic learning methods so the students learn how to design and detect algorithms without using a programming language.

Although we are not primarily interested in programming the way of learning is highly influenced by the Logo style of learning: creative learning by inventing algorithms and performing the algorithms themselves in smaller or even larger groups.



The students are counting themselves

The simply looking task “determine the number of students in a room” is used to demonstrate how all students of a course can be involved to learn how to design efficient algorithms.

Keywords

Logo-like learning, algorithms, group learning

Learning Algorithms

In schools and universities “algorithms” is a key topic in all informatics curricula. Although this topic is very essential and there exists much teaching experience, most of the students think that the algorithms stuff is very hard and boring. Usually there is a set of standard algorithms to be imparted to the students. Well elaborated algorithms that are sometimes very sophisticated are presented and analysed and should be understood by the students so that they are able to program them.

In this article we oppose this attitude and claim that learning algorithms can be a lot of fun even for students that have never done it before. The author gained experience in teaching algorithms in several courses for students that had no pre-knowledge in computer programming. He tried to transfer his experience in Logo education to teaching and learning algorithms.

The here presented algorithms are not intended to be programmed by the students. The students play the algorithms themselves. So they are part of the algorithms and can find easier and with more fun arguments for correctness and efficiency of the played algorithms.

Example: Determine the Number of Students

The students should determine their number. This task is easy to be understood and a first algorithm is easily found, but it is not so trivial if we want to perform this task fast also for a larger group of students. It can be used at the very beginning of an informatics course to demonstrate some basic properties of algorithms. Usually the following two solutions are proposed by the students:

1. **A counting person:** One student starts to count all the other students in a specific order. At the end the student reports the result after hopefully adding one for herself/himself.

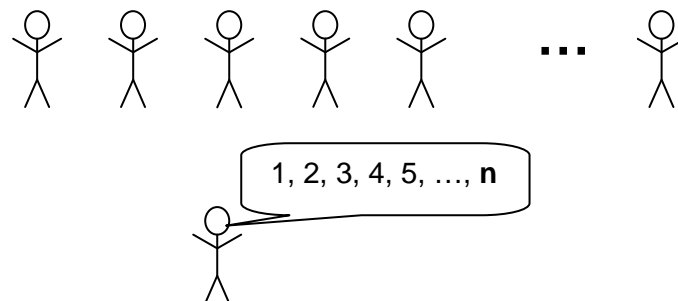


Figure 1. A student counts the other students

2. **Counting students:** Each student adds one to the previous student and tells the result to a student that has not yet counted. The last student, that has no next student, reports the result after adding one for herself/himself.

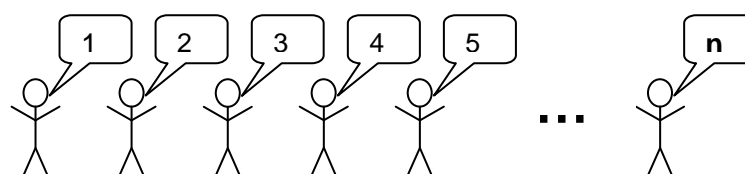


Figure 2. The students are counting themselves

These algorithms should be played by the students and the teacher may discuss the different approaches with her/his students: The first solution needs an algorithm for the counting person. The second solution needs an algorithm for each student! All students have the same algorithm except the first and the last one. The first student passes number one to the next student and the last student reports the total result of this commonly executed algorithm. In both cases the duration of the algorithm is proportional to the number of students.

A faster solution

Usually the students are satisfied with these solutions, so the teacher has to make progress proposing a challenging problem: Imagine there are some hundreds of students. Are the so far discussed solutions still practical to achieve an exact solution in a very short time? As we can observe most of the time the students are inactive and wait for their turn. There should be a possibility to do it faster with an algorithm where the students are more actively involved.

Often the students propose in this situation to count the number of students for all rows of an auditorium in parallel and then add the results of the rows. This is in fact much faster although only the students at the end of the rows have additional work to do. If we assume an auditorium of size $i \times j$ there are only $i + j$ instead of $i \times j$ steps to do.

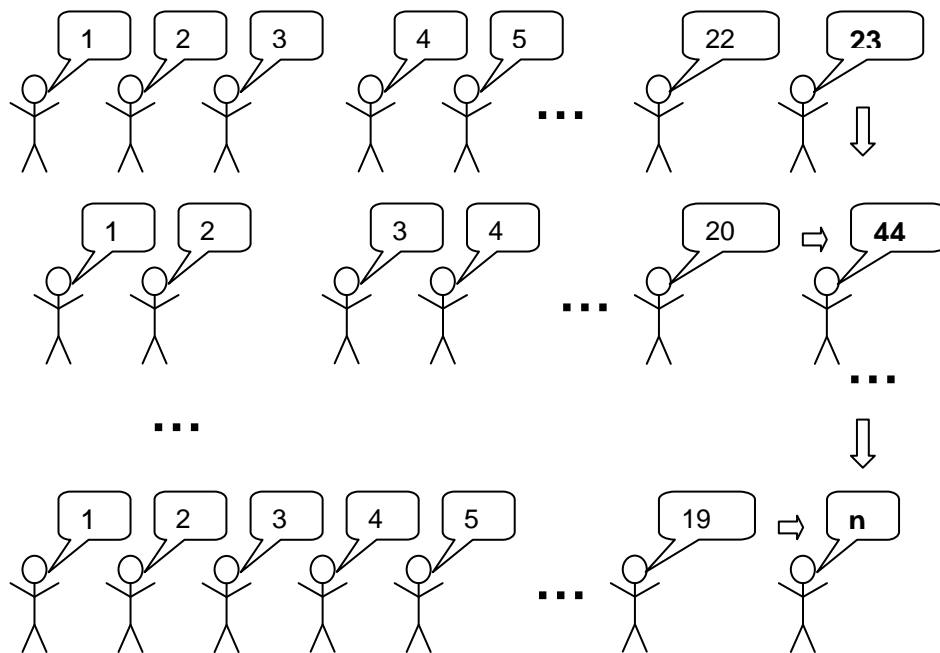


Figure 3. The students of different rows are counting independently and in parallel. The last student in a row has to add 1 (for himself) to the sum of the values from the previous student and the result from the previous rows.

The students at the end of the row have three things to do. First they have to wait until they can complete the number of students of their row. Secondly they have to wait for the number they receive from the student behind them. Thirdly they add these numbers and pass the result to the student in front. They can do these activities in this order or they may change the order of the first two actions.

Necessity of Synchronization

In practice this algorithm is not as easily performed as it looks like. Try to play this with your students! One can observe that the different rows have different speed in processing the numbers per row. So a student at the end of a row may get the result of his row before, after or

at the same time as the number of the student sitting behind him. Since he can understand and process only one of the two results at the same time, there is a need for synchronization in passing information to a student. In my experience a very good and natural way to synchronize two students is looking in the eyes of each other. Looking in each others eyes should be a prerequisite for passing a number.

Using this way of synchronization the algorithm of counting the number of students of a row can be formulated in a Logo-like pseudo-code. It depends on the sitting position of the student which one of these procedures he executes.

to beginOfRow

look at next student in row and wait until he looks at you
 pass 1 to him

end

to countInRow

look at previous student in row and wait until he looks at you
 receive number
 look at next student in row and wait until he looks at you
 pass (1 + number) to him

end

At the end of a row we have three different procedures depending on the sitting position of the student. In the last row there is no student in the back and in the first row the final result of the total number of all students is achieved.

to endOfLastRow

look at previous student in row and wait until he looks at you
 receive number
 look at student in front of you and wait until he looks at you
 pass (1 + number) to him

end

to endOfRow

look at previous student in row and wait until he looks at you
 receive number1
 look at student in back and wait until he looks at you
 receive number2
 look at student in front of you and wait until he looks at you
 pass (1 + number1 + number2) to him

end

to endOfFirstRow

look at previous student in row and wait until he looks at you
 receive number1
 look at student in back and wait until he looks at you
 receive number2
 report (1 + number1 + number2)

end

The synchronization with another student and the action that is performed during this synchronization may be described also with the following subroutine, but due to better readability for students that are not familiar with programming we abstain from using this subroutine.

to synchronizeWith :student :action

look at :student and wait until he looks at you
 run :action

end

Of course the whole algorithm can be adapted to other special arrangement of rows within a classroom or auditorium. In larger auditoria there are several rectangular blocks of rows that can calculate their numbers in parallel.

Although this version of the counting-algorithm is much faster than the first sequential algorithm, there is a huge potential for a still much faster algorithm. One can observe, as earlier mentioned, that most of the students are waiting for just two synchronizations and only once they had to add something. Only the students at the end of the row are waiting for synchronizations with 3 different students.

Counting with a “Divide & Conquer” Strategy

The students are now motivated to think about a still faster algorithm that is fast also in a very large group of people.

One idea can be that all students try repeatedly to synchronize with any other arbitrary students. When two students synchronize, one of the two students leaves the game (and sits down) the other adds the number of the leaving student to his number and starts again to synchronize with any other still active student.

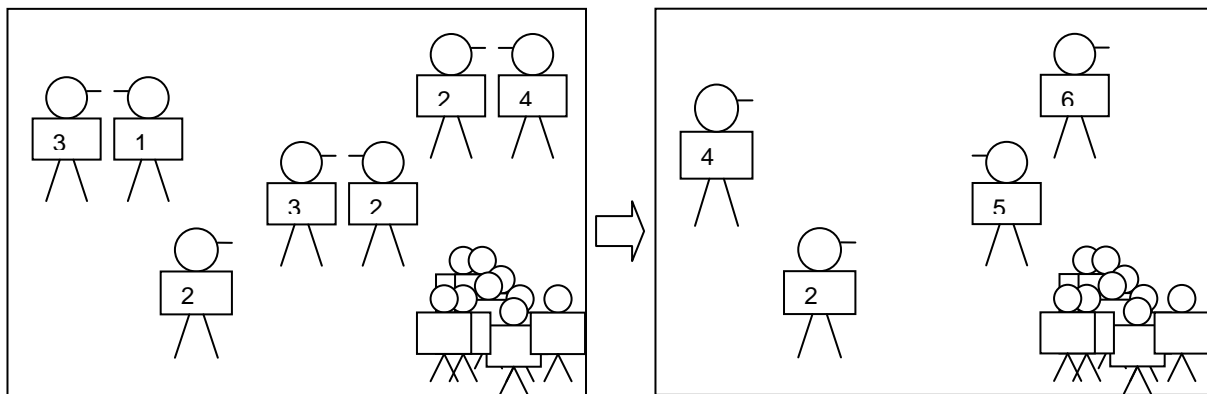


Figure 4. One step of the “Divide & Conquer” Algorithm: The students try to synchronize with a partner. One student of each pair gets inactive and sits down (lower right corner). The other adds the number of the now inactive partner to his own number.

At the beginning all students are standing and hold the number 1. At the end only one student is standing and he holds the total number of all students.

An analysis of the algorithm shows that the time effort can be $\log n$, what is in fact much more efficient than the previously presented algorithms.

The strategy used is also called “Divide and Conquer”. Each pair of students holds the numbers of two groups that represent a **division** of the students in two partitions. These two numbers are **conquered** (by adding) to yield the number of students of both groups.

Further Examples

Similar algorithmic tasks that involve all participants in the solving process are

- Distribute information material to all participants
- Determine the youngest student
- Determine the city, in which most of the students are born

In Futschek, G (2006) also a parallel version of the well-known BubbleSort algorithm is presented as a good example of a sorting algorithm that can be played by students, because the sum of the distances for walking is minimal. In general the advantages of efficient sorting

algorithms like QuickSort, HeapSort and MergeSort cannot easily be understood by playing them. They involve much more organizational work and also many exchanges of values over larger distances, that cannot efficiently be played by persons.

Good algorithmic tasks for pupils and beginners are usually not solvable in a trivial way but they must have an easy understandable problem statement. It is very important that the students invent and improve their algorithms themselves. They can do this alone or better in groups. Playing the algorithms is very important to clarify what is exactly intended by the various steps of the algorithms. Playing gives also an insight in the intermediate goals of an algorithm. This gives advantages in understanding correctness and efficiency considerations.

Summary

By detecting algorithms students can learn not only basic concepts of algorithms but also more advanced concepts like parallel programs, synchronization, divide & conquer strategy, etc.. Even complete novices are able to understand concepts of parallel algorithms, which are often easier to understand for them as sequential algorithms.

Challenging tasks with easily understandable problem statements are important to motivate students to find algorithmic solutions.

Playing algorithms is also a lot of fun for the students. Fun is motivation and motivated students have much better learning progress.

References

Futschek, G. (2006) *Algorithmic Thinking: The Key for Understanding Computer Science*, Lecture Notes in Computer Science 4226, Springer, pp. 159 - 168.

A novel didactical approach of the decision structure for novice programmers

Katerina Glezou, kglezou@di.uoa.gr

Dept of Informatics & Telecommunications, National and Kapodistrian University of Athens

Maria Grigoriadou, gregor@di.uoa.gr

Dept of Informatics & Telecommunications, National and Kapodistrian University of Athens

Abstract

This paper presents a novel didactical approach of the decision structure for novice programmers. More specifically, it introduces a pedagogic intervention, using the multimedia programming environment MicroWorlds Pro (MicroWorlds Pro 1.1 - Greek version and the incorporated Logo programming language) for the teaching of decision structure in its single (if...then...end if) and double (if ... then ... else... end if) form. The educational goal of the particular topic is to enable students to understand the concept, the features and the various forms of the decision structure, in order to be able to apply it properly.

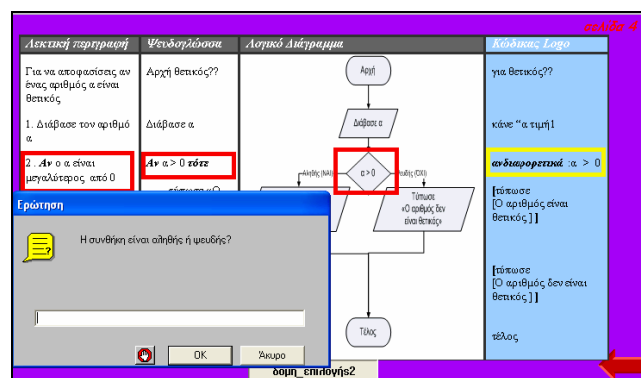


Figure 1. A specific double decision structure example: Visualization of the algorithm and Logo code's flow, and entry of the condition value

The paper's aim is to discuss the basic parameters of an effective alternative didactical approach, in the level of planning, development and implementation, as well as to support the teaching community with particular suggestions for implementation. The particular didactical approach, for which we created microworlds, lesson plans, student worksheets and additional teacher worksheets, was implemented in the framework of teaching computer programming to students (aged 14 years old) of the third grade of Junior High in Greek educational system. The students' response to the teaching process was positive. The students showed interest in their interaction with the MicroWorlds Pro environment and remained active during the lessons, especially in the phase where they had to build their own programs. The students, in many cases, were enthusiastic after being exposed to the "purely" programming part of the lesson and admitted having a "live" interaction with the environment. The findings of this trial implementation reaffirm the view that the use of an open software, like MicroWorlds Pro, which allows the development, reuse and adaptation of microworlds, as well as alternative teaching interventions, promotes the growing of creativity and pedagogical freedom of the teacher, as well as the active involvement of students.

Keywords

Didactical approach; decision structure; Logo; MicroWorlds Pro

Introduction

The international bibliography is full of various approaches for the teaching of the basic programming structures, with a double aim: to deal with the difficulties encountered by students and to ensure the active participation of the latter in the teaching-learning process (Grigoriadou et. al., 2002, Pane and Myers, 2000, Du Boulay, 1989, Soloway, 1986, Spohrer & Soloway, 1986).

The Logo programming language is a powerful tool for the development of algorithmic thought and the visualization of algorithms, especially for primary and secondary education (Glezou et al, 2005, Mikropoulos, 2004, Dagiene, 2003, diSessa et al, 1995, Harel and Papert, 1991, Papert, 1980). Logo's most important feature, which distinguishes it from the other programming languages, is its orientation as an analytical tool for thought and learning processes (Komis, 2005, Papert, 1980). Logo's ability to represent visually the program performance contributes to the understanding of the program function and smoothes out the debugging process (Papert, 1980).

The educational software MicroWorlds Pro, one of the most popular Logo-like environments, is a powerful multimedia environment for programming and an authoring tool and application environment for the development, management and exploration of microworlds.

The study we report here is part of our effort to extend our experience in designing learning environments that support learning through exploration, expression, construction and negotiation. We attempt to gain some more insight into the potential of using MicroWorlds Pro (MicroWorlds Pro 1.1 - Greek version), as a teaching tool for introducing the basic programming concepts to novice programmers in a systematic way, as well as to contribute to the discussion about the main parameters of planning, developing and implementing an effective alternative constructionist didactical approach.

In this paper we present a novel didactical approach of the decision structure in its single (if...then...end if) and double (if ... then ... else... end if) form for novice programmers.

Theoretical framework

A microworld is an incubator of knowledge since, due to its ability to simulate real world, it offers students the ability to explore a cognitive subject from the inside, aiming at the development of high level cognitive skills that can be extended in various situations (Papert, 1980). The best microworlds have an easy-to-understand set of operations that students can use to engage tasks of value to them, and in doing so, they come to understanding powerful underlying principles (diSessa, 2000). A microworld must be defined at the interface between an individual user in a social context and a software tool possessing the following five functional attributes: a) it is domain specific; b) it provides a doorway to the domain for the user by offering a simple example of the domain that is immediately understandable by the user; c) it leads to activity that can be intrinsically motivating to the user-the user wants to participate and persist at the task for same time; d) it leads to immersive activity best characterized by words such as play, inquiry, and invention; and e) it is situated in a constructivist philosophy of learning (Rieber, 2004).

Logo-like environments can be used to plan and develop microworlds, which offer students the possibility to express and exploit their thoughts, ideas and instincts and to support the process of building knowledge by creating learning environments rich in speculation and opportunities for experimentation (Noss, 1995, diSessa, 1995, Hoyles, 1995, Harel and Papert, 1991). The computational environment can function as a mental scaffolding that allows the structuring of more complex and composite commands, reinforcing, this way, the subtractive thought that progressively develops. When children try to write a program, they actually try to teach the computer how to "think", something that urges them to explore and probably get to the bottom of their own way of thinking, in order to teach it to the device (Harel and Papert, 1991).

In Ackermann's words: "A rich learning environment is one that offers the freedom for genuine exploration, reflection, expression and negotiation, while at the same time providing help and support, when needed. Needless to say, it is not easy to decide how much freedom or guidance makes for a nurturing and yet challenging learning experience. And different people need different kinds of feedback, at different times, in different situations! A good clinician, like a good teacher, is someone who masters the art of providing the "right" amount of elbowroom in each singular case" (Ackermann, 2003). Thus, the creation of interesting and demanding environments encouraging the active and constructive participation of students is a great challenge for teachers (Vosniadou, 2001).

Teaching subject: Decision Structure

The decision structure is one of the three basic logical structures of programming found in every contemporary programming environment. This structure allows the algorithm to choose the commands that are about to be executed, with respect to the verification results of a dual condition. We use it to decide between two alternative situations, one of which is true, while the other one false.

In this didactic unit, students face, for the first time, with the need to introduce the concept of the condition, the condition itself and its features. The possibility of experimenting with and exploring the various alternative routes in the program flow, according to the value of the condition, the recognition of the single and double-alternative form of the decision structure, as well as the ability of applying these forms in simple problems, are considered crucial points.

Learning Difficulties in using Decision Structure

The use of decision structure adds special cognitive difficulties in the general difficulties that have to do with understanding programming. These special difficulties (Komis, 2001) are mostly linked with the following:

- the logical content of the condition, the analyticity and exclusiveness, the logical operations of conjunction, disjunction, negation etc,
- the symbolic representations of these cases,
- the semantic and syntactic properties of the control structure in the programming language used (Du Boulay, 1989),
- the interactions with the representations of the sequential form of the task performance.

Other researchers (Ebrahimi, 1994), (Pane and Myers, 1996), (Tzimoyiannis and Georgiou, 1999) point out as the most important learning difficulties:

- the understanding of the decision structures function,
- the definition of the value (true/false) of the logical statement,
- the designation of the requisite logical statement, within the framework of a problem,
- the grouping of commands in nested decision structures.

In our teaching approach of the control structure, we should also take into consideration that the nature (endogenous: defined by the result of a calculation, or exogenous: defined by a user's interactive entry) of the conditions on which the control depends, as novice programmers encounter many difficulties in the endogenous rather than the exogenous conditions. In addition, the previous mathematical knowledge and knowledge of logic play an important role in understanding the concept of the command under certain conditions.

Research setting

The study we report here is part of a broader research on designing learning-rich settings and "objects-to-think-with" that foster exploration, expression, construction and negotiation. Our aim was to study the potential of using the multimedia programming environment MicroWorlds Pro (MicroWorlds Pro 1.1 - Greek version and the incorporated Logo programming language), as a teaching tool for introducing the basic programming concepts (variables, ways of representing

an algorithm, sequence structure, decision structure and repetition structure) to novice programmers in a systematic way, as well as to push forward some aspects of planning, developing and implementing an effective alternative constructionist didactical approach.

Firstly, we proceeded in a bibliographic research on learning difficulties recorded in relation to our studying object; we defined the cognitive obstacles and set out our teaching goals/objectives. Then, we formed a series of lessons in separate didactic units; we separated each unit in different stages and each stage in distinct steps. Afterwards, we developed a) microworlds in MicroWorlds Pro, b) activity worksheets-lesson plans, c) student worksheets and d) additional teacher worksheets, by making a formative-dynamic evaluation during their development. The research tools used were the above, as well as the Greek version of MicroWorlds Pro environment (version 1.1).

The suggested didactical approach was implemented in the framework of the didactic unit on programming in the Informatics course of the 3rd grade of Junior High of the 1st (4 classes) and 5th (2 classes) Zografou Junior High School in Athens during the school year 2004-2005. The students were separated in small groups of 2 per computer of their own choice. Due to the odd number of students in two classes, there were also two teams with three students. The participants were totally 6 classes of 18 students (10 girls - 8 boys), 19 students (11 girls - 8 boys), 20 students (11 girls - 9 boys), 21 students (12 girls - 9 boys), 22 students (13 girls - 9 boys) and 22 students (14 girls - 8 boys) respectively. The students had already been exposed to the educational software MicroWorlds Pro and had acquired a first familiarization with the environment and the basic commands of Logo language, during the lessons that had to do with the didactic unit Multimedia that preceded the introduction to programming. In addition, before the particular lesson about the decision structure (in the chapter of programming), the students had already been taught the different phases for building a program, the concept of variable, the ways of algorithm description, and had already been exposed to a first brief presentation of the three basic algorithmic structures, as well as to an analytical presentation of the sequence structure. The students' level was mostly heterogeneous. This lack of homogeneity was due to the students' different origin and linguistic competence, as well as to their cognitive level, the basic computer skills and programming skills. Apart from that, in four classes there was a significant 10% percentage of students with dyslexia and general learning difficulties that had to be handled with a lot of attention (e.g. oral examination, special support during the process of coding and debugging programs).

We collected data from the teachers' notes kept during each didactic hour, the students' microworlds, the filled in worksheets and the revision tests of the students, as well as from the semi-structured interviews of students after the conclusion of the teaching process. Then, the data underwent a qualitative analysis, whose results lead to modifications, interventions and changes in the ergonomics, the appearance and the function of microworlds, as well as in the gradual ameliorative reshaping of the worksheets and the lesson plan.

This paper presents the didactical approach for the decision structure, as it was implemented for one didactic hour. The basic investigative questions of the particular study are: a) how the learning environment is shaped during the introduction of decision structure to novice programmers and b) which are the special features of the Greek version of MicroWorlds Pro that contribute to or cause difficulty in the creation of an effective learning environment? It is a case study that uses ethnographic and action research elements.

Teaching goal

The teaching goal of the particular unit is to enable students to understand the concept, the features and the various forms of the decision structure, in order to be able to apply it properly.

Teaching objectives

In terms of knowledge and skills, the student is in position to:

- Distinguish the decision structure from the other basic algorithmic structures.

- Acknowledge the importance of introducing the single and double decision structure.
- Reproduce the two forms of the decision structure in verbal description, pseudo code and flowchart.
- Differentiate the condition in the decision structure and the different values (true/false) it can take, as well as the result according to the condition value.
- Track which commands will be executed according to the condition value.
- Collate the single and double decision structure.
- Develop algorithms and programs by applying the decision structure.

Description of the microworld

The particular microworld, which was designed for the teaching of the decision structure, is composed of 5 appropriately formed pages that attend to the stages of the process described analytically below.

In page 1 of the microworld (see Figure 2), we have a presentation of the flowcharts of the three basic programming structures (sequence, decision, repetition) in their general form. The students can execute the program that visualizes step-by-step the flowchart of the single decision structure (if ... then ... end if), assign a value to the condition in the dialog box that appears and observe the alternative program flow.

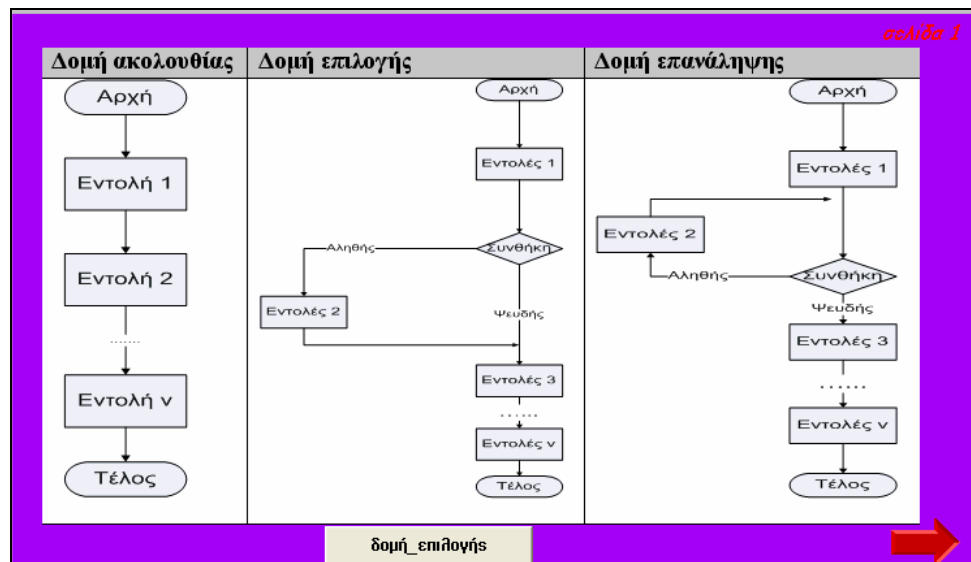


Figure 2: The flow charts of the three basic programming structures (sequence, decision, repetition) in their general form and visualization of the decision structure flowchart (Page 1 of the microworld)

In page 2 of the microworld (see Figure 3), we have a visualization of the three ways in which the algorithm can be represented (verbal description, pseudo code, flowchart), as well as of the Logo code of a specific single decision structure example, which is related with the problem described in Step 2 of the worksheet. The students can execute the program that visualizes the algorithm (in all three possible ways of its representation) and the Logo code's parallel flow of the specific single decision structure example. They can also assign a value to the condition in the dialog box that appears and observe the alternative flow of the program. The execution of this program helps the students to make a connection between the steps-commands that are used respectively in the three ways of the algorithm's representation and the code.

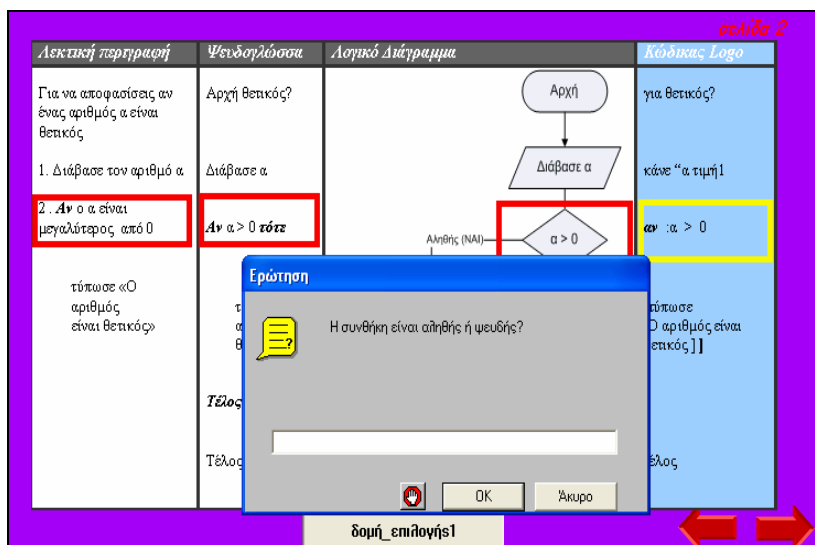


Figure 3: A specific single decision structure example: Visualization of the algorithm and Logo code's flow and entry of the condition value (Page 2 of the microworld)

In page 3 of the microworld (see Figure 4), the students are asked to assign different values to a variable A, to execute and explore the response of two readily executable programs, positive? and positive??. The students realize the impossibility of applying the single decision structure to problems that require a double branching, they are introduced to the concept of the double decision structure (if ... then ... else... end if) and recognize its valuable use.

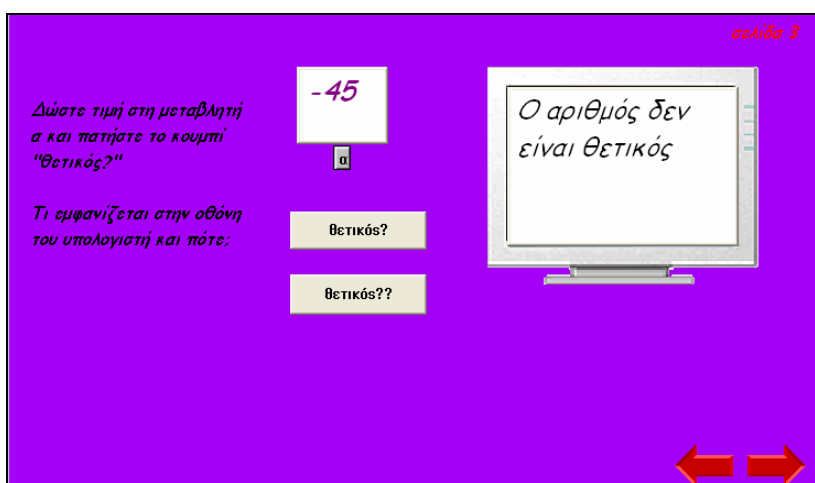


Figure 4: Response of the program "positive??" in case that the entered number is not positive (Page 3 of the microworld)

In page 4 of the microworld (see Figure 5), we have a visualization of the three ways in which the algorithm can be represented (verbal description, pseudo code, flowchart), as well as of the Logo code of a specific double decision structure example, in an analogous way with the presentation of a specific single decision structure example in page 2. The students can execute the program that visualizes the algorithm (in all three possible ways of its representation) and the code's parallel flow of the specific double decision structure example. They can also assign a value to the condition in the dialog box that appears and observe the alternative flow of the program (see Figure 1).

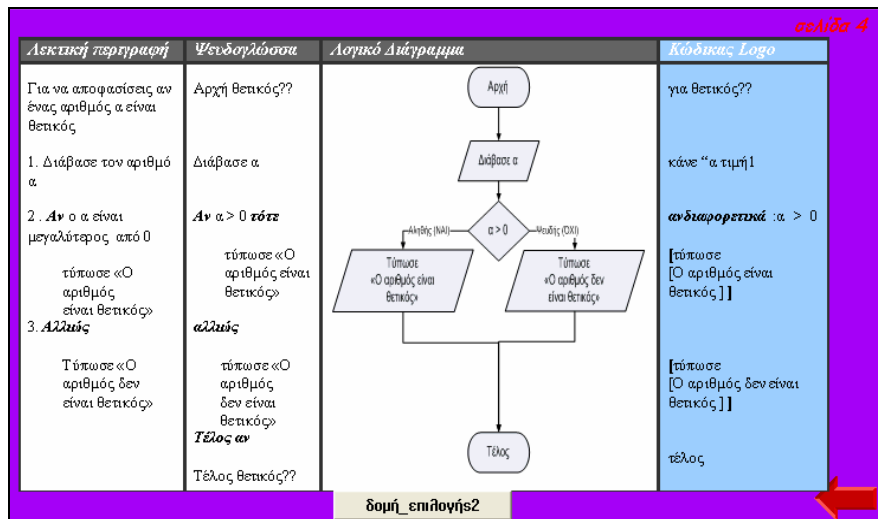


Figure 5: A specific double decision structure example: Visualization of the algorithm and Logo code's flow (Page 4 of the microworld)

In page 5 of the microworld (see Figure 6), the students assign different values to two variables a, b and explore the response of the program that they are asked to build, as described in the 8th step of the worksheet.

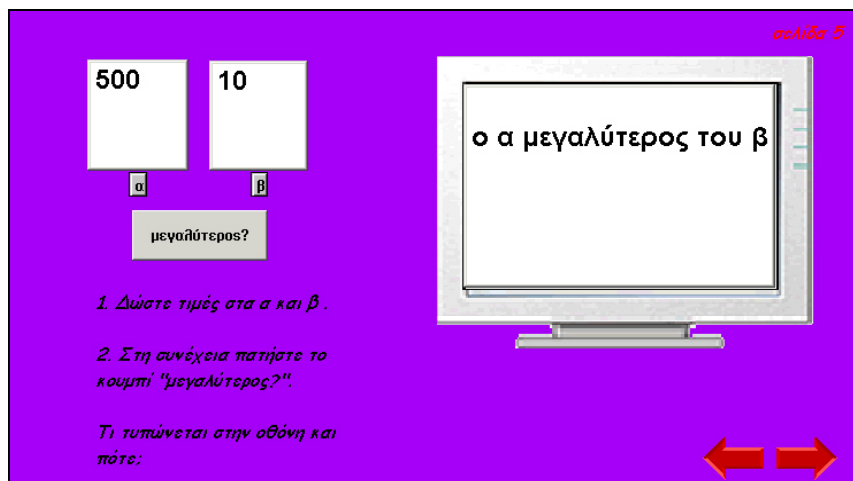


Figure 6: Response of the program "larger?" in case that the entered number a is larger than the entered number b (Page 5 of the microworld)

Lesson Plan

Stage 1 (1st & 2nd step ~5 minutes): Revocation of previous knowledge – Connection with everyday life

In the 1st step, students are asked to tangle with simple problems that include everyday life "dilemmas". On the worksheet, there is a table with statements, which include the concept of the condition, as it is used in our everyday speech, and the students are asked to recognize the condition, to assign to it a value and to predict the result that will occur in relation to this value.

In the 2nd step, the students are given a simple problem, for which they are asked to formulate a solving algorithm in the form of a verbal description. The problem, which the students are asked to solve, is the building of a program that will be able to read a number and, if the number is positive, display the message "the number is positive".

Stage 2 (3rd & 4th step ~10 minutes): Introduction to the single decision structure

In the 3rd step the students are encouraged to open the relevant microworld of MicroWorlds Pro in page 1 and execute the program that visualizes the flow of the logical chart of the single decision structure in its general form.

The students, through interaction with the environment, get acquainted with the general features of the single decision structure and give answers to the relevant questions in the worksheet.

The following are indicative questions of the worksheet: a) What values can be assigned to the condition of the decision structure? b) When are the commands2 executed? c) Are the commands 3...n always carried out despite the condition's value?

In the 4th step students go over to page 2 of the microworld and execute the program of the algorithm and code's flow visualization for the specific example that kept them busy in the previous stage. Afterwards, they answer some questions on the worksheet about the particular example, making clear, at the same time, the features of the decision structure.

Stage 3 (5th, 6th & 7th step ~10 minutes): Introduction to the double decision structure

The 5th step has to do with page 3 of the microworld and the interaction with the "positive?" program, which is given readily executable. The students are asked to give entries of different numbers (positive, negative, zero) in the program and record the program's response, as shown in the relative table of their worksheet. The aim of this step is to introduce to students the concept of the decision structure in its double form (if ... then ... else... end If) and make them recognize the necessity of its use. After that, we have the 6th step's introductive question, which has to do with the response of the program "positive?". If the inserted number is not positive, the students are faced with the problem of what we would do if we wanted an equivalent message to be displayed in this case.

In the 6th step students execute the program "positive??" that has to do with the solving of a new problem, as it was posed in the previous question, and they are asked to make a deduction of the respective algorithm's verbal description from the result.

In the 7th step the students are asked to execute a specific example of a program using the double decision structure and then answer questions that focus on the structure's features and its difference with the single decision structure form that was presented in a previous phase. The students have to track the common features between the two forms and answer the questions on the worksheet. Some indicative questions are the following: a) How many conditions do we have in this structure (double decision structure)? b) Which values can the condition take here? c) How many junctions do we have in the double decision structure? d) Which are the commands executed in this form and when are they not executed? e) What is the difference between this form and the previous one? When do we use it?

Stage 4 (8th step ~15 minutes): Experimentation - Practice - Feedback

In the 8th step the students are asked to build a program of an analogous degree of difficulty as the previous one, a program which will be able to read two numbers a, b and display their sum if $a > b$, or else to display their difference.

This problem requires the use of the double decision structure and makes students participate actively in the solving process. Firstly, the train of thought for solving the problem is discussed in the classroom. The algorithm of the problem is given in the form of verbal description on the worksheet. Then, the students, after consulting the microworld, have to translate the algorithm given in a pseudo code and a flowchart and afterwards develop the Logo code, while experimenting in the microworld's environment.

Findings

From the pilot study's data analysis (we present only indicative facts here) we should stress the following:

The students' response to the teaching process was positive. The students remained active throughout the entire course and showed increased interest as far as the interaction with the MicroWorlds Pro environment was concerned, especially when it came to building their own programs. Students that initially didn't show interest in the Informatics course participated actively and showed great concentration in their cooperation with other students. There were five groups that were not motivated, and cooperated only fragmentarily.

In stage 1, the students recognized easily the section of the statements that had to do with the condition, and were easily able to assign a value (false or true), though they seemed to have difficulty in defining the respective result. Figure 7 presents indicative student answers in the table of the worksheet containing statements of everyday life that include a condition in the 1st step.

Statement	Condition	Value (True/False)	Result
If it rains, I will take an umbrella.	If it rains	False	I will not take an umbrella.
If it is sunny, I will wear sunglasses.	If it is sunny	True	I will wear sunglasses.
If I study, I will write well in the test.	If I study	False	I will write well in the test.

Figure 7: Indicative student answers on the worksheet in the table of the worksheet containing statements of everyday life that include a condition

The students, while facing the problem that introduces the condition, realized that they can't use the sequence structure and recognized naturally the need to introduce a new structure as a tool to solve problems. The students solve a simple problem, where they naturally and spontaneously introduce the word if, which constitutes the key word of the decision structure.

In stage 2, the students, while interacting with the environment, were particularly motivated as they assigned themselves the value of the condition that they select, determining this way the diagram's flow.

In stage 3, the indicative answers of a student's worksheet as far as the response of the program "positive?" is concerned in step 5 are presented in Figure 8.

	$a > 0$	$a < 0$	$a = 0$
Program's response	It prints the message "the number is positive".	It prints nothing.	It prints nothing.

Figure 8: Indicative answers of a student's worksheet during step 5

In stage 4, the students were enthusiastic after being exposed to the "purely" programming part of the lesson, and admitted having a "live" interaction with the environment. When solving the problem, the students handled with great ease the analysis of the problem, the decision of the appropriate structure, as well as the transfer of the algorithm in its three representations.

The repetitive shift between the code analysis and its reforming/extension, which allows the transfer from simple to complex and the gradual familiarization with the programming language

within the scaffolding process, has proved to be equally effective. During the process of developing Logo code and of debugging, the questions posed by the students were often fixed and imperative. The teacher had to proceed in subtle handlings, in order to accommodate the students with no more support than that they need.

We observed an increased difficulty due to the students' lack of familiarity with the syntactic rules of Logo, often leading to disappointment and urge to quit. The following mistakes were observed on a regular basis: the students left no space between the operators, in the process "name" they left a space between two words (e.g. greater 1 instead of greater1), they used the letter "o" instead of 0 (zero), they omitted semicolons before the variable, or forgot to put the word end at the end of the procedure definition.

Although the study was conducted without a control group, the analysis of student answers on the worksheet revealed that (a) students made significant gains in their ability to answer test items covering single and double decision structure, conditions, condition value and program flow; (b) students in all ability levels showed gains in programming skills.

Follow-up interviews of students revealed that experience with the approach was associated with a reduction in anxiety toward programming, greater willingness to see programming as relevant to everyday life and increased willingness to approach programming challenges with a positive attitude.

Discussion

This pilot implementation in actual classroom circumstances, and, as a matter of fact, in different classes in successive days, offered us important feedback that lead to modifications/interventions to the ergonomics, the appearance and the function of microworlds, as well as in the gradual ameliorative reshaping of the worksheets and of the lesson plan.

The determination of the subject, the content and the functional requirements of the microworld, as well as the creation of the worksheet require deep thought, patience and persistence. The planning of the microworld and of the worksheet defines the setting and affects both the role of the instructor and the students during the teaching-learning process. The application of a shaping-dynamic assessment is necessary and efficient during the development and application circle of both the microworld and the worksheet.

The findings of this trial implementation reaffirm the view that the use of an open software, like MicroWorlds Pro, that allows the development, reuse and adaptation of microworlds, as well as alternative teaching interventions, promotes the growing of creativity and pedagogical freedom of the teacher as well as the active involvement of students.

We need a further analysis in order to generalize the conclusions concerning the features of an effective alternative teaching suggestion, in the level of planning, development and implementation, which responds to the special characteristics of the students of the particular class and of different classes as well.

References

- Ackermann, E. (2003). *Hidden Drivers of Pedagogic Transactions: Teachers as Clinicians and Designers*. In Proceedings of Eurologo 2003. Edited by Cnotinfor, Lda. Porto, August.29-37.
- Brusilovsky, P., Calabrese, E., Hvorecky, J., Kouchnirenko, A., & Miller, P. (1997) *Minilanguages: A Way to Learn Programming Principles*, Education and Information Technologies, 2(1), 65-83.
- Dagiene, V. (2003) *A set of Logo problems for learning algorithms*. In Proceedings of Eurologo 2003. Edited by Cnotinfor, Lda. Porto, August.168-177.
- diSessa, A. (2000) *Changing minds: Computers, learning, and literacy*. Cambridge, MA: MIT Press.

- diSessa, A., Hoyles, C., Noss, R. (1995) *Computers and Exploratory Learning*. Springer - Verlag, Berlin Heidelberg.
- Du Boulay, B. (1989) *Some difficulties of learning to program*. In Studying the Novice Programmer, E. Soloway & J. C. Spohrer (eds), Lawrence Erlbaum Associates, Hillsdale, NJ, 283-299.
- Ebrahimi, A. (1994) *Novice programmer errors: language constructs and plan composition*, International Journal of Human-Computer Studies, 41, 457-480.
- Glezou, K., Stamouli, E., Grigoriadou, M. (2005) *An alternative teaching approach of the decision structure for novice programmers by use of MicroWorlds Pro*. In Proceedings of the 3rd Panhellenic Conference "Teaching Informatics". Edited by Tzimoyiannis, A., Korinthos, October http://www.etpe.gr/uploads1/paper_s33.pdf (In Greek)
- Grigoriadou M., Gogoulou A., Gouli E. (2002) *Alternative teaching approaches in introductory programming lessons: teaching suggestions*. In Proceedings of the 3rd Panhellenic (International) Conference "The Information and Communication Technologies in Education", Edited by A. Dimitrakopoulou (Rev.), Rhodes, September, Volume A', 239-248. (In Greek)
- Harel, I. and Papert, S. (1991). *Constructionism*. US: Ablex Publishing Corporation.
- Komis, B. (2005) *Introduction in teaching informatics*. Kleidarithmos Publications, Athens. (In Greek)
- Komis, B. (2001) *A Study of the Basic Programming Concepts in the framework of a Constructive Teaching*, Themes in Education, 2(2-3), 243-270. (In Greek)
- Mikropoulos, A. (2004) *Does Logo have a place as a cognitive subject and hollistic model in compulsory education?* In Proceedings of the Two-day International meeting "Teaching Informatics", Volos. 65-72. (In Greek)
- Pane, J. and Myers, B. (1996) *Usability Issues in the Design of Novice Programming Systems*, Technical Report (CMU-CS-96-132), School of Computer Science, Carnegie Mellon University.
- Pane, J. and Myers, B. (2000) *The Influence of the Psychology of Programming on a Language Design: Project Status Report*. In Proceedings of the 12th Annual Meeting of the Psychology of Programmers Interest Group, Edizioni Memoria, Italy. 193-205.
- Papert, S. (1980) *Mindstorms - Children, Computers and Powerful Ideas*. Basic Books, New York.
- Rieber, L.P. (2004) *Microworlds*. In Handbook of research for educational communications and technology (2nd ed.), D. Jonassen (Ed.), Mahwah, NJ: Lawrence Erlbaum Associates, 583-603.
- Soloway, E. (1986) *Learning to program = Learning to construct mechanisms and explanations*, Communications of the ACM, 29(9), 850-858.
- Spohrer, J. and Soloway, E. (1986) *Alternatives to Construct-Based Programming Misconceptions*. In Proceedings of CHI '86, 183-191.
- Tzimoyiannis, A. and Georgiou, B. (1999) *Secondary education students' difficulties on the application of decision structure for the development of algorithms. A case study*. In Proceedings of the Panhellenic Conference «Computer science and Education», Computer Instructors Association of Hepirus, A. Tzimoyiannis (Rev.), Ioannina, 183-192. (In Greek)
- Vosniadou, S. (2001) *How children learn*, Educational Practices Series, n°7, <http://www.ibe.unesco.org/International/Publications/EducationalPractices/prachome.htm>

Practical Logo Team Competitions

Irina Kuznetsova, *info640@spb.edu.ru*

Sophia Gorlitskaya, *sophiag@yandex.ru*

Saint-Petersburg City Palace of Youth "Anichkov Palace", <http://anichkov.testing.spb.ru/>,
Secondary School 640 of general education, Saint-Petersburg, Russia

Abstract

A school team programming contest is, on the one hand, an exciting game that allows us to develop and apply our intellectual and team-working skills. On the other hand, it is a very efficient way to improve our understanding of the best ways to teach programming and technology in schools.

School and university programming contests have a long lasting tradition in Russia. For many years, Russian teams successfully participate in international programming competitions.

In 1996-1998 Saint-Petersburg Gymnasium 470 organized annual summer schools for students interested in programming and computer science in general. The summer schools welcomed students from Saint-Petersburg, Murmansk, Syktyvkar, Moscow, and other cities. In the course of the summer schools, Logo was proposed as a primary programming language.

In organizing the summer schools, we have been trying to follow the principle formulated by Seymour Papert [1]: to provide students with an environment to realize their ideas, without putting rigid bounds on their fantasy. While a traditional course in informatics was not able to give students enough time and freedom to implement their algorithmic ideas, our summer schools allowed every participant to develop her own systems project and to gain enough background in Logo programming to work independently.

Using the experience we gained in organizing the summer schools, in 1997 we launched Team Olympiads in Practical Logo Programming.

Many Russian schools accepted Logo as the first language for teaching programming. In 2005-2006, the Ministry of Education of Russian Federation officially recommended installing Logo systems in schools, starting from MicroWorlds 2.0. Nevertheless, many teachers of informatics still considered Logo as a programming language bound to encoding simple commands for controlling the Turtle and various peripheral devices. It is often ignored that Logo enables all modern programming techniques, and environments like MicroWorlds 2.0 and especially MicroWorlds 3.0 provide instruments for a wide spectrum of various scientific studies and modeling. Moreover, these instruments are accessible for students of all levels and backgrounds: from primary school kids to doctors of sciences.

In this paper, we summarize the ten years experience we earned while organizing Annual Team Competitions in Practical Logo Programming (Team Olympiads) for students of 5th-8th grades that took place in Saint-Petersburg, Russia in 1997-2007.

We classify around 70 problems that were designed for the Olympiads, and we discuss important features related to formulating and solving Olympiad problems in Logo. We show how these features highlight principal difficulties that arise in teaching the theory and practice of computer science and technology in primary and secondary schools. By presenting examples of Olympiad problems, we also demonstrate the impressive diversity of technological methods that can be applied in the Logo environment.

Keywords

Logo, school informatics, pedagogical aspects in programming, object approach in programming, computer modeling

Introduction

In 1996-1998 Saint-Petersburg Gymnasium 470 organized annual summer schools for students interested in programming and computer science in general. The summer schools welcomed students from Saint-Petersburg, Murmansk, Syktyvkar, Moscow, and other cities. In the course of the summer schools, Logo was proposed as a primary programming language.

Up-to-date Logo environments are characterized by the following features:

- *Object approach.* Object approach is employed (implicitly or explicitly) in all Logo products: in designing her projects, a user is allowed to elaborate built-in Logo objects, by changing its properties and extending their functionality. As we observed, a positive background in Logo environment helps students successfully apply object and system approach in exploring other programming packages.
- *Interactivity.* A logo project involves a number of *personages* – interacting entities whose behavior is programmed by the user. This turns programming into a difficult but exciting process which efficiently develops algorithmic skills and system way of thinking.
- *Multimedia support.* The use of audio and video effects increases the expressive power of Logo projects.

All these features turn Logo into an excellent environment for scientific studies in schools. That is why we chose Logo as a principal language for our programming contests.

Many Russian schools accepted Logo as the first language for teaching programming. In 2005-2006, the Ministry of Education of Russian Federation officially recommended installing Logo systems in schools, starting from MicroWorlds 2.0. Nevertheless, many teachers of informatics still considered Logo as a programming language bound to encoding simple commands for controlling the Turtle and various peripheral devices. It is often ignored that Logo enables all modern programming techniques, and environments like MicroWorlds 2.0 and especially MicroWorlds 3.0 provide instruments for a wide spectrum of various scientific studies and modeling. Moreover, these instruments are accessible for students of all levels and backgrounds: from primary school kids to doctors of sciences.

In this paper, we summarize the ten years experience we earned while organizing Annual Team Competitions in Logo programming for students of 5-8 grades that took place in Saint-Petersburg, Russia in 1997-2007.

The paper is organized as follows. First we specify the goals and organizational principles that we put at the basis of our Olympiads. Then we describe our methodology in formulating Olympiad problems and defining formats of the problem solutions. Next we propose a rough classification of the Olympiad problems and highlight the principal obstacles that arise in understanding theoretical and practical aspects of computer science. We conclude the paper with general observations about the optimal methodologies in addressing Logo modeling problems.

Goals and Organizational Principles of Olympiads

As the organizers of the Olympiads, we were primarily aiming at:

- demonstrating the potential of Logo in enabling students of all ages and backgrounds to study and model processes in mathematics, physics, and other sciences
- promoting Logo, as the best educational environment for setting up computer experiments as well as for teaching programming to primary and secondary school students
- advertising programming itself, as an exciting kind of intellectual activity that helps bridging the generation gap in families and schools [1,4]

- identifying students gifted in programming
- establishing informal links between organizations - participants of the Olympiads

Initially, we proposed two editions of Logo as the basic environment in our Olympiads: LogoWriter 3.2 and LogoMiry 2.0 (Russian version of MicroWorlds 2.0 produced by the Moscow Institute of New Educational Technologies). Faced with the increasing popularity of MicroWorlds 2.0 and MicroWorldsPro (LCSI), we decided later to stop using LogoWriter 3.2. However, because of some inconsistencies in released version of LogoMiry 2.0, we had to maintain both options for the Olympiad problems: LogoMiry 2.0 and MicroWorldsPro.

Since 2004 Logo Olympiads are organized in two modes: on-site and remote. In the remote mode, the registered participants first receive the set of basic instructions. Then the problems themselves are published on-line at the pre-determined time.

In 2005, we designed a web interface that allows users to run Logo applications on the computers that do not have Logo environment installed. The interface is based on the freeware MicroWorlds Web Player (LCSI). However, certain restrictions imposed by the web interface complicated the organization format of our Olympiads. As a result, later we decided so far not to use the web interface in the competitions.

The basic principles we have in mind in selecting the Olympiad problems are the following:

- The problems expect every Olympiad participant to actively work for at least 2 hours. The expected amount of work should not be less than what the best participants can potentially perform in the given time frame.
- The very same problems should be accessible for participants of 2-6 grades as well as for participants of 7-9 grades [2].

After the Olympiad a document summarizing the problems and their solutions is published at <http://logo.dtu.spb.ru>. Teachers can use this document in their classes. In the future, we prospect organizing specialized forums and seminars on the lessons learnt from the analysis of the problems and the proposed solutions.

Defining Problem Statements and Solution Formats

In this section, we summarize the principles we chose to formulate an Olympiad problem and to define an expected format of the problem solutions.

Using Plain Text

Every problem is stated in natural language text. If necessary, a figure or an animated image may be used to illustrate the problem.

In recent years, many teachers point out the frightening decline in the ability of school students to read and understand written text, preempted by video and audio information. The reading comprehension skills remain however necessary for adequate reception of the cultural inheritance of human kind, not mentioning obtaining higher education or high-qualified job.

We deliberately intend to maintain and improve the consistent reading skills of our students. To facilitate the problem perception, the problems proposed for every Olympiad are unified by a common *story*.

For example, in 2004, the Olympiad participants were expected to solve the imaginary problem of surviving on an unpopulated island.

In 2005, all problems were on studying the motion of a given object (Turtle). A particular class of problems involved the formalism of Johan Gielis [3] that describe simple geometrical objects (a circle, a triangle, a rectangle, etc.) as well as more complicated images resembling natural silhouettes such as starfish, snail, etc.

In 2006 all problems were related to the Magic Market where participants met magicians, mysterious animals, dwarfs, etc. In 2007 every participant was in search of the Treasure hidden by the pirates on an island in the Caribbean Sea (see an example in Figure 1).

The problems are presented on a web server. All information related to the Olympiad remains available long after the Olympiad.

Variants of Solutions

The solutions can be presented in either of the following ways:

- a) Natural language text
- b) A sequence of annotated commands for Logo command center
- c) An annotated complete program in Logo Procedures Sheet.

Each of these ways brings certain number of credit points.

The participants are encouraged to submit extensive explanations of the ideas of their solutions, in plain text or comments to their programs.

A solution presented in way (c) is considered complete if it contains a description of how the Logo programs are called.

An Example Problem

Three ships (Fla, Floo, and Chood) are located in different harbors (see the figure). The captains of the ships come to know about the Treasure hidden on a little island in the Caribbean Sea. The ships simultaneously started sailing and simultaneously arrived at the island. Every ship was sailing directly to the island with a constant speed that did not exceed 5 steps per time tick.

- (1) *For each ship, determine its maximal allowed speed and the initial distance from its harbor to the island.*
- (2) *Write commands of motion for each ship.*
- (3) *Write procedures of motion for each ship and encompass the procedures in a program.*

Obviously, students of different ages and backgrounds comprehend the problem statement differently. This will be reflected in the proposed solutions. That is why the solutions are evaluated by taking into account the ages of the participants and their abilities to handle the Logo environment.

On the other hand, this is an arithmetic problem, and its solution can be expressed using an arithmetic approach taught in 5th and 6th school grades.



Solution (a). Let s_1 be the distance between ship Fla and the Island, s_2 be the distance between ship Floo and the Island, and s_3 be the distance between ship Chood and the Island. Let s_1 be larger than both s_2 and s_3 . Since Fla was initially more far away from the Island, it should have had the maximal allowed speed, namely 5 steps per time tick. Thus Fla needed $t_1 = s_1 / 5$ time ticks to reach the Island. Since the ships reached the Island simultaneously, the speeds of Floo and Chood are, respectively: $v_2 = s_2 / t_1$, $v_3 = s_3 / t_1$.

This solution demonstrates that the participant fully understands the problem and knows how to solve it. If he has enough background in Logo programming, he can complement the solution with the following commands:

Solution (b).

Fla, show distance "Island ; distance between Fla and Island

Chood, show distance "Island ; distance between Chood and Island

Floo, show distance "Island ; distance between Floo and Island

These commands return the required distances. The solution demonstrates that the participant is able to work with numbers and perform computations in Logo.

Participants with programming skills may define the parallel processes describing the motions of the three objects. To make the program more general, the solutions should use variables. For example:

Solution (c).

to RASST

Island,

make "sfla distance "Fla ; distance between Fla and Island

make "sfloo distance "Floo ; between Floo and Island

make "schood distance "Chood ; between Chood and Island

end

** Computing the speeds **

to Speeded :fla :floo :chood

make "v_flu 5 ; speed of Fla

make "time :sfla / :v_flu ; time for Fla

make "v_fli :sfloo / :time ; speed of Floo

make "v_chood :schood / :time ; speed of Chood

end

** Programming the ships' motion **

to plavv :v_chud :v_floo :v_flu

Chood, towards "Island repeat :time [fd :v_chood wait 1]

Floo, towards "Island repeat :time [fd :v_floo wait 1]

Fla, towards "Island repeat :time [fd :v_flu wait 1]

end

** Main program **

to index

task1

RASST

SPEED :fla :floo :chud

plavv :v_chood :v_floo :v_flu

end

Returning to the Initial State and Saving the Results

The problems can be tackled in an arbitrary order. Moreover, it is recommended to start solving the problems that are feasible to solve in the given 1.5-2 hours, depending on backgrounds and ages of participants.

Therefore we propose to present the solutions in a certain way which allows us to easily run every task. Once a new task is run, the objects of the previous task should be removed. To

ensure this, we developed special standard procedures for launching tasks, as well as procedures for returning to the initial state.

In the first Olympiads, solutions proposed by the participants were often lost because the procedures of submitting and saving the results were not standardized. Of course, the organizers and coordinators of the Olympiads were trying to help all the participants, especially those of younger ages, to save their solutions in a stable storage, but this did not always work. This is the reason why, since 2003, we include specialized programs for automatic saving of the results in the standard package of Olympiad problems.

In 2006 and 2007 the results we proposed an option to save results in text files. Each time a package with Olympiad problems is accessed, the system automatically creates a collection of files associated with the team name and the current version.

Nevertheless, each participant is expected to consider saving her results as a must. More precisely, she should periodically paste necessary information in a special text window and then click the Save button. As we already mentioned, the name of the file is defined automatically.

As a result, in 2007 no result losses were observed. The use of text format allowed us to keep the amount of submitted data reasonably small so that the Olympiad jury could easily handle it.

Problem Classification

Analyzing the results of our Olympiads allowed us to classify the Olympiad problems according to their difficulties and factor out the principal obstacles that arise in understanding theoretical and practical aspects of computer science.

In choosing the problems we particularly aim at *surprising* the participants. We do not propose similar problems. The problems do not always expect the participants to write programs applying Logo commands. The main requirement we impose is to be able to read and understand the posed problems. Some problems were borrowed from published collections of mathematical and algorithmic problems and adapted to the Logo environment.

Problems on Native Wit and Logic

These problems expect the ability to read comprehensively, to think logically, and to express the corresponding results. Such problems are typically chosen from collections of entertaining problems and proposed for the younger participants.

Having solved such a problem, the participants are usually expected to express the solution in Logo: write the text and save the project.

Mathematical Problems

Many problems assume the ability to work with numbers and to apply basic mathematical skills. These problems are usually proposed to the participants of 5th and 6th grades.

We put a particular emphasis on problems in geometry, since these problems usually admit elegant solutions in the Logo environment.

If a participant does not have necessary mathematical background, she still can handle the given problem in the Logo environment, maybe even guessing the answers. Interestingly, even this approach is very useful, since it brings new understanding of mathematics and computer science.

Modelling and Measurements

Problems that involve various kinds of measurements and selecting parameters of proposed models are of special interest in our Olympiads. The problems expect the participants to measure the parameters of certain objects, perform certain calculations, and apply the results of these calculations in practice. A problem of this class typically includes an image or an animation that facilitate understanding the problem statement. The participants are also provided with the procedure that created the animation. This procedure can be used for finding a solution to the

problem. We are convinced that reading and analyzing programs written by others is an important element in learning how to program. The solution to the problem typically involves measuring the object parameters and undertaking calculations based on the results of the measurements. All these operations are performed using Logo commands.

Problems on Text Processing

The problems related to managing texts, such as converting texts from one format to another and searching the text for a given pattern, are particularly interesting. The younger participants improve their knowledge of the language and invent algorithms for locating given sequences in texts. The elder participants are expected to also apply these algorithms in practice.

Algorithmic Problems

The Logo environment allows the students to learn basic algorithmic abstractions: linear sequences of actions, cycles, if-then and case constructions. The environment maintains the use of global and local variables, parallel processing, semaphores, etc. In our Olympiad problems, we encourage participants to learn how to use all these constructions. The use of variables facilitates the programming process and improves the quality of results. The use of parallel processes is helpful in modeling simultaneous actions of several objects.

Concluding Remarks

Initially, our Olympiads were intended to advertise the Logo environment and to improve the Logo programming skills of the participants of the Olympiads. However, we observed an interesting phenomenon: faced with a problem, many participants do not make an effort to understand what the problem requires exactly. Instead, they often replace the given problem with another problem that they previously studied in school. We thus came to the conclusion that it is particularly important to express the problems in the most concise and unambiguous way. To prepare for an Olympiad, a student should train her skills in factoring out the most important part of a given problem, its idea.

Further, we expect the following methodology to be optimal in addressing a Logo modeling problem. The first step would be to think of how to set up an experiment in Logo. For this, it would be helpful to describe the experiment: the setting typically includes Logo objects and commands (programs) that manage these objects and report the measured parameters. Clearly, the experimental setting (the model) should be portable to any machine on which the corresponding version of Logo is installed. Once the setting is prepared, it can be used for experimental studies. A conclusion drawn from the results of the experiments is often considered as the solution to the problem.

For students with a positive programming background, we foresee the following elements to be important to successfully participate in our Olympiads:

- The use of variables in algorithms.
- The use of *list* variables for elegant programming constructions, such as cycles.
- The use of selection algorithms, such as locating a list element with a given identifier or a list element satisfying a given criterion.

We would like to emphasize our intention to use the Logo environment primarily for computer modeling, and not necessarily for solving problems in a theoretical manner which is often required in traditional programming contests.

To conclude, we would like to cite Nicholas Negroponte who wrote in his preface to Seymour Papert's book "Connection Family":

«Kids bring a new culture to the family landscape, a culture which has at its core the extremes of being simultaneously personal and global. Children understand computers because they can control them. They love them because they can make their own windows of interest» [5].

References

1. Seymour Papert, Mindstorms: Children, Computers and Powerful Ideas, Dsic Books, Inc. Publishers / New York, 1989
2. Abramova G.S., The Age Psychology, Textbook for University students, Moscow, Akadem A, 1998 (in Russian)
3. Johan Gielis, «A generic geometric transformation that unifies a wide range of natural and abstract shapes», a review of the article can be found at <http://astronomy.swin.edu.au/~pbourke/curves/>, 2002
4. Seymour Papert, The Connected Family - Bridging the digital generation gap, Longstreet Press, Atlanta, Georgia, USA
5. Nicholas Negroponte, Being digital, Alfred A. Knopf New York, 1995

Disguised Programming as a Teaching Aid for Students with Special Needs

Paulo C. M. Guerreiro, *paulocmguerreiro@gmail.com*

DEEI-FCT, Universidade do Algarve, Campus de Gambelas, 8000 Faro, Portugal

Vítor M. M. Vieira, *viktor.vieira@gmail.com*

DEEI-FCT, Universidade do Algarve, Campus de Gambelas, 8000 Faro, Portugal

Fernando G. Lobo, *flobo@ualg.pt*

DEEI-FCT, Universidade do Algarve, Campus de Gambelas, 8000 Faro, Portugal

Abstract

This paper describes a project that uses constructionist technologies in the development of a new approach for teaching basic programming skills with the Logo language for students with special needs.

The necessity and motivation of the learning process has been studied, discussed, and stimulated for a long time. But it has normally been directed to “normal” children, while neglecting those with special learning needs. Meanwhile, one finds a growing number of students with special educational needs. Schools are now better equipped (computer wise) than ever before, but are not taking advantage of these equipments to better teach their students.

Our aim with this research project is to motivate and teach basic programming skills for students with learning difficulties. Under this context, we compare two different approaches for reaching the proposed goal, one based on LEGO’s Robotics Invention System v2.0 (RIS) and another based on an application developed on purpose by ourselves.

Keeping in mind that we’re searching for a tool that children can use, even those with learning disabilities, we need an application that is alluring and easy to use. LEGO’s Robotics Invention System v2.0 (RIS) is an example of such an application. With RIS, one creates programs simply by dragging blocks of code into the program. Together with this simple interface, we have the MindStorms Robot itself, which provides additional motivation.

Not finding another application to suit our needs, we took upon ourselves the development of a new one. This new application was entirely written with the Logo language, using LCSI MicroWords EXTM Robotics Edition. As we propose to teach basic programming skills, something often considered dull and complex, we decided that the best approach resided in a sort of game, which is something most children find appealing. Like any good game we needed an interface that was intuitive and easy to use. But more than that, we needed it to be versatile; something that could be molded to our needs.

After the new application was up and running, time came to evaluate which of the two tools provided a better motivational and learning experience. For this, we requested the collaboration of two groups of students, each using one of the applications. Performing similar tasks, like going from point A to point B or drawing geometric figures, we can observe and analyze the motivation of each group, while they use both applications.

Keywords

Logo programming, LEGO MindStorms, Programming in Disguise

Introduction

The necessity and motivation of the learning process has been studied, discussed, and stimulated for a long time. But it has normally been directed to “normal” children, while neglecting those with special learning needs.

Along with the development of new learning motivation techniques, the way to teach has also been submitted to significant changes, either due to the creation of new technologies, like portable projectors or the Smart Board that allow us to present lectures in a more dynamic and interactive way, or due to changes in the way the teaching-learning process is made to work. At the centre of the later approach, we can find the constructionist learning environment as advocated by Seymour Papert’s theories (Papert, 1993; Papert 1996) where the formal roles of teacher and student loose their meaning. The teacher abandons his place as the rigid source of knowledge and assumes the role of a guide, steering the student in the right direction. On the other hand, the student ceases to be a mere receptor of knowledge and gains an active role in the acquisition and construction of his new knowledge.

Nowadays there’s a growing number of the so called *Special Educational Needs* (SEN), students. These are children that are affected by diverse factors which interfere with their cognitive capacities, such as Attention Deficit Disorder (A.D.D.), psychological disorders and even cognitive impairments. SEN students should receive special integration and evaluation conditions and be inserted into classrooms with fewer students to provide a better and more personalized teaching. However, mostly by the lack of resources (human or otherwise), these students end up in classrooms that just cannot respond to their special needs.

At the same time, schools are better equipped (computer wise) than ever before and yet these resources are not being used to help the students who struggle in their academic progression.

Taking all of these facts into consideration, we test two different applications that can be used to surpass the difficulties presented by the SEN students. We developed a Logo application called “*A Minha Viagem*” (My Trip), which is a sort of game that can be used to help the students construct their own knowledge. This application is intended to teach basic programming skills in the Logo language and the Logo language itself. At the same time we use LEGO’s MindStorms and Robotics Invention System v2.0, trying to find which of the applications provides a greater motivation source for two groups of SEN students.

The main motivation behind this work is the search for good applications and/or techniques to facilitate the teaching of basic programming skills, particularly to SEN students.

In the following sections we describe the steps taken to setup the experiment, what was taken into consideration in the development of the new application, the progress of the test groups, and the conclusions we were able to infer based on this study.

Setting up the Experiment

Translating LEGO’s Robotics Inventions System v2.0 (RIS)

Since we’re going to work with Portuguese students, the RIS’ interface language needed to be in Portuguese. This was achieved by translating the strings in key text files that contained all the words and phrases (warnings, messages, menu options, buttons captions, among others) used by RIS. The end result can be seen in Figure 1.

The main difficulty of this task is presented by the fact that one word may differ its meaning given different contexts. Nonetheless, we were able to complete the translation without compromising the original meaning of the words/phrases.

Since we constantly needed to shift between the Portuguese and English version of the RIS, we created a patch to this end.



Figure 1. LEGO's Robotics Invention System v2.0 with the Portuguese translation.

For the same reason behind RIS' translation, there was also the need to encapsulate the Logo functions and error messages needed to perform the experiment.

Encapsulating MicroWorlds EX™ Robotics Edition Logo Language

To follow the intuitive syntax of the Logo language, we felt the need to analyse the proper way to translate the English functions to their equivalent in Portuguese. However, in order to better visualize the turtle's movement, we not only translated the commands but also made some of them progressive. A good example is the “*virar esquerda*” (turn left) command, which not only encapsulates the “lt” command but also provides a glide effect while turning the turtle.

To make the error messages understandable by the students, we took the same approach as the one used with the Logo functions — all the error messages were translated into Portuguese. With this task complete, the students had available to them an intuitive Portuguese interface.

Defining our Goals

This was one of the most important stages in our case study, due to the fact that given two different approaches, the RIS and our Logo application, we needed the presence of common variables to allow us to evaluate the test subjects' response to the applications. It's obvious that by using two distinct programming concepts, we can only evaluate the test subjects' response to the applications and whether or not they can complete the given tasks. These tasks can be as simple as moving the robot and turtle from point A to point B, or, once they are familiarized with the instructions they can be given more complex tasks, such as, moving the turtle and the robot on a trajectory that represents a geometric figure. At a later stage the drawing can also be done using loop instructions. With these simple tasks alone, a child can learn the concept of function, how different parameter values affect a function's output and the need to use a way to repeat a block of instructions.

In our application, the tasks can be performed by exploring specially designed theme areas. In RIS, the tasks after being programmed are downloaded into the MindStorms Robot and can then be executed.

The test groups

To be able to assess the students' motivation, we recruited the help of five students from the 6th grade, all of which presented special education needs. The students were divided in two groups. The first, comprised of three students, working with the RIS, and the second group, with two students, using a newly developed application which is presented in the next section.

Developing the new application

We needed to create an application designed to captivate the student's attention, by creating an appealing and intuitive user interface capable of transmitting our defined goals.

After working with LEGO's Robotics Invention System v2.0, we found its interface very appealing with all its programming blocks that one can simply drag into the program. This is a very intuitive way to program and encourages the young programmer to explore.

This is a characteristic that we wanted to incorporate into the design of our application. But, at the same time, this characteristic conceals (to a degree) the concept of function and the procedural aspect of a programming language. This lack of transparency can interfere in the learning process, not allowing the child to reach the objectives we hope to achieve.

From our personal experience, we often observe that someone loses interest in learning if he has to learn something not to his liking. On the other hand we constantly observe a high level of attention when someone is playing a game that might not even be transmitting any kind of knowledge. So, why not combine the intuitive interface of the RIS with the alluring sensation of playing a game? This is exactly what we tried to achieve when developing this project.

Our application consists in two distinct parts, "*Pátio da Brincadeira*" (Playground) and "*Mundo Virtual*" (Virtual World). In the first one the students can choose between three different theme areas where they can learn the Logo commands that will be used to play the game presented in the subsequent part.

Next, we'll explain each of the theme areas, what can be done in each of them, what can the students learn and how can they learn it.

Pátio da Brincadeira (Playground)

This section has a set of theme areas with the sole propose of teaching the basic Logo commands, the same ones that we encapsulated. In order to achieve this, we designed a template screen, like the one shown in Figure 2. The screen has three distinct areas.

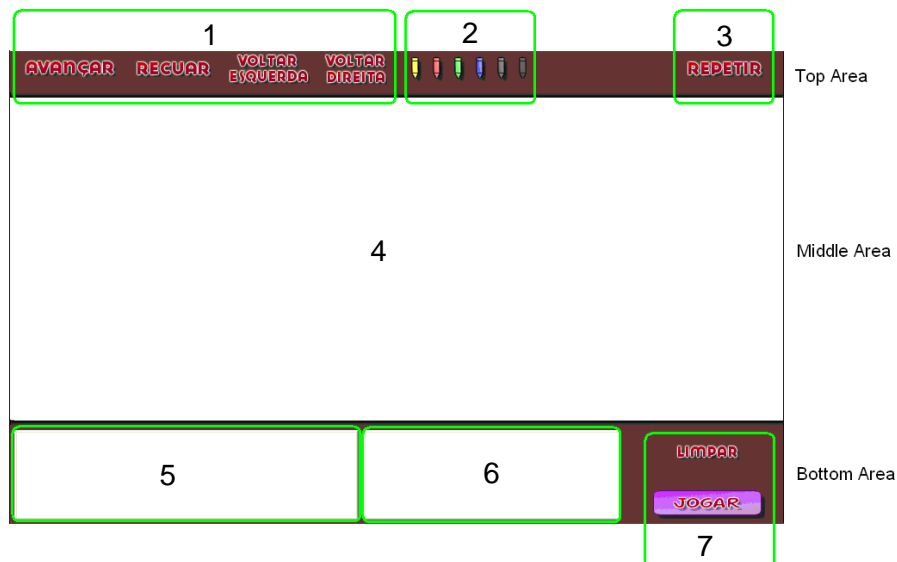


Figure 2. Template Screen for "Pátio da Brincadeira"

At the top area of the template we grouped the movement command buttons (1), the pen colour commands (2) and the repeating block command (3).

At the bottom area there is an input section (5), where the students can create the programs directly by typing at the keyboard or by pressing the buttons available at the top area. To the right of the input section there is a debugger section (6), where the currently running instruction is presented. Still in the lower area, we have available two more buttons (7), which can be used to remove the created program (*Limpar*) and to run (*Jogar*) the commands written in the input section. The run command plays a very important role as it tries to apply one of the key points presented by Seymour Papert (Papert 1996): “*an error should not be faced as a dead end, but instead as mistaken turn*”. This is achieved by running the program from the beginning, command by command, allowing the child to observe the program sequence step by step.

Lastly, the middle area is where the student watches the result of the program he created. For a better usability, all the buttons shown in the user interface change their size and colour when the mouse hovers them.

This area can also be adapted by the developer to supply different themes in order to allow the student to explore a given subject. To this end, we developed three distinct themes, which can be seen in Figure 3, in order to be used in our case study. The first theme was created with the intention to provide the first contact with the application, so it doesn't offer an oriented theme. The second and third theme areas, and taking into consideration that the students have difficulties in some knowledge areas, were designed to explore geometric concepts, like angles and geometrical figures. At the same time we defined a set of exercises to explore the same concepts, like drawing geometric shapes. This way the students are not only learning how to program in the Logo language, but are also learning what is being taught to them in their classroom, and to their knowledge they are only playing a game. This way the game has two main roles, one being the provider of motivation and the other disguising the learning process itself.

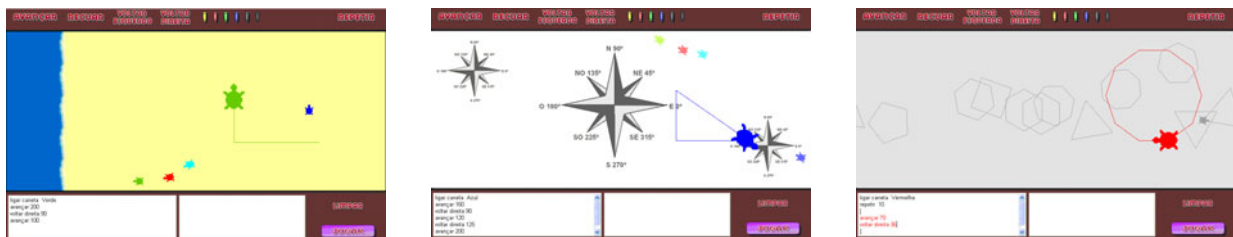


Figure 3. The three themes user in our case study.

Once the students feel comfortable using the application and the logo commands, they can pass to the next level, where they can play a set of small games to apply what they have been learning.

Mundo Virtual (Virtual World)

Mundo Virtual (Virtual World), the second part of the application, consists of two games: Praia (Beach) and Passeio (Stroll).

Praia (Beach)

This game acts as a link between the learning themes and the main game. This is done by keeping available the strictly necessary command buttons and by introducing a world, comprised of a few frames that the student can explore.



Figure 4. Samples frames from the "Praia" game

The objective of this game is to navigate the turtle around the grass and reach the green arrow that gives access to the next frame. The grass acts as a reflective barrier projecting the turtle in a random direction, this way the student cannot simply add instructions to the program in the hope that the turtle reaches its goal. In other words, the child has to give the correct instructions to the turtle so that she doesn't touch the grass. This is relevant due to the fact that, as in any written program, it always starts from its first instruction. This way, if needed, the student has to correct the code he has already written, thus providing a mechanism for him to face the error not as a dead end but as a mistake that needs to be corrected.

Passeio (Stroll)

This is the main game, where the students will be put to test. At this point the student possesses the knowledge he needs to complete the game. He knows how to play the game, what commands are available, and how to use them. But now he doesn't have access to any command buttons, so he is forced to write all the necessary code if he wishes to solve the game.

Our main goal to teach basic programming skills will be achieved if the students are able to finish the game. This is so because in order to complete the game, the students need to create a coherent sequence of instructions that will take the turtle to its final destination. Doing so, the students not only know the instructions, but have also used them in the proper way. Achieving all of this means that we managed to disguise the learning process.

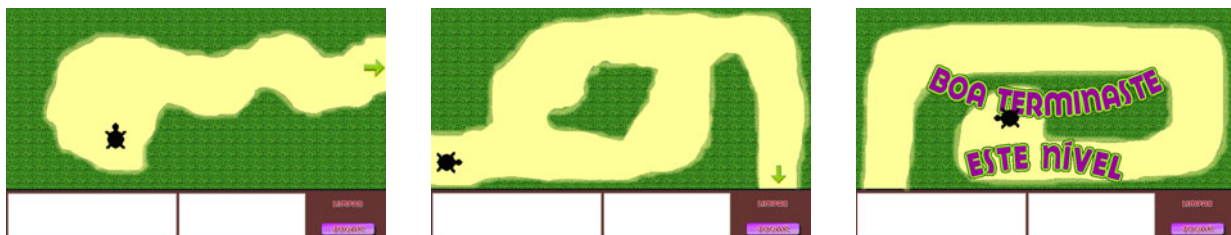


Figure 5. Sample frames from the "Passeio" game

Groups' evaluation

The two groups, the first with three students, using the RIS, and the second with two students, using our application, started the experiment by freely exploring, both RIS and our application, on their own. With RIS, the students controlled a Lego RCX robot, and in our application the students controlled a virtual turtle.

The first tasks given to the students were as simple as moving the robot and turtle from point A to point B. Once they were familiarized with the instructions they were given more complex tasks, such as moving the turtle and the robots on a trajectory that represents a geometric figure.

The experimental sessions were observed and recorded, and later analysed regarding the motivation provided by the applications to the students, and factors such as the time needed to perform a given task, the presented solution and the proper use of the available instructions.

The first good aspect is that both groups succeeded in completing the proposed tasks, and also did it without major difficulties. On a down point, both groups were reluctant to use the loop instruction even when directed to do it.

With the group using RIS, it was noticeable a greater motivation to start with, but it soon faded away. The group using our application showed less interest to begin with, but it never decreased. Just the opposite, the more they “played” the more the motivation grew. While using the application they are using Logo instructions, sequencing them to create a program, so they are actually programming. But, to their knowledge, this is nothing more than a game.

Conclusions

Both applications are able to teach basic programming skills. But our student testers with special education needs were more motivated and obtained better results when using the Logo based application than when using the Lego Mindstorms Robots. The result may be explained in part due to the fact that learning to program the Mindstorms Robot is inherently more complex than giving instructions under a Logo-based application.

Nevertheless, we truly believe in RIS’ capabilities. In fact, it offers a larger array of possibilities than our application, but on the other hand, it is our understanding that the time it takes to create the program, download it into the MindStorms Robot, instructing the Robot to execute the program and actually execute it, is too long to see a response. This is where our application draws its greater strength; it takes no time to see a result from the turtle.

Like a small child that does not want to eat his spinach, but does so when his parents tells him “you have to eat it to be as strong as Popeye”, we gave the students a game to provide the motivation they need to learn a subject that is usually perceived as dull and complex. Much more than that, as they see our application as a game, they managed to learn how to programme in a disguised way.

In our opinion the application that was developed and presented in this paper, is an appropriate tool for teaching the Logo programming language, particularly to SEN students. With it, we found that students could learn without realizing that they were learning.

Acknowledgments

We would like to thank the collaboration of the junior-high-school “Escola Básica 2º e 3º Ciclo Padre João Coelho Cabanita de Loulé, Portugal”, their executive board and to all the teachers and students that participated in this study.

This work was sponsored by the Portuguese Science Foundation under grant POCI/CED/62497/2004.

References

- Papert S. (1993), *Mindstorms: Children, computers, and powerful ideas* (2nd ed.), New York: Basic Books.
- Papert S. (1996), *The connected family: Bridging the digital generation gap*, Longstreet Press, Inc.

PALMA junior – programming competition in Imagine

Ján Guniš, *jan.gunis@upjs.sk*

Dept of Informatics, P. J. Šafárik University in Košice, SLOVAKIA

Ľubomír Šnajder, *lubomir.snajder@upjs.sk*

Dept of Informatics, P. J. Šafárik University in Košice, SLOVAKIA

Valentína Gunišová, *valentina.gunisova@upjs.sk*

Centre for Lifelong Learning, P. J. Šafárik University in Košice, SLOVAKIA

Zuzana Szabóová, *zuzana.szaboova@upjs.sk*

Dept of Informatics, P. J. Šafárik University in Košice, SLOVAKIA

Andrea Hricová, *a.hricova@gmail.com*

Dept of Digital Competencies, Prešov University in Prešov, SLOVAKIA

Abstract

In 2005 we have started to organize new on-line programming competition called PALMA junior using programming environment Imagine based on programming language Logo. The competition is assigned for pupils aged 10 to 15. It runs in two categories: PROFÍK (pupils aged 10 to 11) and EXPERT (pupils aged 12 to 15). The main aims of the competition are encouraging youngsters to solve interesting algorithmic problems, to improve their programming skills and mathematical thinking. Problems, which the competitors solve, are oriented to Programming, ALgorithms and Mathematics (PALMA).

One year of competition composes of four bouts. Each bout consists of six problems. Problems from 1 to 4 are for teams in PROFÍK category, problems from 3 to 6 are for teams in EXPERT category. Students of PROFÍK category can solve the problems in EXPERT category, but it is not possible vice versa. Before and during the competition bouts students can study and use resources we have published on website of this competition. Students work on their own or in pairs.

We have created the authorized area in LMS Moodle where students have an access to discussion forums, assignments of problems for particular bouts, chats and questionnaires. The competitors also deliver their solutions of problems in limited time and they can find there the evaluation of their solutions with comments written by organizers after current bout.

In the paper we describe objectives of the competition in the field of improving competitors' programming skills, algorithmic and mathematical thinking and we also show how we cover the objectives by selected collection of problems. In the frame of the competition we would like also to develop knowledge in other areas (biology, geography, history and sports), aesthetics, creativity, pleasure feeling, EU citizenship, exactness etc. For better illustration of competitions problem types, their formulation, authorial solution, competitors' train of thought and their solution, typical misconceptions etc. we analyse and describe pupils' solutions of 6 selected problems (Kingdom of letters, Treasure Island, Pyramid, A drone bee and its (n x grand) parents, Snow-flake, Flags of EU countries). In future we intend to improve learning materials, evaluation tools for whole competition progress and also prepare on-line course for teachers focused on methodology of programming.

Keywords

Programming competition, primary school, Logo, Imagine, problem analysis

Background of the competition PALMA junior

Programming, algorithms, mathematics - these three words became fundamental elements of online programming competition PALMA organized by P. J. Šafárik University in Košice, Faculty of Science and civic association STROM (Palma, 2007). This competition is dedicated to secondary school students. They have solved the problems using programming languages C/C++ or Pascal. The goals of PALMA are to give opportunity for competitors to compete, to test their ability to analyze problems, to explore algorithms solving these problems and not least to pre-set them quick, effectively and properly. A younger sibling of PALMA – PALMA junior – was established in 2005, too. Essential principles of PALMA junior resulted from our longtime experience in teaching algorithm development and programming. Valuable experience grew from our meetings with computer science teachers which we have organized during several years (Club of Computer Science Teachers). Among other things we have dealt with methods of teaching algorithm development and programming, choosing suitable programming environments and languages, preparing assignments attractive for pupils and supporting their creativity.

Before preparation of our new programming competition we had studied actual programming competitions in Slovakia. All these competitions have similarities – their main goal is to develop algorithmic thinking and programming skills of competitors.

Usually, the long-term competitions are focused on programming languages C/C++ and Pascal (e. g. National Olympiad in Informatics, KSP, ZENIT). They are oriented to secondary school students and are usually organized in attendance way.

Unlike others the Internet Problem Solving Contest (IPSC) is an online competition. It is not focused on particular programming language. It is up to the competitors to select tools which are appropriate for problem solving. There are other differences yet. The competitors can compete in teams (most of trio) and they solve of competitive assignments which are published on IPSC website.

The competitions focused on younger pupils are usually organized in attendance way. They are making use of children programming environments attractive for this age group – Comenius Logo, Imagine, Baltík.

The first and at that time the only competition for pupils aged 10 to 15 – Cologobežka and Imagine Cup celebrates the 10th birthday this year. The goal of the competition is to motivate the pupils which use Logo more seriously for programming activities (Tomcsányiová, Tomcsányi, 2005).

The idea of effectiveness is interestedly integrated into assignment of the competitive teleproject “20 commands” which took place this school year. The goal of the competitive teleproject was to motivate the pupils to work with basic turtle graphics in Imagine environment, to encourage their creativity, imagination and competitiveness.

We take up with the similarly competitions in foreign countries, like National Logo competition-Olympiad in Lithuania (Slapkauskiene, 2005), Logo competition for primary school children – the “miniLOGIA” (Jochemczyk, Oledzka, 2005) or the competition “Beaver” (Dagiene, 2005).

Considering this range of competitions we have tried to come up with another one that should be meaningful and not useless. We have tried to give PALMA junior the most interesting and the most attractive elements for pupils aged 10 to 15 as christening present: creativity, online form, team play, programming environment Imagine, available educational resources and authorial solutions. One of our aims is development of algorithmic thinking of the competitors therefore we consider effectiveness of solutions.

Description of the competition PALMA junior

PALMA junior is programming competition in programming environment Imagine based on programming language Logo. The competition is assigned for pupils aged 10 to 15. It runs in two categories: PROFÍK (pupils aged 10 to 11) and EXPERT (pupils aged 12 to 15). Problems, which the competitors solve, are oriented to programming, algorithms and mathematics.

One year of competition composes of four bouts. Each bout consists of six problems. Problems from 1 to 4 are for teams in PROFÍK category, problems from 3 to 6 are for teams in EXPERT category. Students of PROFÍK category can solve the problems in EXPERT category, but it is not possible vice versa. Before and during the competition bouts students can study and use resources we have published on website of this competition. Students work on their own or in pairs.

E-learning support of contest PALMA junior

We have created the authorized area in LMS Moodle. We have chosen this system because it provides active and dynamic environment for communication with participants and it's open, free and developed continually. Thereby the competition acquires the next dimension and becomes very attractive for competitors.

In the authorized area students have an access to discussion forums, assignments of problems for particular bouts, chats and questionnaires. The competitors also deliver their solutions of problems in limited time and they can find there the evaluation of their solutions with comments written by organizers after current bout.

There is a discussion forum and a chat for each bout. It enables the competitors to discuss uncertainties and troubles before the contest bout, they can communicate in chat between each other or discuss with competition organizers about mistiness consequential from assignments, about delivering the solutions etc.

During the bout there are published assignments of problems and all necessary files in the LMS Moodle environment and on the competition website too. The competitors have three hours for solving the problems. During this period the competitors can deliver their solutions in the LMS Moodle environment. After expiration of the designated time we correct and mark delivered solutions and the evaluation is published. The competitors have appreciated that their solutions are assessed (corrected, marked) straight after finishing of the bout, usually up to two hours.

Apart from the assessment of every delivered solution we have published authorial solutions of problems, cumulative evaluation of contestants' solutions with a list of typical mistakes, gallery of final pictures (results of programs) and ranking of competitors on the competition website. Students who do not take part in the competition have an access not only to additional resources. They can access the assignments of problems and the authorial comments of solutions too. It can help them to learn programming in Imagine environment.

Realization of contest PALMA junior

In school year 2006/2007 we have organized 2nd run of the competition. After realization of on-line bouts we have organized the final, attendance bout of programming competition PALMA junior. This final bout was organized at P. J. Šafárik University in Košice, Faculty of Science and the best teams have taken part in this bout. Besides the competition we have prepared funny form of some activities for the competitors – they can solve several logical problems. Absolute winners of the first year were awarded with nice presents.

We are interested in opinion of teachers about this type of competition, therefore during the final bout we discussed with teachers. We talked about an acquisition of this competition for students and teachers, about abilities to improve it. We are interested in teaching of informatics (computer science), especially of algorithm development and programming.

The present result of discussion is that this competition stimulates students from participating schools to be interested more in programming. Natural emulation of students (in a school) helps them to achieve better educational results. The teachers have appreciated accessibility of educational resources and the competition by itself as very good way of students' motivation to improve their algorithmic thinking and programming skills.

According to information received from teachers, there is very active and exciting atmosphere at schools, and contestants are enjoying during the bouts. Some teachers communicate with us via e-mail or chat regarding problems related to organization of actual bout. After the finish of a bout the teachers can express their attitude to choice and quality of solved problems and to atmosphere in classrooms during the bout, some of them send us photos from the bout.

Objectives and selection of problems for the competition

Objectives of PALMA junior competition

The main aims of the competition are encouraging youngsters to solve interesting algorithmic problems, to improve their programming skills and mathematical thinking.

Programming skills of competitors can be enhanced by programming of problems whose solutions include loops, conditions, own procedures, operations, variables, procedures and operations of turtle graphics and Cartesian co-ordinate system, visual components, events, using several turtles, lists, recursion etc.

Competitors can develop their algorithmic thinking by using algorithms concerning drawing planar objects with given properties (turtle as a processor of algorithms), sorting and searching objects (numbers), traversing through 2D array (e.g. labyrinth), creation and evaluation of own formulas.

Mathematics thinking can be developed by solving problems which need basic knowledge and skills in the areas of relative orientation in turtle geometry, orientation in Cartesian co-ordinate system, properties of planar objects, calculation of angles, sequences of numbers, arithmetical functions, Pythagorean theorem, calculating position in 2D array, factoring numbers, unit conversions, fractions etc.

In the frame of the competition we would like also to develop knowledge in other areas (biology, geography, history and sports), aesthetics, creativity, pleasure feeling, EU citizenship, exactness.

Coverage of the objectives by selected collection of problems

In preparation of problem assignments and solutions we have tried to cover all above described dimensions. We have created problems oriented to drawing objects and/or calculations with aesthetics experience (feeling), using knowledge in various areas. Some problems are games (Racing ball, Guess number game, Coins, Clown's houses, Treasure Island), which competitor can use also after competitions. After each bout we publish gallery of pictures which are results of drawing problems.

For better overview (illustration) how we try to cover all objectives by competition problems we have prepared table (Fig.1) that is based on authorial solutions of competitions problems. We have gradually filled in this table and use it as an aid for creation of new problems during the first two years of the competition. We consider it to be also a good cornerstone (starting point) for more precise creation of competition problems in the 3rd year of the competition.

	Programming													Algorithms							Mathematics											
Problem name	turtle graphics	Cartesian co-ordinate system	procedures without parameters	procedures with parameters	loops - given number times	loops - conditions	nested loops	conditions	variables - using, changing, operations	randomness	visual components, events	using several turtles	recursion	lists	turtle as a processor of algorithms	traversing through 2D array	labyrinth traversing, testing conditions	randomness in bounds	drawing shapes with some properties	sorting algorithms	searching algorithms	creation and evaluation of expressions	relative orientation in turtle geometry	orientation in Cartesian co-ordinate system	properties of planar objects	calculation of angles	sequences of numbers	arithmetical functions	Pythagorean theorem	calculating position in 2D array	factoring numbers	unit conversions
House	*				*										*								*		*	*						
Olympic rings	*														*								*		*	*						
Sun	*				*										*								*			*						
Chessboard	*	*		*		*									*	*							*							*		
EC flag	*	*		*		*		*	*						*								*		*	*		*				
Clocks	*		*					*							*							*		*								*
Santa Claus	*	*		*											*							*	*	*	*	*						
Racing ball	*				*		*								*		*					*	*									
Spiral	*				*		*	*							*												*					
IQ test		*						*			*											*					*	*				
Screen saver	*				*				*	*					*			*				*		*	*			*				
Wallpapering	*		*	*		*		*	*						*	*							*		*	*				*		
Snowman	*			*	*	*		*	*						*								*		*		*				*	
Star	*		*	*				*							*			*					*		*	*						
Pyramid	*	*	*	*	*	*		*							*			*					*		*	*			*			
Guess number game					*		*	*	*	*							*		*		*			*	*			*				
Coins		*	*					*	*		*	*								*		*		*	*	*		*				
Flake	*		*	*	*	*	*	*	*				*		*			*					*		*	*	*					
6 EC flags	*		*	*	*		*	*	*	*	*	*			*			*				*	*	*	*	*						
Guess state game	*		*	*	*		*	*	*		*	*	*					*		*		*		*	*		*	*		*		
Travelling over EC	*		*				*	*		*	*	*	*			*						*		*	*				*			
Pumpkin	*				*										*								*		*	*						
Winner stages I.	*		*	*				*							*			*					*		*	*		*				
Target	*		*	*			*	*							*			*						*		*		*				
Clown's houses	*	*		*	*		*	*							*			*				*	*	*	*	*		*				
Winner stages II.	*		*	*	*		*	*	*		*	*	*		*			*		*		*	*	*	*	*	*	*		*		
Aquarium			*	*		*	*	*														*		*			*			*		
Letter figure	*	*						*					*		*								*		*	*						
Castle	*		*	*	*	*		*				*			*	*		*					*		*	*						
Treasure island	*	*			*		*								*	*	*					*		*								
Parents of bee			*	*			*	*			*											*					*					
Lady-bug			*	*			*	*		*					*							*			*	*		*				*
Shell	*		*	*			*		*						*			*				*	*		*	*		*		*		

Figure 1. Coverage of the objectives by selected collection of problems

Analysis of pupils' solutions of selected problems

Our aim is not to create robust and complex solutions of problems. We would rather show the beauty of programming on simply, elegant and effective solutions.

We selected some few assignments to illustrate the type of competition problems, their formulation, authorial solution, competitors' train of thought and their solution, typical misconceptions etc. The goal of assignment formulation is to teach the pupils to identify the problem hidden in the assignment. This competence – to make sense of assignment – is very important. It is presumption for successful problem solving.

Kingdom of letters

The first assignment of each bout is traditional “drawing”. It provides wide space to the competitors to apply own fantasy and creativity. At first sight a simple problem, but indeed it

offers ability to experienced competitor to use his knowledge and solve the problem by implementation of parameters.

The competitors should have to create arbitrary figure by using only letters and other characters.

The solution of this problem is oriented to working with properties of text, with position and orientation of turtle (turtle co-ordinates and heading). To fit the turtle in place the competitors were in need of correct computation or approximation of turtle position.

Procedure *type_char* has got six parameters defining a position of turtle (x, y), its orientation (*direction*), size (*size*) and colour (*colour*) of text and character (*char*):

```
to type_char :x :y :direction :size :colour :char
  setPos list :x :y
  setHeading :direction
  setFont (list "Arial (list :size 10 0 0 1 238))
  setPenColour :colour
  label :char
end
```

```
to figure
  hideTurtle
  clearScreen
  ;FACE
  type_char -40 125 0 100 5 "O
  type_char 30 178 -90 50 9 ":"
  type_char 5 165 0 20 5 "L
  type_char 22 150 -90 30 4 "("
  ;BODY
  type_char -54 10 0 120 3 "M
  ;LEGS
  type_char 65 10 180 120 8 "V
end
```

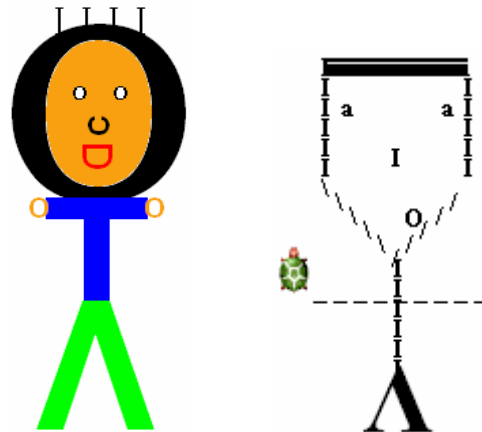


Figure 2. From competitors' gallery

In a few solutions the competitors drew figures by drawing their own characters. They did not take advantage of turtle ability to print text in its current position.

The competitors created several procedures for each part of a figure (head, body, legs, etc.), they worked with properties of text, with position and orientation of turtle, but they did not use parameters. In spite of it, the competitors created patchworky figures.

Treasure Island

There is usually one assignment in every bout which offers prepared environment to the competitors. The competitors' goal is to integrate their own necessary procedures to activate this environment following the instructions of assignment. We try to show programming as funny activity too. The competitors can easily create simple games and enjoy them.

The competitors should have to create an algorithm processor – pirate Kubo – to find the way through the forest. We prepared environment – pirate map with special marks. Pirate Kubo followed these marks telling him which way to go. To make it harder for Kubo, shape of map was changed after every game (but always solvable).

This assignment required good orientation in two-dimensional array. We were interested in effective evaluation of conditions and computing exact position of pirate.

Direction of every move is assigned by direction of arrows on the map. The arrows defining the same direction are drawn in the same colour. So, for pirate Kubo it is enough to test the colour of point he is standing on. Then he can decide which direction he should go on (turtle changes its position). As he finds he stands on white point – he came over the forest and he achieved his aim, he is near the treasure. A procedure *search* tests colour of point below pirate Kubo and according to this colour it changes his position:



Figure 3. Map of Island of treasure

```

to search
  forward 40
  while [not (dotcolour="white")]
    [if dotcolour="blue [ setxcor xcor+40 ]
     if dotcolour="yellow [ setycor ycor+40 ]
     if dotcolour="green [ setycor ycor-40 ]
     if dotcolour="red [ setxcor xcor-40 ]
    ]
  end

```

We can notice that application of nested commands ifelse does not lead to more effective solution. Pirate Kubo can execute more steps in one cycle. Majority of competitive teams used this type of solution. But we do not know it is because they reflected this fact, or it is because they simply did not think about ifelse commands. It is interesting that almost a quarter of the competitors used technique of recursion. It means again insufficient comprehension of meaning and applicability of using recursion.

Pyramid

This assignment represents more complex version of assignment about drawing a chess-board. In contrast to chess-board (where every row consists of same number of squares), row of pyramid contains two triangles less than row below it.

The competitors should have to create procedure for drawing the pyramid. Input of procedure was number of pyramid floors (n).

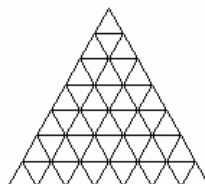


Figure 4. Pyramid with seven floors

It is necessary to put sufficient time to analyze the way of drawing the pyramid. The competitors solved this problem in two different ways.

In the first concept, one cycle draws one row of pyramid. This cycle is repeated in superior cycle, of course, always with declining numbers of repeating.

```
to pyramid :n
  repeat :n [right 90
    forward (:n-counter+1)*20
    left 120
    repeat (:n-counter+1)
      [forward 20 left 120 forward 20 right 120]
    right 60 forward 20 left 30]
end
```

In the second concept, the competitors offered recursive version of this algorithm. Procedure draws one row of pyramid, changes a turtle position and calls its next instance with reduced parameter – the number of floors.

```
to pyramid :n
  if :n > 0 [right 90 forward :n*20 left 120
    repeat :n [forward 20 left 120 forward 20 right 120]
    right 60 forward 20 left 30
    pyramid :n-1]
end
```

The question is whether using recursion is appropriate in this situation. It leads to correct solution, but it is apparently fruitless. A fact is that the competitors know this way of solving problems. In the meantime it shows that they do not appreciate disadvantages, which recursion brings.

A drone bee and its (n x grand) parents

Completing the terms of sequence is fancy mathematical mind-bender. Well-known sequence is Fibonacci sequence, and the most frequent problem, which leads to Fibonacci numbers, is problem of Fibonacci's rabbits. We chose a less known occurrence (Knott, 2007).

The competitors should have to compute how many (n x grand) parents drone bee has got. We showed one important fact – every drone bee has got only one parent (queen bee) and every queen bee has got two parents (queen bee and drone bee).

We were interested in the way which the competitors used to solve this problem without knowledge in field of sequence (concept, scheme). We knowingly mentioned only number of drone bee's parent (1) and grandparents (2) in problem assignment. Our tendency was to illustrate a difference between family trees of man and drone bee. To verify a right comprehension of text and correct analysis of the problem we let the competitors to discover a dependency and consecutively to compute the next values.

To compute the number of drone bee's (n x grand)parents we availed except the variables *ancestor1* and *ancestor2* (which contained two actual members of sequence) also additional variable *a* (to realize the computation of next terms).

```
to ancestry :n
  ifelse :n=0 [type "1]
    [ifelse :n=1
      [type "2]
      [let "ancestor1 1
        let "ancestor2 2
        repeat :n-1 [let "a (:ancestor1 + :ancestor2)
          let "ancestor1 :ancestor2
          let "ancestor2 :a]
        type :ancestor2]]
end
```

We mentioned a solution without additional variable in author's commentary to this assignment. We can monitor how the competitors will use it in future assignments.

The competitors successfully discovered a mathematical model of the problem. In a few of their solutions they did not pay sufficient attention to testing marginal values. The result of it was incorrect output for input value 0 (number of parents). It is very frequent mistake of novice programmers.

We found one atypical – however not more effective – solution. The competitor divided the problem to two branches according to value of n .

```
to ancestry :n
  ifelse mod :n 2 = 1 [make "anc1 1
                        make "anc2 2
                        repeat (:n-1)/2 [make "anc1 sum :anc1 :anc2
                                              make "anc2 sum :anc1 :anc2]]
                        [make "anc1 1
                          make "anc2 1
                          repeat :n/2 [make "anc1 sum :anc1 :anc2
                                              make "anc2 sum :anc1 :anc2]]
  type :anc2
end
```

Snow-flake

By this assignment we wanted to find out how the competitors understand the concept of recursion. We were interested in how they managed computing of angles closed by arms of flake, how they managed orientation by turtle geometry. Recursion is rather challenging programming technique and it calls for a high level of abstraction. If the competitors know ways of drawing typical recursive graphics (binary tree, Koch flake), will they be able to generalize this process and apply it on other graphics?

The competitors should have to draw recursive graphics – snow-flake. There is smaller, half-size snow-flake in every vertex of snow-flake.

We can define this graphics by triplet of parameters: number of arms (*side*), length of arm (*length*) and depth of recursion (*depth*). Then, we can write a procedure flake using technique of recursion:

```
to flake :side :length :depth
  if :depth > 0
    [repeat :side
      [forward :length
       flake :side :length/2 :depth-1
       back :length
       left 360/:side]]
  end
```

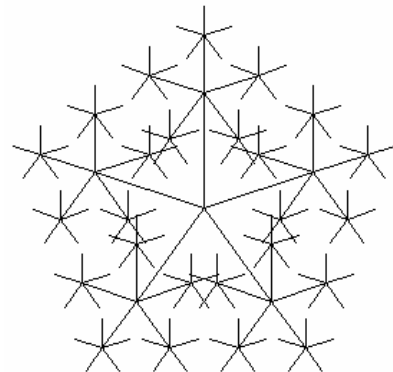


Figure 5. Snow-flake (flake 5 80 3)

Although this assignment contained several samples of snow-flakes with different depth of recursion, we think of this problem as difficult. Almost all competitors mastered the first level of snow-flake. But in next levels the competitors made typical mistake. This mistake is consequence of insufficient analysis of problem and unrecognized recursive character of graphics. Majority of competitors solved the second level by using next, nested cycle. Only one competitive team solved this problem correctly.

Flags of EU countries

This assignment was solved in the final bout by the winners of school bouts. We used the fact, that Slovakia became a member of European Union, so we reminded the history of formation of European Union.

The competitors should have to create procedure drawing flags of foundation states of EU.

The flags of these states are very similar. Every one of them consists of three varicoloured bars. They differ by proportion of sides, colours of bands and their orientation. As we find important to create algorithms solving a class of problems, not just one special situation, we were interested in level of parameterization achieved by competitors.

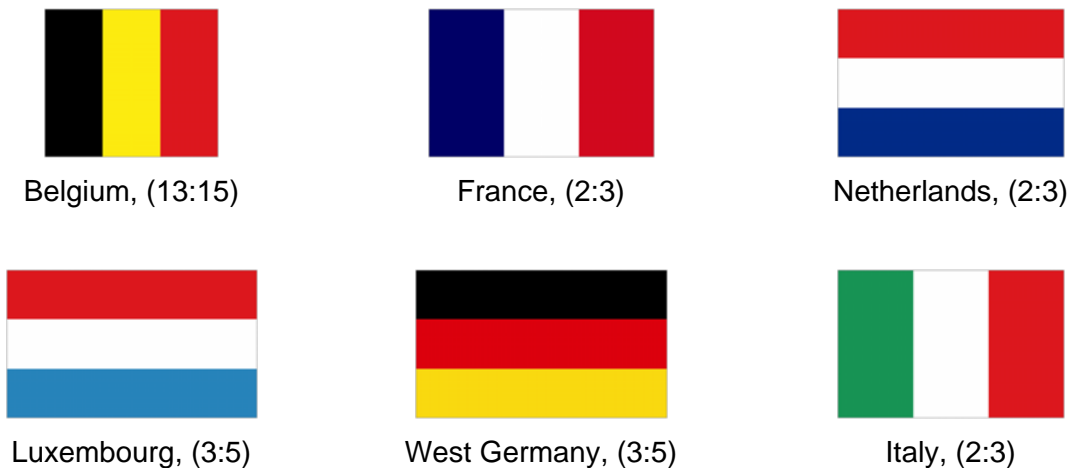


Figure 5. Flags of foundation states of EU

We can solve this problem in several levels of parameterization. Direct solution used by majority of competitors means one independent procedure for every flag. One competitive team used procedure with one parameter determining the state. After testing this parameter one sequence of commands were executed. They used only one procedure, but we can not talk about algorithm solving class of problems.

Decomposition of flags into two groups according to orientation of bands is the next level of problem analysis. We can create self procedure for every group. This solution was not used by any competitive team.

We can reach the highest level of parameterization by creating the only one procedure. Its parameters are orientation of bands (*orientation*), list of colours (*colours*) and length of flag sides (*width* and *height*). Only one competitive team scored this level.

```
to flag :orientation :colours :width :height
  right :orientation
  if :orientation=0 [right 90 forward :width left 90]
  let "height :height/3
  repeat 3 [setpencolour item repcount :colours
    polygon [forward :height
      left 90
      forward :width
      left 90
      forward :height]
    forward :height]
end
```

We can draw the flag of Belgium by `flag 90 ["black "yellow "red5] 182 210`, the flag of Netherlands by `flag 0 ["blue3 "white "red5] 210 140`.

Conclusions

Based on two years experience with running PALMA junior competition we can say that we consider our effort as successful in some areas. The competitors were able to solve the competition problems and we believe that this competition helps them to increase their problem solving and programming skills and mathematical thinking. According to the teachers' opinions this competition helps them to (manage and) encourage learning and self-learning of gifted pupils in programming and also ordinary programming lessons because we have published study material, competition problems with assignments, solutions, commentaries and picture gallery on the competition website that is available for everyone.

We believe that our competition PALMA junior has its own place among other programming competitions in our region because of its approaches - covering aims in 4 dimensions/areas, on-line accessibility, openness to involving for everyone, publishing collection of all competition problems with results, commentaries.

In future we would like to continue in team style of organizing the competition and to create problems for competition covering mentioned aims and objectives using analytical approach. We intend to improve learning materials for the competition, evaluation tools for whole competition progress and also prepare on-line course for teachers focused on methodology of programming. In case of interest we are open to foreign competitors who can solve programming problems in English version of Imagine.

We would like to thanks our colleagues Gabriel Semanišin and Gabriela Andrejková for their support and advices and their help in the field of propagation of the competition and acquirement of prizes for finalists. We also thank to our students Alena Hajdová, Jana Brandoburová for their help in preparing assignment of problems and for checking competitors' solutions.

All results published in the article have been achieved in the frame of APVV project LPP-0131-06 "Increasing of knowledge potential".

References

- Dagiene, V. (2005) *Competition in Information Technology: an Informal Learning*. In Proceedings of the Tenth European Logo Conference, Warsaw: Centre for Informatics and Technology in Education, 2005, pp. 228-234, ISBN 83-917700-8-7
- Dvorský, M. et al. (2007) *Palma – programming competition*. P. J. Šafárik University in Košice, Faculty of Science <<http://palma.strom.sk/>>
- Jochemczyk, W. and Katarzyna Oledzka, K. (2005) *Logo competition for primary school children*. In Proceedings of the Tenth European Logo Conference, Warsaw: Centre for Informatics and Technology in Education, 2005, pp. 383-385, ISBN 83-917700-8-7
- Knott, R. (2007) *Fibonacci Numbers and the Golden Section*. <<http://www.mcs.surrey.ac.uk/Personal/R.Knott/Fibonacci/>>
- Šlapkauskienė, V. (2005) *National Logo competition-Olympiad in Lithuania..* In Proceedings of the Tenth European Logo Conference, Warsaw: Centre for Informatics and Technology in Education, 2005, pp. 367-369, ISBN 83-917700-8-7
- Tomcsányiová, M. and Tomcsányi, P. (2005) *Logo programming competition in Slovakia*, In Proceedings of the Tenth European Logo Conference, Warsaw: Centre for Informatics and Technology in Education, 2005, pp. 377-382, ISBN 83-917700-8-7

Painless Trigonometry: a tool-complementary school mathematics project

Jesús Jiménez-Molotla, jj_molotla@hotmail.com

Escuela Secundaria Diurna #229 "Ludmila Yivkova", Turno Matutino, DF, Mexico

Alessio Gutiérrez-Gómez, alessio@starmedia.com

Escuela Secundaria Diurna #229 "Ludmila Yivkova", Turno Matutino, DF, Mexico

Ana Isabel Sacristán, asacrist@cinvestav.mx

Dept. of Mathematics Education, Centre for Research & Advanced Studies (Cinvestav), Mexico

Abstract

Since 2001-02, we have been working in our mathematics classrooms with the materials and digital tools provided by a government-sponsored national programme: the Teaching Mathematics with Technology programme (EMAT). The main computer tools of the EMAT programme are Spreadsheets (Excel), Dynamic Geometry (Cabri-Géomètre), and Logo (MSWLogo). At the beginning we used these tools independently, but in more recent years we have tried to develop long-term projects that incorporate all the tools, and that also serve as means to introduce students to topics of mathematics that are normally considered too advanced for them (such as trigonometry).

In this paper we report the design and results of one of those projects, a trigonometry project, that we first implemented in the academic year 2005-06 with 6 groups of the first two grades of two junior secondary schools in Mexico. Approximately 250 students of 12-14 yrs of age participated, in total, in the project in that year.

We believe in the importance of using, in an integrated and complementary way, a variety of tools for learning since we consider that each tool brings with it a different type of knowledge and constitutes a different epistemological domain (Balacheff & Sutherland, 1994). We also believe in the importance of constructionist (Harel & Papert, 1991) or programming activities for meaningful learning. With those fundamental theoretical premises, we developed the long-term trigonometry project for introducing young students to the Pythagorean theorem, basic trigonometry concepts, and their applications using explorations and constructive activities with Cabri-Géomètre, Excel and Logo (Figure 1). Students thoroughly enjoyed the activities and gained interest in mathematics. They also developed problem-solving and collaborative skills. Furthermore, in written tests after the project, the students showed an understanding of the "advanced" trigonometry concepts, as well as of other algebraic ideas.



Figure 1. Complementary trigonometry explorations and constructions with Cabri, Excel and Logo

Keywords

Mathematics; trigonometry; Pythagorean theorem; Logo; Excel; Cabri-Géomètre; school project

...children can learn to use computers in a masterful way, and [...] learning to use computers can change the way they learn everything else... (Papert, 1980; p. 8)

Background

In 2001-2002, the Teaching Mathematics with Technology (EMAT) Programme began to be implemented in the junior secondary schools we work at, in Mexico City, Mexico. EMAT is a programme sponsored, since 1997, by the Mexican Ministry of Education to promote the use of new technologies, using a constructivist approach, to enrich and improve the current teaching and learning of junior secondary mathematics in Mexico. A study carried out in Mexico and England (Rojano et al., 1996) involving mathematical practices in science classes, revealed that in Mexico few students were able to close the gap between the formal treatment of the curricular topics and their possible applications. This suggested that it was necessary to replace the formal approach of the then official curriculum, with a “down-up” approach capable of fostering the students’ explorative, manipulative, and communication skills. Thus, a main part of the EMAT programme are pre-designed activities and a pedagogical model that emphasize *exploratory* and *collaborative* learning. And the various EMAT software were chosen on the criteria (Ursini & Rojano, 2000) that they be *open* tools; that is, where the user could be in control and have the power of deciding how to use the software. These open tools had to be flexible enough so that they could be used with different didactical objectives (those of the pre-designed activities, or others as well). The main computer tools of the EMAT programme are Spreadsheets (Excel), Dynamic Geometry (Cabri-Géomètre), and Logo (MSWLogo).

Initially at our schools, we worked with each of these tools separately, covering different themes with each of them. Logo was the last tool that we incorporated into our schools, and one of the immediate things we noticed was how much it enriched, not only children’s motivations for exploring mathematical topics, but also the use of the other tools: because of the programming experience with Logo, students began asking if it was possible to also program the other tools (this led us, for example, to show them how to create macros in Excel and Cabri).

Moreover, we began to look into ways of integrating the three tools. Each tool brings with it a different type of knowledge and constitutes a different epistemological domain (Balacheff & Sutherland, 1994). This, in fact, is one of the reasons why the EMAT designers provided various tools, but their use has generally been independent one from the other. We consider it, however, important to use these tools in an integrated and complementary way.

The trigonometry project

We thus took up the challenge, in the academic year 2005-06, to create an ICT-based project for the learning of a topic which is traditionally difficult to teach and learn, and more so at the junior secondary level (children aged 12 to 15 yrs-old): trigonometry. This is a topic which is in the curriculum only for Grade 3 (the older students) of the junior secondary level; but we decided to work with the younger students of Grades 1 and 2. On the one hand, we concur with the idea that ICT can act as scaffolding for learning more advanced topics; on the other, our strategy was to familiarize students with this topic so that by the time they get to Grade 3 and have to formally learn trigonometry, they will have experiences and useful intuitive ideas (diSessa, 2000) to build upon. For Grade 1 students, we would build upon the curricular themes of *operations with decimal numbers, square root and powers*, and from there create a link to the trigonometry project. In the case of students of Grade 2 we would have less difficulty, as these children already work with algebra; we could thus link it to curricular themes of *polynomials and the use of variables* as well as those of *square root, geometrical figures*.

Our approach was to use not only the EMAT tools –Logo, Cabri and Excel— but also Powerpoint presentations and paper-and-pencil activities, to introduce the theme and the use (for this project) of the various tools.

The stages and implementation of the project

One of the first aims was to combat the apathy that the majority of Grade 1 students have towards mathematics. The use of the technological tools provided a means to motivate our students.

In the first sessions of the school-year we taught students, of both grades, the basic use of the three tools (Logo, Cabri and Excel) through some of the pre-designed EMAT activities (students of neither group had used any of these tools before –although in our school the tools have been available for a few years, the teacher that the Grade 2 students had had the previous year, did not make use of them).

Later, we began to engage students in the trigonometry project explorations: For this project, we began with an integrated theoretical-practical Powerpoint presentation and a paper-and-pencil exercise, followed in later sessions, with technology-based explorations, as described further below.

The explorations were carried over at least a dozen weekly technology-based sessions (i.e. one 50 minute session, once a week). On the one hand, the long-term investigation over the course of the year was motivating and engaging for everyone; it also provided time between sessions to rethink and re-orchestrate the strategies, if necessary. On the other hand, we had the problem, for students, of the long weekly (and sometimes longer) gap between sessions; also, in the case of the group that had their weekly technology sessions, at the end of the week, we noticed weariness that slightly affected the sessions.

The presentation and paper-and-pencil exercise

After covering the curricular theme “Square Roots with Decimals”, we began with a Powerpoint presentation and an exercise on the Pythagorean theorem. In the exercise a square is drawn inside a 7 X 7 grid in such a way that four right-angle triangles are formed (each with legs of sizes 3 and 4), with the sides of the square being the hypotenuses (see Figure 2). In this example, the length of the hypotenuse is the integer 5, but we clarified that it was an exception. The aim of this exercise was to illustrate (see Figure 3) that the sum of the squares of the legs of each right triangle was the same as the square of the length of the hypotenuse (the area of the inside square).

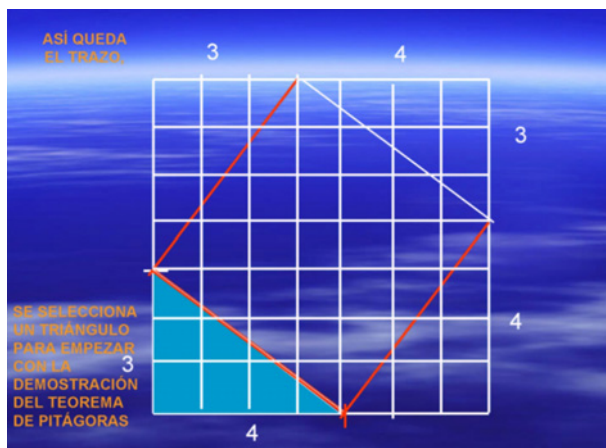


Figure 2. The first step in the Pythagorean Theorem exercise



Figure 3. The last slide in the Pythagorean Theorem presentation

The technology-based explorations

The previous exercise served simply as an introduction to the Pythagorean theorem and as support for the software-based explorations that followed, where students would investigate

further this theorem and its application through construction activities with Cabri, Excel and Logo. In these technology-based activities, the students were the main actors, collaborating and directing the construction process and their peers. As recommended by the EMAT pedagogical model, students worked in teams of two (or sometimes three), with one computer. The structuring of teams was important for discussing and collaborating on the solutions to the problems, and confronting different strategies. The teams then had the opportunity to present to the entire group their work, and engage in whole-classroom discussions (which is also an important recommendation of the EMAT pedagogical model).

(It may be worth mentioning that we did have a difficulty towards the end of the year, when some of our projection equipment was damaged and whole-group presentations were not as easy; but this is part of the challenges of working with technology.)

The Cabri and Excel explorations

The first technology-based activities, were geometric explorations of the Pythagorean theorem using Cabri (see Figure 4), to illustrate that the size of the hypotenuse of a right triangle can always be determined by the sizes of the legs of the triangle; we also introduced the trigonometric circle and basic trigonometric concepts such as sine, cosine and tangent.

With Excel, we introduced exponential formulas for powers and square roots, as well as trigonometric formulas which were related to the explorations with Cabri. In this Excel exploration (see Figure 5), the students programmed the Pythagorean theorem formulas for finding the values of the legs and hypotenuse of a right triangle; as well as formulas, using trigonometric functions, for finding the value of missing angles. They also programmed interactive buttons, something which was motivating for them, and that forced them to understand better the underlying relationships.

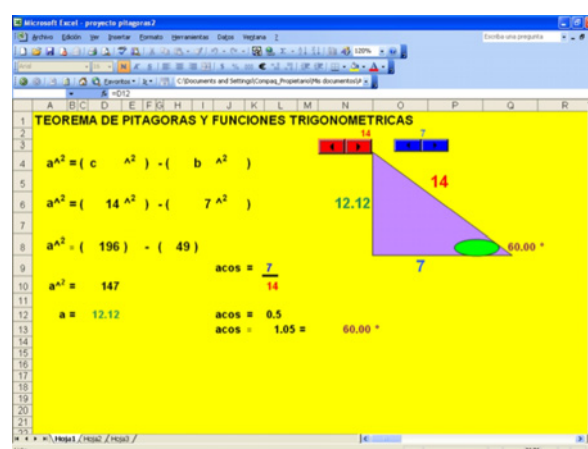
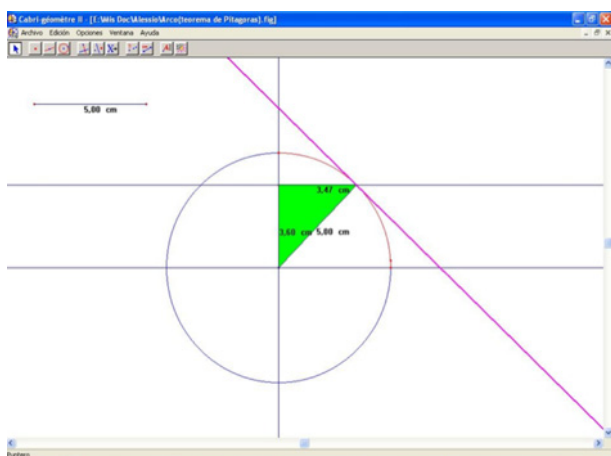


Figure 4. The trigonometry explorations with Cabri.

Figure 5. The Excel explorations of the Pythagorean theorem and of trigonometric functions

The Logo explorations

With Logo, we divided the explorations in several stages:

In the first stage, students had to construct a specific case: a right triangle with legs 30 and 40 (which meant the hypotenuse was a nice, and easy to compute, integer). They soon learned (out of need) to use the primitives *arcsin*, *arctan*, and *arccos*, since without them they would not have been able to compute the rotation angles for drawing the triangle. In this sense, the previous activities with Cabri and Excel were very useful; the first, because it gave them an understanding of the underlying concepts of the trigonometric functions; and the latter, because it familiarized them with the formulas and use of the trigonometric functions. Now, in Logo, in order to use the

formulas for finding the missing angles, they really needed to be able to read them properly, and to understand exactly how to apply them in the new context (a situation where, in many cases – depending on how the procedure was constructed—, the angles needed were the *rotation* angles of the triangle, instead of the inner ones).

In the second stage, the task was to write programs for other right triangles, using any sides (legs) of the triangle they wanted. This time (if they hadn't done it before) they had to use, within their procedure, the formula derived from the Pythagorean theorem and the *sqrt* primitive command for the square root, in order to generate the hypotenuse. At a first attempt, they could compute with paper-and-pencil, or in Logo's direct mode, the rotation angles and simply use the values in their procedure (Procedure 1). But they later had to also program, within the procedure, the formula for the rotation angle (see Procedure 2).

Procedure 1

```
to pitagoras
  fd 100
  bk 100
  rt 90
  fd 100
  lt 135
  fd sqrt(100 * 100 + 100 * 100)
end
```

Procedure 2

```
to pitagoras2
  fd 200
  bk 200
  rt 90
  fd 150
  lt 180
  rt arctan (200 / 150)
  fd sqrt (200 * 200 + 150 * 150)
end
```

The latter was the stepping-stone for the final generalization stage, in which students had to write a general program for a right triangle, using variables. Furthermore, in this stage, students not only had to construct the procedure for the right triangle, but at the end, they were also asked to write a procedure that would draw the squares on each side of the triangle and compute their values for labelling the figure. In this way, they built a model of the concept of the Pythagorean theorem (see Procedure 3 and Figure 6).

Procedure 3: Final procedure for the Pythagorean model programmed by one of the teams of students

```
to triangulo :x :y
  fd :x bk :x rt 90
  setcolor 20
  fd :y lt 180
  wait 30 rt arctan :x / :y
  wait 30 fd sqrt(:x * :x + :y * :y)
  wait 30 setcolor 10 rt 90 repeat 3 [fd sqrt(:x * :x + :y * :y) rt 90]
  lt 90 + arctan :x / :y
  wait 30 setcolor 5 repeat 4 [fd :y rt 90]
  rt 90
  fd :y
  wait 30 setcolor 20 repeat 4 [fd :x rt 90]
  rt 90
  fd :x rt arctan :x / :y
  label sqrt (:x * :x + :y * :y)
end
```

When students had completed their programs, we let them play around freely with their procedures (see Figure 7). This type of activities are important to relax, motivate and reward the students for their hard work.

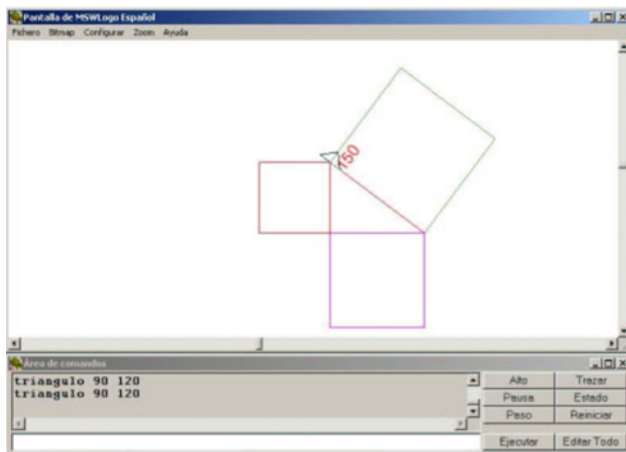


Figure 6. The Pythagorean model produced by the final procedure in Logo.

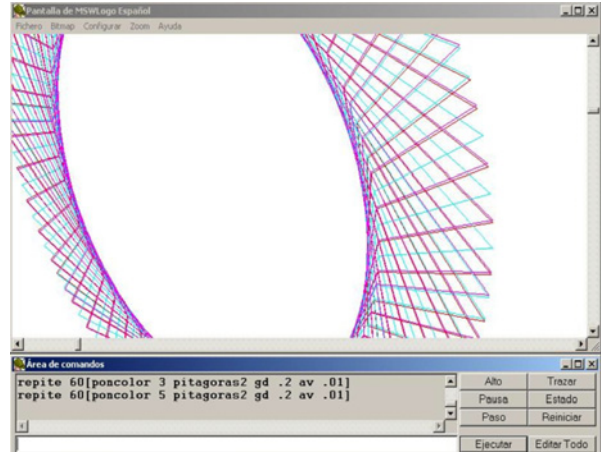


Figure 7. Example of some students playing with their right triangle procedures in Logo.

Some results

In other papers (Sacristán, 2003; Sacristán et al., 2006) some of the benefits (as well as difficulties) of the implementation of the EMAT programme, its tools and pedagogical model, have already been reported, in particular those concerning the development of skills and motivation in students. Here we present the particular observations from our schools.

Observations on the development of problem-solving abilities and collaborative attitudes

Since we began integrating the EMAT ICT tools for our mathematics courses, and particularly Logo and other “programming” or constructionist (Harel & Papert, 1991) activities, we have noticed that our students have developed many valuable learning skills: In particular, there is an increase in their problem-solving abilities, and they tend to reflect more on the problems and even go beyond the tasks required.

In fact, both the students and the teachers involved in the EMAT programme at our schools and in the project presented in this paper, have found new means of solving problems or doing things. Because we do not impose a particular way for solving the tasks, there is more freedom to analyse, conjecture, test, discuss, compare and learn from each other, with a collaboration between the students, the math teacher and the ICT teacher.

We also believe that the team-work during the sessions, and the sharing and discussing of solutions and strategies at the end of each session, are fundamental elements for reaching successful solutions and for stabilizing the learning derived from the technology-based activities and explorations.

Achievements in School Assessments

In the end-of-term mathematics tests, the students involved in our courses had very high scores, with a high percentage of success rate (see Table 1) and it is worth noting that parts of these tests covered trigonometry and algebra, even for first grade students. Samples of solved tests are given in Figure 8.

Group	1° A	1° B	1° C	1° D
Nb. of students with pass grade	35	34	32	42
Nb. of students with fail grade	9	10	15	5

Table 1. Results of the end-of-term mathematics tests for Grade 1

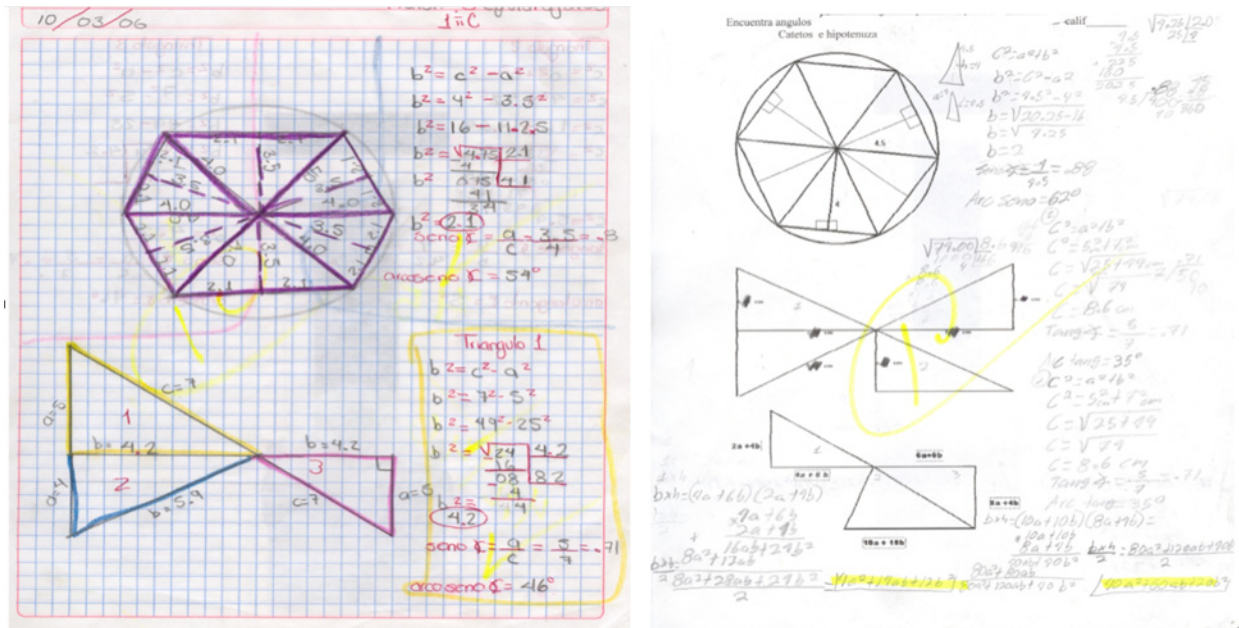


Figure 8: Written tests on Trigonometry by a Grade 1 student and a Grade 2 student, respectively

Observations on affective aspects

With the use of the ICT tools and activities in our mathematics courses, we have noticed a much greater interest on the part of the students for mathematics. This increased when we incorporated Logo, and it became even greater when we began to engage on long-term projects such as the one presented in this paper.

In fact, we face very strong difficulties in making students leave the technology-based sessions, because they are so involved and enjoy their work there so much. When they do leave, they go out discussing amongst themselves their different or possible solutions and procedures.

The atmosphere and dynamics of the classrooms have also changed deeply. We have observed much improved classroom discipline and engagement, with a reduction of behavioural problems. Also students tend to get along better and they have developed values of cooperation and assistance amongst themselves: they appreciate the benefits of collaborative work.

Most importantly, through these technology-based activities, students are given the opportunity to find their own paths for reaching a solution. When they achieve this, they gain confidence in their own knowledge and abilities and this is highly motivating for them.

Remark on students' perceptions of the tools

In terms of the technological tools, the involvement in the project (as well as with other EMAT activities) have shown students the various uses they can give to tools like Logo, Cabri and Excel and how each of these tools complements the other. They have become proficient in the

use of all these tools. Furthermore, students no longer make a memorized “algorithmic” use of the tools, but they actually reflect on what they are doing, why, and how.

Final remarks

What we have achieved with our projects, is to make mathematics explorations, an enjoyable part of students’ life, where working towards finding a solution actually becomes a game for them –like it does for a real mathematician. We observed how they used names for their Logo procedures and objects that perhaps didn’t reflect what they were doing –we let them, because these objects were *their* toys, and the explorations became *their* games. And this is what we believe motivates learning.

Our students are now able to perceive mathematics as something meaningful, something that they can apply for solving a project; instead of something boring, meaningless and forced upon them –as is so often the case. Furthermore, they can use the technological tools as aids for developing their abilities, and learning how to think, instead of simply “clicking buttons”.

From our side, we also achieved to introduce what is normally considered a “too advanced” topic for Grade 1 students, and showed (as is seen in the sample tests in Figure 8) that children did learn and understand some of the ideas they explored: not only the trigonometry ideas, but also the use of algebraic variables and concepts. In its 40 years, Logo has been proved an excellent tool for introducing algebraic concepts and particularly the use of variables. In our project, we have again confirmed that. Whatever may happen, we are now at ease knowing that most of our students are proficient in the curricular topics of Grades 1 and 2, and beyond. Richard Noss told us once, that “knowledge has no age; it rather depends on how the teacher wants to teach it” (personal communication).

We feel that every mathematics programme should have some programming (or at least, constructionist) activities; in our case it was Logo; and its influence made us develop more constructive activities with the other tools. We also appreciate the conceptual perspectives that each of the tools we used, offered. Cabri, with its manipulable dynamic representations allowed us to explore properties of right triangles, such as the relationship between the sides, and between the sides and the angles; it also gave us a means to introduce the trigonometric circle and a basic understanding of some trigonometric functions. With Excel (and Logo), we delved deeper into the structure of the Pythagorean and basic trigonometric formulas and of the algebraic relationships between their elements. With Logo, we gained a further understanding of how and when to apply those concepts for solving a particular project: that of drawing a general right triangle. It was an orchestration of all the tools.

But it was not easy. We faced many challenges –e.g. from difficulties in using the computer labs, to learning how to adapt how we teach. It has taken us several years of experience working with technology in our mathematics classrooms for us to gradually change our practice, the way we teach, and the ways we perceive and use the tools. As is reported in Sacristán et al. (2006), one of the biggest difficulties in the implementation of the EMAT programme, has been teachers’ difficulties in adapting to the proposed pedagogical model and changing their classrooms’ dynamics. We also had those difficulties. A few years ago, we were not used to having children talk and discuss during class. Now, that has changed. But we have to be open to the possibility of change, before that can happen, and we understand how difficult that can be for many of our peers.

In fact, we faced many criticisms (and even obstacles) from some peers and authorities, when we began to develop our technology-based projects, because they didn’t understand what we were doing, despite the fact that the EMAT programme is sponsored by the government.

Again, change is hard. Many of our peers fear the use of technology because they don’t feel proficient enough (as is also reported in Sacristán, 2006); they also fear changing the way they were used to teach. We also faced those fears, but despite our insecurities, we took the risk and have enjoyed discovering we can learn at the same time as the students. Much later we

discovered that, in fact, Papert (1999, p. xv) believes “that there is such a thing as becoming a good learner and therefore that teachers should do a lot of learning in the presence of the children and in collaboration with them”.

Another fear that teachers have, is in engaging in such long-term projects because they feel they lose time that they need to cover the required curricular topics. What we have found out, is that many of the curricular topics naturally arise *from* the explorations. Projects, such as the one described in this paper, not only deal with the topic they are designed for; the need for the use of many other mathematical concepts also emerges¹ and we are reminded of Papert’s vision in his book *Mindstorms* when he described the gears of his childhood (Papert, 1980). This gives us the opportunity to introduce new mathematical topics as children encounter them, and *then* we can develop them. In this way, the students have a means to relate to the mathematics, and it becomes more meaningful for them. Thus, time is far from being lost; it is actually well invested in activities that are much more significant for students than the usual rushing through all the curricular topics.

We believe that in the future, if students learn to use technology in a thoughtful way (moving away from a mechanized use) for understanding, developing projects or solving problems they will become better learners and the teacher will have more of a mediating role between students and mathematics rather than being the traditional presenter of knowledge.

We are a new generation of students and teachers creating experimental environments, computational microworlds and classroom dynamics that are very different from traditional school practices. The catalyst for change are the technological tools, but it is in our power, as teachers, to actually make the change, and use the tools in innovative ways that change mathematical teaching and learning. The tools don’t bring the benefits; it’s the *use* we make of them that does.

In the words of Seymour Papert (1999, p. xv):

Opportunity means more than just “access” to computers. It means an intellectual culture in which individual projects are encouraged and contact with powerful ideas is facilitated.

Doing that means teachers have a harder job. But we believe that it is a far more interesting and creative job and we have confidence that most teachers will prefer “creative” to “easy.”

In that spirit, we continue working, as we have for the past 5 years, in developing interesting mathematical projects for our students with an integral use of technological tools like Logo, Cabri and Excel. And we hope that by sharing our experience, others will also dare change, use technologies in meaningful ways and perhaps also create their own projects, where there is collaboration and partaking of experiences amongst students, between students and teachers, and also amongst teachers, that enriches everyone’s learning.

And if a student of any age comes to us for learning trigonometry, or any other topic, we hope to be able to give him/her the means to learn it “painlessly”.

References

Balacheff, N. and Sutherland, R. (1994) *Epistemological Domain of Validity of Microworlds: The Case of Logo and Cabri-géomètre*. In *Lessons from Learning*, IFIP Conference TC3WG3.3, R. Lewis & P. Mendelsohn (eds), North Holland, pp. 137-150.

¹ In the project described here, as explained before, the students not only explored the basic trigonometric ideas, but it also allowed us to cover many curricular topics and beyond, from number operations to algebraic ideas, such as variables, use of formulas and manipulation of equations.

- diSessa, A. A. (2000). *Changing minds: Computers, learning, and literacy*. Cambridge: MIT Press.
- Harel, I., & Papert, S., (eds.) (1991). *Constructionism*. Norwood: Ablex.
- Papert, S. (1980). *Mindstorms: Children, Computers, and Powerful Ideas*, New York: Basic Books.
- Papert, S. (1999) *What is Logo? Who Needs It?* In Logo Philosophy and Implementation, Logo Computer Systems Inc. Pp. iv-xvi.
- Rojano, T., Sutherland, R., Ursini, S., Molyneux, S., and Jinich, E. (1996). *Ways of solving algebra problems: The influence of school culture*. In Proceedings of the 20th Conference of the International Group for the Psychology of Mathematics Education, Vol. 4. L. Puig & A. Gutierrez (Eds.), Valencia, Spain: Universidad de Valencia. pp. 219-226.
- Sacristán, A. I. (2003) *Mathematical Learning with Logo in Mexican Schools*. In Eurologo'2003 Proceedings: Re-inventing technology on education. Coimbra, Portugal: Cnotinfor, Lda., pp. 230-240.
- Sacristán, A. I., Ursini, S., Trigueros, M. and Gil, N. (2006). *Computational Technologies in Mexican Classrooms: The Challenge of Changing a School Culture*. In Information Technologies at Schools. The 2nd International Conference "Informatics in Secondary Schools: Evolution and perspectives" Selected papers, V. Dagiene and R. Mittermeir (Eds.). Vilnius: Institute of Mathematics and Informatics. Pp. 95-110.
- Ursini, S. & Rojano, T. (2000). *Guía para Integrar los Talleres de Capacitación EMAT*. Mexico: SEP-ILCE.

Acknowledgements

We are grateful for the support of our schools' authorities and of the following organizations in Mexico: SEP, ILCE (the EFIT-EMAT team), OEI, Conacyt (with the research grants: No. G26338S and No. 44632); as well as the encouragement of Professors Celia Hoyles and Richard Noss of the University of London, UK.

We also thank Seymour Papert for his inspiration and dedicate this paper to him.

How Not to Use Computers to Teach Kids

Brian Harvey, *bh@eecs.berkeley.edu*

Computer Science Division University of California, Berkeley

Abstract

In the US, all but the poorest schools have well-equipped computer labs these days. But the emphasis is on multimedia and Web search. In this talk I'll give an example, taken from my experience as a volunteer in an elementary school, in which this emphasis actually hurts learning, leaving kids with false and confusing ideas about the topic (in this case, the US Constitution).

The first problem is that the information found on the Web is drowning in a sea of misleading information not really relevant to the topic; kids would learn better if they found what they need either in a book (or even a Web page) specifically designed to inform kids their age, or were just told by their teacher.

The teacher, meanwhile, is much too busy helping kids figure out which button to click in Photoshop or PowerPoint to pay attention to what the kids are learning about the supposed content of the lesson. The software is much too hard to master, partly because of bad design and partly because it's designed for the complicated needs of graphics professionals rather than for the simple needs of students.

The moral is that just putting computers in a school doesn't guarantee good results. I'll compare good and bad uses of computers and draw lessons from what went wrong in this case.

Policy and Practice: a Logo Vision

Celia Hoyles, *c.hoyles@ioe.ac.uk*

London Knowledge Lab, Institute of Education, University of London

Abstract

In this talk, I will describe some initiatives in Mathematics Education in U.K. following my appointment as Chief Adviser for Mathematics to the Government. I will seek to draw out lessons I have learned from the Logo community in terms of the importance of: modelling to engage students with mathematics while respecting the rigour of the subject; finding appropriate means to express ideas to different communities; working explicitly to widen participation in mathematics and to break down barriers between mathematics and other subjects; and taking care to find ways to scale up interventions from design experiments to national initiatives. All such Initiatives must place the teacher at the heart of the process. So in particular I will report on the work of the National Centre for Excellence in the Teaching of Mathematics (see www.ncetm.org.uk) which has set up a national infrastructure for the continuing professional development for teachers of mathematics, which is responding to teachers' needs and goals including their use of ICT.

Imagine Logo based magazine

Daniela Lehotská, lehotska@fmph.uniba.sk
Department of Informatics Education, Comenius University

Andrea Hrušecká, hrusecka@fmph.uniba.sk
Department of Informatics Education, Comenius University

Abstract

Four years ago we started to develop an interactive on-line computer magazine called Infovekáčik. It is aimed at children at the age of 6-10 years and their teachers. It consists of different interactive educational but still interesting game-like activities for constructivist teaching and learning, collaboration, development of creativity, logical thinking and problem solving. Each issue is oriented to some topic and most of the activities of the issue are adjusted to that topic. The magazine consists of several columns. In this article we would like to introduce the magazine, its columns and the most interesting activities.

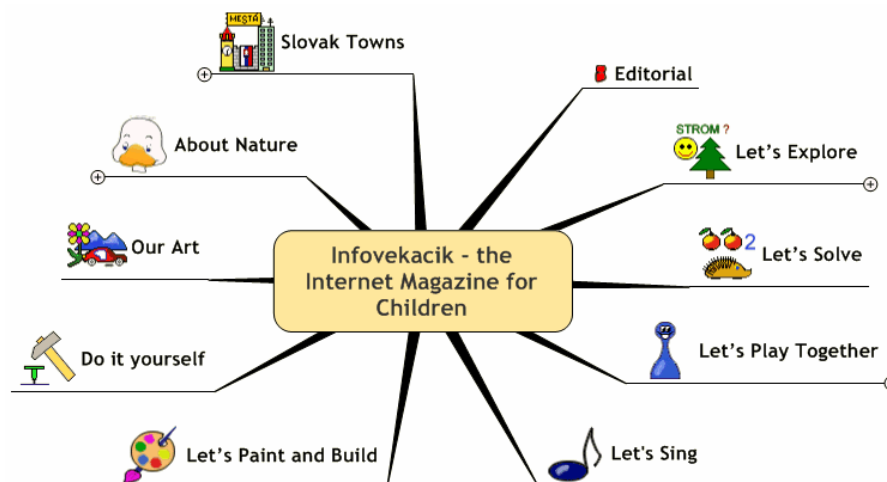


Figure 1 Mind map of the magazine

Editorial includes short text about the actual topic. In **Let's Explore** some term or phenomenon is explained in words and also using a microworld. **Let's Solve** offers various activities for sorting, classifying, matching and others. In **Let's Play Together** online activities enable cooperation of two children. In **Let's Sing** children can learn a song. **Let's Paint and Build** includes activities for creativity, like interactive building sets, decorate things, colouring pictures. **Do it yourself** gives children instructions for creating something by hand, like eye mask. **About Nature** includes specialities and activities dealing with nature. **Slovak Towns** describes nice places of Slovakia, gives tips for trips. **Our Art** is a gallery of pictures sent by children.

In developing the magazine we participate with two primary school teachers. Our inspirations are (beside our own ideas) Slovak textbooks and some game and activity books for little children.

All magazine activities are developed in Imagine Logo environment. Turtles and classes, events, dynamic programmable shapes of turtles, processes and net object are used mostly to program the activities. To browse the magazine, it is necessary to have the latest Imagine plugin.

Keywords

children, online magazine, interactive activities, creativity, constructionism, collaboration

Introduction

Children don't just have „to learn” new technologies but with their help they can better and more effectively learn everything else.

(Kalaš, 2007, p. 2)

To make teaching and learning more effective and interesting with the use of ICT we started to develop an interactive on-line computer magazine called Infovekáčik four years ago. It is aimed at children at the age of 6-10 years and their teachers. It consists of different interactive educational but still interesting game-like activities for constructivist teaching and learning, collaboration, development of creativity, logical thinking and problem solving. Each issue is oriented to some topic and most of the activities of the issue are adjusted to that topic.

The magazine is prepared by four members of our department and two primary school teachers. Sometimes also our students (future informatics teachers) help us with some ideas or simple programs. We can't forget primary school children which contribute their pictures, poems or stories. The activities we have been creating are inspired by our own ideas, by Slovak textbooks or some other game and activities books for little children.

Most of the activities in the magazine are developed in Imagine Logo programming environment. We have chosen Imagine Logo because in our department there is a long tradition in programming in this environment. Also our students have lessons in Imagine Logo programming and so we sometimes use their projects in the magazine. Turtles and classes, events, dynamic programmable shapes of turtles, processes and net object are used at most to program the activities. To browse the magazine, it is necessary to have the latest Imagine plugin.

We will describe all the columns and introduce some interesting activities in following parts.

Editorial

The first page of every issue of the magazine is editorial. Children can read short text about actual theme of issue there. Sometimes it is like a riddle – we don't reveal the title of the theme and children have to guess, what the issue is about (for example about winter, the book, the calendar). The reading is important activity for children. They can **develop their vocabulary** and **linguistic skills** by these activities.

Let's Explore

In the first three years the **Let's Explore** section was meant for K3 pupils. The goal of the section was to explicate three words not well-known for children. These words are connected to actual topic of issue -- we always choose them from editorial. We offer three possibilities of explanation for each word. The task for children is to choose the right meaning. If they choose the right answer, the picture which represents the word will be shown. The answers are often funny and children have no problem to guess the right answer, even if they don't know the word.

In present time we try to focus on explaining some term or phenomenon in this section. We give children a tool – microworld – simple simulation, which enables them **to “grasp” the term and play with its model, explore, change the important parameters**. For example, in the issue *Time* we developed microworld, which shows how sundial works. In another issue we developed environment, which shows how the country is changing, when the temperature is changing, especially about temperature 0°C. Children can control the temperature by the slider. In issue *Snow*, children can learn that the snowflakes can have different shapes and structure and can play with microworld, in which they choose from variety of snowflakes, change their scale and the speed of snowfall and then they can watch, how snowflakes change the country. In the issue *Musical Instruments* children can play virtual piano and observe the funny notes, which are put on the stave while pressing the keys. In the issue *My Room* children can read about cleaning the room and they can also try it virtually – they have to put the scattered things to the right places in

prepared microworld. In issue *Ice Hockey* they can try shooting the puck to the hockey goal with goalkeeper from different positions.

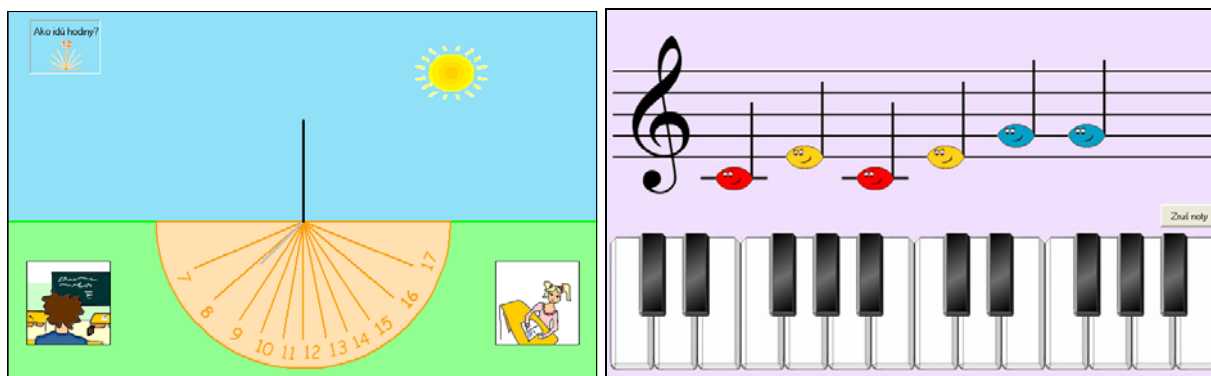


Figure 2 (a) Sundial, (b) Piano

Let's Solve

In **Let's Solve** column we can find various activities in which children can “prove” their knowledge and skills in mathematics, native language and sciences. Activities are oriented to classifying, sorting, looking for pairs, looking for all combinations etc.

Classifying activities

The aim of classifying activities is to categorize a set of objects into several subsets according to a given key or to identify objects with given properties. The tasks are for example:

- to assign fallen leaves to the right trees,
- to distinguish fruits and vegetables,
- to classify words to nouns, adjectives and verbs,
- to classify letters to vowels and consonants,
- to find all multiples of a given number,
- to classify traffic signs according to their geometric shapes.

Sorting activities

In sorting activities children sort objects according to a given criterion. For example they

- place bowknots on a kite tail according to their size or tint,
- sort the packets according to their weight (scales are available),
- change the word order to create a meaningful sentence.

Looking for pairs

In these activities the aim is to find pairs of objects that have some common property or they logically belong together, like:

- an animal and its young one,
- homonyms – pictures illustrating two different meanings of the same word,
- a state on a map and its name, capital, river, typical food,
- an animal and a simile/metaphor related to it (e.g. *as proud as a peacock*).

Combination activities

Examples of combinative activities are:

- to give out three ice scoops of three kinds to two (three, four, five, six) children so that each child has another order of ice scoops and nobody has two or three scopes of the same ice,

- to create all possible pairs of hockey teams,
- to look for all possibilities how to distribute one (two, three, four) ice cubes into two (three) glasses.



Figure 3 – (a) Sorting packets according to their weight, (b) Distributing ice cubes to glasses

Other activities

Other activities include tasks to find out how many objects are there in the picture or vice versa to collect a given number of objects, to balance a scale, to find an object in a set of objects which doesn't logically fit into the set, etc.

About Nature

The goal of the **About Nature** section is to familiarise children with nature and its inhabitants, to make children think about nature, pay attention to their surroundings, respect and take care of nature. The information and activities in this section can encourage them in forming relationship to their environment.

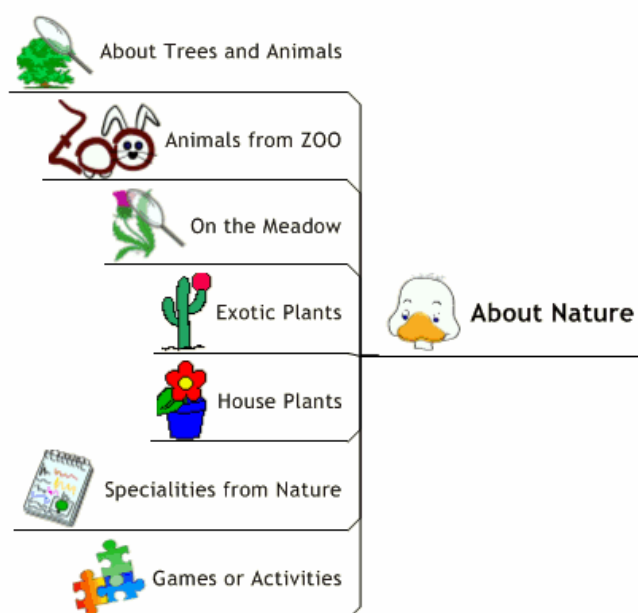


Figure 4 Parts of About Nature section

Children can familiarise with trees and animals which they can see in Slovakia, with animals from ZOO, with plantlets on meadow or in the flat... They can also learn remarkable things from nature such as how to get to the top of the oak, which tree grows fastest, which animals are the fastest, how the plants can protect. Some issues include simple games in this section. Some of these games can be used to repeat and practise the new pieces of knowledge, some are just for fun like composing the puzzle with a nature motif.



Figure 5 (a) Leaves and fruits, (b) Evolution stages of animals

Slovak Towns

This section is aimed mostly at Slovak towns and their surroundings. The goal of the section is to introduce our country to children.

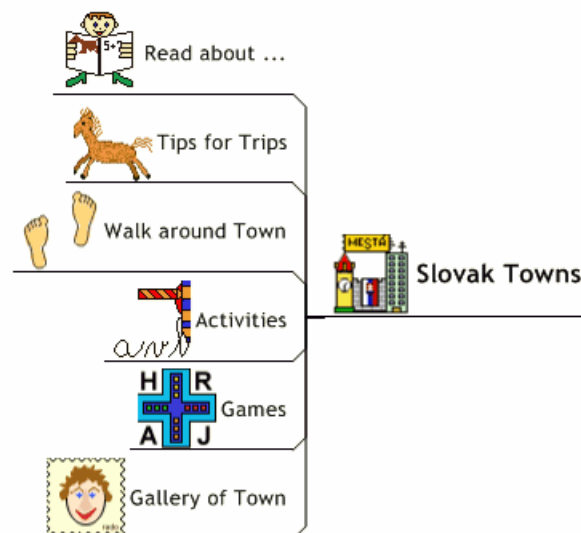


Figure 6 Parts of Slovak Towns section

In this section children can not only learn about Slovak towns, but they can find out how to determine cardinal points, which sights are written in the UNESCO registry, what are Slovak "best of",... In some issues they can find also various games and activities. Children can for example colour the town crests, fill in the right letters to uncover hidden picture, guess the names of Slovak towns, rivers, do a jigsaw.



Figure 7 (a) Colouring the crest, (b) Jigsaw

Let's Play Together

One of the modern ways of learning is collaborative learning. *Collaborative Learning is a philosophy: working together, building together, learning together, changing together, improving together* (Wieserma, N., 2000). Collaborative learning can be realised in different ways. In the magazine we concentrate on cooperation of children via the internet. Most of our net activities are intended for two "net players", but there is also a possibility to use it without the net connection (either a version for two children on one computer or a version for one child). Here are some examples of the net activities and suggestions of their usage for teachers.

Memory Game

Memory Game is aimed at enriching the English, German or Hungarian vocabulary related to some topic. One card in pair illustrates an object, the other one contains a picture of an object with a name of the object in a foreign language. We developed memory games related to, for example, fruit, trees, flowers, means of transport, toys, instruments etc.

Ludo

In our version of Ludo game the aim of the players is to get to the destination on the road full of obstacles. The obstacles can be overcome by answering some questions like in a quiz. If the answer is right, the player can go on, if the answer is wrong, the player has to stay one round.

Drawing activity

This is a simple activity enabling two children to paint in a shared paper and thus to create common works. Children from different places can for example paint with each other, what's the weather like in their countries, how a concrete season looks like in their countries, how they spent last holidays, they can together design a maze, dream house or dream car, paint a bogey, an alien, a clown or anything according to their fantasy.

Cards

The principle of the Cards activity is assigning titles to pictures – a word or a phrase. There is a set of pictures. For each picture two titles can be given – each player writes his/her own.

This activity can serve as a bilingual picture dictionary – each player writes own title for the picture in different language. It would be perfect, if the players were children from different speaking countries – they could teach each other own languages in this way. Other possibilities for application of this activity is looking for attributes of the objects in the picture (one child writes the name of the picture the other one an attribute), looking for synonyms, looking for diminutives.

Fairy Tale

In this activity two children can create a simple fairy tale or an interesting story combining words and pictures together via the internet. A set of cliparts and a tool for writing simple text is

available for them. Children put the cliparts and write pieces of sentences in the common “paper”.



Figure 8 (a) Cards, (b) Fairy tale

Mind Map

In this activity children are given a central concept (a word) and they look for and add other concepts which relate to the central concept together. Applications of this activity can be various: either we can add arbitrary words or we can concentrate to certain types of words, for example attributes of the central concept (adjectives), verbs related to the central concept, synonyms etc.

Associations

Looking for a concept (association) related to another concept is also the aim of the Association activity. For example *cheese* is an association to *milk*. Unlike Mind Map activity, where children look for words related to a central concept and thus create structures looking like a star (see Figure 9a), in Association activity children create a chain of associations (see Figure 9b), that means next concept always somehow relates to previous concept. Activity can be used to create various sequences, for example sequence of consecutive events, sequence of numbers, sequence of words such that next word starts with two last letters of the previous word (fish - shine - net ...).

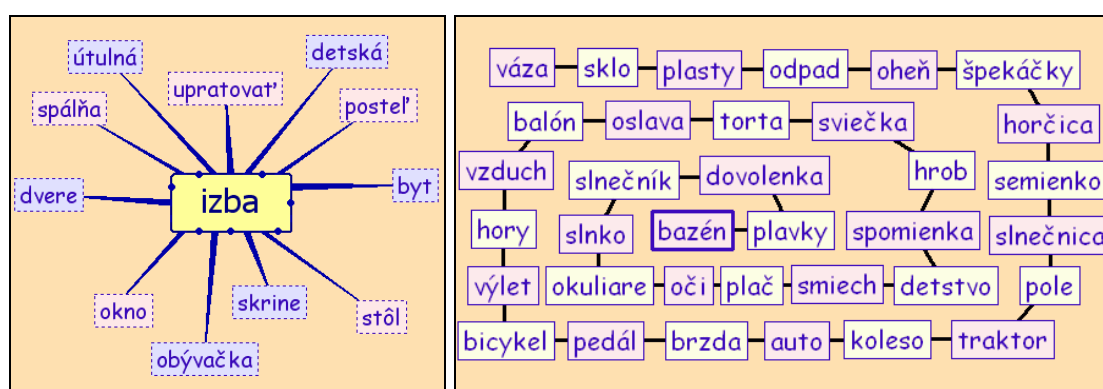


Figure 9 (a) Mind Map, (b) Associations

Guess a Trade

This game was developed for the issue called *Trades*. Player A thinks on a trade. Player B asks player A simple questions via the internet. Player A answers the questions, but he/she uses just *yes*, *no* or *I don't know* answers. On the basis of the answers, player B tries to guess the trade A is thinking on. If B guesses the trade within 10 questions, he/she is a winner – the roles of

players change (A will guess and B will answer) and the game can start again. Otherwise the trade is revealed and the game starts again with the same roles of the players.

This kind of game can be used for practising properties of geometric shapes, properties of animals and flowers, for guessing persons' properties, numbers etc.

Let's Sing

Almost every month children can learn a new song in the magazine. A note record of the song is presented and it is possible to listen to the melody of the song (the whole one, just a verse or a concrete note). While playing the melody the actual note is highlighted. The child can choose from instruments for playing melody (e.g. piano, guitar, violin, flute, bells). Songs are usually related to the topic of the issue or to the actual month.

Sometimes the section contains different musical activity instead of learning a song. There are activities where children can compose a short melody using some instrument or special subjects (for example icicles, screaming ghosts, singing frogs). In others children listen to short melody at first and then they have to repeat it.

Figure 10 Let's Sing: (a) Learn a new song (b) Repeat a melody

Let's Paint and Build

One of the most famous children activities is colouring a black-and-white picture. In the **Let's Paint** column we modify this activity. Children have to decode at first which colours to use to certain parts of the picture. In the black-white template there are just codes of colours, e.g. lower-case letter, arithmetical problem, geometrical figure or a digit. The second part of the code they can find in the palette, e.g. capital letter, result of the arithmetical problem, rotated figure or a Roman numeral.

Besides colouring pictures there are many activities for creating or building things in this column.

Research has shown that many of our best learning experiences come when we are engaged in designing and creating things, especially things that are meaningful either to us or others around us. When children create pictures with finger paint, for example, they learn how colors mix together. When they build houses and castles with building blocks, they learn about structures and stability. When they make bracelets with colored beads, they learn about symmetries and patterns. Like finger paint, blocks, and beads, computers can also be used as a "material" for making things – and not just by children, but by everyone. Indeed, computer is the most extraordinary construction material ever invented, enabling people to create anything from music video to scientific simulations to robotic creatures. Computers can be seen as a universal construction material, greatly expanding what people can create and what they can learn in the process.

(Resnick, M., 2002, p. 33)

One group of creative activities include activities with the character of decorating or stamping. First the child chooses one basic object (e.g. gingerbread, mug, traffic sign, clock face) and then decorates it with varied decorations (we can also call them stickers, stamps or cliparts). Decorations can be coloured and moved. Completed piece of work can be copied to another application through clipboard, where it can be printed or saved to a file (from safety reasons print and save operations are not allowed in Imagine web projects). Here are some examples of this kind of activities:

- creating an autumn collage from varied leaves, chestnuts, mushrooms etc.,
- decorating gingerbreads – gingerbread with the shape of circle, star, flower, house or snowman can be decorated with nuts, candies or coloured “sugar”,
- dressing Christmas tree with various kinds of Christmas decorations like balls, bells, candles, apples, nuts,
- decorating ceramics – a piece of ceramics (jug, plate, teapot, mug) is decorated by stamping varied ornaments (see Figure 11a),
- designing own traffic sign – using cliparts and basic templates of traffic signs it is possible to create signs like *No entry for scooters*, *Dance couples*, *Pathway for ghosts*,
- designing clock showing actual time – choosing a clock face, type of hands, font of numbers and kind of numerals (ciphers or Roman numerals),
- creating Christmas decorative chain lies in designing the chain shape and stringing beads or stars. Beads and stars can be strung “manually” one after another or a pattern for stringing can be designed – short sequence of beads and stars which are then automatically repeatedly stringed on the chain (see Figure 11b),
- building a dream castle – several kinds of walls, bricks, windows, gates and dungeons are available.

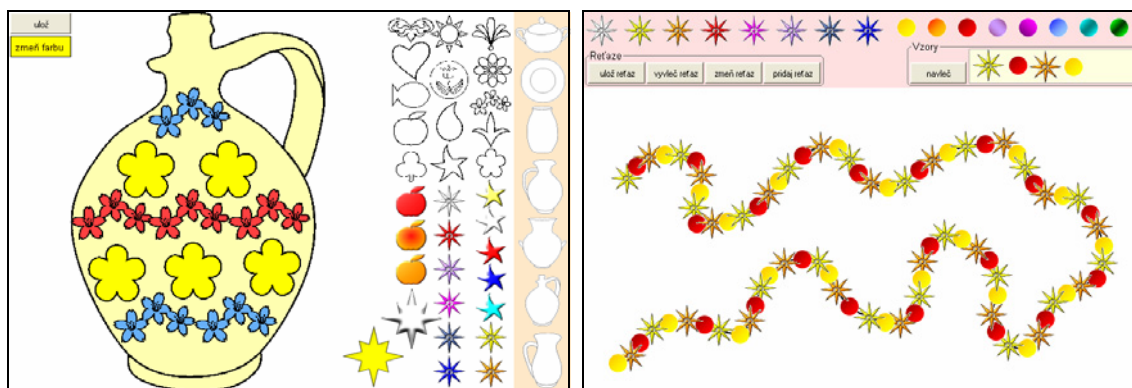


Figure 11 (a) Decorating ceramics, (b) Christmas decorative chain

Another group of creative activities can be called **interactive building sets**. Some of them are famous (mathematic) puzzles. In these activities children construct from varied pieces e.g. triangles, squares, pentominoes and solve given tasks. In many of these tasks rotation and reflection of pieces is necessary, sometimes also dynamic change of the shape (deformation) is used. Examples of these activities are:

- tangram – all tangram pieces, tools for rotating, flipping and colouring pieces are available. Children can either solve a given collection of tasks (pictures, which can be put together from the tangram pieces – see Figure 12a) or create their own picture,
- pentominoe – similar like tangram, children use pentominoes to cover given area or to build own pictures,
- tessellation – at first children create an “Escher tile” deforming a square. Then they use it to tile a plane either automatically (tiling is realized by computer, children just select two

- colours – see Figure 12b), or manually (they place tiles to each other, rotate or flip them if needed and finally colour them),
- building symmetric patterns in a square grid – children place little squares, circles or triangles into the grid. Simultaneously with the children's construction a horizontally or /and vertically flipped construction is created (depends on the setting). In another mode one half of a picture and an axis is given – the task is to complete the picture so that the given axis is its line of symmetry. Size of the grid can be changed.

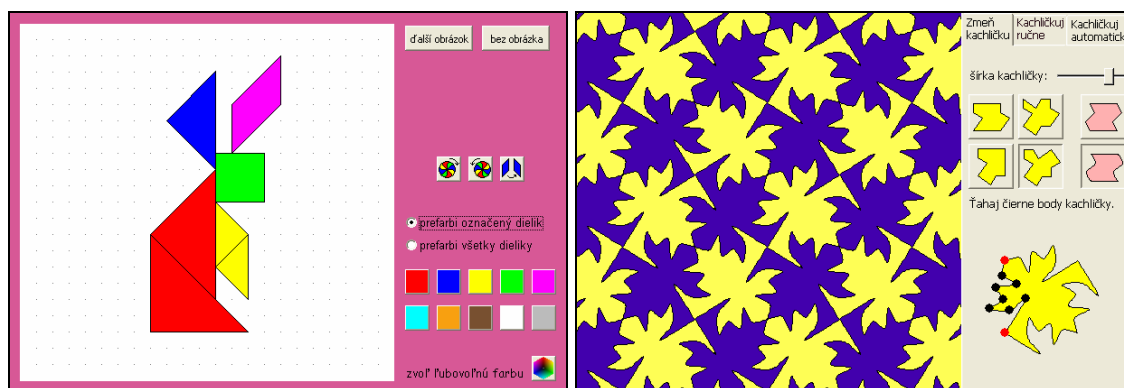


Figure 12 (a) Tangram, (b) Tessellation

Do it yourself

In this section we offer some suggestions for skilful children. Instructions how to fold different origami (cap, boat, frog, steamer), how to make a kite, a bookmark and a book wrapper, carnival masks, finger puppets, paper watch, how to make Christmas stars from narrow strips of paper, paper tulip, recipe for buns, gingerbreads and so on can be found here. One of the goals of this section is to improve children's fine motorics and creativity.



Figure 13 (a) Propeller with millhouse, (b) Picture with penguin

Our Art

In **Our Art** section children can present their pieces of art. It comprises paintings with actual theme (see Figure 14a), which young magazine readers send us. Sometimes we obtain also literary works like stories or poems which are related with actual issue. We were surprised and pleased, when the fourteen years old pupil sent us his musical activity that he developed in Imagine Logo (see Figure 14b).



Figure 14 (a) Our Art, (b) Musical activity

Our plans

In the future we would like to prepare two new sections – one about “programming” and second about brain-twisters. We would like to improve children’ algorithmic thinking by tasks in which children have to manage characters for example with icon commands. In section brain-twisters children will solve tasks for development of logical thinking for example simple sudoku, painted crosswords or they will solve different crosswords.

We also would like to aim at teachers and acquiring feedback from them. We would like to know, how they use particular sections with their pupils, which activities are most popular and which activities are missing in the magazine.

Conclusion

We believe that activities in internet magazine for children *Infovekáčik* contribute to constructivist teaching and learning of primary school pupils. Every month new activities and games appear on the web page infovekacik.infovek.sk, which children can use in schools with their teachers or they can play and learn at home. Children can contribute to the magazine with their paintings, stories or poems teachers can become involved in the development of the magazine by sending us ideas of activities or topics which will fit them for teaching.

References

Kalaš, I. (2007) *Formy rozvoja informačnej gramotnosti a informatickej kultúry detí (výskumná správa projektu Infant)*. FMFI UK, Bratislava.

Resnick, M. (2002) *Rethinking Learning in the Digital Age* [online]. In Kirkman, G. (ed.). The Global Information Technology Report: Readiness for the Networked World. Oxford University Press, 2002, p. 32-37. Retrieved March 28, 2007, from <http://ilk.media.mit.edu/papers/mres-wef.pdf>.

Wieserma, N. (2000) *How does Collaborative Learning actually work in a classroom and how do students react to it? A Brief Reflection*. Retrieved March 29, 2007, from <http://www.city.londonmet.ac.uk/deliberations/collab.learning/wiersema.html>.

Infovekáčik magazine (2003-2007). Retrieved April 2, 2007, from <http://infovekacik.infovek.sk>.

Logo in Lifelong Learning

Janka Pekárová, *pekarova@fmph.uniba.sk*

Dept of Informatics Education, Comenius University, Bratislava, Slovakia

Andrea Hrušecká, *hrusecka@fmph.uniba.sk*

Dept of Informatics Education, Comenius University, Bratislava, Slovakia

Abstract

In our paper we introduce curricula of on-line courses of programming in Imagine Logo environment. Courses were meant mainly for IT science teachers with small or no experience in programming.

We characterize successful participation in the course by following aims: Programmer

- defines and uses own commands for the turtle; recognizes situations where using parameter in command is appropriate,
- creates more turtles; changes shape of the turtle to the image or describes the shape by simple drawing-list,
- changes the behaviour of the turtles; turtles react on events or changing properties of their surroundings,
- controls turtles by launching processes,
- uses buttons, sliders and multimedia objects for enriching own projects.

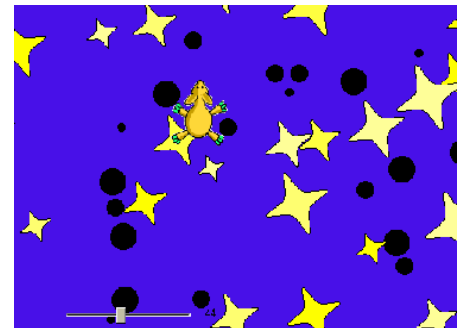


Figure 1 Milky Way project

We developed prototype of the project which contains all aims mentioned above – *Milky Way project*.

We describe ways that we have chosen to fulfil educational aims. We discuss several examples that we used for constructing basic programmer skills in Imagine – turtle geometry, repetition of commands, definition of own commands, reactions on events and processes for controlling turtle's movement. We tried to develop skill to read and understand unknown code, to analyse solution and to find principles on which solution is based. We consider these skills as essential for teaching of programming.

Latter we search for propaedeutic of advanced techniques of programming in Imagine Logo in our courses, especially recursion and object-oriented approach as described in Blaho and Kalaš (2002). We examine final projects of several participants and design alternatives for program. We explore knowledge authors of the programs missed that could help them to reduce their solution.

We deduce a few recommendations for teachers' training in Logo programming from our experiences.

Keywords

Imagine Logo, life-long learning, programming, analyse, object-oriented approach

Course background

We worked as designers and tutors of online courses in Logo programming in last two years. Courses were meant for the IT science teachers which had had small or no programming skills before. However, they used computers daily and acted as flexible learners in discovering new ways how to use computer – using computer to **design** own projects, to **share ideas** about own projects and to realize projects – **create** small programs in Imagine Logo programming environment. We used distance learning form – participants of the course used learning management system to access study material, tutors or other course members. This form of study fitted many employed teachers - they could study when they wanted and where they wanted.

Course content refers to Slovak version of Imagine Logo Primary Workbook by Blaho and Kalaš (2004), the textbook for pupils and extends it into introduction to advanced programmer's skills (e.g. recursion, object – oriented programming).

Units Order	Content of Unit	Level
1 - 6	<ul style="list-style-type: none"> Basic turtle commands Cycle, repetition of several commands [Polygon] Definition of own commands Using own commands to create new ones Commands with one parameter Reactions on click, on drag Creating new turtle using toolbar Addressing concrete turtle 	Beginner Imagine Logo Primary Book
7	Revision	
8 - 10	<ul style="list-style-type: none"> Image as shape of the turtle Frames, animated shape Launching process by every command [Naming and cancelling process] Simple conditions, testing colour of the page 	Beginner Imagine Logo Primary Book
11	Revision	
12	Creating turtle by command	Advanced
13	Drawing-lists as shape of the turtle	
14	Private variables of the turtle	
15	Revision	

Our role as tutors and designers in the course consisted of these responsibilities:

- defining educational aims of the course,
- design and implementation of study material to cover defined aims,
- checking progress of the participants,
- ensuring the quality of study material and quality of on-line study,
- development of learning and profiting community.

Educational aims of the course

We characterize successful participation in the course by following aims: Programmer

- defines and uses own commands for the turtle; recognizes situations where commands with one parameter are appropriate to use,
- creates more turtles; changes shape of the turtle to image or describes the shape by simple drawing-list,
- changes the behaviour of the turtles; turtles react on events or various properties of their surroundings,
- controls turtles by defining processes,
- uses buttons, sliders and multimedia objects for enriching own projects.

We merged defined skills into prototype of the final project – project *Milky Way*. Main hero of the game is yellow lamb – space pilgrim who is picking up the stars. Dangerous Black Holes are appearing from time to time in the sky – the lamb can lose in them.

We can find any of educational aims implemented in the project.

Programmer defines own command for creating Black Holes of various sizes.

Execution of the command is controlled by process.

Stars are turtles with their shape described by drawing-list. User can set their amount by slider at the very beginning of the game.

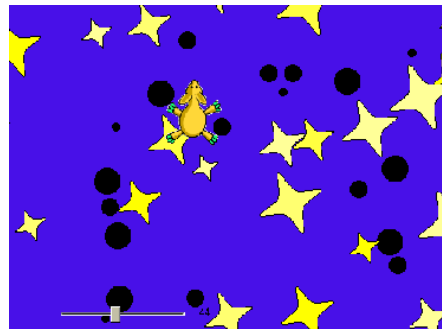


Figure 1 Milky Way project

Space pilgrim is the turtle with image used as its shape. Motion process controls figure's movement. User can change direction of the pilgrim by clicking it or by using direction buttons. When pilgrim meets the star, star will disappear (object will be destroyed). When pilgrim enters black point (piece of Black Hole), he will be destroyed and game will end (all user-defined processes stop).

In following part we describe ways we used to form programmer skills during on-line course of programming in Imagine Logo. They contain not only programs, but also stimuli for discussion among learners.

First ten units of the course cover and deepen content of Imagine Logo Primary Workbook. Teacher develops skills similar to skills pupils will have if the successfully work with the book. He becomes programmer – beginner in Imagine Logo. What do we offer him in our course?

Beginner level of programming

In the first half of the course we concentrate on development of skill to create and understand Imagine programs. Main activities include defining own commands, joining them to more complicated ones and design of first small games where turtle is controlled by buttons. Ability to

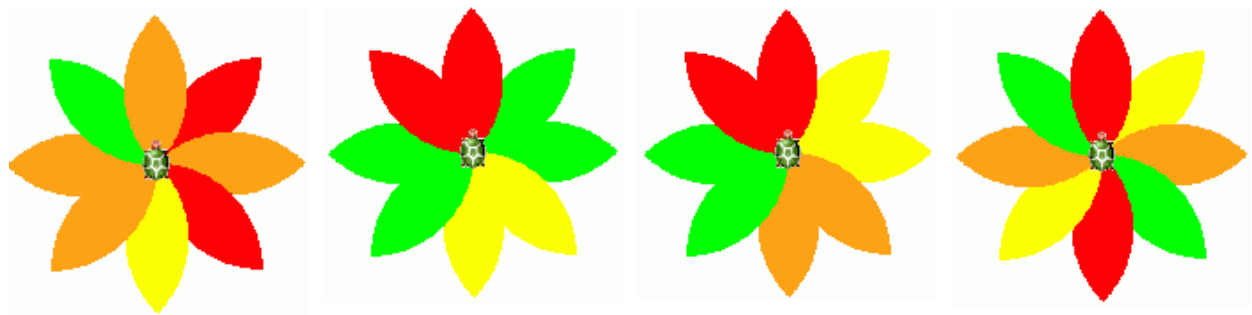
compose program from small pieces belongs to good practice among programmers and it is essential to build it from the first experiences with programming.

Teacher of programming shall dispense not only by knowledge of commands and several programming techniques, but also by perfectly-developed ability to analyse solution of pupil and recognize errors. We mean not only syntactic errors, but mainly errors in program logic.

We designed many test-like tasks where teachers could prove their comprehension:

- some tasks were concentrated on **skill to understand program**,

Which programs generate following drawings? Assign program to drawing it produces.



```
Repeat 2 [
  setPC "yellow
  petal right 45
  setPC "orange
  petal right 45
  setPC "green
  petal right 45
  setPC "red
  petal right 45
]
```

```
repeat 4 [
  setPC ?
  repeat 2 [
    petal
    right 45
  ]
]
```

```
repeat 8 [
  setPC ?
  petal
  right 45
]
```

```
setPC "yellow
repeat 2 [petal
right 45]
setPC "orange
repeat 2 [petal
right 45]
setPC "green
repeat 2 [petal
right 45]
setPC "red
repeat 2 [petal
right 45]
```

Teacher shall also analyse and accomplish simple programs:

We defined command `bedflower :B` – turtle will draw `:B` colourful dots. Number of dots is given by `:B`. Distance between two following dots is 40 points.

Complete the command `garden` – turtle will draw 4 bedflowers of various colours. Distance between bedflowers is 40 points. Turtle will return to start position after drawing all bedflowers.

```
to garden
  cs
  pu right 90
  repeat ...[
    setPC any
    bedflower ...
    back ...
    left ...
    forward ...
    right 90
  ]
  right 90 forward 4 * 40 right 180
end
```



Figure 2 Bedflowers

We also designed fill-in type of task in chapter *More turtles, more opportunities*. Teachers should fill in names of turtles according picture:

```
askEach [ _ _ ] [setPC "green]
askEach [ _ _ ] [
  setPC _ left 45
  repeat 4 [fd 50 bk 50 right 90]
]
ask _ [
  point 100 setPC "white point 70
]
ask _ [
  fd 50 right 90
  repeat 180 [fd 1 right 1]
  right 90
]
```



Figure 3 Addressing turtles

We wanted also to profit from specific character of children programming languages: from openness, visual attractiveness, clearness and talkativeness. Therefore we started to develop small games as soon as possible - after managing basic turtle's commands. We supposed that developing games would fasten teachers' motivation – they could create useful and funny products for children.

Easter – egg project is a kind of painting book where children choose one of variety of patterns, drag it and stamp it on blank Easter – egg following own fantasy. Teachers studying Logo programming should examine functionality of the project and suggest improvements/changes so that project better serves children painters' intentions.

Easter – egg project represents special kind of task – **explore** unknown project, notice scientifically its goals and functionality, then **express** opinion – design changes and finally **exchange** ideas with other members of learning community – three Xs which Harel (2003) finds necessary for children to thrive. In *Easter – egg activity* we find idea of three Xs applied into the world of adults. Fisher (2003) mentions: "When other learners are engaged in answering questions, learners explore and examine concepts more, **think** more and **share** more".

What do beginning programmers learn in this kind of activity? They synthesise pieces of own knowledge, judge the program and try to find solutions of problems observed.

Participants of the course designed valuable modifications of the project:

- adding new button to the project. User can set different colour of blank Easter-egg by pushing the button;
- editing patterns so that user can drag whole patterns also by holding transparent part of patterns;
- patterns should be stamped only on the egg, not in its surroundings.

Inclusion of third suggestion into project leads to the need to distinguish colours – colour of the egg and its surroundings. Project can serve as introduction to simple conditions – situations in which turtle changes its behaviour according to colour of the page.

That's why we re-used and extended *Easter - egg activity* in chapter focused on "deciding" – turtle's reactions to various situations and designed another small project – project *Design Your Clown*. Idea of the project is the same – user can drag and stamp some pattern (eyes, nose, butterfly) to the clown's face and clothes. User can stamp part of face (eyes, mouth, nose) only to beige face of clown. If he tries to stamp it elsewhere in place with different colour, pattern will automatically return to green stamp bar – to home position.



Figure 4 Design Your Clown project

Introduction to advanced programming skills

Object – oriented Approach

Blaho and Kalaš (2002) designed several steps to develop understanding of object oriented metaphor:

1. Elementary Logo for Single Turtle.
2. Elementary Logo Activities for Multiple Turtles.
3. Multiple Turtles with Private Information.
4. Clones of Objects.
5. Instances of Instances.
6. Hierarchy of Several Layers.
7. User-defined Classes.

We implemented first four steps of mentioned approach in the whole curricula of the course and designed several projects with the goal to support using of object-orientedness of Imagine Logo.

[Elementary Logo for Single Turtle] *My own boardgame.*

Using one turtle, programmer can draw plan of boardgame. Extending command for drawing the plan by length parameter causes easy scalability of the plan.

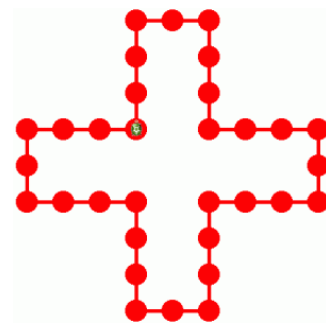


Figure 5 Plan of the boardgame

[Elementary Logo Activities for Multiple Turtles] *Living Picture* project. Programmer creates three turtles named **settler**, **kelpie** and **forester** by direct manipulation (using New turtle button from the tool-pane). Each turtle reacts on click in special manner – forester plants (i.e. stamps) trees, kelpie stamps fish or frogs and settler creates stones and people.



Figure 6 Living Picture project

[Clones of Objects] Project *Typewriter*. Programmer creates one turtle as prototype – letter. He sets automatic dragging to it, sets its pen up and changes turtle's behavior – stamping and returning to home position. Then he creates several copies of the prototype – clones with same behavior. After changing frames of clones typewriter is ready for use.



Figure 7 Typewriter project

[Multiple Turtles with Private Information] Project *My Tones*. After loading the project several notes discover on the page. Each note stands for one tone. When clicking the note, note is enlarged and its tone is played.

Each note represents member of the special family of the turtles. The family is characterized by special note-like shape of turtles and reaction on click – resizing the shape.

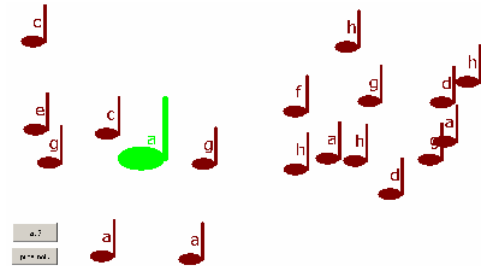


Figure 8 My Tones project

Concept of private information which turtle stores should be quite familiar also to beginners. As soon as they work with more turtles, they learn that each turtle has its own settings – pen, position, shape. We can deepen awareness of private information by using them in extraordinary way. Project *Frogs* illustrates this idea:

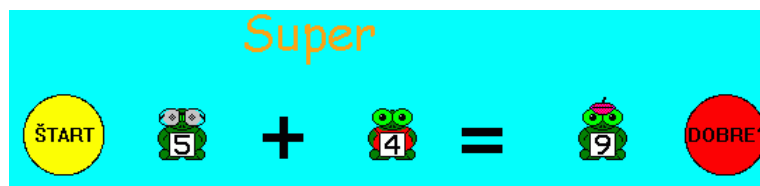


Figure 9 Frogs project

Project contains three frogs with random numbers on their T-shirts. User changes number of first and second frog by clicking it. His task is to achieve number of the third frog as the sum of the first two frogs' numbers.

How to find the sum? Clue lies in using number of frame of frog shape. Image of the frog consists of nine frames – each frame contains frog with different number. Frame number equals number on T-shirt of the frog. We can therefore provide checking sum very simply:

```
ifelse frog1'frame + frog2'frame = frog3'frame [label "Super"][label "Nooo.]"
```

Project *Frogs* inspired math and IT science teacher to extension to more analytic type of task:

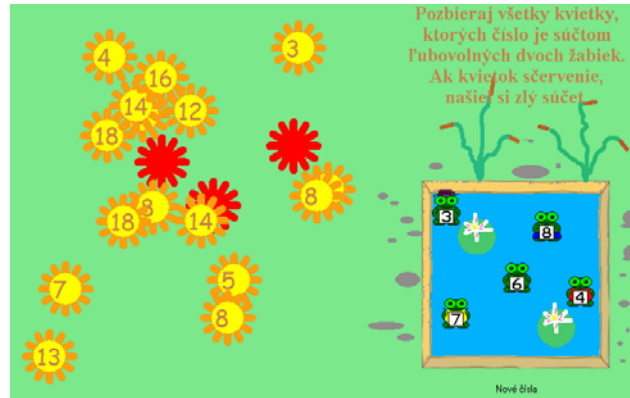


Figure 10 Extension of Frogs project

User shall click on each flower which is equal to the sum of any two numbers of the frogs in the pond. In the pond there are four frogs living. When user clicks incorrect flower, flower gets red, otherwise it disappears.

Teacher thought out simple solution – she enumerated all combinations of frames of frogs when clicking the flower. We find this solution appropriate to teacher's knowledge of programming and relatively small number of combinations. If she wanted to find more clever solution, she would possibly work with list where all possible combinations would be stored. This list shall generate when creating frogs. Other solution is going through list of frogs every time when any flower is clicked and sequential check of the sum (we would at first check all possible sums containing frog number 1 and another frog, then frog number 2 and another frog etc.)

Although participants of the course didn't learn anything about classes and instances of instances, we tried to prepare them for future understanding of these concepts. In project *Sort Numbers* child differs between two types of numbers: odd and even. Task of child is to put all even numbers into special box. If he tries to put odd number to the box, number jumped back.

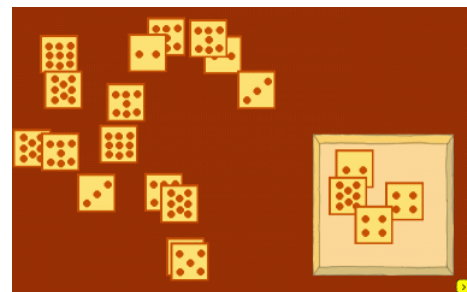


Figure 11 Sort Numbers project

In programmer's point of view: we create two “families” of objects, group of odd numbers which react on dragging by obstinate return to home position and group of even numbers which distinguish whether they are in the box or not (if not, they jump to home position).

Project can be re-programmed to object-oriented approach easily – families will be instances of two types of object – one object represents odd number and the other even number.

Recursion

Recursion comprises very powerful tool in Logo world, for instance

- combination of recursion and turtle geometry can produce magnificent effect because programmer can easily describe complicated shapes as fractals (as Foltynowicz and Walat (2005) show).
- recursion can be used to process some information, as method of special data structure – we can describe family of turtle having special properties and settings in lists and then compile the information by simple recursion:

```
make "myfamily [[Stan [0 180] 90 "taupe] [Hilda [0 90] 180 "green] [Clare [0
0] 270 "orange]]

to createNewFamily :family
  if empty? :family [stopme]
  new "turtle ![
    name (first first :family)
    pos (item 2 first :family)
    pencolour (last first :family)
    heading (item 3 first :family)
    shape [fd 50 point 30]
  ]
  createNewFamily butFirst :family
end
```

We – tutors discussed inclusion of recursion into course content. In our opinion deep understanding of recursion as well as design of own data structures for projects goes beyond beginning programmer's level. For this reason as well as limited number of units and our decision to cover the content of Imagine Logo Primary Book we haven't created unit engaged specially to recursion. Another reason is that solution of the problem by simple form of recursion -tail recursion can be easily replaced by processes in Imagine Logo.

However, we used several examples which can be used propaedeutically in understanding to recursion. These examples aim at simple form of recursion – tail recursion. Although concept of recursion is not mentioned in the assignments, discussed solution contains anticipation of developing this concept latter. Popular examples of this kind, also used in distance course are so called “carpet patterns” which Hrušecká, Kalaš (2006) introduces:

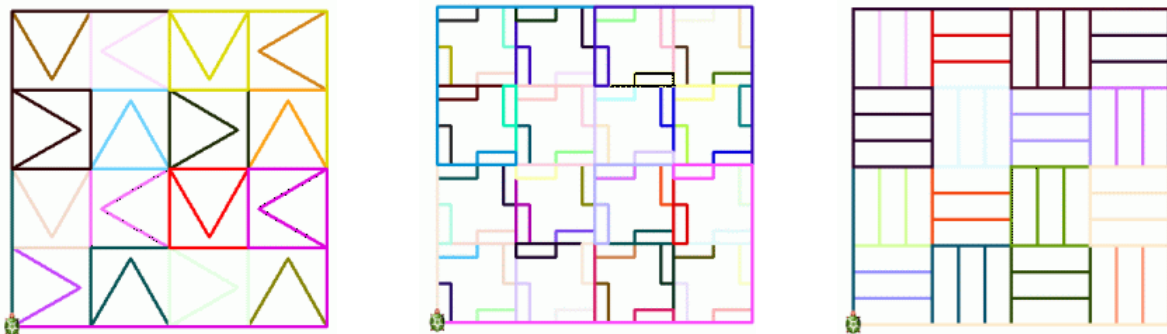


Figure 12 Carpet patterns

One pattern creates basis of each carpet in the picture. Pattern is repeated 4-times and forms small square. Turtle moves and forms greater square by drawing four small squares.

Pattern of first carpet consists of square and triangle of random colour:

```
to myPattern
  setPc any
  square 60
  triangle 60
end
```

Four patterns form small square...

```
to smallSquare
  repeat 4 [
    myPattern
  ]
  fd 120
```

...and four small squares form bigger one.

```
to bigSquare
  repeat 4 [
    smallSquare
  ]
  fd 240
```

```
    rt 90
  ]
end
```

```
    rt 90
  ]
end
```

Beginner can notice application of same rule in drawing big square as in small square. The only difference is the distance the turtle moves after drawing the pattern – it's always size of square multiplied by two. Definition of giant square consisting of 4 big squares should be very simple.

How to describe our solution without need to define new and new commands which differ only by distance in forward command? We can discuss re-using same procedure with different parameter and in this way discover tail-recursion as simplest form of recursive calling.

Participants of the course evaluated carpet patterns in different ways:

"I admit these patterns shocked me a bit at first but I managed all three patterns according to solved example."

"I consider carpet patterns task most difficult of all. I don't like drawing such complicated shapes." Same student adds: "I liked all tasks in unit". Difficulty doesn't obviously eliminate attractiveness of the task. We believe that course participants will naturally discover power of recursion in future.

Becoming Game-maker

Milky Way project became base of *Ecology Project* – final project of K5 teacher of nature science and technical training. Her game represents food chain – stork catches mice which are fed by grain – some of grain is poisoned by pesticides and some not. Stork – disguised space pilgrim - doesn't know which mouse will die because of eating poisoned grain. Its task is to pick up all mice. Mice are moving all the time and pick grain; player cannot affect their choice of grain. That's why player never knows result of the game – whether stork survives or not.

Teacher tried to design all figures of the game – grain, mice and stork – as turtles. She soon discovered serious obstacle of her access – when stork tests overlapping with other object, programmer needs to distinguish whether it is grain or mouse. In this point she needs some data structure to remember type of object. One solution hides in user – defined classes *Grain* and *Mouse*. When using object-oriented access, testing of overlapping is quite simple:

```
if stork'overlap? allof "mouse [
  ; stork remembers count of eaten mice
  stork'setmouse  mouse + 1
  eraseObject allof stork'overlapList allof "mouse
]
```

However, teacher as beginning programmer could not define own classes and found recognition of types as difficult problem with her knowledge. Another solution of the problem is storing names of mice and grain objects in lists as basic data structure of Logo. Work with lists as abstract data types in contrast with direct manipulation with turtles exceeds beginner level of programming. So does applying a procedure or procedural object – programmer can store information about type of object in private variable and use standard Imagine Logo *map* procedure to distinguish between objects.



Figure 13 Ecology Project

These reasons led us to combination of turtle's drawing and objects of one type displayed on page. Teacher applied our access successfully in *Ecology Project* – red poisoned and healthy yellow grain are coloured points and stork can safely test overlapping with all turtles – the other turtles living on page are all mice.

Design of *Milky Way* project illustrates limits of course content. Teachers use “families” of turtles with similar properties and behaviour, but don't know effective way how to recognize between different families. Their own game-making and practice in Imagine programming will perhaps bring the need for advanced object – oriented level of programming in future.

Conclusion

Which skills and abilities determine advanced level of programming? Where do bounds between beginning and advanced programmer lie? We try to find partial answer to this question:

- all programmers should understand at least own code of program,
- all programmers should compose their program from smaller part,
- design of data structure brings new possibilities for extending own program. Still, with no knowledge of manipulation with data structures we can design attractive and constructivist microworlds in Logo programming environment,
- object-oriented approach seems as natural solution to variety of problems. However, deep understanding of this approach requires some time. We evaluated one of numerous ways how to start with the concept of object.

Mastery of teacher does not mean to show all we know, but also in secret: “I know special tricks how to solve your problem quickly and more effectively...but would my student understand it just now?” Understanding of the children isn't given all the time; learning is process of developing understanding.

That's why we would recommend to split course content into two levels. Last five units can serve as starting point for building advanced programmer skills in Imagine Logo world.

References

- Blaho A., Kalaš I. (2002) *Object Metaphor Helps Create Simple Logo Projects*. In Proceedings of EuroLogo 2001, A Turtle Odyssey. Linz, August 2001. pp. 55 – 65.
- Blaho A., Kalaš I. (2004) *Imagine Logo Primary Workbook*. Logotron, Cambridge
- Foltynowicz I., Walat A. (2005) *Fractal Variations*. In Proceedings of EuroLogo 2005. Warsaw, August 2005. pp. 33 – 43.
- Fisher M. (2003) *Designing Courses and Teaching on the Web*. Scarecrow Education, Oxford.
- Harel I. (2003): *Learning Skills for the Millennium*. The Three Xs. Online http://www.mamamedia.com/areas/grownups/new/21_learning/three_xs.html
- Kalaš I., Hrušecká A. (2006): *Programming in Imagine environment* (in Slovak), Metodicko-pedagogické centrum, Bratislava.

Multimedia application in teaching computer skills

Ingrid Nagyova, *ingrid.nagyova@osu.cz*

Dept. of Information and Communication Technology, Ostrava University of Ostrava

Jana Hruskova, *L05851@student.osu.cz*

Dept. of Information and Communication Technology, Ostrava University of Ostrava

Abstract

One of the essential principles of PC working are the algorithms, which are also the basis of the main idea of a computer. Therefore those should be a part of every kind of PC working-learning – of course in a hidden, spontaneous way. And it is not important whether the students master the basics of algorithms through the work with a text editor, internet or multimedia.

The Imagine Logo environment enables working with multimedia – with pictures, sounds, melodies and videos. The particular multimedia elements become objects which can be changed and set up, moved, run, stopped etc. using various methods. Within creative playing with them various multimedia applications can be created. We will try to demonstrate the way it is possible to use this playing and multimedia applications creation to introduce the basics of making project and algorithms creation or PC working to the students.

We start with visual applications creation – setting up turtle parameters, animations and making the turtle move. Next we add a sound and we try to synchronize it with a picture. Then it's the work with video, which is integration into the general conception of a created application that follows. In the end we create a multimedia work on the basis of a nursery rhyme.

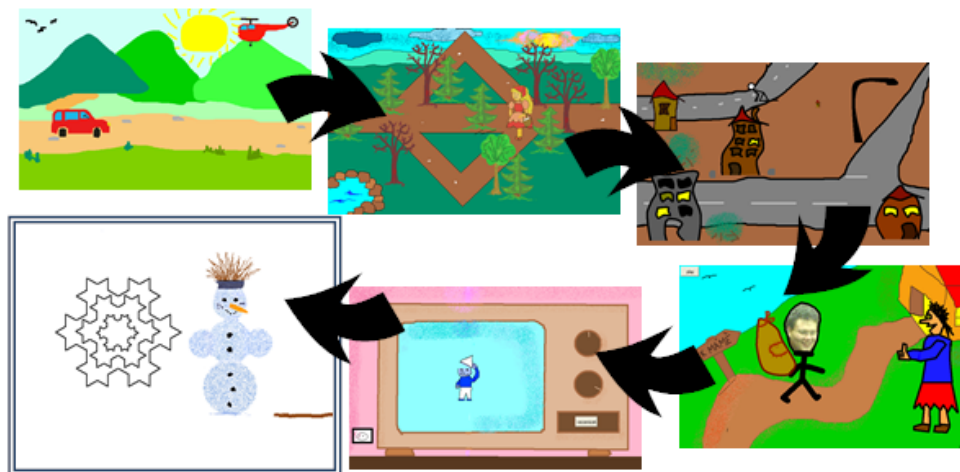


Figure 1. Scheme of multimedia applications creation

The created multimedia outputs can be afterwards used as means of the information and communication technologies integration into the process of non-information subjects' education. We will show you a way of using the multimedia application in religion education and introduce the set of the biblical stories explaining the individual symbols we meet commonly in religion.

Keywords

Algorithm development, ICT, multimedia, Imagine Logo

Introduction

We live in an information society and changes, which are brought into society life by information and communication technologies (ICT) are of such an importance that they influence all spheres of our lives, including education. The goal of ICT integration into education, though, is not only to teach pupils how to work with a computer, but to change the teaching methods through education informatization according to the requirements of the society development and scientific and technical development.

Searching for ways and possibilities of the computer utilization in the educational process occupies many both educators and ICT specialists. The whole range of concepts are being developed and enforced. They come from different theories and styles of teaching and offer possible solutions. These concepts further influence not only incipient educational programs but also the way of their utilization in the educational process.

As well as during solving most of the problems, it shows most suitable to try to get to the point of a taught topic or a process and understand its inner structure. So if we are looking for ways and possibilities in teaching basics of the work with a computer, it is necessary to look for the core and principles which the idea of a computer comes from and the work with a computer is based on. The effect of this searching will enable us to look for images of these ideas and principles, suitable pre-concepts (Papert, 1980). These will further lead students to the construction of needed knowledge, in our case to gaining knowledge and skills of working with a computer.

The question is: What is the core of a computer and what principles is the idea of a computer based on? What skills and knowledge make the basis of the work with a computer?

If we look to the history when looking for these answers, we realize, that originally it was inevitable to know the computer language (computer code, assembler, later it was higher programming languages - Basic, Pascal and the like), how to define a sequence of commands in it, principles of an algorithm creation and how to program to work with a computer.

These days the knowledge of programming is not required for the work with a computer anymore. Nevertheless, during more detailed examination of this work, we realize, that we are still defining a sequence of activities for a computer, deciding about their mutual succession and repetition. Everyone who does any work on a computer uses the basis of algorithm development as well. The language for this communication isn't programming language anymore, but direct operations and relations between them.

Algorithm development is one of the basic principles necessary to manage the work with a computer. But still it is not important whether we master the algorithm development basics through work with a text editor or for example work with multimedia. But it is obvious that work with multimedia is the one more interesting and more attractive for pupils and students. Below we mention a way and a process of teaching the students create multimedia applications in Imagine Logo environment – we went from the book (Blaho, Kalas 2004) – and familiarizing them with foundations of design and algorithms creation during these activities.

Process of teaching

1st Stage: Animation and turtle parameters

We start with a creation of a simple animation of a selected mean of transport by spinning its wheels in the bitmap editor Logo Motion. Students draw the background of the traffic situation (countryside with a road) afterwards and they place their mean of transport in it (turtle). By setting the parameters of the turtle (shape, direction, drawing pen and the like) and putting it in motion (so far only rectilinear motion) the first simple animation is created.

For this application not to lack an idea, the mean of transport changes during its ride into a flying object (see picture - transformation of the pram into a fish).

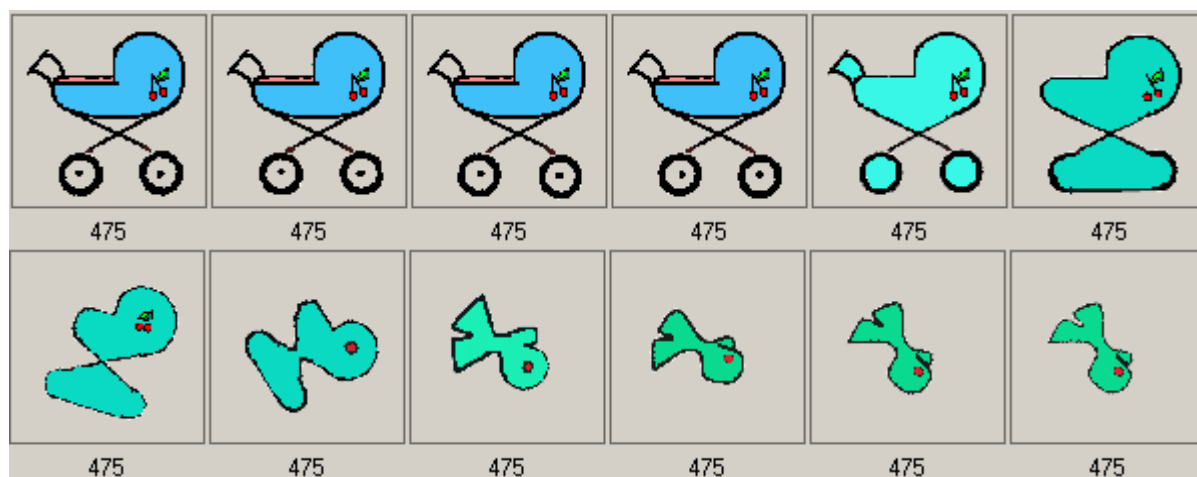


Figure 2. Animation – transformation of the pram into a fish

2nd Stage: Turtle movement and overlapping

The basic feature of a turtle in Imagine Logo environment is a possibility of its movement. Turtles can move forward, backward, turn left and right. Turtles can be “disguised” into the Fairy Amálka, who, she is struggling through the thick forest makes quite complicated movements on the screen: turns left and right, goes round obstacles etc. By adding the requirement of the right movement of the Fairy among trees (turtles) a project with a goal to solve not only the movement but also the overlapping (setting the level of illustration) is created – see figure nr. 2.

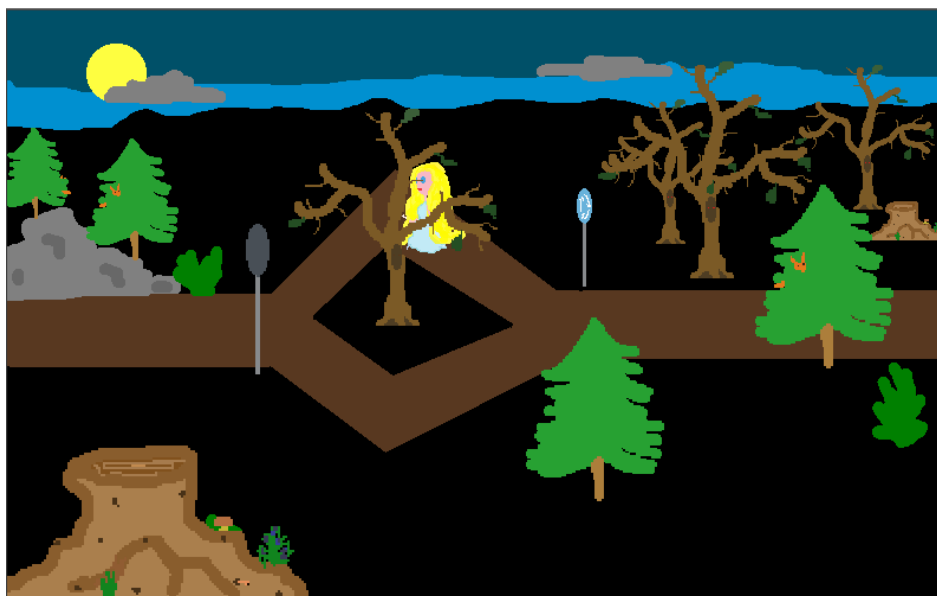


Figure 3. Fairy Amálka in a dark forest

3rd Stage: Working with sound, picture and sound synchronization

Sound tracking of created animations and moving pictures is taught by synchronization of changes of the turtle shape (the turtle changes its shape in numbers) and the corresponding sound recording. The main principle is cutting the recorded sound into pieces, which correspond to the individual pictures (phases of the animation). The final creation will be developed as a result of setting time delays between the individual pictures and the corresponding sounds.



Figure 4. Pictures of numbers completed by sound recordings

For sound recording, its modifications and editing we use Audacity sound editor, which is freely available and easily controlled. Imagine Logo environment enables to play created sound files in WAV format.

4th Stage: Creation of a multimedia work on the basis of a nursery rhyme

Previous teaching leads to a creation of a multimedia application on the basis of nursery rhyme. The students select a nursery rhyme, create necessary animated pictures and record the sound of the selected nursery rhyme. They cut the sound according to the verses. Then they create background for the nursery rhyme and gradually enter individual animated pictures as turtles and define their movements. All of these have to be perfectly timed and set.

By defining and setting the activities of the individual turtles, setting and changing their parameters in a certain time sequence, students get to know the project and creation of the algorithms in a natural way. In such a way they learn about the basic principles of a computer and learn how to work with it.

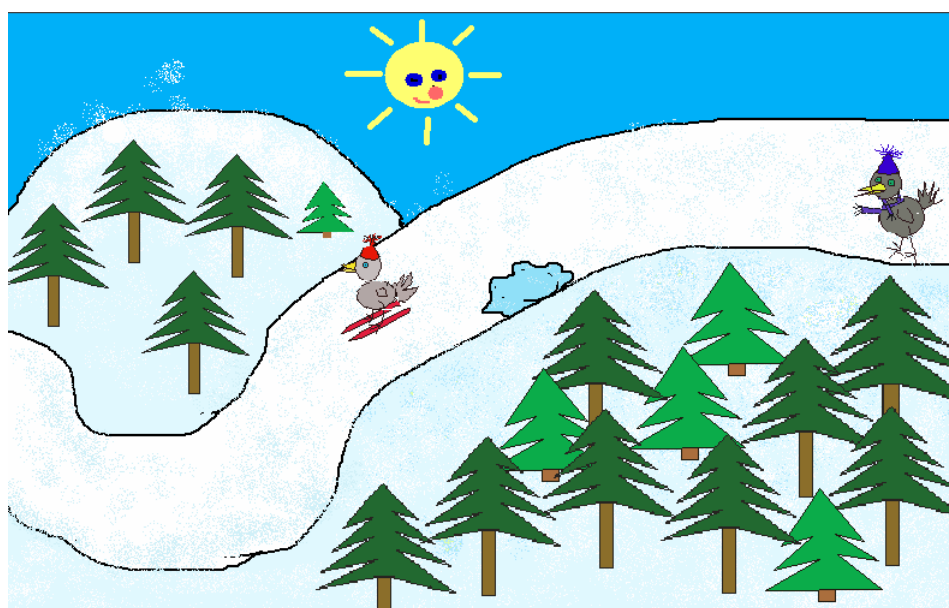


Figure 5. The multimedia application on the basis of a nursery rhyme

Application in teaching

The students can use the obtained knowledge and skills for the design and the creation of multimedia applications useful directly in children education within their work experience and other school activities. The created pieces can be used for the motivation of children and during the relaxation part of the class (especially with younger children). They can also help to start a discussion and make the class work more interesting.

The multimedia pieces of the Biblical stories set made in Imagine Logo helped in the religion education. The individual stories explain the symbols we commonly meet not only in religion, but also in everyday life – see the table.

Symbol	Story
Light	Saul's conversion
Goblet	The legend of the goblet
Water	Conversation at the well
Faith	Jób
Bread	Feeding
Lamb	The only son sacrifice
Shepherd	The story of David
Hope	The meaning of life
Bible	About dwarfs

Table 1: The list of symbols and corresponding biblical stories

Each of the ten parts of the set has the same structure:

- a short introduction of the problem – a short fragment of the Bible or a short thought.
- the particular story according to the symbol. The symbol either takes a part in the story or is described indirectly. To increase the motivation and concern of the student and for his better imagination and ability to bear the thoughts the indirect explanation of the symbol was mostly used. For example: "With hope it is like when..."



Figure 6. The David and Goliath fight

- after the end of the story the test questions are displayed to stimulate a discussion and help the student remember the topic (symbol).

The biblical stories set helped the religion education at the family church service for the children of the early pupilage. In the future we are planning to include it in the first stage of basic schools class work.

Conclusion

It stands to reason the programming makes possible to create a new things. By means of the multimedia applications creation we try to show that programming may be creative and enjoyable playing, that may be to promote up to an art. Such work is able to be source of instruction, but also can bring pleasure and gladness.

Multimedia application examples in Czech language can be found in the web address <http://www1.osu.cz/home/inft1/index.htm>.

References

Blaho, A. and Kalas, I. (2004) *Imagine Logo Primary Workbook*. Logotron, Cambridge

Papert, S. (1980) *Mindstorms: Children, Computers and Powerful ideas*. Basic Book. Inc. New York

ICT competition for kids

Ivailo Ivanov, iivanov@fmi.uni-sofia.bg

Dept of Information Technologies, Sofia University, 5 James Baucher Str., Sofia, Bulgaria

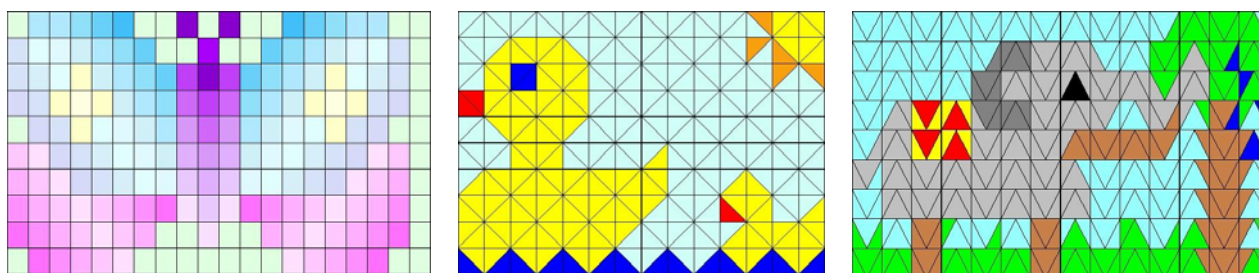
Vessela Ilieva, vessela.ilieva@abv.bg

Private Language School "St. st. Cyril and Methodius", Sofia, Bulgaria

Abstract

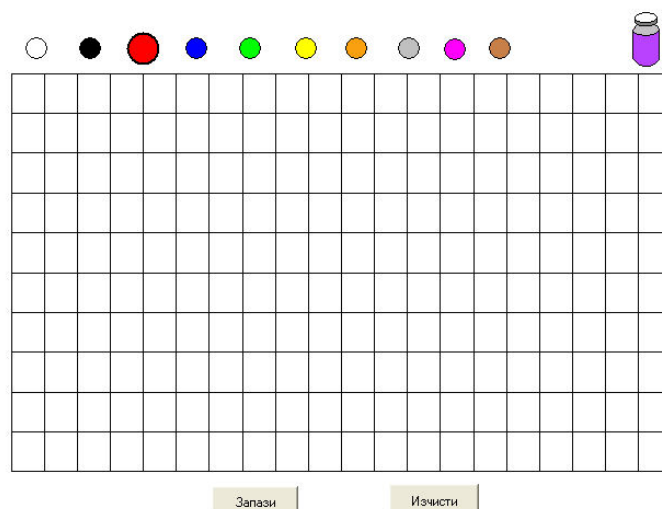
The first National IT competition in Bulgaria for children from 1st to 4th grade was held during the 2006/2007 school year. The competition was held on-line at a specialized site: <http://it-kids.eu>. Pupils were divided into three groups according to their age. The day before the competition the students had to register on the web page.

The task was coloring a picture. For example, in a network formed by geometrical figures of the same kind the kids had to create a shape or a complete image. The variations in the different age groups came from the different types and complexity of the network, given to the students.



After the end of the contest, all problem solutions were published on the web site to be voted on. Only students, from the corresponding contest groups, who were registered for the contest, were allowed to vote and every one of them was allowed to choose the 5 best solutions.

At the beginning of 2007 an adaptation for Bulgarian language was created on Imagine. This adaptation in addition to Plug-in for web pages allowed its use as a platform for development of software solutions for the competition.



Keywords

Primary school; competition; internet; Imagine

The Preconditions

As a part of the National ICT Educational Strategy during the 2005/2006 school year all the schools in Bulgaria were equipped with one or more computer labs.

During the 2006 / 2007 school year IT became an obligatory subject for all the students from 5th to 7th grade and elective for the students from 1st to 4th grades. In this respect the Ministry of Education assigned the creation of standards and curriculums to groups of experts who were ready before the beginning of the school year.

Training teachers to teach IT from 1st to 4th grades has been done since 1998, when a standard curriculum for "Working with computers and IT for 1st – 4th grades" [1] was created. The training was conducted at schools, which were able to provide a computer lab with compatible software and a well-prepared primary teacher.

During the 2005/2006 school year 34,307 children were able to study IT by means of this educational form. In contrast, during the 2003/2004 school year, their number was 12 670. In order to ensure the IT education in the primary school, a specialized software environment ToolKid, based on the Comenius Logo [2], has been developed and used. Over the last couple of years more than 1500 teachers have been trained in different after-graduate forms and specializations and gained the needed qualification to teach IT in primary schools.

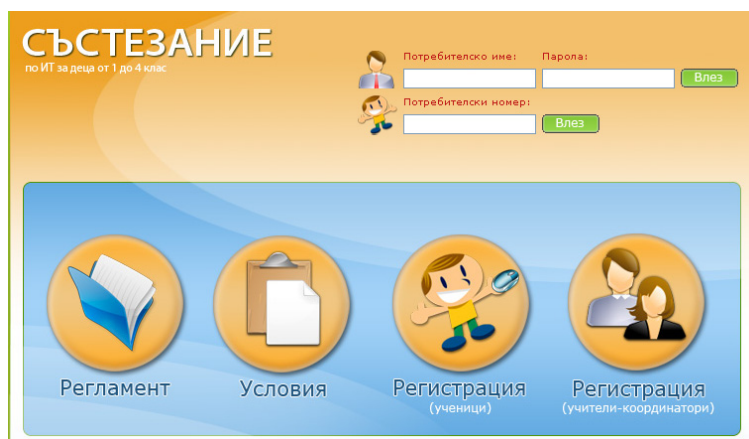
To conclude, all this made it possible to organize the first National IT competition for children from 1st to 4th grade during the 2006/2007 school year.

The organization

The participants were divided into 3 groups according to their age:

- group A: students from 1st grade (7 years old);
- group B: students from 2nd and 3rd grades (8 – 9 years old);
- group B: students from 4th grade (10 years old);

The competition was held on-line at a specialized site: <http://it-kids.eu>.



The different software programs used by the students on the day of the contest were published on this site. In this way were ensured:

- usage of the same software by every participant;
- possibility for evaluation of the students' ability to learn and use unknown software;
- possibility for participation in the competition from home, which provided a solution of the problem with the shortage of computers in schools.

The day before the competition the students had to register. The registration provided the options of choosing a city, school, grade and entering a participant's name. After the registration, the participants automatically received a number that was used in the competition day.

On the day of the competition (12 may 2007) each group was allotted 90 minutes within a different time zone to complete their task. The reason for doing this was because at the schools the number of the participants was bigger than the number of the computers and a computer access for all groups was ensured.

The task

The task was coloring a picture. In a network formed by geometrical figures of the same kind the kids had to create a shape or a complete image. The variants in the different age groups came from the different types and complexity of the network, given to the students. By using this task we wanted to assess not only the students' abilities to work with computers but also their abilities for creative and combinatorial thinking, their keenness of observation, and quickness of mind as well as their abilities for aesthetical combining and harmony of color and forms. We chose a task connected to the fulfillment of a graphical project, having in mind that this kind of activity is a favorite one for the children from this age group and at the same time – highly motivating. We created several kinds of networks in advance which were then given to 1st – 4th grades students from our school in order to check their reaction to such tasks and what the most suitable kind of graphical network for each competing group is (Fig. 1).

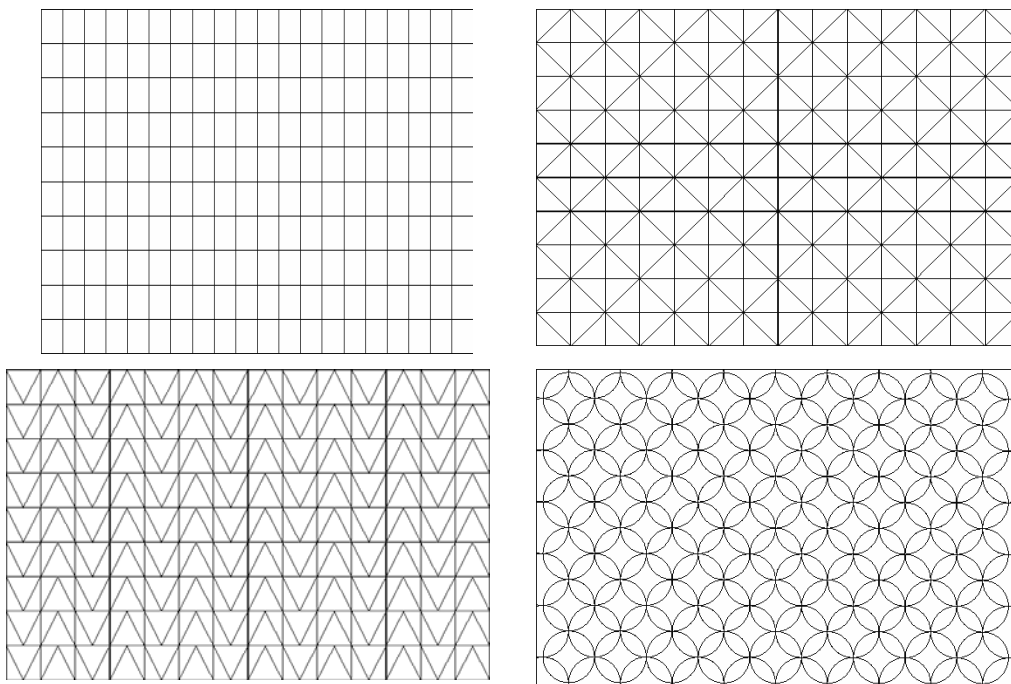


Figure 1. Kinds of networks

This research was important in order a better balance between the complexity of the contest problem and the children's skills from the different age groups to be effected. As a result we chose 3 variants of graphical networks for the different groups. Each one of them allows variants for end results varying from traditional and plain to unexpected and quite complex combinations. Practically it allows every single child to solve a problem and to have a personal achievement and at the same time the level to vary in a quite a wide range, showing the individual abilities of every single child (Fig. 2).

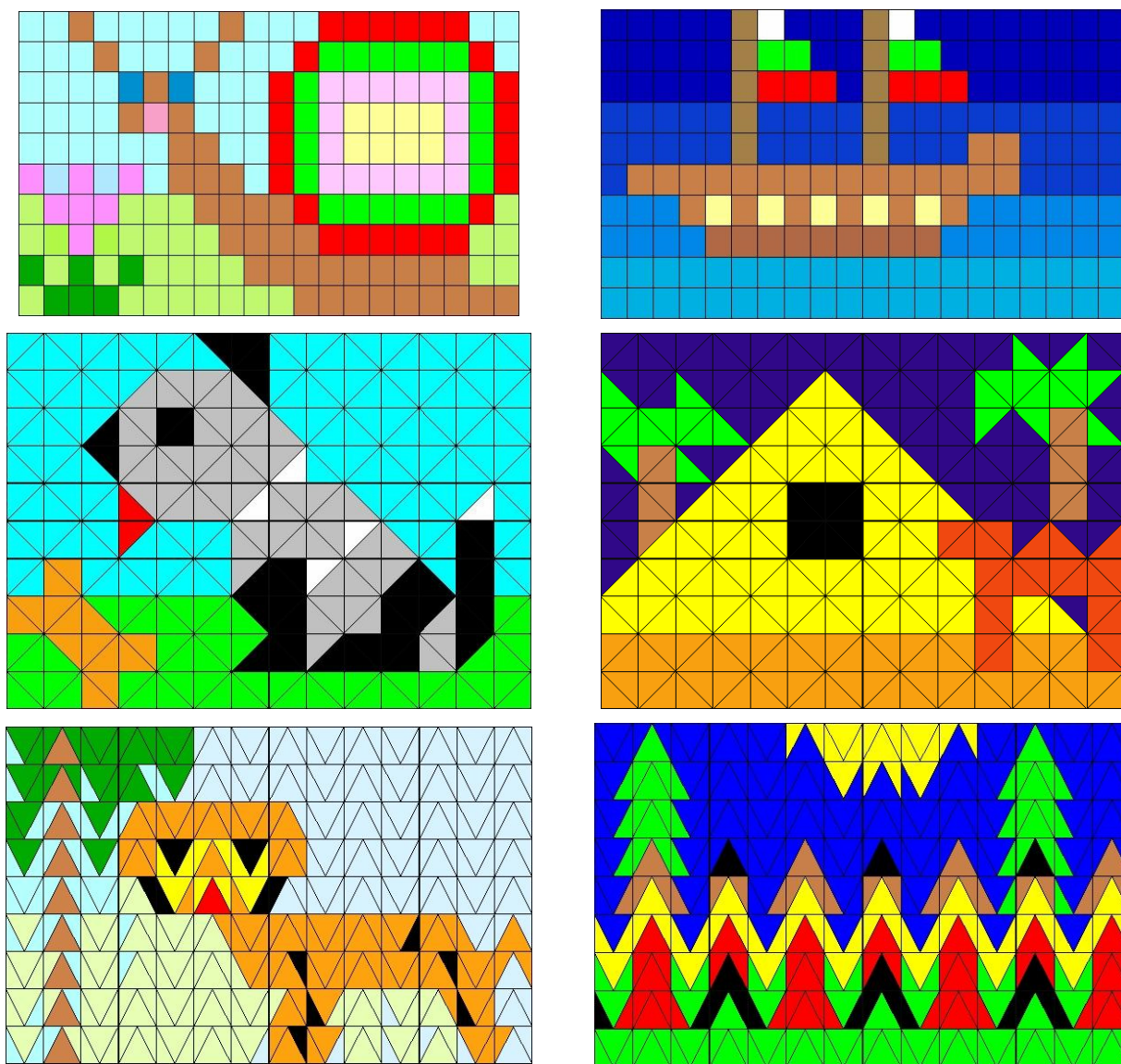


Figure 2. Children's works

Purposefully we refuse our idea to give a specific theme the children to work on. The most important for us was to follow out how children at the same age, having similar experience and abilities, placed at the same situation, and using the same resources, will achieve similar or entirely different results. At the same time the results do not depend directly from the students' artistic talent but demonstrate more universal abilities as keenness of observation, quickness of mind, creativity, nonstandard in using the available instruments. In another words – we wanted with solving the problem and the realization it i to come from the children, without the involvement of the teachers.

The Evaluation

After the end of the contest day all problem solutions were published on the web site for voting. Only students form the corresponding contest groups, who were registered for the contest, were allowed to vote and every one of them was allowed to choose the 5 best solutions. In this way we wanted to ensure the children – participant's active position not only at the time of the contest but in the process of evaluation and choosing the winners.

By means of including the children in the process of evaluation we sought to achieve:

- Popularizing of the all participants projects among the students from their age. It creates conditions for active exchange of ideas between the kids; we thought, that looking at the published on the web site projects will sharpen the kids' keenness of observation not just informative but purposefully; it will provoke a lot of comparisons between the published solutions and the own one; additionally, it will provoke much more emotional engagement to the activity as whole and to the personal involvement into it.
- Forming of criteria for giving an opinion, displaying of critical attitude and self – critical attitude to the end result from different activities; at this age the egocentric position of the kid is still very strong. Giving to the kid the possibility to point not only one but five from the best solutions we ensure the possibility to feel well, voting for himself and at the same time to come out of this position, to evaluate the other solutions and publicly to give them the well-deserved.
- Maximum objectivity of the results by means of a big number of voting kids; this was the way to escape the subjectivity of a jury, composed of exact number of people, even carefully chosen.
- Authenticity of the evaluation criteria, due to the fact that not adult but kids evaluate kids. We wanted to escape the very often met talking at cross-purposes between adults and kids. We wanted to understand the kid's point of view and not to impose our one. We think that an evaluation, given by students at the same age has a much bigger value for the kid in such a situation.

The voting lasted for one week. After that the winners were announced and awarded in the schools.

Meanwhile, to see how the land lies, we gave an opportunity to the primary school teachers who had conducted the competition in their schools to prepare a poll and an unofficial top list of their own. The teachers were warned that their votes are not important for the participants' final positions, hoping that they would give their best objective score without being partial to anyone. Our desire was to compare the results of the teachers' votes with the children's votes in order to assess to what extent the teachers notice and value that, which is important for their students, and if there were any similarities and differences in the evaluation.

The Software

From the beginning of 2007 an adaptation for Bulgarian language was created on Imagine [3]. That with the presence of Plugin for web pages allowed its usage as a platform for development of software solutions for the competition. Other significant advantages, which the program environment lends to its users, is the opportunity to select certain options in dialog boxes to program the behavior of the turtles; to create images by defined instructions; to work with pages, which allowed the use of internal graphic files only.

These advantages allowed the creation of software solutions in the last days of the competition by a team of primary teachers using IT (fig. 3).

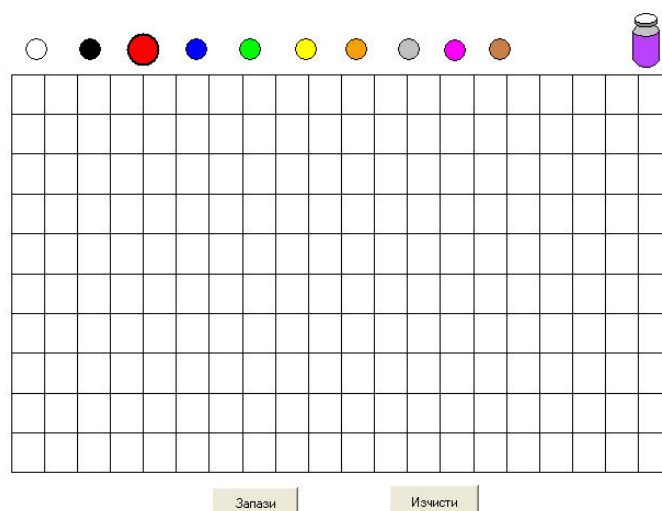


Figure 3. Software

Imagine's main disadvantage is the impossibility to save and print the typed text when it is saved in a web page format. This made necessary the creation of a special ActiveX control for Internet Explorer with the main purpose of saving the content of the clipboard as a graphic file and then saving it on the server.

At the end of the task, the children had to perform a series of actions using buttons, such as ,copying the created information on the clipboard and sending it to the site of the competition (saving was kept secret, and each name was generated using each competitor's personal number).

Outcomes

As of today the competition has not yet been carried out. Therefore, the results (in respect to the expected educational effect on children as well as the decisions taken by them) will be presented at the conference.

Results

Participants

1546 have registered to take part in the competition. The number of participants in the day of the competition was 1265. We think that the less number is because some of the children did not get access to the PCs with internet connection. It's also possible that some of the children did not manage to send the completed tasks.

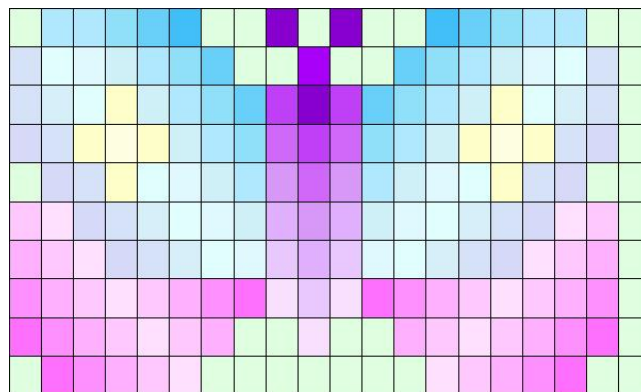
Analyzing the information of the participating children showed that they are coming from different regions in the country: not only from the big towns, but also from the smaller ones. The number of boys and girls is almost equal. Taking into account that in the last 10 years in Bulgaria the birth rate of boys is larger than girls' this is quite good activity from the girls.



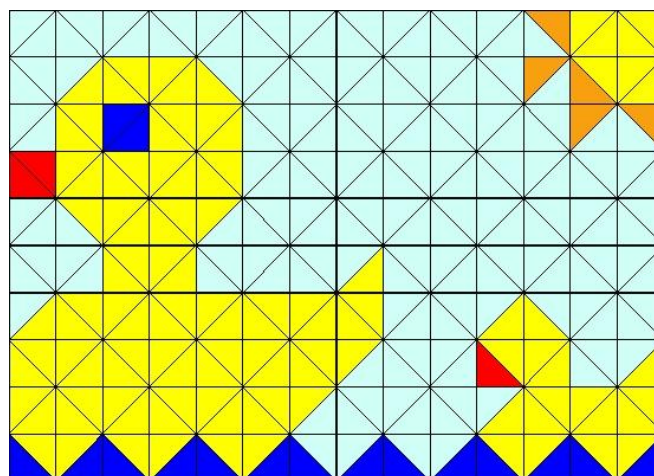
Graphic projects

The analysis of the children's project shows several tendencies:

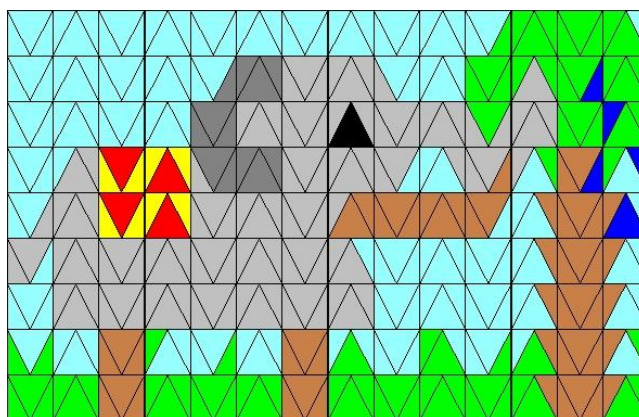
- Big amount of variety of graphic projects. This reveals that the competition task has provoked children's imagination and has aroused the artistic potential of every child. We consider this result as very important, because we had as our main purpose to see not only if children can work with computers, but also if they can use the new technologies to present their new ideas.
- A bigger part of the children did not created only a separate object, but they have created a whole graphic projects with one or several main objects. They were placed on a background and enhance the impression and puts forward the main idea of the project. This is witnessed even in the projects of the youngest children. This shows children's commitment to the projects, wholeness of the idea, and strong motivation for realization of the proper development of the idea.
- The analysis of the tasks shows that for children it was not enough to use the ready colours, but they have used the tools of the software to create new colours and shades to use in their projects. This shows ability for quick orientation in the instrumentarium and proper usage of it. This fact proves their desire for work, finding optimal options for graphic development of the project, precise work, showing thought to the project, searching for different means of impression.



1st place in group A, Joana, girl, 7 y.o., Sofia



1st place in group B, Svetlina, girl, 8 y.o., Varna



1st place in group C, Bojidar, boy, 10 y.o., Pleven

Summary

If we take into consideration that this is the first initiative of this kind for the country, it was carried out in a short time, without support from the media, but only informing teachers in different schools, we evaluate the project as very successful. There was big amount of interest and activity not only from teachers, but also from children in their early stage of using IT and technologies of that kind. In our opinion the competiution should be carried out in future and the results should be populized more widely among schools and the society as awhile. We also think that the task should be creative, because it will provoke children's interest and motivation for performing the task and showing their potential.

References

Ilieva, V., Ivanov, I. (1999) *Informatics and Information Technologies in a Primary School – based Logo Environment*, In Proceedings of EuroLogo 1999. Edited by J. Sendova. Sofia, August. pp. 227-234.

Ilieva, V., Ivanov, I. (2005) *ToolKID – Logo based software package for children*. In Proceedings of EuroLogo 2005. Edited by Gregorczyk, G et al. Varsaw, August. pp. 390 – 397.

Blaho, A., Kalash, I. (2001) *Object Metaphore Helps Create Simple Logo Projects*. In Proceedings of EuroLogo 2001. Edited by G. Futschek. Linz, August. pp. 39 – 43.

Logo in Polish Schools - cases

Wanda Jochemczyk, wanda@oeiizk.waw.pl

Witold Kranas, witek@oeiizk.waw.pl

Katarzyna Olędzka, katarzyna@oeiizk.waw.pl

All from Computer Assisted Education and Information Technology Centre (OEIIZK), Warsaw, Raszyńska 8/10

Abstract

Logo is widely used in Polish schools on first two levels – primary school and lower secondary school. We will present different examples of Logo use.

In early years we use microworlds (made in Imagine) helping young pupils to learn elementary skills: drawing, reading, writing, calculation, understanding environment. In primary school there is an introduction to computer science and ICT use. In lower secondary – some elements of algorithmic – an introduction to programming. Finally in upper secondary – microworlds – animated models helping to understand fundamental topics in different subjects of study. There are some questions arising concerning Logo use.

Seymour Papert (1999) wrote: *“The rapid and accelerating change that marks our times means that every individual will see bigger changes every few years than previous generations saw in a lifetime. So this is a choice we must make for ourselves, for our children, for our countries and for our planet: acquire the skills needed to participate with understanding in the construction of what is new OR be resigned to a life of dependency.”*

So the main question is: What is changing in our school Logo use? Educational system is changing slowly. We try to work simultaneously with teachers and pupils to accelerate the changes. Logo competitions and common e-learning platform described here are the examples. We are introducing Logo in modern object-oriented environment Logomocja – Imagine. We use animations and models also prepared in Imagine. But there are still some classic problems – mainly programming techniques – a very demanding problems for pupils and teachers. This problems need a good didactic methods helping our pupils to achieve understanding. It's important task for Logo community.



Figure 1. Logo in Polish schools - examples

Keywords

Logo; Computer Science; education in Poland; informatics; Logo competition; Imagine; Logomocja

Computer Science in Polish schools

Computer Science is a school subject in Poland since 1987 (it was not obligatory in first years). In a present school system it's represented on three levels:

- Primary school – 80 hours course (2 hours in timetable) called Informatics (pupils age 10 – 12). It's typical ICT course preparing to use computer.
- Lower secondary school (gimnazjum) – 80 hours course (2 hours in timetable) called Informatics (pupils age 13 – 15) This course has some elements of programming (algorithmics).
- Upper secondary school (liceum) – 80 hours course (2 hours in timetable) called Information Technology (pupils age 16 – 18). On this level students may also choose Informatics as a subject (with 200 hours course – 5 hours in timetable).

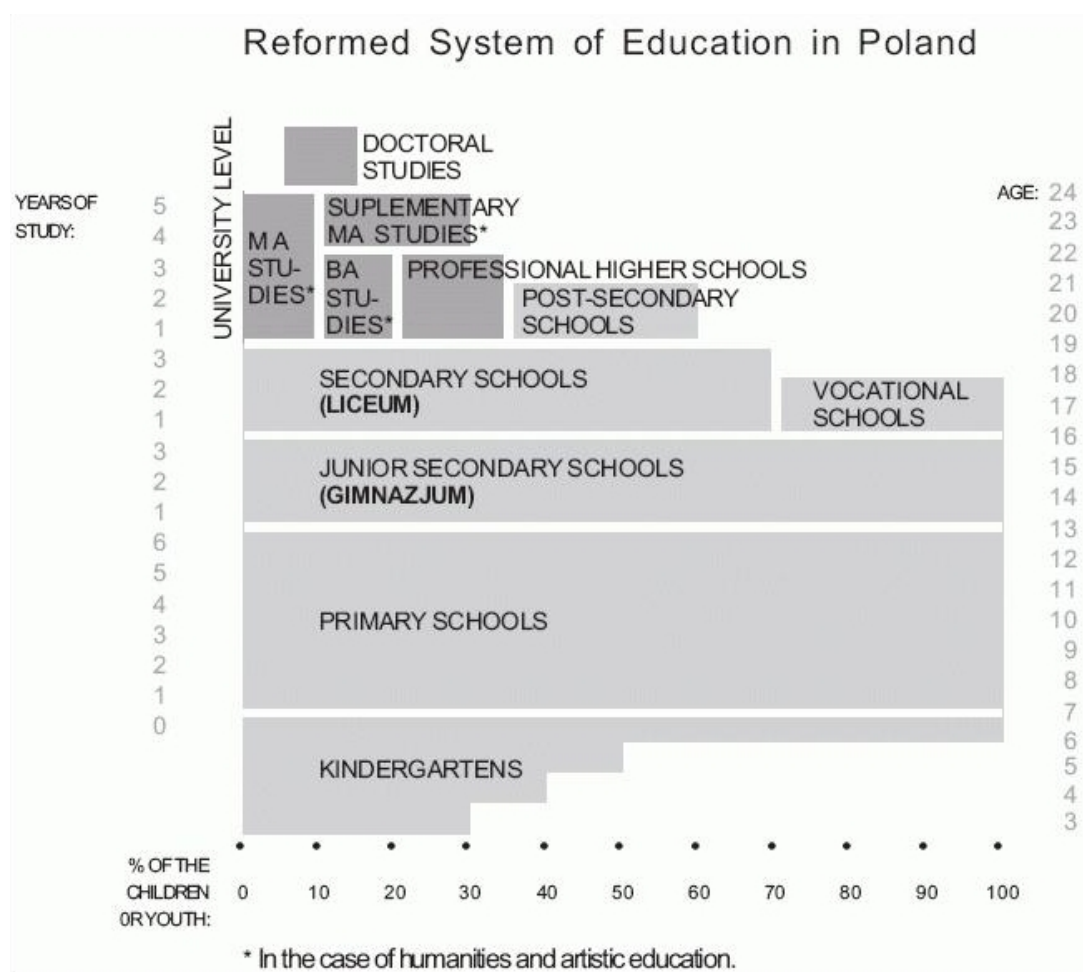


Figure 2. Polish system of education (source Art Academy Kraków)

In early years

First three years of primary school are called integrated teaching. One teacher is teaching the class. Only few additional subjects are thought by others. We notice a growing interest in using ICT among teachers in first three grades. We prepared a workbook for young pupils titled *Lessons with computer for integrated teaching* [Jochemczyk et al. 2006] as an introduction to ICT. The idea of this workbook was presented on Eurologo2005 Conference in Warsaw [Jochemczyk & Olędzka 2005 a].

Building elementary ICT skills

There is more than a hundred different microworlds on a disk enclosed in a workbook. Here is one example: building a vicinity plan. Drawing a vicinity plan is a topic in second grade curriculum. A microworld allows to construct this plan out of a symbols placed in a legend.

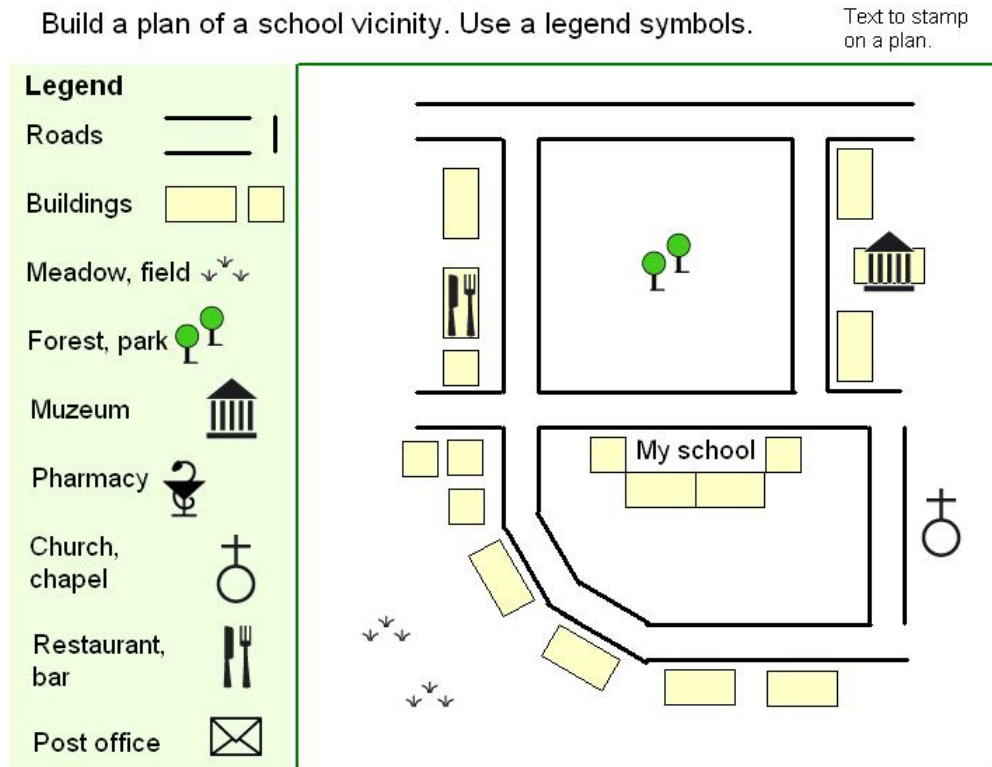


Figure 3. Building a school vicinity plan

It is possible to insert roads and buildings by dragging a piece of road or a building on a plan. It's also possible to rotate it by clicking a right mouse button and drag it to another place. In a legend there are some typical symbols. Those symbols may be dragged on a plan. It is possible to insert a text by clicking a right mouse button anywhere on a plan. The text should be written in a blue window. If an element is dragged outside a plan it's deleted.

Introducing simple turtle drawing

In our workbook two units are devoted to programming in the Logo. It is the first step in programming for children. They can easily learn how to solve some turtle graphic problems.

Pupils can control a way the turtle moves. They start with simple drawings, the turtle can go forward, backward, turn right, left and change a pen colour and pen width. They steer a turtle using buttons. The next step is to drag and drop Logo instructions to build a list of commands. Pupils learn how to create their own procedures.

Teacher can carry next lessons in computer lab in Imagine environment. During a lesson children write commands in a command line. Than they start to write simple procedures without parameters.

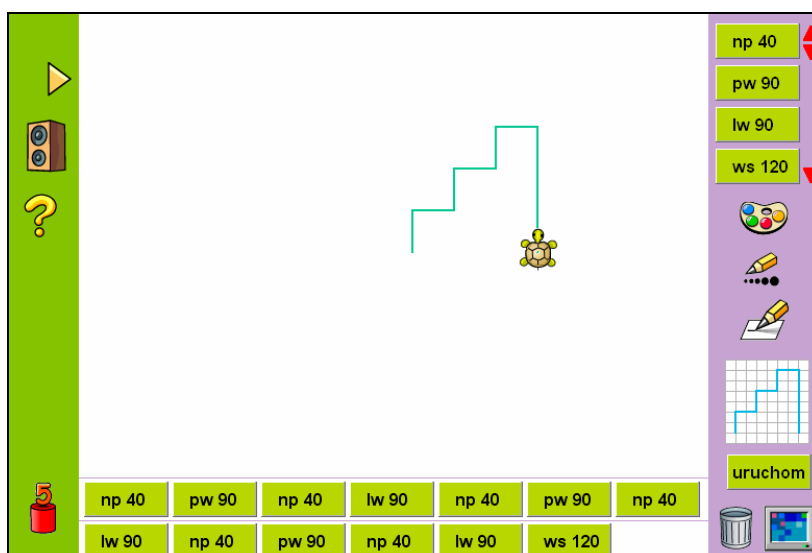


Figure 4. Build a list of commands.

Primary school

Teaching Logo

According to a basic (state, obligatory) curriculum Computer Science course in primary school is rather ICT course. A main aim is to learn how to use ICT with a stress on pictures, texts and communication. The course is usually done in 4th to 6th grade (pupils 10-12 years old). From our point of view it's occasion to introduce also some elements of programming using Logo. As environment we use Logomocja – Polish version of Imagine. Logo lessons are a part of schoolbook *Lessons with computer for primary school* [Jochemczyk et al. 2004].

On this basis we prepared a cycle of lessons on the Moodle e-learning platform. It was the innovative idea. There were different topics for each month: *First lesson with computer* in September, *Adventures of Mr Spelling* in October, next was *Round the Christmas Tree*, *Together with the Turtle* etc. Each lesson included different activities like quizzes, assignments, forums, creating dictionary or database. Many of the lessons were based on Imagine environment. Sometimes a task was to write a procedure, otherwise to create a multimedia project using events and processes or just play an educational game.

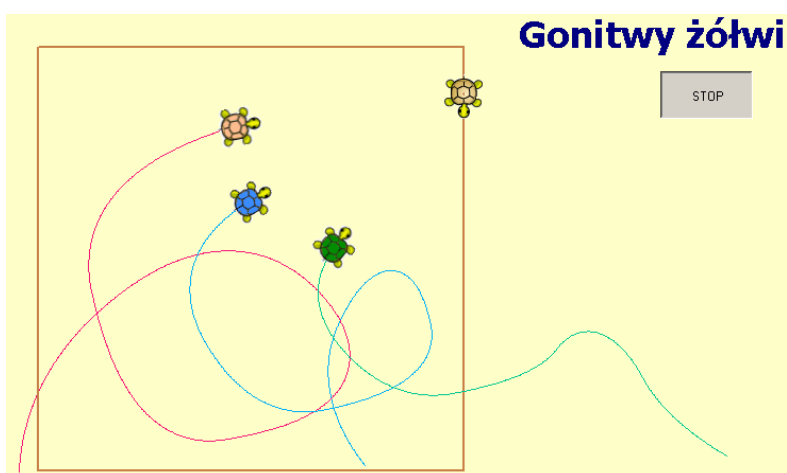


Figure 5. Turtle chasing – project.

In addition, we systematically met with teachers. At these face-to-face meetings we worked with teachers who played a role of pupils. They carry out all tasks included in a lesson to learn how to work on the Moodle platform and to give us a feedback. After each meeting there were prepared methodical materials. In the next step, teachers realized these lessons with pupils.

Lessons met a great teachers and pupils interest. Elements of distance learning in combination with introduction to programming skills and advanced use of Imagine environment were inspiring and professionally developing challenge. From teachers' reports we know that pupils look forward to participate in those lessons.

Working with gifted pupils - Logo competition

Polish Logo competition for primary school children is described in [Jochemczyk & Olędzka 2005b]. This year there was a fifth edition of the competition. Here we will introduce one task from a final stage of the first competition. Similar competition in Slovakia is described by Tomcsanyiova & Tomcsanyi (2005).

Task EKI

Write a procedure **EKI :n**, which will draw in the middle of a screen, as large as possible drawing alike on the pictures. Parameter :n, means the complexity of a drawing. For :n=1 it is a big square divided to four similar squares Upper left square is empty, and three remaining include a letter E, like shown on the first picture. Parameter :n can have values from 1 to 5.

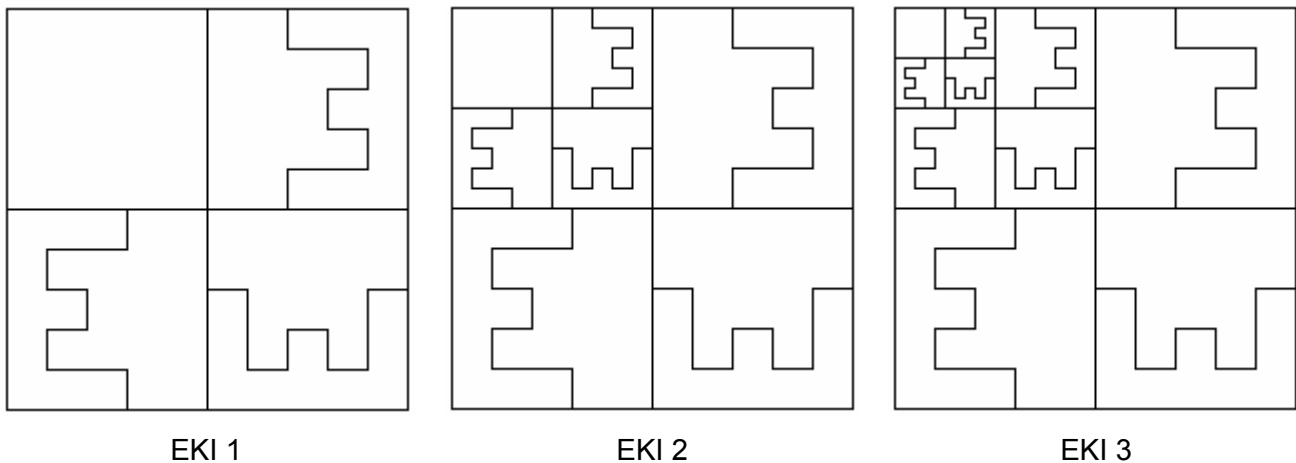


Figure 6. Results of EKI procedure

The task is “algorithmic” in a sense that most important is a drawing algorithm containing loop. It was a demanding task for 12 years pupils.

We were dreaming about solutions like:

```
to EkiSquareR :n :size
  if :n = 0 [stop]
  repeat 3 [SquareWithE :size / 2 right 90]
  square :size / 2
  pu forward :size / 4 right 90 forward :size / 4 pd
  EkiSquareR :n - 1 :size / 2
end
```

where **SquareWithE** draws a square with E shape (upper right corner of EKI 1 picture).

Only 25% of pupils managed to write a procedure which has a satisfactory effect. Among them there were some step by step solutions. Only about 15% of pupils were able to organize a loop

using parameter. All this solutions used iteration loop REPEAT with variable changes inside a loop. Typical solution (main part) was similar to:

```
to EkiSquare :n :size
  repeat :n
    [3SquaresWithE :size
     forward :size / 4 left 90 pu forward :size / 4 right 90 pd
     let "size :size / 2]
end
```

where 3SquaresWithE draws a picture EKI 1.

This shows that probably best pupils are studying other languages like Pascal or C – mainly outside of a school. Talks with them are supporting this guess – for some of them Logo was not the first programming language.

Lower secondary school

Teaching recurrence

Trying to teach fundamental algorithms on middle level of education we introduce operations on words and lists in Logo. Then we try to use recursion and to prepare pupils for understanding of fractal constructions. The idea how to do that was described by Brian Harvey (1985). First procedure is “eating” words. It’s an example of tail recursion.

Procedure definition	? EAT "turtle
	turtle
to EAT :word	turtl
print :word	turt
if empty? butLast :word [stop]	tur
EAT butLast :word	tu
End	t

The second procedure is “eating” and then turning back words. Here we have non tail recursion. We ask pupils: “What will happen if we add anything after recursive call” and encourage them to experiment. Then we show the result of NOTCH procedure (shown below) and ask them to obtain the same result by changing procedure EAT.

Procedure definition	? NOTCH "turtle
	turtle
	turtl
	turt
to NOTCH :word	tur
print :word	tu
if empty? butLast :word [stop]	t
NOTCH butLast :word	tu
print :word	tur
End	turt
	turtl
	turtle

Often pupils are astonished with the solution and some of them ask “How it is possible?”. This is much harder question. To answer we have to trace deeply NOTCH procedure. The best “tracing” is to let pupils to act as Logo procedures. So we are preparing a show. We have lines of NOTCH procedure. Needed actors should play the roles of: NOTCH, print, if, empty?, butLast, stop, NOTCH (we should be prepared for this next NOTCH), end.

Instead of costumes there are sheets of paper with names of procedures. Than we have a “dress rehearsal” – pupils have to say what their procedure is doing. A final performance has

some dramatic moments. The first - when we come to NOTCH call inside NOTCH – we resolve it by engaging another actor with NOTCH role – this actor is now taking a lead. The next moment is when we come to the first STOP and his role is to fire out the latest NOTCH.

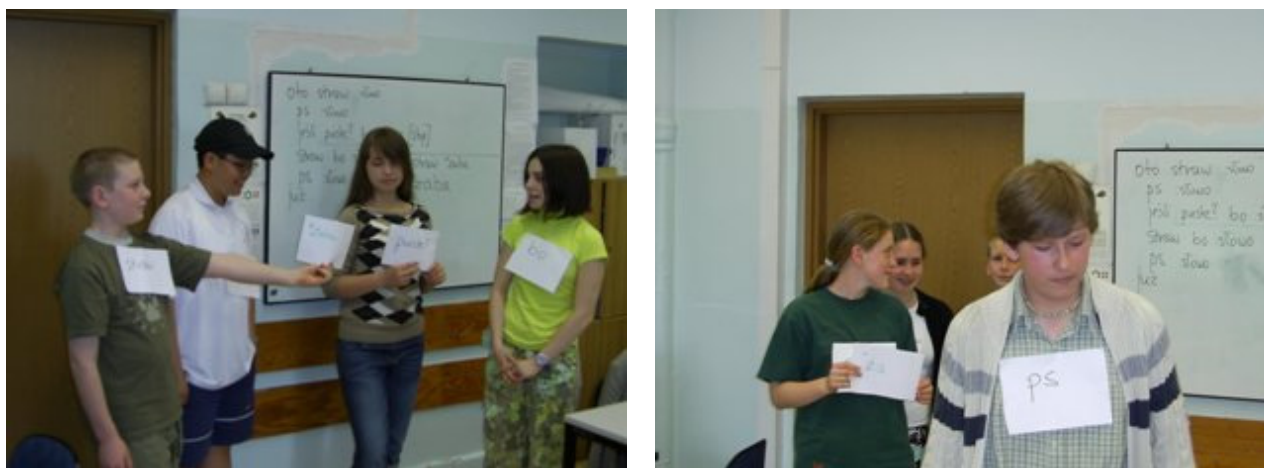


Figure 7. Pupils performing NOTCH procedure

There is some fun and a play message is when we hear “Now I understand”.

Upper secondary

There is only one schoolbook proposing to teach Logo during ICT course on upper secondary level, prepared by team from our Centre guided by Andrzej Walat (2002). Logo part is mainly devoted to artistic turtle graphics – production of series of pictures, mainly square variations.

We rather try to use models – animations prepared in Imagine. Here is one example – a microworld *Velocity and Force*. The microworld is a product of CoLabs project guided by Marta Turcsanyi-Szabo. Some interesting CoLabs project results were presented by Marta Turcsanyi-Szabo and Ivan Kalas at Eurologo2005 [Kalas & Turcsanyi-Szabo 2005]. The idea was developed by Andrzej Walat, programming is done mainly by Witold Kranas, many improvements are due to Katarzyna Olędzka.

The microworld is developed to support applied mathematics and physics teaching / learning on secondary education level (age 12 - 18). The aim of this microworld is to introduce vectors in physics and to analyze the role of velocity and force (acceleration) vectors. It helps to understand how force is changing velocity vector and thus how force rules the body movement (second Newton's Law). It contains 2 important examples:

- body movement in uniform gravitational field – near the Earth surface – to analyze different kinds of “shots”,
- body movement in central gravitational field, for example satellite or planetoid movement.

The example of steering the body movement on a round track gives the possibility to understand the role of centripetal force.

Learning with models

1. Free control (steering a rocket in space). A task is to steer the movement of a rocket. A rocket is somewhere deep in space, far from other bodies. The only force changing its movement is due to rocket engine. Steering centre is a black arrow of a force vector. Force value and direction may be changed by dragging black arrow. It's also possible to change initial velocity vector.

2. Constant gravity (throwing a stone). A task is to throw a stone by setting initial velocity. Then the motion of a stone is under gravity force directed top-down. You can set initial velocity vector. During the motion it's possible to watch how velocity is changing due to gravitational force.

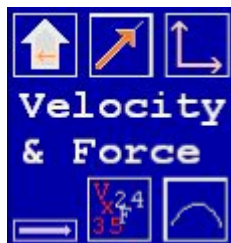


Figure 8. Pictures from Velocity & Force microworld

3. Central gravity (launching a planetoid). A task is to let a little planetoid move around the Sun. The only force acting on a planetoid is gravitational pull from the Sun. It's possible to change initial velocity vector. The force vector is dependent only on the distance from the Sun. It's worth trying how the planetoid is moving when it starts with different velocity values.

4. Round track (controlling circular movement). How can be circular motion achieved? A task is to steer the rocket in such a way, that it could move inside round track. It's necessary to control movement all the time by changing force vector.

5. Painted track (game for two). A track may be painted with a pencil. A task is to control movement on a track by changing force vector.

The exercises are "open". There are possibilities of different kinds of activity, gathering the experience, answering questions: what will happen when we change...?

The microworld is used to help Physics teacher. It is useful to improve understanding of the Newtonian laws of motion and also for summing up classic dynamics. Students like a "game way" of presenting motion problems.

Instead of conclusion – a bunch of questions

We (at least two of us) started our educational Logo activity about 20 years ago. In those happy days we were giving Logo programming courses focused mainly on programming techniques, developing Polish versions of Logo language, preparing some code examples and educational materials. Then blowing begun and IT started with Windows, games and Internet. Now there is a diminishing interest in programming courses but what's instead? Computer driving licence training? Maybe for teachers, pupils rather don't need it.

Is Logo still useful in education? We like the examples we presented here. But we have a feeling that nowadays many of them are either too hard for teachers and students or out of their interests. We try to shift a bit to educational microworlds prepared in Imagine helping to learn different subjects. But this direction means focusing rather on subject matter not on programming language. So what's the main reason to use Logo in education? Not least is – to have some fun in teaching/learning.

Sorry for this very personal summary of our work. We are looking toward the discussion during a conference.

References

Harvey, B. (1985) *Computer Science Logo Style. Vol. 1: Intermediate Programming*. The MIT Press, Cambridge, Massachusetts

Jochemczyk, W. and Olędzka, K. (2005 a) *Imagine in educational projects for young children*. In Proceedings of EuroLogo 2005. Edited by G. Gregorczyk et al. Warsaw. pp. 386 – 389.

Jochemczyk, W. and Olędzka, K. (2005 b) *Logo competition for primary school children*. In Proceedings of EuroLogo 2005. Edited by G. Gregorczyk et al. Warsaw. pp. 383 – 385.

Jochemczyk, W.; Krajewska-Kranas, I.; Kranas, W. and Wyczółkowski, M. (2003) *Lekcje z komputerem Podręcznik dla ucznia gimnazjum*. WSiP, Warszawa.

Jochemczyk, W.; Krajewska-Kranas, I.; Kranas, W.; Samulska, A. and Wyczółkowski, M. (2004) *Lekcje z komputerem Podręcznik dla szkoły podstawowej Klasy 4-6*. WSiP, Warszawa.

Jochemczyk, W.; Krajewska-Kranas, I.; Olędzka, K.; Opęchowski, W.; Samulska, A.; Wilk, E. and Wyczółkowski, M. (2006) *Lekcje z komputerem w nauczaniu zintegrowanym Zeszyt ćwiczeń*. WSiP, Warszawa.

Kalas, I. and Turcsanyi-Szabo, M. (2005) *Collaboration – a tool for learning*. Proceedings of EuroLogo 2005. Edited by G. Gregorczyk et al. Warsaw. pp. 54 – 65.

Papert, S. (1999) *Introduction: What is Logo? And Who Needs It?* In: *Logo Philosophy and Implementation*. Logo Computer Systems Inc., USA.

Tomcsanyiova, M. and Tomcsanyi, P. (2005) *Logo programming competition in Slovakia*. In Proceedings of EuroLogo 2005. Edited by G. Gregorczyk et al. Warsaw. pp. 377 – 382.

Walat, A. ed. (2002) *Technologia Informacyjna Podręcznik do kształcenia podstawowego w liceach i technikach*. Oficyna Edukacyjna Krzysztof Pazdro, Warszawa.

Moodle e-learning platform: <http://www.e-nauczanie.wsip.com.pl>

Turtle and children on Moodle e-learning platform

Wanda Jochemczyk, wanda@oeiizk.waw.pl

Computer Assisted Education and Information Technology Centre, Warsaw

Katarzyna Olędzka, katarzyna@oeiizk.waw.pl

Computer Assisted Education and Information Technology Centre, Warsaw

Abstract

Computer Assisted Education and Information Technology Centre organized workshops for primary school children on Moodle e-learning platform. The objectives of the project were to study programming in Logo, getting familiar with the Imagine environment, improving electronic communication and learning to gain new knowledge in Internet-based courses. There were 2 face-to-face meetings and 56 working hours on e-learning platform. Children solved different programming tasks, quizzes, actively participated in the forum and performed other activities. During courses the children created beautiful multimedia projects. For students it was a good opportunity to create a collaborative community and to discuss problems with fellow classmates from others schools.

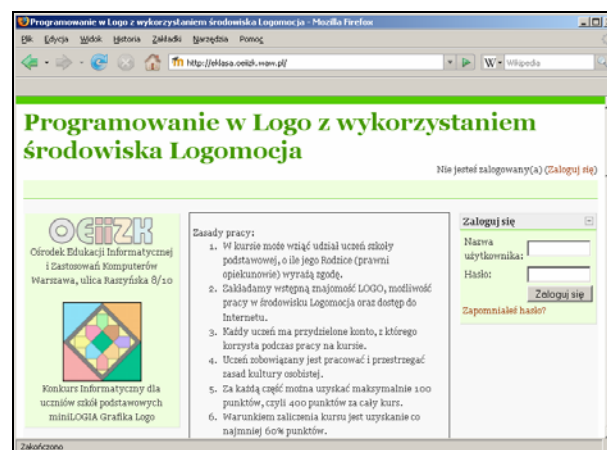


Figure 1. The main webpage for the project (<http://eklasa.oeiizk.waw.pl>)

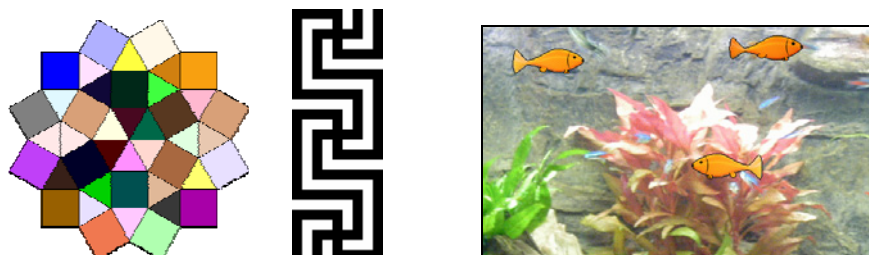


Figure 2. Examples of tasks

Keywords

Logo, Imagine, primary school children, e-learning platform, Moodle

Introduction

Computer Assisted Education and Information Technology Centre is an in-service teachers training institution, which specialises in computer science and ICT. It provides support for teachers in their professional development as far as computer skills and didactic applications of information technology are concerned. Since 1994, programming contests for students have been organized. Currently, the competitions are held at two different levels - miniLOGIA - for primary schools children (up to 13-year-old) and LOGIA for lower secondary (up to 16-year-old). In response to demand we have started organizing an online logo programming courses for children.

The workshops for primary school children entitled "Programming in Logo environment" have been already organized twice. For the first time from December 2005 to February 2006, the second one was a year later – from December 2006 till January 2007. About 60 children from different schools in Mazowia district took part in both courses. Most of them were participants of miniLOGIA contest. Each course consisted of 4 hours face-to-face activities and 56 hours of distance learning on Moodle platform.

The objectives of our project were to study programming in Logo and getting familiar with the Imagine environment. Moreover, while taking part, students improved their computer communication skills and they learned how to gain new knowledge through the intermediary of the Internet-based courses.

Learning programming is difficult, so we were a little bit concerned about it. But the students proved that they did not have problems connected with virtual communicating and they were also ready to make a great effort to acquire new abilities.

Work principles

Every student of primary school, who had an Internet access and possibility to work in Imagine environment, could participate in our courses. Although most pupils had Internet connection in their homes, some of the participants could only work at school under their teacher's guidance. To sign up for the course you had to fill in the form on the proper webpage. Registration was done by teachers of computer science, parents or children themselves.

Each student had an account on the Moodle platform to be used in the project. It allowed them to gain access to the system from any place with an Internet connection. On the other hand, it provided controlled access, so only the enrolled participants could use the system, so no user was anonymous.

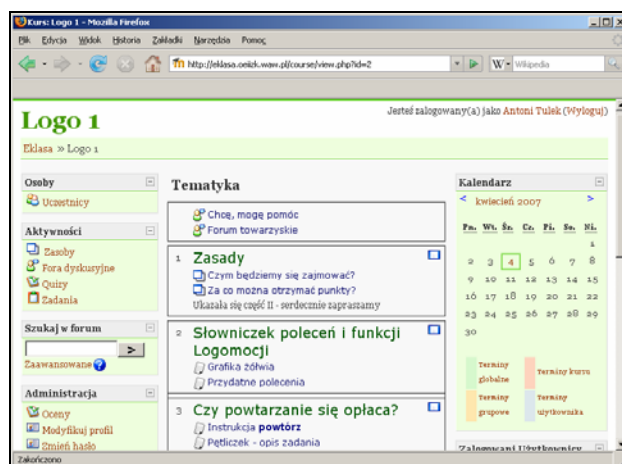


Figure 3. A webpage with resources and tasks for participants

To complete the course a minimum of 60% of all possible points were required. Every student who successfully completed the course received a certificate. For many children it was the first time in their lives when they were given such a document.

Curriculum and organisation

We divided a project organization in three different sections:

- first a face-to-face introductory meeting,
- working with the Moodle e-learning platform,
- final a summing up meeting.

The first meeting was organized in order to know each other and to integrate. This allowed us to take up the challenge to learn programming. During classes students got familiar with Moodle platform – how to navigate, where to find useful resources and how to do different assignments. These skills were very useful, because in next few weeks the students independently surfed. Moreover, in the first meeting they solved some algorithmic problems. The following weeks everyone worked on the e-learning platform which is described below.

In our last meeting we solved some more difficult algorithmic tasks in Logo graphics. Especially we were concerned for difficulties which the participants had during the course. But the most exciting was chatting with people known from working via the Internet and to watch other participants' work.

The course material was divided into four parts according to logical rules and the level of difficulty. The first part was dedicated to remind primitive logo functions and simple examples of using iteration were presented. In the second and the third part we improved ability to apply "repeat" instruction introducing more difficult problems. We also dealt with different tasks including fulfilling, using random functions, measurement and drawing pictures in various sizes. Fourth part was concentrated on recursion and testing. Moreover, in every part students had to create some multimedia project.

During the course students solved variety problems to develop programming skills - perceiving and formulating algorithm problems, constructing and analyzing solutions in the form of algorithms, expressing algorithms as procedures and testing them. We encourage students to read instructions carefully and to consider variety of ways to solve a problem to choose the best solution. We also practised coding procedures with frequent testing after each step. Moreover, students were encouraged to test completed tasks for different parameters, in random procedures checking many times whether the solution was correct and if it fulfilled all conditions required by instruction.







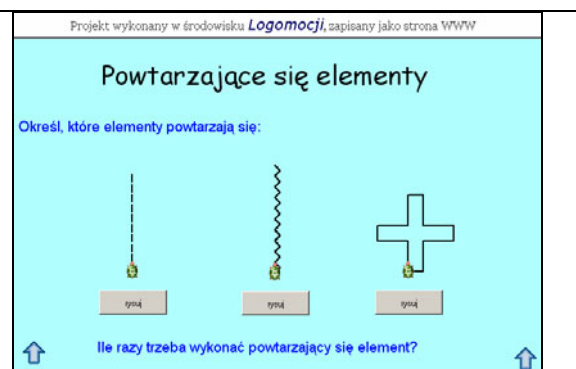
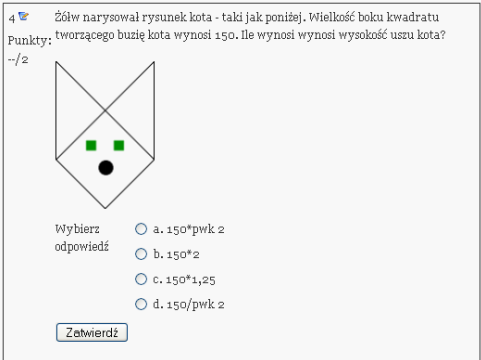
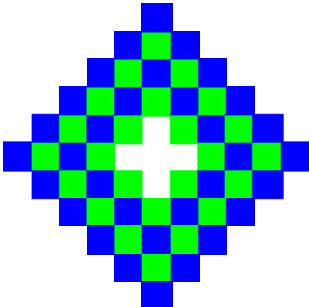

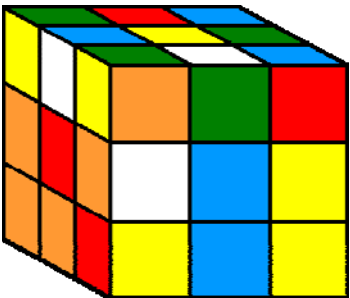
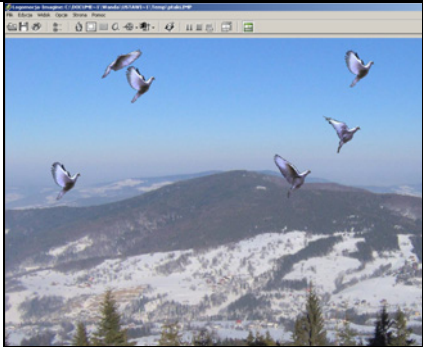
Imię / Nazwisko	Miasto
 Ignacy Stefanik	Milanówek
 Katarzyna Masłowska	Warszawa
 Karolina Grodzka	Trewnin
 Michał Radek	Warszawa
 Krzysztof Kwiatek	Warszawa
 Ewa Tomszys	Warszawa

Figure 4. A part of students list

Resources

Moodle supports a range of different resource types that allow us to include a variety of different materials – we usually used web pages and Imagine projects.



<p>Quiz</p> <p>In our course there were several quizzes. Most often they had 10 multiple choice questions. The major advantage of this module is that it is self checked and students receive a feedback just after solving it.</p>	
<p>Forum</p> <p>This collaborative task allows us to arrange discussion. The topic in this example was to find out different ways of drawing this picture.</p>	
<p>Assignment</p> <p>Assignments were usually similar to those one can find in miniLOGIA contest. The main aim was to write the procedure which would draw a picture shown in the example with conditions described in the task. There were procedures with and without parameters.</p>	
<p>Glossary</p> <p>Another collaborated task – student build their own dictionary containing different pictures drawn by logo procedures.</p>	
<p>Assignment Multimedia project</p> <p>During the course students prepared several multimedia projects in Imagine using object oriented programming.</p>	

Advantages of using e-learning Moodle platform

Using e-learning platform allows us to make our course materials available and organize effective communication in a group of participants. For the sake of the course character we prefer an asynchrony access. Assignments, quizzes and other modules allow us to verify students' progress in a systematic way and to give a quick feedback and appreciate students' achievements. Easy to use platform instruments allow an efficient course management and make it easier to analyse and report progress of each participant.

Putting emphasis on working online resulted in high activity of participants and requirement of methodical learning. Working on e-learning platform allows pupil not to waste their time on travelling. In addition, it was easier to save up some time for learning without strict hours, but of course there were deadlines for each task. Participating in the project was a good opportunity to create a collaborative community and to discuss problems with fellow classmates from other schools. It was a chance to interact with similar interests students and to exchange experience in forum discussions and presenting own work.

Conclusion

Our course was awarded in eLearning Awards 2006 (<http://elearningawards.eun.org>) organized by European Schoolnet. This is an annual event which identifies and rewards excellent practice in using ICT for learning across Europe. Programming in Logo using the Imagine Environment – Project for Primary Students has been classified in 100 top products.

We also got a feedback from the ICT teachers. It occurred that they were “jealous” of children's experience practices and they demanded a similar course for them. To meet their expectations we organized a workshop for them in September and October 2006. The next programming course was for children from villages and we hope it is not the end of logo activities on Moodle platform.

References

- Jochemczyk, W. and Olędzka, K. (2005 b) *Logo competition for primary school children*. In Proceedings of EuroLogo 2005. Edited by G. Gregorczyk et al. Warsaw. pp. 383 – 385.
- Jochemczyk, W.; Krajewska-Kranas, I.; Kranas, W.; Samulska, A. and Wyczółkowski, M. (2004) *Lekcje z komputerem Podręcznik dla szkoły podstawowej Klasy 4-6*. WSiP, Warszawa.
- Borowiecka, A., Borowiecki, M., Chechłacz K., Jochemczyk, W.; Olędzka, K. and Samulska, A.. (2005) *Konkursy Informatyczne LOGIA i miniLOGIA 2002/3 - 2004/05*. OEliZK, Warszawa.
- Papert, S. (1999) *Introduction: What is Logo? And Who Needs It?* In: *Logo Philosophy and Implementation*. Logo Computer Systems Inc., USA.
- Moodle e-learning platform <http://eklasa.oeiizk.waw.pl>
- Logo competition for primary school portal <http://minilogia.oeiizk.waw.pl>

A Logo-based Task for Arithmetical Activity

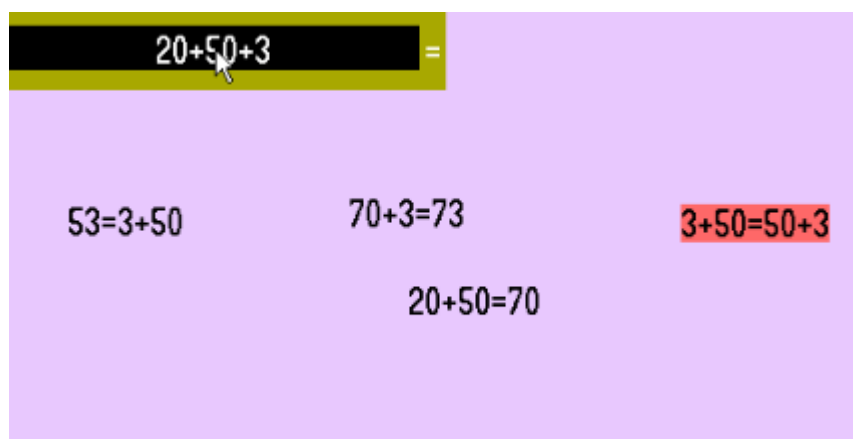
Ian Jones, *I.Jones@warwick.ac.uk*

CeNTRE, Institute of Education, University of Warwick, UK

Abstract

Young children attend to answer-getting readings of arithmetical notation. This is evidenced by many children's exclusive acceptance of $a + b = c$ syntaxes that lend themselves to computational readings (e.g. Behr et al., 1976; Carpenter and Levi, 2000; Knuth et al. 2006). Even those children who do accept a wider variety of syntaxes, such as $a + b = b + a$ and $c = a + b$, adhere to a computational view involving getting answers to both sides of the equals sign and checking they are the same (Jones, 2006).

I report here on the trialling of a Logo-based task (see screenshot below) designed to foster meaningful engagement with the structural properties of arithmetical equality statements. The design rationale is that of *diagrammatic reasoning* where arithmetic inscriptions onscreen are observed and manipulated according to precise operational rules. The dual functionality of selecting equality statements and substituting terms is intended to afford the specific diagrammatic activities of *iconic matching* and *transformation making* towards the construction of purposeful meanings for differing statement syntaxes.



The data show that the children's readings of arithmetic notation were broadened from computation to iconic matching during trialling. This pattern-based observation of notation combined with the activity of transformation making resulted in commutative and partitional meaning-making for $a + b = b + a$ and $c = a + b$ syntaxes respectively. The data also suggest that various factors impacted on the diagrammatic strategies developed by the children to complete the task. These factors included blind experimentation, systematic testing, existing computational readings, and emergent commutative and partitional readings.

Keywords

arithmetic, equals sign, meaning-making, task design

INTRODUCTION

Arithmetic is about performing computations in order to get a result (Hewitt, 1998). Accordingly, written arithmetic commonly appeals to answer-getting readings in the form of a string of numerals and operation signs followed by an equals sign and then the result, a sequence familiar to anyone who has used a simple calculator. There is rarely any need to appeal to commutative or partitional readings, as in $32 + 45 = 45 + 32$ or $77 = 45 + 32$, because all that is usually required is a final answer. Various studies have demonstrated that as a consequence many school children exclusively favour $a + b = c$ syntaxes. (e.g. Behr et al., 1976; Carpenter and Levi, 2000; Knuth et al. 2006). Even those children who do accept $a + b = b + a$ and $c = a + b$ syntaxes adhere to a computational view involving getting answers to both sides of the equals sign and checking they are the same (Jones, 2006). This closed, computational reading of arithmetic notation is of concern to mathematics educators on two levels. First, it is bereft of the operationalised relational thinking that Piaget (1952) reported present in young children in non-notational contexts. Second, many children encounter significant difficulties in later schooling when formal algebraic notating is required by the curriculum (McNeil and Alibali, 2005).

My focus here is on the first of these issues: the apparent gulf between young children's powerful mathematical thinking reported by Piaget and their narrow answer-getting readings of formal notation. I am not explicitly exploring learners' difficulties with the curriculum transition from arithmetic to algebra at the start of secondary schooling, although this issue is of implicit interest. My specific aim is to render structural readings of notation noticeable and "useful" (Ainley et al. 2006) by providing tools for manipulating onscreen arithmetic symbols towards a specified task goal.

DESIGN RATIONALE

Children's observations and manipulations of physical objects led Piaget (1952) to theorise an "operational plane" (p.220) of reversible equivalence and non-equivalence relations that is integral to the very notion of number itself. Quantitative and qualitative relationships were investigated by pouring liquids between differently shaped containers, arranging lines and piles of beads, matching eggs with egg-cups and so on. My research question might be framed as: How can we provide opportunities for children to observe and manipulate arithmetical symbols as they do physical objects? A way forward might be to provide learners with opportunities to engage in "diagrammatic activity" (Dörfler, 2006). Diagrammatic activity is the emergence of mathematical meaning-making through the systematic observation and transformation of "inscriptions" according to precise "diagrammatic operation rules" (p.105). Such activity is empirical and observation based, involving hypothesising and experimenting with inscriptions on a page or computer screen. Diagrammatic activities make no appeal to real-world metaphor or concrete referents. Instead, meaning-making is a social phenomenon requiring "a discursive context which offers a rich language to speak about the diagrams and their transformations" such that students "learn this language simultaneously with their development of the diagrammatic practice" (p.108).

In the remainder of this section I will describe a diagrammatic microworld I developed using *Imagine Logo*¹ (I chose Logo because it lends itself to the fast prototyping of microworlds). An example screenshot of the software, called *Sum Puzzles*, is shown in Figure 1. The software supports two basic functionalities: selecting equality statements and clicking terms. In Figure 1, $5 + 6 = 11$ has been selected by clicking on the equals sign. In this sense the inscription $=$ might be considered as a handle for taking hold of the equality statement. A term can be substituted by

¹ Imagine (Kalas and Blaho, 2003) has been developed by a team at Comenius University, Bratislava, Slovakia and is published by Logotron.

clicking on it and the resulting substitution (if any) is specified by the currently selected equality statement. For example, in Figure 1, if the inscription $5 + 6$ in the black box were to be clicked, it would change to 11 as shown in Figure 2. Likewise, if the inscription $5 + 6$ in the statement $5 + 6 = 6 + 5$ were clicked the statement would become $11 = 6 + 5$; if the inscription 11 in the statement $11 + 3 = 14$ were clicked the statement would become $5 + 6 + 3 = 14$. Note that substitutions are reversible: if the inscription 11 in the black box in Figure 2 were clicked while the statement $5 + 6 = 11$ is still selected the situation would return to that shown in Figure 1.

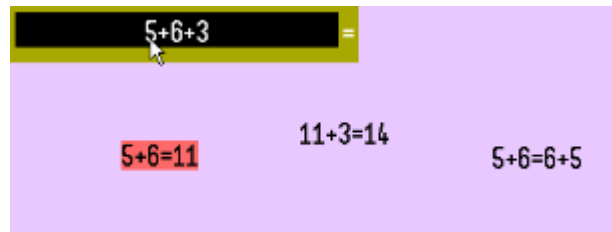


Figure 1: Sum Puzzles software

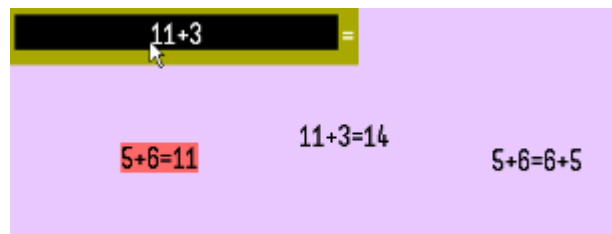


Figure 2: Substituting a term

The central goal is to transform the term in the black box into its answer by using the statements to make substitutions. The software presents a series of puzzles of increasing complexity (e.g. Figure 3). It is predicted that the embedded functionalities of selecting equality statements and clicking terms will afford the diagrammatic activity of *iconic matching* (Conjecture 1). Iconic matching is the process of looking for visually identical inscriptions. The emergence of such activity during trials would provide evidence that children's readings of arithmetical notation can broaden from computation to include pattern awareness through resource and task design.

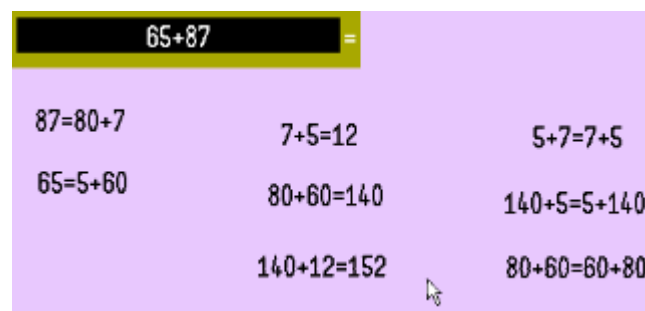


Figure 3: Puzzle containing eight statements

Gray and Tall (1994) have described operationalised readings as involving the ability to make reversible interchanges of equivalent arithmetic terms. In *Sum Puzzles* such reversible interchanges are carried out explicitly onscreen through *transformation making*, i.e. substituting a term for a second, equivalent term. Such transformation making potentialises a powerful

design principle, “using before knowing” (Papert, 1996). Evidence for emerging operationalisation would include children making commutative and partitional meanings for $a+b=b+a$ and $c=a+b$ statements respectively (Conjecture 2) when engaging in diagrammatic activity.

According to diSessa and Sherin (1998), existing and emerging ideas co-exist in fragmentary, often conflicting, patterns. Their “co-ordination classes” account predicts that children’s empirical experimentations with arithmetic turtles will be no simple story of diagrammatic activities replacing computational readings. Rather, diagrammatic activity and computation will impact upon one another and the trial data should be expected to show evidence of interference between the two when solving arithmetic puzzles (Conjecture 3).

METHODS

The two trials reported in this paper constitute one iterative step of my wider study. This iteration provides a test of the above conjectures, which were informed by previous pilot trials of the software (see Jones, in press).

Each trial involved two children working collaboratively through a sequence of 11 onscreen puzzles and lasted almost 40 minutes. The role of collaboration in task trialling is to foster a shared dialogue about diagrams in order to enable meaningfulness to emerge. The resulting composite data of onscreen manipulations and naturalistic discourse offers a window onto deep mathematical thinking (Noss and Hoyles, 1996). This method of data capture tends to be most insightful where the task is designed to challenge and reshape existing notions thereby rendering key shifts in mathematical thinking readily identifiable.

During the trials, each pair of children was first shown how to operate the software. For each puzzle I challenged the children to make the “answer” to the “sum” appear in the black box (Challenge 1). Twice in each trial I then set another challenge to get the original sum back again (Challenge 2). Challenge 2 was intended to emphasise reversibility but I dropped it soon into each trial as it proved ineffectual for generating eventful data. On other occasions when the data became uneventful I set a more difficult challenge: make it so that the term in the black box can be changed into its answer with a single click (Challenge 3). The distinction between Challenge 1 and Challenge 3 can be illustrated for the case of Puzzle 10 (Figure 3): Challenge 1 would be to transform $65+87$ in the black box into 152; Challenge 3 would be to transform the statement $140+12=152$ into $65+87=152$, and then use it to transform the term in the black box into 152 in a single click. Other than demonstrating the software and setting challenges, my role as researcher was to prompt for verbal elaborations from the children, mainly by asking “why” questions of the type “Why do you think that didn’t work?”

The children in the trial were aged 9 and 10 years (Year 5). The first pair of children, T and A, were boys deemed by their class teacher to be above average in mathematical ability. The second pair, K and Z, were girls deemed average in mathematical ability.

DATA AND DISCUSSION

In this section I will present a brief overview of the two trials followed by data and analysis relating to the three conjectures set out above. All excerpts are referenced by the time at which they began, in minutes and seconds.

Overview

Trial 1: T and A

T and A were confident with atypical arithmetical syntaxes at the start of the trial, as illustrated by their reaction the first time a statement of the form $a+b=b+a$ appeared onscreen (Puzzle 3):

08:37 T: 1 add 9 equals 9 add 1.

A: Yeah it does.

T: Yeah I know.

T and A got to grips quickly with the software and were able to develop strategies for solving puzzles (Challenge 1) confidently and efficiently. The most telling data was generated when I set Challenge 3 as this caused significant perturbations to their emerging (and successful) puzzle solving strategies. T and A's level of engagement was high throughout the trial, and characterised by cycles of (amused) frustration and triumph. The boys reflected on this at the end of the trial:

35:50 A: It's quite challenging [T: Yeah] It's kind of addictive in a way. You don't want to stop. [T: Yeah] Kind of like, really determined.

T: Bit fiddly. And it can sometimes get really annoying but it's still fun.

Trial 2: K and Z

K and Z were hesitantly accepting of $c = a + b$ syntaxes and unfamiliar with $a + b = b + a$ syntaxes as illustrated at the end of the trial when I asked them whether they had encountered such statements before. Only K claimed to remember encountering $c = a + b$ syntaxes previously ("I remember doing one of them in Year 4"), and neither girl remembered encountering $a + b = b + a$ syntaxes.

K and Z were somewhat reserved at the beginning of the trial but gradually opened up over the first 15 to 20 minutes. They solved some of the puzzles with apparent effortlessness and others caused them significant problems. Their inconsistent performance reflects a trial-and-error approach in contrast to T and A's strategic approach. Challenge 1 proved adequate for generating eventful data and the girls were not set Challenge 3 during the trial. Their engagement over the duration of the trial might be described as cautious experimentation increasingly punctuated by confident strategising during the final few puzzles.

Conjecture 1: Evidence for iconic matching

Conjecture 1 predicts that the functionalities of the software will broaden the children's readings of arithmetical notation to include iconic matching.

Initially during the T and A trial, readings were mainly computational, as in T's explanation of Puzzle 2 (Figure 4):

04:20 T: I think it might be this one. Yeah, [A: Yeah] 'cause 7 add 1 equals 8, 8 add 8 is 16.

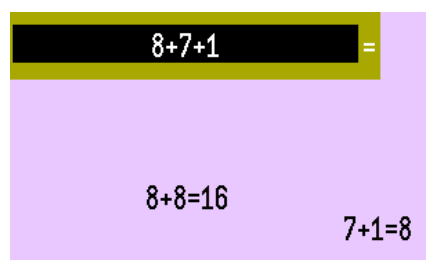


Figure 4: Puzzle 2

Evidence for a shift to iconic matching first occurred during Puzzle 4 (Figure 5):

10:46 A: 1 add 7 add 1 add 1.

T: 1 add 7 equals 7 add 1. Try that.

A: 7 add 1 add 1 add 1.

T: Okay. Let's try this one [$1+1+1=3$]. You want to do this one now [$7+3=10$]. Yeah. Now you do this one. It did it!

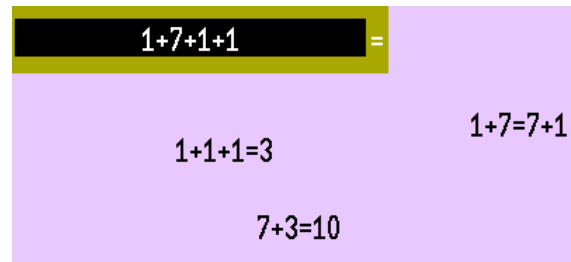


Figure 5: Puzzle 4

This approach, guided by iconic matches, came to dominate T and A's activity most of the time as can be seen in many of the data snippets in the following subsections. However, on occasion, when the boys became stuck, they attempted non-iconic substitutions. For example, at 28:35, T attempted a transformation of $65+87$ into 152 using the iconic mismatch $80+60+5+7=152$. At other moments of impasse they ignored iconic matching entirely and resorted to selecting a statement and systematically clicking every other term onscreen, though not with much expectation:

30:02 A: Try 12 equals 7 plus 5. ... Try all the numbers.

T: [laughs] Yeah, that's what I'm doing.

A: They're not going to work though.

T: Yeah, don't think it will.

K and Z were somewhat reticent during the first three puzzles and appeared to engage in arbitrary statement selecting and term clicking. The first explicit evidence for a computational reading occurred during Puzzle 4 (Figure 5):

07:05 K: 8 add 2 are 10.

Z: Yeah.

K and Z's grasp of iconic matching emerged gradually, and was preceded by attendance to non-iconic perceptual properties of equality statements. For example, when I asked why $1+7=7+1$ had made a substitution in $1+7+1+1$ (Figure 5) but $7+3=10$ had not, K answered (08:50) "is it because that's only got two numbers to add to 10 but that one's got four", and Z answered (09:15) "Is it because that one's got the equals sign in the middle? And that one hasn't."

Evidence for an emerging iconic reading first occurred during Puzzle 7. The statement $53=50+3$ had been selected and I asked why it made a substitution in $20+50+3$ when the inscription $50+3$ was clicked but not when $20+$ was clicked. Z answered (17:00) "Because you have to click on the 50 and there's more 50s there." From then on the girls became iconically guided in their decisions of which statement to try next. However, throughout their trial they also attended to iconic near-matches, such as attempting to make a substitution in $20+3+50$ using $20+50=70$ (at 17:20), much more frequently than did T and A.

Conjecture 2: Evidence for commutative and partitional readings

Conjecture 2 predicts the emergence of commutative and partitional readings of $a+b=b+c$ and $c=a+b$ statements respectively during diagrammatic activity.

Commutation

The children's awareness of commutation appeared to arise from the need to make implicit knowledge explicit in order to operate the software. For example, early on in the T and A trial, T attempted to transform $9 + 1$ into 10 using $1 + 9 = 10$ (R is researcher):

09:21 R: Why do you think that wasn't working?

T: [selects $1 + 9 = 9 + 1$] Maybe because... 1 and 9 is...

A: Oh, because it hasn't got that sum in it.

R: What do you mean?

A: Well 'cause that's got 1 add 9 but then the end of that's got 9 add 1.

T soon found a use for commutation when wanting to transform $6 + 5 + 3$ into $11 + 3$ using $5 + 6 = 11$:

12:45 T: I think we try that one [$5 + 6 = 6 + 5$]... try that and yeah [$6 + 5 + 3$ becomes $5 + 6 + 3$]. It's changed it. That makes it 5 add 6 so you can do this one now.

Around halfway through the trial T began to express commutation more consistently. Initially he used the phrase "changed around", then "turned round", and then "swap it round". From then on he consistently used the verb "swap" when suggesting and discussing commutative transformations.

K and Z's explication of commutation occurred more gradually. Early in their trial, the girls offered a computational equivalence (rather than a commutative) explanation for $5 + 6 = 11$ failing to transform $6 + 5 + 3$:

11:50 K: Is it because... um, that one won't work because it doesn't equal the same as the top.

R: What do you mean?

Z: That one's got 3 more.

K: That's a different answer to that one.

Just over halfway through the trial the girls began to express commutation using the verb "swap", as had T. This first occurred when K commented "that's just swapped it" after transforming $40 + 30 + 4$ into $30 + 40 + 4$ using $40 + 30 = 30 + 40$. However, across the two trials there was a notable contrast in the role of commutation. T commonly made strategic use of commutative transformations to produce statements required for a subsequent step towards solving a puzzle. Two examples from the transcript illustrate how commutation was a means not an end:

24:20 T: No. Oh you need to move 'em, you need to swap 'em round.

A: Hm.

T: No. Those need to be swapped round as well. ... There.

27:50 T: I'm thinking. 140 add 5, if you swap those over somehow. Try that.

In contrast, K and Z's use of commutation tended to be as part of a step-by-step iconic matching approach. For K, commutative transformations "just swap" numbers; but for Z, at least they do *something*:

31:15 Z: Try that on the other one.

K: No, it's just swapped them.

Z: Shall we try swapping and then we can try... [does not finish]

33:45 K: No, that's just swapping them.

Z: Yeah but, we could do it if it was swapped or something.

Only towards the end of the trial did Z come to notice the role of commutation in a sequence of transformations:

35:20 Z: I get this now 'cause like that one swapped those two, and there was two of those which means they've swapped, and like when you clicked on that one I thought they would have added to make 15 so it would be 300 and, add 15 and then click on that one 300 add 15 equals 315. That make sense?

Following this insight K utilised commutation strategically, as a means not an end:

36:20 K: So if we try and click on that one again [Z: Can we try on that one?] then that one would swap and then that would swap as well.

The data in this subsection suggest transformation making afforded T using-before-knowing; he made strategic use of commutation early and became increasingly adept at expressing it. In contrast, transformation making afforded K and Z knowing-before-using; they noticed they could “swap” things early and found a strategic use for doing so later. It should be noted that commutation was expressed exclusively as an action; for example, none of the children used “swap” as a noun.

Partition

The first $c = a + b$ form appears in Puzzle 6. After some initial discussion about which statement in the puzzle comes last (see snippet 19:40, next subsection) T commented on $41 = 40 + 1$:

20:00 T: Oh! That's the one that you do first! It has to be.

R: Why?

T: Because it's splitting up the 40 and the 1.

T's reaction here demonstrates his ability to infer a partitioning use for $c = a + b$ syntaxes from the diagrammatic rules supported by the software. From then on, making partitional transformations became the starting strategy for both T and A. For example:

22:50 A: Which one's most split up? That one.

T: Forty... 34 add... splits it up there.

A: Yeah.

T: Right. That's the only one that splits up so you just try the rest...

In one tantalising excerpt T appears almost to use “split” (or a derivative) as a noun to describe $87 = 80 + 7$:

24:00 T: Okay! 65 add 87. Any spli...? 8... Yeah. That one. That splits it up.

During Puzzle 9, K suggested using $34 = 30 + 4$ to transform $40 + 34$ in terms of the visual size of the resulting statement:

27:10 K: Try that one. Make it bigger. A bit bigger, then the answer.

However the data provide no other evidence for K and Z noticing or employing partition during the trial. There was no notable reflection on $c = a + b$ syntaxes and no specific vocabulary, such as “split”, employed when considering them (bar snippet 27:10, above). At the end of the trial I asked the girls what they thought the different types of statements on the screen meant. In reference to $109 = 100 + 9$ and $206 = 200 + 6$ they offered computational readings:

38:43 Z: Those mean, like, those two will equal that when you click on them.

K: That... the answer is 109, but like separate that would be 100 add 9 so it's just the... sum is like swapped round.

38:55 K: ...109 would normally be where the 100 and... the 100 plus the 9 would normally be where the 109 is so it's just that they're swapped.

It should be noted for contrast that, immediately prior to this, Z had described $a + b = b + a$ statements in terms of commutating:

38:41 Z: Erm, I think they mean that they swap.

The data in this subsection suggest that when T first encountered an $c = a + b$ inscription (snippet 20:00, above) he inferred a partitional meaning from his emerging grasp of the diagrammatic rules. However, Conjecture 2 was refuted for partitioning in the K and Z trial. When prompted, K and Z read $c = a + b$ inscriptions as computations written backwards as the wider literature would predict. This contrast in meanings for $c = a + b$ statements appears to be key to the contrast in puzzle solving performance across the two trials, discussed in the following subsection.

Conjecture 3: Evidence for interaction of diagrammatic activity and computation

T and A's puzzle solving strategy throughout much of the trial was sophisticated and efficient. Their emergent "splitting" meaning for $c = a + b$ statements enabled them to make sense of increasingly complex puzzles, and their strategic use of "swapping" to generate statements needed for subsequent steps enabled them to progress without difficulty. The impact of computational readings on their diagrammatic strategies appeared to be minimal and consisted merely of identifying which statement in a given puzzle needed to be left until last. For example, A correctly eliminated $70 + 1 = 71$ from his choice of two statements on the grounds that it is computationally equivalent to the term in the black box ($30 + 41$):

19:40 A: That one goes with that one, because that one's the sum you do at the end isn't it?

R: Which one's the sum you do at the end?

A: That one. Because 70 plus 1, because you get 71... that must be the same.

K and Z did not attend to the partitioning effects of $c = a + b$ statements and were less able to make sense of increasingly sophisticated puzzles. However, when starting a puzzle, they commonly identified the last statement as the one that contained the answer to the sum in the black box (e.g. snippet 07:05, above) as had T and A. Their general approach was that of experimenting with step-by-step iconic matches. This experimentation was guided by systematic testing of each and every statement rather than computational readings:

34:49 Z: I think we tried everything.

35:11 Z: I think we still haven't done that one yet. Yeah. So it will be 300 add 15 if it works.

Overall, the data suggest that Conjecture 3 is only weakly supported. Computational readings of equality statements had less impact on diagrammatic strategies than the previous pilot trials had led me to expect (see Jones, in press). Rather, partitional and commutative readings, along with looking for iconic matches one step ahead ("iconic strategising") and systematic testing of every statement and every term, appear to be the dominant features of the children's puzzle solving strategies.

CONCLUSION

Conjecture 1 states that the software and task design privilege pattern finding readings over computational readings. This is supported because all the children looked for iconic matches throughout the trials. Conjecture 2 states that the diagrammatic activities of iconic matching and transformation making afford meaning-making for the structure of equality statements. This is supported across the two trials for the case of commutation. It is also supported in the T and A trial, but refuted in the K and Z trial, for the case of partition. Conjecture 3 states that computational readings of equality statements impact on diagrammatic strategies and is only weakly supported across the two trials. It seems instead that commutative and partitional readings, iconic strategising and systematic testing were the more significant attributes of the children's diagrammatic experimentation and puzzle solving.

References

- Ainley, J., Pratt, D., and Hansen, A. (2006). Connecting engagement and focus in pedagogic task design. *British Educational Research Journal*, 32(1), 23-38.
- Behr, M., Erlwanger, S., and Nichols, E. (1976). *How children view equality sentences* No. PMDC-TR-3). Tallahassee, Florida: Florida State University.
- Carpenter, T., and Levi, L. (2000). *Developing conceptions of algebraic reasoning in the primary grades*. (Res. Rep. 00-2). Madison, WI: National Center for Improving Student Learning and Achievement in Mathematics and Science.
- diSessa, A., and Sherin, B. (1998). What changes in conceptual change? *International Journal of Science Education*, 20(10), 1155-1191.
- Dörfler, W. (2006). Inscriptions as objects of mathematical activities. In J. Maasz, and W. Schloeglmann (Eds.), *New mathematics education research and practice* (pp. 97-111). Rotterdam/Taipei: Sense Publishers.
- Gray, E. M., and Tall, D. O. (1994). Duality, ambiguity, and flexibility: A "proceptual" view of simple arithmetic. *Journal for Research in Mathematics Education*, 25(2), 116-140.
- Hewitt, D. (1998, Approaching arithmetic algebraically. *Mathematics Teaching*, 163 19-29.
- Jones, I. (in press). Meaningful engagement with the structure of equality statements. *Working Papers of the Warwick Sumner Group*, 3.
- Jones, I. (2006). The design of equality statements. *Working Papers of the Warwick Sumner Group*, 2, 63-82.
- Kalas, I., and Blaho, A. (2003). Exploring visible mathematics with imagine: Building new mathematical cultures with a powerful computational system. In G. Marshall and Y. Katz (Eds.), *Learning in School, Home and Community* (pp. 53-64). IFIP Conference Proceedings. Manchester: Kluwer Academic Publishers.
- Kieran, C. (1981). Concepts associated with the equality symbol. *Educational Studies in Mathematics*, 12, 317-326.
- Knuth, E., Stephens, A., McNeil, N., and Alibali, M. (2006). Does understanding the equals sign matter? evidence from solving equations. *Journal for Research in Mathematics Education*, 37(4), 297-312.
- McNeil, N., and Alibali, M. W. (2005). Why won't you change your mind? knowledge of operational patterns hinders learning and performance on equations. *Child development*, 76(4), 883-899.
- Noss, R. and Hoyles, C. (1996). *Windows on Mathematical Meanings: Learning Cultures and Computers*. Dordrecht, The Netherlands: Kluwer Academic Publishers.
- Papert, S. (1996). An exploration in the space of mathematics educations. *International Journal of Computers for Mathematical Learning*, 1(1), 95-123.
- Piaget, J. (1952). *Child's conception of number*. London: Routledge and Kegan Paul Ltd.

LEGO Robots for Classroom Experiments

Martina Kabatova, *kabatova@fmph.uniba.sk*

Department of Informatics Education, Comenius University in Bratislava

Janka Pekarova, *jana.pekarova@fmph.uniba.sk*

Department of Informatics Education, Comenius University in Bratislava

Abstract

LEGO Mindstorms NXT Kit with programmable NXT Brick and variety of sensors is an interesting tool for programming and projecting various robots in many different subjects - science, technology and math. Students can put their knowledge into practice by building smart robots, collecting data and presenting their results. They are challenged to solve problems and make up their own new ideas. Behind programmable LEGO there is the classical constructionist idea - children learn best by doing things, especially if they are engaged in common project. Lego programming environment is designed to create icon-based programs that instruct NXT sensors and motors.

Since year 2000 programmable LEGO kits have been distributed to many schools in Slovakia. Each year new teachers can attend courses to learn how to use LEGO kits in the classroom. Working with LEGO helps to develop communication and relationships between students in the classroom as well as between students and the teacher.

Activities with robot in detail

Participants of the workshop will work in several 3-people groups. Each group shall complete their own robot developed for special purpose. Tasks will differ among the groups therefore robots should be completed by adding different sensors. After constructing the robot the group will program the robot by instructing it with a short program that will accomplish the given task.

Examples of simple tasks for robots:

1. Go forward. If there is a noise, change the direction of movement.
2. Go forward. If there is an obstacle, change the direction.
3. Go forward. If you bump into the wall, move back.

Follow the black line. If you find the light area, stop.

Keywords

LEGO, robot, programming, sensor, NXT

Should LOGO Keep Going FORWARD 1?

Ken Kahn, toontalk@gmail.com

London Knowledge Lab, Institute of Education, University of London and Oxford University

Abstract

LOGO has been evolving in incremental steps for 40 years. This has resulted in steady progress but some regions of the space of all programming languages for children cannot be reached without passing through unacceptable intermediate designs. What are the ultimate aims of LOGO? What criteria and aesthetics should be used in determining which areas of the design space are most promising? What would the ideal programming language look like? Would a family of special-purpose languages be more effective than a single language?

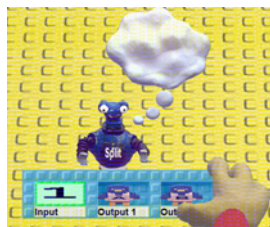
In looking to the future what can we learn from the history of LOGO? What can we learn from other programming systems for children? Alan Kay is leading a new project entitled, "Steps toward the Reinvention of Programming". What are its strengths and weaknesses?

We can conceptualise the design alternatives as defining an n-dimensional space. Some dimensions represent major alternatives for syntax, others for dealing with concurrency, others for the underlying computational models, and others for features of the programming environment.

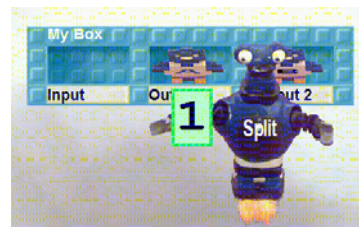
The goal of this paper is to spur a discussion of these issues. I will present my personal opinions based upon 30 years of research experience in this field.

Keywords

Future of LOGO; Smalltalk; ToonTalk; StageCast Creator; AgentSheets; NetLogo; StarLogo



Training the *Split* robot



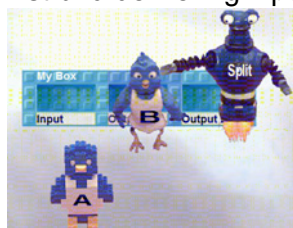
Giving the input to the first bird



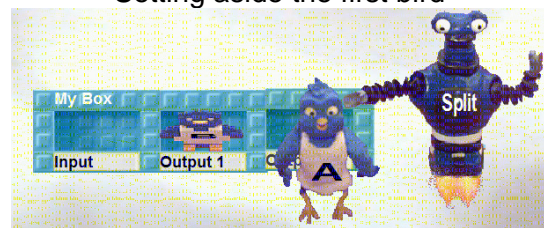
First bird delivering input



Setting aside the first bird



Moving second bird to first bird's box



Putting first bird in second bird's box

Figure 1 – Training a Robot to Split a Stream of Numbers

The Trajectory of LOGO

I could not agree more with you about current Logo being out of date and am planning to immerse myself in thinking about what a language for 2005 (or so) could be.

- Seymour Papert (personal email to the author, June 14, 1999)

The Past

In the last 40 years LOGO has moved FORWARD 1 many times. Sometimes it has been cloned and copies have headed off in somewhat different directions. Colour was added. 3D was added to some branches of the LOGO tree. Object-orientation and multiple turtles were incorporated. Concurrent processes were supported. Advanced user interface gadgets were added. There have been many valuable improvements.

Sometimes language designers imagined languages that could not be reached incrementally from LOGO and new child-oriented programming languages appeared elsewhere in the design space. Smalltalk (Kay 1993) was probably the first language that was inspired by the LOGO “philosophy” but not part of the language evolution. Boxer (diSessa 1997) and ToonTalk (Kahn 2001b) are other examples. These languages borrowed some of the powerful ideas of LOGO but did not grow out of LOGO itself. Other languages such as AgentSheets (Repenning et al 2000), StageCast Creator (Smith et al 2000), and Alice (Conway et al 2000) share many of the goals and ideas underlying LOGO but developed without explicit influence from LOGO.

In some cases, separate language trees have branches that share the same region of design space. The Etoys system of Squeak/Smalltalk (Allen-Conn and Rose 2003), for example, is similar to some modern LOGO systems.

Possible Futures

One future is for LOGO to continue to improve incrementally. The difficulty here is that radical improvements are not incremental and are frequently disruptive. The intermediate points in the design space are often not viable: being “neither fish nor fowl”.

Another future is described by Alan Kay and colleagues at the Viewpoints Research Institute in a recently funded 5-year project (hereafter called the VPRI Project) to reinvent programming, especially for children (Kay et al 2006). It proposes to build a new computing system and programming language with many innovative properties: self-explanatory systems, separation of intent and optimisations, a self-modifiable implementation, extreme portability, well-integrated and transparent tools and operating system functionality, and a classless prototype-oriented object model.

Another future is that papers such as this one sparks discussions in the wider community that lead eventually to the design and implementation of a new language (or languages).

What are the Ultimate Goals?

What are our ultimate goals in designing the ideal programming language for children (and other learners)? One answer is that we want to give kids (and as many of them as possible) the power to express themselves computationally. Alan Kay expressed it well in his 1984 *Scientific American* article (Kay 1984):

The protean nature of the computer is such that it can act like a machine or like a language to be shaped and exploited. It is a medium that can dynamically simulate the details of any other medium, including media that cannot exist physically. It is not a tool, although it can act like many tools. It is the first metamedium, and as such it has degrees of freedom for representation and expression never before encountered and as yet barely investigated. Even more important, it is fun, and therefore intrinsically worth doing.

... Computers are to computing as instruments are to music. Software is the score, whose interpretation amplifies our reach and lifts our spirit. Leonardo da Vinci called music “the shaping of the invisible,” and his phrase is even more apt as a description of software.

We want a language that is truly general purpose and not based upon a limited special-purpose computation model. Or, as discussed below, maybe the goal shouldn't be a single programming language but a family of languages.

Besides giving children the ability to express themselves in fundamentally new and powerful ways, we also want to give them objects to think and learn with. This is the deep idea underlying Papert's *Mindstorms* (Papert 1980).

Thirdly, we want to give children new mathematically interesting objects to think *about* as well as think *with*. One can direct one's thoughts at a programming language itself and appreciate it as a mathematical object and as a model of computation. Perhaps Lisp and its underlying lambda calculus are at least as worthy of study as, say, trigonometry.

I think LOGO has compromised the goal of providing a language to think about by giving priority to other goals. Lisp and Prolog, for example, can describe themselves in very short programs. LOGO, with its distinction between functions and commands, its syntax, and the special forms (*IF*, *WHILE*, etc.) of many dialects, make it much harder to implement LOGO in LOGO. And it makes it harder to think about how LOGO works and what it is.

The VPRI Project led by Alan Kay aims to build a language and system that is simpler and better suited for self-description and self-inspection. It also is exploring the possibility of making the system *self-explanatory*. Their plan is that the system can explain its own structure and operations. Using AI techniques, it will automatically generate explanations to support end-user exploration.

Whose Design Aesthetics?

The design of LOGO was largely based upon Lisp. The design of Lisp was largely based upon the *lambda calculus*, a branch of mathematics. This makes Lisp (and to a lesser extent LOGO) a language that is not only a good tool for expressing programs but an object to think about and with. One consequence of this is that Lisp is well-suited for meta-programming. Programs can construct other programs. Programs can reflect upon themselves. This is a consequence of the small powerful kernel underlying Lisp that is based upon lambda calculus. Alan Kay in a talk at Stanford in 2003 described Lisp as “a mathematical object that can see itself”. The idea of a self-referencing language kernel is one of the inspirations behind the VPRI Project. The kernels of some programming languages such as Lisp, Prolog, concurrent constraint programming languages, and functional programming languages have a mathematical beauty. A very small basis set can generate incredible richness. In contrast, languages like Smalltalk, Oz, Python, and Java were designed by computer scientists. The beauty and elegance of these languages seems more akin to that of engineering than mathematics. We either have to choose which aesthetic will drive the design or take the gamble that we can design something that simultaneously is a mathematical and engineering jewel.

If one is pursuing mathematical beauty then basing the language design on well-established mathematics is much easier and safer than also inventing new mathematics. For example, a language can be based upon lambda calculus which is a theory of functions or instead on a calculus of processes such as the *pi calculus*. Perhaps we should be building languages based upon processes rather than functions since processes are more valuable and fundamental. If so, the design of the ideal programming language for children should focus upon doing concurrency right and not be as concerned about functional programming.

It may be that in practice even the mathematically beautiful programming languages are complex engineering artefacts. Maybe the structure and elegance of the kernel matters little if programmers need to put most of their efforts into understanding large engineered libraries of

useful components. Not much real programming is done completely bottom up from language primitives. And even if someone builds a large program “from scratch”, much of their cognitive effort typically goes into designing and using the higher level chunks of code.

I think this situation is analogous to the idea of reductionism in science and philosophy. It is a great achievement that one can understand everything in the universe in terms of atoms, or elementary particles, or quarks, or super strings. It is important that things can, “in principle”, be reduced to primitive elements. In theory, it can explain why you can't put a round peg in a square hole. But usually it is not the level at which informative, helpful, or satisfying explanations come from. Similarly, I think it is great if one can see that a complex program bottoms out ultimately in a small set of primitives even if most of one's thinking about programs occurs at higher levels of organization. In science one discovers new entities and laws that emerge at different levels. Maybe we should be looking for these emergent properties at higher levels of software as well.

The VPRI Project aims to go one step beyond an elegant software kernel by connecting the kernel primitives to the physical hardware in a transparent and customisable fashion. By building upon some clever bootstrapping techniques they plan to build self-describing kernels that also describe mappings to the machine language of the target machines. The hope is that curious non-professionals will be able to understand the entire system down to the metal.

A very Brief Introduction to ToonTalk

I began designing and building ToonTalk in 1992 (Kahn 1996). I was inspired by Seymour Papert's description of LOGO as taking the best ideas from computer science about programming languages and environments and “child-engineering” them (Papert 1977). I took computational ideas from the concurrent constraint programming field (Saraswat 1993) and user interface ideas from computer games and made a language that looks and feels like a game but is really a powerful concurrent programming language. The child enters a virtual world and constructs programs by training animated robots to manipulate boxes, birds, trucks, numbers, and various other tools. Programs are constructed by demonstration with examples (Kahn 2001a). The fundamental idea is that sophisticated computational abstractions can be made accessible by providing concrete analogues without loss of expressive power.

Design Aesthetic of ToonTalk

Is ToonTalk an example of a design guided by engineering aesthetics? It appears to be in the engineer/computer scientist family of languages. It has many more primitive elements and constructs than say lambda calculus underlying Lisp or the Horn clauses underlying most logic programming languages. And yet its design was directly inspired by concurrent constraint languages which are mathematical beauties. What happened?

ToonTalk has very few true building blocks. There are pads (atomic data), boxes (compound data structures), birds/nests (communication channels), and robots (program fragments). What about all the other things in ToonTalk? They are ways of expressing certain kinds of actions, not things themselves. A truck is not really a part of a ToonTalk computation but is a way of expressing the spawning of a new process. The Magic Wand is not a thing but a way of expressing the copying of other things. The helicopter is a way to monitor an ongoing computation at different scales and locations. Notebooks were designed as a way to obtain the functionality of a file system for persistence and sharing. But notebooks can also be used as an alternative to boxes and from a mathematician's view this overlapping of functionality is ugly.

ToonTalk often operates at a level below that of the concurrent constraint programming languages. In Janus (Saraswat et al 1990), you can understand communication as the asking and telling of constraints. In ToonTalk, the corresponding constructs are waiting for things to arrive on a nest (asking) or giving things to birds (telling). Despite a direct mapping between Janus and ToonTalk, it is hard to see the underlying constraint programming of ToonTalk.

Is the ToonTalk world too rich? Would a sparser set of primitives lead to a language that is a better object of study and contemplation? Would a sparser ToonTalk have more layers of

software? Could these higher-level components be concretised as well as ToonTalk currently does? Many good questions remain.

The Ideal Programming Language

After 15 years of ToonTalk work and experience I now imagine an ideal language as retaining many of the concepts and strengths of ToonTalk with these differences:

1. **Tiny kernel.** The kernel of the language should be as elegant and powerful as possible. The design of a language like ToonTalk could be guided by the aesthetics of mathematicians.
2. **Multiple representations.** ToonTalk's support for programming with concrete examples in an action-oriented fashion using metaphorical analogues of computational abstractions could be augmented with corresponding static pictorial representations such as comic books (Kindborg 2001) and a textual or tile-based symbolic language. Discussed in detail below.
3. **Composing components.** One lesson from several research projects that I have participated in (Playground (Hoyles 2002), WebLabs (Kahn et al 2005), BBC (Kahn et al 2006), ReMath, and Constructing2Learn (Kahn 2007)) is that a very effective way to enable beginners to quickly build programs they care about is to provide them with a library of easily composable high-level components. These components should run as autonomous processes with no or minimal interfacing. Children can start programming "in the middle" and move "up" by composing program pieces and move "down" to understand, modify, or create new components.
4. **Exact arithmetic.** ToonTalk's exact rational arithmetic (Kahn 2004) could be augmented with exact irrational numbers (Boehm, H-J. 2004). Rather than mislead or confuse children with "leaky abstractions" for numbers (e.g., limited precision floating point numbers), we could provide them with exact implementations of rational and real numbers. One technical challenge with computable irrational numbers is that the determination of whether a number is equal to another may not terminate.
5. **Exact geometry.** Perfect or ideal geometry could be supported. We should explore whether it is feasible to build geometrical objects that are pixel perfect at any scale or size. Imagine a circle or a fractal consisting of dimension-less points that glow. As one zooms in on an increasingly small portion of the object, the screen pixels are computed accurately. At extreme scales, such as a googol-fold zoom, the system may begin to slow down as the exact rational or real arithmetic may become relatively expensive. Turtle geometry could also be implemented using exact real arithmetic (or an approximation that ensures that no anomalies are observable). Such an implementation would make "real" and "concrete" the idealizations of geometry.
6. **Time travel.** A better version of time travel could be supported. ToonTalk provides users with the ability to travel in time (Kahn 2006). They can go back to an earlier time in their session and replay or revise the past. These time travel records can be copied and shared with others. The implementation sometimes imposes an unacceptable performance penalty upon the system but in other cases it is very useful for undoing, reviewing, and creating demos for others. A new implementation could dramatically reduce the performance cost in many cases. ToonTalk's time travel enables one to replay the past from the viewpoint it was viewed originally. A more general and useful version would allow one to "move the camera" while replaying and maybe to join the scene as an (observer-only) ghost until such time as one is ready to fork the time line.
7. **Debugging tools.** Children are rarely given a powerful set of debugging tools. Too often they have few alternatives to adding *SHOW* or *PRINT* statements to their programs. Time travel is a powerful debugging tool but it is not a complete solution. Children should

have the kinds of tools that a modern Integrated Development Environment such as Eclipse offers. The challenge here is to child-engineer the user interface without reducing its power.

8. **Run everywhere.** Programs can run on mobile devices, web browsers, and robotic construction kits. While I expect that programs will typically be constructed and debugged from inside a virtual world, they could run as well in browsers, mobile devices, and other platforms. ToonTalk programs currently can be automatically converted to Java and run as applets in browsers. Perhaps something similar could be done to run on other platforms as well. It probably isn't practical to expect the programming environment to run on these more limited devices or contexts but when an application can be isolated from the environment it should be possible to generate a stand-alone version with much lower run-time requirements.
9. **For professionals too.** ToonTalk is rarely used by professional programmers to accomplish their tasks. Perhaps the ideal language for children and non-professionals should provide support for professionals as well. NetLogo has succeeded in simultaneously supporting students and researchers in building and exploring simulations of multi-agent models.
10. **Distributed implementation.** ToonTalk's model of concurrent and distributed computation should be preserved but with a more general and flexible implementation. A distributed persistent implementation (perhaps building upon distributed hash tables) could provide a foundation for inter-process communication to achieve peer-to-peer distribution, persistence, and scalability. This could provide a unique and powerful means of collaboratively programming over the network.
11. **Inside a virtual world.** Programming would take place within a 3D persistent shared on-line virtual world with an integrated physics engine. Discussed in detail below.

Another idea is that the language should run with zero installation, possibly as a web service accessed via a browser. The storage of programs on the web service would enable students to seamlessly continue to work on projects as they move from school to home and back again. The implementation constraints entailed by web services will make the above list of enhancements even more challenging to accomplish.

How should the Ideal Language Support Concurrent and Distributed Computation?

Over the last ten years, I've had many technical discussions about why I believe that ToonTalk's model of concurrency is better than that provided by multi-threaded LOGO implementations (including StarLogo and NetLogo) and Squeak.

Very briefly, what we want are not multiple processes that cause side effects on the environment, but processes that also consume and produce data. These processes need ways to communicate, synchronize, and coordinate via primitives that are accessible by non-expert programmers. The way ToonTalk does this enabled, for example, the computational exploration by children of infinity and cardinality (Kahn et al 2005).

Concurrency combined with destructive operations upon shared data and variables leads to race conditions and other very hard to track down bugs. Attempts to introduce locks and atomic actions add complexity and the risk of deadlock. I strongly believe that destructive operations upon shared data should not be part of any language with a general model of concurrency. And non-general models of concurrency such as that of StarLogo and NetLogo are useful only in limited situations.

Furthermore, I believe we want to support children in building distributed computations. This need not add complexity to the system if the model of concurrency generalizes to processes running on different computers communicating over a network, as is the case in ToonTalk. Despite limitations of the partial implementation of distributed computing in ToonTalk, various

networked games have been built such as *Battleships* and two-player *Pong*. Swiss high school students have implemented a chat system in ToonTalk.

Also the communication mechanism used by programs should be of use for human-to-human communication as well. In the Playground Project young children used ToonTalk's long-distance birds to exchange their ToonTalk games as well as text messages.

The concurrency of a programming language should have a solid theoretical foundation such as that offered by the pi calculus.

What is lost by making computational abstractions concrete as ToonTalk does?

Let's consider an example. Suppose we want a robot that can take a stream of incoming objects and produce two outgoing streams each containing every other element. Figure 1 is a series of pictures of the training of such a robot.

Here's the equivalent robot in pseudo-code:

```
to Split In Out1 Out2
  if In receives X then
    send X on Out1 and
    Split In Out2 Out1 // notice the arguments are swapped
end
```

I believe there are many children and adults that understand how the *Split* robot works, especially after watching it in action, will find the equivalent *Split* procedure perplexing. And maybe an even smaller proportion of those who can program the *Split* robot could construct the *Split* procedure.

But have those who can program the *Split* procedure lost something by using ToonTalk instead? One thing they may have lost is the ability to see “at a glance” what the program does without running it. A series of pictures as in Figure 1 helps but currently there are no tools for producing them automatically. While ToonTalk can generate a verbal description of a robot, it is often at too low a level of detail.

One answer to the question of what is lost is a formal representation of the program. This is only of value to those who are good at thinking using abstract formalisms. But what about those learners who aren't good at using abstract formalisms but would become so if they put in the effort to learn an abstract programming language?

Another answer is that the ideal language should support multiple representations that range from ToonTalk-like concretizations, to series of pictures, to symbolic representations. And the language should support the easy movement between these representations for the same program fragment. There are many challenges to designing a seamless multi-representation language without comprising any of the individual representations.

One Language or Many?

The community has been producing, and will continue to produce, many programming languages for children. Should we design a coordinated suite of languages and offer that to children or should we focus our resources on the “ideal” language? The argument for multiple languages is that each one may offer different strengths and ways of thinking about computation. The set of ideal languages for children may include those based upon different paradigms of programming as well as different levels of concreteness. Special purpose languages may have a role in such a basket of languages since they may provide superior support for certain classes of problems.

Multiple languages that are too similar are counter productive. The LOGO community has suffered from many incompatible dialects; (Boychev 2007) lists 173 LOGO dialects.

Ideally different languages should interoperate together well. In some cases it may even be possible for some languages to emit equivalent code for import into a different language.

If these languages can be tightly integrated then is it best to think of them instead as a single multi-paradigm, multi-layered language?

Some have argued for different languages for different age ranges of children. My experience with ToonTalk is that this is not necessary. Children, as young as 3, have built ToonTalk programs (Morgado 2003), while university students have used it to explore concurrent algorithms. By having a diverse user base, children of different levels of experience and expertise are more likely to help each other than if they are using different languages.

One danger of multiple languages is that they can obscure how apparently different things may be fundamentally similar. If, for example, a model of the spread of diseases and a model of the spread of technological innovations are built in the same language then the similarity of these processes should be easier to perceive.

Where should Program Construction Take Place?

This may seem like a strange question. Some people construct programs on paper and then enter them into a computer. Some people type their programs into their favourite text editor. Others use editors specialized for the programming language. Some programming languages support program construction from within the programming environment. A recent trend here is to use composable tiles representing program fragments as in Etoys (Allen-Conn and Rose 2003) and Alice (Conway et al 2000).

ToonTalk is unique in that program construction takes place from within a game-like virtual world. It takes place, not by supporting the editing of textual or pictorial programs from within a virtual world, but instead by taking direct “everyday” actions in this world. The ToonTalk world is cute and playful and popular with pre-teens but it isn't a place where people spend much time for purposes other than constructing, debugging, and running ToonTalk programs. Some children spend many hours decorating houses, filling notebooks with artwork, playing with text-to-speech engines, and doing mathematical explorations with ToonTalk's exact rational arithmetic. But “being there” is rarely the main purpose of visiting the ToonTalk world.

How might persistent shared on-line virtual worlds fit into the big picture? *Second Life* is a nice example of such a place where the “residents” have built a great variety of places and things in this world (Ondrejka and Cook 2005). *Croquet* has a similar vision (Smith et al 2003). About ten million people regularly visit such places. People visit these places primarily for entertainment and social reasons but there are more “serious” activities.

What if one could construct programs from within these worlds in a manner similar to ToonTalk programming? It could be done in a way that is quite similar to how ToonTalk currently works but with these differences:

1. **3D.** ToonTalk uses sprite animation and 2½ dimensional graphics to provide a virtual world. This simplifies but limits things. 3D enables the explorations of many topics in science and mathematics. And it enables the programming of very popular classes of games. It supports a wider set of contexts for activities. The challenge is to make a 3D world that can be navigated and programmed by all. Progress has been made in making 3D programming broadly accessible with AgentCubes (Repenning and Ioannidou 2006), Alice (Conway et al 2000), Elica (Boychev 2003), and forthcoming versions of NetLogo and StarLogo.
2. **Realistic physics.** Many virtual worlds today have “physics engines” that support collisions, gravity, friction, rigid body motion (with joints), and more. It is unreasonable to expect such physics engines to be built by children. But they can be customised and parameterised by all. A programmable world with a built-in physics engine enables many

explorations in sciences (e.g., sports science or mechanics). And it opens up a whole new class of games that children can create.

3. **Virtual communities.** Inhabitants of a shared on-line virtual world can provide help to each other in ways that are currently feasible only in face-to-face encounters. One can meet and build things together. Someone half way around the world can help others build and debug a program by being in the same place, manipulating the same objects, and conversing the whole time. People currently collaborate in ways that are more awkward using web sites, email, and the like. These communities can also have out-of-world support from associated "Web 2.0" sites.
4. **Living with your creations.** One is more motivated to build things that work in the place where one spends one's time. If people inhabit these spaces for reasons other than programming, then programming becomes a tool for enhancing their "living space". Their creations are things they can enjoy while they are in a virtual world for social or entertainment reasons, rather than objects that exist only when they are in some programming system. Examples include virtual pets, useful gadgets, kinetic sculptures, and long-lasting simulations.

Where does Turtle Geometry fit into the Picture?

Despite my love of turtle geometry, it is not part of ToonTalk. In 1995, I tried to add it and began user testing. The idea was that there was a bird for each picture. If you gave the bird a box containing the text pad *FORWARD* and the number pad *100* then the bird would deliver the message to the picture and it would move forward 100 units in the direction of its current heading. Non-programming adults and a fourth-grade class seemed to understand the concepts of ToonTalk (despite its primitive state) but were unhappy with this message passing interface. It was too indirect and clumsy.

I redesigned ToonTalk to include remote controls to support picture programming in a more direct intuitive manner. For example, a sensor for the horizontal position of a picture appears just as numbers do (except it has an animated marquee to indicate its special status). One can manipulate this remote control as an ordinary number and the associated picture's position changes accordingly. This also works in the other direction: if the picture is moved, the number is updated. This scheme is similar to the property sheets used in a huge variety of software where the properties show the current state of an object and can be edited to change the object. In ToonTalk the equivalent of a property sheet is broken up into small pieces corresponding to single properties. This provides a nice "declarative" interface for objects. Children as young as six in the Playground Project made heavy use of these ToonTalk remote controls (Hoyles 2002). But these remote controls only support Cartesian geometry and don't support turtle geometry.

So why not add *FORWARD* and *RIGHT* sensors to ToonTalk? The equivalent of a *RIGHT* sensor would be straightforward; just add a sensor for the heading of a picture. The equivalent of *RIGHT 90* would be to drop a number pad with *90* on the heading sensor and the current heading will be incremented by 90 degrees. But what sensor would play the role of *FORWARD*? This stumped me for many years. The best idea I have is to introduce a sensor for the picture's distance to where you want it to be (in the direction of its current heading). So if you dropped *100* on such a sensor the picture would move forward 100 steps. The really odd thing is that this sensor would always display *0* since as soon as it is changed the picture moves and it is where you instructed it to be. Maybe a better version would be a sensor for the distance to its "goal". If its speed is infinite then it will also always display *0* but if it is given a finite speed it will glide towards its goal and the value displayed will decrement at the speed until it reaches zero. Some other design issues remain such as how to integrate the trails left by turtles' pens. Are they new objects or are they alterations of the surface they are drawing on?

I now believe that rather than choosing between message passing and declarative user interfaces (property sheets/remote controls) that they should coexist at different levels. The

primitive low-level way of manipulating pictures could be via message passing as I originally intended a dozen years ago. This will probably be used only by more advanced users, particularly those enhancing the system itself. The ideal system should provide primitive support so that remote controls can be built within the system. They would behave much as they currently do in ToonTalk but their implementation would be transparent. Ordinary users could inspect, edit, and create new kinds. The challenge here isn't just to implement sensors within the language but to do so in a manner that non-experts can understand. This is one of the challenges the VPRI Project led by Alan Kay is addressing.

More generally, I think a good way to proceed is to open the development of new "primitives" to a wide community. The kernel of the system can be small but include a way to "plug in" modules written in nearly any programming language. This could be based upon an improved version of ToonTalk's foreign birds that provides a way to use birds to interface external code. The bird or message passing level of interface can then be built upon to provide higher-level functionality if need be.

Conclusions

In the 40 years since LOGO was born there has been good incremental progress. Computer science has also made substantial progress in models of computation and user interfaces. Building upon this progress and drawing upon our experiences with LOGO, Smalltalk, Boxer, AgentSheets, StageCast Creator, ToonTalk, Alice, StarLogo, NetLogo, and Scratch we should be discussing where to go next. We should jump to promising regions of the design space rather than making slow and steady turtle-like progress. Rather than *REPEAT n [FORWARD 1]* we need to determine a good (n-dimensional) *heading* and go *[RIGHT heading FORWARD 1000]* to the next generation of programming languages for children of all ages.

References

- Allen-Conn, B. J., Rose, K. (2003) *Powerful Ideas in the Classroom Using Squeak to Enhance Math and Science Learning*, Viewpoints Research Institute.
- Boehm, H-J. (2004) *The constructive reals as a Java Library*, HP Laboratories Palo Alto, HPL-2004-70, <http://www.hpl.hp.com/techreports/2004/HPL-2004-70.pdf>
- Boychev, P. (2003) *Turtle Metamorphoses (From "FD 1" to 3D Animated Characters)*, 9th European Logo Conference EuroLogo 2003, Porto, Portugal, <http://www.elica.net/download/papers/TurtleMetamorphoses.pdf>
- Boychev, P. (2007) *LOGO TREE PROJECT*, <http://www.elica.net/download/papers/LogoTreeProject.pdf>
- Conway, M., Audia, S., Burnette, T., Cosgrove, D., and Christiansen, K. (2000) *Alice: lessons learned from building a 3D system for novices*, SIGCHI Conference on Human Factors in Computing Systems, The Hague, The Netherlands, April 2000, ACM Press, New York, 486-493, http://portal.acm.org/ft_gateway.cfm?id=332481&type=pdf&coll=GUIDE&dl=portal,ACM&CFID=14816456&CFTOKEN=33685471
- diSessa, A. A. (1997), *Twenty reasons why you should use Boxer (instead of Logo)*, in M. Turcsányi-Szabó (Ed.), *Learning & Exploring with Logo: Proceedings of the Sixth European Logo Conference*, Budapest Hungary, 7-27, <http://www.soe.berkeley.edu/boxer/20reasons.pdf>
- Hoyle, C., Noss, R. and Adamson, R. (2002) *Rethinking the Microworld Idea*, Journal of Educational Computing Research, Volume 27, Number 1-2, 29 - 53.
- Kahn, K. (1996) *ToonTalk -- An Animated Programming Environment for Children*, Journal of Visual Languages and Computing, June 1996, <http://www.toontalk.com/Papers/jvlc96.pdf>

- Kahn, K. (2001a) *Generalizing by Removing Detail: How Any Program Can Be Created by Working with Examples*, in Lieberman, H. editor, *Your Wish Is My Command: Programming By Example*, Morgan Kaufmann.
- Kahn, K. (2001b) *ToonTalk and Logo - Is ToonTalk a colleague, competitor, successor, sibling, or child of Logo?* EuroLogo Conference, August 2001, <http://www.toontalk.com/Papers/logott.pdf>
- Kahn, K. (2004) *The Child-Engineering of Arithmetic in ToonTalk*, *Interaction Design and Children Conference*, College Park, Maryland, <http://www.toontalk.com/Papers/idc04.pdf>
- Kahn, K., Sendova, E., Sacristan, A.I., and Noss. R. (2005) *Making Infinity Concrete by Programming Never-ending Processes*, 7th International Conference on Technology and Mathematics Teaching, Bristol, UK, http://www.toontalk.com/English/card_abs.htm
- Kahn, K., Noss, R., Hoyles, C. and Jones, D. (2006) *Designing digital technologies for layered learning*, *Informatics Education – The Bridge between Using and Understanding Computers*, *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg.
- Kahn, K. (2006) *Time Travelling Animated Program Executions*, *Software Visualisation Conference*, Brighton, UK, <http://www.toontalk.com/papers/timetravel3.pdf>
- Kahn, K. (2007) *Comparing Multi-Agent Models Composed from Micro-Behaviours*, *Third International Model-to-Model Workshop*, Marseille, France, March 2007.
- Kay, A. (1984) *Computer Software*, *Scientific American*, September 1984.
- Kay, A. (1993) *The Early History of Smalltalk*, *History of Programming Languages II*, ACM, <http://gagne.homedns.org/~tgagne/contrib/EarlyHistoryST.html>
- Kay, A., Ingalls, D., Ohshima, Y., Piumarta, I. and Raab, A. (2006) *Steps Toward The Reinvention of Programming a Compact and Practical Model of Personal Computing as a Self-Exploratorium*, *VPRI Research Note RN-2006-002*, <http://www.viewpointsresearch.org/pdf/NSFproposal.pdf>
- Kindborg, M. (2001) *Representing ToonTalk Programs as Comic Strips*, *Playground International Seminar*, Porto, Portugal, <http://www.ida.liu.se/~mikki/comics/KindborgToonTalkPorto.doc>
- Morgado, L., Cruz, M.G., and Kahn, K. (2003) *Taking Programming into Kindergartens*, *EuroLogo Proceedings*, Portugal. August 2003, <http://home.utad.pt/~leonelm/papers/eurologo2003/EuroLogo2003.htm>
- Ondrejka, C. and Cook, J. (2005) *Brace for Impact: How User Creation Changes Everything*, *Games, Learning and Society Conference*, Madison, Wisconsin.
- Repenning, A., Ioannidou, A., and Zola, J. (2000) *AgentSheets: End-User Programmable Simulations*, *Journal of Artificial Societies and Social Simulation*, 3(3), <http://jasss.soc.surrey.ac.uk/3/3/forum/1.html>
- Repenning, A. and Ioannidou, A. (2006) *AgentCubes: Raising the Ceiling of End-User Development in Education through Incremental 3D*, *IEEE Symposium on Visual Languages and Human-Centric Computing*, <http://www.cs.colorado.edu/~rale/papers/PDF/vl2006RaisingCeiling.pdf>
- Papert, S. (1977) *MIT Logo Project meeting notes*.
- Papert, S. (1980) *Mindstorms*, Basic Books, New York.
- Saraswat, V. A., Kahn, K., and Levy, J. (1990) *Janus: a step towards distributed constraint programming*, *North American Conference on Logic Programming*, MIT Press, Cambridge.
- Saraswat, V. (1993) *Concurrent constraint programming languages*, *Doctoral Dissertation Award and Logic Programming Series*, The MIT Press.

Smith, D.C., Cypher, A., and Tesler, L. (2000) *Novice programming comes of age*, Communications of the ACM, Volume 43, Number 3, Pages 75-81.

Smith, D., Kay, A. Raab, A., and Reed, D. (2003) *Croquet - A Collaboration System Architecture*, First Conference on Creating, Connecting and Collaborating through Computing.

The BehaviourComposer: A way for students to build computer programs without first learning to program

Ken Kahn, kenneth.kahn@oucs.ox.ac.uk
Learning Technologies Group, Oxford University

Abstract

The LOGO tradition is to help students to learn to program from the bottom up. They learn the basic building blocks of the programming language before building the programs they care about. This has many advantages but is sometimes infeasible due to constraints of time or lack of expert support. Also some students lose patience learning to programming before they have progressed far enough to see the benefits.

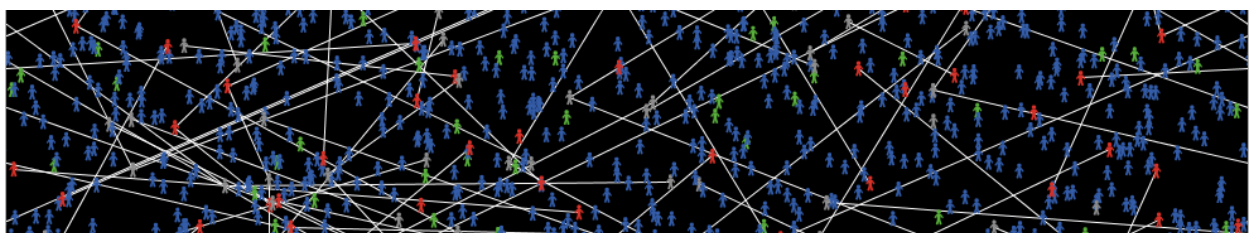
An alternative is to build tools and code libraries that enable students to very quickly build programs they care about by composing and customising pre-existing code fragments. At Oxford University we have built a freely available tool to support this “middle-out” style of programming called the *BehaviourComposer*. The prototype works together with NetLogo and contains an extensive library of code fragments we call *micro-behaviours*. It is oriented towards building scientific models but the approach is more general. Note that the details of low-level programming are not hidden and those students with the interest and time can go deep after starting in the “middle”.

In this workshop, you will be given the chance to experience first hand how one can build a model of an artificial society in a short time. You don't need to have any previous experience with NetLogo (or any programming language!). The *BehaviourComposer* incorporates a web browser component that provides familiar, and yet powerful, ways to browse, customise, and compose small bits of code.

The *BehaviourComposer* is free and open source but currently only runs in *Microsoft Windows*. The *BehaviourComposer* was built as part of the Constructing2Learn Project described at <http://dfl.cetis.ac.uk/wiki/index.php/Constructing2Learn>.

Keywords

BehaviourComposer; NetLogo; computer simulation, modelling, StarLogo



Editorial

When we started working on the 11th EuroLogo conference, scheduled for August 2007 in Bratislava, a few regular participants pointed out to me that although Logo would celebrate its 41st birthday, it would be exactly 40 years since its first official publication as a tool for learning. Good point! That was a challenge we couldn't resist – that is the reason why the main title of the conference says *40 Years of Influence On Education*. We decided to constitute the 11th conference as an opportunity to analyze the previous 40 years and reflect upon the future development and strategies. We recognize those years as especially rich of powerful ideas about modern education. Evidently, we have succeeded in understanding much about the potential of digital technologies for learning processes, about how to integrate them into contemporary education. There is hardly any country in this world where constructionist addicts have not managed to make at least a single small step in the wishful direction in real school settings. And yet we all hoped for more, we hoped for much bigger change to happen. We thought that by the beginning of the 21st century much more attention would be paid to the development of children's creativity, their computational competencies in official schooling systems. We hoped that Logo – either as a language by itself or in any manifestation as a philosophy of new education – would gain much bigger popularity and wider presence. In his keynote speech Wallace Feurzeig is saying:

Throughout its history, the Logo movement has played a significant role in fostering these goals, and it continues to do so, both within schools and outside. Logo may have gone underground in U.S. schools. But the influence of the ideas and philosophy underlying Logo remains powerful and pervasive, in the world and even in my country.

European and world community of Logo people obviously has a lot to discuss also in 2007. This is proved by high interest in coming to the conference, high number of submitted papers and the presence of all generations of Logo devotees in Bratislava this summer – from the inventors and fathers to young doctoral students and teachers from all levels and types of schools. For us, for our Department of Informatics Education and our so called Bratislava Logo group it is a great honour and privilege. If I look **back 25** (I mean back through 25 previous years of our active involvement in developing and implementing educational software platforms) or if I scan back of the fundamental books on my shelves and inspect the list of participants of the conference, I am thrilled to bits and bytes: The authors of my BOOKS – my teachers are here in Bratislava! Unfortunately, this isn't true about Seymour Papert – for well-known reasons – although in December 2006 in Hanoi we started working on his visit to Bratislava EuroLogo 2007.



Seymour Papert after his keynote presentation at the Study conference of the 17th ICMI Study at the Hanoi Institute of Technology, Vietnam, in December 2006. Seymour is talking with Celia Hoyles and Richard Noss, London Knowledge Lab, University of London. If the accident had not happened the very next day, he would – as he intended – be with us in Bratislava during this conference. We are all sending him warm greetings and our best wishes!

Fortunately, I can see here Wallace Feurzeig, representing the Logo founders and fathers (and as a virtual panel member also his co-writer Paul Goldenberg who kindly helped us to prepare the discussion titled *Learning vs. Creativity: a False Dichotomy*). I can also see here other key

Logo authors – Brian Harvey, Celia Hoyles, Richard Noss, James Clayson, Eric Klopfer and more!

Another outstanding Logo author and key personality in the field is Jenny Sendova. It is a great honour for us that Jenny and her colleagues are so actively involved in this year EuroLogo conference programme. I intentionally mention Jenny because she has always played an exceptionally important role for our Logo developing group and for the presence of Logo in Central and Eastern Europe in general. We were perfectly legitimate and honest with Andrej Blaho when we wrote in the preface to our *Learning by Developing* book:

We first met Logo in the early 80s thanks to two remarkable Bulgarian textbooks by Nikolov and Sendova, then through a number of MIT technical reports and later (when they managed to get to our part of the world) from the excellent Logo books of Abelson and Berentes. We were hooked! Then came the amazing Turtle Geometry of Abelson and diSessa!

With great pleasure I notice among the participants and authors of the conference our partners from several previous research projects from Greece, United Kingdom, Bulgaria, Hungary, Poland, Portugal, Austria, Lithuania and Brazil. With the equal pleasure I notice among the authors also many young people – newcomers to Logo community – young researchers and teachers.

I want to express my hearty thanks to our very strong group of the keynote speakers who have kindly accepted our invitation and play such an important role in the conference programme! Here they are:



Wallace Feurzeig



Celia Hoyles



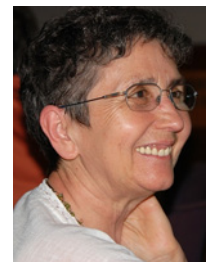
Eric Klopfer



Chronis Kynigos



Richard Noss



Jenny Sendova

I also warmly thank the authors of 11 plenary presentations and 48 selected papers for their contributions! My appreciation belongs also to our Comenius University for the continuous support of our work and namely for the support in organizing the conference. At last but not at least I want to express my deep respect to exceptionally action group of my colleagues and friends who made this conference happen. Good luck to you, 11th EuroLogo!

Ivan Kalas

The Effect of Learning Programming with Autonomous Robots for Elementary School Students

Shuji Kurebayashi, *eskureb@ipc.shizuoka.ac.jp*
Department of Education, Shizuoka University

Susumu Kanemune, *kanemune@acm.org*
Computer Center, Hitotsubashi University

Toshiyuki Kamada, *tkamada@aecc.aichi-edu.ac.jp*
Department of Technology Education, Aichi University of Education

Yasushi Kuno, *kuno@gssm.otsuka.tsukuba.ac.jp*
Graduate School of System Management, University of Tsukuba

Abstract

It is possible to evaluate the effect of learning programming and controlling objectively by examinations. However, investigation the amount of knowledge on programming and controlling seems not to be enough because it only shows how much of knowledge is established to learners. Thus, this approach does not unveil what kinds of abilities are gained in students through learning computer programming.

We first try to find out the method of evaluation other than the objective test in order to answer the above question. As a result, we finally set to study the effect of learning programming from the viewpoint of "Technology Literacy" instead of conventional issue of creative thinking or logical thinking.

Recently, numbers of incidents and accidents relating to computer technologies are occurred and we mundanely receive news reports on that. People those who have technology literacy can understand the background of the report, but people those who are lack in technology literacy do not understand it and they cannot explain why the incident or accident is occurred.

Then, we set up a hypothesis that there is a difference of understanding the outline of incidents or accidents between students those who have learned the computer programming and students those who have not learned it. We think if there were a difference, we would be able to conclude that the learning computer programming has a good effect for students on developing technology literacy itself.

To bear out our hypothesis, we made a survey on the difference of the knowledge contributing to understand the reason of an accident between two groups of elementary school students with the same age; one is a group those who are learned computer programming by controlling robotic cars and the other is a group those who are not learned computer programming at all. The intended accident was the fatal accident by an elevator manufactured by Schindler Elevator Company occurred in Tokyo, 2006. As a result, we could see the difference between them.

In this paper, the programming language "Dolittle" which is a teaching material for programming lessons and the target robotic car which is controlled by Dolittle are introduced. Then the content of programming lessons conducted in an elementary school and questionnaires to the students are explained. The survey result and the analysis of it are also reported.

Keywords

Learning Programming, Autonomous Robot, Educational Evaluation, Technology Literacy

Introduction

The educational value of learning programming is described in various practices (Saeki, 1986). However in Japan, the range of dealing with the programming is limited as written in the course of study by the government (Ministry of Education, Culture, Sports, Science and Technology) for junior and senior high schools; “The educator should not deeply involve the programming” (MEXT, 1998 and 1999). Thus there are a few secondary schools which take up the programming for their lessons. Similarly, in spite of the information education in elementary schools are prescribed in the “Period for Integrated Study”, the example of the practice of the programming is quite limited after the revision to the current course of study (Sato *et al.*, 2005). The reason of this circumstance is that the grand design of current information education in Japan is too much emphasized fostering the “Information Application Ability” using computers (Wada, 2006). However, it is impossible for students to understand the mechanism of software and hardware only by learning the usage of application software such as word processors, spread sheets and web browsers. Moreover, it is difficult to understand or have interest in accidents or incidents regarding to information technologies. Because if the person can clearly conceive the overview of the accident by hearing the terms like “error in the program” or “system down” in news reports which are included in the description of the cause of accidents is depend on the knowledge of not only the usage of computers but also the mechanism of computers.

Therefore, we have made a survey on the knowledge regarding to the understanding of the cause of the fatal accident by an elevator developed by Schindler Elevator Company occurred in Tokyo on June 3, 2006 (Japan Times, 2006; Wikipedia) by showing the report of a newspaper to two groups of elementary school students; one has a experience of learning programming and controlling, the other does not have them at all. As a result, we find the difference of students’ understanding of the accident between these two groups.

In this paper, firstly the programming language “Dolittle” (Kanemune, *et al.*, 2004 and 2005) which is a teaching material for lessons and the target robotic car which is controlled by Dolittle are introduced. Secondary, the content of programming lessons conducted in an elementary school is explained. Thirdly, the material distributed to students and the content of the questionnaire are presented. Lastly the result of our survey and the analysis of it are reported.

Lessons

We asked a teacher in Osu Elementary School in Fujieda city, Shizuoka prefecture to conduct lessons on programming and controlling robotic cars in the “Period for Integrated Study” subject. The teacher gave lessons in two classes of 6th grade (11 years old). The total number of the students is 64. The lessons are held from June to July in 2006. Each period is 45 minutes. The curriculum of the lessons is shown in Table 1.

Table 1. The curriculum of the lessons

	The theme of the lesson	Periods
1	How to start up “Dolittle”	2
2	Move the turtle in Dolittle	2
3	Change the looks of the turtle and drawing simple figure	2
4	Let’s move the turtle and figures with timer	2
5	Let’s move the robot	2
6	Let’s challenge the maze (1)	2
7	Let’s challenge the maze (2), Summary	2

Lessons are held in the computer room. Because of the number of installed PCs in the computer room is 20, pairs of students shared each PC. The specification of PCs is as follows:

- CPU: Celeron 400MHz
- Memory: 64MB
- Operating System: Windows 98

Graphics Drawing Program Using Dolittle

Dolittle is an object-oriented programming language designed for school education. It has simple language syntax and the programmer can use their local language such as Japanese, Korean and English for instructions and identifiers including variable names. Then Japanese students can program using Japanese keywords. In Japan, English classes are not held in elementary schools, then students in Japanese elementary school use Japanese in programming Dolittle.

Figure 1 shows the sample program of animated graphics in Dolittle. Students wrote this kind of program and enjoyed creating their own animated graphics.

```
Kameta=turtle ! create.  
[Kameta ! 100 forward 120 rightturn] ! 3 repeat.  
Triangle=Kameta ! makefigure (red) paint.  
Clock=timer ! create 1 period 10 duration.  
Clock ! [Triangle ! 36 rightturn] execute.
```

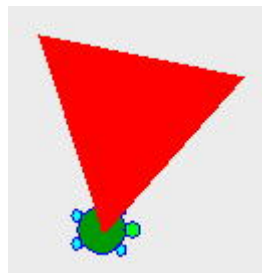


Figure 1. A sample program of Dolittle and the screen shot of this program

The first line of the program creates a turtle object named “Kameta”. The second line draws a triangle with turtle graphics. The third line changes drawn graphic into figure object named “Triangle” and paint it with red color. The fourth line creates timer object named “Clock” and set up some properties of it. The fifth (last) line realizes an animation by asking timer object for rotating “Triangle” object.

Controlling Program for Robotic Cars

The robotic car mounts biaxial robot controller board which we have developed (Kurebayashi *et al.*, 2006). The controller board has embedded microcontroller with our original firmware. Thanks to this firmware, users of this robot can write a program with up to 39 steps into the microcontroller. The program is transferred from PC to the controller board with an embedded infra-red lay device. Figure 2 shows the robotic car which is used in our experimental lessons.

Table 2 shows robot controlling instructions for Dolittle. Figure 3 shows an example controlling program written in Dolittle. In this program, a serialport object (which is used to communicate between the PC and the robot via serial port) named “Robo” is created first, then a method named “transfer” is defined to “Robo” object and several robot control instructions are written in it.

This method is invoked after the serial port is opened. With the written control instructions, the robot executes following motion:

- Start executing the program when the sensor switch is pressed
- Go forward until the front sensor is pressed by touching to the front wall
- Go backward and turn to the left
- Go forward until the front sensor is pressed by touching to the front wall
- Go backward and turn to the right

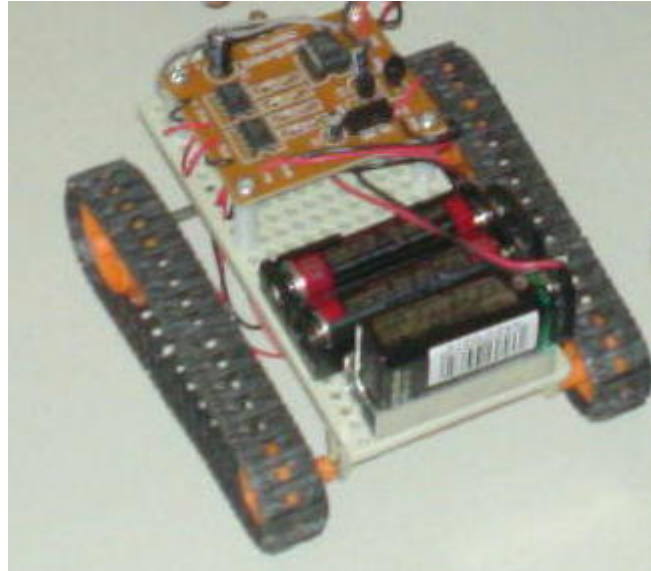


Figure 2. The robotic car used in our experimental lessons

Table 2. Robot control instructions

Instruction	Explanation	Example
opensesame	Open the serial port	"com1" opensesame
closesesame	Close the serial port	
leftforward	Turn the left wheel forward	10 leftforward
rightforward	Turn the right wheel forward	10 rightforward
forward	Turn both wheels forward	10 forward
leftback	Turn the left wheel backward	10 leftback
rightback	Turn the right wheel backward	10 rightback
back	Turn both wheels backward	10 back
startrobot	Start of the robot controlling	
endrobot	End of the robot controlling	
forwarduntilcollision	Go forward until the sensor switch is pressed	
switchstart	Execute instructions when the sensor switch is pressed	

*The unit of the argument of instructions which turn a wheel is 0.1 second.

```
Robo=serialport ! create.  
Robo:transfer=[!  
    startrobot  
    switchstart  
    forwarduntilcollision  
    10 forward 15 rightforward 15 leftback  
    forwarduntilcollision  
    10 back 15 leftforward 15 rightback  
    endrobot].  
Robo ! "com1" opensesame.  
Robo ! transfer.  
Robo ! closesesame.
```

Figure 3. A sample robot controlling program with Dolittle



Figure 4. Students controlling their robotic cars with their programs

We have built simple maze made with wooden panels and offer it to the lesson. Students input a program in Figure 3 and try to modify or arrange it in order to go through the maze. Figure 4 shows the scene of students controlling their robots.

Questionnaire

Document Distributed to Students

Before questionnaire, we have provided a document to students as background information for them. The content of the document is regarding to the fatal accident of an elevator developed by Schindler Elevator Company occurred in Tokyo on June 3, 2006. The reason why we adopt

this accident is that the fundamental cause is a flaw in controlling program of the elevator and this is related to computer programming and control. Furthermore, we hope the situation of the accident could be recognized even by elementary school students because elevators are familiar machines and everyone seems to have experiences of using them.

The document is basically based on the explanatory article from Asahi Shimbun newspaper Internet edition (Asahi Shimbun, 2006) and the explanatory pictures from Japan Elevator Association website (JEA, 2006). The document is composed of two pages. In the first page, we have placed articles which explain the outline of the accident. In the second page, we have placed an article explains the cause of the accident and supplementary pictures explaining the architecture of elevators.

Content of Questionnaire

We have made students to read the document before answering questions. The number of questions is six. The format of the answer to question from number 1 to 4 was five level scales (5: strongly agree, 4: agree, 3: not sure, 2: disagree, 1: strongly disagree) and number 5 and 6 was description. Question number 5 and 6 are for examining the extent of understanding the content of question number 3 and 4.

The actual questions to students are as follows:

- Question 1:** Did you know this accident of the elevator?
- Question 2:** Can you understand the architecture of elevators?
- Question 3:** Can you understand what is wrong by reading the sentence that the cause of the accident is “flaw in controlling program (errors in controlling program)”?
- Question 4:** Can you explain the cause of the accident to other people?
- Question 5:** What is controlled by the control board? Answer by looking at the picture in the document.
- Question 6:** Please write your opinion of what kind of matters do you want to be taken care by the people in the elevator company in order to prevent the accident like this anymore.

Procedure of Sending Out the Questionnaire

On sending out the questionnaire, we have explained the following notices to the teacher. The time to answer questions had set to 20 minutes. Because there are some difficult words and statements for elementary school students in the document, the teacher had to give supplemental description to the students. However, we asked for the teacher not to explain the architecture of elevators and the role of controlling programs.

- Notice 1:** Please read out the document.
- Notice 2:** It is allowed to teach how the questioned Kanji (Japanese letter) to read.
- Notice 3:** Never answer to the question regarding to the architecture of elevators and the role of controlling programs.

Control Group

As a control group, we have chosen 6th grade (11 years old) students in Fujieda Chuo Elementary School in Fujieda city, Shizuoka prefecture. The number of students is 29. The students' experiences of operating computers are limited to using application software such as a web browser, word processor and paint software. There are no students who have an experience of a programming or controlling robots.

Result

The summary of two groups of students; (1) 64 elementary school students (6th grade, 11 years old) who have experienced computer programming and controlling robotic cars with computer in October, 2006; (2) 29 students with the same age, not experienced computer programming or controlling is shown in from Table 3 to Table 6. In these tables, answer number 5 and 4 are positive answers, and answer number 1 and 2 are negative answers.

Table 3. Question 1: Did you know this accident of the elevator? (%)

	5	4	3	2	1
Experienced	29.7	50.0	6.3	6.3	7.7
Not Experienced	24.1	44.8	17.2	3.4	10.5

Table 4. Question 2: Can you understand the architecture of the elevator? (%)

	5	4	3	2	1
Experienced	4.7	23.5	40.7	28.2	2.9
Not Experienced	3.4	6.9	58.6	10.3	20.8

In the summary of question 1 (see Table 3), there is no clear difference between the students who have experienced control programming (hereinafter referred to as “experienced students”) and the students who have not experienced it (hereinafter referred to as “not experienced students”). These are equivalent in the recognition of the news report. In the summary of question 2 (see Table 4), the ratio of experienced students who have positive answers (the sum of the answer number 5 and 4) is 28.2%. On the contrary, the ratio of not experienced students is 10.3%. From these results, it seems to affect the extent of understanding the architecture of the elevator for students that the presence or absence of experience of the control programming.

Table 5. Question 3: Do you understand what is wrong? (%)

	5	4	3	2	1
Experienced	11.0	39.1	32.8	14.1	3.0
Not Experienced	0.0	0.0	44.8	20.7	34.5

Table 6. Question 4: Can you explain the cause of the accident to other people? (%)

	5	4	3	2	1
Experienced	4.7	25.0	32.9	29.7	7.7
Not Experienced	0.0	3.4	27.6	37.9	31.1

In the summary of question 3 (see Table 5), the ratio of experienced students who have positive answers is 50.1% and the ratio of not experienced students is 0%. In the summary of question 4 (see Table 6), the ratio of experienced students who have positive answers is 29.7% and the ratio of not experienced students is 3.4%. From these results, it seems to be inferable that the experience of control programming affects the understanding of the cause of the accident.

Figure 5 and 6 are graphs of the ratio by classifying contents of answers to question 5 and 6 respectively. Answers to the question 5 are classified into 4 categories; “moving machinery such as winch and doors”, “counter weight”, “other elements (including unclear statement)” and “not sure”. From this summary, the ratio of experienced students answering “moving machinery” was

37.6% and the ratio of not experienced students was 13.6%. Furthermore, the ratio of experienced students answering “not sure” was 35.9% and the ratio of not experienced students was 62.1%. Besides, the ratio answering “counter weight” which is not related to controlling is small in experienced students meanwhile it is more in not experienced students. As for question 6, answers to it are classified into 3 categories; “point out the inspection point (such as a control board or programs)”, “not point out” as represented by the description of “People of the elevator company should inspect steadily”, and “unclear”. From this summary, we could see two large differences between experienced students and not experienced students; the ratio of the answer “unclear” is 0% in experienced students and 27.6% in not experienced students. The ratio of pointing out the concrete inspection point is 20.6% in experienced students and 0% in not experienced students. From above results, it is clear that there is a difference in the capacity of understanding the cause of the elevator accident. Consequently, we can conclude that the experience of controlling program affects the ability to figure out the concrete cause of the accident and the perspective on the way of inspection.

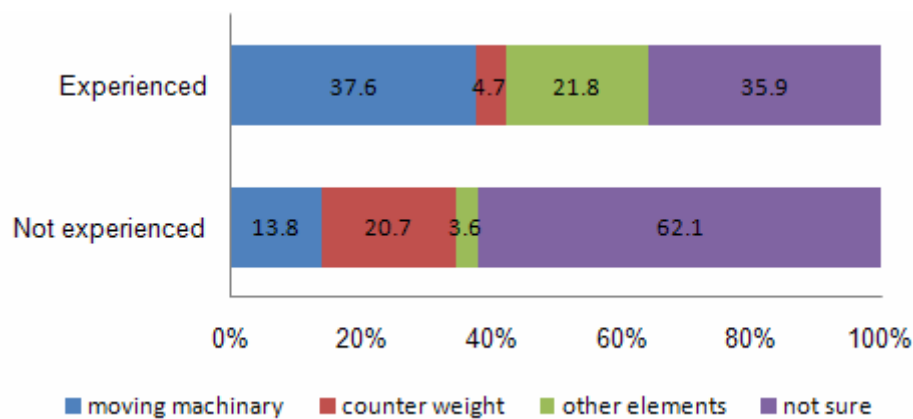


Figure 5. Summary of the answer to the question 5: What is controlled by the control board?

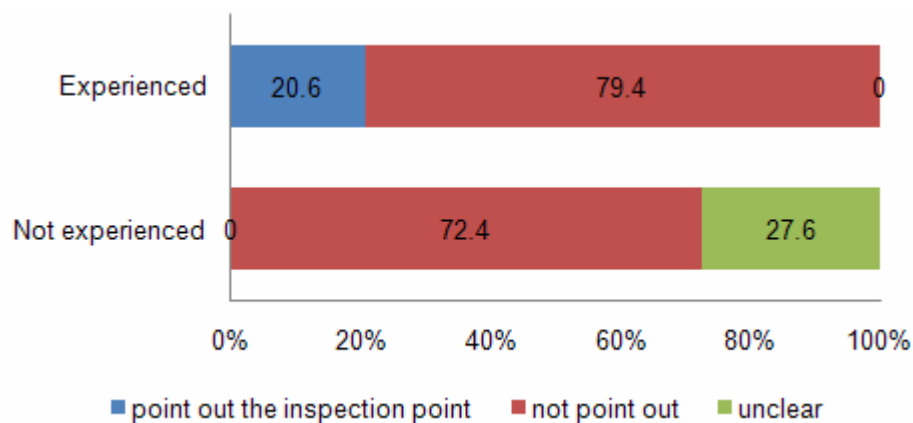


Figure 6. Summary of the answer to the question 6: What should be done to prevent accidents like this?

Discussion

The ratio of the students with positive answers to question 3 and 4 is more in experienced students than not experienced students. The reason of this seems to be that the experienced students can easily imagine which components are controlled by the computer program. Indeed, 37.6% of experienced students answer moving machineries to question 5 as a target of control board. Experienced students should have a concrete image of controlling mechanism through

the robot programming such as controlling motor rotations and selecting proper behaviour according to sensor inputs. Then they should have an ability to imagine the motion of each component of elevators. Furthermore, the ratio of not experienced students who can point out concrete inspection point is zero. This would strongly support above thought.

Concluding Remarks

We have investigated the difference of the extent of understanding the elevator accident between the existence and nonexistence of the experience of learning controlling programming. The 6th grade (11 years old) students have shown the clear difference.

Sometimes teachers try to make their students to engage in research work in order to induce interest in some accidents or incidents in the world. However, the success or failure of the class is not only willingness of students but also an education that can understand the cause or the background of the incident from documents they have obtained. For this reason, educators of information education should approach from the view point of “Technology Literacy” which is an ability of “hearing from a specialist about technology, and understanding, using, assessing and managing it”. Then, learning computer programming have to contribute to make people living in highly computerized world understand the substantial risk of accidents or incidents.

We would like to thank the teachers and the students of Osu Elementary School and Fujieda Chuo Elementary School for the contribution to the conduction of lessons and the answer to the questionnaires.

References

- Saeki, Y. (1986) *Computer and Education*. Iwanami Shoten, Tokyo. (Japanese)
- Ministry of Education, Culture, Sports, Science and Technology (1998) *The Course of Study for Junior High School*. (Japanese)
- Ministry of Education, Culture, Sports, Science and Technology (1999) *The Course of Study for High School*. (Japanese)
- Sato, K., Kurebayashi, S. and Kanemune, S. (2005) *An Proposal of Using Programming in Elementary School*, IPSJ SIGCE, CE(78), pp. 57–63. (Japanese)
- Wada B. T. (2006) *Information Education for Primary and Secondary Education in Korea and Subject “Information” in Japan*, Symposium on Subject “Information” for High School 2006, IPSJ, pp. 50–59. (Japanese)
- Japan Times (2006) *Elevator Safety Must be Priority*, June 17, 2006.
<http://www.japantimes.co.jp/weekly/ed/ed20060617a1.htm>
- Wikipedia *Minato Ward 2006 Elevator Accident*.
http://en.wikipedia.org/wiki/Minato_Ward_2006_elevator_accident
- Kanemune, S., Nakatani, T., Mitarai, R., Fukui, S. and Kuno, Y. (2004) *Dolittle – Experiences in Teaching Programming at K12 Schools*, In the Proceedings of Second International Conference on Creating, Connecting, and Collaborating through Computing (C5), IEEE, pp. 177–184.
- Kanemune, S. and Kuno, Y. (2005) *Dolittle: An Object-Oriented Language for K12 Education*, EuroLogo 2005, Warszawa, Poland, pp. 144–153.
- Kurebayashi, S., Kamada, T. and Kanemune, S. (2006) *Learning Computer Program with Autonomous Robots*, Lecture Notes in Computer Science, Vol. 4226, pp. 138–149.
- Asahi Shimbun (2006) *Elevator Killed Male 2nd Grade High School Student at Shiba, Tokyo*.
<http://www.asahi.com/special/060718/TKY200606030420.html> (Japanese)
- Japan Elevator Association (2006) *About Elevator*.
http://www.n-elekyo.co.jp/square/elevator_01.html (Japanese)

Using Microworlds-Pro to support Effectively the educational Process

Sofia Kasola, *skasola@upatras.gr*

Researcher, Post-graduate student, University of Patras

Chris Panagiotakopoulos, *cpanag@upatras.gr*

Assistant Professor of Educational Technology, University of Patras, Dept. of Education

Panagiotis Pintelas, *pintelas@math.upatras.gr*

Professor of Computer Science, University of Patras, Dept. of Mathematics

Abstract

Microworlds are typical examples of digital open environments of learning, which give the pupils the possibility of exploration and active training. They are composed of a number of objects and relations as well as a number of operations that affect the objects, modifying their relations and creating new objects. They are open systems which the pupils can explore with minimal guidance, combining the commands of some language. Microworlds are usually interactive training environments

which allow the users to manage them at a very high level. A recently released software for the construction of microworlds is Microworlds Pro (MWP), which is based on the Logo language and offers several advantages. With the embedded capability of optical representation at the implementation of a program, the pupils can learn programming easily as well as to comprehend a program.

In this study we present the attitudes of candidate teachers with regard to the environment of MWP, and the effects of an application, developed with MWP, about understanding the phenomenon of alternation of day – night by young pupils. In order to explore the opinions of the selected sample, the participants were introduced to the MWP environment as well as to short applications constructed for 3 hours laboratory training. After this, candidate teachers answered a short questionnaire with open and closed questions. The findings of this research are that the environment is very easy to handle, enriches the educational process, promotes the active attendance and strengthens the interest and self-activity of the pupils. Furthermore, in order to explore the physical phenomenon of the alternation of day – night by young pupils, a MWP application was used. This application (SUN-EARTH-MOON: SEM), was constructed using the MWP environment and it contains simulations of planets and their movements. From the results of a pre-test, we realised that the pupils had confusions and misconceptions about this phenomenon even though they had taken school lessons. Using the SEM application, the pupils worked by themselves, with active attendance, they had the chance to explore the phenomenon and understand it working step by step. The post-test showed that they had approached the scientific model. From the final conclusions it is obvious that the MWP environment can improve and enhance the learning procedure, using appropriate applications.

Keywords

Microworlds, Logo, ICT, Exploratory Learning, Educational process, active attendance

1 Introduction

Recent theories concerning the way of teaching and learning include those that focus in the interaction entanglement of the pupils with the training materials, with the schoolmates and with their schoolteachers, so that the energetic learning is promoted (Hake, 1998; Novak, 1999). As it has emerged through the specific research, the use of such proceeding influences positively the comprehension of various significances (Hake, 1998). Often, the constructors of educational software try to adopt such theories. The present tendencies are directed in the manufacture of open environments of learning which either support or extend the theory of Papert (1980), where the knowledge is acquired through interaction with objects of real world or with digital objects that have all the attributes of the real ones.

The microworlds are characteristic formal examples of digital open environments of learning, which give to the pupils the possibility of exploration (diSessa et al., 1995). According to Ventrella (1993): “a microworld is a tool with which, one can learn about some aspect of the world and also about him or herself. An effective microworld encourages exploration, manipulation, and some degree of designing. Microworlds can model scenes from the real world in simple form - like animals, people, things - or they can even model collections of ideas, which can be manipulated and explored”. As Rieber (2005) mentions, the common points of the various theories regarding the microworlds in word bibliography, are that:

- (a) They offer a way for more people, especially young, to understand and explore concepts and principles underlying complex systems.
- (b) They focus primarily on qualitative understanding based on building and using concrete models, and
- (c) There is a deliberate attempt to reduce the distinction between learning science and doing science.

According to diSessa (2000), a microworld that is created in an electronic environment of learning assembles and incorporates all the important significances, which the pupils can explore. In order to be effective a microworld should have simple operations so that the pupils focus in important training activities, which help them in the comprehension of fundamental values.

Microworlds are composed of a number of objects and relations as well as a number of operations that affect the objects, modifying their relations and creating new objects. They are open systems which the student can explore with minimal guidance, combining the commands of some language (Papert, 1980; Jonassen et al., 1998). Microworlds are perhaps the absolute example of active training environments, as the users have the possibility to manage them in a very high level (Jonassen et al., 1998).

As Papert mentions (1980), using Logo language, which has been the subject of extensive research, we can create such microworlds. In point of fact, the significance of microworlds began with the Logo programming language, which was developed in the MIT in the decade 1960 and became exceptionally popular with the charge of personal computers in the decade of 1980 (Papert, 1980; Rieber, 2006).

MicroWorlds, is one of a family of computer software applications generally known as multimedia. Designed as a true successor to the Logo environments of the 1980s, it still bears the stamp of Seymour Papert whose original idea for Logo was to give the power of the computer to children. Designed as a child-friendly piece of software, it is visually rich, has a good text editing facility, can manipulate text as well as graphics in a graphical way, and has the full Logo implementation as its scripting language. It is deliberately designed to fit into a constructivist classroom environment, or, using Papert's construct of constructivism, “constructionist.” (Vincent, 2001).

A recently released software for the construction of microworlds is MicroWorlds Pro (MWP), which is based on Logo language and offers several advantages (LCSI, 2006, OWL, 2006). With the possibility of optical representation at the implementation of program, the pupils can learn programming easily as well as to comprehend a program (Logo Foundation, 2006).

Summing up, we could say that MWP is an open training and programming environment which:

- Offers capabilities for collaborative learning without particular difficulties for the user.
- Provides the user with facilities such as text processing, sound, video and animation.
- Includes the Logo programming language as well as other facilities such as buttons, links, etc.
- Provides the user with the capability of programming the turtle easily without requiring any particular knowledge of mathematics or algebra, as is the case with other programming software.
- Gives the user the capability to handle the turtle using elementary commands or design and develop sufficiently complex programming applications

On the other hand, it is known that the values, the attitudes, the convictions and the perceptions of teachers: (a) constitute the base in which they will be supported in order to seek their improvement and their professional development, and to maintain the communication with the scientific community (Matsagouras, 1998), and (b) (2) they influence their instructive - training practices (Reynolds, 1992).

In this paper, initially, are present the attitudes of a sample of fourth year pupils of the Pedagogic Department, at the University of Patras, Greece, for the MWP environment. These pupils will very shortly work as primary school teachers. Considering this, their opinions have particular importance. The data were collected after a 3 hour laboratory presentation of MWP, during which the possibilities of the program, and presentation of applications based on Micro words Pro were analyzed. Following is presented a study with the use of an application in the MWP environment, which was built in order pupils of Primary School to comprehend the phenomenon of alternation of day and night, which, as Vosniadou mentions (1994 and 1998) is one of the phenomena that pupils often have misapprehensions. The results from this study are also interesting.

In following section we present the attitudes of candidate teachers for the environment MWP, the experimental procedure for the phenomenon of alternation of day-night, the findings and the general conclusion about the use of MWP.

2 Attitudes of candidate teachers for the environment MWP

The sample of the research, which was carried out in November 2006, consists of 63 fourth year pupils of Department of Primary Education of University of Patras. The ages of the sample range from 19 to 21 years. 15 from them they were men (23.8%) and 48 women (76.2%).

After the presentation of the MWP environment, in laboratory, the pupils working in teams familiarized themselves with the geometry of the turtle and the additional multimedia capabilities. The presentation was focused in two applications with the use of MWP which we are reporting in the next paragraphs of this article. At the end of the 3 hours presentation, the pupils answered and recorded their opinions in a short questionnaire with open and closed questions.

In the question for the self-assessment of their knowledge in combination with their experience in handling of computer, in the scale Likert (1= non-existent, 2=few, 3=so-so, 4=enough], 5=perfect) the results showed mean 3.52 with standard deviation 0.76. The frequencies and the answers are in Table 1:

Non-existent	Few	So-so	Enough	Perfect
0 (0.0%)	5 (7.9%)	25 (39.7%)	28 (44.4%)	5 (7.9%)

Table 1: Self-assessment of the sample about the knowledge using computer

A small part of 11 pupils (17.5%) out of the total sample of 63 had some prior experience with MWP while the remaining 52 (82.5%) had no previous experience or knowledge of this environment.

In relative question about if the sample considers that the MWP environment promotes the exploratory learning the answers were 100% affirmative.

In the question for the degree of easiness or difficulty in handling the environment, with answers in the scale Likert (1= very difficult, 2= difficult, 3=So-so, 4= easy, 5= very easy), the results showed mean 4.08 with standard deviation 0.48. The frequencies and the answers are depicted in Table 2:

Very difficult	Difficult	So-so	Easy	Very easy
0 (0.0%)	0 (0.0%)	5 (7.9%)	48 (76.2%)	10 (15.9%)

Table 2: The opinions of the sample about the degree of facility or difficulty of MWP

The next question concerned whether the sample considered that MWP could be applicable in the daily educational process. The justification presented by those pupils who gave negative answers was that to use MWP in the daily practice requires total reformation of the Primary schools' analytic program. It should be pointed here, that in the Greek Primary Education system for the moment, there is no planned use of computer assisted learning.

We asked the sample to record positive points in the use of MWP for the schoolteacher. The aggregated of answers appear in Chart 1. Next, we asked the sample to record positive points in the use of MWP for the student. The aggregated answers appear in Chart 2.

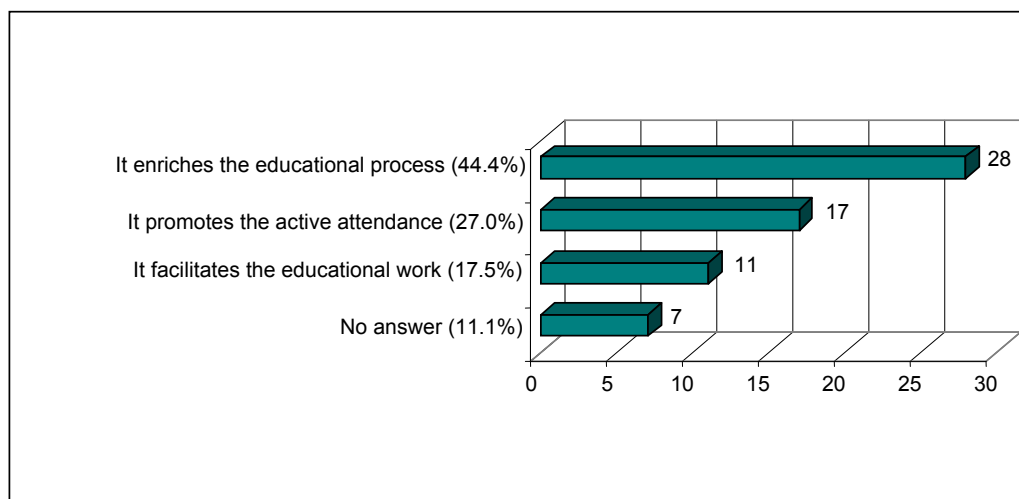


Chart 1: Positive points using MWP for the teacher, according to the sample

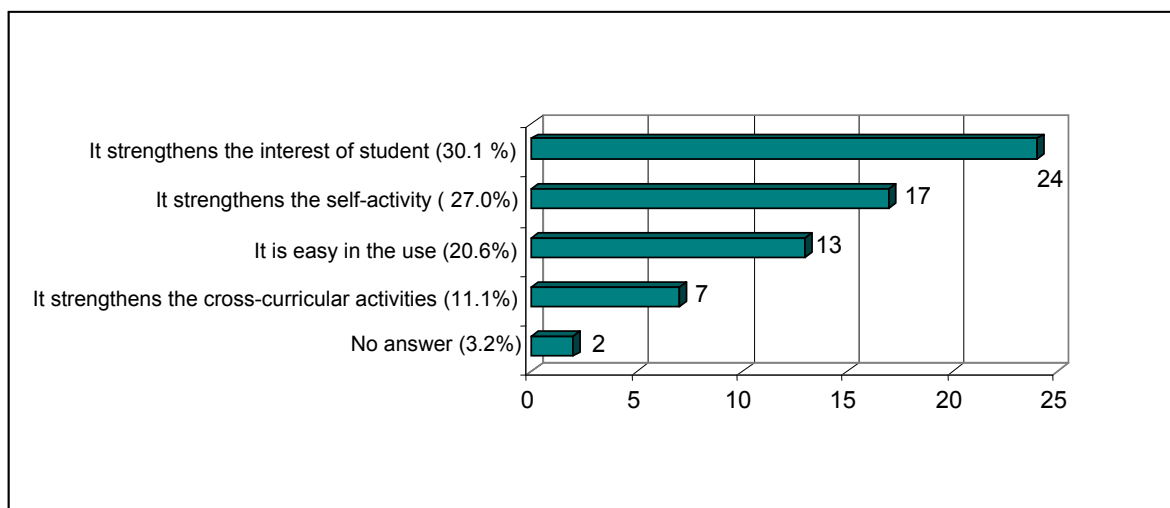


Chart 2: Positive points using MWP for the student, according to the sample

In our attempt to make the sample identify the negative points, from the teacher's as well as from the pupil's point of view, we had no such identification.

From the previous it results that the future teachers of the sample, have a positive attitude about the use of MWP software in the educational process. They expressed the opinion that it constitutes a tool which can strengthen the interest of the pupils, strengthen the self-activity and the active attendance, enrich and facilitate the educational process.

These views in combination with the views of the pupils which find MWP environment attractive can lead to conditions proper for learning by doing and learning by discovery.

3 The phenomenon of alternation of day - night

The main aim of this part of our research is to explore the effect of a simulation in the understanding of a physical phenomenon. Specifically, how 10/11 and 11/12 year old pupils understand the physical phenomenon of the rotation of the earth and the alternation between day and night. In order to perform this specific research, we constructed the application "SUN-EARTH-MOON" (SEM) using the software MWP v. 1.1, from Rainbow Computer Company, which is translated in the Greek language.

For this study we contacted a typical primary school in a suburban region near the city of Patras. In our research 46 pupils participated. None of the sample pupils had sight, hearing problems or learning difficulties. The average age was 10.51 years with a st.d. 0.31 of the 5th grade and 11.48 years with a st.d.=0.32 of the 6th grade. In the sample there were 26 boys (15 from 5th grade and 11 from 6th grade) and 20 girls (6 from 5th grade and 14 from 6th grade). The research took place during January 2005 in the PC lab of the school with "MWP 1.1" and the application SEM.

The research was performed by employing teams of eight pupils, where each student worked alone under the supervision of a tutor. Responsibility of the tutor was to take care to preserve the verbal framework in order to avoid any effects of what was said to the pupils. Pupils of the 6th grade had already been taught about the rotation of the planets around the sun, the structure of the solar system, the rotation of the earth around its own axis and the sun, and the existence of the moon as a satellite of the earth during their geography lessons.

3.1 The experimental procedure

The application SEM is designed and implemented in a way that approaches the study of the phenomenon with specific simulation on the screen, with the possibility of direct input by the pupils to address the objectives. The experimental procedure is divided into three parts. In the

first part, a questionnaire was given to the pupils with four questions marked as K1, K2, K3 and K4:

(K1) "Why do we have day?"

(K2) "Why do we have night?"

(K3) Why does the sun change its position in the sky?"

(K4) Why does the moon change its position in the sky?"

After completing the questionnaire, the pupils started working with the SEM application. In Figure 1a we can see the first phase of the application. When the pupils "clicked" on the earth, it started to rotate. The pupils then had to complete an appropriate questionnaire. The first question (D1) put to the pupils was: "*What do you notice?*" The correct answer was "*The earth rotates*". After that, the tutor asked them to "click" onto the arrow at the bottom right corner of the monitor. Then, the pupils had Figure 1a in front of them.

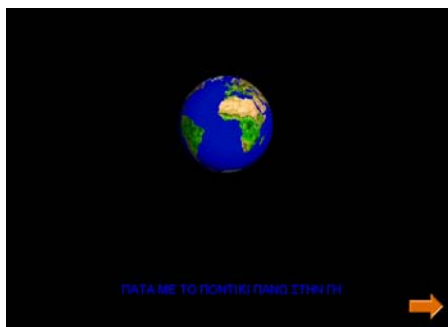


Figure 1a

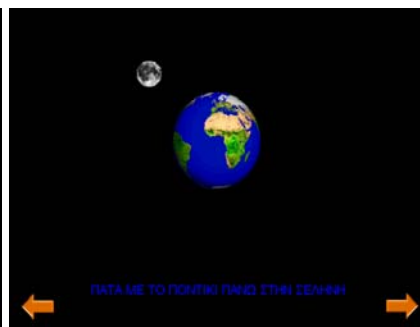


Figure 1b

Figure 1a, 1b: The first (1a) and the second phase (1b) of the application SEM

In the Figure 1b we can see the second phase of the application. When the pupils "clicked" onto the moon, it started to rotate around the earth. The pupils then were asked to answer the question (D2) which was: "*What do you see?*" The correct answer was "*The moon is rotating around the earth*". They had to write their answer on the questionnaire. Next, the tutor asked to them to "click" onto the arrow at the bottom right corner of the monitor again.

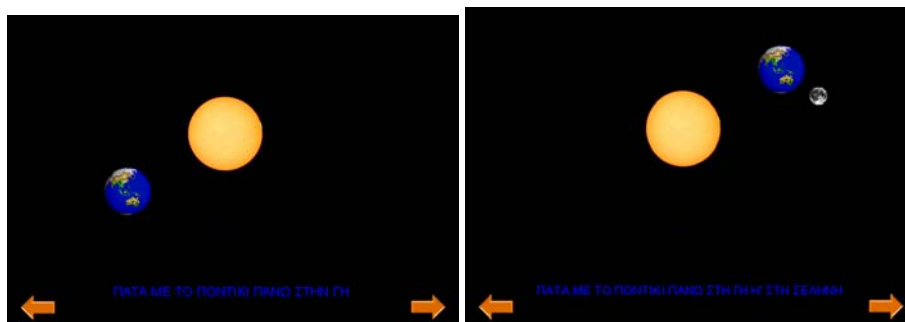


Figure 2: The third (left) and the fourth phase (right) of the application SEM

In Figure 2 - left, we can see the third phase of the application. When the pupils "clicked" onto the earth, it started to rotate around the sun. The question put (D3) to the pupils this time was: "*What do you see?*" The correct answer was "*the earth is rotating around the sun*". The pupils provided their answer on the questionnaire and then the tutor asked them to "click" on the arrow at the bottom right corner of the monitor again.

In Figure 2-right, we can see the fourth phase of the application. When the pupils “clicked” onto the earth or onto the moon, the earth started to rotate around the sun and the moon rotated around the earth. The question (D4) to the student was: “*What do you see?*”. The correct answer was “*The earth is rotating around the sun and at the same time the moon rotates around the earth*”. The pupils provided their answer on the questionnaire. The tutor asked them once again to “click” onto the arrow at bottom right corner of the monitor.



Figure 3: The fifth phase of the application SEM

In Figure 3 we can see the fifth phase of the application. When the pupils “clicked” onto the grey button down to the left corner (>>>), the earth starts to rotate around the sun with the moon rotating around the earth at the same time. After a while, the tutor asked the pupils to “click” again on the grey button and then the earth and the moon stopped. The next question (D5_1) to the student was: “*Can you drag the house and drop it onto the earth at the side where is daytime?*”. The procedure with the grey button was repeated and when the earth and the moon stopped again, the question (D5_2) was put forward: “*Can you drag the house and drop it onto the earth on the side where it is night?*”.

The questions D5_1 and D5_2 were put to all the pupils in turn and under the guidance of the tutor. The question Q5 was considered correct only if the pupils had answered correctly both question D5_1 and D5_2. At this point we asked the pupils to answer the same four questions, just as before the procedure with the SEM application.

3.2 Findings and discussion

The statistical analysis has been performed using level of significance 0.05. The obtained results came from a sample that was, in general, from families with parents of a relatively low social and financial background. From the statistical analysis there has not been any significant difference observed in the responses of pupils originating from different social and financial backgrounds, or due to the professions of their parents. The same is also true concerning the gender of pupils.

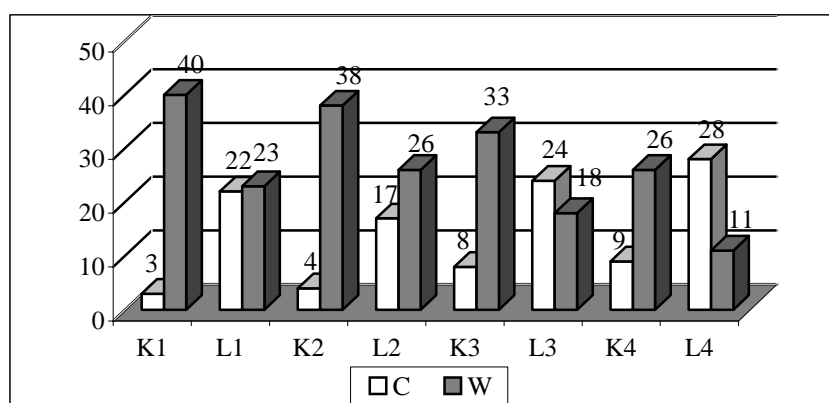
Of the 46 pupils in the sample, 4 (8.7%) claimed that they use the computer for their homework (educational software, encyclopaedias, Internet). Of these 4, 2 originated from the 5th grade and 2 from the 6th grade. Thus we may consider that, the sample had no experience using computers and educational software.

As previously mentioned, the questions K1, K2, K3, K4 were asked of the pupils just before the use of SEM Microworld application. The exact same questions were asked of the pupils just after the use of the SEM and they are noted in table 3 as L1 to L4, respectively. The overall results are shown in table 3 per question and per grade, before and after the use of the software.

		Questions K1 to K4 and L1 to L4							
		K1	L1	K2	L2	K3	L3	K4	L4
5 th grade	Correct	0	11	0	7	1	9	1	12
		0.0%	52.4%	0.0%	33.3%	4.8%	42.9%	4.8%	57.1%
	Wrong	20	10	19	14	20	10	15	6
		95.2%	47.6%	90.5%	66.7%	95.2%	47.6%	71.4%	28.6%
	No answer	1	0	2	0	0	2	5	3
		4.8%	0.0%	9.5%	0.0%	0.0%	9.5%	23.8%	14.3%
6 th grade	Correct	3	11	4	10	7	15	8	16
		12.0%	44.0%	16.0%	40.0%	28.0%	60.0%	32.0%	64.0%
	Wrong	20	13	19	12	13	8	11	5
		80.0%	52.0%	76.0%	48.0%	52.0%	32.0%	44.0%	20.0%
	No answer	2	1	2	3	5	2	6	4
		8.0%	4.0%	8.0%	12.0%	20.0%	8.0%	24.0%	16.0%

Table 3: The frequencies and the percentages of the correct (C) and wrong (W) answers of the pupils of the 5th and 6th grade in each of the four questions before (K1 to K4) and after (L1 to L4) the use of microworld.

It is evident from this table and from Graph 3, which gives the frequencies of the correct and wrong answers, that the wrong answers have been significantly reduced after the use of SEMI application. From the data analysis it is determined that there are significant statistical differences in the responses of the pupils, overall and per grade, before and after the use of the software. Using the t-criterion, the results for the 5th grade are: $t(41)=-6.089$ and $p<0.01$, while for the 6th grade they are: $t(22)=-3.53$ and $p<0.05$. For the total number of pupils the corresponding results are: $t(18)=-5.43$ and $p<0.01$.



Graph 3: Graphical representation of the frequencies of the correct (C) and wrong (W) answers of the pupils of both grades in each of the four questions before (K1 to K4) and after (L1 to L4) the use of microworld.

The statistical results of the answers, per grade, before and after the use of the microworld are very interesting. Using the t-criterion, the result of the answers to the questions L1 to L4 per grade before the use of the microworld verifies the presence of significant statistical differences

($t(40) = -2.66$; $p < 0.01$). From the findings it was determined that the pupils of the 6th grade gave correct answers in a greater percentage than those of the pupils of the 5th grade and this can be considered as normal. The factors that affect this result could be the age/grade and existing knowledge and it must be mentioned at this point that the pupils of the 6th grade had already been taught by their teacher about the concepts in SEM. The remarkable point is that the results in the answers L1 to L4 per grade, after the use of the microworld, using the t-criterion did not show the presence of significant statistical differences ($t(43) = -0.724$; $p > 0.05$). What has happened between the pupils giving answers to the questions K1 to K4 and to L1 to L4, was the use of SEM microworld, consequently this differentiation is derived from its effect.

4 GENERAL CONCLUSIONS

The candidate schoolteachers of the sample have, in general, positive opinions for the MicroWorlds Pro environment and they believe that using it, can strengthen considerably the educational process. These opinions, combined with the views of the pupils which find MWP environment attractive can provide the justification for the assertion that MWP effectively improves the educational process.

From the second part of our research, data analysis has shown that a significant percentage of the pupils disprove their initial misconceptions and they recognize the role that the rotation of the earth around its axis plays in the alternation between day and night. They approach the scientific model without the need for additional instruction from the tutor. It appears that the pupils using the simulations had the chance to explore the phenomenon and step by step to understand it. An important number of pupils showed that they can exceed the misconceptions and they can approach the scientific model for the particular phenomenon.

Form the final conclusions it is obvious that the MWP environment, using appropriate applications like SEM, can improve and enhance the learning procedure.

References

- diSessa, A. (2000). *Changing minds: Computers, learning, and literacy*. Cambridge, MA: The MIT Press.
- diSessa, A., Hoyles, C., Noss, R., & Edwards, L. D. (Eds.) (1995). *Computers and exploratory learning*. New York: Springer.
- Hake, R. (1998). Interactive engagement vs. traditional methods, *American Journal of Physics*, 66, 64.
- Jonassen, D. H., Carr, C., Yueh, H.-P. (1998). Computers as Mindtools for Engaging Learners in Critical Thinking *TechTrends*, v43 n2 p24-32.
- LCSI (2006). MWP. Available on line: <http://www.microworlds.com/> [retrieved on 12/20/2006].
- Logo Foundation (2006). Available online: <http://el.media.mit.edu/logo-foundation/index.html> [retrieved on 12/20/2006]
- Matsagouras, H. (1998). "Theory of teaching: The personal theory as a frame of thinking and critical analyzes". Gutenberg Publications, Athens, Greece.
- Novak, G., Patterson, E., Garvin, A., Christian, W. (1999). *Just-in-Time-Teaching*, Prentice Hall, NJ.
- Open World Learning (2006). *MicroWorlds in Action*. Available online: <http://mia.openworldlearning.org> [retrieved on 12/20/2006].
- Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York: BasicBooks.
- Papert, S. (1991). *Situating Constructivism*, in *Constructivism*, Harel, I., Papert, S. (eds), Ablex Publ., NJ.

- Reynolds, A. (1992). What is competent beginning teaching? A review of the literature. *Review of Educational Research*. Spring, 62, 1, pp. 1-35.
- Rieber, L. P. (2005). Multimedia learning in games, simulations, and microworlds. In R. Mayer (Ed.), *The Cambridge handbook of multimedia learning* (pp. 549-567). New York: Cambridge University Press.
- Rieber, L. P. (2006). Supporting Discovery-Based Learning within Simulations. Available online: <http://www.iwm-kmrc.de/workshops/visualization/rieber.pdf> [retrieved on 12/20/2006].
- Ventrella J. (1993). Attractions and Repulsions - A Gravity-based Animated Microworld for Exploration of Dynamical Systems. Available on line: <http://www.ventrella.com/Ideas/attractions.html> (retrieved on May 26, 2007).
- Vincent, J. (2001). Microworlds and the Integrated Brain. 7th World Conference on Computers in Education, Copenhagen, July 29–August 3, 2001.
- Vosniadou, S. (1994), The conceptual change in the children's age: examples from the field of astronomy. In: *Representations of natural world*. V. Koulaidis (ed.), Gutenberg, Athens (in Greek).
- Vosniadou, S. (1998), *Cognitive Psychology: Psychological studies and essays*, Gutenberg, Athens (in Greek).

StarLogo TNG – Making game and simulation development accessible to students and teachers

Eric Klopfer, klopfer@mit.edu

Associate Professor, Director, MIT Scheller Teacher Education Program, MIT Media Lab

Abstract

Prior Work on Learning Science Through Simulations and Models

For years we have been working with students and teachers on the Adventures in Modeling (AIM) program, a program which helps secondary school students and teachers develop an understanding of science and complex systems through simulation activities using our StarLogo programming environment. AIM offered computer modeling to science and math teachers as part of their professional development. AIM-trained teachers developed simulations that have become integrated into their standard classroom practices. However, activities in which students program their own simulations have been limited to a small number of classrooms. Our research has shown that there are several barriers to students developing their own models in science classes (instead of isolated computer classes) These barriers include the time it takes to teach programming basics, and teacher and student comfort with the syntax of programming languages. These barriers stand in the way of the deeper learning we have observed when students are given the opportunity to develop their own models.

StarLogo: The Next Generation (TNG)

To address these problems and to promote greater engagement of students in programming, we have designed StarLogo: The Next Generation (TNG). StarLogo TNG provides two significant advances over the previous version. First, the programming is now done with graphical programming blocks instead of text-based commands. This innovation adds a powerful visual component to the otherwise abstract process of programming. The programming blocks are arranged by color based on their function, which enables students to associate semantically similar programming blocks with each other. Since the blocks are puzzle-piece shaped, blocks can fit together only in syntactically sensible ways. This eliminates a significant source of program bugs that students encounter. Procedures are made from snapped together blocks, forming a visual chunk which helps students chunk the goal of the procedures in their mental models of the program. StarLogo TNG encourages students to spatially organize their block programs into “breed drawers,” which affords an object-oriented style of programming. StarLogo TNG’s second significant advance is a 3D representation of the agent world. This provides the ability to model new kinds of physical phenomena, and allows students to take the perspective of an individual agent in the environment, which helps them bridge the description of individual behaviors with system level outcomes.

Learning Programming Through Games

Our first take on introducing game development through programming was a course focused specifically on game design and development. The course was first implemented at a local secondary charter school for students ages 13-15. The classes were largely taught through “coaching” the students through projects. As the students conceptualized new game elements, they created for themselves new programming challenges that built on their current understanding, while requiring them to seek out or design new commands and algorithms. For example, the idea of a 3D Pac-Man style maze led to the need for a first person perspective and associated controls. To take direction from the game player, the idea of a loop that continuously checks for input was introduced.

We found that it took students less time to get started and use StarLogo TNG than traditional Logo (or StarLogo), thus lowering the floor for the language. Students were able to readily use the block programming, focusing more on concepts and less on syntax. The programming blocks prevented students from making errors that formerly frustrated beginning programmers and provided a certain amount of implicit programming guidance that text does not offer. When bugs did occur, the students only needed to debug their programming logic, rather than syntax. The textual abstraction of their idea was visually represented in StarLogo TNG, and they often pointed to and followed the programming blocks as they were debugging.

This game design component has been extended and become the basis for a workshop for middle school students learning to develop computer simulations to better understand issues in their community. Instead of jumping right into the development of research-based simulations, the students first become familiar with the fundamentals of models and interactive design through developing model-based games. This empowers them to take programming and simulation into their own hands.

Learning Physics Through Programming

Many high school kids find that analyzing the world, as they are asked to do in the physical sciences, is difficult and uninteresting. But ask a kid to create a world and you get a different response. This motivation was harnessed to teach basic mechanics concepts in high school physics. Building on the experience in the game design courses, we designed a physics curriculum around game development. We felt that this unit was justified on the programming experience alone, but believed that the potential existed for significant physics content learning as well. StarLogo TNG basics were introduced through a series of task-oriented activities.

Programming skills learned in a game design unit were then harnessed to teach physics content in a unit we called Modeling Change. Beginning physics students have a hard time believing that the vertical and horizontal motions of a projectile can be independent of one another. The concept of simultaneous but independent change is difficult for them to grasp. A programming task was used to introduce the idea of simultaneous but independent change. They saw that changes in color, size and location which they programmed separately, could be run simultaneously. They had no difficulty seeing that their agent's color change was independent from its movement in the x-direction. When students programmed vertical motion in the manner describe above, they were not satisfied. Their agents seemed to float up and down. To get realistic vertical motion, students needed to use procedures for accelerated change that were developed through a series of programming challenges. To get a turtle to jump over a ditch, they had to build and simultaneously execute separate move-forward and vertical-jump procedures.

The programming unit was followed by a unit on vectors and projectile motion. Assessments showed that the programming experience helped students learn vectors and see horizontal and vertical components of motion operating separately and simultaneously in projectile motion. In a report on a projectile motion lab, one freshman student wrote, "Programmers program motions independently in the virtual world, on the other hand physicists see the vertical and horizontal motion as independent of one another in the physical world. ... I believe when an object is moving vertically it is independent and not interfered by horizontal motions. Our ball fell within our predicted value. This confirms our assumptions of vertical and horizontal motions being independent."

Conclusions and Future directions

Teaching students to write programs can help them become digital producers as well as computer consumers. But they need powerful tools that can engage, motivate and empower them as they learn to program. StarLogo TNG is such a tool. Young people can use it to build 3D video games. This is highly motivating. TNG also supports the construction of scientific models. There is a productive overlap between games and models, and we have seen student game-makers become student model-builders, building and using scientific models to enhance

their content learning. Our work is driven by the vision of many students using StarLogo TNG to program exciting 3D video games driven by scientifically sound models.

A Glance at the Foyer of the Future

Márta Kőrös-Mikis, korosnem@oki.hu

Hungarian Institute for Educational Research and Development

Abstract

Preparing children for the information society is a prominent task of public education. The first ages of primary education are often called the entrance room of the future. Hungarian educational researchers have been conducting research and development projects since the 1990s in the field of applying ICT effectively among kindergarten and elementary school children to assist their skills and knowledge development. In the new version of the National Core Curriculum (2003) greater emphasis is given to ICT culture, the basics of which should be acquired as early as possible.

Thus the aim of our research during the 2005/2006 school year (Foundation of digital literacy in early childhood) was to investigate the recent utilisation of ICT and the adaptation of digital teaching methods (including Logo-pedagogy) in the introductory phase of education. In the interest of defining innovative and adaptable models, we primarily focused on best practices. The paper summarizes our experiences and presents some examples from our case studies.



Figure 1. My turtle – my friend. Logo-inspired drawings on the bulletin board

In schools with the best practice staff members typically exhibit an active interest in computer usage. The lessons we saw as well as the prevalent teaching methods reflected a modern approach: the computer was only a tool used in the course of practical activities dealing with information not an end in itself. Children concentrated on the given exercise, applying already acquired computer skills. The teachers took advantage of the children's flexibility and further improved their user knowledge, motivating them to tackle the given task with only a limited amount of instruction and information.

Keywords

digital literacy, educational researches, best practice, LOGO-pedagogy, methodology

Observing ICT based best practices at primary school level

In the new version of the National Core Curriculum (2003) greater emphasis is given to ICT culture too, the basis of which should be acquired as early as possible. ICT as a school subject was compulsory only from age 12 (from the sixth grade) with a minimum number of classes. Now ICT as a subject has to be introduced at the primary school level. The aim of research conducted by the National Institute for Public Education (OKI) during the 2005/2006 school year was to investigate the utilisation of ICT and the adaptation of digital teaching methods in the introductory phase of education. We were curious to find out the following: *to what extent are primary schools in Hungary able to provide a foundation for digital literacy; how prepared are they to help children acquire the basics of ICT knowledge in creative way with a view toward developing their skills, how familiar are schools with the didactic materials developed by OKI and similar organisations, and if so, what sort of teaching methodologies are used and how?*

We employed a complex *method of research* to answer these questions, the goal of which was to produce case studies for further assessment. ICT is not yet regarded as a typical subject or method of application in Hungarian primary schools and therefore lacks the technical background and teaching content that is currently present at the secondary school level. A representative survey on the situation of ICT in primary grades may provide numerical data concerning hardware-software and personal conditions, but in general, it would only reveal deficiencies. Such an approach would not lead to progressive educational policies or result in further dissemination in terms of popularising the topic.

For this reason, we began working with a small, but *carefully selected sampling of schools* with characteristics that might to some extent demonstrate - albeit not statistically - the current situation of ICT in primary education. The 7 schools we chose are located in both urban and rural communities. One of these is a "leading-edge" institution with more than two decades of ICT tradition in lower grades, but the sampling also includes a school where the subject has only recently been introduced among young school children - as of September 2005 - inspired in part by methodological developments posted on the OKI web-site (www.oki.hu).

Case studies also involved *the analysis of school documents* in order to reveal how these have incorporated ICT usage as stipulated by the National Core Curriculum. We examined the personal and material conditions in schools as well, and by *visiting lessons*, we were able to observe the developmental application of ICT in the classroom. Educators had the opportunity to express problems and make suggestions in the course of *guided interviews* with the help of a *questionnaire* we compiled especially for this purpose. In the interest of defining innovative and adaptable models, we focused primarily on "*best practice*" and also recorded personal interviews with a single student who was recommended by a teacher in each school.

Institutional priorities

It is interesting to note that in the majority of schools under observation there is a correlation between the personality of teachers who engage in best practice, comprising a well-intentioned initiative, and the appearance of informatics as an institutional priority. A question for further research may be whether institutional priorities attract and keep innovative educators or whether their teaching practice has an impact on the prestige of informatics within the given institution i.e. what sort of mutual effect these two factors have on one another. We are able to draw intriguing conclusions by examining the pedagogical programs of various schools from the aspect of whether they include informatics training as a priority for students in lower grades. Schools that place a high priority on informatics are those where schools officials consider development in this area to be a personal issue.

Informatics knowledge among educators

Since we regard the use of ICT in the course of acquiring digital literacy to be especially important both within and outside of the lesson framework, the level of user knowledge and experience among educators is a significant factor. It is worth noting that in all of the schools engaging in best practice, including those we have encountered in the course of previous research, the number of teachers with strong qualifications and user knowledge is very high. During our interviews, several school directors mentioned the beneficial impact of official efforts to assist educators in obtaining computer equipment. We estimate the rate of computer usage among the faculties we observed to be as high as 90%, but this does not necessarily mean that the direct application of IT in the teaching process. According to the active teachers we asked, their colleagues primarily use computers for word-processing, surfing the internet and electronic correspondence.

Our special survey revealed that in schools where best practice exists, faculty members typically exhibit an active interest in computer usage, which is generally manifest in their high rate of participation in computer training courses, competitions to obtain hardware, internet usage, correspondence and the skilled application of IT to create exercises for their lessons. Based on discussions with the colleagues we visited, it can be said that with the exception of a few older staff members, 80-90% of the teachers at the schools we surveyed use computers – but almost *NEVER during their lessons!* Use of computers in the teaching process is limited to preparing exercises and collecting data on the internet. It is worth pointing out the decisive role of the school director: the development of informatics training can be directly attributed to the director himself, who does in fact regard improvement in this area to be one of his most important institutional priorities.

Teaching or application?

Informatics lessons

The lessons we visited as well as the prevalent teaching methods reflected a modern approach: the computer was “only” a tool used in the course of practical activities dealing with information; it was not an end in itself. All of the lessons were referred to in their respective timetables as informatics, actual computer usage was preceded by a lengthy presentation phase, including elements involving movement and drama. It is important to mention that the classroom arrangement was specifically designed to support such activity. The centre of the classroom was occupied by a suitably large, roughly square-shaped space that served to accommodate group games, which the teachers used to prepare students for their computer exercises. In this way, computer usage was integrated with other activities, thus avoiding a situation where an informatics lesson is reduced to an isolated student-machine relationship.

Informatics appeared as a medium for transmitting pedagogical aims. It provided an important context for nurturing talent and creativity or it was a significant tool for compensating handicaps. In each case, the informatics lesson acted as a realisation of the local applied informatics curricula, the central aim being to develop intellectual, linguistic and social competencies through the use of ICT. It may have been for this reason that, irregardless of the curricular format, the teachers referred to their lessons as “composition classes” – and this is what they actually were: composition classes assisted by computers in the framework of informatics lessons. (Pányi-Segesdi, 2007) The teachers themselves used professional approaches that were very similar, but each school must perform different tasks as a result of their divergent socio-cultural environments, and informatics tools provided excellent solutions in both circumstances.

Informatics applied in other school subjects

In terms of methodology, the situation is entirely different when a classroom is equipped with one or two computers as opposed to one where every student has access to their own unit. All

of the teachers we visited saw opportunities for methodological diversity in adapting to the equipment that was available to them. During a mathematics lesson we witnessed two kinds of computer usage. First, a laptop and a projector were used to give a Power-Point presentation to assist the entire class in reviewing and practicing previously learned material, after which groups of 4 students each continued with more practice and skills development exercises using the “Manó-Math” software (Manó=Gnome). Meanwhile, the rest of the class engaged in group-work without the aid of computers, each group dealing with a different task. The visited drawing lesson in a little village school made maximum use of multimedia opportunities: after a lively, informative and interactive frontal presentation that served not only to set the mood, but also to help students brush up on their knowledge, the students worked individually to create their own winter landscapes using the drawing program on their computers.

In both of these lessons, interaction related to the operation of computers and software was minimal. The children concentrated on the given exercise, applying computer skills they had already acquired. In practically all of the lessons we visited, the teacher made one or two gestures that took advantage of the children’s flexibility and further improved their user knowledge, motivating them to tackle the given task with only a limited amount of instruction and information.

Developing children’s abilities

Once again, it should be pointed out that the *classroom arrangement* is also an important factor in the effective application of computers during the teaching process. Obviously, the arrangement of the study space should be well-designed to support computer usage while remaining subordinate to pedagogical functions in a broader sense. For children with special needs the available computers are discretely located at the back corners of the room, which radiates a relaxed atmosphere and contains an appropriately large space to accommodate tools of physical therapy. There is no separate computer room because in this case the children complete computer exercises in the framework of lessons that incorporate a variety of activities based on differentiated, individual development plans tailored to fit their needs (from physical exercise to drawing, and from sound-formation to vocabulary building). (Fig. 2.)



Figure 2. Child with special needs – computer is the only way of written communication

As the pupils carried out their tasks, we again observed that they possessed the necessary cleverness to operate both the computers and the software. Hence they were able to focus on the special need development exercises, which was the actual aim of the lesson. Also worthy of attention is the fact that teachers currently dealing with skills improvement are still isolated, working with software ordered and developed through their own personal initiatives, and yet the beneficial impact of computers on their activities is undisputable. Among others, they help to eliminate delicate movements that pupils are not capable of, provide opportunities for instant feedback and the correction of mistakes, break monotony with pleasant signals (images, melodies, “gifts”), and offer possibilities to measure improvement etc. A central initiative would

be very important, both in the area of training and further education as well as in the development of teaching aids, so as to make computer-supported skills improvement activities more common.

The outstanding features that computers provide in terms of simplifying certain tasks - the power to build confidence, measure achievement and motivate performance as well as the feeling of security inspired by concrete feedback - were clearly apparent in both of the aforementioned institutions.

Methodology

Thinking over the lessons we observed gave us the opportunity to summarise methodological issues arising in the course of computer usage and applications at the lower primary school level in order to focus on the solutions that were applied in best practice.

Space and timeframes

Successful activities were fostered by comfortable spaces; only one activity took place in a “partitioned” classroom, but even this room had a central space large enough for games involving movement. This means that opportunities for interactive multimedia utilisation and subsequent changes in the content of informatics as a subject led to a transformation in the optimal environment for computer-supported activities. As separated cubicles presumably designed for independent work are replaced by more open areas, good practice dictates that instead of small classrooms containing “computer cabinets”, informatics labs are increasingly being located in larger spaces.



Figure 3. Guess what! – game for motivation

Concerning the aspect of time, it was an intriguing experience to watch how appropriate pedagogical supervision enabled children to maintain their concentration throughout a 45-minute lesson, 15-20 minutes of which provided ample time for them to complete their work with the help of creative, open ended software. Experience has shown that primary school children grow tired after more than 15 minutes of computer work, but in the lessons we visited, their activity was playful and motivating. (Fig. 3.) Therefore, methodological variety, dramatic elements and games involving physical movement not only stimulated work from a didactic standpoint, but were also appropriately tailored to meet the children’s age-specific needs whereas they would have been overburdened by 45 minutes of continuous computer activity.

Dramatic games

Animated dramatic games added colour to lessons, but they had a significant impact on the development of skills from the standpoint of computer knowledge as well. Such games stimulate the personality as a whole, and in this regard it can be said that they are an excellent tool for developing skills in a variety of “disciplines” – not to mention social competency.

For example, in the memory game entitled “*What would you bring...?*” the children were each asked to name one or two important objects that they would take with them on a long journey, but before doing so, everyone had to repeat what their classmates had mentioned. This exercise develops attention and memory, but in terms of informatics, it also helps to practice the concepts of matching and ordering while giving the class an opportunity to listen to one another and become acquainted. Like during the well-known turtle games – according to Logo-pedagogy –, which are very popular, too. (Fig. 4.)



Figure 4. Traditional turtle-game in the computer room

When pairs of children were asked to put images in the correct order, we observed a very useful game that actually connects to several different areas of knowledge. Once the pictures were in the right order, each pair acted out the two-character story told by the images. At a later stage, one of the stories was illustrated using the computer, and since the children were personally involved, they were presumably capable of concentrating on the computer-generated drawings and text more than they would have if the story had been dictated for them to copy, or if they had been given an unprepared composition exercise. (Fig.5.)



Figure 5. Picture stories to motivate computer-generated drawings and compositions

Work organisation

Depending on the physical conditions, questions arise concerning how many children should work at a single computer unit and how exercises should be organised? *Individual work* is conducive to developing independence and creativity, but it requires that the children be

provided with a well-defined task and certain points of reference, and then be allowed to freely complete the exercise. (Fig. 6.) The teacher has requirements as well:

- The given exercise should focus on developing a specific skill and should be able to “teach” children the desired study material.
- Any problems that might arise should be anticipated beforehand since the teacher is under pressure to assist everyone simultaneously when the kids are working individually.



Figure 6. Individual work – creating a winter landscape

Teachers who use informatics for the personal development of pupils strongly support utilising the computer for *pair-work*. In each case, one child was working with the computer while the other acted as an “observer” and assistant. This work method divides the given task so that each child has less to pay attention to, and in this way, both pupils are capable of creating a “perfect” product. (Fig. 7.) At the same time, their cooperative skills, tolerance and sense of critique are further developed.

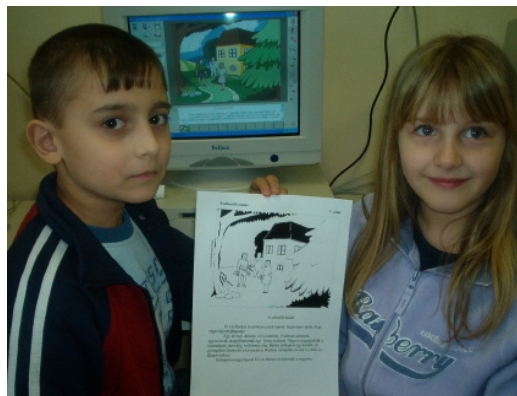


Figure 7. Printed product of pair-work at the end of creative writing lesson

The use of computers in *group-work* presents another well-known problem: each child should have a specified task and a distinct role within the group, and all participants should be active and able to cooperate well with each other. Due to the inherent physical conditions of computer work, however, the effectiveness of cooperation in groups of more than 3 participants is questionable. Even a single computer can be used for a *frontal approach* in the classroom (a more frequent solution is to make a presentation using an “equipment cart” with a laptop and a projector). In place of traditional forms of explanation by the teacher, frontal presentations provide many opportunities to motivate students – this method of presentation requires only that a suitable amount of freedom be available to answer and discuss spontaneous questions from the pupils.

The acquisition of user-knowledge

We made no secret of the fact that the aim of our visits was to observe digital applications in teaching practice, and for this reason our colleagues were prepared to present us with this kind of activity during their lessons. At the same time, it was surely not by accident that we saw no direct informatics training (programming), not even during informatics lessons! In spite of this, all of the teachers consciously chose enjoyable exercises and activities that enabled the children to learn a new element of user-knowledge with the help of instructors at the appropriate time within the lesson process! This approach seemed to be an effective way of transmitting new computer skills, especially if we take into account that the children we visited were very clever in handling computers, and that their existing skills were presumably acquired through the same teaching approach. (Németh, 2004) It would be highly useful to publish a teacher's handbook containing examples of this approach along with methodological recommendations so that at the widest possible range of educators could be exposed to best practice in this area.

Reflections from children

In the course of our research, we asked one or two children at each site about their computer habits, about the difference between “computer usage” at home and at school, and about the mutual impact that these activities had on one another. The majority of interviews revealed that children are happy to use the skills development software provided to them by teachers (or downloaded from the Internet) at home as well, making use of the skills they have already learned at school. Some pupils also use software copied for them or given to them as gifts by friends and relatives. The children's “partners” at home include their parents, older siblings, cousins and friends – in other words, anyone in their immediate environment can serve as a role-model in terms of computer usage. In addition to what they have learned at school, the children also learn computer functions that other individuals have shown them how to use, which means that each child's image of computers varies greatly. Based on our interviews, we suspect that computer habits are largely determined by the socio-cultural environment. It would be worth launching more in-depth research with regards to the connection between computer habits and social strata. It is likely that such research would assist in the development of teaching material that is more suited to parental needs, producing a structure in which the knowledge children have acquired at school has a greater impact on their computer usage at home.

Summary

When observing best practice in the course of informatics training for primary school children, it was striking to see how *computer-application came to be the central focus* of lessons as opposed to the actual subject of informatics – even during informatics lessons! In schools where teachers engage in best practice, even their colleagues possess reliable user-knowledge and use computers on a regular basis in the course of their daily work. The work of colleagues actively using computers in the teaching process reflects an overall trend based on methodological experience gained through professional workshops or through personal initiative, in some cases supported by more than 10-15 years of practice. At present, enough *general experience is available* to make collecting and organising it worthwhile, and the data accumulated through our research at participating schools can contribute to this process as well. (Kőrös-Mikis, 2006) Based on further analysis of our results and experiences, and in the interest of improving the quality of informatics training and ICT-application at the primary school level, we intend to draft a *separate proposal* to the Ministry of Education in order to facilitate further development and decision-making with regards to education policy in the future.

References

Márta Kőrös-Mikis (2006): *ICT for Children. Motivating Primary School Teachers to Use ICT.* <http://www.oki.hu/article.php?kod=english-art-Koros-ICT.html>

Zoltán Németh (2004): *On the Road to the Information Superhighway. Using ICT in 1st grade classes at Móricz Zsigmond Primary School, Győrszentiván, Hungary.*
<http://www.oki.hu/oldal.php?tipus=cikk&kod=english-art-Nemeth-Superhighway>

Nóra Pányi-Segesdi (2007): *Fogalmazástanítás IKT-val (Composition Classes with ICT).* Tanító. XLV. 2. 15-17.

Half-baked Logo microworlds as boundary objects in integrated design

Chronis Kynigos, *kynigos@ppp.uoa.gr*

Educational Technology Lab, University of Athens and RACTI

Abstract

The paper addresses the problem of fragmentation of the communities involved in the design of digital media for education. It draws on the experience gained at the Educational Technology Lab in the design of Logo-based microworlds with three different platforms respectively based on component computing, 3D game engines and 3D navigation with a GIS. The construct of half-baked microworlds is used to discuss how the Logo community can contribute to the quest for integrated design. These kinds of microworlds implicitly exist within the community, but they can be explicitly designed mediated and put to use in the role of facilitators for integrated design and development to enable a growing communication amongst researchers, technicians, teachers and students.

I use the term “half-baked” microworlds to describe digital media designed to facilitate communication between researchers, technicians, teachers and students as they become engaged in changing them. Microworlds have been the main Logo-based vehicles through which the key ideas of generation of meanings through communicational and constructionist activity have been mediated within the field of instructional design (Goldenberg, 1999). After 40 years, we have a lot of experience of how difficult it is to convey such meanings in education and more so, to have hopes for a progressive infusion of such practices in formal schooling (Kynigos, 2004). The demands for re-addressing epistemologies and personal pedagogies, changing roles and communicating about such changes, let alone the sheer logistics within the socio-systemic educational context, have been too big to see some serious scaling-up of Logo-like educational practices. At the same time, the production of microworlds and escorting materials has not scaled up either. It has been hard to convey the new kinds of pedagogy involved as well as the kinds of interaction with the technology envisaged in such educational environments. The process of designing educationally principled materials for microworlds requires integrated expertise which is hard to find. Many pieces of educational software and corresponding materials at large look more like products of fragmented views of the problem, emerging from traditional curriculum designers who miss the opportunity for added educational value, from technicians in the software industry perceiving design for education as just another field of application of generalized design processes, researchers who have little understanding of the pragmatics of education, students who inadvertently perceive their role as requiring memorization and response to tests.

The microworld community has a lot to offer to the task of integrating expertise in the design of educational software and addressing innovation in educational systems. What is needed is to develop a much more articulate language to convey the idea of design for reflection and for the generation of integrated expertise. Out of the communities mentioned, only technicians and maybe researchers perceive themselves as designers. Yet, design can be an activity which is tightly linked to constructionism for learners and to reflexive practice for teachers. Communal design can generate the need to be explicit, to reflect and to express meanings through argumentation. During a collaborative activity, a community works towards a common goal, which can be an ideal object to be created, or a specific /improvable/ object (Bereiter & Scardamalia, 2003). This object is both the centre of the activity, and also functions as a communicational tool to shape a common language within the community. Cobb et. al. (2003) extend this notion by proposing the term “boundary objects”, e.g. specific objects within different communities, which are «/relatively transparent means of conveying meaning among the

members of the community who created them/». They can also be the centre around which community members organize their activities and can additionally operate as tools for communication among the members of the same community, and the members of other communities.

Half-baked microworlds are meant to be “improvable boundary objects” and the process of changing them can be orchestrated to enhance integration in microworld design.

Half-baked microworlds are pieces of software explicitly designed so that their users would want to build on them, change them or de-compose parts of them in order to construct an artifact for themselves or one designed for instrumentation by others. They are meant to operate as starting points, as idea generators and as resources for building or de-composing pieces of software. This constructionist activity is seen as part of inquiry and argumentation leading to the generation of mathematical and scientific meaning. At the same time, half baked microworlds can be designed for teachers to engage in pedagogical or epistemological reflection as they de-construct or re-construct the microworlds in a context of designing tools for students.

Expressing meanings by changing the functionalities afforded by microworlds is something which has been happening through the years in the relatively small world of Logo friends, the microworlds originally designed or used by one or more of these communities. However, they have not frequently been explicitly conveyed as media to be questioned and changed in themselves, but rather as tools to explore and construct with and possibly to extend. Moreover, the questioning of microworld functionalities and interface has seldom been perceived as an activity for people from such communities to interact with each other in order to acquire some shared perception of concepts, perspectives and goals.

The evolution of perceptions of microworlds

Although the term ‘microworld’ was born with the advent of Logo a long time ago, perceptions of its substance and its purpose and functionality have evolved quite drastically over the years. The term was originally borrowed by Seymour Papert from artificial intelligence (A.I.). Its meaning evolved through the years within the mathematics and science education community. Papert described it as a self-contained world where students can “learn to transfer habits of exploration from their personal lives to the formal domain of scientific construction” (1980, p. 177). A more recent description of micro-worlds is that they are computational environments embedding a coherent set of scientific concepts and relations designed so that with an appropriate set of tasks and pedagogy, students can engage in exploration and construction activity rich in the generation of meaning. The first example of a microworld was ‘Turtle Geometry’ within the Logo programming language which generated a bulk of research on the construction of children’s geometrical meanings (Laborde et al, 2006). Later research however, involved learning with the use of micro-worlds embedding a much narrower set of mathematical, scientific and other concepts, escorted by more focused theories on learning and pedagogy (Noss & Hoyles 1996, Edwards 1995, Clements & Sarama 1997, Sarama & Clements 2002).

In the early days, microworlds were seen as strictly programmable environments and their main characteristic was the ability to construct graphical models using a programming language. The semantics and rules of the language as well as the graphical representations were integrated with the concepts and relations of the domain characterizing the microworld. The key features of the students’ activities enabled with such media were the ability to make constructions and change and extend the rules and relationships of the microworld itself. The highly editable nature of these media and their executable representations providing immediate and epistemologically succinct feedback (they can be characterized as conceptual mirrors) enabled exploration and bricolage and the kind of learning which took place was characterized by Papert and his team as a special kind of constructivist learning which they termed ‘constructionism’ (Kafai & Resnick et al, 1996). Seen in a wider context, microworlds were examples of the idea of ‘deep structural access’ (diSessa, 2000, Kynigos, 2004, 1995) to technologies for non-technical people, allowing for creativity, customization and personal construction of technological tools. This idea was used

as an argument for the use of technologies to facilitate the growth of a technological culture rather than to inhibit creativity and expressivity with technologies.

In the field of education, with the advent of new interface technologies and the broadening of the domains over which microworlds were designed including that of science, programmability and constructionism faded in favor of dynamic manipulation of representations and higher quality graphical and iconic representations offered by multimedia and enabled by a suite of authoring systems. In science the microworld rhetoric quickly gave place to that concerning simulations. One key issue was the primary importance given to the accuracy and detail of the behaviors and relationships between objects in the respective simulation. This was based on epistemologies primarily focused on the phenomenological nature of the domain rather than on meaning construction. Another issue was the extent to which the simulation represented objects as realistically as possible or provided an abstraction from the objects' shape and form (as in Interactive possible or provided an abstraction from the objects' shape and form (e.g as in Interactive Physics). A third was the ways in which the simulation was mediated to students and the way they themselves perceived it as a real infallible phenomenon in itself or as a human-made simulation consisting of an inevitable approximation of the objects and the simulated phenomena.

In mathematics on the other hand, the problem lies with the epistemological nature of the domain itself. Mathematics can be seen as an abstract domain of human thought, its objects being axioms, relations and mathematical proof. On the other hand it can also be seen as an applied science explaining various phenomena (economic, scientific, geometric, navigational, etc). Microworlds were seen to make mathematical representations useable and learnable and thus gave rise to some questioning about the learnability of traditional mathematical representations, such as formalization, providing alternative representations designed to facilitate meaning generation (Hoyles et al, 2002).

In both cases of science and mathematics education, however, programmability and constructionism re-appeared recently with the advent of technologies allowing for programmable simulation-multimedia style tools such as E-slate microworlds (Kynigos, 2004, Penner, 2001), Boxer microworlds (diSessa, 1997), Micoworlds Pro and Imagine Logo. These kinds of microworlds constitute simulations which can be used at different levels by teachers and students alike. They can be used as phenomenologically authentic simulations providing iconic interfaces to change parameters within worlds with unchangeable rules and at the same time, as questionable worlds where rules can be changed and tampered with to generate inquiry of what it would be like if they were different. So, the issue in question is not 'when is a microworld half-baked' and 'when is it complete'. Any respectable microworld in that sense is half-baked otherwise it becomes doubtful if it is a microworld at all. In this paper I use the term half-baked to describe a microworld which is explicitly designed to engage its users with changing it as the main aspect of their activity. Furthermore, it is necessary for this kind of microworld to be conveyed as half-baked to its users. In this paper I discuss this kind of microworld as a tool for people with diverse expertise and/or roles to communicate.

Microworlds as tools for pedagogical innovation in schools

Engagement with working with half-baked microworlds involves the assumption that the overall goal is to design a learning environment which has good chances of providing some added educational value and which at the same time will be based on the use of these digital artifacts. However, both design for innovation and use of digital technologies have posed difficult challenges for education. These challenges have changed in essence during the past 40 years but not in magnitude.

The microworld idea was first coined in the late seventies by Papert. In his argumentation he presented this kind of technology as a medium to challenge and bring about re-thinking of the educational paradigm of schooling and the epistemological approach of content domains such as mathematics. This rationale came about in the historical context of some bold attempts for

curricular changes in the United States based on the problem-solving movement which focused on problem-solving methods rather than the understanding of mathematical concepts in themselves. Microworlds were seen as the ideal technological boost to the movement. Inevitably, they thus became victims of the educational innovation pendulum (Agalianos, 1997, Noss, 1992). They also became victims of the early stage in which they were created with respect to the spread of technology in the culture (Papert, 2002). Technologies were very hard to access, machines were slow and non-dependable, the internet had not arrived and people were not widely using digital technologies for something else. However, microworlds continued to be designed and used in design research initiatives providing evidence that when placed in the role of tools within a carefully designed and supported educational environment, they could greatly facilitate the generation of mathematical and scientific meanings in students. Furthermore, they appeared in contexts different to the one they originated from in countries such as Mexico, Kosta Rica, Greece and others (Blikstein&Cavallo, 2002, Kynigos, 2002). Recently there seems to be growing indication of a comeback of microworlds in a different role, that of digital artifacts for argumentation and inquiry learning in a digitally rich culture, which by itself will provide an external challenge to the schooling paradigm (Papert, 2002). At the moment, however, that culture is lagging greatly in integrating the use of digital media and reciprocally in evolving with respect to the traditional school-ish research paradigm. One of the reasons is that digital media are designed and developed by fragmented communities.

Microworlds for scientific and mathematical thinking with

The term 'microworld' has been used in the Educational Technology Lab to describe an explorable and malleable digital environment characterized by an embedded knowledge domain, acting as a coherent world, inside which a student can explore alternatives, test hypotheses, and discover facts that are true for this world. The basic characteristic of a microworld is thus that it is designed as a rule-governed environment which is accessible for manipulation and exploration by the learner. The kind of manipulation may vary according to the subject domain and the decisions of the designer. The technological affordances which we used and emphasized at the Lab were such so that a microworld:

- Constitutes an exploration space
- Mediates between informal and formal
- Provides executable representations
- Offers dynamic manipulation
- Evokes interplay between private and public expression
- Generates interdependent representations
- Allows deep structural access to its functionality by means of a programming language

Microworlds constructed in this way provide powerful sites for inquiry based learning. They offer to the student opportunities for construction, exploration, observation, reflection and collaboration (see for instance, Noss and Hoyles, 1996). When, on the other hand, we consider how microworlds are actually used and function as learning environments, we may describe them as a digital environment where the learner can

- manipulate the objects dynamically in order to discover their properties and understand the laws which dominate their behavior
- make hypotheses and use the feedback from the environment to test them, in order to correct his or her own ideas about the domain modeled by tinkering with and debugging the microworld and
- use programming as a rigorous language to understand and change properties and behaviors of the objects

Within learning environments based on the use of microworlds the learner is not told about a mathematical or scientific concept; on the contrary he/she is supposed to generate meanings about the concept through interaction with the microworld by playing a game or solving a problem.

Although there seems to be a distinction between programmable-constructionist and simulation based microworlds, the ETL group has adopted an integrated approach providing the first type of affordances to simulation-like tools. The emphasis was to design for the potential for users to change the rules and the laws that dominate the microworlds themselves. In that way, when the didactical design and time constraints allow, students can question the rules and the concepts underlying those rules e.g. by asking 'what if' questions and being able to test them. This notion is hard to convey to technicians and teachers who perceive programming to be a black-box infrastructure to some GUI and digital artifacts to be valid unquestionable pieces of curriculum material respectively.

Microworlds as tools for integrated design

The ETL group has been engaged over the years in the collaborative design of microworlds to be used as constructionist tools for inquiry and argumentation. This collaboration has involved all four communities of researchers, technicians, teachers and students. This is very hard to do, especially amongst actors with different expertise, role and views about the task working in a variety of contexts.

Microworlds have been perceived by the community at large as tools designed by researchers for radical innovation to be implemented in small scale situations. The era when this approach was valid and seductive to some communities outside our own seems to be ending. There is now demand for large scale initiatives and accreditation of new efforts before they have had the chance to become infused in educational curricula. The ideas behind the microworld culture are proving hard to grasp and accept not only by school systems but also by other stakeholders in education such as new computer science and telecommunication communities. This poses new challenges to the microworld community of finding methods and avenues for communicating these ideas to a wider audience in a language which is widely understood.

This kind of language can only emerge from experience of collaborating with such communities it is not something which can be defined at a theoretical level. The process will require being explicit about what happens when collaborative design and implementation is taking place. In this paper, we have an example of such an effort where different kinds of communities tried to find a common language to describe microworlds. This language may seem well understood and not very consequential to our own community. What's important however is that it seemed to be understood by others. Half-baked microworlds are tools designed to facilitate the production of such a language. They are *improvable boundary objects* in that they afford changes which operate as an ice-breaker between different communities. In discussing and implementing such changes people with different expertise, role and background negotiate and make themselves explicit. The results don't really matter, they can be one jointly agreed upon microworld or each community can go their own way and present alternatives. The importance is that in doing so, there is better understanding between them which often leads to the creation of hybrid actors (Kynigos, 2002), i.e. people with one expertise who understand enough about another so as to make sense and use it for joint design. In the next section I describe our own platforms for developing microworlds which were designed to address wider communities than our own. Subsequently, I provide an analytical description of a half baked microworld explaining how it was taken up and reformed through integrated design.

Some alternative platforms for developing Logo-based microworlds

The microworlds that have been developed at ETL have so far been based on three platforms for microworld development: e-Slate, MachineLab and Cruiser. The main characteristics of the platforms are presented below.

The e-Slate platform

E-Slate is an authoring environment which looks like a construction kit for creating exploratory software. The 'bricks' of this kit are software components of a large technical and functional variety ranging from a relational data-base to a slider or to a Newtonian simulator scene. The 'glue' to put bricks together comes in the form of pre-fabricated links which look like puzzle bits for users to stick together. However, there is a Logo language which plays the role of programmable ways of connecting components and of determining various functionalities within these. Our aim was to meet non-technical people half-way, i.e. to provide them with interesting black-box functionalities but also with the potential for them to easily create component configurations or custom ways in which they communicate with each other. In this way non experts can become software developers without being involved with data-models, algorithms, and low-level programming. Software construction resembles assembling a "kit" rather than "writing a program."

The MachineLab platform

Machine-Lab is an authoring environment and test-bed for the creation and exploration of interactive virtual reality simulations (VR-microworlds). The representational infrastructure provided by Machine-Lab concerns controlling and measuring the behaviors of objects and entities in virtual environments simulating real three-dimensional phenomena and spaces. The design of MachineLab includes various kinds of representation of three dimensional worlds and objects enhanced by interactive and dynamic display affordances for the user.

MachineLab is implemented over the 3impact game engine (www.3impact.com) which wraps the 3D graphics of the Direct3D engine and the distributed/multi-user capabilities of the DirectPlay engine, together with a 3D sound engine and the Open Dynamics Engine (www.ode.org), which is a Rigid-Body Simulation engine with realistic physics support for 'collidable' objects and articulated bodies (rag-dolls / humanoids). There is a Logo language available to develop microworlds providing multiple 3d external representations of some aspect of the physical world. The platform offers the possibility of multiple observations from different point of reference (cameras). The user may adjust a virtual camera on any object, moving or still, and observe the phenomena using multiple system of reference.

The Cruiser Platform

Cruiser is a geographically oriented information system. It resembles GoogleEarth-like technology providing geo-coded data on 3D maps. The main representation of its interface is a geographical and/or cartographical three-dimensional space, supporting dynamic navigation. However it is particularly interesting for the development of educational/mathematical games since it supports:

- a) a higher level of personalization features, such as custom channels with structured designer-defined access rather than one global map-for-all and,
- b) educationally rich capabilities such as the mediation of geographical data, including location signals in real-time.

Originally it was not meant to be a platform for Logo-like microworlds. However, a Logo language was developed and ported onto Cruiser. In this paper I describe how this provided the possibility to create a microworld for programming airplane avatar flights which we called 'Cruislet'.

Some negotiated descriptions of microworlds

At the ETL we have been engaged in a number of multi-organisational projects where the development of microworlds has been a significant part of the project activity. Typically, their design and development has been the result of collaborative activity between teams with different expertise corresponding to the four communities mentioned above. Each community inevitably had its own agenda both with respect to the point of taking part in such activity and with respect to what they could do with the products. These agendas had to be communicated amongst teams and objectives had to be articulated at the level of abstraction which satisfied all parties (see Kynigos, 2002). Also, putting together all these kinds of expertise proved a very difficult labor intensive task involving the gradual growth of a common language and understanding of what each party could bring to the joint work.

We designed and put to use all three platforms as facilitators for the integrated design process. We've had the most experience by far with E-slate which is being developed since the mid-nineties. However, both MachineLab and Cruislet allow the creation of a variety of 3D microworlds based on the use of Logo programming. The use of MachineLab involves the creation or use of 3D assets and the provision of programmable properties and behaviors to each one. This is not necessarily so with MaLT, the MachineLab-based 3D Turtleworlds software which is focused on only one such asset, the turtle. The use of Cruislet takes the underlying 3D geographical representation as a black-box infrastructure. Microworlds so far have consisted of different configurations of airplane avatars and their individual or dependent behaviors. All three platforms are based on the idea of 'black and white box' design (Kynigos, 2004). That is, instead of having a generic platform where everything can be developed through the use of a generic language (as is the case in Boxer or Logo based environments), there is a combined one, where some selected functionalities or objects are given as black box entities. This enables the user to start design and development with some technically complicated objects as a given (components in E-slate, 3D assets in Machinelab, 3D navigation in a GIS in Cruislet). It is not as conceptually powerful as generic designs but it meets users half way since it allows them to create impressive software rather quickly.

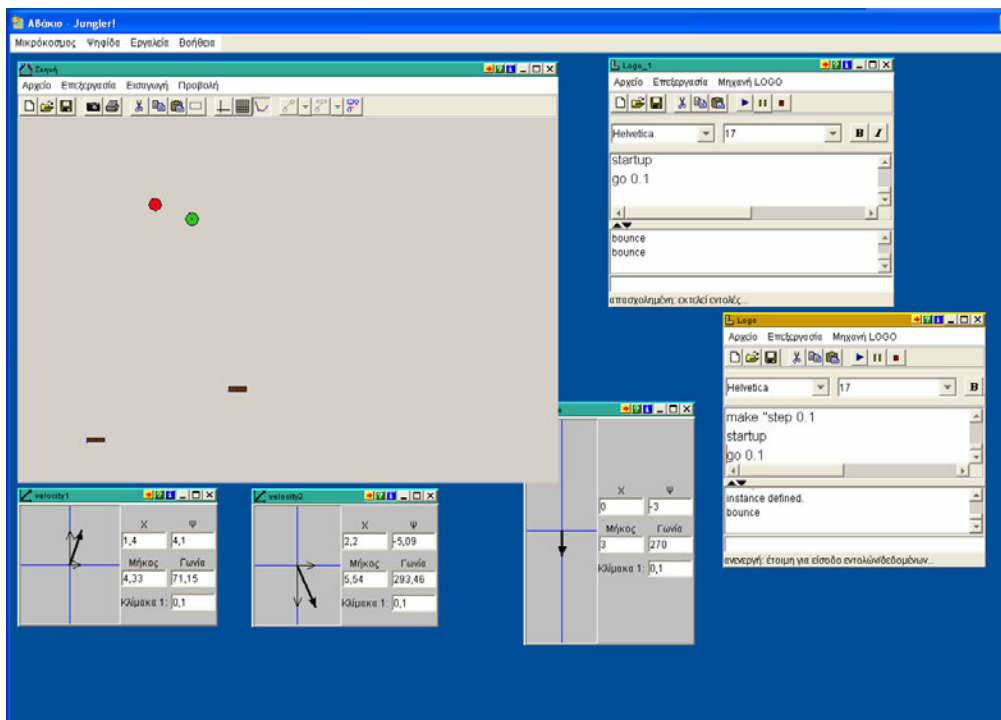
Over the years we have tried different models of collaboration during the integrated design process. One model is 'the two big machines' where a large group of developers work with a group of researchers linked with a group of teachers. The interaction between the two groups is done through one or two hybrid actors. This model has the advantage that a lot of effective work gets done, but the problem that real integration can be rather thin (diSessa, 2004). Another model is that of a researcher, a teacher and a developer working closely together. Much better integrated work is achieved but development has to begin at a much higher level and developers need to have the motivation to delve into the specifics of the domain and the functionalities of objects and relationships. Conversely, educational researchers need motivation to create the microworld and find some added value for its use.

Half-baked microworlds have either been designed as such from scratch by the researchers at ETL or have emerged in the process of microworld development. That is, microworlds which began as projects to construct ready-for-use digital media turned into objects for re-negotiation and design of new microworlds. In the next session, I describe four microworlds, two of the former kind and another two of the latter. The focus here is on the way they are described for which I'm using a common template. This template emerged as a reasonable way to describe microworlds so that they are understood by all communities concerned and was a result of many discussions aiming to make some sense between the groups. I've added two fields at the end, 'procedure of integrated development' and 'emergent uses of the microworld' to describe the context and the itinerary of design which began from the generic half-baked microworld.

The Juggler

Represented Phenomenon

The subject domain for this microworld is velocity and acceleration in a Newtonian space. A 'Juggler' with two bats tries to keep a green and a red ball in the air. Only one bat can be moved at a time. The initial velocities and field force are set by means of dynamically manipulating respective vector representations. Once the game starts, the velocity vectors become measures of velocity and thus change continually. The field force vector of course does not change once the game has started. The force can be set to the magnitude and direction of gravitational pull, but the student is free to set its direction and length to any parameters.



Level of abstraction

The representation of the entities is abstract, in the sense that the objects are represented as geometric figures. On the other hand since the shape of the actual objects, balls and bats, is very close to the specific figures used for the representation, we may say that besides its abstraction, the representation has some realistic characteristics. The representation of the velocities and field force however is abstract and in the form of a vector which is manipulable prior to starting the game. Also, some specific conventions made with respect to the simulated physics (such as the collision rules and the angle of the bats) give a sense of abstraction to the game.

Concepts embedded

Concepts at the kernel of the microworld are velocity, acceleration, gravity, collision. The microworld's 'conceptual field' (Vergaud, 1987) however, also consists of functional relations (time vs point on plane), change and rate of change, vectors and vectorial entities, angles (in collision rules).

User Interface

Description of the scene

The scene contains the two balls and bats. It is possible through the programming interface to add or take away balls and bats. There are the three manipulable vector representations. Also there are two instances of the Logo code, one for each ball, and the game starts by executing the start commands.

Manipulation

The velocity vectors operate as controls before the game starts and as measures after it has begun. The field force vector operates in the same way, however, as a measure it remains constant. Each bat needs to be selected and then can be dragged anywhere on the field before or during the game. Only one bat at a time can be dragged.

Programmability

This microworld is designed for questioning the respective concepts through changes in the code of the simulation which for each ball is about one page long. The students can make changes to the velocity and field force equations, can change the rules of collision and the properties of the balls and the bats (size, mass, shape, color etc). They can also add game features like scoring, ten best scores etc.

Feedback from the computer

When changing the parameters

Parameters can be changed either by changing values in the programming code or by manipulating the vectors before the game starts. Feedback is given by means of the behavior of the bats and balls during the game. The user needs to act in time to keep the balls in the air. In doing so, she/he develops strategies to anticipate the balls' trajectory based on the concepts and relations at hand.

As result from the experimentation

The results of the experimentation are open and related to the ways in which the game is setup either through the programming code or through the manipulation of the vectors and bats before the game. In a sense there is continuous results given by the game since it only stops when the user presses the stop button on the programming components. There is no 'success-failure' element to the game, when the bat misses the ball, the latter continues to travel 'out of site' and its velocity can only be seen by means of the numerical measure on the vector component. The user can stop the experiment and start again.

Designed learning process

The simulation provides the learner with a tool for investigating and questioning the scientific laws and underlying mathematical relationships in Newtonian space. When the game is setup so that these laws apply, the user can play the game and get a feel for the behavior of a particle (ball) in motion in Newtonian space. By using the vector representations they can at the same time get a feel for the essence of vectors as mathematical objects and as measures of certain entities such as velocity. However, and perhaps most importantly, the user can question Newtonian laws and make up their own ones by changing the rules of the game. Conversely, the game can be given to students with non-Newtonian rules and they can be given the task to convert it.

Didactical design

The microworld is designed as a game to be played, questioned, changed and designed for others to play. There is a variety of didactical designs which can be applied, from task setting to more informal projects of devising games and making rules explicit. They key to the whole

process is frequent reflection, argumentation and articulation of generated meanings, rules and hypotheses.

Type of representations used

The Juggler microworld combines phenomenological representations on the Scene with mathematical representations through the vector interface and programming code. The game can be played exclusively by means of manipulating the vectors or by an integrated approach including changes in the code of the simulation.

Procedure of development

The Juggler was developed by researchers as a half-baked microworld. One emergent microworld, the Balloon (see below), was built by a science teacher as designer-in-training and a Logo literate mathematics teacher doing the code helped by a developer on demand ('SEED' project). Another (called 'Marbles') was done initially as specifications given by a team at the University of Neuchatel ('ESCALATE' project) and subsequent realization that very close co-design between the team, a Logo literate science teacher and a developer was necessary. Finally, another one was done by high-school students helped by their computer science teacher simulating a custom arcanoid game.

Emergent microworlds

The original Juggler was used in a mathematics student dissertation to teach functions and in a case study to teach the meaning and functionality of vectors. It was also picked up by a the science teacher within the 'SEED' project who built a simulation representing relative velocities of a balloon and a particle thrown at it. The balloon microworld included graphs and sliders. The programmability and gaming features were meant to be less apparent to students. For the 'ESCALATE' project, the Neuchatel team had a specific agenda to build a simulation of a Piagetian experiment involving gravitational pull, friction and collision in a system of a marble sliding down a slope to hit another two onto an upward slope. Sliders were used for a range of parameters. Finally it was used as a starting point by high-school students in a computer club to develop an 'arcanoid' game. The students took away the Newtonian rules and focused on adding new custom features to the arcanoid such as user placement of the bricks and free movement of the bat (Kynigos et al, 2006).

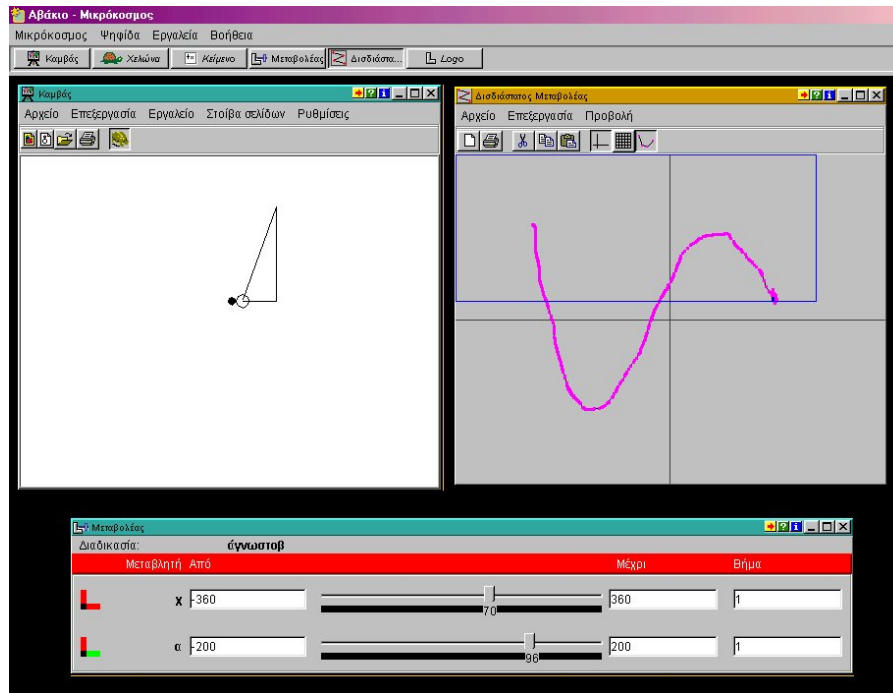
The Right Triangle

Represented Phenomenon

A geometrical figure is created by means of a Logo procedure with two variables :x and :a by using the Turtleworlds software. For some pairs of values the figure becomes a right triangle, for others it is just a jagged line. When it is a triangle :x stands for an internal acute angle and :a for one of its sides. The other side is created by relating :x and :a with the sin function. Students are asked to explore the kind of relationship between :a and :x so that the figure always becomes a triangle. They are encouraged to use the uni-dimensional and bi-dimensional variation tools of the software to do this.

Level of abstraction

Within the context of mathematical epistemology, this representation is rather concrete since it provides a graphical output of an instance of the abstraction signified by the variable procedure.



Concepts embedded

Sinusoidal functional relationships within a right triangle and of course other corollary concepts such as angle, function, variable.

User Interface

Description of the scene

It's a Turtle Geometry setup with editor and graphics windows. On top of that however, there is a uni and a bi – dimensional variation tool. Once a variable procedure has been executed the user can click on any part of the turtle trace to activate the former tool which understands which is the underlying procedure and the number of variables. It provides a slider for each variable. When the slider is dragged the figure changes Cabri-style in connection with the changes in variable values. The bi-dimensional tool is connected to the uni-dimensional one, provides a plane where free dragging produces co-variation between two variables and Cabri-style movement of the figure.

Manipulation

Apart from the usual Turtle Geometry functionalities, the user can drag the variation tools, edit their range and step of change. They are supposed to go back and forth from editing the code to trying out how the figure changes with the variation tools.

Programmability

This microworld is based on a program give to the students from the beginning. The students are told that the program is buggy and they are in the role of fixing it by means of experimenting what it does and trying out changes to the code.

Feedback from the computer

When changing the parameters

Parameters in this case re the procedure variables. Feedback is in two forms, one is a direct forming of the figure when a procedure with specific variable values is executed. The other is in Cabri-style figural changes when either the uni- or bi- dimensional variation tools are used. There is also the usual Logo textual feedback.

As result from the experimentation

It's very hard to hit two values which create a triangle without the variation tool. By using the uni-dimensional one, student reach specific standalone pairs of values but it's still very difficult to make a sense of what kind of relationship there is between them. By using the bi-dimensional one, dragging the mouse on the plane leaves a sinusoidal trace once the students get the hang of moving it so that it always makes a triangle.

Learning process

Students are of course not given a hint about the necessary relationship. The point is for them to integrate the use of all three representations, text, figure and variation tool to experiment, explore and express some theorems in action. The trace on the bi-dimensional tool provides a hint as to the nature of the relationship. It is then up to didactical engineering and the level – expertise of the students to move on to working out the solution and expressing it by means of de-bugging the procedure so that it always creates a right triangle. The students can subsequently put the theorem to use by creating shapes and animations based on right triangle shapes.

Didactical design

A lot depends on the experience and expertise of the students of course. The main didactical agenda could be focused on the process of doing mathematical exploration and on the support for engagement with argumentation over the concepts of sinusoidal functions periodicity, variables. The activity is open to a range of social orchestrations from one on one to collaborative pairs, large group debates, presence or distance argumentation and exchange of ideas and constructs.

Type of representations used

The representations are mathematical and thus address abstract concepts and relations. However, the figural representation constitutes an instance of a generalized figure and the segments are created to signify the concept of Euclidean line (i.e. without substance). The figures can be animated through the variation tools providing a sense of the ways in which they can change.

Procedure of development

Emergent microworlds

This microworld was used in the context of in-service teacher education courses on the use of technology for mathematics education (Kynigos, in press). An equivalent microworld was built as a result of debate and reflection of these teachers, consisting of a procedure for an arc and its relationship to the corresponding cord of the respective circle. Furthermore, the microworld was used as a starting point of a PhD thesis on the concept of periodicity of trigonometric functions, where the right triangle was transformed into the shape of a clown (Kynigos and Gavriliis, 2006).

Heron's Automata

Represented Phenomenon

An ancient Greek temple is the central feature of the microworld. A priest uses torches to ignite Heron's technology, an underground boiler creating the energy to open the temple gates. He can use one to six torches enhancing the angle at which the gate opens. The user can either choose the number of torches or navigate around and inside the temple with six arrow and letter keys, one for each direction on the three axes of the 3D space. The user can also navigate underground to observe a representation of the boiler technology. The microworld is constructed in MachineLab.

Level of abstraction

The representation is concrete since there are no visible mathematical representations. However, navigation of the camera in the three dimensions is done through abstract means, i.e. arrow keys plus a random couple of other keys.



Concepts embedded

The concepts which were primarily meant to be embedded were to do with the functioning of Heron's technology and the history of the times.

User Interface

Description of the scene

The scene consists of the temple and the priest in front of the gate. The only thing that can be put to motion out of the objects on the scene is the gates. The user can navigate below the ground to observe the technology for opening the gate.

Manipulation

Only two things can be manipulated. The positioning of the camera which can be moved in a continuous sense through the navigation keys and the number of torches used by the priest which requires a selection of an equivalent number of keys.

Programmability

This microworld was not originally meant to leave programmability open to students or teachers. Logo was used to program the basic functionalities of navigation and selection of number of torches. This was done so that a Logo literate teacher could communicate with a historian and a developer. The historian was not aware of the learning potential of interactivity before engagement with this project. However, a study was carried out with students who used the Logo code to set the camera at specific points in space experimenting with the Cartesian coordinate inputs.

Feedback from the computer

When changing the parameters

Parameters in this case were the number of torches to open the gates. Feedback is in the form of the gates slowly opening at respectively different angles. Also, moving the camera provides a sense of navigation around and inside the temple.

As result from the experimentation

There is very limited scope for experiments with the number of torches. However, students unexpectedly found navigation fascinating, trying to figure out and control moving the camera.

Learning process

The design of the learning process turned out to be an issue for debate and communication between the historian (not history educationalist) and the teacher – mathematics educator. What was important to the former was the historical accuracy of the phenomenon and the temple itself. A lot of effort was spent on getting the 3D asset right according to historical sources. The mathematician on the other hand was interested on issues of mathematical relationships between the magnitude of heat and the energy created by the boiler. In the end he felt that the ground for mathematical exploration was minimal.

Didactical design

Again there were two visions of didactical design. One involved a revelatory approach to learning and an unquestionable role for the technology. The other a facilitator role for the teacher with students carrying out experimentation and possibly questioning the relationships between measure of heat and energy.

Type of representations used

The representations were in the form of a realistic simulation of a phenomenon. There were no mathematical representations apart from the underlying Logo code mainly useable with respect to the navigation of the camera.

Procedure of development

The development involved an organization with know-how on the creation of 3D assets and on historical information. These people collaborated with a developer who created some Logo commands on top of the Delphi interface for programming the MachineLab game engine. The development team also consisted of a researcher with knowledge in the design of Logo-like microworlds and the two mathematics teachers with knowledge of Logo and microworld pedagogies. As you can imagine communication was extremely difficult and the result seemed limited to all.

Emergent microworlds

This first version of MachineLab was built to see whether it was a workable strategy to add a Logo programming interface to realistic game engines. Our black and white box approach was looking for means to have a rich 3D environment as a black box basis for Logo-like microworlds. However, the Automata microworld although adequate for the project at hand, made us realize that without a mechanism to create mathematical representations and combine them dynamically with the 3D simulations, we could only achieve limited interactivity which would be of interest for the potential for students to generate meanings through the use of the microworlds. We thus used funding from subsequent projects to re-build MachineLab and port our Logo language onto a Java interface for the game engine. This enabled us to create mathematical representations such as variation tools and programmable 3D objects such as a turtle or a rope-like trace for its displacements. A full turtle geometry environment can thus be integrated with a 3D virtual reality world including geometrical or any other objects.

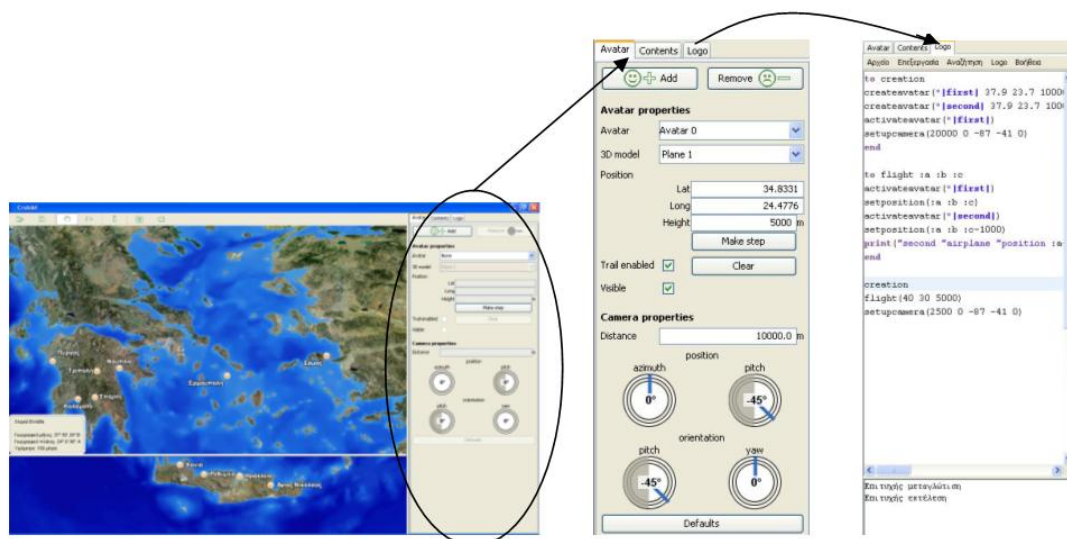
Navigational Mathematics

Represented Phenomenon

The user is in the role of giving directions for the creation and displacement of airplane avatars on a 3D geographical information system containing maps with geo-coded information. Displacements can be carried out either through cartesian (lat, long, height) or vector (angle r , angle f , length r) coordinates. They can also be done declaratively by providing the name of a location. All of these ways are dynamically linked so that a change in an input from one system creates changes in the inputs of the other and vice versa.

Level of abstraction

The geographical space and the plane avatars are pretty concrete. Displacements however are done through mathematical representations of geometrical positions of a Mercator projection of the earth's elliptical shape. These displacements are done either through an interface where the values are put in for each displacement followed by the pushing of a 'go' button or by means of a Logo language with primitives for these displacements and for addressing avatar names.



Concepts embedded

Orientation and spatial awareness is one area of concepts. Cartesian coordinates and vector displacements are another, leading to the notion of geometrical systems and their link to analytical algebra.

User Interface

Description of the scene

The scene is based on 'Cruiser', a GoogleEarth-like software originally addressed to other domains such as fleet management, publishing, real-estate and public information platform for municipalities. The 'Cruislet' microworld has the mathematical interface for avatar displacements and also camera displacement in relation to the avatars. It also has the Logo language for programmable avatar trips. At the moment there are maps of Greece and Cyprus but that is changing as more geo-coded information is added onto the Cruiser platform. The usual geographical data is contained as well as the possibility for users to add their own.

Manipulation

Manipulation is indirect. It comes in the form of avatar creation and displacement through either the special user interface for inputs to displacement values or through Logo.

Programmability

A full Logo language is available with special primitives for avatar creation, addressing, coordinate, vector and geographical displacements.

Feedback from the computer

When changing the parameters

Values to coordinates or vector angles and length are input. Pressing a 'go' button causes the change.

As result from the experimentation

A large variety of experiments are of course possible by means of predicting where an avatar will go when a value or a combination of values are input.

Learning process

Users can get the avatars to make specific displacements or sequences of displacements to create a trip. These can be done in any direction in 3D space. The view of the avatar also changes with respect to the position of the camera following it. This provides a significant freedom for activities and interests. Activities can also involve more than one avatar and the user can build dependencies on avatar movements using Logo programs. All these activities can be done by choosing to use the coordinate or the vector system and by inevitably making comparisons between the two. It's like using the London underground, you get used to thinking in absolute or relative coordinates.

Didactical design

Didactical engineering will determine when and how a teacher might intervene to help the students make sense and articulate how the two systems work and how they can be made use of to determine locations and displacements. There is a variety of social orchestrations possible here from competitive (guess a flight given by the teacher) to game-like (guess my flight) to creative (create a flight and collect information on the way).

Type of representations used

The representations for displacements and the Logo language are executable formal mathematics. The displacements resulting from executions are mathematical in the sense that they are strictly defined on the Mercator projection. On the other hand they are realistic since the terrain is a 3D map with a lot of multimedia information on specific locations and the possibility for inserting your own.

Procedure of development

Developing this microworld required the negotiation of agendas of a research team and a corporate organization. Could this microworld become the basis for a future commercial application as an add-on to Cruiser? Could it at the same time be a platform to build freely available mathematical games? Researchers and developers from two organizations collaborated in its development. They had had years of experience in such collaborations and this made it possible for the software to be developed on schedule.

Emergent microworlds

Cruislet is a platform rather than a microworld in the sense that it has a very broad scope for activities. The process of using it to design more specific microworlds has just begun. The first one is based on the game 'guess my flight' where an avatar's displacements are dependent on another ones. The students explore the dependency and when they discover it play a game of creating dependencies for their peers to investigate. I'm sure many more microworlds will come.

Conclusions

Four half-baked microworlds were described, the first two were designed as such, the latter two were used as half-baked in that they provoked the design of other emerging microworlds. The breadth of technologies point to that the notion of half-baked microworlds is not technology dependent, it's rather a construct to think about the design and use of malleable educational software. What is in discussion here however is the template for describing such microworlds. This may look rather normal to the Logo community. However, it was the result of negotiation between teams within and outside this community and used as a tool to communicate design ideas for learning environments. It therefore portrays the kinds of things that communities outside our own may consider important in understanding the essence and function of microworlds in educational environments. I'm sure that given another configuration of teams, we would see a different, perhaps more elaborate such template. What's important however is to engage in the process of constructing a language to communicate with broader educational communities, now that technologies are becoming widely available and used. After 40 years, integration of Logo-like ideas and technologies in wider educational communities is proving essential for their presence in education beyond a recluse club of equally ageing enthusiasts.

Acknowledgements

'ESCALATE' Enhancing SCience Appeal in Learning through Argumentative inTEraction FP6-2004-Science-and-Society-11, 020790 (2006-2008).

'ReMath' - Representing Mathematics with Digital Media FP6, IST-4, STREP 026751 (2005 – 2008).

'Greekworks': Interactive tools for presentation and study of Ancient Greek Technologies. Ministry of Development, General Secretariat for Research and Technology, R&D Consortia in sectors of national priority, Action 4.5.1., P1 (2003-2005)

'SEED' - 'Seeding cultural change in the school system through the generation of communities engaged in integrated educational and technological innovation', European Community, IST, School of Tomorrow, IST-2000-25214.

References

- Agalianos A. S. (1997). *A Cultural Studies Analysis of Logo in Education*, unpublished doctoral thesis, Institute of Education, Policy Studies & Mathematical Sciences, London.
- Balacheff, N., Kaput, J.J.: 1996, "Computer-Based Learning Environments in Mathematics". In Bishop, A.J. et al., *International Handbook of mathematics education* (pp. 469 - 501). Kluwer Academic Publishers. Dordrecht, Netherlands.
- Bereiter, C., & Scardamalia, M. (2003). Learning to work creatively with knowledge. In E. De Corte, L. Verschaffel, N. Entwistle, & J. van Merri (Eds.), *Unraveling basic components and dimensions of powerful learning environments*. EARLI Advances in Learning and Instruction Series; Retrieved from <http://ikit.org/fulltext/inresslearning.pdf>
- Blikstein, P. & Cavallo, D. (2002). Technology as a Trojan Horse in School Environments: The Emergence of the Learning Atmosphere (II). In *Proceedings of the Interactive Computer Aided Learning International Workshop*, Carinthia Technology Institute, Villach, Austria.
- Clements, D., & Sarama, J. (1997). Research on Logo: a Decade of Progress. *Computers in the Schools*, 14(1-2), 9-46.
- Cobb, P. & Yackel, E. (1996). Constructivist, Emergent and Sociocultural Perspectives in the Context of Developmental Research. *Journal of Educational Psychology*, Vol. 31, 175 – 190

- diSessa, A. (1997). Open Toolsets: New Ends and new Means in Learning Mathematics and Science with Computers. In E. Pehkonen (Ed.), *Proceedings of the 21st Conference of the International Group for the Psychology of Mathematics Education*, 1. Lahti, Finland, 47-62.
- diSessa, A. (2000). Changing minds, computers, learning and literacy. Cambridge, MA: MIT Press.
- diSessa, A. A., Azevedo, F. S. & Parnafes, O. (2004). Issues in component computing: A synthetic review. *Interactive Learning Environments*: 12, 109-159.
- Edwards, L. D. (1995). Microworlds as Representations. In A. diSessa, C. Hoyles & R. Noss (Eds.) *Computers and Exploratory Learning*, 127-154. Berlin/Heidelberg: Springer-Verlag.
- Guin, D. & Trouche, L. (1999). The Complex Process of Converting Tools Into Mathematical Instruments: The Case of Calculators. *International Journal of Computers for Mathematical Learning* 3(3): 195–227.
- Hoyles C., Noss R. and Adamson R. (2002) 'Rethinking the Microworld Idea'. *Journal of Educational Computing Research, Special issue on: Microworlds in mathematics education*. 27, 1&2, 29-53.
- Kafai Y., Resnick M. (Eds) (1996) *Constructionism in Practice*, Lawrence Erlbaum Associates Publishers, Mahwah, New Jersey.
- Kynigos, C. (1995). Programming as a Means of Expressing and Exploring Ideas in a Directive Educational System: Three Case Studies. *Computers and Exploratory Learning*, diSessa, A, Hoyles, C. and Noss, R. (eds), Springer Verlag NATO ASI Series, 399-420.
- Kynigos, C. (2002). Generating Cultures for Mathematical Microworld Development in a Multi-Organisational Context. *Journal of Educational Computing Research*, 27 (1 -2), 185-211.
- Kynigos, C. (2004). Black and White Box Approach to User Empowerment with Component Computing, *Interactive Learning Environments*, Carfax Pubs, Taylor and Francis Group, 12(1–2), 27–71.
- Kynigos, C. (2007, in press) Half-baked Microworlds in use in Challenging Teacher Educators' Knowing, *international Journal of Computers for Mathematical Learning*. Kluwer Academic Publishers, Netherlands.
- Kynigos, C. and Gavrilis, S. (2006) Constructing A Sinusoidal Periodic Covariation. *Proceedings of the 30th Conference of the International Group for the Psychology of Education 4-9-16*, Novotna J., Moraova H., Kratka Mm. & Stehlikova N. (Eds), Charles University, Faculty of Education, Prague.*
- Kynigos, C., Yiannoutsou, N., Alexopoulou, E. & Kontogiannis, C. (2006), A half baked Juggler game in use, *Proceedings of the 1st World Conference for Fun 'n Games*, Preston, England, 13-19. © 2005 Child Computer Interaction (ChiCI) Group.
- Laborde, C., Kynigos, C., Hollebrands, K. and Strasser, R. (2006) Teaching and Learning Geometry with Technology, *Handbook of Research on the Psychology of Mathematics Education: Past, Present and Future*, A. Gutiérrez, P. Boero (eds.), 275–304, Sense Publishers.
- Noss R & Hoyles C (1996). *Windows on Mathematical Meanings*, Kluwer academic Publishers
- Noss, R. (1992). The Social Shaping of Computing in Mathematics Education. In: Pimm D. & Love E. (Eds), *The Teaching and Learning of School Mathematics*. Hodder & Stoughton.
- Papert, (2002). The Turtle's Long Slow Trip: Macro-Educological Perspectives on Microworlds. *Journal of Educational Computing Research*, 27(1), 7-28.
- Papert, S. 1980. *Mindstorms: Children, computers and powerful ideas*. New York, NewYork: Basic Books.

Penner, D. (2001). Cognition, Computers and Synthetic Science: Building Knowledge and Meaning Through Modelling, in Secade, W., (2001) (Ed), *Review of Research in Education*, Washington, American Educational Research Association.

Rabardel, P.: (2001). 'Instrument Mediated Activity in Situations', in A. Blandford, J. Vanderdonckt & P. Gray (eds.), *People and Computers XV - Interactions Without Frontiers*, Springer-Verlag, Berlin, 17-30.

Sarama, J., & Clements, D. (2002). Design of Microworlds in Mathematics and Science Education, *Journal of Educational Computing Research*. 27(1), 1-3.

Vergnaud, G. (1991). La Théorie des Champs Conceptuels. *Recherches en didactique des mathématiques*. 10(2/3) 133-169.

Turtle's navigation and manipulation of geometrical figures constructed by variable processes in a 3d simulated space

Chronis Kynigos, *kynigos@ppp.uoa.gr*

Educational Technology Lab, School of Philosophy, University of Athens

Maria Latsi, *mlatsi@ppp.uoa.gr*

Educational Technology Lab, School of Philosophy, University of Athens

Abstract

Issues related to 3d turtle's navigation and geometrical figures' manipulation in the simulated 3d space of a newly developed computational environment, MaLT, are reported and discussed here. The joint use of meaningful formalism and the dynamic manipulation of graphically represented 3d figures seem to offer new resources and to pose new challenges as far as geometrical activities and construction of meanings are concerned, which are strongly related to the representational infrastructure of MaLT.

Abilities such as spatial orientation and spatial visualisation come into play and are interwoven with the software's functionalities and semantics. Although the body-syntonic metaphor remains critical while navigating the turtle in the 3d simulated space, it seems that it has to be co-ordinated with other –often conflicting one another- frames of reference. The strong link between spatial graphical and geometrical aspects, that was accentuated by the dragging functionalities of the software, helped students go beyond an immediate perceptual approach, relating geometrical figures with real 3d objects and the change of planes in 3d space with physical angle situations. In this framework the concept of angle as turn and measure with emphasis on directionality but also as a relationship between the planes defined by 2d figures has arisen as central.

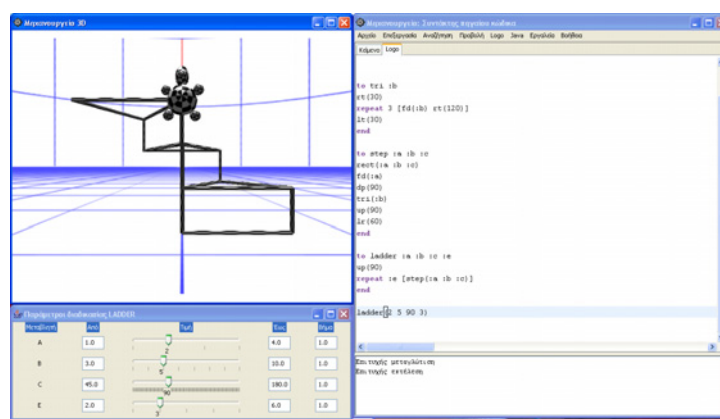


Figure 1. The interface of MaLT

Keywords

3d logo, dynamic manipulation, spatial visualisation, frames of reference, angle

Theoretical Background

In this paper we report a small-scale preliminary research aiming at shedding light upon issues related to turtle's navigation and geometrical figures manipulation that have arisen while 11 year-old students were carrying out collaboratively constructionist activities in the computational environment of a newly developed software, MaLT. MaLT was conceived as a constructionist microworld environment (Papert, 1980, Harel & Papert, 1991, Kafai & Resnick, 1996) for programmable constructions in 3d geometrical space. It is designed so as to extend the integrated 'programming-dynamic manipulation' (Kynigos, 2004) idea to 3d space, while most ICT tools (logo-based environments, DGEs) have been designed to operate within 2d environments.

The new kinds of representations offered (e.g. 3d turtle in a simulated 3d space), the new kinds of access offered to conventional representations (e.g. dynamic manipulation of the values of the variables of parametric procedures) and the way these changes affect the symbolic activity and the construction of mathematical meanings need investigation. So far the pedagogical value of graphical representations has been the object of big debate in the teaching of Geometry, since it is not clear exactly what a geometrical shape represents, an actual tangible figure or a class of ideal, but not real, objects. This is at the heart of geometry's dual nature as a field for abstract mathematical thinking or for understanding space (Laborde et al, 2006). As far as geometry curriculum is concerned, in primary education emphasis is placed on formal symbolism and naming and not enough on spatial exploration, analysis and synthesis. Students learn of 3d geometry through presentation of static 3d objects, either in a concrete form or in printed form in their textbooks, without the chance of manipulating them.

Constructions in the context of MaLT could possibly bring in the foreground issues concerning the mathematical nature of 3d geometrical objects and how interactivity, control and experimentation in virtual reality microworlds can be a versatile vehicle for enhancing mathematically driven navigation, orientation and spatial visualization. Studying in a dynamic way 3d geometrical objects students have to analyse a 3d figure, break it into smaller parts and determine angle measures and lengths of line segments. Projecting themselves into the place of the turtle and moving from the visual to the descriptive level of thought students have to search for ways to reconceptualise 3d objects in terms that can be explained to the 3d turtle through logo commands. Moreover through the use of sliders students are provided with a direct manipulation metaphor for sequentially changing variables' values and simultaneously observing the variation both of 3d object and of their place in 3d space. As a result students have the chance to observe the behaviour of the varying parts in relation to each other and to the invariant ones and acquire a sense of generality and abstraction.

While interacting with each other and with the computational medium students come in contact with multiple linked, integrated and interdependent mathematical representations. In particular students come in contact with three representational registers: a) meaningful formalism by means of symbolic programming b) graphical-figural representations in the simulated 3d space c) kinaesthetic activity and use of various interwoven but in many cases conflicting one another frames of reference (Wickens & all, 2005). However an instrument does not exist in itself but only in relation to its utilisation by the user (Verillon and Rabardel, 1995, Mariotti, 2001). As a result an instrument is not a fixed and permanent object but rather an evolving and versatile matter. For instance, in MaLT functionalities and phenomenological cues available are chosen by experts because they see on them an embodiment of something that they already know and understand (Kynigos & Latsi, 2006). The specific representations are media for what the experts already know. On the other hand for students these are only presentational models that may be used and conceptualized in quite a different way than the initially intended one. The construction of mathematical meanings as a result of students' interaction with a specific learning environment raises (among the others) semiotic issues related to screen phenomenology and semantics. Thus a careful analysis of the utilisation schemes developed and representational models used is needed so as to investigate the interplay between mathematical meanings and

computer phenomenology. The interplay of the aforementioned representational registers in the process of knowledge construction viewed from an interactionist perspective will be the focus of our research interest.

Research setting and Method

The technology used

Machine-Lab is a programmable environment for the creation and exploration of interactive virtual reality simulations developed within the ReMath project. Machine-Lab is implemented over the 3impact game engine (www.3impact.com) which wraps the stunning 3D graphics of the Direct3D engine and the distributed/multi-user capabilities of the DirectPlay engine, together with a 3D sound engine (with Doppler effects) and the Open Dynamics Engine (www.ode.org), which is a Rigid-Body Simulation engine with realistic physics support for 'collidable' objects and articulated bodies (rag-dolls / humanoids). The version of MaLT that was used consisted of three components namely:

- *Turtle Scene (TS)*. The main part of this component is a three-dimensional grid-like interface. A perspective projection is used with three active vanishing points, so that a realistic effect of 3d space representation and navigation is created. The 3d space representation is based on the metaphor of hemisphere. The two active vanishing points have a deviation of 90° and they are located in an iconic line of horizon which is conceived as the circumference of vertical circle. The third active vanishing point is conceived as the pole of the hemisphere. The turtle, which is a 3d object of .x type, is java-based and compiled so that the response from 3d graphics is fast. It can be navigated in the 3d simulated space providing rich representations of geometric objects and behaviours.
- *Logo Editor (LE)*. This component is the Logo-like programming interface that engages students in logical procedural thinking and it is linked to the Turtle Scene. The user is able to write, run and edit Logo programs.
- *Uni-dimensional Variation Tool (1dVT)*. The main part of this component consists of 'number-line'-like sliders, each corresponding to one of the variables used in a Logo procedure.

Research Context

The research was carried out at the computer laboratory of a primary school in Athens with four sixth grade students divided in two groups. It should be noted that students didn't have any experience with any programming language or with any dynamic geometry environment either 3d or 2d. Two tasks were carried out by students: in the first task, which was introductory, students were firstly asked to move the turtle in the right and left corner of the 3d space and then to bring it back at its initial position. The researcher didn't specify on purpose what they meant with the left and right corner, so as to leave students explore freely the 3d space and develop their own 'navigation' strategies. In the second task students were asked to develop a parametric procedure that constructs a parallelogram and then to change its position in 3d space using the 1d Variation Tool.

Research Method

In our research we used a design-based research method (Cobb & all, 2003) which entailed the 'engineering' of tools and task, as well as the systematic study of the forms of learning that took place within the specific context as defined by the means of supporting it. Initially the aim was to develop the software and the task in order to improve the educational process and to bring about new forms of learning, based on prior research and our theoretical framework. In retrospect, the research method that we used aims both at improving the initial design and at resulting in a situated understanding of the relationship among theory, designed artifacts and practice.

Espousing the role of naïve and participant observer the researchers took a generative stance allowing for the data to structure the results and to pilot their analysis. It should be stressed that we are not aiming at final and self – evident answers but at focusing better and at reformulating our initial questions. A team of two researchers participated in each data collection session using one camera and two tape-recorders. One researcher was occasionally moving the camera to all groups to capture the overall activity and other significant details as they occurred. Background data were collected (e. g. students worksheets, observational notes) and all audio-recordings were analyzed verbatim.

In analyzing the data we firstly looked for instances where meanings related to the visualisation and conceptualisation of the computer simulated 3d geometrical space were expressed by the students. The identified illustrative episodes can be defined as moments in time which have characteristic bearing on pupil's interaction with the available tools and mathematical meanings construction. We have focused particularly on the process by which implicit mathematical knowledge is constructed during shared student activity while being sensitive to the interconnection between the construction of personal meanings and standard mathematical discourse. However in certain cases word episodes were meaningless if they were not related to the sequences of actions that students carried out while constructing their procedures or more importantly if they were not related to the gestures that they used. Thus in the analysis that follows word episodes are accompanied by photos with indicative students' gestures.

Results

Moving around in the 3d space

Moving the turtle around in the 3d simulated space of MaLT seemed to necessitate a dynamic visualisation of angle, which integrates two schemes:

- turn as body movement in 3d space which inevitably involves directionality and
- turn as number - measure

It should be noted that in MaLT there are 3 kind of turns: right/left turn in relation to turtle's trunk-vertical axis, rightroll/leftroll which moves the turtle around its trunk/vertical axis and uppitch/downpitch, which pitches the turtle's nose up and down. The multifaceted nature of the concept of angle as well as the difficulties that children encounter in various contexts -and while working with 2d Logo- are well documented in the literature (Clements & all, 1996, Keiser, 2004, Micheltmore & White, 2000). Although in first thought it seems that in this kind of computational environment the concept of angle is approached in it's a more natural and intuitive aspect, analysis of our results showed that new kinds of challenges arise while navigating a 3d turtle in the 3d simulated space.

St1: No, as it is from here, it turns that way and goes left.

St2: It goes that way, there is left.

St1: With the face of the turtle... it should come here.

St2: So left, look at it a bit.

St1: Should I correct it..Rt..we proceeded below.

From the above extract, as it was illuminated by students gestures, it was evident that students oscillated between different frames of reference (Wickens & all, 2005), when they were trying to visualize 3d turtle's motion on the computer screen, either in order to decide what commands they should give or in order to find out what has caused the unexpected turtle's movement. The frames of reference used by students could be categorized in five groups:

- A world frame: defined in terms of directions 'up' and 'down'
- An ego frame: defined in terms of the orientation of the trunk or location of the observer
- A head frame: defined in terms of the orientation of the head
- A vehicle frame: typically associated with the orientation of a moving entity, here the turtle.
- A display frame, defining the orientation and movement of information on a display

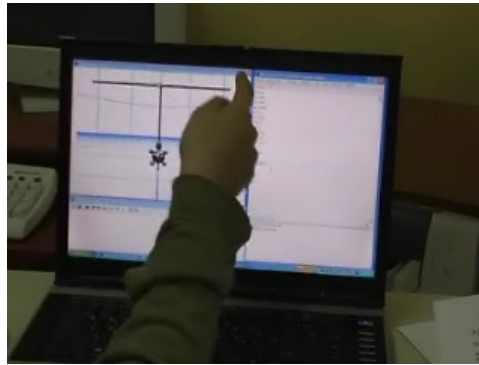


Figure 1. Showing at the screen using a display frame of reference

All the above frames of reference were intermingled when students were trying to find out the right kind of turn. Initially an 'ego-frame' was projected to a 'vehicle-frame'. In other words children drove the turtle in a body-syntonic way, projecting the orientation of their trunk to the orientation of the vehicle of the motion, the turtle. Moreover the standard 'display – frame', the computer screen, was interconnected with the 'world-frame' and the 'ego-frame' and was defined in relation to four points, the standard up/down direction and right/left, in relation to the viewing point of the observer. In the 3d motion of the turtle another frame of reference was also present, which we can call the 'head-viewing' frame. In literature the head-frame is defined in terms of the orientation of the head, which may sometimes be different from that of the ego-trunk. While moving in the 3d turtle-scene, in order to understand where right, left, up and down is, attention should be given to '*where the turtle was looking at*' or '*where the turtle's face was*', to use children's expressions. So, many times students had to explain to each other where right was, using simultaneously their hands for right and left, moving their bodies around its axis to imitate turtle's motion and then showing on the screen. It follows that students had to co-ordinate all those frames in order to visualise turtle's motion. It should be also noted that all these frames often conflict one another. For instance, right could be defined in relation to the display screen and the standard viewing point of the observer (display frame) or the turtle's trunk axis (ego-vehicle frame), or the turtle's view-point (head-viewing frame).

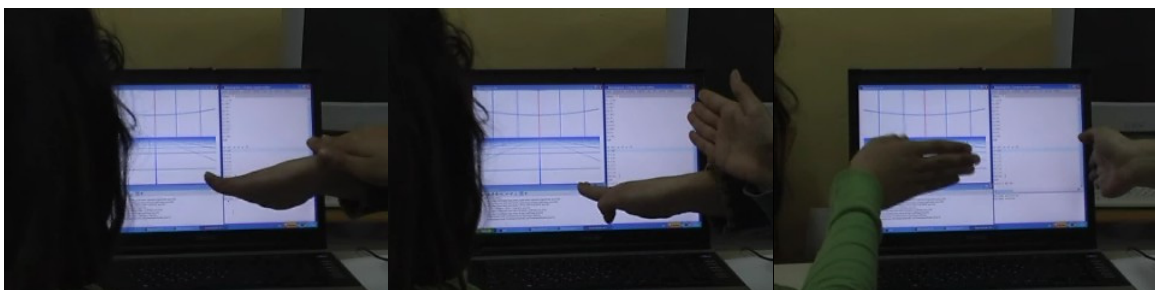


Figure 2. Using hands so as to project an ego-frame to the vehicle-frame of reference

Moreover it could be added that, although 3d simulated space is closer to real life and every-day experiences, the body-syntonic metaphor is less strong in 3d turtle geometry than in 2d. In other words when we move in real 3d space the up and down directions are usually stable (although

not when we turn upside-down) because of gravity. Moreover, we walk in a 2d horizontal plane while the 3d turtle moves in different planes in 3d space. For instance, we can easily simulate 2d turtle motion with our body but we cannot simulate 3d turtle's motion. Thus, it seems that the body-syntonic frame in 3d space is shrunk in favour of the 'vehicle frame'.

Another finding, relative to students' difficulty in navigating in 3d space using the 3d body-syntonic metaphor, is their preference in working initially in the vertical plane using only the four standard commands of 2d Logo (right, left, forward, backward). As shown in the following word episode their navigation in the vertical plane seemed more convenient, closer to their point of view and quicker. Initially they didn't find any utility in having the turtle moving around its axis (rightroll or leftroll command) or pitching its nose up and down (uppitch and downpitch commands). Only after more experimentation did they realise that by rolling the turtle around its axis the position of right and left changes accordingly. Their preference in working on the vertical plane could be possibly explained by the fact that children were more accustomed to 2D representations of geometrical figures. Three-dimensional representations on the computer screen are quite new and possibly students are not yet accustomed to the conventions used so as to represent a 3D object in a 2D form to look 3D in nature (Lowrie, 2002). On the other hand if we accept the view of Dalgarno & al. (2002) that we understand 3D models through multiple 2D representations, maybe students had focused subconsciously on a simplified 2D way of visualising 3d space.

R: I see that you use more the right and forward commands, don't you?

St1: Yes.

R: Is it easier? Why don't you use the other commands? You didn't need them?

St1: No, we didn't need them.

St2: It's closer and more convenient. You can see easier the turtle.

St2: Why should we have the turtle moving around itself? There is no point in it.

R: There is no point in moving around herself?

St: No, it manages everything this way and it's easier.

It could be also said that in order to visualise turtle's motion and find concrete analogues in real space children used two perspectives: an intrinsic- internal one and an external one. An intrinsic-internal one was used in cases that children were trying to co-ordinate their ego-frame with the vehicle frame and thus to view the surrounding environment from inside, as the space immediately surrounding them, while trying to conceptualise it in three dimensions constructed out of the axes of their body (Tversky, 2005). In these cases they used their hands and bodies in order to concretize turtle's motion. However when they were manipulating the figures created by the turtle they adopted an external view, they viewed those objects from outside and they sometimes needed concrete 3d objects, as transitional ones between real space and simulated 3d space. In fact, in order to visualise the change of planes of a parallelogram as a result of the change of the initial position of turtle in 3d space, children didn't use only their bodies or their hands but also concrete 3d objects, in our case a cassette that they moved in 3d space.

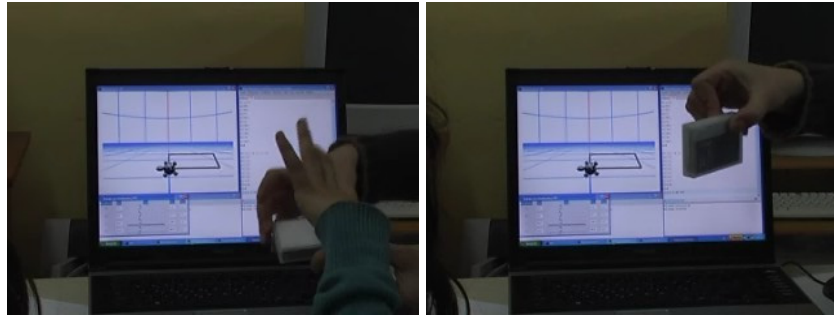


Figure 3. Using a concrete 3d object to help them foresee the changes of planes of the parallelogram in the screen

As far as the aspect of angle as number-measure of a turn is concerned, it seemed that students could easily foresee the degree of change of turtle's position and of the change of plane of the parallelogram drawn by the 3d turtle using as benchmarks the standard measures of 180, 90 and 45 degrees, drawing apparently upon their experience in 2d geometry in their mathematics course, a finding which is in accordance to other researches with 2d Logo environments (Clements & Burns, 2000).

R: So if I delete this downpitch command, and run the procedure 'parallelogram' what is going to happen?

St1: It is going to be flat.

St2: Yes

R: Flat, what has happened with the downpitch command?

St2: It has raised it that way.

R: That way.

St1: It has descended it down.

R: If we command downpitch 45, how do you think it will appear?

St1: Somewhere in the middle.

R: Somewhere in the middle. If we input 90?

St2: It is going to appear that way.

R: If we input 50?

St1: It will be not in the middle but somewhere above the middle.

Semantics of the software

The aforementioned results cannot be analysed in all their dimensions independently from the tool set with which students' construction strategies are expressed. In particular here we are going to try to explain students' visualisation of the simulated 3d space and construction strategies relating them to software's semantics and affordances.

Although construction in one plane is a meaningful choice as it minimizes the variables involved (children use only the standard 2d logo commands), it is still to be explained why children preferred to construct their figures in the vertical level and not on the horizontal one which is closer to the 'flat earth model' (Driver & all, 1994) that we experience everyday. Working with the software we realised that the vertical axis that students preferred coincides with the viewing axis of the display. As a result and using children's words from a previously presented word episode, working in this level, '*you can see easier the turtle*'. This finding is in accordance to other researches conducted with 3d computational tools (Kynigos & Latsi, 2006) in relation to a 3D representation of vectors in computer screen, which have also shown that children have focused in a simplified 2D representation and that they had extra difficulties in appreciating the third dimension concerning depth (z coordinate).

While designing the tool we were sensitive to the difficulties present in trying to represent a 3d object in a 2d form to look 3d in nature and we inserted a grid in the 'Turtle Scene' as a visual hint to help students appreciate depth. However it seems that it wasn't helpful enough as far as the third dimension is concerned, a point which should be taken into account in a further development of the software. To make matter worse, in cases where the trace of the turtle coincided with the lines of the grid the appreciation of the depth was more complicated. Moreover the fact that the grid followed the convention of the linear perspective and of a vanishing point visible in the line of horizon confused students as far as the course of the turtle in the z coordinate was concerned. Did the turtle follow a perpendicular route to the vertical level (called by the students as straight line in the following excerpt) or did it follow a slanted one?

St2: With the depth, I don't know. Should I input 5?

St1: Yes, do it.

R: Where is it? Is its trace obvious?

St1: No, it is more intense. It went inside, it didn't go straight.

R: Oh

St1: If this line weren't there, it would be visible.

Finally the affordance of manipulating dynamically and in a kinaesthetic way the values of the variables of their procedure offered students the opportunity to animate their constructions, while connecting the visual features of their constructions to their mathematical underpinnings. Moving their constructions around in 3d space and changing planes helped students acquire a sense of generality and abstraction underlying the mathematical concepts involved -such as the notion of angle in 3d space- which are usually presented in a static way. Angle was conceived not only as the turn of the turtle and its measure but also as a relationship between the planes defined by the 2d parallelogram drawn by the turtle. In the following excerpt it is evident that students could easily move their parallelogram in any direction and find its analogue to everyday physical 3d objects. It is interesting that children did not only connect their construction with a physical 3d object but also with physical angle situations, as the limited rotation of a door, which usually involves extra difficulties for students. As Micheltore & White (2000) points out, while opening and closing a door the main difficulty lay in identifying its initial position as one arm of the angle, as the standard angle concept first develops in situations where both arms of the angle are visible.

St2: It's like a door.

R: Like a door? How can we have it upright?

St2: I will move it vertically.

R: So you made it a door. Do you want it to turn this way?

St1: No, vertically... or this way.

Discussion

New representations enabled by MaLT can place spatial visualization concepts in a central role for both controlling and measuring the behaviours of entities and figures in virtual 3d environments. We find the 'distance' between this new representations and the conventional, static ones as a challenging point to provoke unexpected pupil's responses while giving in parallel rise to new kinds of problems requiring geometrical knowledge. Three-dimensionality in the computational environment of MaLT poses special challenges concerning abilities such as

spatial navigation and spatial visualisation which seem to be interwoven with the use of various frames of reference and mathematical formalism.

In particular, while navigating the 3d turtle in the 3d space of MaLt or when manipulating dynamically geometrical figures it seems that the concept of angle as turn and measure with emphasis on directionality is central. Angle was approached more as a turn that is neither the static pair of sides nor the enclosed planar or spatial part but the process of change of direction and planes in 3d space. Moving around in 3d space involves the coordination of various frames of reference (Wickens & all, 2005) which in many cases conflict one-another. Although the notion of body-syntonicity is critical, it is evident that it is intermingled with the 'display' frame of reference, the 'vehicle' frame of reference and the 'head- face' one. In other words in order to navigate in the 3d simulated space students have to co-ordinate their static view point of the 2d computer screen while projecting their knowledge about their bodies and how they move into the 'vehicle' of the motion, the turtle, whose heading-face has a special importance (In 2d turtle's motion attention is paid only to turtle's heading, while here attention should be given to turtle's 'face', as the turtle can be rolled around its vertical axis). Moreover the 'world' frame of reference where the up and down are fixed as well as the fact that we move in a horizontal 2d plane are challenged and 'the body-syntonicity' metaphor has to be applied in a 3d turtle that can change planes and orientation without any limitation.

As a result students seemed to use two perspectives while they were trying to visualise turtle's motion: an external view perspective and an internal one. The first perspective was used in cases that they were trying to navigate the turtle in 3d space and to see the 3d space from 'inside' conceptualising it in three dimensions constructed out of the axes of their bodies. In these cases they used their hands and bodies so as to concretize turtle's motion. It should be noted that in contrast to researches conducted with 2d logo we didn't observe a curtailment of physical strategies (Clements & Burns, 2000). Students didn't proceed from large body movements to smaller ones, using initially their body and then their hands to help them visualise turtle's motion. They used their hands to concretize the standard 2d Logo commands, and especially right and left, and their trunk in order to concretize the rightroll and leftroll commands. It is interesting that they didn't use any body-movement to imitate the uppitch/downpitch commands, probably because pitching our nose up and down is not followed by movement of our trunk as it happens with 3d turtle. The 'external' view perspective was used in order to visualise the change of planes of a parallelogram as a result of the change of the initial position of the turtle in 3d space. In this case children viewed this object from outside and they used not their bodies but concrete 3d objects, as transitional ones between real space and simulated 3d space. In this framework an interesting point for a further research could be the integration in task design of an absolute frame of reference where the place of the turtle would be also defined by 3d coordinates, a functionality which is also available in MaLT.

Moving from the visual to the descriptive level of thought students had to search for ways to reconceptualise geometrical figures in terms of 3d logo commands. Translation of visual features into conceptual relations was facilitated by the dragging functionalities of the software that seemed to have acted as a mediator between figures on the one hand and mathematical formalism and concepts on the other (Kynigos, 2004). Dragging didn't provide just a kinesthetic sense of dynamic manipulation and animation of mathematical objects changing only the visual characteristics of the figures. It provided an action/notation context that rendered parametric procedures descriptors of evolving geometrical objects by means of dragging a slider and observing how the object changes in relation to the values of the variables. In this way the distinction between the conceptual part of the notion of angle as turn and measure in 3d space and the figural one was blurred unlike what is usually the case in geometrical tasks (Mariotti, 2001). Initiating from the use of a parallelogram as a class of ideal but not real object students passed to its use as an actual tangible figure, bridging geometry's dual nature (Laborde & all, 2006). In this research a parallelogram was used as an external object that was connected to real-life experiences and whose behaviour and feedback required interpretation by the students. The strong link between spatial graphical and geometrical aspects helped students go beyond

an immediate perceptual approach, conceive angle as turn and measure and connect it with physical angle situations (e. g. the limited rotation of a door) that usually cause special difficulties to young students. Thus turn was viewed not only as a relationship between the headings-faces of the turtle and its traces but also as a relationship between the planes defined by the 2d parallelogram drawn by the turtle.

Student's visualisation and construction strategies are inevitably related to the representational registers used and tool's affordances. This small scale research brought in the foreground issues related to software's semantics. The preference of students in working initially in the 2d vertical level should be related not only to the complexity of 3d space but also to students' viewing axis of the display and the endogenous difficulties of represented 3d objects in the 2d computer screen so as to look 3d. Although certain conventions are used in MaLT, so as to help students appreciate depth (the z co-ordinate), it seemed that they are not enough and that in some cases they can be even confusing. This finding stresses the need for the provision of further visual cues and conceptual hints so as to help students appreciate 3dness.

In this small scale research we formed the impression that MaLT can be a new resource for activity and construction of meanings where the mathematical formalism of 3d geometry and graphical representation of objects and relations can be dynamically jointed in interesting ways, strongly related to the representational infrastructure of MaLT. However further researches are needed where more attention should be given to tasks' design and classroom orchestration so as to investigate in depth and shed light upon the various interconnected mathematical concepts that are involved in 3d space visualisation and conceptualisation.

Acknowledgments

ReMath: Representing Mathematics with Digital Media, Sixth Framework Programme, IST-4, Information Society Technologies, Project Number: IST4-26751, <http://remath.cti.gr>

Thanks to our research assistant S. Kitsou for her participation and help in data collection and analysis.

References

- Clements, D.H., Battista, M.T., Sarama, J. and Swaminathan, S. (1996) '*Development of turn and turn measurement concepts in a computer-based instructional unit*'. Educational Studies in Mathematics 30, 313–337.
- Clements, D. & Burns, B. (2000) *Students' development of Strategies for turn and angle measure*, Educational studies in Mathematics, 41, 31-45
- Cobb P., Confrey J., DiSessa A., Lehrer R., and Schauble L. (2003) *Design Experiments in Educational Research*. Educational Researcher, Vol. 32-1, 9-13.
- Cope, P. and Simmons, M.: 1991, *Children's exploration of rotation and angle in limited Logo microworld*, Computers in Education, 16, 133–141
- Driver R, Squires A., Rushworth P., Wood-Robinson, (1994), *Making Sense of Secondary Science*, London: Routhledge
- Dalgarno, B., Hedberg, J. & Harper B., (2002). *The Contribution of 3D Environments to Conceptual Understanding*. In Proceedings of ASCILITE 2002 Conference. pp 44-54, Auckland, New Zealand
- Goetz, J. P. and LeCompte, M. D. (1984) *Ethnography and Qualitative Design in Educational Research*, Academic Press, London.
- Harel, I. and Papert, S. (eds). (1991) *Constructionism: Research Reports and Essays*, Ablex Publishing Corporation, Norwood, New Jersey.

- Kafai, Y. and Resnick, M. (eds.) (1996). *Constructionism in practice: Designing, thinking and learning in a digital world*. Lawrence Erlbaum Publishers, Mahwah
- Keiser, J. (2004) *Struggles with developing the concept of angle: Comparing sixth-grade students' discourse to the history of angle concept*, Mathematical Thinking and Learning, 6 (3), 285-306
- Kynigos, C. (1995). *Programming as a Means of Expressing and Exploring Ideas in a Directive Educational System: Three Case Studies*. In A diSessa, C Hoyles, & R. Noss (Eds.), Computers and Exploratory Learning, (NATO ASI Series) Heidelberg: Springer-Verlag, 399-420.
- Kynigos, C. (2004). A 'Black-and-White Box' approach to user empowerment with component computing, Interactive Learning Environments, 12 (1-2) 27-71.
- Kynigos, C. & Latsi, M. (2006) *Vectors in use in a 3d Juggling Game Simulation*. International Journal for Technology in Mathematics Education, 13 (1),
- Laborde, C., Kynigos, C., Hollebrands, K. and Strasser, R. (2006) *Teaching and Learning Geometry with Technology*, Handbook of Research on the Psychology of Mathematics Education: Past, Present and Future, A. Gutiérrez, P. Boero (eds.), 275–304, Sense Publishers.
- Lowrie, T. (2002). *The influence of visual and spatial reasoning in interpreting simulated 3D worlds*. International Journal of Computers for Mathematical Learning, 7, 301-318
- Mariotti, M.A. (2001). *Introduction to proof: the mediation of a dynamic software environment*. Educational Studies in Mathematics, 44, 25-53.
- Mitchelmore, M.C & White, (2000) *Development of Angle Concept by Progressive Abstraction and Generalisation*. Educational studies in Mathematics, 41, 209-238
- Papert, S. (1980) *MindStorms – Children, computers and powerful ideas*. The Harvester Press Limited, London
- Tversky, B. (2005), *Functional Significance of Visuospatial Representations*. In The Cambridge Handbook of Visuospatial Thinking, Shah, P. & Miyake A. (eds) (2005), Cambridge University Press, Cambridge, pp. 1-34
- Wickens C., Vincow, M. & Yeh, M. (2005), *Design Applications of Visuospatial Thinking: The Importance of Frame of Reference*. In The Cambridge Handbook of Visuospatial Thinking, Shah, P. & Miyake A. (eds) (2005), Cambridge University Press, Cambridge, pp. 383-425

Introduction to Nondeterminism

Gabriela Lovászová, glovasz@ukf.sk

Dept of Informatics, Constantine the Philosopher University in Nitra, Slovakia

Viera Palmárová, vpalmarova@ukf.sk

Dept of Informatics, Constantine the Philosopher University in Nitra, Slovakia

Abstract

The term of nondeterminism discussed in this paper is usually not included in the introductory programming courses. Secondary school students are familiar with the idea that programs have to be deterministic because they are processed by computers automatically. Later, when they come to the university, it is difficult for them to understand the concept of nondeterminism. In our opinion, the gentle introduction to nondeterminism may be done even at the secondary school level.

A good motivation for introducing the nondeterminism and nondeterministic algorithms might be some logic-based puzzle games. We mean those puzzles, which are really difficult to solve by deterministic algorithms. One needs to use logical reasoning and several strategies and their combination to make right decisions when constructing a solution of such puzzles.

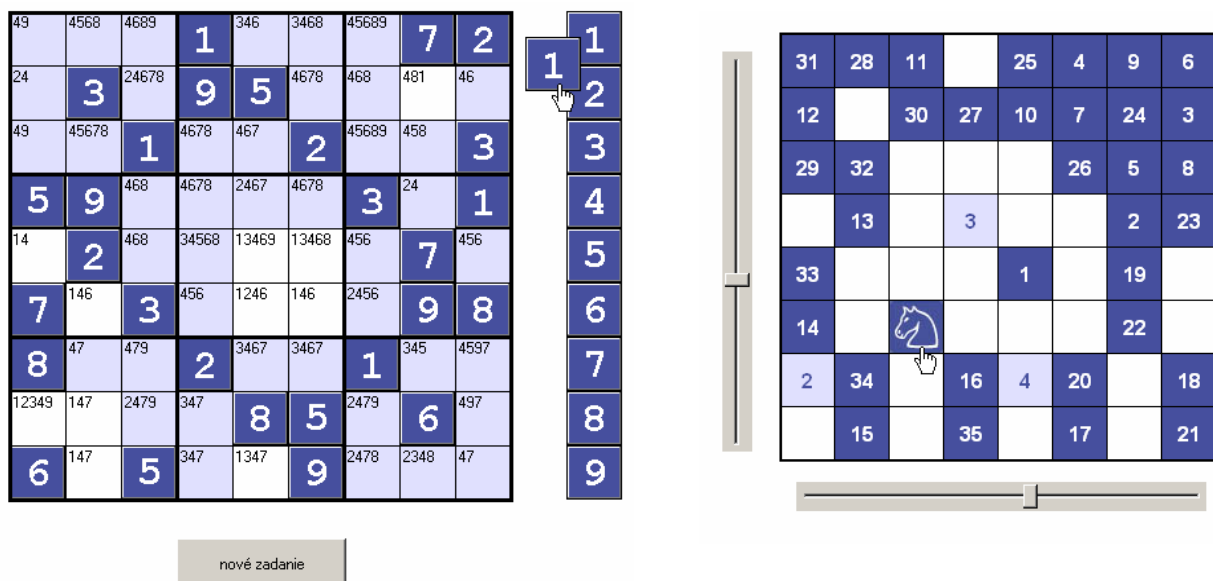


Figure 1. Screenshots of the Imagine Logo projects

In this paper, two Imagine Logo projects are presented. They provide interactive environments for solving two well-known logical puzzles – Sudoku and Knight's Tour. However, the main goal of these projects is not to find the solution but to assist a player in making good decisions when executing a nondeterministic algorithm. We want students to construct their understanding of the nondeterminism concept on the basis of their experience.

Keywords:

nondeterminism; computational complexity; heuristic algorithms; Imagine Logo; Sudoku; Knight's Tour problem

1 Introduction

The theory of computation and complexity theory are the fundamental parts of theoretical computer science. The first one deals with the question which problems are algorithmically solvable and which problems cannot be solved by any algorithm. The complexity theory provides methods and tools of quantitative measures enabling us to compare the efficiency of algorithms and to classify computing problems according to their computational hardness.

Algorithms can be evaluated and compared by a variety of criteria. Most often we are interested in the time or the space required to solve some problem by given algorithm. Both the time and the space differ from input to input, that means the algorithm's running time and memory space are functions of the input. The time complexity of an algorithm is measured by the number of elementary operations executed by the algorithm processing the given input. The space complexity of an algorithm is the number of elementary memory cells used in the computing process on some input. The computational complexity of the problem is the complexity of the best (optimal) algorithm solving the given problem.

We can classify problems according to their computational complexity into tractable problems admitting polynomial-time solution and intractable (computationally difficult) problems for which no polynomial algorithm is known. As computers have a finite memory and we have a finite time for waiting for a result, the complexity function of the algorithm bounds the size of inputs which can be processed. For that reason intractable problems are solvable in practical use only for some small inputs. Nevertheless there may be a large practical interest to have a program solving the hard intractable problems.

Some of these difficult problems become tractable using nondeterminism. We say that such algorithm is nondeterministic, in that some computing steps are optional and the algorithm guesses which of the available option is better. Consequently, it generates various computations for the given input. We can represent all possible computations of a nondeterministic algorithm on a given input as a tree. A deterministic computation, in contrast, is a linear sequence of computing steps.

How can nondeterminism help us to solve difficult problems effectively? Many intractable problems admit short polynomial-time certificates that means we can verify whether the solution is correct in real time. Such problems may be solved by polynomial nondeterministic algorithm, which can guess the solution and then check in reasonable time whether this guess was correct. Unfortunately, a deterministic simulation of the work of nondeterministic algorithm is usually a search for a solution in the computation tree and it usually causes an exponential increase of time complexity. In spite of this we can use the following approaches to obtain a solution in reasonable time (Calude and Hromkovic, 1997):

- Deterministic algorithm. If the inputs are in the range where time-complexity is not too large, then deterministic algorithm (despite of its exponential complexity in general) is a useful solution for hard problem.
- Problem restrictions. In several cases, we do not need to solve the given problem in generality and some restrictions on original formulation decrease the computational difficulty of the problem.
- Approximate algorithms. If we have a hard optimization problem, we can solve it easier searching for a solution which is not necessarily the optimum, but not too far from the optimum.
- Probabilistic algorithms. They are based on nondeterministic ones. Each nondeterministic step is interpreted as a random decision. Instead of surely correct outputs produced by deterministic algorithms probabilistic one computes output whose probability to be correct is large enough.
- Heuristics. They are similar to probabilistic ones; they make decisions during nondeterministic computation. The difference is that it is unable to prove that heuristic

algorithms provide correct solutions despite of the fact that they produce good outputs in practice.

2 Term of nondeterminism in secondary education

Algorithmics and programming are the fundamental and enduring concepts of computer science. They provide long-term knowledge in a rapidly changing discipline, in contrast of short-term computer-specific and application-specific skills. Understanding of these fundamental concepts is essential to effective use of computers and information technology. They also develop and improve students' general intellectual capabilities such as problem solving, logical reasoning and expressing ideas in a formal way. These are the reasons why fundamentals of algorithmics and programming are also a part of general education in secondary schools in Slovakia.

The term of nondeterminism discussed in this paper is usually not included in the introductory programming courses. Secondary school students are familiar with the idea that programs have to be deterministic because they are processed by computers automatically. Later, when they come to the university, it is difficult for them to understand the concept of nondeterminism. In our opinion, the gentle introduction to nondeterminism may be done even at the secondary school level.

A good motivation for introducing the nondeterminism and nondeterministic algorithms might be some logic-based puzzle games. We mean those puzzles, which are really difficult to solve by deterministic algorithms. One needs to use logical reasoning and several strategies and their combination to make right decisions when constructing a solution of such puzzles. Therefore, we suggest the puzzles as a suitable tool for showing the difficulty of intractable problems and the above-mentioned approaches to solving hard problems.

3 Imagine Logo projects

We have prepared two projects which provide interactive environments for solving two well-known logical puzzles. However, the main goal of these projects is not to find the solution but to assist a player in making good decisions when executing a nondeterministic algorithm. We want students to construct their understanding of the nondeterminism concept on the basis of their experience.

3.1 The Sudoku Project

Sudoku is a very popular number placement puzzle. The rules are quite simple: Fill a 9x9 grid so that every row, every column, and every of the nine 3x3 boxes contain each digit from 1 to 9 only once. There are usually some numbers already provided in the grid which represents an initial state and cannot be changed (Figure 2).

			1				7	2
	3		9	5				
		1			2			3
5	9					3		1
	2						7	
7		3					9	8
8			2			1		
				8	5		6	
6		5			9			

Figure 2. Sudoku puzzle

The Sudoku project implements two strategies that decrease the number of available options when looking for a cell where some digit can be placed. Let us describe the strategies briefly:

Strategy 1: Determine possible numbers for every single cell in the grid. If some cell contains one and only number, then that is the right solution for the cell. Figure 3 illustrates the output of scanning the grid for the puzzle depicted on Figure 2. Unfortunately, there is no cell with the only candidate to write in.

4,9	4,5,6,8	4,6,8,9		3,4,6	3,4,6,8		4,5,8	
2,4		2,4,6,7,8			4,6,7,8	4,6,8	1,4,8	4,6
4,9	4,5,6,7,8		4,6,7,8	4,6,7		4,5,6,8,9	4,5,8	
		4,6,8	4,6,7,8	2,4,6,7	4,6,7,8		2,4	
1,4		4,6,8	3,4,5,6,8	1,3,4,6,9	1,3,4,6,8	4,5,6		4,5,6
	1,4,6		4,5,6	1,2,4,6	1,4,6	2,4,5,6		
	4,7	4,7,9		3,4,6,7	3,4,6,7		3,4,5	4,5,7,9
1,2,3,4,9	1,4,7	2,4,7,9	3,4,7			2,4,9		4,7,9
	1,4,7		3,4,7	1,3,4,7		2,4,8	2,3,4,8	4,7

Figure 3. Strategy 1

Strategy 2: Determine possible cells for a given number. If some cell is the only possible cell in a row, a column or a box, then the given number can be placed in it. Figure 4 illustrates the output of scanning the grid for cells, which the number 1 can contain.

			1				7	2
	3		9	5				
		1			2			3
5	9					3		1
	2						7	
7		3					9	8
8			2			1		
				8	5		6	
6		5			9			

Figure 4. Strategy 2 applied for the number 1

The blue-coloured cells cannot contain the number 1. Notice the cell in the second row and eighth column. It is the only not coloured cell in its row, column and box, what makes this cell an acceptable place for the number 1.

It depends on difficulty level of the puzzle whether we can eliminate candidate digits for cells to get just one choice and find the solution in a deterministic way or whether we do need to do some nondeterministic decisions and then use the trial-and-error method as well.

The generalization of Sudoku puzzle to larger grids ($n^2 \times n^2$ board of $n \times n$ blocks) is known to be NP-complete problem (Yato, 2003).

3.2 The Knight's Tour Project

The second puzzle is concerned with a knight chess piece. The knight starts on an arbitrary chessboard's square and proceeds to move according to the rules of chess (*a chess knight moves along an L-shaped path: two squares horizontally and one square vertically, or two squares vertically and one square horizontally*) in order to visit each one of the remaining squares exactly once. The question is, whether it is possible for the knight to make such a tour around the board.

The very first idea is to test all possible ways that the knight could follow while touring. When the knight lands on a square from which it cannot get further without getting to an already visited square, it takes the last move back and tries another direction. Theoretically, if there is a solution, it will be certainly discovered. However, due to exponential complexity of the backtracking algorithm, this strategy does not help. For the classical 8x8 chessboard, it would take too long, to get a result.

In our project, an old but efficient heuristic method known as the Warnsdorff's rule is implemented. To build the solution quickly, we have to avoid creating dead ends. For that reason the possible squares to be chosen next are examined before every move. During the knight's tour, the number of free entrances to all squares of the board is gradually used up. Warnsdorff's rule serves to direct the knight to the squares with the least numbers of free entrances left - before these squares have turned into dead ends (Törnberg, 1999).

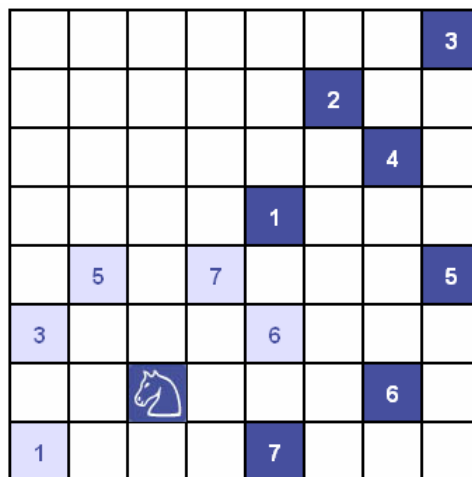


Figure 5. The knight is waiting until we make the next move

Take a look at the situation depicted in Figure 5. When the knight is standing on a square, all next legal moves are highlighted. For each one of the light blue squares, the number of free entrances is counted and displayed. Now, the knight should be moved to the square with the lowest number. Dark blue squares are the ones already visited. Each dark blue square is labelled with the proper ordinal number.

The Warnsdorff's rule gives solutions, but not all possible solutions. One can make moves opposing the rule and yet get a complete tour. In many cases there are equal choices and sometimes one might spare a square with the lowest number (in order to have it as the end square of a re-entrant tour). There is a certain limited freedom of choice, but squares with one entrance are due to be visited at once - otherwise they will turn into dead ends. One such square can serve as ending square for the whole tour, two means trouble.

It is possible to refine the Warnsdorff's rule so that it fails less often: if two or more possible next squares are equally good, then the one farthest away from the centre of the board should be chosen.

In graph theory, the knight's tour problem is an instance of the more general Hamiltonian path problem. And if we demand the knight's tour to be closed, we get an instance of the Hamiltonian cycle problem. Both of the graph problems are NP-complete. However, unlike the general Hamiltonian path problem, the knight's tour problem can be solved in linear time (Conrad et al., 1994).

4 Inside the projects

In both described projects, a new base class named *Card* is defined. This *Card* class extends the Imagine Logo built-in *Turtle* class. The auto dragging attribute of every *Card* object is on, its pen is up. After dropping the card inside the predefined area, it is automatically aligned to grid. The Sudoku puzzle and the Knight's Tour puzzle are played on similar square boards but differ a lot as for their rules. So there are another two classes derived from the general *Card* class. In both cases, a special functionality is added. The *SudokuCard* object can be put into a cell only if it is a card with proper number. The *ChessCard* object can be placed on a square only as a result of a legal chess knight move. The general *Card* class could also be used as the base class in other games played with cards on some kind of grid.

5 Conclusion

The teacher's role is to provide environment in which learners solve problems with the aim of learning and understanding something new. Our methodology focuses on the fact that solving process of the discussed puzzles is not deterministic. Learners gain valuable experience in nondeterminism whilst solving the puzzles. This is what they should learn:

- algorithm is nondeterministic when it generates computations with some optional steps,
- nondeterministic algorithm generates various computations for a given input, which may be represented as a tree,
- deterministic time corresponds to the complexity of the search for a solution while nondeterministic time corresponds to the complexity of verifying whether a given solution is correct.

The Imagine Logo development environment offers everything needed to implement an interactive and visually attractive educational application (e. g. object orientation, event-driven programming, multimedia support). Therefore, we used it for our projects as well (Hrušecká and Kalaš, 2006).

References

- Calude, C. and Hromkovic, J. (1997) *Complexity: A Language-Theoretic Point of View*. In Handbook of Formal Languages, G. Rozenberg and A. Salomaa (eds), Volume 2. Springer, pp.1-54.
- Conrad A., Hindrichs T., Morsy H. and Wegener I. (1994) *Solution of the Knight's Hamiltonian Path Problem on Chessboards*. Discrete Applied Math, volume 50, no.2, pp.125-134.
- Hrušecká, A. and Kalaš, I. (2006) *Programovanie v prostredí Imagine*. MPC v Bratislave, Bratislava.
- Törnberg, G. (1999) *Warnsdorff's rule*. <http://web.telio.com/~u85905224/knight/eWarnsd.htm>
- Yato, T. (2003) *Complexity and Completeness of Finding Another Solution and its Application to Puzzles*. Master thesis, University of Tokyo, Tokyo.

Student-Centered Support Systems to Sustain Logo-like Learning

Sylvia Martinez, *sylvia@genyes.com*
Generation YES, Olympia, Washington, USA

Abstract

Conventional wisdom attributes the lack of effective technology use in classrooms to a shortage of professional development or poorly run professional development. At the same time, logo-like learning environments require teachers to develop more expertise not only in technology but also in pedagogy.

This paper proposes that the perceived lack of technology professional development is a myth and that traditional professional development is ill-suited to teaching teachers how to create logo-like learning environments. Furthermore, it proposes models of student-centered, student-led support for teachers that support classroom practice aligned with the attributes of logo-like learning environments. These models situate teacher learning about technology in their own classroom, reinforce constructivist teaching practices, provide support for technology use in the classroom, and enrich learning environments for students.

Keywords

Student-centered; logo-like; technology; professional development; learning environment

Problems in Professional Development for Logo-like Learning

Logo-like learning is characterized by learning by doing, experimentation, authentic work, student agency, serendipity, reflection, collaboration, and community expertise. It requires teachers with intellectual curiosity, creativity, ongoing personal learning habits, and the ability to collaborate with students while maintaining a classroom that has both purpose and freedom. Such teachers are able to create learning environments that are distinguished by intellectual challenge, wonder, social interaction, and student engagement. (Harvey 1993; Papert 1992; Stager 2005)

Although technology enables logo-like learning, it requires substantially more technical and intellectual fluency from a teacher than the typical computer application courses found in many schools. This makes the professional development challenges more substantial.

Regardless of its merits, logo-like learning has proven difficult to sustain in traditional schools. While Papert attributes this to school's "immune reaction" to an invading foreign body (Papert 1997), conventional wisdom ascribes this failure to lack of teacher professional development.

The myth of insufficient professional development

The speculation that the primary barrier to effective technology use in schools is the result of insufficient professional development goes unquestioned and has become a myth used to excuse the lack of progress.

Lack of professional development for technology use is one of the most serious obstacles to fully integrating technology into the curriculum (Fatemi, 1999; Office of Technology Assessment, 1995; Panel on Educational Technology, 1997). (Rodriguez and Knuth 2000)

Even when the style of professional development is called into question, the typical remedy is more of the same, with additional structure and more emphasis on external expertise.

But traditional sit-and-get training sessions or one-time-only workshops have not been effective in making teachers comfortable with using technology or adept at integrating it into their lesson plans. Instead, a well-planned, ongoing professional development program that is tied to the school's curriculum goals, designed with built-in evaluation, and sustained by adequate financial and staff support is essential if teachers are to use technology appropriately to promote learning for all students in the classroom. (Rodriguez and Knuth 2000)

Critiques from a wide spectrum of researchers focus on variables such as evaluation, seat time, correlation to mandated curriculum standards, testing outcomes, and compensation. Occasionally, issues of teachers' motivation to learn are mentioned. (Brand 1997; Rodriguez and Knuth 2000)

The truth is, American (and most Western) teachers are receiving quite a bit of professional development related to technology.

Indeed, most teachers had participated in multiple professional development activities in the year prior to taking the survey, and yet more than 80 percent indicated a need for training in how to integrate technology into the curriculum. (SRI 2002)

Most teachers indicated that professional development activities were available to them on a number of topics, including the use of computers and basic computer training, training on software applications, and the use of the Internet (ranging from 96 percent to 87 percent). Among teachers reporting these activities available, participation was relatively high (ranging from 83 to 75 percent), with more experienced teachers generally more likely to participate than less experienced teachers. (NCES 2000)

In a 2003 study of teachers in the Chicago Public Schools, a large majority of teachers confirmed that lack of professional development was not a great barrier to technology use. (CCSR 2003)

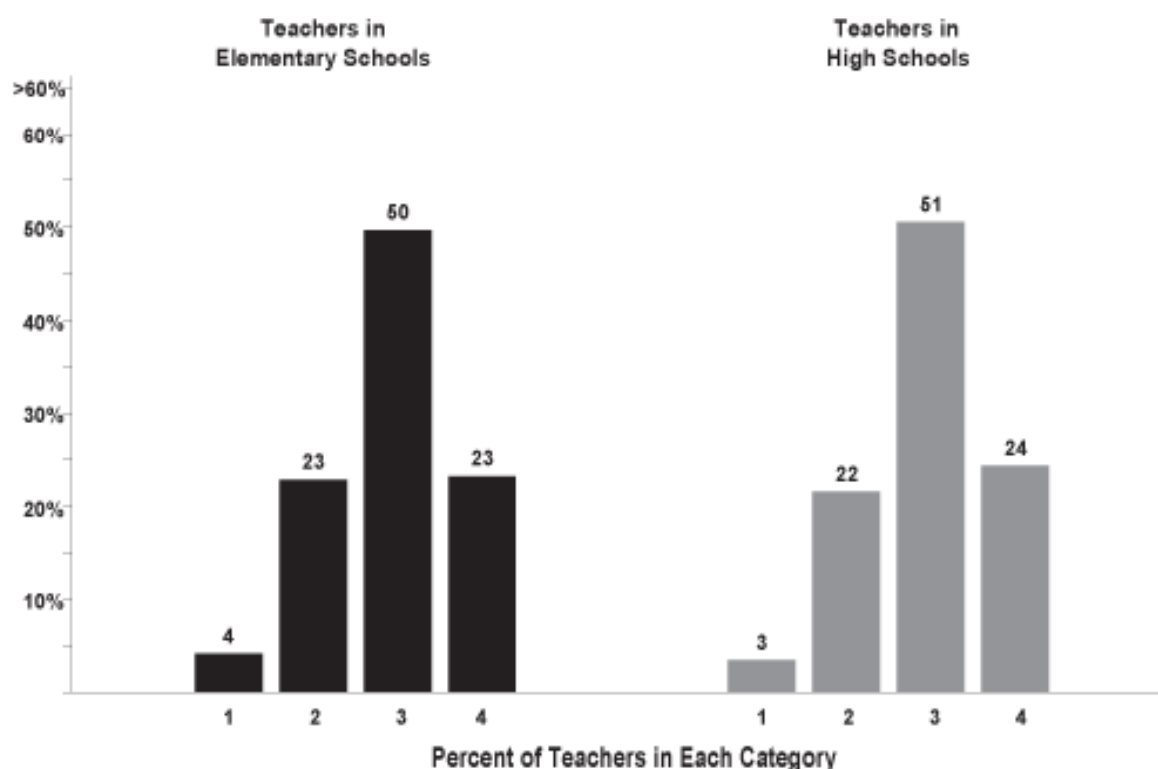


Fig. 1 - Teachers' assessments of their awareness of and participation in professional development activities designed to integrate technology in the classroom.

Definition of Categories Charted Above

Category	In Chicago Public Schools:
1 - Very Weak	Teachers agree that the lack of both appropriate professional development and release time for learning and planning are great barriers to technology use.
2 - Weak	Teachers somewhat agree that lack of both professional development and release time to be great barriers to technology use.
3 - Strong	Teachers find lack of both professional development and release time as small barriers to moderate barriers.
4 - Very Strong	Teachers describe the lack of professional development and release time as not a barrier.

Failure of traditional professional development

The failure of traditional professional development to change teacher practice with technology may be due to several factors. Effective use of technology, such as those sought in a logo-like learning environment, extends well beyond understanding the specific technology. It requires a different teaching style. Traditional approaches to professional development tend towards the didactic, while the real power of the computer is when it's used in a personally meaningful, constructive fashion.

Federal, state and local agencies are investing billions to equip schools with computers and modern communication networks, but only one-third of our nation's teachers feel well prepared to use computers and the Internet in their teaching. (NCES 2000)

This is evident in reports by students of computer use in their classes. In the 2005/06 school year, thousands of GenYES students in the United States were asked about their own computer use. This is astonishing, considering that GenYES schools are a self-selected group with a deliberate intent to improve the use of technology in every classroom. These results have not changed significantly in five previous annual surveys.

In the classes you took last semester/quarter, how often were computers used by you or your teachers?	Computers were never used	Computers were used once	Computers were used a few times	Computers were used about once per week	Computers were used several times per week
Math	46.5%	9.4%	26.1%	8%	9.9%
Language Arts, Reading or English	30.4%	11.2%	33.6%	12.2%	12.7%
Science	47.2%	13.2%	25.4%	6.7%	7.4%
Social Studies, Geography or History	36.8%	12.2%	27.8%	11%	12.3%

In the same time period, the number of computers available for classroom use in the U.S. went from approximately one computer for every six students to one to every four students. (SAUS 2006) Even if the pace of computer introduction has slowed from the rapid changes of the 1980s and 1990s, there is certainly sufficient availability. There should be significantly more use of computers in the daily lives of students.

Despite the evidence that traditional professional development is ineffective, there remains an insatiable appetite for more. Perhaps the important variable isn't quantity, but constancy and community.

Community of practice

The primary community of practice for teachers is within the confines of their own classroom. The participants are the teacher and her students. Other peripheral participants can visit, but for most classrooms, these visits are few and short. While teachers may participate in other communities in a professional capacity, for most, the classroom is the only setting for their professional practice.

Wenger, in discussing designs for learning inside communities of practice, makes the point that they, "...cannot be based on a division of labor between learners and nonlearners, between those who organize learning and those who realize it, or between those who create meaning and those who execute." (Wenger 1998)

Traditional forms of professional development remove the teacher from their classroom and attempt to create a community of practice made up of teachers and technology experts. This community exists only for the purpose of imparting information from the experts to the teachers. While there is certainly a place for collegial discussion and access to professional improvement, it is not unreasonable that teachers often reject transparent efforts to force them into participation.

Common recommendations for technology professional development include that teachers be given more time for "independent practice without fear of embarrassment," to watch expert practitioners, go to conferences and workshops, or participate in online learning communities. (Rodriguez and Knuth 2000)

These attempts to improve technology professional development only serve to reinforce the separation between the teacher learning new skills and real change in classroom practice. Schlager & Fusco construct a compelling argument based on years of design and facilitation of

Tapped-In, an online teacher community. This type of professional development, "... tends to pull professionals away from their practice, focusing on information about a practice rather than on how to put that knowledge into practice." (Schlager and Fusco 2004)

Mere discussion about practice does not create a community of practice.

Even if professional development excites teachers about new possibilities and tools, the teachers are removed from the successful context and sent back to the classroom to fend for themselves. They are expected to use their new skills without colleagues or experts present. One-on-one coaching that provides in-class mentoring is expensive and rarely available. Online teacher communities can only take place outside of classroom time, too late for any intervention or advice to be useful. As teachers struggle alone in their classroom with questions, issues, and problems, valuable teachable moments are missed.

In an interview discussing what changes need to take place in classrooms to allow project-based learning, Papert says, "What we need is kinds of activity in the classroom where the teacher is learning at the same time as the kids and with the kids. Unless you do that, you'll never get out of the bind of what the teachers can do is limited by what they were taught to do when they went to school." (Papert 2001)

Creating sustainable systems that allow teachers to learn alongside students in the classroom is imperative to support teachers responsible for sustaining logo-like learning.

The GenYES model

GenYES is a model of reverse mentoring developed specifically for K-12 teachers learning to use computers in their classrooms. GenYES students learn about technology, develop planning and collaboration skills, and learn how people learn. As a culminating project, each student is partnered with a teacher in the school. These student/teacher teams review the teacher's curriculum and decide on a future lesson that could be enhanced with technology. The student then researches and creates the project, with the teacher providing insight into pedagogy and the curricular content. When the project is completed, the teacher has something they can use in their classroom, understands of technology better and can directly observe the impact on their students. (Harper 1998)

A single teacher or staff member facilitates all of this as the GenYES teacher. A decade of research demonstrates that this model improves teachers' use of technology in the classroom and changes teacher attitudes about the usefulness of technology as they view the benefit to their students. (Coe 2003; Harper 1998) Over the last 10 years, the model has been called Gen Y or Generation www.Y, but is now known as GenYES

"The format provides a model of project-based, authentic, student-centered, multidisciplinary teaching and learning enhanced by technology." (Coe 2003)

This partnership creates a two-person collaborative learning community grounded in constructivist pedagogy. Both the student and the teacher are bringing important skills and knowledge to the partnership. Improving technology use in schools is an authentic problem worth solving. Working together creates mutual respect and understanding about each other's roles.

The model facilitates a constructivist atmosphere in the whole school, not by teaching teachers how to teach, but by giving students an authentic problem to solve and asking teachers to help. GenYES builds on the pedagogical models developed by Dr. Papert and the years of research about what constructivist teachers do in classrooms.

Teachers working with GenYES students report that the experience helped them better understand technology and its place in education.

- 89% agreed that as a consequence of GenYES, their students learned content better

- 98% reported that as a consequence of GenYES, they would continue rebuilding their lessons to make more use of technology
- 82% reported that the GenYES experience would change the way they teach in the future
- 95% of the partner-teachers consider GenYES a good method for providing support and assistance to teachers as they integrate technology into their classes, (Coe 2003)

Constancy and community

Although the GenYES model is a structured mentoring process with a specific focus, it sheds light on the steps and key factors needed to create a classroom community of practice that supports logo-like learning.

In the classic explanation of social constructionism, Papert describes the Brazilian samba school where, “Novice is not separated from expert, and the experts are also learning.” (Papert 1980)

By creating a classroom-based learning community like GenYES, the problems of constancy and community are solved. Teachers and students can be teachers, learners, novices or experts. Roles can shift from day to day, task to task. Teachers can try new technology knowing that their students have expertise, or if not, the class can learn it together. Together, they are doing the real work of learning. This community of practice is available all the time and not dependent on outside expertise.

There is of course a role for outside expertise, but accessing an outside expert and sharing it with a community allows the whole community to benefit. Modelling that behavior for students creates an expectation that learning to learn is the key to success.

Creating and Sustaining Classroom Practice that Supports Logo-like learning

Even if student-led models of support like GenYES are initiated in a school, the question remains, how will teachers know what to do and how will teachers and students learn how to use technology? Surely, they will need textbooks, lesson plans, curriculum, experts, and reference material?

While there is certainly a need for reference material, there is ample evidence that when students and teachers work together to solve authentic problems, their search for the solution, and their ownership of the process creates a deeper understanding than a pre-planned lesson. In the age of the Internet, fixed lesson plans become less important as resources change rapidly, and search engines bring answers immediately when questions come up. The answer to “how do I ...” is easy to find. Having an authentic reason to ask that question is hard. This should be the focus of the teacher’s role in creating a logo-like learning environment.

In an interview about the One Laptop Per Child Initiative, Papert responds to the question, how will teachers and students be trained, with, “In the end, they will teach themselves. They’ll teach one another. There are many millions, tens of millions of people in the world who bought computers and learned how to use them without anybody teaching them. I have confidence in kids’ ability to learn.” (Papert 2006)

During the professional development process for the GenYES lead teacher, no technology instruction occurs. GenYES lead teachers are encouraged to teach collaboration and communication skills to their students. Lesson plans and role-playing scripts for the students are provided since most have little experience working collegially with an adult. Our lead teachers are encouraged to have students work on projects as quickly as possible and provide sufficient time and space necessary to create an atmosphere that rewards risk-taking.

Getting out of the way

One of the most common comments from beginning GenYES teachers is that they saw the class start to work when they “got out of the way.” Although the value of student self-reliance is

stressed in the GenYES teacher resource materials and professional development activities, many teachers need to see students assume responsibility for their own learning to be convinced. Some teachers experience an epiphany when they suddenly realize the creative chaos in their classroom is not only functional, but valuable. They then allow themselves to surrender some control. This tendency for technology projects to follow many varied paths can in and of itself be the catalyst that forces teachers to step out of their leadership role in the classroom.

Fear of failure

Fear of failure, of looking stupid, and of losing control also play a large part in whether a teacher can create an environment that allows students to take the lead. Teachers often create carefully staged lessons in technology based on their own fear of the computer, rather than a true understanding of what students need to know. By creating a delay in the process of students getting to the real work, teachers assuage their own fears, but bore students. Recommendations that teachers practice new technology skills in private only serve to intensify this fear by acknowledging that mistakes are shameful and will somehow harm students.

Technology choice, implementation, and support as an authentic problem

Authentic problems inspire creative thinking and empower students to exceed expectations and think creatively. Technology in education has many authentic problems beyond the use in the classroom. Selecting the right tools, creating acceptable use provisions and penalties, making decisions about security, analyzing costs, and planning for maintenance are all considerations.

Teachers can sometimes act as roadblocks in technology use as their quest to find the perfect technical solution and design a flawless implementation plan takes months or even years to complete. Students can shortcut this process and learn valuable lessons along the way.

By including students in the process of researching, choosing, testing, and implementing new technology, students experience not just the use of technology, but the decision-making process and tradeoffs that surround it. By including them in making and enforcing safety or usage policies, they are learning to think beyond their own experiences, make predictions, and take responsibility for community actions. Students who participate in these tasks will communicate and explain policies to their peers more effectively than adults. Students will also be more forgiving of technology shortcomings that inevitably takes place when implementing new technology if they have been a part of the discussion of tradeoffs.

Papert makes the case for student's leading the way to creating their own learning environments, "I believe in "Kid Power." Our education systems underestimate kids. It INFANTALIZES them by assuming they are incompetent." (Papert 2006)

Of course, there are times when a teacher introduces an application or technology that supports specific curricular or pedagogical goals. The process of including the students in authentic problems does not mean that students need to be included in every aspect of the process.

Students as teachers

To teach is to learn twice - Joseph Joubert

It is not unusual for students to serve in the role of teachers in schools, although these are usually near-age peer tutoring programs. In technology settings, the research done on children as software designers (Harel 1991; Kafai 1995) outlines multiple benefits of students collaborating and teaching each other as they design educational software and games. However, there are ways that students can teach peers, teachers, and other members of their community about technology that benefit the entire school. By creating a legitimate role for students of this digital generation to share their experience and facility with technology, schools can create in-class professional development opportunities for teachers, support teachers as they use technology in the classroom, and provide better technical support to teachers in the classroom.

This does not mean that students can just be told to help out and this will automatically have a significant impact. The success of the GenYES model shows that these students need guidance to learn new roles and be successful collaborators and teachers. Teachers and staff need guidance as well and time to learn to trust and accept new roles for students.

Students as teachers of teachers

In the GenYES model, students serve as teachers in one-on-one partnerships with teachers. The model supports both students and teachers as co-learners and co-teachers. Building on a time honoured practice of students “helping out” GenYES is a non-threatening, yet subtly subversive way to have students contribute to the technology-enabled learning environment.

Papert said about the GenYES model, “The genius of this idea is that by contributing to solving a recognized problem facing schools, it rallies support from schools for something that goes against the grain of their traditional ways of thinking.” (Papert 1998)

Our research indicates that teachers are not embarrassed by students teaching them about technology. Teachers became teachers because they like children. The key is creating a school-wide environment where the teacher and student interaction and knowledge sharing is a normal and accepted practice.

Students as teachers of peers

Student peer mentoring can also serve as a primary support system for logo-like learning environments. Student mentors leverage student expertise and benefit both students and teacher. Peer mentors:

- Increase available opportunities for all students. Other students can get assistance more quickly and in more areas than if the classroom must rely on a single expert (the teacher) or wait for outside expertise.
- Allows the teacher to teach. The teacher is relieved of the role of sole technology guru and can focus on helping struggling students, focus on bigger ideas, and be attuned for teachable moments.
- Expand the capacity for multiple applications and hardware. Individual students can become expert in an application or a particular use of technology and help others.

Encouraging student expertise in technology mentoring also creates leadership opportunities for students that support their own personal interests and learning styles. They will learn these skills more deeply and with greater enthusiasm than a student asked to learn a little bit about a lot of applications. A spreadsheet expert can be as important and celebrated as a student with expertise in animation, 3D design or music composition. If these unique gifts and interests are recognized and honored in the classroom, teachers, students, and the mentors themselves benefit.

In addition, the social, academic, and behavioral benefits of peer mentoring for both the mentor and the mentee are well documented (Gartner & Riessman 1993, Benard, 1990, Viadero 2003). This is especially true in at-risk communities, where peer mentors can bridge cultural gaps between teachers and students. (Snow 2003, Rohrbeck 2003)

We have formalized this model in a student technology literacy program called TechYES. Research on the TechYES model shows that teachers report increases in a wide array of student technology skills, as would be expected, but also report increases in student and mentor self-esteem, interpersonal skills and academic skills. In addition, teachers reported that this model increased their understanding and ability to teach in a project-based, open-ended way than prior to teaching the TechYES class. (Schneider, 2006)

Students as technical support

Another way students can serve as support for teachers using technology in the classroom is as technical support. This solution has many benefits for both teachers and students. Teachers with

consistent access high-quality technology support use technology more with students, and in a wider variety of ways, than teachers with inadequate technology support programs. (Ronnkvist et. al., 2000)

Students benefit from the challenges and responsibility given to them. Students learn how to troubleshoot and find answers by researching them, not being told. The problems found in school computers are authentic, varied and the need is high. Students also work well with teachers, at least as well as hired technical support staff. Students' knowledge of the school, the culture, and the teachers helps them solve problems in ways that support the learning environment.

Conclusion

The only resource in abundance in schools is students and their boundless energy, passion and enthusiasm for learning. By creating structures in classrooms and schools that encourage and support student participation in creating logo-like learning environments, we improve teacher practice and diminish dependency on outside expertise for teacher professional development.

References

- Benard, B. (1990). *The Case for Peers*. Portland, OR: Northwest Regional Educational Laboratory. ED 327 755
- Brand, G. (1997). *What Research Says: Training Teachers for Using Technology*. Journal of Staff Development, Winter 1997 (Vol. 19, No. 1). Retrieved April 2, 2007 <http://www.nsd.org/library/publications/jsd/brand191.cfm>
- Coe, M. T. (2003). *Students, teachers, and technology building better schools: Generation Y Project Evaluation*. Portland, Oregon: Northwest Regional Educational Laboratory. Available: <http://genyes.org/products/geny/genyresearch>
- Consortium on Chicago School Research (2003). *Key Measures of School Development*. Retrieved April 12, 2007 http://ccsr.uchicago.edu/web_reports/keymeasures/keymeasures.html
- Gartner, A. & Riessman, F. (1993). *Peer-tutoring: Toward a new model*, ERIC Digest. ED362506
- Harel, Idit (1991). *Children Designers*. NJ: Ablex.
- Harper, D. (1998). *Generation Y: Second annual report*. Washington, DC: U.S. Department of Education.
- Harvey, B. (1993) *Symbolic Programming vs. Software Engineering -- Fun vs. Professionalism -- Are These the Same Question? Eurologo 1993* Retrieved April 10, 2007 from University of California, Berkeley website: <http://www.cs.berkeley.edu/~bh/elogo.html>
- Kafai, Y. (1995). *Minds In Play: Computer Game Design as a Context for Children's Learning*. NJ: Erlbaum.
- National Center for Education Statistics (2000). *Public school teachers' use of computers and the Internet*. Washington DC: U. S. Department of Education.
- Papert, S. (1980). *Mindstorms: Children, Computers and Powerful Ideas*. New York: Basic Books, Inc.
- Papert, S. (1992). *The Children's Machine: Rethinking School in the Age of the Computer*. New York: Basic Books, Inc.
- Papert, S. (1997) *Why School Reform Is Impossible*. The Journal of the Learning Sciences , Vol. 6, No. 4, Pages 417-427. Retrieved April 12, 2007 from http://papert.org/articles/school_reform.html

- Papert, S. (2001) *Seymour Papert on Project-Based Learning*. Edutopia: The George Lucas Educational Foundation. Retrieved April 10, 2007 from http://www.edutopia.org/php/interview.php?id=Art_901&key=037#paragraph6
- Papert, S. (1998) *Technology in Schools: To Support the System or Render it Obsolete*. Milken Family Foundation. Retrieved April 1, 2007 http://www.mff.org/edtech/article.taf?_function=detail&Content_uid1=106
- Papert, S. (2006) *Professor Papert Discusses One Laptop Per Child Project*. U.S. International Information Programs (USINFO) Webchat transcript. Retrieved April 14, 2007 <http://usinfo.state.gov/usinfo/Archive/2006/Nov/14-358060.html>
- Ronnkvist, A., Dexter, S. and Anderson, R. (2000). *Technology support: Its depth, breadth, and impact in America's schools*. Center for Research on Information Technology and Organizations, University of California, Irvine And University of Minnesota. Retrieved April 10, 2007 from <http://www.crito.uci.edu/tlc/findings/technology-support/startpage.htm>
- Rodriguez, G. and Knuth, R. (2000). *Critical Issue: Providing Professional Development for Effective Technology Use*. North Central Regional Technology in Education Consortium (NCRTEC). Retrieved April 12, 2007 <http://www.ncrel.org/sdrs/areas/issues/methods/technlgy/te1000.htm>
- Rohrbeck, C.A., Ginsburg-Block, M.D., Fantuzzo, J.W., & Miller, T.R. (2003). *Peer-assisted learning interventions with elementary school students: a meta-analytic review*. Journal of Educational Psychology, 95(2), 240-257.
- Schlager, M. S. and Fusco, J. (2004). *Teacher professional development, technology, and communities of practice: Are we putting the cart before the horse?* In S. Barab, R. Kling, and J. Gray (Eds.), *Designing Virtual Communities in the Service of Learning*. Cambridge University Press.
- Schnieder, S. (2006). *Verizon california technology literacy project interim evaluation*. Private evaluation study delivered to Verizon. Retrieved August 2006 http://www.geny.org/verizon/evaluation/TechYES_Interim_Evaluation_Report.pdf
- Snow, D. (2003). *Classroom strategies for helping at-risk students*, Noteworthy Perspective. Mid-continent Research for Education and Learning, available from www.mcrel.org
- Stager, G. (2005). *Papertian Constructionism and the Design of Productive Contexts for Learning*. In *Proceedings of EuroLogo 2005*. Retrieved April 5, 2007 from <http://eurologo2005.oeiizk.waw.pl/PDF/E2005Stager.pdf>
- Stanford Research Institute (2002). *Integrated Studies Of Educational Technology: Professional Development And Teachers' Use Of Technology*. Prepared for the U.S. Department of Education Policy and Program Studies Service.
- Statistical abstract of the United States (2006) Washington, DC : U.S. Dept. of Commerce, Economics and Statistics Administration, Bureau of the Census, Data User Services Division. Retrieved April 10, 2007 <http://www.census.gov/compendia/statab/education/>
- Viadero, D., *Studies Show Peer Mentoring Yields Benefits for Students*, Edweek.org, July 9, 2003, <http://www.edweek.org/ew/articles/2003/07/09/42peer.h22.html>
- Wenger E. (1998) *Communities of Practice: Learning, Meaning, and Identity*, Cambridge University Press, Cambridge.

Musical Method in Programming Education

Takeo Tatsumi, tttt@cc.tuat.ac.jp

Information Media Center, Tokyo University of Agriculture and Technology

Mitaro Namiki, namiki@cc.tuat.ac.jp

Graduate School of Technology Management, Tokyo University of Agriculture and Technology

Abstract

To learn computer programming language(CPL), student must learn many terms, definitions, syntaxes and semantics. For young pupils/students, there are too many elements to learn CPL. That is why the number of students who are learning CPL in K12 is on the decrease worldwide.

August 1998, Tatsumi, first author, pointed out that there are many similarities between learning musical scores and learning CPL. There are many typical structures in music. For example, the notion of correct sequences of notes in several measures, a branching several forms of notes by visiting times.

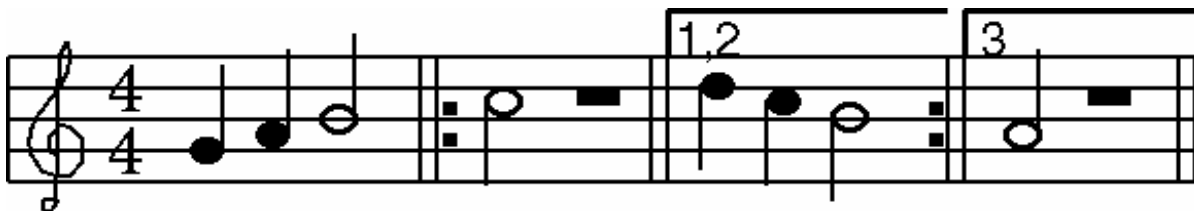


Figure 1: playing different measures by visiting times

This example is good to explain of the similarities of musical score and computer program. Also, "Tripartite form: A-B-A", "Rond form: A-A-B-A", "Sonata form: proposal, disclosure, re-propose." Students will know that any computer can read only the script he/she wrote. So our proposal is that musical method in education of CPL must be included to teach programming for music teachers and to teach musical script for ICT teachers. We think that music is good educational subject for scripting education in K12. The reason is as follows: 1. Music is independent from another educational subject, 2. Student can find a bug in the score in playing, 3. Music includes almost technique in computer programming, 4. Music is joyful for study.

October 2001, he proposed the musical method for learning CPL. His method can be one of helpful methods in learning CPL.

Dolittle is a name of a programming language and integrated tools. Dolittle project is organized by Prof. Kanemune, Prof. Kuno and their group from 2000. First implementation of Dolittle has only Turtle Graphics commands, controls. So many researchers think Dolittle as some Japanese version of Logo. However, the syntax of Dolittle based on OOP(Object Oriented Programming) style which is different style of Logo. Namiki, second author, made some contributed works to Dolittle project from 2002. November 2003, Tatsumi got a post of associate professor in TUAT to which Namiki belong. We started collaborated research.

September 2005, we proposed many specifications for Dolittle project. In this paper, We describe why musical method is effective in a class of learning CPL. We proposed adding many musical terms to Dolittle.

Keywords

computer programming education, music

Education of Computer Programming Language

For these years, the development of Information and Communication Technology (ICT) is changing our daily life and the way of thinking. Education is affected too. Many teachers and students seek good efficiency in their classes using ICT in education. In many researches and reports, many people confuse "ICT education" with "education using ICT". These concepts must be distinguished. Although "education using ICT" is important, "ICT education" is important too. In Japan, many teachers of K12 think little of education of "Computer Programming Language (CPL)". In many classes of CPL, teacher like to set a goal too high. The textbook of CPL is too difficult for beginner. Many lectures of CPL are tedious because the teacher of CPL has no experience in entertainment of art, music and sports. In this situation, there are many experiments to break through. Some experiments are in successes and some are in fail.

Music education using ICT

Some result of music education using ICT

In BBC's web page, the article about class of music by Mr Adrian Pitts who is music teacher of England.

Computers are playing an increasing role in music, especially in secondary schools. Now software can enable students to compose and arrange music and develop their musical creativity, even when they can't play an instrument. But provision is patchy and can depend upon the enthusiasm of your child's music teacher.

When used appropriately ICT can deepen pupils' understanding of music without being reliant on the pupil's physical ability to perform the music written

Students made the best use of ICT when they had been taught the building blocks needed to compose. Musical software alone was not enough to ensure success

In a class of music, students learn how to sing a song, how to play musical instruments and how to read musical scores. In recent years, music education are changed by many computers and networks. Before this change, student had to use pencil or pen to take down in musical notation. Student had to count numbers to check correctness of their rhythm. Student had to play many musical instruments to adjust the harmony of their music. After this changing, composer software play the great role in the field of music education. Student use composer softwares to compose, edit, check and play their score. It is easier way than old way of composing. Many music teacher got an easier way to write down a practice sequence for their students.

Application of ICT for another school subjects

In this section, We describe the school subject of "Information" and the application of ICT education for another school subjects.

School subject of ICT in Japanese high school

In Japan, the Ministry of Education decided the national regulation of K12 schools. After 2003, "Information" is listed as a name of school subject of Japanese high school. In many countries, there is no school subject on "Information and Communication Technology" independently. In stead of "Information", many countries set the regulations of many school subjects including computers and networks technologies.

Application of another school subjects

Also in Japan, besides teachers of “information”, many teachers of many school subjects use computers and networks in teaching their own classes.

For example, some teacher teach mathematical geometry using computer graphics application software. Some teacher teach dynamics using computer simulation software. Some teacher teach Japanese language using word processor software. In this paper, I define the such improvement of the school subject as "Informationalization".

Teacher of another school subject can teach computer programming. For example, teacher can teach making a program of sweeping any matrix. Teacher can teach making a program of simulating the rule of genetics. Teacher can teach making a program of estimating the national population.

The aim of a school subject does not involve the aim of another subject.

As I wrote above, the aim of another school subjects does not involve the aim of ICT. Mathematics teachers want to teach not programming but mathematics. Physics teachers want to teach not programming but physics. If a teacher want to teach programming, then he/she must put his/her goal in ICT education.

If a teacher of ICT want to teach programming relating another school subjects, then he/she would use the methods of these school subjects. So, August 1998, Tatsumi thought of the musical methods to teach computer programming. The goal of this method is in not music but ICT.

Musical Method in ICT education

In 1998, Tatsumi talked about “Musical Method in ICT Education” at “Joho-Kagaku Wakate no kai” (Young researchers annual forum on Informatics) about this similarities. Musical scores are written using some rules: repeating, jumping, skipping and other controls. These rules are also used in a programming language. So understanding these musical rules can help to read computer program. That is why musical method is good candidate to develop new method of learning CPL. October 2001, Tatsumi wrote the paper titled as “Musical method in ICT Education” and he posted proposals for several grants. However, all proposals were rejected. Because Tatsumi do not know the name who writes codes using MIDI and understand the project meaning, Tatsumi has paused this project after 2003.

Dolittle - from birth to v1.24

In 2000, Prof. Kanemune who was researching educational programming language (EPL) has developed "Dolittle". Many teachers might think that Dolittle look like Logo because Dolittle adopt Turtle Graphics. Dolittle is designed in object oriented programming (OOP) structure. The language specification of Dolittle allows multiple definition using natural language like Japanese, Korean, and English. For example, you can use Japanese words and Kanji characters in Dolittle to write a program which is composable using English words. Prof. Kanemune and Prof. Kuno developed Dolittle in cooperation and several Japanese teachers of K12 joined the project of Dolittle. They had many experimental class in elementary schools, junior high schools, high schools and universities. They added many functions to Dolittle. Displaying characters, operation of robot and communication via network. June 2004, Dolittle got new function of playing simple melody. Students have been able to control MIDI device using new Dolittle. However Prof. Kanemune has added new function thinking the structure of MIDI specification. So, Dolittle had some strange grammar in playing music.

Rendez-Vous avec Dolittle - after v1.25

Tatsumi has deeply studied "Music education using ICT" and "ICT education using Music" from 1998. Namiki is a professor of TUAT(Tokyo University of Agriculture and Technology) from 1993. October 2003, Tatsumi moved to TUAT. Then our collaborated work has started. Namiki

is a researcher of computer operating software and system software. Namiki also related the project of Dolittle. Tatsumi was invited to Dolittle project 2004. September 2005, Tatsumi tried to make manuscript of the textbook of musical method which can be used in the class of Dolittle. Then, Tatsumi noticed which terms in Dolittle are inadequate for music. So, Tatsumi proposed new grammar and specification to include adequate musical terms into Dolittle. Prof. Kanemune added new code of Dolittle interpreter. November 2005, Dolittle v1.25 was born. New Dolittle was used at Oyumino elementary school in Chiba city, Japan. To control musical devices for pupils, it was easier new Dolittle than old one.

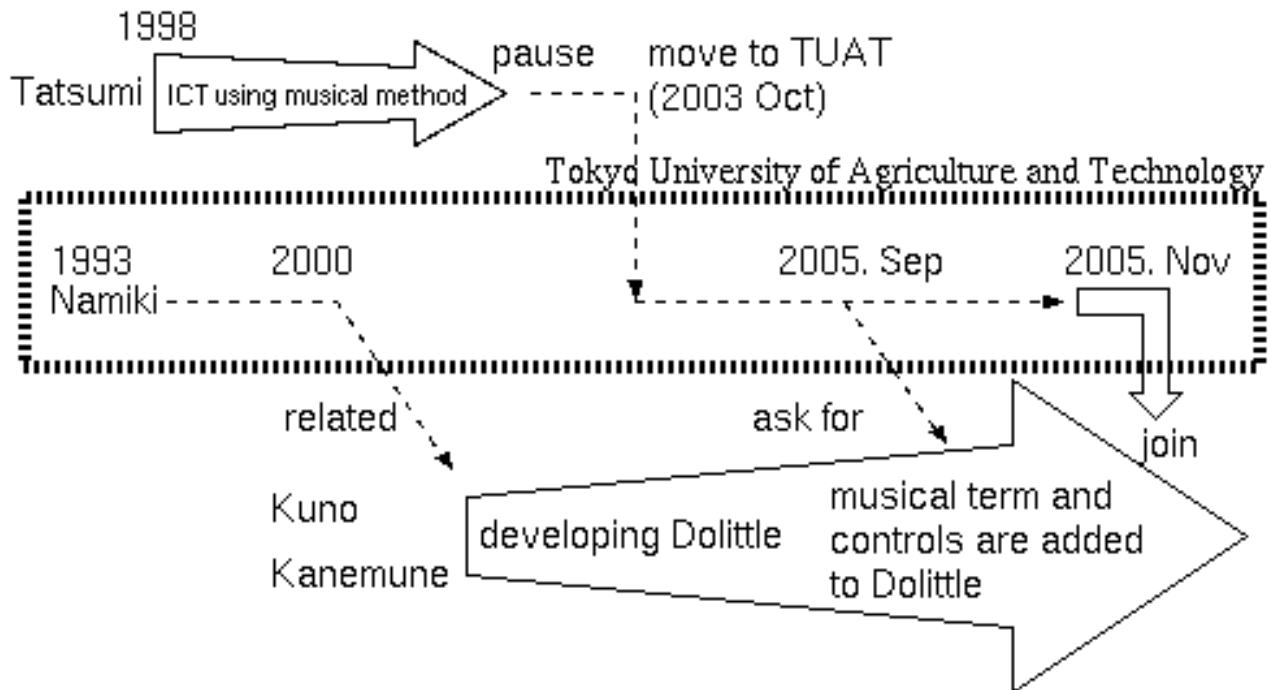


Figure 2: The history of our projects

Should programming be included or not?

After 1990's, in Japan, there were many discussions about the adoption of K12 subject programming. Keyboard typing, mouse handling, word processing, spreadsheet, presentation, e-mail, web, media literacy, ethics, copyright. Which subject have priority? Or, what subject have priority? Also there were many discussions about the adoption of programming as K12 subject. October 2005, Japanese universities and high schools teachers discussed about this topic on IPSJ symposium "now and future of high school ICT education."

The goal of ICT education

The committee in the ministry of education whose goal is researching the plan of ICT education of K12 reported "For the future ICT education in structural curriculum." The report said the goals of ICT education are put in three sections. First is "literacy of information", second is "understanding information with science method", and third is "behaviour in participating informationalized society."

IPSJ's project

Many researchers of computer science and computer technologies think "understanding computer's conduct" is helpful for achievement of three goals. For example, computer architecture, mathematical rules of computing, computer software, and the role of protocol

harmonization. We and my co-researchers - Prof. Kuno, Prof. Kanemune and many - think CPL is essential in ICT education. In SIGPS (Special Interested Group on Primary schools and Secondary schools education) which is the project committee to research of IPSJ (Information Processing Society of Japan), Prof. Kuno is the chief and Tatsumi works at the promotion. Before 2003, we made sample textbooks. After 2003, some members made real textbooks in some publishers which are certified by the ministry of education. We continuously make sample tests, visit many schools, have experimental lessons in various kind of schools. SIGPS is relating to many projects outside of IPSJ.

Programming is important in ICT education!

People who do not have a knowledge of computer programming tends to think that a learning of computer programming is worthless for the goal of ICT education. Some people say adoption of programming have difficulties because the total hours is on the decrease in Japanese K12 schools. Many teachers of mathematics, physics, chemistry, art, music and physical exercises are losing their hours in K12 schools. They may complain to the adoption of programming CPL in K12 schools.

Musical method in CPL education

Music is good subject for scripting education in K12. The reason is as follows.

- Music is independent from another subject.
- Student can find bug in playing easy.
- Music include almost technique in computer programming.
- Music is joyful for study

Structure of Music/Score

There are many typical structures in music. For example, in classic music, we can enumerate following forms.

Micro structures of Music/Score

Correct sequences of notes



Figure 3: notes forms in the correct sequences

repeats



Figure 4: repeat notes

repeats within few measures

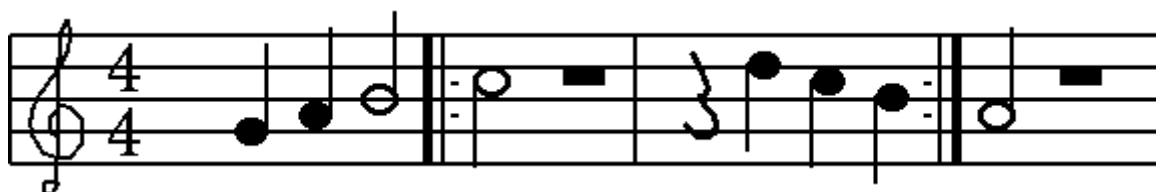


Figure 5: repeat in two measures

jump to the starting point and stop.

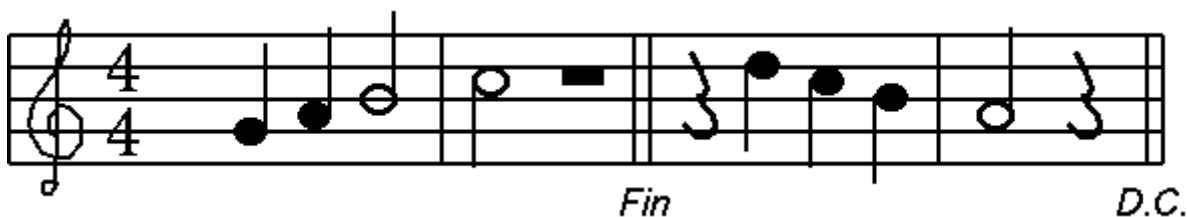


Figure 5: D.C. and Fin. Jump to starting point and stop.

branching some forms of notes by visiting times.

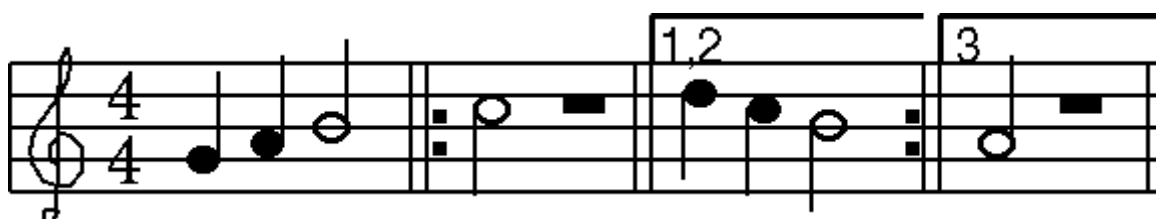


Figure 7: playing different measures by visiting times

Macro structures of Music/Score

1. Tripartite form, for example

- A-B-A
- A-B-A-C-D-C-A-B-A

2. Rond form, for example

- A-A-B-A'
- A-B-A-C-A-B-A-C'

- A-B-A-C-A-C'
- A-B-A-C-A-D-A

3. Sonata form, for example

- proposal: 1st theme - bridge - 2nd theme
- disclosure: varieties of main theme
- re-propose: 1st theme - bridge - 2nd theme - fin.

You can find "repeat in score" as follows.

- essential in playing
- D.C. and Fin.
- D.S. and Coda

Synchronizations in Score is appear in

- harmony in a code
- many instruments
- lyrics and music

What is important for education of scripting?

We think that understanding the structure of scripting/programming is important because musical score system involve many controls. Students will know that any computer read only the script you wrote. So our proposal is that musical method in education of CPL must be included to teach programming for music teachers and to teach musical scripts for ICT teachers.

Dolittle programs for music

In this section, we describe the music specification in Dolittle.

Turtle Object

A typical program of Dolittle is shown as follows.

```
Crush = Turtle ! create.
Crush 100 forward.
```

Basic style of Dolittle programming is:

1. create object and name/nominate object
2. send a message to the object

Melody Object

Old version of Dolittle has several object name. Melody object is used as follows.

```
Twinkle = Melody ! create.
Twinkle ! "DO DO SO SO RA RA SO -" add.
MyInstrument = Instrument ! "Piano" create.
MyInstrument = (Twinkle) ! set.
Instrument ! Play.
```

In first line, create melody object and name "Twinkle". Add short melody to the object and, play the melody.

Reputation

```
'Score ! "DO DO SO SO RA RA SO -" add.' ! 2 repeat.
```

Control

```
x = 1.
'
  if 'x == 1' ! then
```

```
Score ! "FA FA MI MI" add.  
else  
  Score ! "RA SO FA MI" add.  
do.  
  x = x + 1.  
'Score ! "RE RE DO -" add.'  
' ! 2 repeat.
```

The definition of melody

In music subject, there are three types of melody.

- very short melody motif
- short melody phrase, theme
- long melody Score

In old Dolittle, there were confusions of the definition of 'melody'.

Description of Player(s)

In old Dolittle, there are few strange usages in description of music player(s).

1. Score created as melody
2. set score to instrument(s).
3. send 'play' message to instrument(s)

For example, in old Dolittle, you will wrote,

```
MyInstrument=instrument! "Piano" create.  
instrument ! play.
```

So we propose new usage.

1. Score created as melody
2. set score to MyInstrument
3. send 'play' message to MyInstrument

```
MyInstrument=instrument ! "Piano" create.  
MyInstrument ! play.
```

Synchronization

Old Dolittle had a waiting function for only the timer waiting. Student can not write a program using synchronization between music and display. We proposed a waiting function for the end of playing the part of score. Using new Dolittle, you can make "Dolittle KARAOKE" in synchronized lyrics and playing.

Experiences in schools

We had two experiences in real school. First one is at a elementary school in Chiba city held by Mr. Sato, Mr. Nishigaya, Mr. Aoki, Mr. Kurabayashi and Mr. Hara. Second one is at a junior high school in Matsusaka city held by Mr. Idosaka, Mr. Kuno, and Mr. Kanemune. Using old Dolittle, pupil must write at least five lines and that was difficult for them.

```
Twinkle=melody ! create.  
Twinkle ! "DO DO SO SO RA RA SO - " add.  
MyInstrument=instrument ! "Piano" create.  
MyInstrument ! ( Twinkle ) set.  
instrument ! play.
```

Example 1 of New Dolittle

Using new Dolittle, pupil can write only one line to complete score.

```
MyInstrument!" Piano" create
"DO DO SO SO RA RA SO -" add play.
```

In this exercise, all pupils wrote musical score in Dolittle.

Example 2 of New Dolittle

Control and Reputation are treated as follows.

```
Twinkle = melody !create.
Twinkle ! "DO DO SO SO RA RA SO -" add.
Twinkle ! "FA FA MI MI RE RE DO -" add.
Twinkle ! "DO DO SO SO RA RA SO -" add.
Twinkle ! "RA SO FA MI RE RE DO -" add.
Twinkle ! "DO DO SO SO RA RA SO -" add.
Twinkle ! "FA FA MI MI RE RE DO -" add.
Twinkle ! play.
```

convert to

```
MyScore = melody !create.
x=1.
'|x|
MyScore ! "DO DO SO SO RA RA SO-" add.
'x==1'!
    then 'MyScore ! "FA FA MI MI" add.'
    else 'MyScore ! "RA SO FA MI" add.' do.
x=x+1.
MyScore ! "RE RE DO -" add.
' ! 3 repeat.
MyScore ! play.
```

Example 3 of New Dolittle

In this program, pupils had understood the concept of "array" and the randomizing function.

```
Okinawa-scale = array ! create.
Okinawa-scale ! "Do" insert
    "MI" insert
    "FA" insert
    "SO" insert
    "SI" insert.
my-Score=melody !create.
'my-Score
    ! ( Okinawa scale ! (random(5)) look-up) added' ! 16 repeat.
```

After experiments

After this experiments, some girl pupil became enthusiastic in programming after musical method. And few girl pupils answered "I like musical method." Musical method seems to be better for girl students than boy students.

Conclusion

In this paper, we described these topics.

1. Education of CPL is important for K12 students to fill the gap in school and social skills.

2. Musical method in ICT has good properties for K12 which is joyful, easy to verify, structural - continues, control, repeat, synchronization.
3. Dolittle is good CPL for ALL beginner students.
4. New Dolittle is designed for musical method programming
5. Tested result shown that all pupils can script score. Some girl pupils became to like programming after musical method.

Planning next examination

We and my co-researchers are now conducting the plan of musical method of ICT education in this school year with helping the grant by YAMAHA Music Foundation. In this examination, we will prepare some placement tests, questionnaires and post tests.

References

- TATSUMI Takeo and Kakehi Katsuhiko. (1998) *What subject of EPL should be taught in high school(In Japanese)*, In the proceedings of IPSJ Summer Programming symposium, pp.55-66.
- TATSUMI Takeo. (2001) *Musical Method in ICT education(In Japanese)*, In the proceedings of IPSJ research report on Computer and Education, Vol.2001, No.101, 2001-CE-61, pp.39-46
- Adrian Pitts. (2001) *How computers can help children learn music*. In the web article of Computers & Music in Schools, Parents' Music Room, on BBC.CO.UK.

Musical abstractions and programming paradigms

Erich Neuwirth, erich.neuwirth@univie.ac.at

Didactic Center for Computer Science, Faculty of Computer Science, University of Vienna

Abstract

Many educational programming environments (e.g. Imagine) have facilities to enrich programs with music. In most cases, this programming constructs allow to play simple melodies. It is not easily possible to play more complex music either containing chords or polyphonic structures. On the other hand, some tools (like MSWLogo and FMSLogo) implement low level MIDI commands. These commands allow talking to the sound producing device directly, but the code needed to produce music this way conceptually is very remote from the usual representation of music. Therefore, we will study some conceptual abstractions for implementing music with two different tools. These tools are a toolkit for making music with LOGO and a toolkit for making music with Microsoft Excel; both toolkits are available for download.

Since the programming concepts are quite different in spreadsheet tools like Excel and functional programming languages like LOGO, it is interesting to see how core ideas of music representation can be implemented in different programming philosophies.

To illustrate this, we can describe the topic of this paper as a discussion of the question how one should connect the following standard representation of a musical phrase:



with these two representations

Excel

0	note	1	60	500
500	note	1	62	500
500	note	1	64	500
500	note	1	67	500
500	note	1	72	250
250	note	1	67	250
250	note	1	64	250
250	note	1	62	250
250	note	1	60	1000

or

```
[ [ 0 [ note 1 60 500 ] ]
[ 500 [ note 1 62 500 ] ]
[ 500 [ note 1 64 500 ] ]
[ 500 [ note 1 67 500 ] ]
[ 500 [ note 1 72 250 ] ]
[ 250 [ note 1 67 250 ] ]
[ 250 [ note 1 64 250 ] ]
[ 250 [ note 1 62 250 ] ]
[ 250 [ note 1 60 1000 ] ]]
```

and how these representations can be used to understand musical structure and how much knowledge of musical structure is necessary to convert between these representations.

Keywords

music, programming, musical structure, functional programming, spreadsheets

Basic representation of musical elements and MIDI

Musically speaking a tone is an event characterized by starting time, pitch, volume, and duration. A tone also has tone color or timbre, which is determined by the voice or instrument producing the sound. One standard way of producing notes with computers is MIDI (=Musical instrument digital interface). MIDI defines a set of command understood by sound producing devices (e.g. the MIDI synthesizer included in Microsoft Windows on in Apple Quicktime). The MIDI command set has commands `noteon` and `noteoff`. This already shows a problem: a musical note is not represented by one MIDI command, but by two MIDI commands. To be able to play different timbres, MIDI allows to select from 128 different instruments. MIDI also allows different instruments to play simultaneously. This is implemented with MIDI channels. There are 16 channels, and each channel is assigned an instrument. `noteon` and `noteoff` then need 3 (or 2) parameters: channel, pitch, and (only for `noteon`) volume. In our toolkit this is implemented similarly in Excel and in LOGO. LOGO (at least in some aspects) is a procedural programming language, therefore it is executing commands in time. Using `noteon`, `noteoff` and the additional command `waitmilli` we can implement songs in the following way:

```
to song1
(foreach [ 60 62 64 67 72 67 64 62 60] ~
 [500 500 500 500 250 250 250 250 1000] ~
 [noteon 1 ?1 1 waitmilli ?2 noteoff 1 ?1])
end
```

From the musical view this is not a good representation since the two main properties of notes, pitch and duration, are contained in separate lists. Therefore, here is a representation closer to the concept of scores (combining the values for pitch and duration)

```
to song2
foreach [[60 500] [62 500] [64 500] [67 500]
 [72 250] [67 250] [64 250] [62 250] [60 1000]] ~
 [noteon 1 first ? 1 waitmilli last ? noteoff 1 first ?]
end
```

The basic MIDI commands do not deal with the duration of the sound, instead, the `noteoff` command has to be delayed (by `waitmilli`) after the `noteon` command by the duration of the sound. A reasonable analogy is that these commands tell a player either to hit a key and let it pressed or to release a key. The program itself is in charge of timing the commands.

For simple monophonic melodies this does not pose a problem since in such cases the end of one note coincides with the beginning of the next note. The musical expression for this kind of phrasing is legato. Playing the initial phrase staccato (with silence between the notes) like this



cannot be achieved by changing the representation. To play our motif this way, the program needs to be changed to separate duration of a note and delay until the start of the next note.

Instead of doing this, we could define a new command,

```
to note :channel :pitch :volume :duration
noteon :channel :pitch :volume
wait :duration
noteoff :channel :pitch
end
```

But in this kind of representation, we do not have an easy way of keeping track when the next note should start sounding. Therefore, we change our representation. We give the timing of the start of each note as the delay since the last note was started. This allows us to write a list of commands like

```
[[0 [note 1 60 1 250]] [500 [note 1 62 1 250]]
 [500 [note 1 64 1 250]] [500 [note 1 67 1 250]]]
```

consisting of time stamped notes. This “score” cannot be played directly since it is not a sequence of commands with wait statements placed between the commands, but a list of time stamped events. Our LOGO toolkit has a command `play.stream` which accepts lists like this one and plays the music thus described. This command takes care of all the internal details. We might say that `play.stream` is an interpreter for the simple musical language we just defined. Our language consists of time stamped commands, and the timestamp indicates the delay between the execution of two successive commands. At the moment, we only have one command, `note`, with 4 parameters, channel, pitch, volume, and duration.

It is worthwhile noting that the simple musical model—one note smoothly goes into the next note—is quite common for simple songs or melodies played by simple instruments like flutes. Only when we try to deal with more complex musical structure, we also have to change the representation for playing music programmatically.

`play.stream` also understands the more simple commands `noteon` and `noteoff`, so to hear the same 4-note melody as previously we also could do something like

```
[[0 [noteon 1 60 1]] [250 [noteoff 1 60]]
 [250 [noteon 1 62 1]] [250 [noteoff 1 62]]
 [250 [noteon 1 64 1]] [250 [noteoff 1 64]]
 [250 [noteon 1 67 1]] [250 [noteoff 1 67]]]
```

If we wanted to change the staccato mode of the notes to legato, however, we would have to change both the durations of the notes and the delays before successive notes are played. Therefore the representation with `note` is closer to the musical concepts that a melody is mainly characterized by the delays between the beginnings of notes; the duration of the sound itself is a second order effect creating variations of the same melody. Changing the mode using the `note` command is easier; in that representation, the delays (or time stamps of the note events) do not need to be changed to change the melody mode between legato and staccato. We will call such a list of timed midi events a *midistream* in the rest of this paper.

Our Excel toolkit allows the same way of describing music. Each row defines a musical event which might be `note`, `noteon` or `noteoff`. The number in front of the command name is the delay since the last event, and the numbers to the right are channel, pitch, volume, and duration (if necessary). Therefore, we can describe monophonic melodies in LOGO and in a spreadsheet in a very similar manner. To play a song we just select the spreadsheet range with the note event descriptions and click a button on an additional toolbar.

The spreadsheet model offers an additional advantage. The duration values can be expressed by formulas referring to a base duration, and so by changing the content of just one cell, we can experiment with the musical properties of a melody. This means that the spreadsheet contains a parameterized score of the song. If it is set up the right way, it is easy to change different aspects of songs. In LOGO, we would have to write more complicated functions to transform our legato song into a staccato song, or we would have to write the modified version as a completely new program. Alternatively we could write a program which produces a parameterized version of our melody.

```
to melody :basetime :baseduration
output `[
  [0 [note 1 60 1 ,[:baseduration]]]
  [[:basetime] [note 1 62 1 ,[:baseduration]]]
```

```
[,[:basetime] [note 1 64 1 ,[:baseduration]]]
[,[:basetime] [note 1 67 1 ,[:baseduration]]]
]
end
```

Using this function we can create a playable melody by specifying basetime and baseduration.

```
show melody 500 250
```

```
[[0 [note 1 60 1 250]] [500 [note 1 62 1 250]]
 [500 [note 1 64 1 250]] [500 [note 1 67 1 250]]]
```

This function uses the backquote mechanism (the character ` in front of the first square bracket) to output a list with some of its elements replaced by parameters. This mechanism is a higher order concept available in UCBLLogo and its derivatives, and it allows us to write programs translating between different representations of our music easily, but at the price of needing to understand some rather abstract concepts.

Playing with aspects of songs therefore can be done in a much more informal way in spreadsheets than in LOGO.

As we will see, some other aspects of complex musical structures can be handled more naturally in LOGO.

Ratner (1983) gives a very readable account of the different musical “dimensions”. It does not connect these concepts to computers and programming, but the very structured presentation makes it easy to adapt aspects of musical structure to programming.

Polyphony and parallel execution

Now let us play our phrase as a canon, with the second voice delayed by 1 bar. A simple musical representation is the following:



We see immediately that writing a LOGO program using `noteon` and `noteoff` is not a very convenient way because in such a program the commands have to be arranged in time order, and that does not express the musical structure. Using `note` works better, since in that case the idea of the concurrently playing voices can be expressed more easily. We also note that normally we would write the score of our two voice canon differently, namely like this:



This score much clearer expresses the idea of a canon: time shifted copies of identical voices. In our Excel toolkit, this is implemented in the following way: to play a range defining a melody or a musical phrase the range has to be selected and then placed in a buffer by pressing a button on a toolbar. When a buffer is filled, it keeps track of the duration of the whole phrase and when another phrase is added, it will normally place it just behind the current contents of the buffer. An additional button on the toolbar, however, allows us to reset the current time, and then the new

phrase will be added relative to the starting point of the whole buffer. So we can create a copy of the range with the original melody (by using spreadsheet formulas) and then change only the very first delay (before the first note). Adding this modified copy of the melody to the buffer will create a playable representation of the canon.

Here is our melody in a LOGO midistream representation.

```
to ourmelody
output [[0 [note 1 60 1 500]] [500 [note 1 62 1 500]]
      [500 [note 1 64 1 500]] [500 [note 1 67 1 500]]
      [500 [note 1 72 1 250]] [250 [note 1 67 1 250]]
      [250 [note 1 64 1 250]] [250 [note 1 62 1 250]]
      [250 [note 1 60 1 1000]]]
end
```

In LOGO, we need to use a list manipulation function to create the timeshifted copy of the melody. A reasonable way is a LOGO function with two arguments, a melody and a timeshift, outputting the timeshifted copy of the melody. `play.stream` accepts more than one stream arguments, therefore something like

```
(play.stream ourmelody timeshifted ourmelody 4000)
```

will play the canon (our dialect of LOGO needs parenthesis if a non-standard number of arguments is used).

This function is easily implemented in the following way:

```
to timeshifted :midistream :delay
output fput fput first first :midistream + :delay ~
      butfirst first :midistream ~
      butfirst :midistream
end
```

In both our implementations—Excel and Logo—we still play the voices on the same MIDI channel. As a consequence, both voices are played on the same instrument. If we play the timeshifted copy of the melody on a different channel, we can

Therefore, it is helpful if we are able to play the different voices on different channels. In Excel, this is accomplished easily by changing the column containing the channel since all the time all the details and all the data are accessible immediately.

In LOGO, this creates the necessity to reassign all timed MIDI events in a midistream from one channel to a different channel. Since we have to cope with different list nesting levels, it is useful to have some functions dealing explicitly with our midistream data structure. Generally speaking a midistream consists of a sequence of timed MIDI events

```
[[time1 [eventname1 channel1 dataa datab]]
 [time1 [eventname1 channel1 dataa datab]]
 ...]

to timestamp :timedmidievent
output first :timedmidievent
end

to channel :timedmidievent
output first butfirst first butfirst :timedmidievent
end

to eventname :timedmidievent
output first first butfirst :timedmidievent
end
```



```

to params :timedmidievent
output butfirst butfirst first butfirst :timedmidievent
end

to make.timed.midievent :timestamp :eventname :channel :params
output list :timestamp (sentence :eventname :channel :params)
end

to rechannel :timemidievent :oldchannel :newchannel
output make.timed.midievent timestamp :timedmidievent ~
      eventname :timedmidievent ~
      ifelse (channel :timedmidievent) = :oldchannel ~
        [:newchannel] [channel :timedmidievent]
      params :timedmidievent
end

to rechanneled :midistream :oldchannel :newchannel
output map [rechannel ? :oldchannel :newchannel] :midistream
end

```

The function `rechanneled` will reassign all events on a given channel to a different channel, and therefore we now can define the following musical performance:

```

instrument 1 1
instrument 2 9
(play.stream ourmelody (timeshifted (rechanneled ourmelody 1 2) 4000))

```

Another musically very important concept is transposition. Transposing a melody in our representations is very simple; we just have to add the same constant to all pitches. Pitches always are the first element after the channel for `note`, `noteon` and `noteoff` events. Therefore, we can easily define transposition by using our helper functions.

```

to transposed.event :timedmidievent :offset
if not (or (eventname :timedmidievent) = "note) ~
  (eventname :timedmidievent) = "noteon) ~
  (eventname :timedmidievent) = "noteon) ~
  [output :timedmidievent]
output make.timed.midievent timestamp :timedmidievent ~
      eventname :timedmidievent ~
      channel :timedmidievent
      fput :offset + first params :timedmidievent
      butfirst params :timedmidievent
end

```

```

to transposed :midistream :offset
output map [transposed.event ? :offset] :midistream
end

```

`rechanneled` and `transposed` both use `map`, which is the LOGO way of applying a function to each element of a list. The important concept is that we define function handling single midi events and then apply these functions to all elements of a midistream using `map`. Harvey (1997) gives a detailed account of how to introduce these higher order functions in a didactically sound way.

The same technique also can be applied to apply other transformations to our midistream. We can speed up or slow down melodies by manipulating the timestamps by multiplying them with a constant larger or smaller than 1.

Computer science and programming abstractions and music

In the section on basic representation we represented songs and melodies as tuples of numbers. From the computer science point of view, this introduces the concept of lists in a natural way. These numbers could be used as parameters for programs containing loops and executing basic MIDI operations. The introduction of polyphony shows that this simple representation is not good enough for musical purposes. Therefore, we created new abstract data representations, timed MIDI events, and sequences of timed MIDI events. Manipulating these data representations created the need for functions to easily access the components in a way making sense musically. Therefore, we defined accessor functions and constructor functions giving musically relevant components of the data representations. So we created abstract data types for creating and manipulating music. The concepts and tools illustrated in this paper can be extended. MIDI also has facilities for putting sound sources in different locations in space and even for moving sound sources around. Therefore, by extending the toolkit, we can implement moving marching bands or moving emergency vehicles. The emergency vehicle project may even be extended to include the Doppler effect for the vehicle passing by.

Creating and manipulating seems to be a good way of motivating students to deal with abstract concepts in computer science, and projects with students in teacher training programs for computer science at the University of Vienna seem to indicate that this kind of work can be successful.

A very important aspect of our project is the implementation both in the spreadsheet paradigm (using Excel) and the functional programming language paradigm (using LOGO). In many cases, a first implementation of a musical concept is easier in the spreadsheet model, since one only needs to change a few spreadsheet formulas. The formulas, however, do not make it easy to recognize the musical ideas when the ad hoc implementation is finished. Implementing the musical ideas as functions in a programming language is more tedious at first, but then the functions are reusable and can also serve as role models for other functions targeting different musical aspects, but still working with the abstract music data types we defined. Trying to implement some more advanced projects—for example playing chords—with these toolkits makes the advantage for better support for abstraction in a full programming language more easily understood.

Experience also seems to indicate that given both implementations the students understand the merits of a structured approach when the projects reach a higher level of complexity.

The complete toolkits and more examples can be downloaded from
<http://homepage.univie.ac.at/erich.neuwirth/eurologo2007/>

References

- Brian Harvey (1997), *Computer Science Logo Style*, vols 1-3, MIT Press, Cambridge
- Imagine, *Educational computer environment* by Kalas et al. available from
<http://www.logo.com/cat/view/imagine-secondary.html>
- Erich Neuwirth, *Music as a Paradigm to Teach Computer Science Principles*, Information Technologies at School (Proceedings of the Second International Conference "Informatics in Secondary Schools, Evolution and Perspectives), Vilnius, 2006.
- Leonard G. Ratner (1983), *The musical experience*, W. H. Freeman, New York.

Another look back and forward

Richard Noss, *r.noss@ioe.ac.uk*

London Knowledge Lab, Institute of Education, University of London

Abstract

Some fifteen years ago, Celia Hoyles and myself wrote a paper with the title "Logo and the learning of Mathematics: Looking back and looking forward". A somewhat foolhardy title – looking back may be relatively unproblematic, but looking forward is notoriously difficult, especially when technology is involved. Still, rereading the paper for Eurologo has catalysed some thoughts about what has actually happened in the intervening fifteen years, and what might happen next.

In fact, much has happened to the evolution of technology in learning, and it is not uniformly bad! At least on the software front, Logo itself, in its new manifestations (NetLogo, Imagine, Scratch etc.) have appeared and are in the process of becoming embedded in at least some school and college curricula, and – already in the short life of Scratch – outside.

Nevertheless, I do not intend to focus much on the evolution of the technology. Instead, I will structure my remarks around three interesting themes that centre around the cultural embedding of the technology over the last fifteen years, and see how this analysis might point to the future, using – but not exclusively - my own research as it has evolved and is now evolving.

The first theme is the question of knowledge. How is the knowledge associated with Logo and logo-like systems changing as a result of their use, and reciprocally, how is the evolving vision of what counts as knowledge in the twenty-first century shaping the ways that the technologies are socially constructed?

The second theme is the emergent construction of a technical 'literature': things to 'read' and interpret rather than the only things to build – a broadening of the notion of constructionism that I think is probably overdue. And finally, I want to consider the implications of connectivity – it is striking how, in 1992 when the web had not yet been born, we conceived of students' activities as essentially solitary, even though we made tremendous efforts (for the time) to exploit the potential of discussion and collaboration.

There is an old joke about mathematicians: there are 3 kinds – those who can count, and those who can't. So forgive me for adding one final theme to the list – computational intelligence. Throughout the last fifteen years I have steadfastly maintained a strong prejudice against attempting to incorporate computational intelligence into the culture and technology of learning, on a variety of grounds that I won't elaborate in this abstract. Now, perhaps, is the time to reconsider this prejudice, and I will share our plans for a new project at the London Knowledge Lab that is about to start. It may be that we are premature, and that the time for computational intelligence is still to come. But for sure, if we look forward far enough, it will come – and perhaps it is better to enlarge the constructionist vision to incorporate and shape it, rather than letting it overwhelm us.

Logo practice: from “turtling” to interactivity

Márta Turcsányi-Szabó, *turcsanyine@ludens.elte.hu*

Dept. of Media & Educational Technology, ELTE University, Hungary

Attila Paksi, *pakati@hotmail.com*

Ph.D. student at ELTE University and teacher at Radnóti Secondary School, Hungary

Abstract

Programming is a subject that is taught by informatics teachers in Hungary, but the same might well apply elsewhere. Even though the government decree stresses the integration of subject areas within project work, it is very difficult for teachers to fulfil these requirements. Their professional ambitions concentrate on concepts of computer science and they do not have adequate experiences in merging other subject areas with their own.

Learning becomes a life long practice if exercised through adequate motivation. If clear-cut programming concepts fail to boost interest by themselves, then inspiring topics of designing interactive stories or games and modelling natural phenomenon might be a more valued approach as they connect better with the everyday experiences and interests of children.



Figure 1. An example story: *Sleeping Beauty*

The paper discusses the attitude of teachers and children towards Logo “turtling” and proposes a conversion towards more interactive topics which could tackle the same subject concepts in a different “coat”. This keeps motivation high enough to make learning an enjoyable experience. Networking teachers and learners can provide a helpful environment that boosts creations and develops Communities of Practice.

Keywords

Logo programming; National Curriculum; interactive self-expression; game development

Logo practice within the National Curriculum

Hungarian National Curriculum as of 1995

Since the launch of personal computers into schools way back in 1982 and the introduction of the well debated Hungarian National Curriculum (130/1995 government decree), which became compulsory in 1998, informatics classes were treated mainly as the platform for teaching programming, see *Table 1.* (Turcsányi-Szabó, Ambrusztter, 2001).

	Till end of 6th grade	Till end of 8th grade	Till end of 10th grade
Algorithms	Composing algorithms in text, diagrams and their understanding.	Developing algorithms through text and diagrams. Coding a simple algorithm.	Developing algorithms and coding. Knowledge of a few commands in a programming language.

Table 1. Material to be taught within Informatics – extract (130/1995 government decree)

Although the decree suggested integration into other cultural fields expressing: “Educators of all subjects have to transmit materials taking several common concepts into consideration, among others the craft of learning, thinking, self education, exploration and problem solving using all available tools” (130/1995 government decree). Several publications pinpointed possible areas:

- “Performing events with sequences of pictures and simple animation. Preparation of illustrations that represent change, development, or a phenomenon. ... Construction of text and picture, layout techniques. ... Connection between the context, the message and visual appeal.” (Visual studies)
- “How a movie develops a composition: tools for constructing meaning. Organising visual and sound elements in space and time. Fields of application of movies. ‘Information superhighway’ - interactive media.” (Movie and media Art)

The first author has presented several times at conferences for teachers expressing the need to switch from systematic teaching of programming methodology to design of educational games into which all topics can be integrated together with other subject areas. Yet, it was thought to be a far fetched idea and teachers protested, saying: “Gaming is not for school classes!”.

Hungarian Comenius Logo

Although Logo has been a legitimate choice and the Hungarian version of Comenius Logo was available from 1997 (Turcsányi-Szabó, Kossuth, 1997), it was treated by informatics teachers just as a first programming language “to make children believe as if they were programming”, restricting to basic commands of “turtling” and the turtle hardly ever got a colourful shape. Exceptional activities dealing with animations were highly appreciated by children (Dancsó, 2003), but treated as “just play” by most teachers. Several books on Logo appeared on the market, but most of them emphasised the programming side, even if some connections were mentioned in relation to other topics. Naturally, these could not convince teachers of any other subject areas to dive into programming topics for any reason.

As a consequence of these:

- Children interested in programming would reflect on Logo as being a “childish thing” and consider serious programming to start from something like Pascal.
- Children not interested in programming would rather choose any other topics like word processing or creating presentations, and would not be too successful in programming related areas. A large majority of girls fall into this category.

- Teachers (not professionals in IT) would not take the pain of getting to know Logo.

Hungarian National Curriculum as of 2003

The modified National Curriculum (243/2003. (XII.17.) government decree) came into effect from 2004 September, starting at elementary 1st grade, progressing then on in yearly succession, emphasised titles of integrated new media use and ICT, see *Table 2*.

	1-4th grade	5-6th grade	7-8th grade	9-12th grade
Algorithms	Recognising, composing, and executing simple algorithms. The use of simple authoring tools.	Developing algorithms for a given problem using computer. The use of simple authoring tools.	Design, development and execution of algorithms, or their elements for a given problem using authoring tools, algorithmic abstraction and the theory of stepwise refinement.	Designing, analysing, developing, and executing algorithms for a given problem. The use of typical algorithms. The use of authoring tools.
Computer aided problem solving	Practicing problem solving using ICT.	The composition of problems using ICT terms. The general concept of algorithms. Investigation of control instruments in educational programs.	ICT tools and methods needed in solving problems. Investigation of models of random phenomenon and the effect of changing parameters.	ICT tools and methods for solving complex problems. Measurements and simulations, effects of changing parameters, and composition of regularity.
Constructing, editing text & pictures	Using and developing simple music applications and animations.	Editing basic elements (text, picture, music, illustration, animation, film) of a multimedia document.	Preparing multimedia documents with (text, picture, music, illustration, animation, film).	Preparing multimedia documents with (text, picture, music, illustration, animation, film) and modifying existing files.
Media informatics	Knowing the possibilities of ICT use in new media.	The use of internet portals, text and picture based resources.	The possibilities of ICT in conventional media (book, journal, radio, film, TV) and their use in the learning process.	The possibilities of ICT in non-conventional media and their use in the learning process.

Table 2. Material to be taught within Informatics – extract (2003 government decree)

The National Curriculum stresses on integration, project activities, and group work. Besides the government decree, several frameworks of possible subdivision of cultural areas and timing for school deliveries have been published. Besides, schools can develop their own configured

version of curriculum (compliant to the government decree) that suits and serves better the image foreseen by the staff and local community in every topic of the specified cultural areas.

	7-8th grade	9-12th grade
Communication	The reconstruction of simple events (combined in time and space) into picture series. The design and reconstruction of experiences, fictive or heard events into moving images (story-board, animation, interview). Narration.	The reconstruction of complex events (combined in time and space) into picture series. The design and reconstruction of experiences, fictive or heard events into moving images (études). Complex narration.

Table 3. Material to be taught within Movie and Media Art – extract (2003 government decree)

Thus, it could be a matter of choice at the school level: how and with what tools and methods each area (like e.g. Communication in Movie and Media Art, see Table 3.) is tackled.

HÁLogo portal and mentoring



Figure 2. HÁLogo portal – Table of Content page (click on picture to access)

By that time TeaM lab managed to publish HÁLogo educational material initiated by NETLogo project (NETLogo, 1999) and the extended Hungarian version together with its supporting portal (Turcsányi-Szabó, 2004b) was funded by KOMA XLVIII (Educational Modernising Foundation - <http://www.koma.hu/>) as "Embedding computers within pedagogical methodologies" project. The material used the Hungarian version of Comenius Logo to build multidisciplinary educational microworlds and support both their use in different subject areas (even areas like e.g. Language: creating expressions and stories, interpretation of words, language driven movies; and Drama: developing action poetry, moving postcards, story slides, and movies) and their development within informatics classes (Turcsányi-Szabó, 2001). These resources were used both in teacher training and as capacity building in tele-houses, which are public ICT access centres (Turcsányi-Szabó, 2004a), as well as spread in broader scale through tele-mentoring activities of undergraduate students within our teacher training (Turcsányi-Szabó, 2006c).

Our experiences in schools and tele-houses showed that such e-learning materials are very well applicable both in classes and in autonomous learning situations, where some local help is available and virtual mentoring is acceptable.

Unfortunately the material appeared a bit too late to be used widely in education:

- Logo phobia was far too strong, which also contributed to the fact that funds were provided for publication of HÁLogo material only four years after its development and six years after Comenius Logo entered schools.
- A lot of school computers switched operating systems to XP, which could not support the existing Comenius Logo versions and Kossuth Publishing was not eager to buy the upgrade of the country licence.
- Although there are around 900 registered members of HÁLogo portal (both children and teachers) the culture of virtual communication was not alive enough to allow the mentors of our teacher training to be able to help with the use of the e-learning materials. Users (probably being too shy within this new media) preferred to just use the portal as repository of recourses and did not ask for help or advice except in very scarce cases.

However, we did experience quite a few incidences of children tackling the material by themselves or with the help of their parents, since the school teacher did not use (or even was not aware of) these materials. This strengthened our beliefs, that such activities should indeed aim at children, who could look for help locally or virtually (through our mentoring services).

Hungarian Imagine

A successful EU project and a local resource development project, as well as four years of hard lobbying was needed till a new Logo version, “Imagine” (Abonyi-Tóth, et al. 2006) appeared officially, this time free for education. The success of the *Colabs* Minerva project (2002-2004) has initiated further development of the course materials developed within *Colabs* project, which guided children in learning how to develop their own projects, to provide a full scale of increasing levels of program development by motivation through games, boosting creativity through artistic expression. It was published both in Hungarian as “Digital Literacy” (Turcsányi-Szabó, 2006a) as well as in English language as “Creative Classroom” (Turcsányi-Szabó, 2006b) by Logotron (see *Figure 3*). These materials are used both in teacher training at ELTE university and is also accessible freely for educational purposes.

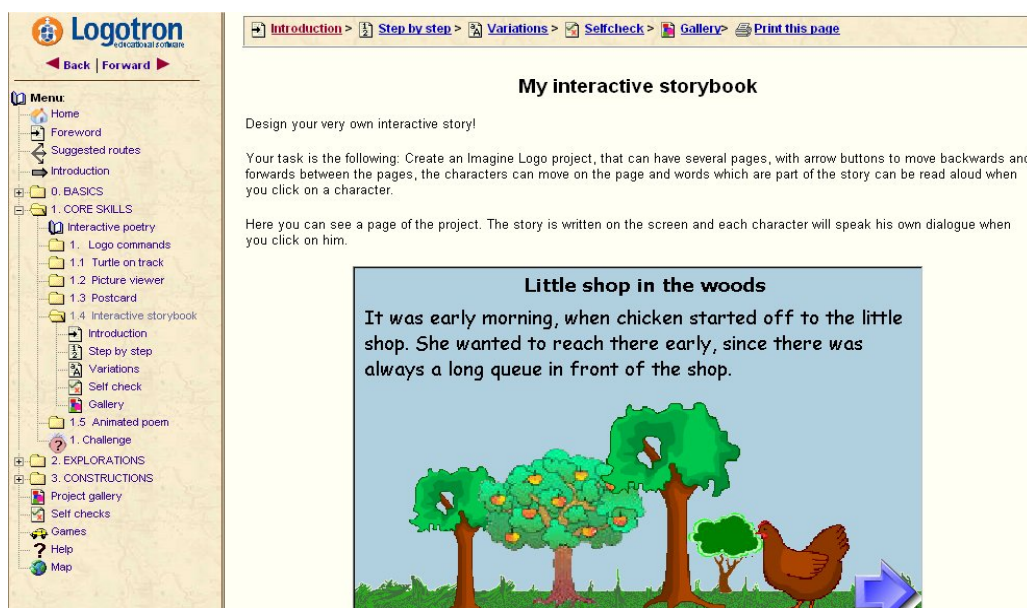


Figure 3. Creative Classroom CD – My interactive storybook unit (click on picture to view demo)

The material consists of three levels (apart from the 0 Basics level, *Core Skills, Explorations, Constructions*), where each level consists of five project approach units (developing projects mainly through direct manipulation) and one Logo approach unit (developing projects through Logo programming). Motivation is always initiated through games and simple, but interesting projects (more than 150) and units contain *step-by-step development, variations* to expand knowledge, *self checks* and a *gallery* containing new projects based on actual knowledge.

While “Creative Classroom” (which is now published on CD) is expected to be more widely distributed through a Learning Management System (LMS), “Digital Literacy” (which is now distributed through an LMS), called SDT (SchoolNet Digital Repository) is expected to have more effective use when published on a CD too. The reason is probably based on the fact, that while the schools in the UK are already functioning through the use of LMS technology (and require all learning materials to be integrated in this way), Hungarian schools are happier to access such locally, as technical facilities are not yet so perfect in all schools and the virtual *Zone of Proximal Development* (Vygotsky, 1978) is not yet a code of practice. Also, the LMS used in public education is not a perfect solution for all types of learning materials developed for K12. So, while the same material published on CD (based HTML technology) provides a highly interactive workbook of resources, SDT is less easily accessible for all age groups and needs more clicks to access individual LOs (Learning Objects), which thus become less coherent as a whole (see Figure 4.).

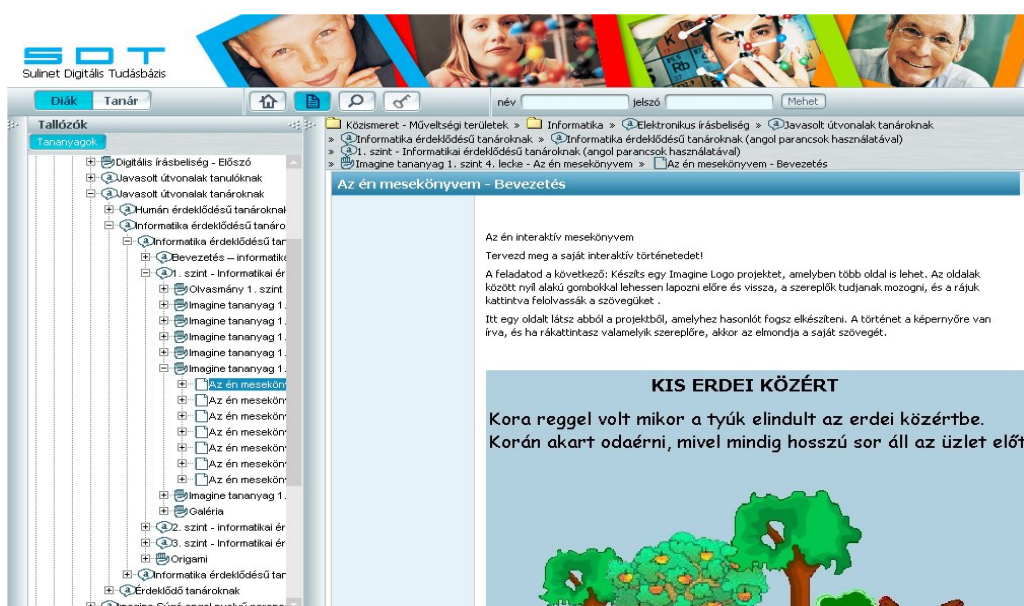


Figure 4. SDT Digital Literacy course – My interactive storybook unit (click on picture to access)

Teachers were happy to have free access to Imagine and the course material, but were unable to select suitable sets of activities that fit into the amount of hours allowed in their course work for a defined topic. We have set up the Imagine portal to give help for teachers and children in finding their way through activities with Imagine (Abonyi-Tóth, 2006), which now has more than a thousand registered users and their number increases rapidly. The portal summarises news about Imagine, projects launched, resources available and a networking area to upload projects, comment and develop dialog in the community. However this is mainly aimed at grownups and concentrates on general issues. Children would need a more focussed approach.

We have produced several such theme oriented portals for children, which concentrate mainly on visual representations and boosts creativity, like “Telling you in pictures” portal (Abonyi-Tóth, et al. 2005), that allows communication of smaller children through visuals; and “Logo

brushstrokes” portal (Rónai, Vörös, 2007), which motivates the creation of Art pieces, using some mathematics and Logo coding.

Similarly, the “Digital Literacy” course material is huge and it is difficult to have an overview of activities and isolate the necessary resources for activities, it is not really suitable for the SDT environment. Besides, a more focussed set of activities needs a portal to activate constructions through networking by allowing a real Community of Practice (CoP) to develop for the specified age group and defined themes. E.g. developing stories is a topic that is very much enjoyed by the 10-14 age group (both girls & boys) and it is a perfect topic to get into the basics of Imagine.

Self expression through stories

A lot of courses within the Informatics teacher training program are at master’s level as non-compulsory electives, which mainly relate to developing e-learning materials and running projects in public education (Turcsanyi-Szabo, 2006c). In case of the course dealing with authoring tools for children, the assignment for student teachers is mainly to develop modelling activities for different disciplines practicing constructivist pedagogy and giving interesting examples for children to start out with. Thus, in case of Imagine, student teachers had to develop stories themselves for the Gallery area, giving children ideas to develop themselves.

Imagine Story Portal

The story portal was first developed by Mátyás Szőke as assignment work for Telementoring course, which was later redesigned by the second author (Paksi, Turcsányi-Szabó, 2007) to run a research project as Ph.D. work, see *Figure 5*. The aim of the portal is to:

- Motivate the use of the existing “Digital Literacy” course material in SDT, putting Imagine into practice and evaluating its effect in school use.
- Compose a minimum set of course activities that can be tackled in any school setting extended by individual learning, which would later lead to more complex modelling.
- Providing 10-14 years old children to experience creative constructions through self-expression and quality work by developing interactive hypermedia stories, poems, games.
- Provide a portal where one can find examples (to get an appetite for creation), course material (to learn how to do), upload area (to create and publish own works), communication possibilities (to network with peers, authors and audience).
- Introduce children to activities with Imagine by teachers at school and strengthen their competencies to lead them on to autonomous learning phase, using the portal area, where they can find learning materials and also seek help if needed.



Figure 5. Imagine Story Portal – Gallery (click on picture to access)

The scope of the course material

The minimum set of course materials were put together by the second author and a trial activity with the lower age group was also conducted to see how it works and what might be missing, after which it was uploaded to the portal's course area. We had to make sure that this is indeed a minimum series of activities that would allow a child with basic computer knowledge to possess the competency needed for developing an interactive story on basic level, after which he/she can catch further bits of tricks from the SDT Digital Literacy course material to enhance own creations. Thus it is intended as a starter, which is enough for creative self expression, which could later be developed further into modelling techniques.

Interactive story competition

A competition for 10-14 aged children was finally launched in mid May, which seeks for Interactive story compositions developed with the Imagine authoring tool and we shall repeat this competition in autumn to see how it might work in school time within school classes.

Case studies

Imagine stories – initial classes

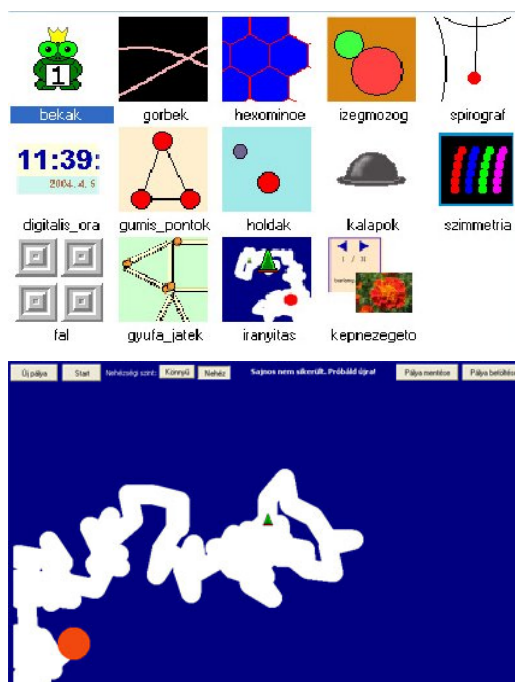
The second author teaches at the Radnóti Miklós, which is a practice school of ELTE University. There, he had a chance to try out the starting course materials after which he put together the final resources present in the portal. He taught in two classes, with 18 children each 10 years of age, mixed boys and girls. Children already possessed knowledge of basic computer use and out of the 36 children 10 had already some Comenius Logo classes in their previous school.

When they first heard that they will be dealing with the turtle, they became very disappointed, believing that there will be again “drawing pictures, fence, house, ...” like in their previous school all year. This illustrates well the attitude of this age group towards drawing algorithms.

Both classes had some activities with word processing and drawing before the Imagine topic appeared. All activities were related to stories: they either had to finish a story in a different way or start one from scratch using 10 words that were previously proposed by them. They then illustrated the story using the picture editor. It was a natural line to go over to Imagine and create stories that can not only be seen, but also heard (using the Hungarian Text-to-Speech and multimedia features of Imagine).

In order to be able to create interactive stories the following were considered as necessary knowledge: *drawing tools, inserting background, creating new turtle and configuring it, giving new shape to the turtle, using text-boxes, inserting new pages and jumping to pages, inserting new buttons and learning about their properties, changing the shape of the button, adding events to turtles and buttons.*

The starting class had to introduce Imagine to let children see the types of projects that can be developed with this authoring tool. So, the demo games were played, where boys mainly preferred logical and technical games and girls spent time with those programs that allowed creating drawings. However, both boys and girls enjoyed the *Control game*, so it was a natural task to ask children to draw backgrounds for the game (see Figure 6.). This task allowed the introduction of the first four knowledge items, which were learned at need, since children requested bits of tricks to produce quality works. By the end, everyone used backgrounds drawn by peers played each other’s games and discussed the quality and motivation of products.



The turtle moves continuously (directions depend on right or left control keys pressed) with chosen speed (depending on the chosen level of the game) but only go ahead on a connected white road, otherwise the game ends.

Figure 6. 1st row Demo games – Description of Control game – 2nd row backgrounds for Control game

As a next step (in order to stabilise this knowledge), children had to create a similar game, but instead of using control keys, they had to create buttons to control the turtle. This seemed at first for children to be a drawback, due to the use of buttons, but later they realised that this was more, as they produced the game itself (not just the background) by themselves.

The next step was to access the *1.1 Turtle on track* project within “Digital Literacy” course. This turned out to be a disaster as it took the whole class exactly 25 minutes to access the very same page due to it being deeply embedded within the course structure and the user interface did not allow a clear view of navigation (see Figure 4.). It was immediately concluded that the course material has to be extracted and presented in a more straightforward way for easy access.

The next two classes dealt with the two unit projects: *Turtle on track* and *My interactive story book*. Some children were at first bothered, why they need to do exactly as the step-by step instructions say, but it later turned out to be helpful not having to ask the teacher how to produce a new trick, since it was all written there. These last two classes allowed to finish introduction of the list of prescribed knowledge bits necessary for creating interactive stories and even gave

some time for defining events that would make the turtles move upon a click. So, children were satisfied to see that they are able to create an interactive storybook of their own interest.

In the next classes, children could create their own stories. They realised the importance of planning at the beginning, before starting out any project work, so that it would be designed properly and smoothly developed.

After these classes, one of the most interesting tasks emerged: drawing animations. Logomotion was used to produce different shape series that would suit the developing stories and children used their full creativity in producing different creations, having great fun all the way.

Later, children accessed the *Imagine Stories Portal* and enjoyed looking at the stories in the gallery, which gave them further ideas for their own creations. They realised that written text can be spoken as well, so that was their next challenge to make their texts interactive and be spoken out. They registered on the portal and learned how to install Imagine, to be able to install it on their own computer at home and show their parents their own creations.

By now all children are fluent with Imagine story creation and can produce simple animations with Logomotion.

Other Logo activities

“Evaluation of educational software” is another course at ELTE teacher training, which deals with the formative and pedagogic evaluation of educational software with respect to the National Curriculum where we often use educational microworlds and portals for evaluation that have been developed by our students in previous semesters or are parts of running projects. This year – among others – various types of Logo environments and activities have been chosen for evaluation as case studies, adding the following short comments as summaries:

- As little as 2x2 hours of activity with guidance was enough for a 10 year old girl with absolute minimum knowledge on computers (max. 15 minute daily emailing, no interest, no motivation, no knowledge of other applications) to be able to produce a 4 page interactive story with animation within Imagine. She used the course material on the *Interactive Story Portal*, finishing it alone, which developed after all the motivation to do so.
- Scratch (<http://scratch.mit.edu/index.html>) not only allows novice programmers to get acquainted with programming through game development, but also highly motivates those that already know how to program well. A good example of this is the level of motivation the two future teachers Péter Bernát and Marcell Balaton (who are both males and excellent programmers) had by also producing the Hungarian Scratch pages (Bernát, Balaton, 2007) and made their experiment (on purpose) with secondary school boys who are good programmers of at least the Pascal programming language. The networking possibility on the portal seemed to be also among the main attractions of activities.
- Secondary students (at first being disappointed to do anything with Logo) after a few hours of introductory activities joined in with great motivation to tackle problems using StarLogo (<http://education.mit.edu/starlogo-tng/index.htm>) & NetLogo (<http://ccl.northwestern.edu/netlogo/>). In both cases it seemed to be very helpful to be able to gain knowledge from an existing network and teachers found great interest in continuing to use the environment next semester too (even though both tools are only in English language).

Conclusion

It seems that inadequate practices of teachers putting only “turtling” into school activities has somehow degraded Logo into a childish thing. Why degrade Logo, when there are plenty of different variations that allow lots of interesting features to explore different subject areas, enjoy gaming, and self expression, which are all perfect subjects for learning about algorithms and programming and can be easily extended to more enhanced topics of modelling.

Exploring interactivity in an object oriented environment can be easily extended from the simple metaphors in game development (where there are actors that have defined behaviours) to more complex models, based on real life experiences.

In case of the English language cultures it is easy to choose from several resources, but much less options are available in case of less popular language areas, like Hungarian. We should make good use of localised products and developed learning materials through constructivist learning environments, which can develop effective CoP that are self-motivating and can very well increase the learning curve to produce effective communities of learners.

References

- Abonyi-Tóth A., Bodnár E., Turcsanyi-Szabo, M. (2005). "Telling you in pictures" – communication bridging languages, Proceedings of Eurologo 2005, pp. 307-312, 27-30 August, Warsaw, Poland. <http://eurologo2005.oeiizk.waw.pl/PDF/E2005AbonyiEtAl.pdf>
- Abonyi-Tóth, A. (2006) Imagine portal, at <http://imagine.elte.hu/>, TeaM Lab.
- © Abonyi-Tóth, A., Turcsányi-Szabó, M., Windisch, J. (2006). „Magyar Imagine” (Hungarian Imagine), English original (© Blaho,A., Kalas, I., Salanci, L., Tomcsányi, P.) downloadable from <http://www.sulinet.hu/>
- Bernát, P. Balaton, M. (2007) Hungarian Scratch pages, at <http://scratch.inf.elte.hu>, Team lab.
- Colabs Minerva project (2002-2004) web page: <http://matchsz.inf.elte.hu/Colabs/>
- Dancsó, T., (2003) Talent improving with the help of Logo pedagogy - ICT as reflected by interschool competitions, Proceedings of Eurologo 2003, 27-30 August Porto, Portugal.
- Rónai, O., Vörös, V. (2007) Introduction, methodological analysis and experiences of Logo brushstrokes portal, ELTE Masters thesis and portal: <http://www.ecsetvonas.ini.hu/>
- Paksi, A., Turcsányi-Szabó, M. (2007) Imagine Story Portal, at <http://meseportal.ini.hu/>, Team Lab.
- © Turcsányi-Szabó M., & Kossuth Publishing Co., (1997) "Comenius Logo 3.0", original version (© Andrej Blaho, Ivan Kalas, Peter Tomcsányi), Kossuth Publishing Co.
- Turcsányi-Szabó, M., Abonyi-Toth, A (1999). NETLogo teacher training material - Microworlds, A CD product of NETLogo MM1020 project (NETLogo – <http://www.netlogo.org> – not accessible any more).
- Turcsányi-Szabó, M. (2001) "Subject Oriented Microworld Extendible environment for learning and tailoring educational tools", Informatika Vol 1 No 37, pp. 16-27, Matematikos ir Informatikos Institutas, Vilnius.
- Turcsányi-Szabó, M., Ambrusztér, G. (2001) "The past, present, and future of computers in education – the Hungarian image", International Journal of Continuing Engineering Education and Life-Long learning., UNESCO, Volume 11, Nos 4/5/6., pp. 487-501 https://www.inderscience.com/search/index.php?action=record&rec_id=413&prevQuery=&ps=10&m=or
- Turcsányi-Szabó, M. (2004a) „Informatics teacher training in Hungary: building community and capacity with tele-houses”, ed. A., Brown & N., Davis, World Yearbook of Education 2004: Digital Technology, Communities & Education, pp.277-288, RoutledgeFalmer
- ed. Turcsányi-Szabó, M., (2004b). "HÁLogo portal" (Hungarian NETLogo portal with e-learning material), ELTE University TeaM Lab. Accessible at <http://kihivas.inf.elte.hu/halogo>
- ed. Turcsányi-Szabó, M. (2006a). "Digitális Írásbeliség" (Digital Literacy e-learning material), Sulinet Digitális Tudásbázis (Schoolnet Digital Repository), <http://sdt.sulinet.hu/>
- Turcsányi-Szabó, M. (2006b). Creative Classroom CD Logotron Ltd, Cambridge, http://www.logo.com/cat/view/creative_classroom.html

Turcsányi-Szabó, M. (2006c). Blending projects serving public education into teacher training. In Kumar, Deepak; Turner, Joe (Eds.) Education for the 21st Century - Impact of ICT and Digital Resources, IFIP 19th World Computer Congress, TC-3 Education, IFIP series Vol. 210, pp. 235-244, Springer. <http://www.springerlink.com/content/k8q6107r3gu60838/>

Vygotsky, L. S. (1978). Mind in society: The development of higher psychological processes. Cambridge, MA: Harvard University Press. Published originally in Russian in 1930. <http://www.marxists.org/archive/vygotsky/works/mind/>

130/1995 Government Regulation the National Curriculum, in Hungarian Gazette, Official paper of the Republic of Hungary 1995. No.91.

243/2003. (XII.17) Government Regulation the National Curriculum, on web page of the Ministry of Education: <http://www.okm.gov.hu/main.php?folderID=391&articleID=1478&ctag=articlelist&iid=1>

Process of educational software development

Andrej Petráš, petras@edi.fmph.uniba.sk

Dept of Informatics Education, Comenius University Bratislava, Slovakia

Abstract

This paper describes iterative development of educational software, named Cube, for mathematics in elementary school. The aim of the program is to develop students' spatial imagination. Based on our previous experiments, we created this project, in which we tried to verify our skills from educational software's development and define more accurately the whole process.

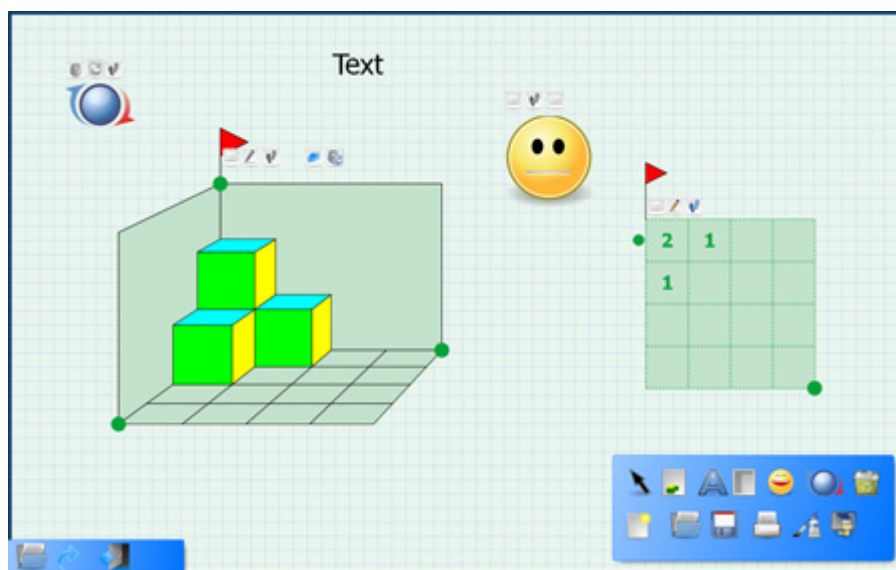


Figure 1. Screenshot of program's environment (<http://cube.easyedu.eu>)

It is a microworld for students, which is made for mathematics. Microworld gives children and students opportunities to build their own knowledge of fundamental concepts.

Many of our knowledge are achieved by co-operation. Students learn better when they can observe work of other people in their surroundings. The best conditions for learning are created when students are exposed to requirements, which extend their knowledge. Students like internet and ability of communication, changing of information and presenting themselves on the internet.

Keywords

computer literacy, educational software, action research, methodology of software development, experiment, project

Introduction

Our research's aim is the development of educational software. We would like to find out what methodology and methods are needed to use in order to obtain quality, good worked, but created in a short time and the cheapest software product and how this kind of development and software shape the cognitive process. Life in modern informative society and effective applying of the knowledge economy (see UNESCO) require modern education to develop new competencies in students, like critical perception and thinking, flexibly decision, controlling of unexpected situations, effective communication and cooperation. Informative and communicative technologies are important in development of these skills as resource. Educational software and internet put expression tool into another dimension. New technologies allow children to create, work or play. It is needed to think about questions: How do the proposal and development of educational software affect effectiveness of cognitive process? How affect methodologies of software's development and pedagogical research each other to achieve effective process of educational software's development?

We applied our current skills and knowledge of educational software's development in project Cube. It is microworld for students, which is aimed to mathematics. Microworld gives children and students opportunities to build their own knowledge of fundamental concepts. According to Kalaš and Winzer (2006), when students work with microworlds, they are encouraged to encounter concepts and relations through active engagement and exploratory learning. Next chapter deals with our development in theoretical aspect.

The Development of Educational Software

After evaluating and analyzing development of previous educational software, we came to the conclusion that one of the possibilities how to develop educational software is combination of agile software methodology with action research. We will analyze both these fields of study.

During the development of some products, it gets to paradox that customer is satisfied with product, which is relatively not usable. It can be caused by fact that when the customer requires some project, he does not know exactly all the properties of product himself. This is also one of the problems of educational software's development for children. If we want to define properties of certain educational software for children we need research, therefore we try to combine this research with development of software.

Agile Software Methodologies solve this problem by connecting a customer (pupil and teacher) with a developmental team. The term Agile Software Methodology denotes a group of methodologies, which assume that the only way how to check the correctness of the proposed system, is to develop it as quickly as possible, deliver it to the customer and use the feedback for improvements, see Kadlec (2005). Our development of educational software which we have already done is closest to this methodology. One of the groups of agile methodologies, which we use, is Extreme Programming.

Extreme Programming is effective, easy, flexible and funny methodology, which allows adding new versions of program continuously. It saves time spent by creating of documents and specifications. From the view of the Extreme Programming there are four basic activities: testing, code's writing, design and customer (communication with customer).

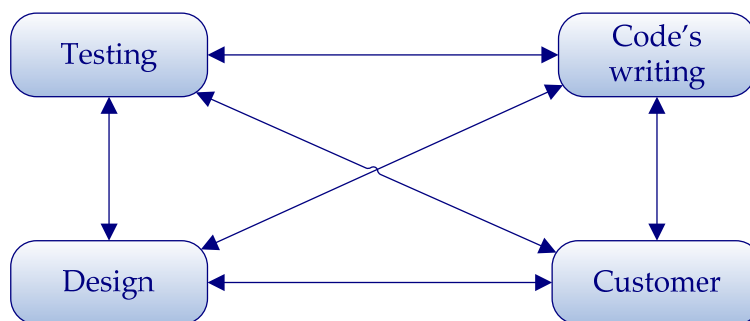


Figure 2. Interaction between elements within extreme programming, Kadlec 2004

According to Kadlec (2004), one of the most important elements of developmental team is expert, who rectifies development on the basis of his own research with children and according to this research he tries to define properties of software. By the term *expert*, we mean a person from the academic environment who deals with media for children. His/her experience and knowledge should come forward throughout the development of the whole application. The expert should have a combination of academic knowledge and experience with production of pedagogic software. By *media for children*, we mean media for print, TV, multimedia, Internet or other technology.



Figure 3. The researcher's role

In our project expert applies action research. There is a search for ways how to implement the results of research in practice and how to speed up the process of necessary changes. Action Research developed based on the critique of the traditional research, see Hendl (2005).

Action research is based on theory that effect of research is bigger, if all the people (teachers, students, employees ...), who are connected with to research, will work together on it, see Hendl (2005).



Figure 4. Action Research

The course of the Action Research follows the terrain conditions and mostly uses the methods of qualitative research. Action Research needs to flexibly react to situations and obstacles and develop at the same time. Its cycles are the reactions to new knowledge and terrain problems. New cycles help to test and advance the interpretations from previous cycles, see Hendl (2005). Action research is applied research whose properties and progress are the similar to progress of agile methodology.

In our project we determined plan of software development, we put together group of students for this project and realized our program.

Planning iterations

Our plan was to create software, which will develop students' spatial imagination. We can divide this plan into two parts:

- In the first part, we studied some materials and realized two iterations with a teacher. Based on these studies we created our first design of environment and its functionality. We put together some students and we familiarized them with project.
- In the second part, children became testers and developers of software and we realized three iterations. During these iterations we applied to software some of students' and teacher's remarks, results of our research and we corrected some software bugs.

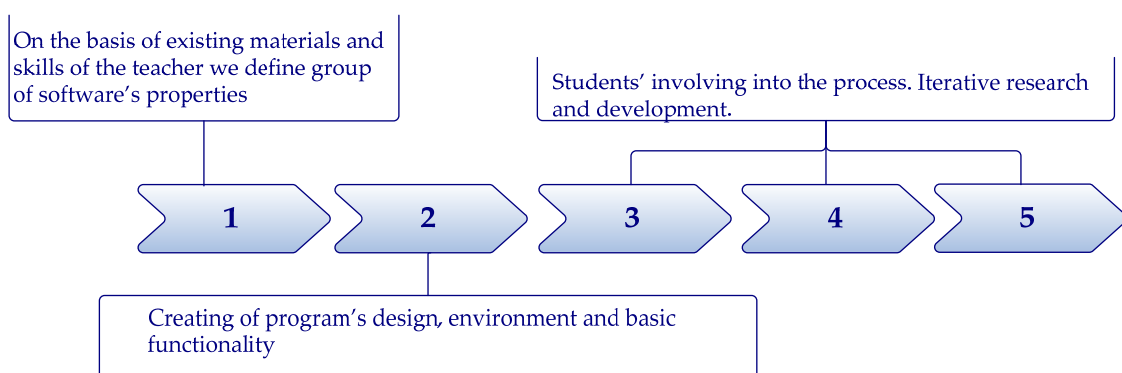


Figure 5. Planning iterations

Group of children

We involved students of fifth grade (aged 10 - 11 years) of elementary school Košická in our project. There are children, who feel like working with computer and they use it in their everyday life in the school or at home. Furthermore, they have computer training once a week at school.

In our students' group there were two boys and two girls. During the first iteration with students we had discussion about their using of computer, how and when they use computers and which of the programs they are able to use. Each the students wrote their own list of programs and activities.

We can divide students' answers into several groups:

- Programs – we involved these programs there: Imagine, LogoMotion, Anvil studio, Paint, Word, Power Point a others. Students mostly knew these programs from school and they also used them for doing their homework.
- Games – each of these students could list several games. In many cases, these games were aimed to force.
- Internet – number of answers and activities, which were written by students, was the same as those in first two groups. Many of different interacted games and activities were mentioned there. Students often search some information for their homework on the internet and they also like to communicate with other students by chats. Each of them has at least one email address and one student had his own "blog".

It is evident from discussion that students like, except games and homework, internet and ability of communication, changing of information and presenting of themselves on the internet.



Figure 6. Photo of students' group (Andrej, Jakub, Natália a Natália)

Project Cube

We analyse individual iterations of software's development.

Iteration 1 – In this iteration we had some meetings with teacher, who defined problems and her own experiences with current topic. She also proposed first software's properties based on her own experiences. She familiarized us with available schoolbooks, which are used by students.

Iteration 2 – After discussion with teacher we studied literature and we designed first environment and basic functionality. First design was draw on work books for students of elementary school. Then we began to program first version of our program.

Iteration 3 – After programming of first program's version, we involved group of students into project. Application was without sounds and it did not support editing of activities. It contained

only one task, but it was sufficient for students to create their own ideas about program's environment.

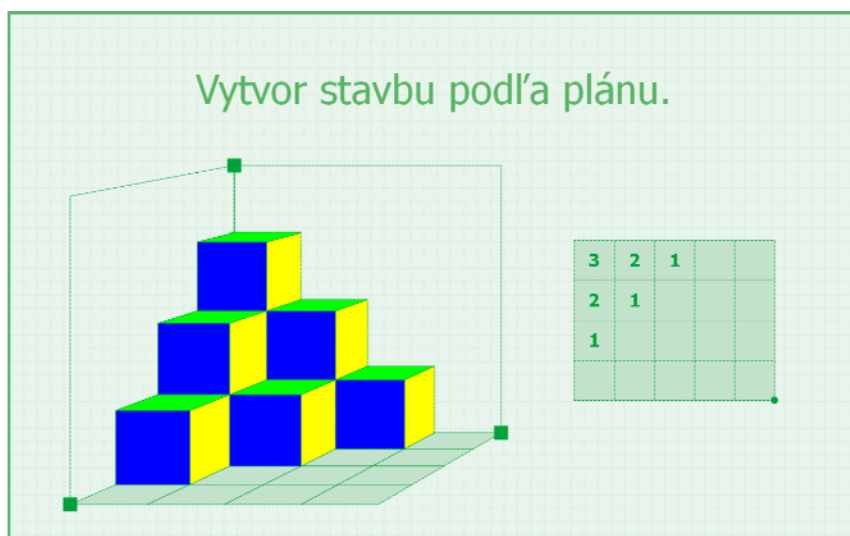


Figure 7. Screenshot of the first version

Students liked our program. We gave them papers, where they could draw their own imaginations about pictures and figures in program. Then we discussed about what and how program should do. There are some of students' notes:

- I should have possibility to choose, if I am girl or boy and according to this there would be different environment's colour.
- Program should encourage me, if I solve some task.
- I would like to make my own exercises.
- Cubes should have different colours and we could create different pictures by them.
- There could be also another shapes or models except cubes (it does not matter, that the name of program is Cube)

Some of graphical designs and designs of functionality were the same as our, but students also created unique designs, which we decide to involve into software's development.

Iteration 4 – We created version, which already contained objects for creating of activities. We could also save and load tasks in program. Some of new objects were created according to students' design. Students were able to work with program's activities quickly. After they did eight prepared tasks, they began to create their own tasks. They prove us that our tasks were easy and it was not problem to do them. Students changed and showed their own tasks among themselves. In this iteration we came to the conclusion that program would have more possibilities to be useful, if we created network version.

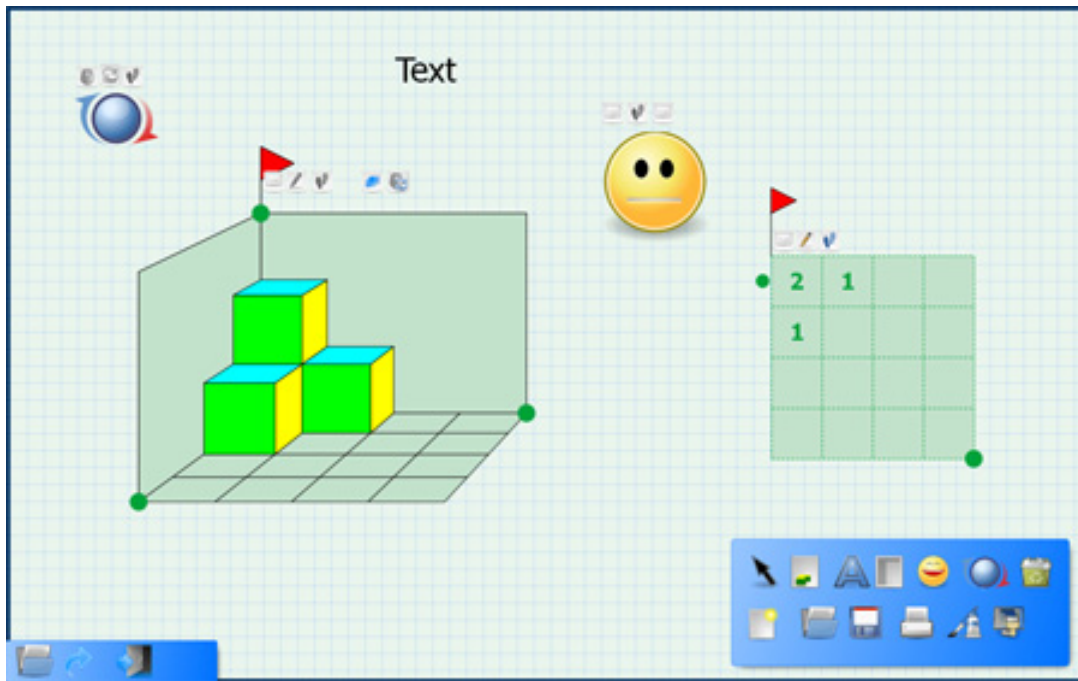


Figure 8. Screenshot of second version

According to Fisher (2004), many of our knowledge are achieved by co-operation. Students learn better when they can observe work of other people in their surroundings. The best conditions for learning are created when students are exposed to requirements, which extend their knowledge.

Iteration 5 - During the previous iteration, students always run our program on full screen. We observed that they do not like this program's property. Therefore, we decided to try to change program's environment, displaying of icons and to not run it on full screen, but in the window. There are several students' perceptions on this environment:

- Now, I can minimize it and browse the web pages.
- There are much more icons, but it is good.
- Previous environment was nicer.

Program does not contain more icons as previous environment and it missed some functionality, but students considered this environment more complicated. During this iteration, students found another program's bugs, which they did not find in previous iterations.

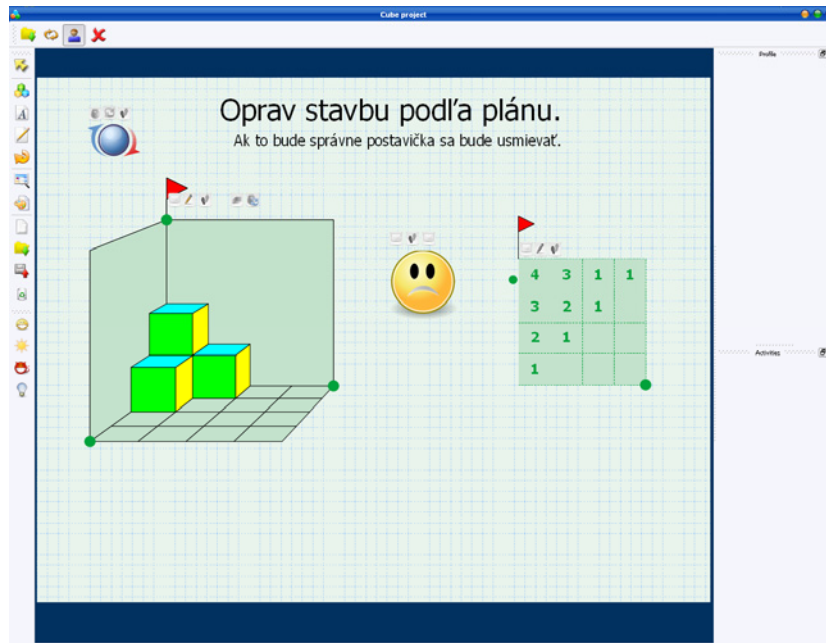


Figure 9. Screenshot of the third version

Process of software's development

Based on our experience in commercial projects and educational software developed so far, we experimentally tried the educational software development cycle based on Extreme Methodology and Action Research. The presence of the whole team at the lessons had a complement effect. Enthusiasm of children working with the program also supported internal motivation of all team members.

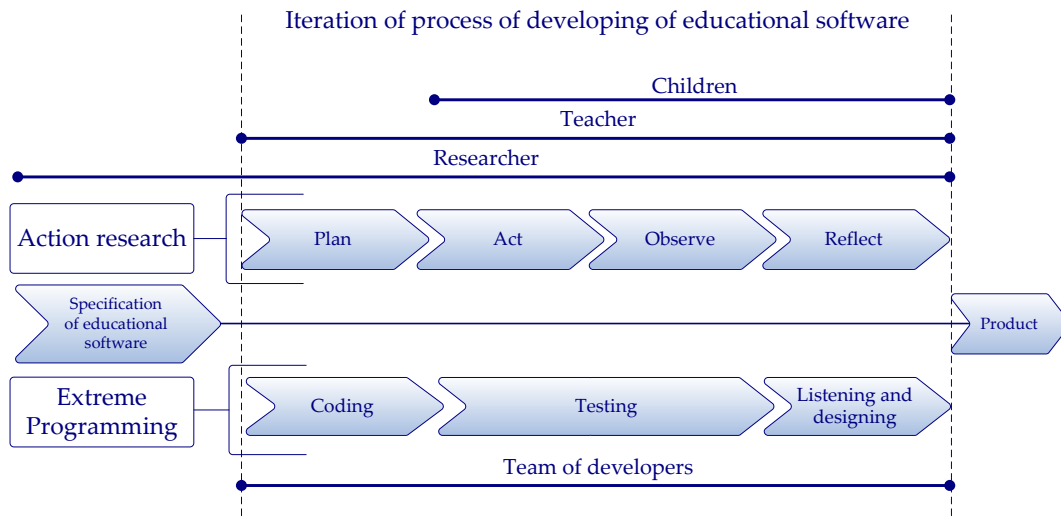


Figure 10. Process of developing of educational software inside project.

There were three people in the team: graphic designer, teacher and programmer. Only the teacher was present at the first testing iterations at lessons. Even though he was able to provide information about methodology and software errors, these lessons turned out to be inefficient from the development process point of view and few misunderstandings arose. When the whole team participated at the testing, the development efficiency increased. After each testing, we

discussed the lesson as a team and the results influenced the following iterations of either software development or pedagogic research.

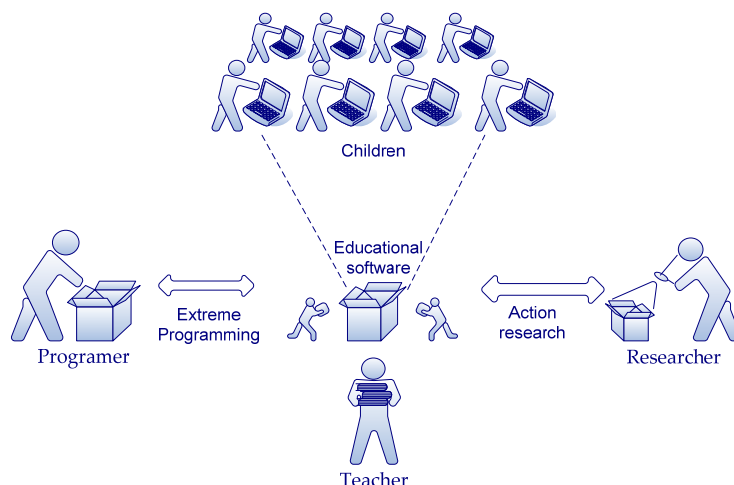


Figure 11. Abstract scheme of developing of educational software.

Conclusion

„What a child can do today in cooperation, tomorrow he will be able to do on his own “

Vygotskij, 1962

According to Unesco (2005), there are researches who prove that social co-operation can be helpful for students, when interactions are supported, which contribute to learning. Furthermore, interactions with other people are interested for students and they help to involve students into school work. Students intensively work because of results of their work. They know that it will be presented to other students by network.

Our experiments also proved that it is important for students to exchange information and findings among themselves. Therefore, it would also be supported by program, which will use Internet in it. Students will not solve tasks created by teacher but also tasks from other students. They will be able to create their own tasks to teach one another. In our subsequent research we will create network version and do additional iterations with students.

References

- Kadlec, V (2004) *Agilní programování, Metodiky efektivního vývoje*, CP Books, Praha, ISBN 80-251-02364-8
- Hendl, J (2005) *Kvalitativní výzkum: základní metody a aplikace*, Portal, Praha, ISBN 80-7367-040-2
- Fisher, R (2004) *Učíme děti myslet a učit se: praktický průvodce strategie vyučování*, Portal, Praha, ISBN 80-7178-966-6
- Unesco (2005) *Efektivní učení ve škole*, Portal, Praha, ISBN 80-7178-556-3
- Druin, A (1999) *The Design of Children's Technology*. Morgan Kaufmann Publishers, San Francisco CA, ISBN 1-55860-507-X
- Kalas, I. and Winczer, M. (2006) *Building interfaces for on-line collaborative learning*. Journal Education and Information Technologies, Springer, ISSN 1360-2357, 371-384.
- UNESCO (2002) *Information and Communication Technologies in Teacher Education (A Planning Guide)*, Division for Higher Education

Program your NXT robot with Imagine

Pavel Petrovič, ppetrovic@acm.org

Department of Computer and Information Science, NTNU Trondheim

Abstract

Our aim is to develop a versatile toolset for constructive learning. We achieve it through the marriage of powerful learning tools: a universal learning environment Imagine Logo (Kalaš and Hrušecká, 2004) and the recent and most popular educational robotics kit LEGO® Mindstorms® NXT.

LEGO Mindstorms NXT is the second generation robotics toolkit. Performance and robustness have been improved, infrared communication medium was replaced by BlueTooth® radio, new sensor types and motor actuators are more powerful with built-in encoders. Educational software RoboLab has been completely rewritten based on the collected experiences. And perhaps the most important is that the platform is open and well documented down to the low-level hardware layer. Schools and educational centres already own the sets. However, the chosen approach used in the RoboLab educational software does not necessarily suit all needs. In particular:

- Despite the communication capabilities of the robots, the software does not provide any means for writing applications for the PC, which could interact with the robots;
- The software requires learning yet another language, which may be a too high obstacle for educators;
- The coding style is based on drawing flow-chart diagrams, which easily become complex and difficult to understand, modify, debug, and analyze; the language lacks structure, the possibility to view programs as plain text; and it implies serious limitations;

Other environments for NXT programming are already available, but they usually suffer from some of these shortcomings. Our approach is different. Let us use the existing power of Imagine Logo, and give its users the possibility to control and even program their NXT robots. We demonstrate how this on three different levels, all utilizing the BlueTooth communication:

- Direct interactive control of robot sensors and actuators from Imagine environment;
- Loadable Imagine project with procedures for interfacing robots from Imagine projects;
- Possibility to download and run a piece of Logo program directly on NXT robots, while this program may communicate with an Imagine project running on the PC or other NXT robots.

The programs running on NXT robots may feature multithreading and get access to all features of the NXT brick. User works only with the procedures that were implemented in Imagine Logo. The logo interpreter – a program written with Next Byte Codes is automatically downloaded to NXT from Imagine environment. The solution does not require the firmware replacement. Our implementation is open-source and available for further development and improvement.

In this paper, we describe all three interaction modes on demonstrative examples. In the second part, we develop two educational experiments for physics and mathematics. These include the Imagine project and robot setup.

In the mathematics experiment, the children are presented with a black-box with gears, where their task is to write a program in Imagine Logo, which will utilize the sensors in order to measure the angular velocity of the input and output of the gear box, compute the conversion ratio, and thus determine the contents of the box given the known set of gear wheels. The physics experiment demonstrates the use of pulleys and a division of force. Students measure and observe the difference in force applied by the motor in various pulley setups.

Keywords

Imagine, LEGO Mindstorms NXT, Educational Robotics



Figure 1. Robogjeng – after-school robotics club at secondary school Katedralskole in Trondheim at public presentations: MiniSumo playing robots (left), and Labyrinth-navigating and Line-following robots, right.

Robotics in Education

Robot is a machine which behaves, works, and often looks like a human being. Robots – as other machines – are usually used for the benefit of the human: to liberate us from heavy work or provide functionality that would otherwise be inaccessible. Thanks to the advance of the technology, robots are part of everyday life today. From this perspective, it is natural to expect their appearance in the educational process. This for the sake of: assisting the teachers and students in various tasks (for instance for disabled) on one hand, and becoming part of the learning experience on the other hand. Robots in education may:

- ✓ demonstrate phenomena in novel and more ample ways,
- ✓ provide creative platforms for hands-on exploration for individual or group student work,
- ✓ increase entertainment experience during the learning process,
- ✓ increase motivation for learning,
- ✓ spawn interest in technology among students.

Using robots in education has severe drawbacks and challenges:

- ✓ high cost of robots,
- ✓ extra time, space, work and competence required from teachers and schools,
- ✓ shortage of curriculum materials and guidelines.

These can be overcome generally only through added-value enthusiasm of teachers and students who see the potential and value in such work. Once enough experience and potential is generated, institutions may eventually integrate robotic platforms into more or less standardized curriculum. In this sense, LEGO is doing a pioneering work, which naturally is not always a profitable business.

Important role play after-school robotics clubs and centres, which bring together young people with enthusiasts and generate deal of material that can possibly be digested into curriculum by educators. Figure 1 shows the students from secondary school in Trondheim who meet regularly to program robots for fun and robot competitions (Balogh, 2005) – such as RoboCup Junior (Sklar et.al. 2000), MiniSumo, or Micromouse.

Teaching Robotics Materials

The work towards the use of robots in schools is slowly starting to materialize into teaching materials. Most of the pilot programs performed up to day focus on teaching basics of control and programming – learning about sensors, feedback, and mechanics. Examples of such are

the project of London Grid for Learning or works of (Rosenblatt M. and Choset, 2000, Johansson, 2001). LEGO provides a set of inspiration booklets on mechanics, buildings, energy, and robots, and a set of 16 activities with worksheets and supporting CD: Science & Technology Activity Pack. Extensive work of LDAPS team (LEGO Design and Programming System), which was at the start of RoboLab system produced multiple creative project ideas (LDAPS url). These are described also in the book (Erwin, 2001) and in the recent book of Barbara Bratzel, a teacher at Shady Hill School, a preK through 8 independent school in Cambridge, Massachusetts, who has developed a project-based course that teaches classical mechanics through engineering (Bratzel, 2005). Comprehensive set of materials is provided by CMU Robotics Academy (CMU Robotics Academy url). Yet, there is a large potential for more curriculum material that would be easy to use for teachers without extensive previous experience and technical competence.

LEGO Robotics

Today, LEGO has a long tradition of producing educational toys and robotics programmable sets. The LEGO Educational department founded in 1980 was renamed to LEGO Dacta in 1989 and produced programmable sets that could be controlled from computer running a dialect of Logo. Already since 1984, Dr. Papert had been working with LEGO's development staff on linking the LOGO computer programming language to LEGO products. LEGO Logo that was developed in that cooperation was a successful product that allowed controlling non-autonomous LEGO robotics sets. However, today, this product is almost completely forgotten somewhere in the history of the early 90s. With the autonomous robotics sets, new concepts arose and ought to be treated anew.

A precursor of the autonomous programmable brick was CodePilot released in 1997 as part of the 8479 Technic set. It could be programmed by swiping over barcodes. However, the highlight, LEGO Mindstorms with wonderful programmable brick RCX arrived in 1998, and it is still enjoying attention (RoboCup Jr. Slovakia url). RCX can be programmed using graphical iconic parallel language of Robotics Invention System, using RoboLab – an educational iconic language with greater functionality, using Lejos – a limited version of embedded Java, using Not Quite C (NQC) – a C-like interpreted language with limitations, or even using BrickOS – a standard C/C++ based on the GNU C/C++ with libraries for RCX control. Many other systems were developed, for instance PbForth – minimalistic stack-oriented procedural language. Several more or less compatible flavours of the LEGO Mindstorms were produced, targeting mainly entertainment market: CyberMaster, Scout, Micro-Scout, Spybotics. An important step forward was achieved by releasing the LEGO camera, which however, requires wired connection to the computer, and provides very limited functionality. The recently released LEGO Mindstorms NXT sets support compatibility with previous LEGO electric parts (motors, sensors), and align the LEGO robotics product with the current standard for embedded devices: radio Bluetooth communication, I2C bus, fast ARM7 microprocessor, direct USB connection, graphical display, servo motors, large flash memory, EIA-485 fast serial interface, compact Li-ion rechargeable battery. However, the main break-through is that LEGO released the source-code of the firmware, detailed schematics of all parts and documentation. This allows the non-LEGO developers to maximize the educational benefit of the new robotics system. In addition to new version of RoboLab – NXT-G, which is now the main programming iconic language for NXT, less than a year after its release, NXT can be programmed in limited Java – remotely using iCommand, and autonomously using Lejos, in a new version of NQC (now called Not eXactly C), in C-like language RobotC, and using the low-level byte-code language Next Byte Codes (NBC). Third-party developers play a very important role in making the LEGO robotics products useful by providing various sensors. The main players are HiTechnic, Mindsensors, and Techno-stuff.

The LEGO company has a strong focus on the user community, and provides ideas, hints and inspirations of professional quality at their websites (mindstorms.lego.com, legoeducation.info/nxt/). In addition, thousands of users and third-party developers exchange their experience at various forums, which include LEGO Users Group (lugnet.com), and blogs (thenxtstep.blogspot.com, nxtbot.com).

Logo for NXT

Logo programming language has a successful history of being used for teaching programming. Recently, LEGO introduced its second generation robotics educational sets on the market. Their combining is made possible and easy by using the radio Bluetooth link between a PC running Imagine Logo and a NXT brick. Both learning tools complement each other. In concert, they create a learning studio of endless series of possibilities for educational projects. That is our main motivation for the Logo for NXT project. It consists of the following parts:

1. Imagine project with simple controls for immediate remote control and status querying of NXT brick.
2. Imagine loadable text project `nxt.imt` containing set of elementary procedures that directly interact with NXT brick.
3. Logo interpreter running on the NXT brick that allows running simple logo programs on NXT brick autonomously.
4. Set of example programs for both the second and the third level.

Imagine NXT Controller

For immediate control of NXT robot, testing the functionality, calibrating sensors, or even simple navigation including speed regulation and direction steering, one can use the provided controller project, see figure 2. The user can setup types of sensors, configure and operate the motors. In the upper-right part of the screen, the nine controls allow steering a two-wheeled robot in all main directions. Since NXT motors have built-in rotation sensors, NXT brick firmware supports automatic synchronization of the motor speed to a specified value. Furthermore, the firmware can automatically control the ratio between the rotational speed of two selected motors. The controller project has controls to test this functionality. The NXT brick only needs to be on and have standard or compatible firmware installed. Logo interpreter program is not required.

Control NXT from users' projects

Users can load the `nxt.imt` project and call many different procedures that interact with NXT brick. These are of two main types:

1. Direct commands and commands according to LEGO NXT Communication protocol (LEGO, 2006), 34 different commands at the time of writing.
2. Higher-level commands that are based on direct commands and provide some higher-level functionality, or easier to understand interface, about 10 different commands at the time of writing.

Examples of the commands from the first set are **`nxt_getoutputstate`**, **`nxt_resetmotorposition`**, or **`nxt_openappenddata`**, which determine the state of the motors, reset the built-in rotation sensors, and open a file in NXT's flash memory for appending, respectively. All commands are described in the documentation of the Logo for NXT project (Logo for NXT url).

Higher-level commands provide the following interface:

`(motor m cmd [arg])` – operates on motor output `m`, supported commands are: `on`, `off`, `brake`, `regulate [speed]`, `sync [ratio]`, `onrot [nrot]`, `onms [ms]`, `power [pwr]`, `rst`;

`(readmotor m cmd)` – reads the rotation sensor of specified motor in one of three ways (rotations, `tacho`, `blocktacho`);

`(sensor N)` – reads the value of specified sensor;

`(setsensor N type_or_cmd)` – configures the specified sensor as: `raw`, `switch`, `tmpC`, `tmpF`, `refl`, `angle`, `lightA`, `lightI`, `db`, `dba`, `lowspd`, `spd9V`. Also allows resetting sensor: `rst`.

`(nxt_connect port)` – establishes connection to NXT brick on specified virtual serial BT port;

(nxt_disconnect) – closes connection to NXT brick;

(download filename) and **(upload filename)** – transmits a file to or from NXT;

User projects can use commands from both groups interchangeably. Attention needs to be paid to the fact that Bluetooth radio communication despite its relatively high bandwidth in one-directional communication has an inherent delay when switching the direction of communication. Therefore the commands that require responses from NXT will usually take about 60ms to complete.

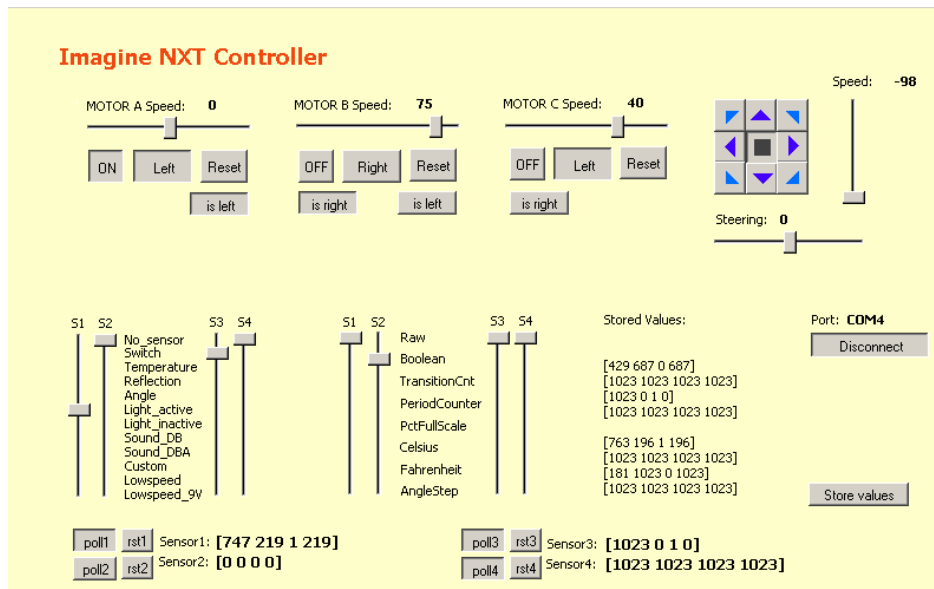


Figure 2. Direct control of NXT brick using Imagine project.

Logo interpreter for NXT

Finally, as a simple and straightforward extension to the interface described above, the command **(remote expression)** allows evaluating Logo expressions directly on the NXT brick. This command assumes the logo interpreter program is already running on the brick, but if it is not, the running program can be controlled from Imagine with the help of **nxt_startprogram**, **nxt_stopprogram**, and **nxt_getcurrentprogramname** commands. The expression is transmitted to NXT over Bluetooth radio, evaluated on the NXT, and the return value is sent back to Imagine Logo running on the PC. The expression can contain all supported commands. For example, the command **(remote [load "myprogram.lgo])** will load the specified program from the file stored in the NXT flash. NXT Logo programs can define procedures and operations, and they can themselves, in an analogical way, request expressions to be evaluated on the PC by Imagine Logo (provided that the expression-polling is turned on) using the same remote command.

At the time of writing, only very limited functionality of the interpreter is available, but we expect the following features to be available shortly:

NXT Logo supports the following primitive procedures and operations:

1. Arithmetic: *add*, *sub*, *mul*, *div*, *mod*, *neg*, *abs*
2. Logic: *and*, *or*, *xor*, *not*
3. Bitwise: *bitand*, *bitor*, *bitxor*, *bitnot*
4. Lists and words: *first*, *bf*, *last*, *bl*, *item*, *se*, *list*, *word*
5. Maps & co.: *map*, *apply*, *foreach*, *run*, *remote*

6. Loops and conditions: *repeat, while, foreach, if, ifelse*
7. Predicates: *empty? number? list? word? equal? gt? lt? ge? le? member?*
8. Variables and procedures: *make, let, to, op, stop, erase, launch*
9. NXT file handling: *load, printto, fopen, fwrite, fread, fclose, fremove*
10. Sounds: *tone, sound*
11. Motors, sensors: *motor, readmotor, sensor, setsensor, battery*
12. Other: *wait, random, print, draw, ascii, char, boot, gc, fm*

The expressions must be in the prefix notation. Currently all procedures have a fixed number of arguments, and therefore the parenthesis are not needed (and are not recognized by the parser). Variables are used in the same way as in Logo: **"name** denotes variable name, and **:name** refers to its value. There are global variables (**make**) as well as local variables (**let**, and arguments). Lexical scoping applies as usual in Logo. There is currently no support for objects, or images, or other primitive types. However, the **draw** command allows drawing images stored in the files in the NXT flash, and the **sound** command allows playing NXT sound files. The NXT software architecture has a limited support for multithreading, and the logo contains the **launch** command for starting a separate thread. See the documentation (Logo for NXT, url) for more details on the syntax and purpose of all supported commands.

Example programs

A set of documented example programs demonstrates the basic use of the Logo primitive commands. In this way, users who are not familiar with Logo programming language can learn to use Logo for NXT directly. In fact, we do not require the user to own Imagine Logo. Logo programs that are uploaded to NXT can be started directly from the interpreter. The communication with the PC will be disabled, and the remote command will have no effect (return "false), but the logo programs will run.

In addition, example programs for line-following, and mini-sumo playing robots, as well as some other are provided.

Implementation issues

Logo for NXT interpreter is written in NBC, a very low-level programming language that is compiled into byte-code, which is then interpreted by the standard NXT firmware. NBC programs have at their disposition 32 Kbytes RAM for "dynamic" memory. Logo interpreter allocates the whole available memory as one large array, and features its own C-style memory management (malloc, free) with support for garbage collection on demand. That means that all data that can be de-allocated immediately are de-allocated immediately. However, lists, and words are being freed only after verifying that there are no pending references. We aimed at very compact and memory-efficient representations utilizing almost every data bit.

System works with 16-bit unsigned words, where typically the upper 14 bits form a value or a pointer (given max. 32 KB memory, 14 bits are sufficient to address 16-bit words). The remaining two bits are used for the type information: direct integer value (00), pointer to word (01), car-pointer to list (10), cdr-pointer to list (11). List elements are normally stored in a continuous block, and the cdr-pointer is used only if the list grew out of its original memory block.

Lists of all words, all global variable names, and all procedure names are maintained. These lists are not simple linked-lists, rather sorted linked-lists of tables containing 8 records each in order to make the search faster (**symbol tables**). When any of the tables becomes full, it is automatically split in two. Logo words are stored in a single memory block each. There is a separate support for the columned and quoted words ("hello, :var) so that there is no duplicity (i.e. columned and quoted words are only manifesting its type and point to the memory block with the unquoted word). Unquoted system reserved words are especially marked so that they

are never de-allocated. All unquoted words have one 16-bit slot reserved for the variable pointer to the current context, which contains the current value, and one 16-bit slot reserved for the procedure pointer to procedure symbol table. The former is updated each time a variable is created/erased/shadowed. Words are never stored in duplicates in the memory, and as a consequence, comparison of pointers is sufficient. List items are 16-bit values as described above. The pointers to list nodes can point to middle of a continuous sequence of list elements. When lists are allocated, there is automatically a small buffer of free elements provided before the start of the list to allow for growing forward without the need of re-allocation. This is to avoid memory blocks containing only a single list element, which would mean too large overhead. Lists are stored in reversed direction so that garbage collection process can safely traverse through a pointer to a random element in the middle of the list to the start of the block and mark it as used. Each memory block requires one word (16-bytes) of overhead containing a 14-bit value – the block size, one bit indicating if the block is free, and one bit used by the garbage collection. The procedures are represented as lists of commands, preceded by a value containing the number of arguments, and a pointer to the list of argument names. The names of primitive procedures are loaded to memory from a separate file (logo.def) during the start-up of the interpreter.

The **core engine** consists of parser, evaluator, and printer. The **parser** is started when an expression is sent using the remote command, or when a logo program file is loaded from the NXT flash. Its role is to simply convert the string expression into internal expression/value representation. The opposite transformation is the responsibility of the **printer**. The **evaluator** has two flavours: one that evaluates list-expressions – such as procedure or loop-command bodies. These contain commands that do not return values, and the whole expression can optionally return a value using the **op** (output) command. The second flavour evaluates single immediate values, such as procedure arguments, and always returns a value. The evaluator supports recursion, and variable shading. It works with a current context, which is a structure containing a reference to local variables symbol table, parent context, and expression being evaluated as well as the position in the evaluated expression. Since the parser translates all variable references to pointers, the values can be immediately retrieved through the symbol-table pointers contained in the block of the word. When returning from a call, current context is released, and the previous symbol-table pointers are retrieved from the local symbol-table that stores the previous pointer.

The communication with PC is maintained by a communication thread perpetually running in the background. It calls the parser and evaluator when an expression arrives. If communication is not established, user can enjoy a limited interaction with the logo interpreter using the buttons and LCD display.

Garbage collection works only on demand. The logo program must call the **gc** command in order to free unused memory. At that point of time, all global variables, all local variables in all parent contexts, all referenced lists, words, and procedure bodies are marked as used. Finally, the whole memory is scanned, and all blocks which are not marked are claimed to be free. The fast ARM7 processor running on 48MHz frequency is able to scan the whole 30kB memory thousand times a second, therefore even if the logo interpreter itself is interpreted, the performance can be acceptable, provided that the garbage collection is not started in critical periods. We believe this is a feasible approach for programming embedded systems in general, where sudden unplanned interruption by garbage collector could cause undesirable results. On the other hand, the programs for embedded devices typically contain safe periods when garbage collecting break is possible.

Two Projects

While LEGO and robotics provide multiple advanced challenges, such as robotics competitions, which are strongly motivating students with over-average performance, LEGO in the schools can be the motivational force for the students who are struggling with the usual curriculum material due to their difficulties with maintaining attention and finding interest when links between

theoretical knowledge and practical applications are missing. This section discusses two simple LEGO Logo-style projects, to be used directly at the standard math or physics lessons.

Gearboxes – learning about fractions

The first project can be used at mathematics courses in junior high-school for teaching computing with fractions. Students are presented a closed box with the same kind of gear wheel in the middle of both sides of the box. The wheels are connected using gears in some way, i.e. turning the wheel on one side by 1 rotation turns the wheel on the other side by a/b rotations (see top of the figure 3, the inside of the gearbox is shown on the right). Teacher can prepare several gearboxes with different ratios. It is possible to simply attach a motor with a rotation sensor on one of the wheel, by connecting it on top of the gearbox, and another rotation sensor to the second wheel by connecting it to the bottom of the gearbox (both modules are shown at the bottom of the figure). The task for the student is to first measure the ratio between the numbers of rotations on both sides, then describe this ratio by a fraction. Finally, the student needs to make estimation (a drawing) of the contents of the gearbox, knowing that only standard gearwheels with 40, 24, 16, and 8 teeth are used inside of the gearbox. This requires mastering fraction algebra. Students can use Logo for NXT to write program that measures the number of rotations, see figure 4.

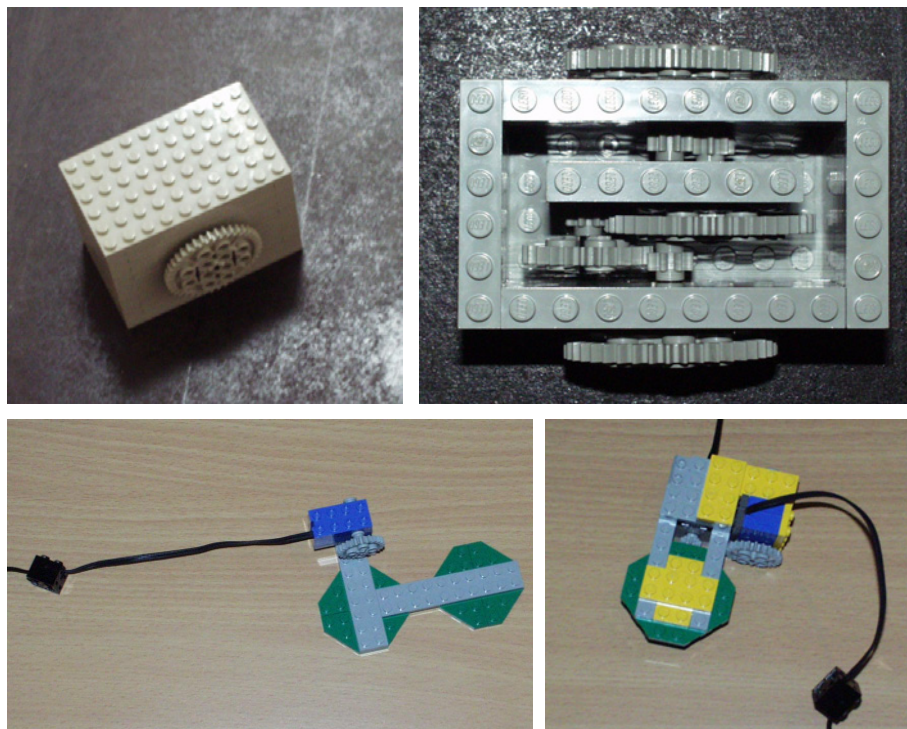


Figure 3. Gearbox project: motivated learning about fraction algebra.

```
to measureRatio
  setsensor 1 "angle
  setsensor 2 "angle
  setsensor 1 "rst
  setsensor 2 "rst
  motor "A "power 50
  ; rotate the first wheel 10-times
  while [gt? 160 sensor 1]
    ; sensor2 measures the opposite side, output the ratio
    op list sensor 1 sensor 2
  end
```

Figure 4. Gearbox project: Logo for NXT program to measure the ratio between numbers of rotations.

Analysis: Including a practical experiment in a series of mathematics lessons not only improves the manual skills of the pupils, but mainly creates important connections between the theory material that is part of the curriculum and the real life. Such practical demonstrations of the learned theory are **essential**, if the school aims at preparing the pupils for the real life, particularly in the lower-grades and in the schools for the widest general public.

Pulleys and division of force

The machine shown on figure 5 is a demonstration of practical application of pulleys. It is inspired by a model in the science museum in Bremen. A mobile platform can slide in the track from right to left and opposite. It is connected using a LEGO plate brick to a string, which can be rolled on a coil using a motor controlled by NXT brick. A light sensor can signal when the platform reaches its left-most position. The platform can either be connected directly, or using the two pulleys. There is a load (on the figure, it is a LEGO 9V battery box) placed on top of the platform. Various loads can be used, especially such that allow gradual increase of weight – for example a set of weights from the physics laboratory). The task for the student is to measure the force the motor applies on the platform in both machine configurations. For reasonable results, the motor is powered with only 10-15% of power. The force can be measured using Newton-meter from the physics laboratory, or by loading the platform with different weights (and taking into account friction coefficient), or by attaching a string with weights hanging down from the other side of the platform over another pulley. Figure 6 shows a simple program in Logo for NXT that can be used to run an experiment.

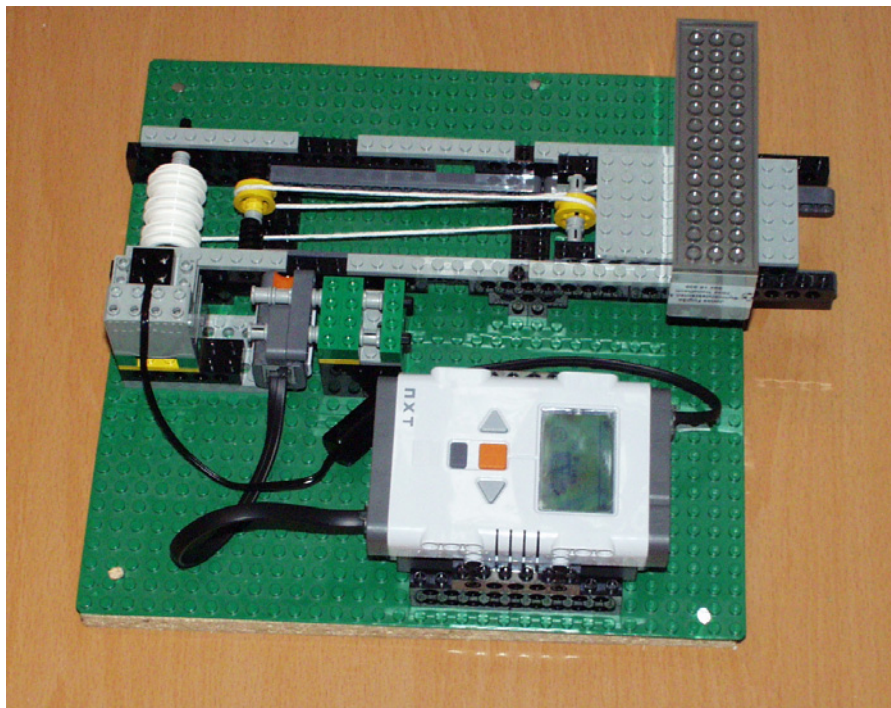


Figure 5. Pulleys project: learning about pulleys, force, friction, work.

```
to movePlatform
  setsensor 1 "lightA
  let "threshold 30
  motor "C "power 15
  motor "C "on
  while [lt? sensor 1 :threshold] []
  motor "C "off
end
```

Figure 6. Pulleys project: Logo for NXT program to run the platform and stop it at the end of the ramp.

Conclusions and Future Work

We presented an alternative way of programming NXT robots, which we believe is the way the LEGO programming system could have been designed in the first place. In fact, the original LEGO Dacta systems that were connected using the wired interface were using a powerful dialect of parallel Logo. The transition from the wired remote control to autonomous RCX required abandoning that path due to the limited memory, CPU, and communication capabilities of the RCX. NXT resolves these bottlenecks and it is time to come back with powerful Logo language as the programming platform for the educational use of LEGO robotics systems. Our work does exactly that. It still has major limitations due to the fact that it is implemented using interpreted NBC, that is Logo programs are currently interpreted by an interpreter that is interpreted by the firmware. We chose this solution for three reasons: 1) to determine the limitations of the NBC and to see if it can provide a reasonable performance and 2) to liberate us from the arduous challenge of understanding how to write our own firmware, and 3) to provide as compatible solution as possible, one that does not require the user to change the firmware on the NXT brick. Despite the limitations (these are: 1) 30 kB memory limitation for all data, code, and internal stack compared to normally available 64 KB RAM, and 2) limited performance compared to interpreter that would execute directly on the CPU), our system can already be used successfully to program NXT robots in Logo. In addition, it smoothly plugs into Imagine Logo educational suite. This paper presents the system as we wish it will perform. At the time of writing, portions of the functionality are still under intense development. The Logo for NXT project is open-source, and available for the user customization. Source-code is extensively commented, and documentation is provided. We demonstrate two simple projects, where Logo for NXT is used in practical student tasks within standard curriculum of junior high school.

Acknowledgments

Tomas Gunnarsson from Nardo Robot Club, and Gustav Henrich Bernhardt from Katedralskole in Trondheim provided extensive support and photographs from public presentations.

References

- Balogh, R. (2005) *I am a Robot - Competitor, A Survey of Robotic Competitions*, International Journal of Advanced Robotic Systems, vol.2, number 2, p. 144—160.
- Bratzel B (2005) *Physics by design*, College House Enterprises, LLC.
- Erwin B. (2001) *Creative Projects with LEGO MINDSTORMS*, Addison-Wesley.
- Johansson H. (2001) Robotic courses based on LEGO, Master Thesis, Department of Computer Science, Lund University.
- Kalaš I. and Hrušecká A. (2004) *The Great Big Imagine Logo Project book*, Logotron.
- LEGO (2006) *LEGO Mindstorms NXT BlueTooth Development Kit*, LEGO NXT'reme document, version 1.00.
- Rosenblatt M. and Choset H. (2000) Designing and Implementing Hands-On Robotics Labs, IEEE Intelligent Systems, vol.15, nr. 6, p. 32-39.
- Sklar, E.I. and Johnson J.H. and Lund H.H. (2000) *Children Learning From Team Robotics: RoboCup Junior 2000*, Educational Research Report, Department of Design and Innovation, Faculty of Technology, The Open University, Milton Keynes, UK.
- CMU Robotics Academy url, <http://www-education.rec.ri.cmu.edu/>
- LDAPS url, <http://www.ceeo.tufts.edu/ldaps/htdocs/>
- Logo for NXT url, http://virtuallab.kar.elf.stuba.sk/robowiki/index.php/Logo_for_NXT
- RoboCup Jr. Slovakia url, <http://www.skse.sk/>

Programming Robots in Logo

Pavel Petrovič, ppetrovic@acm.org

Dept of Computer and Information Science, NTNU, Trondheim

Ronald Weiss, ronald@microstep-mis.com

Microstep-MIS, Bratislava

Abstract

The field of robotics is important for several reasons: Robots allow humans to perform work that is not suitable for us (dangerous, require superhuman power, endurance, precision, or reliability), robots can produce things at lower cost; they can enable mankind to reach beyond our known horizons, and they can improve quality of our life in various ways.

The general public must be prepared for the presence of robots and must pay attention to and understand the principles and issues of man-machine interaction, robot control and behaviour. Schools and teachers can and should play a role in this process, or at the very least they should be informed.

The workshop will be a hands-on tutorial into basic robotic control with the help of Logo. It will consist of several hands-on activities where the robots are controlled using Logo language with the goal of using the robots in the school curriculum. The activities will include both the traditional LEGO Mindstorms with RCX and the newer LEGO Mindstorms NXT robots communicating with Imagine Logo, and controlled directly by Logo scripts (in case of NXT). In addition, control of Robotnačka, the turtle robots, from Imagine, including the publicly available Remotely-Accessible Robotics laboratory at <http://virtuallab.kar.elf.stuba.sk/> Participants of the tutorial will be able to build their own simple robots and write their own programs in Logo to control them.

References

LEGO Mindstorms, <http://mindstorms.lego.com/>

Žurina, D. and Petrovič, P. and Balogh, R. (2006) *Robotnačka – The Drawing Robot*. In Proceedings to Robtep 2006.

Petrovič, P and Lúčny A. and Balogh R. and Žurina D. (2006) *Remotely-Accessible Robotics Laboratory*. In Proceedings to Robtep 2006.

Design and evaluation of Maths related programs for special education

Brigitta Réthey-Prikkel, *rpb@ludens.elte.hu*

Dept. of Media & Educational Technology, ELTE University, Hungary

Márta Turcsányi-Szabó, *turcsanyine@ludens.elte.hu*

Dept. of Media & Educational Technology, ELTE University, Hungary

Abstract

Dyscalculia means the partial lack or error in the counting ability, which is not to be mistaken by acalculia, where these are totally absent. Dyscalculia can occur within any level of intelligence and is a type of disturbance that is typically noticed in school, but is less researched than dyslexia. There is a whole list of syndromes that could indicate dyscalculia and it needs a complex dyscalculia testing by psychologist, neurologist, and logopedia specialist in order to determine the exact type and syndrome of the disturbance, after which any possible treatment could be started. Even though it would be possible to set up diagnosis at the early age of 5, there is no standard dyscalculia test in Hungary, which could determine – with great probability – the fact of being endangered.

TeaM lab provides courses within ELTE teacher training as non-compulsory electives, with relation to developing e-learning materials and running projects in public education, where students develop programs for different curriculum areas, take them out for use in schools and prepare a report about their effects in classes, suggesting some areas of possible modifications for improvement. There is an active collaboration between ELTE Bárczi Gusztáv Practice Elementary School and Methodological Centre for Special Education and TeaM lab since 2003 to develop ICT tools supporting their pedagogical program and hold experiments to evaluate their effective use.

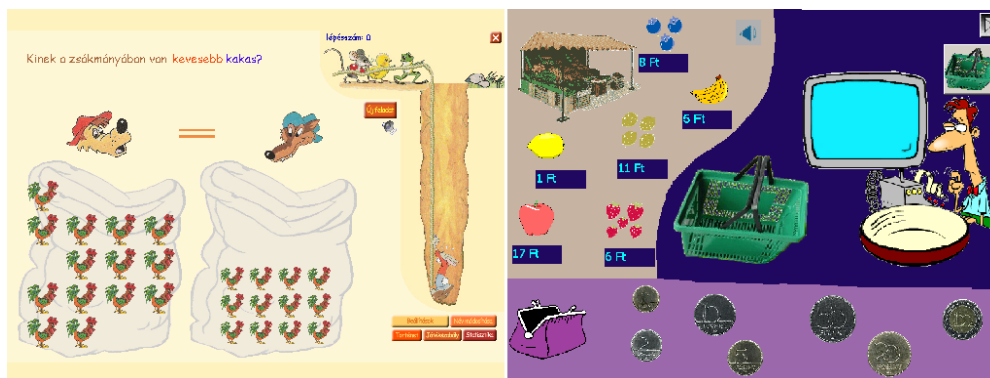


Figure 1. Screen shots of two programs: comparison of quantities and purchase with money.

The paper discusses requirements forced on the developed programs that emerge from practical experiences, the development process is analysed as well as the formal and pedagogical evaluation. Three tests have been used to measure levels of achievement, the Hiskey-Nebraska Test of Learning Aptitude (H-NLAT) and change in skills, the mathematical ability test created at Bárczi Gusztáv Faculty of Special Education, and a general test on fractions. The results are reported and shall be presented at the session.

Keywords

ICT, game design, children with special needs, developing skills

Dyscalculia diagnosis and treatment

Dyscalculia means the partial lack or error in the counting ability, which is not to be mistaken by acalculia, where these are totally absent (Hrivnák, 2003). Dyscalculia can occur within any level of intelligence and is a type of disturbance that is typically noticed in school, but is less researched than dyslexia (Mesterházi, 1996). Due to cultural reasons the society is much more forgiving if a child has problems in maths than if (s)he cannot read properly, or writes with bad spelling (Kulcsár, 2005).

There is a whole list of syndromes that could indicate dyscalculia, just to extract a few problematic areas: perception, recognising numbers, understanding the concept of numbers, leaving out some numbers when calculating, going over 10, understanding and performing functions with fractions, considering signs when subtracting, poor mathematical logic, use of symbols, serialising numbers, grouping, understanding ratios, time, volume, dimension, ... etc. (Mesterházi, 1996; Hrivnák, 2003). It needs a complex dyscalculia test administered by psychologist, neurologist, and logopedia specialist in order to determine the exact type and syndrome of the disturbance, after which any possible treatment could be started. There are several motives, which could indicate that a child might be endangered and thus needs to be thoroughly diagnosed, example: when (s)he is late or retarded in speech, lisps, or is dyslexia endangered; in other cases there might be problems with visual or acoustic perception; or even problems with memory, large or fine motoric functions. Even though it would be possible to set up diagnosis at the early age of 5, there is no standard dyscalculia test in Hungary, which could determine with great probability the fact of being endangered and could be administered on a large population for filtering purposes.

Therapy includes developing skills using personalised programs for developing mathematical abilities and increased use of demonstration, e.g.: Montessori-tools, coloured rods, using fingers (Mesterházi, 1996) and should increase levels of perception, concentration, memory, thinking, language skills, establishing basic mathematical concepts on numbers and their use in measurements, functions and symbols, ... etc. (Hrivnák, 2003).

The binding situation

The curriculum for special needs schools

Because of the special needs of the pupils, the school's curriculum is different from that of the usual. Here pupils need much more learning time than most children and they are definitely granted extra time. The Hungarian National Curriculum (NAT, 2003) emphasises within the general issues of obligatory schooling for handicapped children's training and teaching:

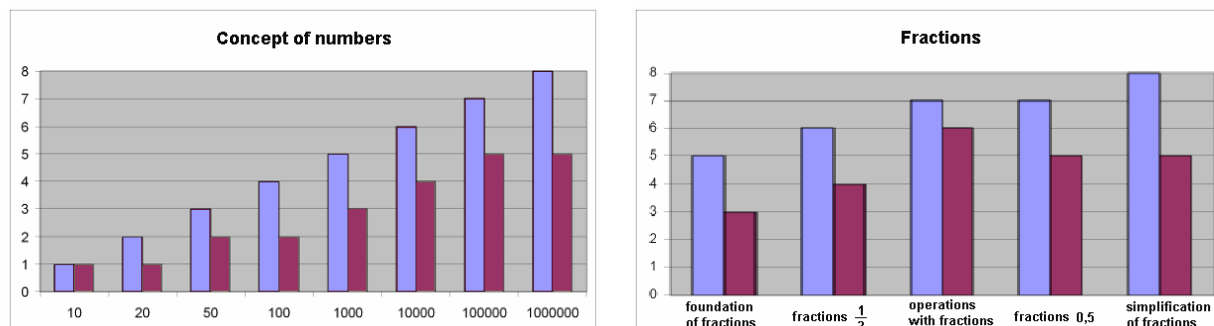
„In case of pupils with special educational needs, the following primary principles must be applied, while being adjusted to individual needs and limitations:

- Extended time periods must be given to accomplishing tasks, as needed;
- If necessary, special content and requirements must be developed and used in accordance with the nature of their disability;
- Schools must use positive discrimination and differentiation with these pupils, providing them with individual help and assess them mainly on the basis of development in light of their condition.

The special tasks required in case of certain disabilities are governed by the guidelines for the curriculum of pupils with special needs as well as examination regulations.” (NAT, 2003)

The Hungarian curriculum for special needs children suggests that children with learning difficulties needs two years in order to master the materials defined in first grade, and for the first two grades material three years are suggested, while the schools are free to decide about the terms of progress. Teachers in schools pay attention to the development of each pupil, but the local curriculum does not suggest any further requirements.

For the above reasons, we can only calculate with as much as the maximum of expectations, but always have to have in mind, that class are usually far behind the expected curriculum, which can also be the case with normal children. Nevertheless, classes often lag behind the suggested timing and would be able to fulfil tasks suggested within 6 years of schooling only within the time frame of 8 years. It cannot be predicted how far children can get in development of their skills, abilities and knowledge.



Blue is the schools with learning difficulties, and purple is normal school.

Figure 2. Difference in levels of achievement (in terms of grade) in normal and special schools within different subject areas, based on data available (NAT 2003).

ELTE teacher training specialties

TeaM lab provides courses within ELTE teacher training as non-compulsory electives, with relation to developing e-learning materials and running projects in public education, where students develop programs for different curriculum areas, take them out to use in schools and prepare a report about their effects in classes, suggesting some areas of possible improvements for better use (Turcsányi-Szabó, 2006c). Among courses are the following:

- **Designing educational programs:** Students develop complex educational programs (using Flash or Imagine) to be used in different disciplines practicing experiential and constructivist pedagogy on a specific focused area.
- **Evaluation of educational software:** Formative evaluation and pedagogic evaluation of software (usually developed by other students in previous years), by analyzing the National Curriculum and teaching strategies in order to define a hypothesis for pinpointing the scope of development, designing activity to go through the process, and composing pre- and post test to prove the presumed hypothesis.

ELTE special education school

ELTE Bárczi Gusztáv Practice Elementary School and Methodological Centre for Special Education engages children with learning difficulties, where they can learn to read and write with dyslexia prevention methods, which assures the individual improvement of each child and makes an effort to exclude the usual failures. They also try to assure continuous development of skills and abilities within all classes, for instance mathematic classes. There are special pedagogical services for the first graders, like: generative activities (for attention, memory, orientation in space and time, motion development and motion skills), speech therapies, habilitation and rehabilitation sessions and therapies. For these activities they have special rooms, gym room and a swimming pool. Children can choose different afternoon activities like drama, ICT, journalism, household management, needlework, swimming, table-tennis, singing, playing on the piano and the flute.

Our collaboration

There is an active collaboration between Bárczi School and TeaM lab since 2003 to develop ICT tools supporting their pedagogical program and hold experiments to evaluate their effective use.

The school has so far tried out different software within their classes and just kept a handful of them, since the rest did not suit their special needs. After our established relationship, they provided us with a list of ideas they could use in their developmental work and suggestions for modification of software they have encountered so far. At the same time we prepared for them some skill building programs, which we thought would suit their needs.

Children have now different classes like reading, mathematics, journalism taking place in the computer room. This semester the fourth grade has one out of seven Hungarian language lessons in the computer room, and the seventh grade has two out of four mathematics lessons there too. The fifth grade has four mathematics lessons in a week and they spend two of them in the computer room. 36% of the topics mentioned in the curriculum are covered by different programs. Among the seven main themes in fifth grade a lot of programs used at the school originate from ELTE TeaM lab and were developed using Imagine.

In most cases, programs are used for practicing, showing visual examples, modelling problems, helping to find algorithms for different problems, and conforming constructive learning practices. Of course not all topics can be and should be covered using computer programs, but it can be a motivating alternative. It is also important that making a mistake is not so embarrassing for a child while being involved with a computer than doing it in front of a teacher, and also graphics and animations can be of great help in visual modelling, and creating algorithms.

Case studies of developments

Our courses support program developments both with Macromedia Flash and Imagine, but it must be well investigated which authoring environment to use in each case. In case a closed program is envisioned with need for compact and well compressed media files, then most probably Flash technology would be chosen. But, in case we wish to leave the tool open and transparent for further configurations to be inserted (not necessary done by programming experts) or a possibility to change media elements, then most probably Imagine would be chosen as the environment for development. It is important to note, that while a program originally developed in Flash by one student is **never** changed by any other student, due to the fact that it is hard to figure out another Flash developer's logic. The situation however is not so difficult to handle in case of Imagine and students are more at ease to make changes according to defined criteria.

To demonstrate the need for configurable programs, we chose two examples. In the first case an existing game was changed into a learning tool, and in the second case a new learning game was developed.

Frogs

The Frogs game (originating from the Comenius Logo package and later redeveloped in Imagine; the task of the game is to order the frogs, but a frog can jump only to a free space if there is one next to it, or only one frog away – see *Figure 3*.) is a very useful, helpful and amusing game, which helps children to practice the order of numbers and the concept of numbers, but it could only handle numbers from 1 to 9 without any changes in the sequence of the numbers and numerical order. Teachers however, needed a tool that also allows the practice of sequences from 8 to 12, or sequences like: 13, 17, 21, or also reverse ordering. After having discussed a framework with the teachers one of our students started to work on its realisation. We decided that the following configurable settings as required:

- Number of the frogs
- Ordering (normal or reverse)
- Starting number
- Steps of the sequence (every number, every second number...)
- Scaffolding (the possibility to see the correct order as a sequence of numbers)

A log file was also required, recording the types of clicks and the time spent on thinking and a visual playback simulation of the log files was also realised in order to trace the process.

The student that got the program development assignment spent lots of sessions at the special education school to see how children (and teachers) manage to make use of it, where modifications might be needed to enhance the user interface, and what parameters might be useful to be inserted in the log file, that records activities. After a long period of iterated developments, a very useful tool was developed with the satisfaction of both teachers and children. The game thus became a very helpful tool to be used in all areas where sequencing numbers is required. It also proved to be a helpful tool for children for self-reflection, in order to construct a more useful algorithm for solving a problem.

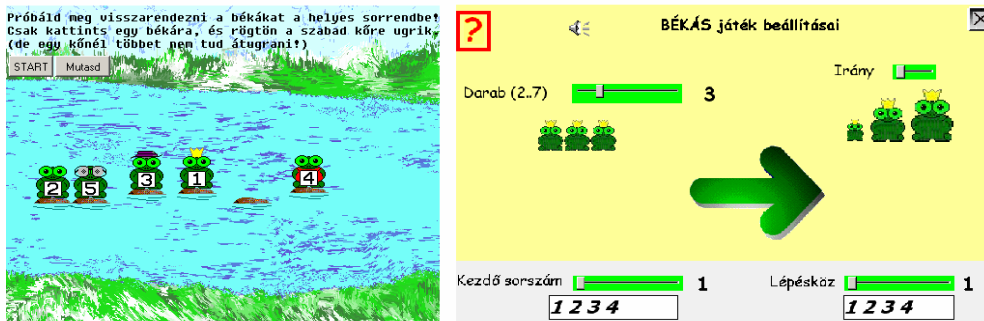


Figure 3. Screens of the Frogs program: original game and configuration page of our version

Grouping

Once teachers have pointed out to the student teachers the need for a program that helps understand the concept of grouping numbers, which is actually a preparation for understanding division. They explained the difficulties in the use of small plastic discs (within the official maths package for elementary schools): they always get lost, it is not interesting enough for children, and it is also difficult to check the result of each child's work. So they asked for such a test like program with configurable number of discs and distribution parameters. Small coloured discs would appear in a given number, which can be dragged, grouped as a visual aid for distribution exercises. So one student teacher made a first version to cover the draft of the ideas, this is shown in the picture on the left of Figure 4. Later an idea for an interesting metaphor emerged from another student: distributing acorn or hazelnut among chipmunks, and so the other students re-designed it with the new user interface. Then, after some trial period, teachers suggested that chipmunks could give the children tips when they went wrong, and what is already solved correctly. Later the problem of bigger numbers appeared, and finally a division part was also requested, where remainders can be handled step-by-step in the process of division.

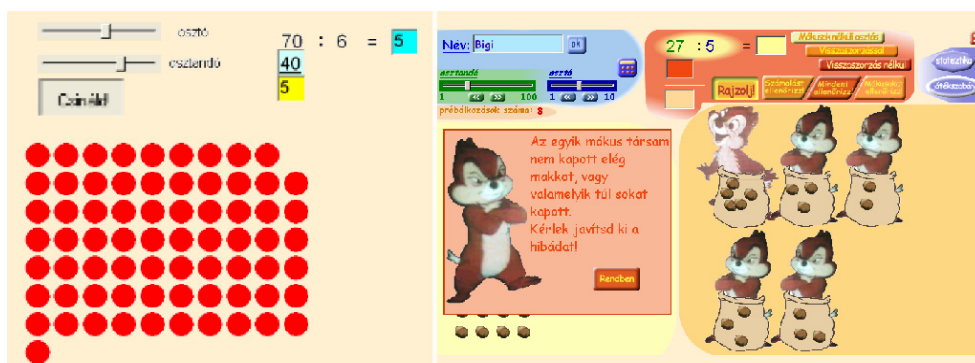


Figure 4. Screens of two iterations of the Grouping program

Evaluation of programs developed

The design process

During the long process of cooperation, we both gained knowledge on requirements for development of effective learning tools. It was and is a win-win relationship. Teachers at Bárczi school have deep knowledge on the needs of children and effective pedagogic methods, at TeaM lab we have many years of experiences in developing learning games for children, besides we bring up a new generation of developers, that are sensitive also to the practical needs of children. It is a fortunate bind between the two institutions as our student teachers are able to experience the use and effects of developed programs by visiting school classes from time to time, chatting with children on their preferences and consulting with teachers on their needs. Programs developed by student teachers (as developers) in such a situation are of much higher quality than if they would be just a normal course deliverable, since it not only has to comply to the course requirements, but also has to fulfil all needs required by school teachers (as customers making the orders) and most of all that of children (as individual users) who are the most critical factor. Student teachers are much more motivated in producing quality work for children, since children are absolutely honest and immediately give feedback if they like it or not. For the past (more than ten) years, we have been practicing this process of program development, having close ties with several types of schools and kindergartens and have experienced the benefits in terms of quality and overall attention of student teachers towards learners (Turcsányi-Szabó, 2006c). Thus TeaM lab has already come out with hundreds of such small educational programs developed within such a process and has produced two considerable learning materials in both English and Hungarian language. These materials also provide learning paths for constructivist developmental learning for children, and for teachers to learn how to configure the programs themselves, one using Comenius Logo (Turcsányi-Szabó, Abonyi-Toth, 1999; Turcsányi-Szabó, 2004) and the other using Imagine as authoring tool (Turcsányi-Szabó, 2006a; Turcsányi-Szabó, 2006b).

Special design elements

At the moment Bárczi school uses 36 different programs developed at our university 34 of which were developed at TeaM lab. 28 of these programs are built into the curriculum and are used in classes. From the 36 programs 15 are built to suit the mathematic curriculum and 11 are designed for the ICT curriculum from fifth to eighth grade. Among the programs 18 produces different kinds of log files, 9 saves the final image when a task has been finished and 9 records the whole process.

The main features of the package so far developed for Bárczi school are:

- **Motivation:** programs are highly motivating and amusing for children.
- **Focus:** all programs focus on a specific theme handled with flexible perspective.
- **Personalization:** programs can be set for the different needs of children concerning level of difficulty, complexity, visual needs, background knowledge and interest.
- **Learning curve and feedback:** activities can be flexibly set in terms of time frame, repetition, feedback, scaffolding, and applicable rewards in order to support the learning curve.
- **Traceability:** activities of children are recorded and can be traced later by checking the log files and saved screen-shots to evaluate the development of the child and the the user interface.
- **Transparent and open:** all programs are developed with the Imagine authoring tool and have similar internal structure that helps further re-developments if needed or media re-configurations by teachers themselves if requested.

The design of log files are of great importance as they are to be used by student teachers during school evaluations in order to monitor progress and pinpoint deficiencies of the program itself.

Most of the log files record the whole sequence of the working process, but it might not be necessary to produce such detailed trails in all cases. E.g. in the program for comparison of quantities, it is enough to record when and which answer was chosen and what the question and the right answers were (see *Figure 5.*). Statistics can also be made immediately and can be stored in files for a quick summery. But in case of the program for using money other events could be more important, since the answer is not a single click, but a sequence of clicks and drag & drops. In this case all clicks that start drags and stop drags are important and are stored in the logs, but as mentioned previously in case of the Frogs program, the visual process can also be reconstructed.

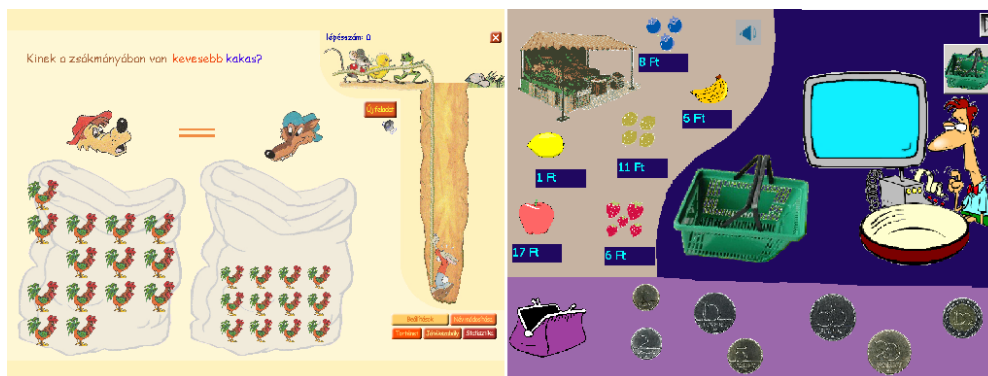


Figure 5. Screen shots of two programs: comparison of two numbers and purchase with money.

We also learned how important it is for children to get immediate feedback on their final answers before pressing the OK button. So we put checkpoints into the programs to allow self-checks before submitting answers. We were interested to learn how often children use these checks and we found out that after working out a solution, they indeed used the self-check every time before submitting a final solution. This feature gave them self-assurance of their thinking process. When such features were not available, children would ask teachers themselves for feedback, which would acquire considerable time from teachers. This is one good reason for the teachers to use such programs in their classes and it also supports self reflection by children and thus provides a more effective learning process.

Scaffolding was also a feature that was required by teachers, e.g. in practicing multiplication, after two wrong answers the multiplication table was shown, and after another wrong answer, the line and column to be used were also highlighted, and next time the needed number was highlighted as well. We had to think of workable algorithms to help the children solve different problems within their abilities, algorithms that provide clear solutions for them as learning skills.

Formal evaluation

The programs developed in earlier semesters were evaluated by student teachers also within course work. During the past ten years we have constructed and refined a rubric for the formal evaluation of educational programs, which has been later adapted by Schoolnet (Sulinet) in Hungary to evaluate the learning objects developed for their repository (SDT). This rubric contains more than 400 items for check, which seems to be an alarming number, but once a single program is tested in such details, the important features for consideration at the evaluation of any educational tool definitely sticks into the head of the evaluator and next time the main titles of check will automatically retrieve in mind the overall feeling of the features to consider. Thus, student teachers could become good advisors for educational programs in other subject areas as well.

All 34 programs were also tested for usability against pedagogical criteria (Nokelainen 2006): learner control, learner activity, cooperative/collaborative learning, goal orientation, applicability, added value, motivation, evaluation of previous knowledge, flexibility and feedback. From these, the teachers rated 29 as usable, 14 were found to have creative

challenges, 25 could be used with an interactive white-board, 25 were found to be suitable to practice everyday life's problems, 18 were rated to be flexible enough for easy configuration to suit the needs of pupils. 18 of these produced different kinds of log files (pictures, statistics or step by step logs) that could be used for evaluation later and 6 of them had text logs on the whole working process.

Among the general programs for maths classes: 15 were built into the mathematics curriculum and teachers are absolutely satisfied with 11 of them; 12 have explicit goals, and 14 allow explicit goals to be given by teachers; 11 are flexible to suit children's needs, 5 have complex settings also for the different special needs of the pupils; and 9 produce log files, 3 create statistics, and 6 have text logs recording the whole working process. Among the programs 6 give direct and immediate feedback to the children, and 6 have motivating feedback animations if the exercise is solved correctly.

The Visual Fractions microworld is a program where exercises (activities) can be designed and integrated as interactive workbooks (scripts). For the fifth and sixth grades 22 scripts have been created, with different tasks and difficulties, starting from tests like distributing a whole, reading out values and pairing values with representations (for the fifth graders) and comparing, up to ordering and notations of fractions (for the sixth graders).

Evaluation of children's progress

Standardized tests do not exist in Hungary to measure dyscalculia or dyslexia, and there are also no tests devised to measure progress of special needs children. So, we had to search for alternative measurement systems that might suit these special circumstances. We found two such to be useful in our inquiry.

The Hiskey-Nebraska Test of Learning Aptitude (H-NLAT) is an individually administered, nonverbal test of learning ability developed for and standardized on a population of children with hearing impairments. This is the only test system on learning abilities which is standardized on individuals who are deaf. The test is suitable for children between ages 3 and 17 and is activity oriented, thus it can definitely not decrease performance if applied with children having reading, writing or communication disorders.

The mathematical ability test (Dékány, 2003) has been used successfully for many years now and was created at Bárczi Gusztáv Faculty of Special Education. According to experiences (not published yet) all children diagnosed to underscore, were later proven to suffer from dyscalculia.

H-NLAT

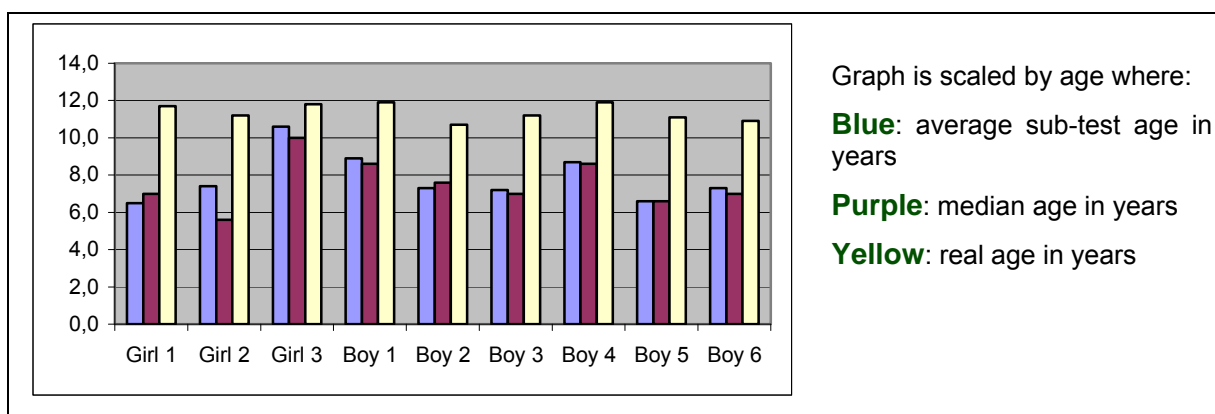
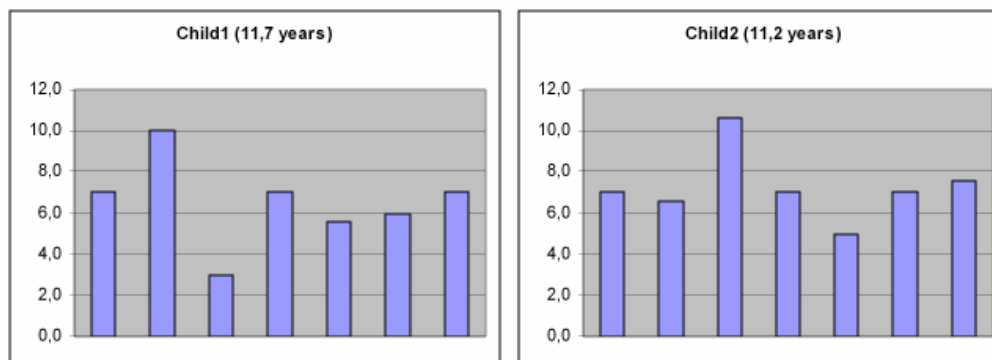


Figure 6. H-NLAT test results of administered group of children

Most children with special needs have big differences between their real age and their mental age. At first we administered the H-NLAT test in 2004 September for the fifth grade (9 children) (Hiskey, 1966; Watson, 1982), where we found out that with an average of 11,4 years of real

age of children, an average of 7,6 years in median mental age could be detected (see *Figure 6.*), and there was also a big difference between each individual child, as well as their mental age in different attitudes, as can be seen in *Figure 7.*



Titles of skills tested are (from left to right): recollection of colours, recognition of pictures, association of pictures, paper folding, extent of visual attention, building out of cubes, completion of drawing.

Figure 7. Individual differences in attitudes

Because of the big difference between individual children's skills and abilities, differential teaching is needed. At this point it is a big help for teachers that they have only a small amount of children in their classes, although these days the situation is worsening and the average number is increasing to ten.

However, the curriculum is not suitable either for the pupils real age nor their mental age, it is somewhere in between the two, and the teacher also needs to take into account the skills and abilities of each individual child to be able to find suitable methods for personalised development and enhancement of the learning process.

Thus it is clear that there is a great need for educational tools that can be customised sensitively enough to suit several levels and stages of knowledge acquisition, where the tools allow the tracing of each child's activities in order to pinpoint any development that might increase level or force some change in configuration to ease its use.

Tests

Mathematical ability tests

Fifth graders were tested on mathematical ability, using rubrics developed at ELTE Faculty of Special Education (Dékány, 2003). These were administered twice, once in September 2004 and then in February 2005. The six children were tested for 11 different skills and abilities (1. attention; 2. graphomotorium; 3. visual perception; 4. acoustic perception; 5. serialisation; 6. thinking functions; 7. thinking in terms of time; 8. speech and language; 9. addition, subtraction, multiplication, inclusion; 10. learning ability; 11. motivation). All aspects were rated from 1 to 5. Part of these tested skills are expected to produce improvements using computer programs (1, 3, 5, 6, 8, 9, 11), and the use of the mathematics programs proved to have effect on five items from the 11 (1, 3, 5, 6).

From the 6 children in the experiment 5 were dyscalculic, and one was found to be endangered. After half a year of using the described programs in classes 33 scores within the test have changed, 20 did not and 16 of the changed scores were in the categories that could have been affected by computer programs. Only one negative change was noted within numerical operation, but it turned out to belong to the only non-dyscalculia child. It is worth mentioning that in two cases significant changes took place in the results, both in the skills area which could have been affected by the computer programs. There was one pupil whose every score has improved by one. It is also important to note, that all children were very well motivated. In

September 5 children got 5 in motivation, and in February all of them got such results in motivation.

As we can see from these results, computer programs had a very good effect on children, helped to keep up their motivation and developed their skills.

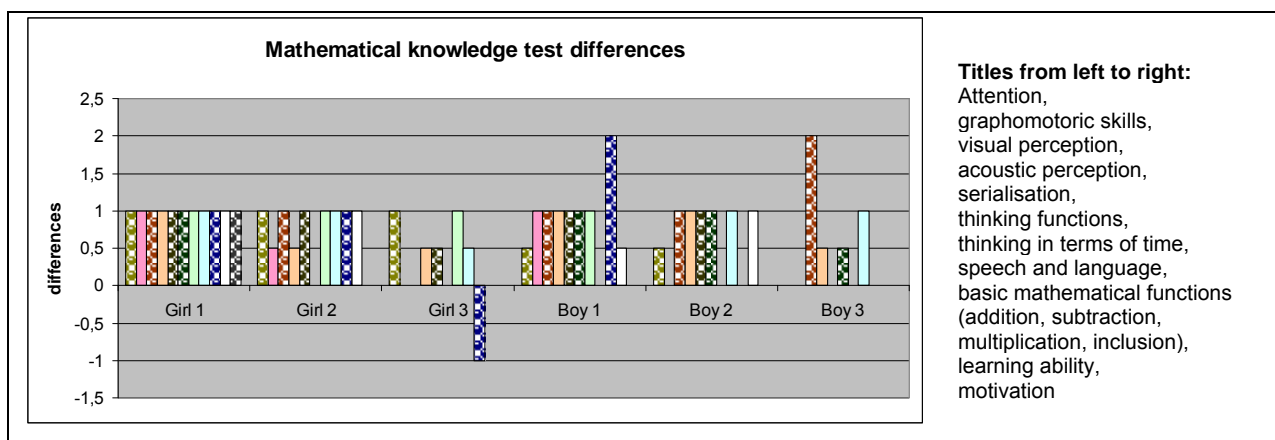


Figure 8. Judit Dékány's test results

Fraction's test

After using the Fractions program for two years in classes, in May 2006 we tested all senior grades (from fifth to eighth), that is 48 children with a written test for understanding and using fractions.

At the time of the testing, fifth and sixth graders learned fractions with the Fractions program, while seventh and eighth graders without it. The 15 test questions were on: 1. reading values (same denominator), 2. inserting relational symbols (same denominator), 3. giving values (by colour), 4. reading the smallest value (nominator is 1), 5. reading the biggest value (nominator is 1), 6. inserting values within their main categories (nominator is 1), 7. reading values on number lines (nominator is 1), 8. inserting relational symbols (nominator is 1), 9. reading values (on scale, same denominator), 10. text assignment on pairing, 11. reading missing values, 12. addition of 2 values (to one whole), 13. taking notes on counting, 14. difference between two numbers, 15. taking notes on counting.

As result, we found 3 questions (3, 7, 11) where the smaller children had a better solving rate then the bigger ones. In the 3rd exercise they attained 96% to 95%, in the 7th exercise 77% to 73% and in the 11th 50% to 45% respectively. These were questions concentrating on reading out values, and understanding values. So we can say that the use of the Fractions program was helping children to develop better models of fraction values for themselves.

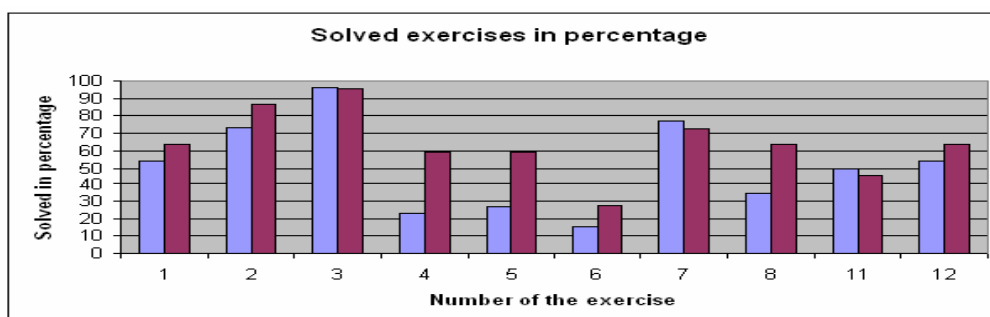


Figure 9. Fractions test results on the different exercises solved by all classes (1,2,3,4,5,6,7,8,11,12)
 Left is for the fifth and sixth grade, and right is for the seventh and eighth graders

Conclusion

Children with learning difficulties need much more time and practice to learn new bits of knowledge. They need much more experience, representations, experiment, explanation, example to extend their knowledge.

As we can see personalise-able programs provide adequate help for teachers working with special needs children. It helps to achieve differential teaching. The motivating effect and the direct feedback of computer programs is also of great help, that makes children more self-confident. Unlimited exercises generated with these programs give children the opportunity for the practise they individually require, to develop new mental models and the ability to be able to extend their own knowledge.

References

Dékány, J. (2003) Kézikönyv a diszkalkulia felismeréséhez és terápiájához (Manual book for diagnostic and therapy of dyscalculias), Logopédiai Kiadó.

Government Decree on the National Curriculum, on web page of the Ministry of Education and Culture: <http://www.okm.gov.hu/main.php?folderID=137&articleID=6982&ctag=articlelist&iid=1> and <http://www.okm.gov.hu/main.php?folderID=391>

Hiskey, M. S. (1966) Hiskey-Nebraska Test of Learning Aptitude.

Hrivnák, I. (2003) "Lusta? Nem szeret számolni? – Diszkalkuliások a közoktatásban" (Is he/she lazy? Doesn't he/she like count? – Dyscalculics in the common education), at <http://www.oki.hu/oldal.php?tipus=cikk&kod=2003-02-be-Hrivnak-Lusta>, Paper presented at Új Pedagógiai Szemle, 2003/2.

Kulcsár, M. (2005) "Tanulási nehézségek" (Learning difficulties), at www.tanulasinehezsegek.hu Logo-ped KM Bt., Bicske.

Mesterházi, Zs. (1996) Diszkalkuliáról pedagógusoknak (From dyscalculic to the educators), Bárczi Gusztáv Gyógypedagógiai Tanárképző Főiskola, Budapest.

Nokelainen, P. (2006). An empirical assessment of pedagogical usability criteria for digital learning material with elementary school pupils. Educational Technology & Society, 9 (2), 178-197.

Turcsányi-Szabó, M., Abonyi-Toth, A (1999). NETLogo teacher training material - Microworlds, A CD product of NETLogo MM1020 project (NETLogo – <http://www.netlogo.org> – not accessible any more).

ed. Turcsányi-Szabó, M., (2004). "HÁLogo portal" (Hungarian NETLogo portal with e-learning material), ELTE University TeaM Lab. Accessible at <http://kihivas.inf.elte.hu/halogo>

ed. Turcsányi-Szabó, M. (2006a). "Digitális Írásbeliség" (Digital Literacy e-learning material), Sulinet Digitális Tudásbázis (Schoolnet Digital Repository), <http://sdt.sulinet.hu/>

Turcsányi-Szabó, M. (2006b). Creative Classroom CD Logotron Ltd, Cambridge, http://www.logo.com/cat/view/creative_classroom.html

Turcsányi-Szabó, M. (2006c). Blending projects serving public education into teacher training. In Kumar, Deepak; Turner, Joe (Eds.) Education for the 21st Century - Impact of ICT and Digital Resources, IFIP 19th World Computer Congress, TC-3 Education, IFIP series Vol. 210, pp. 235-244, Springer. <http://www.springerlink.com/content/k8q6107r3qu60838/>

Watson, B. U. (1982) Test-Retest Stability of the Hiskey-Nebraska Test of Learning Aptitude in a Sample of Hearing-Impaired Children and Adolescent.

LOGO and NQCbaby – programming Microworlds and Robots according to Logo Philosophy

Simonetta Siega, *simonetta.siega@istruzione.it*
I.C. “A. Fogazzaro”, Primary school of Baveno (VB) - Italy

Abstract

This workshop is related to Barbara Demo's and Giovanni Marcianò's long work “Contributing to the development of Linguistic and Logical Abilities through Robotics”, to demonstrate the experiences of my pupils in these last four years.

My goal was to realize a cognitive laboratory in which pupils could develop their logical concepts, according to Feuerstein's and Papert's pedagogical approaches, and using the resources offered by technology. First, using only computers, using the linguistic tool LOGO to give rules and life to the turtles, and then using NQC to repeat the same activity with small robots, the LEGO RCX.

Thanks to Giovanni Marcianò we used an Italian release of David Baum's NQC open-source programming language for LEGO RCX, NQCbaby, in which pupils found once more the LOGO primitives (avanti, indietro, destra, sinistra, ripeti n ...) that they already know well.

Since the first experiences the pupils realized that “The robot is a turtle that became intelligent enough to go out of the computer for running all around the world!”.

Pupils can manipulate the double environment, the virtual multi-medial microworld and the real LEGO RCX robot, taking advantage of both in learning how to program with an artificial language, which is similar to their natural language. But also to develop and reinforce in children logical and linguistic competences.

Turtle competitions in MicroWorlds EX

Sergei Soprunov, *logo@int-com.ru*

Logo Team, Institute of New Technologies in Education, Moscow, Russia

Dorodnicyn Computing Centre, Russian Academy of Sciences, Moscow, Russia

Elena Yakovleva, *logo@int-com*

Logo Team, Institute of New Technologies in Education, Moscow, Russia

Abstract

Creating games of all kinds is a traditional genre of Logo-programming, one of the most fascinating, motivating and important in terms of development of problem-solving skills. (Papert, 1996). The most challenging and most enlightening programming experience is provided by the situations when there is no explicit “best” or winning strategy or describing such strategy is too complicated. (Harvey, 1997). In such cases the programmers are constrained by the necessity to implement just a “good” algorithm instead of the “perfect” one. This gives a very good opportunity for comparing algorithms (more precisely, the programs implementing them) by making the programs to compete, just like it happens between LEGO robots.

Logo and MicroWorlds EX in particular presents a very natural environment for such competitions. Each participant creates a Logo turtle “trained” to follow its own strategy to play a particular game. These turtles are put then in a specially created environment where they compete “on their own”, without human intervention.

The latest version of MicroWorlds, called MicroWorlds EX, has several features that make creation the environments for competitions easy and natural. First, it is the idea of a turtle as an “all-sufficient” object. Each turtle has its own so-called “backpack” – a collection of shapes, properties, rules, procedures, media, etc. - all of its personal characteristics and knowledge. The turtle can be saved as a file along with all its “belongings”. Second, it is the new easy way of event-driven programming. It is implemented in the form of the list of rules kept in the turtle’s backpack. And the last but not least is turtles’ ability to communicate. Any turtle can send a message to a particular turtle in the project. Besides the text of the message, the message contains the “signature” of the sender, so the recipient always “knows” whose message is this. This feature allows to easily organise communications between the participants of the game. “Players”, “referee” and others just send each other messages and properly react to them.

From the other hand MicroWorlds EX lacks the important feature that would be extremely useful for competition projects: primitives regulating the access to the turtles, their procedures, rules and other “belongings”. Such kind of competition may provoke a human-competitor to use some kind of “hacker” tricks: try to control not only his own turtle but also the opponent’s ones and so on. We decided to enrich MicroWorlds EX with commonly practised public/private structure. A turtle can have some primitives and procedures declared public - just like in standard MicroWorlds EX, so everybody else in the project can ask this turtle to execute them. The primitives and procedures that are declared private for this turtle can only be used in this turtle’s backpack. They are not executed when called from outside the turtle.

To present the very structure of the competitions projects a very simple game of Tic-Tac-Toe is described in the paper.

Keywords

Competition, MicroWorlds EX, programming, private procedures, public procedures

Introduction

Have you ever seen robot races or other robot competitions? Here's how it happens.

Each robot is built and programmed by a human-competitor to perform a particular mission. In the course of the contest, people do not control their models, the robots act on their own. People just bring their creatures to the start position, turn them on, and step aside. Every player hopes his creature is smart enough to find a way to the finish and to make it there first. For a programmers competition between virtual creatures see, for example, <http://www.windowsforms.net/Terrarium/WhatIsTerrarium>

Why not have similar competitions for programmers between Logo turtles instead of robots. Relative to MicroWorlds, the idea of cooperative "living" the turtles created by different people in a common environment was expressed by E.Kabakov.

Enriched version of MicroWorlds EX

The latest version of MicroWorlds, called MicroWorlds EX, has some features that make creating the environments for competitions easy and natural.

1. We would like to mention three features. First, it is the idea of a turtle as an "all-sufficient" object. The turtle, which in previous versions of Logo already was "the kingpin", "the star" of the program, becomes in MicroWorlds EX an object that can live independently of the project. Now, each Logo-turtle in the project has its own so-called "backpack" – a collection of shapes, properties, rules, procedures, media, etc. - all of its personal characteristics, knowledge and belongings.

You can save a turtle as a file and, for example, e-mail it. When you get this "fish back into the water" (in a project), it will continue living its life on the new place.

2. Another MicroWorlds EX feature that appeared rather convenient for our purposes is the new easy way of event-driven programming. Each MicroWorlds EX turtle "carries" in its backpack a list of rules. A rule consists of two parts: the description of an event and the list of instructions to run in case the event happens. There's a number of pre-set types of events that happen in turtle's life most often and you can teach it to react to. They are called:

`onclick` – the turtle runs its instruction when clicked by a mouse
`oncolor` – ... when comes across a particular background color
`ontick` – the turtle runs its instruction repeatedly after a preset time interval
`ontouching` – ... when it collides with another turtle
`onmessage` – ... when it gets a "message".

Besides that, you can describe as many events and turtle reactions to them as you wish – and all this is saved in turtle's backpack. So when you place the turtle in another project, it already "knows" how to act in different "situations" and starts acting immediately.

3. And one more brand-new MicroWorlds EX feature that we'd like to mention. It is turtle's ability to communicate. Any turtle can send a message to a particular turtle in the project. The primitive used for that is `tell` recipient message-text. The message contains the "signature" of the sender, so the recipient knows the sender's name.

From the other hand MicroWorlds EX lacks the important feature that is useful for competition projects. In such projects we need to create turtles that not only are complicated but also have a restricted access to them. For example, in the case of football game the turtle playing the role of the ball has to accept rather restricted set of the commands. Otherwise a turtle-player can order the ball to move to an arbitrary place of the game field – for instance, directly score a goal! In the case of the chess game a player shouldn't answer the question "What is your move in this position?" if it's asked by the opponent, but has to answer if it's asked by the referee.

Of course, it's possible to rely on competitor's honesty or check his or her programs manually to make sure that nobody's cheating, but we are playing another "type of a game" – we'd like to have an environment where everything happens automatically. So we decided to enrich MicroWorlds EX with commonly practiced public/private structure. A turtle can have some primitives and procedures declared public – just like in standard MicroWorlds EX, so everybody else in the project can ask this turtle to execute them. The primitives and procedures that are declared private for this turtle can only be used in this turtle's backpack. They are not executed when called from outside the turtle.

We implemented two dual primitives `private` and `public` to set a list of private/public commands and reporters for a current turtle. For example after execution

```
turtle1, public [shape pos]
```

everybody is allowed to ask `turtle1` about its current shape and position, but all other requests will be rejected. Nobody in the project would be able to order `turtle1`, for example, to move or to change its shape.

Now everything is ready to develop the environment for the type of competitions that you like: races, football game, going through the labyrinth – just about anything.

Sample competitions: Tic-Tac-Toe

Let us consider as an example the game of Tic-Tac-toe. Actually, this game is not extremely interesting for organising real turtle competition. We've picked it for an example just because it is quite simple and "good for show".

For real turtle competitions better suite the games that do not have obvious winning strategy or this strategy is hard to program. In this case, actual competition is possible. There's no predetermined winner like in the game of Nim. The players could explore the strategies that are not "best" – and find out experimentally which of them are "better" and which are "worse".

Tic-Tac-Toe is not like that but the main features in the implementation of this game are the same as for more complicated and profound games.

Just like in the robotics competitions, there should be a "game field" and a set of game rules. The field for a game in our case is actually a Logo project, prepared in particular way. For the game of tic-tac-toe, the project includes the 3 by 3 board for placing "ouths" and "crosses" and some kind of "creature" which will supervise the game. This will be, of course, a turtle. We'll call it referee.

The referee would control the game process:

- give a turtle-player a command to make a move,
- get player's move
- check if the move is legal
- and if it is, the referee would accomplish the player's move by "drawing" an X or an O on the board.

The referee communicates with the players via messages, as described above. There is no use in having public commands/reporters for the referee, so all referee's primitives and procedures are private. We launch the game by clicking the turtle-referee, the game is over when the referee announces somebody's win or a tie.

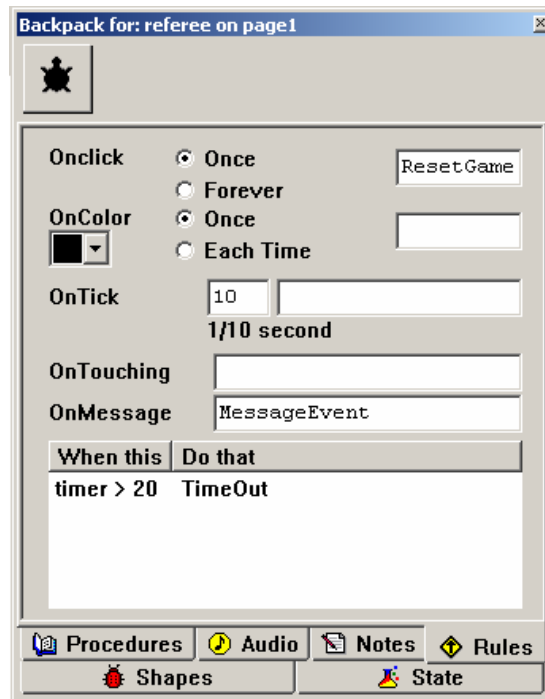


Fig1. The Rules tab of the Tic-Tac-Toe Referee's backpack.

There could be different ways to represent the Tic-Tac-Toe board. We use 9 turtles with the names "cell11", "cell12", ..., "cell33" for the board cells. Each cell can have one of the 3 shapes: **X**, **O** or empty. Everybody in the project can "see" what shape the cell wears, so the `shape` reporter is public for every turtle-cell. From the other hand we don't want to allow a player to directly change cell's shape, so all other primitives (including `setshape`) and procedures are private for the turtle-cells. Only referee can (by sending a message) give the cell the order to change a shape.

Certainly the players (humans) may know the full content of the turtle-referee and the cells procedures but can't change them. Though all these procedures are very simple and straightforward we cite the fragment of the referee instance to make the description a bit clearer.

On the **Fig1.** and **Fig2.** you can see two tabs of the referee's backpack.

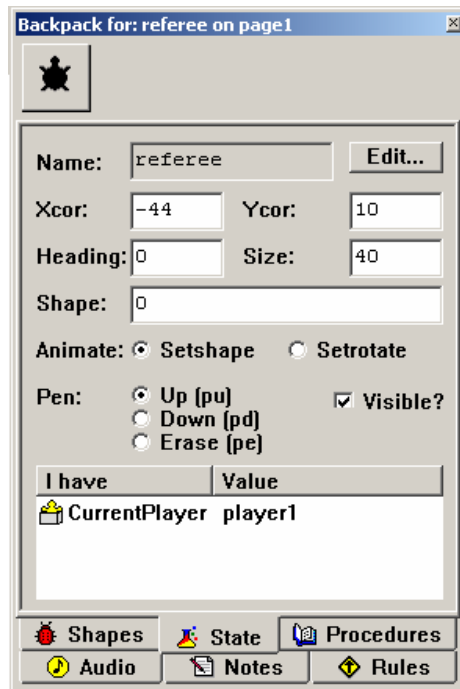


Fig2. The State tab of the Tic-Tac-Toe Referee's backpack.

The referee watches for two events: one predefined (getting a message) and one user defined (communication timeout) (see **Fig1**). The private variable `CurrentPlayer` keeps track of the turns (see **Fig2**).

Here is the fragment of the referee's backpack Procedures tab:

```
to ResetGame
CleanBoard ;CleanBoard sets all cells' shape to "empty"
SetCurrentPlayer "player1
AskForMove
end

to AskForMove
tell CurrentPlayer "your_turn
resett ;this command resets timer
end

to MessageEvent
if not equal? sender CurrentPlayer
    [announce "|The turn is not yours!| stop]
let [cell meassage]
```

```
ifelse CorrectMove? :cell ;CorrectMove? reports if the move is legal
  [tell :cell ResetShape
    if GameOver?      ;GameOver? reports if the game is over
      [announce "|The Game is over| stopall]]
  [announce "|This move is not legal.|]
SetCurrentPlayer ifelse equal? CurrentPlayer
  "player1 ["player2]["player1]
AskForMove
end

to ResetShape
op ifelse equal? CurrentPlayer "player1 ["X"]["O]
end

to TimeOut
  announce (se CurrentPlayer "| seems to fall asleep. His opponent
    wins. The game is over.|)
stopall
end
```

The text of three procedures is missing here: `CleanBoard`, `CorrectMove?`, and `GameOver?`

The latter two are reporters:

`CorrectMove?` reports `false` if the player tries to make a move in the cell that is already busy and `true` in all other cases.

`GameOver?` reports `true` when somebody wins (three X of O in a row) or if all cells on the board are already busy.

The content of the `turtle-player` is almost completely at the programmer's discretion. The player could use just any strategy for playing the game. One, simple strategy, is to put an **X** or an **O** to a randomly chosen out of the still empty positions. Another strategy is to get each cell to have some "weight" and pick an available position with highest weight. There are other strategies possible. There is just one restriction for player's procedures, and it's about the ability to exchange messages with the referee:

- first, the player should know how to accept the referee's message informing that it's time to make a move and
- second, it should be able to tell the referee what empty cell it would like to change to X (or O) on the current move. In our model, this should also be done by sending a message.

As we already said above, Tic-Tac-Toe on the 3 by 3 board is not very challenging game. For real programmers competitions it's much more interesting to use more complicated games, like turtle races, turtle football, or if nothing else Tic-Tac-Toe on the unlimited- size board. "Game

fields” for these competitions can be created in a way, rather similar to a presented Tic-Tac-Toe realisation.

References

- Papert, S. (1996) *The Connected Family*, Longstreet Press, Atlanta, Georgia, pp. 146–151.
- Harvey, B. (1997) *Computer Science Logo Style*, 2nd Edition, Volume 3: *Beyond Programming*, MIT Press

Towards the Construction of a Language for Describing the Learning Potential of Computing Activities

Gary S. Stager, gary@stager.org

Graduate School of Education & Psychology – Pepperdine University – Los Angeles, CA USA

Abstract

This paper represents a first attempt at constructing a language for describing the potential learning value of computers as a learning material. A lack of precision in describing the value computers add to the learning process has paradoxically made it easy for people to elevate the significance of using computers in pedestrian ways while simultaneously marginalizing higher-order uses such as Logo programming. Colleagues are invited to extend or challenge this paper's hypotheses.

In the early 1980s Seymour Papert was dissatisfied with Robert Taylor's metaphors for the use of the computer in education. Taylor wrote about the computer as a tool, tutor or tutee (Taylor, 1980) while Papert described the computer as "mudpie" (Papert, 1980a; Papert, 1984) and then later more generally as material. (Papert & Franz, 1987) The tool metaphor dominates most discourse regarding the use of computers in education. Educators and policy-makers alike use it to describe nearly every application of "technology." It would be impossible to list all of the examples of "computer as tool" in common usage or even scholarship.

This work attempts to define the continuum that lies between the use of computers to reinforce traditional practice and the powerful ideas Papert writes of in Mindstorms. (Papert, 1980b) While Papert's subsequent work provides examples of the construction of powerful ideas he fails to identify less powerful uses of computers. This may be the result of simple omission or a desire to appear polite. In either case all manner of computer-based activities have been granted equivalence by an education community lacking a precise metric for assessing value. When combined with the liberal and often inaccurate use of terms like constructivist we are left with a culture of intellectual relativism in which the loudest voice sets the standard.

Dichotomies like conservative/liberal, traditional/progressive, Democratic/Republican are inadequate for describing educational philosophy and its resulting translation into practice. Papert's instructionism vs. constructionism seems a more precise way of describing one's learning theory and the practice that follows.

It seems impossible to invent an empirical metric for measuring the efficacy of computer use in the context of education. There are simply too many variables involved in a complex system such as education. The nature of learning is even more difficult to quantify in anything but a reductionist fashion. Therefore, I propose the creation of a continuum that spans the gulf between traditional education routines possibly enhanced by the use of a computer and the sort of powerful idea construction only possible with the purposeful use of the computer. The subjectivity of the examples are acknowledged, but are intended to generate discussion.

Compelling examples of children, computers and powerful ideas will be presented at Eurologo.

Keywords

constructionism; powerful ideas; descriptive language

Other attempts to assess educational computing

Efforts to describe differences in education approach or outcome often descend into the creation of an assessment system. It is human nature to then label, rank, sort and assign merit or value to each action or result. Existing paradigms for describing educational computing are often reduced to simple rubrics or checklists that may be used to “grade” performance. The assumption is that such external measures will then be used to motivate or shame educators. Neither is my objective.

Even if we were to succumb to such behaviorism, the politics of schooling often values forms of learning devoid of powerful ideas. Memorization, mechanics and conformity are often prized at the expense of critical thinking, creativity and the free exchange of ideas.

Existing models of evaluating the use of computers in education are flawed in critical ways.

- The focus is too often on pedagogy or product.
- The use of computers is used to support the existing curriculum.
- “Successful” use is based on the integration of computers into the existing curriculum. The focus of computer integration is on what the teacher does.
- Technocentrism leads to the hyperbolic description of existing practices.
- A lack of imagination and curricular arrogance predicts narrow outcomes and diminishes the serendipitous possibility of learning new things.
- A shortage of compelling models of powerful idea construction narrows what learners might do with computers.
- Content is rarely questioned.

The focus is too often on pedagogy or product

Some states, school districts and national departments of education have created instruments for assessing the impact of educational technology. It is beyond the scope of this study to review all but the best known of these schemes. The Levels of Technology Implementation (LoTI) is a popular “instrument for measuring technology use.” (Moersch, 1996-97) The LoTI framework describes six levels of computer efficiency from non-user to refinement. “As a school site progresses from one level to the next, a corresponding series of changes to the curriculum is observed. The instructional focus shifts from teacher-centered to a learner-centered orientation.” (Moersch, 1996-97)

The following table describes the LoTI Scale used to evaluate educational technology use. (see <http://www.drchrismoersch.com/loti.html> for details)

0) Nonuse	1) Awareness
2) Exploration	3) Infusion
4a) Integration (mechanical)	4b) Integration (routine)
5) Expansion	6) Refinement

The LoTI Scale represents just one variable in the complex arithmetic calculation required to calculate a classroom’s level of computer efficiency. Moersch validates his metric by comparing his findings to those of Becker (1995) who used a survey to determine “exemplary computer-using educators.” (Moersch, 1996-97) Apparently the results of Moersch’s Computer Efficiency formula mirror the results of Becker’s survey. This hardly proves the accuracy or educational value of a set of calculations dependent on such variables as the number of computers in a classroom and the amount of time they are used.

Moersch defines computer efficiency as “the degree to which computers are being used to support concept-based or process-based instruction, consequential learning, and higher order thinking skills (e.g., interpreting data, reasoning, solving real-world problems).” (Moersch, 1996-97)

Moersch reinforced his stance when he wrote, “the level of computer efficiency is influenced directly by how teachers are using computers to develop students’ higher-order thinking skills.” (Moersch, 1996-97)

If one could set aside the Dickensian goal of measuring computer efficiency and peculiar formula for deriving it, LoTI is consumed by larger intellectual inconsistencies. U.S. States offer “LoTI training” and require teachers to take 20-minute LoTI surveys which in turn make recommendations for “increasing their current levels.” (<http://www.nheon.org/oet/loti/>) Other agencies overlay LoTI on top of Bloom’s Taxonomy (<http://www.fisd.us/LoTi/lotisniffest.htm>) when one system is about learning and the other teaching. Several examples of LoTI Teacher Self-Assessments published on the Web don’t even include the use of technology despite that being an integral part of LoTI. While it would be unfair to dismiss a theory based on its application by laypeople, LoTI itself is replete with inconsistencies, not the least of which is its constant use of the term, instruction, despite a commitment to constructivism. The fact that LoTI describes the “school site” shifts the locus away from the learner. Such anomalies undermine Moersch’s assertion that LoTI is empirical (Moersch, 2001)

Although Moersch writes extensively about a desire to shift the focus from teacher to learner his practice and the examples he offers remain firmly focused on teaching rather than on learning. One need not read more than the LoTI Framework to determine that nearly every example of technology use described by Moersch (Moersch, 1995, 1996-97, 1998a, 1998b, 1998c, 2001) is teacher-centered despite rhetoric to the contrary. Despite protests to the contrary the number of computers in a classroom, seat-time and externally imposed curricular goals are critical elements in Moersch’s calculus. His expressed commitment to constructivism and a “learner-centered orientation” is at best confused and at worst serves to camouflage the very practices he seeks to reform. Since this paper seeks a precise language for describing the learning potential of computing activities, LoTI is of limited value.

4MAT is another taxonomic system purporting to support and respect individual learning styles, except the theory’s application is focused explicitly on the creation of lesson plans for teaching specific content. Again, the distinction between learning and teaching is blurred in a way favoring pedagogy. (anonymous, 2007; McCarthy, 2007)

Porter’s work in evaluating student digital products is more consistent in its approach and language, but suffers from a focus on curriculum related products. Some of these products are more personal than the result of imposed curriculum, but the focus on the quality of the artifact does little to assist my quest for a language for describing learning activities. (Porter, 2001)

Integration and technocentrism

Computer integration into the existing curriculum regardless of its rigor, creativity or level of student engagement holds limited potential as a catalyst for powerful ideas. Efforts at integration assume the relevance and value of the existing curriculum while curriculum by its very nature is a map used to steer teaching practice. Efforts to improve curriculum integration support instructionism, the belief that education results from transmission and is informed by forces outside of the learner. On the other hand, Papert’s theory of constructionism builds upon the Piagetian notion of constructivism in which knowledge is constructed by the learner and suggests that the best way to ensure such learning is through the act of making something sharable. (Ackerman, 2001; Papert, 1993, 1991; Papert *et al.*, 1991; Stager, 2002; Stager, 2007; Turkle & Papert, 1991) The computer expands the range of things one can construct and provides a means for sharing ones invention; whether it’s a poem, a computer program, a robot or a film.

Few examples of computers being used as incubators for powerful ideas exist in the educational technology literature or in common practice. Either lack of imagination or a desire to preserve the status quo leads to the creation of formal documents, such as the National Educational Technology Standards in the USA produced by august sounding bodies like The Partnership for 21st Century Skills or the International Society for Technology in Education (ISTE, 2000, 2007;

Partnership for 21st Century Skills", 2000). In fact, the new NETs fail to mention either computer science or programming despite an expressed commitment to technical fluency, creativity and invention. Such documents and their creators suffer from what Papert called *verbal inflation* at the 2005 K-12 Conference on School Networking in Washington D.C. (Papert, 2005)

Verbal inflation, Papert explained, was the use of exaggerated language to describe very little actual transformation or change in practice. Verbal inflation is often accompanied by technocentrism when an educational activity is overvalued due to the presence of a computer. "Technocentrism is the fallacy of referring all questions to the technology." (Papert Technocentrism) Examples of the intersection of verbal inflation and technocentrism include the use of "office" software to "prepare children for the real world;" word processing your book report rather than writing it with a pen; using PowerPoint to present five facts about frogs or using the web for "research" instead of an encyclopaedia when the goal is paraphrasing a couple of paragraphs. Paradoxically, it is the technocentric focus on mechanical skills or specific software applications that denies children any deep understanding of computing or agency over the device central to their lives.

Disruptive semantic trends

Over the past twenty years I have witnessed a semantic shift transforming the words used to describe our field from *educational computing* to *technology* to *information technology* or *ICT*. Computing is a verb, something one does. Technology is a noun made even more passive when modified by information. The implication is that the dominant metaphor for computer use in school is information retrieval, not the personal construction of knowledge.

Part of learning is getting information. Somebody stands in front of the classroom and preaches, and information is somehow flowing into people's heads, or so it is said. But that's only one part of education. The other part, which Dewey would have emphasized, is about doing things, making things, constructing things. However, in our school systems, as in the popular image of education, the informational side is again dominating.

There is a parallel between an unrecognized dichotomy in digital technology and a generally unrecognized dichotomy in the education system. In both cases the informational side is best known to the general public. So the image of computers in school supports the traditional role of the teachers in their part of education-providing information." (Papert, 1998)

The use of the word *technology* is almost exclusively synonymous with *computer*. However, the generic term, *technology*, implies less potential for revolutionizing learning than *computing* which requires the purposeful actions of a user expressing new fluencies. This rhetorical trend mirrors the recent political shift in schooling away from individuality towards conformity and homogeneity. National standards and curricula move the locus of control from the learner to the system; from construction to delivery.

Content

Most efforts at educational reform are concerned with changes in pedagogy or the materials used. Rarely is the content reviewed, removed or changed. Educational leadership must be concerned with subtraction as well as addition. The desire for students to master new content and develop modern skills cannot always result in the addition of new requirements to an brimming list of requirements. Some content must go.

Content dictates what children do. Since knowledge is the consequence of experience (Smith, 1995), content influences the learner's actions and determines the relationship to the knowledge they construct. The seemingly simple question, "What do you do with computers?" provides more information about the learning experience than any complicated rubric.

A failure to make new content accessible not only reduces a learner's opportunity to construct modern knowledge, but runs the risk of making education less relevant and students more

passive. New content may not only inspire learners, but also provide a context in which additional concepts gain power. For example, a student “messing about” with a number theory problem will internalize arithmetic. A student writing a program in French will learn a lot of computer science, mathematics and problem solving, plus become more fluent in French and perhaps learn about the system being simulated as well. Building a robot designed to pull a great deal of mass requires an understanding of friction, force, gearing, ratio and a host of other concepts. Most importantly, prior knowledge is used to construct new understanding. New compelling models of learning with computers are essential if others are to follow our example.

Engagement

The desire to achieve a different learning outcome without changing content is evident in educators who speak of student engagement with computers. I often hear, “the children are so engaged.” Hardware and software companies use engagement as a marketing tool. This is a wonderful result if authentic engagement is possible. However, it may not be. Papert argues that some “school math” is so toxic that it is impossible to make it engaging without trivializing the experience. The result is a lack of rigor and powerful ideas that leaves progressive educators exposed to unpleasant criticism from instructionists.

“When ideas go to school they lose their power, thus creating a challenge for those who would improve learning to find ways to re-empower them.” (Papert, 2000) Papert describes how even big ideas, such as probabilistic thinking, are disempowered by traditional curriculum and the pencil and paper technology of school. “It’s been disempowered because you couldn’t give kids any way of using it.” (GLEF, 2001)

“In a pencil and paper environment, it is very hard to be creative with mathematics. The great contribution of computers is that, it is now possible to use mathematical ideas to make things that kids care about. Making their own game. Making artwork. Turning mathematics through these activities into a useful tool for something that kids really care about. This is the secret to mathematics education. NCTM is just blind because it assumes that mathematics will always be done with pencil and paper.” (*Professor papert discusses one laptop per child project*, 2006)

Probability is a powerful idea fundamental to modern mathematics, science, economics, social science and even the arts. Yet, this powerful idea is often sacrificed by directed activities in which children ask classmates their favorite flavor of ice cream and then “predict” a new student’s preference. (Papert, 2000) This school version of probability is predicated on primitive technology.

Papert suggests that rather than find yet another way to teach math that kids hate, we should invent a mathematics they can love. Such a mathematics is likely to more closely resemble the real work of actual mathematicians and have more authentic application in the 21st Century than what is taught in math class. Building a robotic “bee” trying to find pollen or programming a StarLogo simulation situates students in a context for using the powerful idea of probability.

“We’ve got the technology to be able to have kids solve for themselves the kind of problem that nature solved using randomness. But of course, that doesn’t fit into the second-grade curriculum, so we don’t do it. Or we reduce probability to some little spinner and see how often (the number) six comes up. Who cares how often six comes up? You can’t do anything with it.” (GLEF, 2001)

Frankly, very few educational practices are borne of student desire. “They are so engaged” is often used as justification for questionable practice. The belief that learning should be hard and unpleasant often accompanies cries of engagement. However, engagement need not be superficial or technocentric. It may accompany rigor, purpose and creativity. Engagement is the result of powerful ideas, not a substitute.

“Kids like computers... I think it corresponds to children wanting to control an important part of the world... They can feel the flexibility of the computer and its power. They can

find a rich intellectual activity with which to fall in love. It's through these intellectual love affairs that people acquire a taste for rigor and creativity." (Brand, 1988)

Some content leaves learners hostile or reluctant to learn. If the old content or skills are so invaluable, they will be learned in the context of learning something else. Repetitive demands to learn what may be, at least temporarily, unlearnable may diminish a student's motivation, result in learning pathologies and reduce the chances of learning that content at a later date.

Abandoning content, after careful reflection, is not an admission of failure. It may be an act of liberation - opening the door to new learning adventures.

When faced with declining enrolment in university computer science and substantial attrition rates following the introductory course, Guzdial and Soloway did not search for a new way to teach better. They examined the course content and decided to replace curricular staples, such as sorting algorithms, with the creation of web spiders and graphic manipulation programs. This content was more current, relevant and challenging. The content shift allowed students to not only do more sophisticated work, but it also improved student attitudes towards the study of computer science; leading to further matriculation. (Guzdial & Soloway, 2003)

"We should change the way we talk about schools by talking less about learning and teaching, and more about doing. When we focus on teaching specific skills, students frequently fail to learn them and rarely become enthusiastic about engaging in them voluntarily. When we concern ourselves with engaging students in interesting and comprehensible activities, then they learn." (Smith, 1995)

A reluctance to review traditional content may be based on heuristics, but it may also be based on the reluctance of some teachers to develop new skills and subject matter knowledge. Digital learning communities extending beyond the four walls of the physical classroom may offer students access to expertise unavailable in school.

Describing the potential of an activity

If we define content as the focus of an activity, then we may evaluate the quality of the activity. A more precise language is needed to describe the potential for encountering powerful ideas. The primacy of the activity must be our focus if we are to articulate the ways in which computers may enrich the learning process.

I have grappled with the creation of a matrix suitable for explaining complex learning theories. I learned a lot personally, especially while brainstorming with a colleague about whether the increase in the educational value of a computing activity is additive, multiplicative or something else. However, I have yet to determine a formula for predicting the probability of encountering powerful ideas. Deriving such an algorithm is likely to be impossible.

It is clear that there are overlapping ways of describing any educational activity. A series of continua represent agency, the novelty of activity, the learning theory expressed, the contribution of the computer and the degree of creativity involved. These descriptions fall along continua, including:

Traditional Activity	To	Novel Activity
No Computer Use	To	The Computer is Integral
Teacher Agency	To	Learner Agency
Instructionism	To	Constructionism
Replication	To	Invention
Routine Activity	To	Transformational Activity

When these continua are combined, activities may be described by points along the span between *Routine Activity* and *Transformational Activity*. At one end of the continuum traditional

content is presented in a teacher-centered fashion with little or no use of the computer. At the other extreme a person learns in a personally meaningful fashion resulting from the critical role the computer plays in maintaining a conversation with the human user. The activity is impossible without computational power. The learner might experience “flow” (Csikszentmihalyi, 1991) while the answer to a good question leads to an even better question or a more complex hypothesis. “Bugs” are an opportunity for the learner to rethink their strategy or try an alternative approach. A successful action by the learner may lead to a serendipitous discovery or motivation to attempt a more challenging feat. Activities falling in the right-hand column are demonstrably richer because of access to computers and open-ended software or programming languages, such as Logo. Transformational activities offer the greatest potential for encountering powerful ideas.

My goal in life... has been to find ways children can use this technology as a constructive medium to do things that no child could do before, to do things at a level of complexity that was not previously accessible to children. (Papert, 1998)

Vignettes along the continuum

For the purposes of this study, the examples provided will be mathematical in nature. Other domains may be explored in subsequent work. However, it seems obvious that an activity like digital movie making would progress along the continuum based on well-established aesthetic values. Evaluating how well the movie entertains, communicates, inspires, surprises, enrages or engages the audience are of greater value than how many transitions were in the movie, if special effects were included or if there were more than three people interviewed. The isolated technical skills assessed by teachers armed with rubrics are of less importance than the learning experience of the learner and her audience or peers.

Although far from empirical, it may be possible to divide the continuum into ten units or five pairs of units.

Routine Activity – Teacher-centered	Little Impact of Computers
Level 1 A student solves dozens of similar arithmetic problems on a worksheet in an attempt to memorize his multiplication tables.	Level 2 A student uses a piece of computer-assisted software to play a game in which the frequency of problems presented increases after a correct answer. This is thought to increase recall of math facts.
Explanation of Levels 1 & 2: Level 1 describes an activity that is teacher-directed, routine and does not require or benefit from the use of a computer. In Level 2, the computer may make the activity a bit more fun or even lead to slightly greater efficiency. It hardly improves the learning of arithmetic or situates it in a meaningful context.	
Level 3 A student uses tactile manipulatives to make patterns on her desk. Tangrams or pattern blocks may be used. The teacher may expect that terms like symmetry or tessellation will be remembered as a result of the activity.	Level 4 Computer software provides virtual manipulatives on the computer screen that allow a child to produce an infinite number of a piece, change their color, save and print the designs created.
Explanation of Levels 3 & 4: Level 3 uses tactile objects to make geometric concepts more concrete, but those concepts remain decontextualized and the activity only exists because of a teacher’s insistence. Many advocates of educational computing would view this level 4 activity as innovative even though a purpose for using manipulatives remains inauthentic and a mystery to the user (perhaps the teacher as well). The features of the software may lead to an impression of what David Squires called, “false complexity,” even when the activity itself may be of little merit.	

<p>Level 5</p> <p>A child uses Logo to write procedures replicating the shapes found in the assortment of physical manipulatives. The teacher may explain the “total turtle trip theorem” at the board.</p> <p>or</p> <p>The teacher uses Geometer’s Sketchpad and a projector to present a new concept to the class.</p>	<p>Level 6</p> <p>A child develops a strategy for writing Logo procedures that allow the virtual manipulatives she created to be moved, oriented and tessellated.</p> <p>or</p> <p>A student uses Geometer’s Sketchpad to explore forms of symmetry or to draw a line through the perpendicular bisector of a figure.</p> <p>or</p> <p>The teacher challenges students to use LEGO robotics materials and Logo to build a vehicle that goes down an incline very slowly. This requires the use of gears and exploration of physical science principles.</p> <p>or</p> <p>The teacher instructs each student to create an Excel spreadsheet to find the average of five numbers.</p>
<p>Explanation of Levels 5 & 6: In both level 5 activities, the computer is used to teach geometric concepts that the teacher or set curriculum requires. Student motivation is not a concern. The turtle geometry activity does offer the possibility that students will learn the shapes with greater understanding and comprehension since they are “teaching” the turtle to draw them; therefore describing the relationships that form the shapes.</p> <p>Although the level 6 activities are anchored in the curriculum, the computer is essential and the students may express a bit more autonomy, ownership and divergent thinking. However, it is possible to have Geometer’s Sketchpad draw the perpendicular bisect without the student having any greater understanding of the concept than had it been presented without a computer.</p> <p>Level 6 represents the point at which students are first engaged in projects where they are actively engaged in making something as a way of constructing knowledge.</p>	
<p>Level 7</p> <p>A child designs an interface for her virtual manipulatives that allow the pieces to be stretched, shrunk, colored differently and overlapped. The interface is designed for her friends to use in making their own original designs.</p> <p>or</p> <p>A student uses Geometer’s Sketchpad to understand a concept that would otherwise taught three years later.</p> <p>or</p> <p>The class is engaged in a thematic unit about carnivals. A group of eight year-old girls decide to use LEGO and Logo to make a stuffed teddy bear dance. A skeletal system must be built that can transfer the rotational motion of the motors into the up and down motion of arms and legs.</p> <p>or</p> <p>Tim is able to use Excel to create a catalog of his baseball cards, complete with each card’s current value and is able to find out how much he might earn if he sold the entire collection.</p>	<p>Level 8</p> <p>A student uses Geometer’s Sketchpad to help perfect a skateboarding move.</p> <p>or</p> <p>The girls decide they would like their robot teddy bear to sing. They locate a piece of sheet music, convert all of the notes, rests and durations to numerical values Logo will understand and once they complete their program they ask it to play. The music plays too quickly, but the intervals appear to be correct. The girls brainstorm and determine that multiplying each duration by a constant will slow the music down.</p> <p>or</p> <p>Tim manipulates Excel so he may explore how much money he might earn if he just sold the cards of Yankees players. He can also project how much his collection might be worth by the time he goes to university based on information he found on the Web.</p> <p>or</p> <p>Michael invents a LEGO robot, programmed in Logo that graphs fluctuations in temperature over</p>

<p>or</p> <p>Each student in Miss Crabtree's class is asked to create a database containing the address and phone numbers of at least four friends.</p>	<p>multiple days using one roll of adding machine tape and a mechanism with a complex gear ratio.</p> <p>or</p> <p>A five year-old girl wants to make a dancing ballerina out of LEGO and programs it to spin via Logo. The ballerina has two touch sensors that allow the girl to spin it left or right. She changes the rate of spinning by using different combinations of gears, by changing the voltage being sent from the computer to the LEGO brick and by inserting wait commands to her Logo program.</p>
<p>Explanation of Levels 7 & 8: The level 7 activities are much more dependent on the computational power of the computer, although the projects themselves remain consistent with the artificial nature of the curriculum in which teachers are told to teach specific concepts or tools at a specific time. Using Geometer's Sketchpad to learn something previously taught at a later time demonstrates the value of the computer in making sophisticated concepts accessible at an earlier age by concretizing them.</p> <p>Level 8 activities mark a significant shift in agency between the desires of the teacher and those of the learner. Learners engage in personally meaningful projects requiring the use of the computer as material. Invention, ingenuity and intrinsic motivation are critical aspects of levels 8-10.</p>	
<p>Level 9</p> <p>Rather than use Geometer's Sketchpad to draw geometric figures and observe corresponding tables of values. Students use Microworlds EX to design their own geometry toolkit. The addition of each successful feature leads to the addition of new functionality. Defining midpoint becomes a tool for finding the area of a triangle. Using sliders representing <i>length</i> and <i>exterior</i> angle allows the students to design a tool for drawing regular polygons. A more sophisticated understanding of geometric terms results from teaching those concepts to the computer in the form of a program.</p> <p>or</p> <p>Each student locates census, economic, health, agriculture or political data for an entire state or nation. Thousands of records are involved. Importing that tab delimited data into a spreadsheet or database program allows each student to interrogate the data and perhaps answer a question nobody has ever asked before. Graphs and charts of trends may be presented to their peers.</p> <p>or</p> <p>An unsolved number theory problem, the Hailstone Problem, becomes a source of good natured rivalry between students looking for interesting patterns while simultaneously using a Logo-based toolkit to discredit the hypotheses of their peers.</p> <p>or</p> <p>Michael uses calibrates and validates the accuracy of his LEGO instrument and uses it to monitor an experiment in the science lab.</p>	<p>Level 10</p> <p>Students present what they learned from their careful data analysis to the government in order to advocate for a new swimming pool, cleaner rivers or after school programs for children of single parents.</p> <p>or</p> <p>Susan "Googles" "the Hailstone Problem," learns that there is an annual conference for mathematicians dedicated to the problem, emails the organizer of the conference and develops an ongoing dialogue about number theory.</p> <p>or</p> <p>The graph produced by Michael's scientific instrument leads to further investigations in the lab.</p>

Explanation of Levels 9 & 10: The sophisticated activities described in level 9 are learner-centered, yet consistent with curricular objectives. The activities are completely dependent on computers and open-ended software. The projects allow for a significant amount of student creativity, problem solving and critical thinking. Correct and incorrect answers are no longer the goal or perhaps even possible. New forms of modern knowledge are accessible to the learners because of the nature of the activities and the power of computer. Learners construct powerful ideas related to a variety of disciplines.

Learners in level 10 are able to use communication and computational technologies to engage in an intellectual (or creative) community of practice outside of their classroom. They may not only share their newly constructed artefacts and the resulting knowledge with peers, but with the community and other experts. It is at this level that learners are doing the real work of mathematicians, engineers, scientists, composers, poets, etc. It is quite possible for level 10 students to make genuine contributions to knowledge.

Transformational Activity – Learner-centered	Computer Use is Essential
--	---------------------------

Activities at Levels 1-5 do not require the use of the computer. Its use tends to be gratuitous in such activities and contributes little value to the learning experience. Activities at Levels 6-10 are dependent on the computer. The computer not only enhances the learning experience, but is the material at the center of the knowledge construction. The value added by the computer increases as the nature of the activity becomes more modern, learner-centered, constructionist, complex and inventive. It is at the nexus of these factors that powerful ideas become accessible.

“One can take two approaches to renovating School - or indeed anything else. The problem-solving approach identifies the many problems that afflict individual schools and tries to solve them. A systemic approach requires one to step back from the immediate problems and develop an understanding of how the whole thing works. Educators faced with day-to-day operation of schools are forced by circumstances to rely on problem solving for local fixes. They do not have time for big ideas.” (Papert, 2000)

Experiences, such as Logo, remain viable as long as educators are able to articulate compelling descriptions of the activities in which the learner participates. The telling of these “learning stories” (Papert, 1993) is dependent on more precise language capable of differentiating between the potential value of an activity. This paper represents an attempt to discuss the construction of such a language.

References

- Ackerman, E. (2001). *Piaget's constructivism, papert's constructionism: What's the difference?* Paper presented at the 2001 Summer Institute, Mexico City.
- anonymous. (2007). Learning styles and the 4mat system: A cycle of learning. *Learning styles and the 4mat system: A cycle of learning*. Retrieved April 1, 2007, 2007, from <http://volcano.und.edu/vwdocs/msh/lc/is/4mat.html>.
- Brand, S. (1988). *The media lab: Inventing the future at m.i.t.* NY: Penguin.
- Csikszentmihalyi, M. (1991). *Flow: The psychology of optimal experience* (Reprint ed.). NY: Harper Perennial.
- GLEF. (2001). Seymour papert on project-based learning. In Edutopia (Ed.), *Expert Interviews*: George Lucas Foundation.
- Guzdial, M., & Soloway, E. (2003). Computer science is more important than calculus: The challenge of living up to our potential. *ACM SIGCSE Bulletin*, 35, n. 2(2).
- ISTE. (2000). *National educational technology standards for students: Connecting curriculum and technology*. Eugene, Oregon: International Society for Technology in Education.

- ISTE. (2007). *Nets refresh standards (draft)*. Unpublished manuscript, Eugene, Oregon.
- McCarthy, B. (2007). About learning web site. *About learning web site*. Retrieved April 1, 2007, 2007, from <http://www.aboutlearning.com>.
- Moersch, C. (1995, November 1995). Levels of technology implementation (loti): A framework for measuring classroom technology use. *Learning & Leading with Technology*.
- Moersch, C. (1996-97, November 2001). Computer efficiency - measuring the instructional use of technology. *Learning & Leading with Technology*.
- Moersch, C. (1998a, May 1999). Assessing current technology use in the classroom: A key to efficient staff development and technology planning. *Learning & Leading with Technology*, 26.
- Moersch, C. (1998b, March 1998). Enhancing students' thinking skills: Exploring model technology-integration sites. *Learning & Leading with Technology*, 25.
- Moersch, C. (1998c, November 2002). Measures of success: Six instruments to assess teachers' use of technology. *Learning & Leading with Technology*, 30.
- Moersch, C. (2001, November 2001). Next steps: Using loti as a research tool. *Learning & Leading with Technology*, 29.
- Papert, s. (1980a). Computer-based microworlds as incubators for powerful ideas. In R. Taylor (Ed.), *The computer in the school: Tutor, tool, tutee* (pp. 204-210). NY: Teacher's College Press.
- Papert, S. (1980b). *Mindstorms: Children, computers, and powerful ideas*. New York: Basic Books.
- Papert, S. (1984). Computer as mudpie. In D. Peterson (Ed.), *Intelligent schoolhouse: Readings on computers and learning*. Reston, VA: Reston Publishing Company.
- Papert, S. (1993). *The children's machine: Rethinking school in the age of the computer*. NY: Basic Books.
- Papert, S. (1998, June 2). *Child power: Keys to the new learning of the digital century [lecture transcript]*. Paper presented at the Eleventh Colin Cherry Memorial Lecture on Communication, Imperial College, London, UK.
- Papert, S. (2000). What's the big idea? Toward a pedagogical theory of idea power. *IBM Systems Journal*, 39(3&4), 720-729.
- Papert, S. (2005). Beyond wires and boxes: Technology as a tool for educational transformation, *Presentation at COSN's K-12 School Networking Conference*. Washington D.C.: Consortium on School Networking.
- Papert, S. (Ed.). (1991). *Situating constructionism*. Norwood, NJ: Ablex Publishing Corporation.
- Papert, S., & Franz, G. (1987). Computer as material: Messing about with time. *Teachers College Record*, 89(3).
- Papert, S., Harel, I., & Group., M. I. o. T. E. L. R. (1991). *Constructionism: Research reports and essays, 1985-1990*. Norwood, N.J.: Ablex Pub. Corp.
- Partnership for 21st Century Skills. (2000). *Learning for the 21st century: A report and mile guide for 21st century skills*. Tuscon, Arizona: Partnership for 21st Century Skills.
- Porter, B. (2001). *Evaluating student digital products: Training and resource tools for using student scoring guides*. Denver, CO: Bernajean Porter.
- Professor papert discusses one laptop per child project*, (2006).
- Smith, F. (1995). *Between hope and havoc: Essays into human learning and education*. Portsmouth, N.H.: Heinemann.

Stager, G. (2002, June 18, 2002). *Papertian constructionism and at-risk learners*. Paper presented at the 2002 National Educational Computing Conference, San Antonio, Texas.

Stager, G. S. (2007). *An investigation of constructionism at the maine youth center*. The University of Melbourne, Melbourne.

Taylor, R. (1980). *The computer in the school: Tutor, tool, tutee*. NY: Teacher's College Press.

Turkle, S., & Papert, S. (1991). Epistemological pluralism and the revaluation of the concrete. In I. Harel & S. Papert (Eds.), *Constructionism* (pp. 161-191). Norwood, NJ: Ablex Publishing Corporation.

Synchronising processes in Imagine Logo: Why and How

Peter Tomcsányi, tomcsanyi@fmph.uniba.sk

Dept of Informatics Education, Comenius University, Bratislava, Slovakia

Abstract

Several modern Logo implementations allow running concurrent processes. Imagine Logo is one of them (others are, for example, MicroWorlds EX and Terrapin Logo 2.3).

This feature is implemented not (only) because it is modern and recent operating systems allow its efficient implementation, but because it makes programming of many simple projects much easier.

On the other hand, from the theory of concurrent processes we know, that if we start using concurrent processes which share data, some specific challenges will inevitably emerge solving of which is easily possible only by introducing some means of interprocess synchronisation.

The article introduces the reader to processes in Imagine Logo. It shows a simple Logo project demonstrating the emerging challenges. The example is discussed to its details and the way how to solve it will be shown.

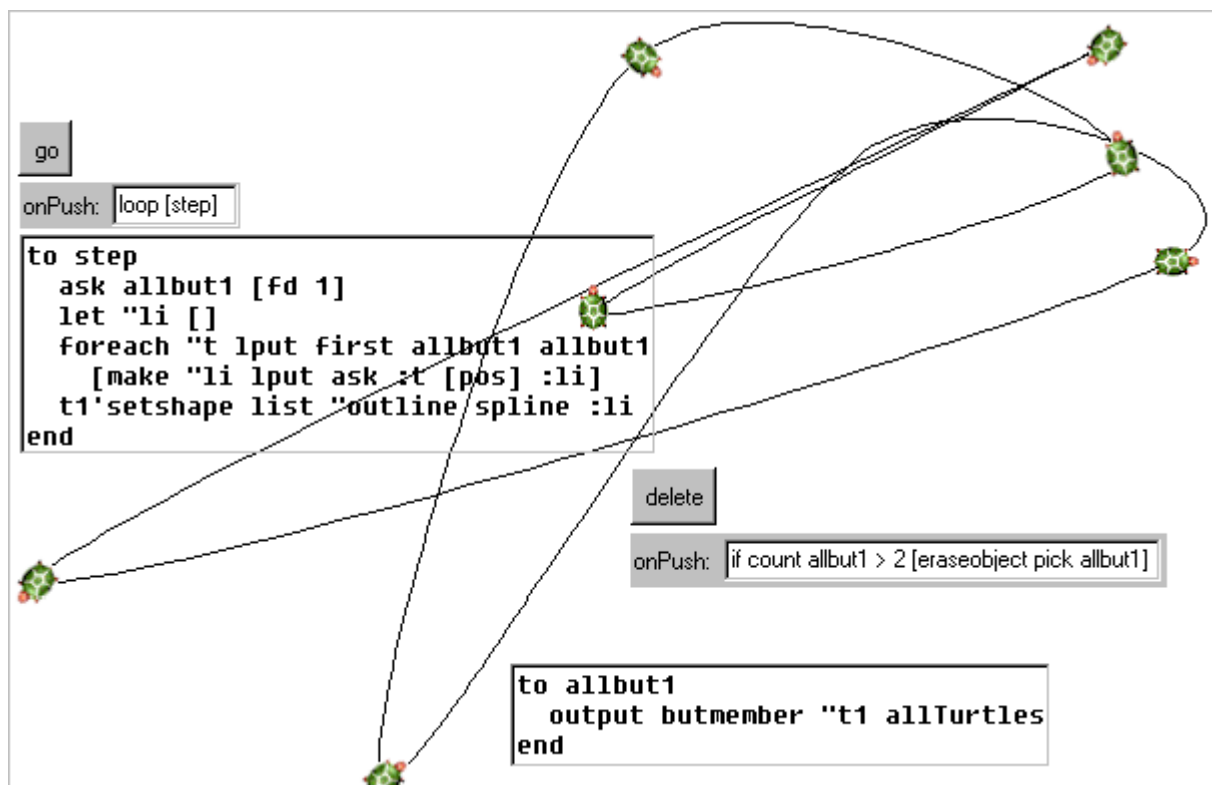


Figure 1. Our example project

Keywords

Imagine Logo, concurrent processes, race condition

Introduction

Several modern Logo Implementations allow running concurrent processes. Imagine Logo is one of them (others are, for example, MicroWorlds EX and Terrapin Logo 2.3).

This feature is implemented not (only) because it is modern and recent operating systems allow its efficient implementation, but because it makes programming of many simple projects much easier. Especially multimedia Logo projects and projects with user interaction can benefit from concurrent execution of processes. Thanks to processes it is easy to make things happen in the same time without complicated concepts like simulation calendars or timers.

On the other hand, from the theory of concurrent processes we know, that if we start using concurrent processes which share data, some specific challenges will inevitably emerge. Computer scientists name them race condition and deadlock.

Are these challenges only hypothetical or they are real? Can we make a simple Logo program which can demonstrate them? It seems that not only many Logo users but also Logo designers do not think that Logo is immune to such problems. And maybe also that the solutions would be too technical and not belonging into the Logo language.

In this article we would like to dispute such opinions by showing an example of simple Logo program, which can exhibit a race condition. The problem can be easily explained to anybody with intermediate level of understanding processes in Logo. And the remedy is not at all that complicated and technical if the used Logo dialect contains well-designed means for it.

Processes in Imagine Logo

A process is a running program. Whenever a computer starts to execute some code independently of other code it starts a new process.

In this section we will give a brief overview of processes in Imagine Logo so that the reader can better follow the example programs, which we will show later in this article.

In Imagine Logo there exists always one process, it is called the Commander process. It reads the text typed into the command line and then executes it. While it is executing we cannot type another line.

A new process can be started in two ways:

- Programmatically by using the primitives `launch`, `forever`, `every` and `after`.
- As a reaction to some events.

Starting a process programmatically

It means that at least one process must be already running and as the result of execution of the mentioned commands another process is created and starts running in parallel to the original one.

If we type into the command line:

```
loop [fd 1 rt 1]
```

Then the turtle will start moving in a circle and the command line will be occupied by executing the commands so we cannot type anything else.

If we type

```
forever [fd 1 rt 1]
```

The effect on the turtle will be the same, but the commands were launched in another process and the command line becomes usable again.

Reacting to events

A mouse click to a turtle or pushing a button on the screen are examples of events, which can launch new processes. These run in parallel to other already running ones.

Many objects define several events, which may happen to them and the programmer can attach some code, which starts running (i.e. becomes a process) when that event happens. We will name them event processes.

To avoid firing many processes by repeating the event (e.g. a mouse click) before the previously launched event process finished, there are special rules for most kinds of event processes: a new event process can be started only if no event process for the same object and the same event is running.

Processes run simultaneously in a random way

It is quite clear that if we have one processor and more processes then they must not run really in parallel, but they must share the processor in such a way that a small piece of each process is executed first, then a small piece of next process is executed etc. So from the user's point of view they seem really running in parallel. The planning i. e. determining when which process is running is in Imagine Logo left to the operating system. This planning depends on many factors and we should not expect any regular way of executing.

For example, we launch two processes. One will forever turn the turtle right by one degree and another will forever move it forward by one step:

```
forever [rt 1]
forever [fd 1]
```

We may expect that these two processes will run in turn. First process performs `fd 1` second `rt 1` then again `fd 1` etc. So the turtle should move on a circle. This would be the most regular way to execute these two processes. But it is not the case. This is, what happens in the reality:



Figure 2. Two concurrent processes

It is nearly a circle. But not exactly. It is because many other things happen in the operating system and must be planned in-between these two processes. So sometimes it happens that one of the processes is allowed to go for several steps until the second one is allowed to go. Therefore the result is not an exact circle.

This simple example can serve as a test how a Logo implementation implements processes. The test can show whether it tries to implement them entirely by its own or leaves the implementation on the operating system. When running this test in MicroWorlds EX or Terrapin Logo we get a perfect circle, which indicates that these Logo implementations probably handle processes themselves and loops in two running processes strictly alternate. Although this seems to be more regular for the user, we prefer the way how processes are implemented in Imagine

Logo because it allows the user to learn something deeper about parallel processes in real operating systems.

Stopping processes

A process can stop by itself. Processes launched by the `launch` or `after` commands as well as all kinds of event processes stop when their execution reaches the end of their code. On the other hand, processes launched by `forever` and `every` commands execute in an endless loop until stopped explicitly. Any kind of process can be stopped if it executes the `stopMe` command. All processes can be stopped by the `stopAll` command.

Any process can stop another process (or itself) using the `cancel` command. To use it we must know the name of the process to stop. Normally the process name is equal to the list of instructions, which it carries out. For longer lists it may be not convenient and therefore the programmer can add an optional input to the commands that starts processes to define the process' name, which can be used for subsequent `cancel` calls.

Event processes get special names composed of the special character `@`, the name of the object and the name of the event. For example, a process started by pushing button `b2` will be named `@b2onPush`.

Note that stopping the Commander process means something different than stopping any other process. Stopping the Commander process means that the currently executed input line of code stops executing but the Commander process does not disappear, it will prompt us for typing another line. If we stop any other process then it completely disappears.

Listing processes and waiting for processes

The primitive `allProcesses` outputs a list of names of all existing processes.

The primitive `done?` outputs `false` if the process, which name is given to the command, exists. Otherwise it outputs `true`. `Done?` in combination with the `waitUntil` primitive allows us to wait until a process finishes.

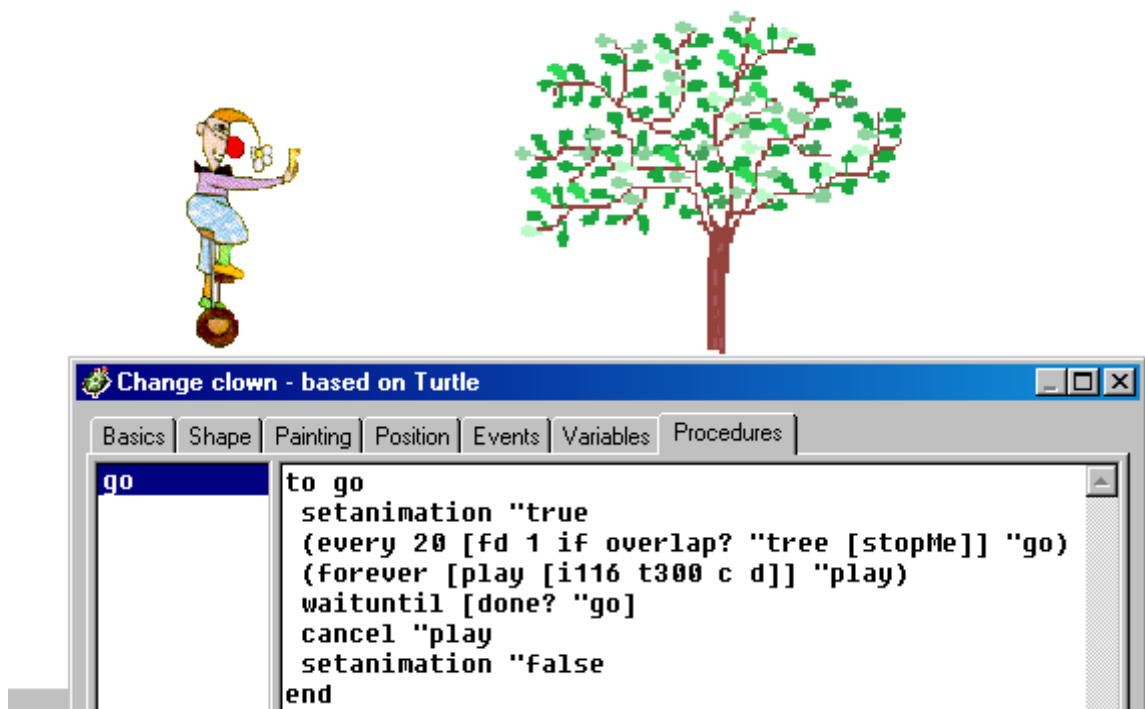


Figure 3. The clown is animated, he is moving and playing

The procedure `go` of the clown starts animating him (animating is made by Imagine Logo itself without the need of running any process), then starts moving (the process named `"go`) until hits the tree and in the same time some sound effects are started. Then the program just waits until the process `"go` stops and then it stops the playing process and also stops animation.

The Animated Curve project

We have seen that processes are powerful and easy to use. And there seem to be no problems. So let's try another example.

We will try to make an animation, which is used commonly in screen savers. We will generate an animated curve, which changes its shape and bounces around the screen.

Start with a new project. The goal is to make some turtles on the screen with random headings, which will bounce off the page boundaries. Therefore we define an `onClick` event for Page1:

```
new "turtle [pos ( mousepos ) heading any rangestyle bounce pen pul]
```

Now click three or four times to the page to create some new turtles.

Turtle `t1` will have a special task - it will change its shape to a smooth curve connecting the positions of all the other turtles. We will need several times to instruct all the turtles except `t1` to do something so we define a function to get the names of all turtles except `t1`:

```
to allbut1
  output butmember "t1 allTurtles
end
```

Now let's add a button named `go` and program its `onPush` event to:

```
loop [step]
```

and define its procedure `step`:

```
to step
  ask allbut1 [fd 1]
  let "li []
  foreach "t lput first allbut1 allbut1
    [make "li lput ask :t [pos] :li]
  t1'setshape list "outline spline :li
end
```

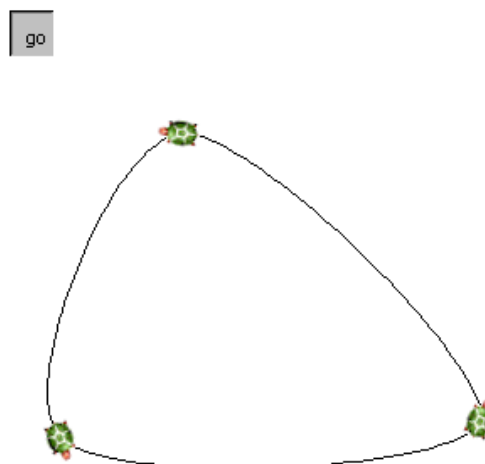


Figure 4. A curve defined by three turtles

Now when we push the button, it will run its own process. The process at first moves all turtles a bit and then it will make t1's shape to be a smooth curve going through the positions of the three other turtles.

As the process is programmed for any number of turtles we can still click into the page. It will create another turtle and it will automatically participate in the curve.

After creating some more turtles by clicking the page we can write this to the command line:

```
ask allbut1 [ht]
```

and we will see only the bouncing curve.

Our next idea is to program another button, which effect will be deleting a randomly chosen turtle. Not to delete all of them we will at first check whether there are more than 2 moving turtles. So we create a button, name it "delete" and define its `onPush` event as follows:

```
if count allbut1 > 2 [eraseobject pick allbut1]
```

If we try the button then a few times it may work correctly, but then we will see an error like this:

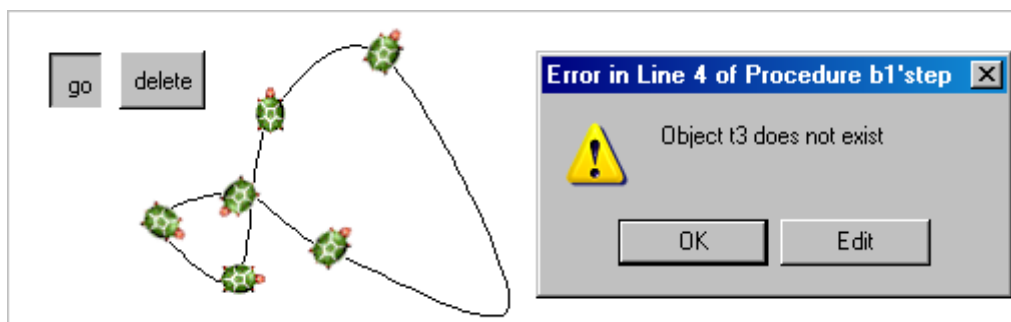


Figure 5. The error message after trying to delete a turtle while all the turtles are moving

Another strange thing may happen: after deleting a turtle the movement stops and the `go` button is released.

What did happen? If we look into the code then we can see that the third and fourth line of procedure `step` is a loop going through a list of turtle names. The list is constructed by calling `allbut1` and then inserting the first item of that list again at its end. So if we have turtles `t1`, `t2`, `t3` and `t4` then the result of `lput first allbut1 allbut1` will be `[t2 t3 t4 t2]`. So the loop will start with variable `t` having at first the value `"t2`, then `"t3` then `"t4` and then `"t2`. Inside the loop we use the value of variable `t` as an input to the function `ask` to get the position of the turtle whose name is stored in `t`.

If we manage to push the button at the moment when this loop is being executed and we delete a turtle, which has not yet been examined in the loop, then later the code in the loop's body will want to read its position. And the turtle will not exist. So we get an "object does not exist" error. That's it. The presence or non-presence of the error depends on when we delete the turtle i.e. when we push the "delete" button.

After all we understand what has happened. But what can we do?

We found the part of the code, which is responsible for the bad behaviour. If we were able to achieve that the process of deleting does not run while the process of moving is executing the loop in third and fourth line of procedure `step` then we would be able to solve the problem. But how to achieve it?

The solution would be not so easy without specific support for such situations, which is built into Imagine Logo. There is a primitive named `runCritical`. It has two inputs. The first input is a

word and the second input is a list of instructions. It runs its second input in such a way that no two `runCritical` commands with the same value of their first inputs can be run in the same time. It means that whatever is written inside the second input to `runCritical`, it will be protected against being interrupted by any other process running also a `runCritical` command with the same value of its first input.

We can now use the new command. We must insert it into two places. The first one is in the step procedure:

```
to step
  ask allbut1 [fd 1]
  let "li []
  runCritical 1 [
    foreach "t lput first allbut1 allbut1
      [make "li lput ask :t [pos] :li]
  ]
  t1'setshape list "outline spline :li
end
```

And the second place is in the `onPush` event of the delete button:

```
if count allbut1 > 2 [runCritical 1 [eraseobject pick allbut1]]
```

Now we can try again and we will see that we will never get the error message "object does not exist". But, if we try for a longer time (and it depends a bit on the speed of the computer) we can sometimes get another strange behaviour: after deletion of a turtle all the movement suddenly stops. Is this another mystery? No, it is the same kind of problem. Procedure `step` has another problematic place. It's its first line. Here we ask all turtles to move forward. Here it may happen that a turtle, which is currently executing `fd 1` is erased. And one of the rules in Imagine Logo says that if an object is being erased and any running process is being currently executed for that object, then that process must be stopped. Therefore the process of the `go` button stops.

The remedy is the same, the first line of the procedure `step` must be also protected by a `runCritical` command:

```
to step
  runCritical 1 [ask allbut1 [fd 1]]
  let "li []
  runCritical 1 [
    foreach "t lput first allbut1 allbut1
      [make "li lput ask :t [pos] :li]
  ]
  t1'setshape list "outline spline :li
end
```

Now our small project is complete and we should never get any errors regardless of how quickly we are trying to delete the turtles while the project is running.

A bit more theory

As we already mentioned in the introduction, the situations shown in the previous section, are well-known to computer scientists and also to professional programmers - see, for example, Tanenbaum (2001).

So depending on the age and level of programming knowledge of our pupils or students, we can either stop at the level of details presented in the previous section, or we can go on and use the opportunity to tell them more from the theory of concurrent processes.

Computer scientists have studied such problems at the times when first operating systems were able to run several concurrent processes and programmers started to get into such problems. Since then our problem has a name. It's called **race condition**.

If the result of work of several processes depends on the exact order of execution of these processes by the operating system, we name it **race condition**. It means that running the processes for the same data we can many times get good results, but then suddenly we get a strange result or an error or a crash. So race conditions cause hard to debug bugs in our programs.

And exactly this happened in our previous example. We can push several times the delete button (which launches an event process running in parallel with the main process of `go` button) and a turtle is just deleted with no error. But then suddenly for another click on the button we get an error.

The reason of the race condition is that in our code we have **critical regions** - lines of our code that manipulate data, which can be shared by other running processes at the same time. We can eliminate the race condition if we identify all critical regions in our program and achieve that no two processes will be inside their critical regions at the same time. Such execution of critical regions is named **mutual exclusion**. Both operating systems and programming languages use to give some tools for achieving mutual exclusion. The tool found in Imagine Logo is the `runCritical` command, which can frame our critical sections and will arrange mutual exclusion of them. Some other Logo implementations, even if they include the ability of running several processes, do not contain such tools.

Conclusion

We have shown one simple example of Logo program, where race condition happens. It demonstrates that in Logo implementations, which add concurrent processes some kind of synchronisation tools are needed as well. Imagine Logo provides synchronisation mechanisms for critical sections.

Showing such examples to intermediate and advanced learners can, in our opinion, enhance their understanding of processes, which can lead to both better using them in projects and getting some knowledge, which may be used also outside the Logo world.

The author of this article uses this example when teaching concepts of operating systems to future teacher in pre-service courses. These students already passed an intensive course of Imagine, several of them are advanced Imagine programmers and this example can well connect their previous knowledge with the new concepts found in the operating systems course.

References

- Blaho, A. and Kalas, I. (2001) *Object Metaphor Helps Create Simple Logo Projects*. In Proceedings of EuroLogo 2001. Edited by G. Futschek. Linz, August. pp. 55 – 66.
- Tanenbaum, A. S. (2001) *Modern Operating Systems (2nd Edition)*, Prentice Hall, New Jersey

Transforming activities from workbooks for preschoolers into computer games

Monika Tomcsányiová, tomcsanyiova@fmph.uniba.sk
Dept of Informatics Education, Comenius University Bratislava, Slovakia

Abstract

Children develop rather quickly. Admittedly, their mathematical, logical and mental skills can be influenced by books, which they are exposed to. There are some interesting workbooks for young children, which train their observation, give them first ideas about counting, colours, shapes, similarities and differences. A programmer can easily see a potential to convert parts of the workbooks into computer activities. The computer can enhance the activity by animations and sounds or by the ability to move the pictures around using the mouse or the keyboard. Computer games could become parts of such workbooks. Some activities can be made both in the workbook and in the computer. For other activities it is better to do them only in the workbook in order not to forget also about skills to paint using a real pencil or a crayon or to write and draw lines.

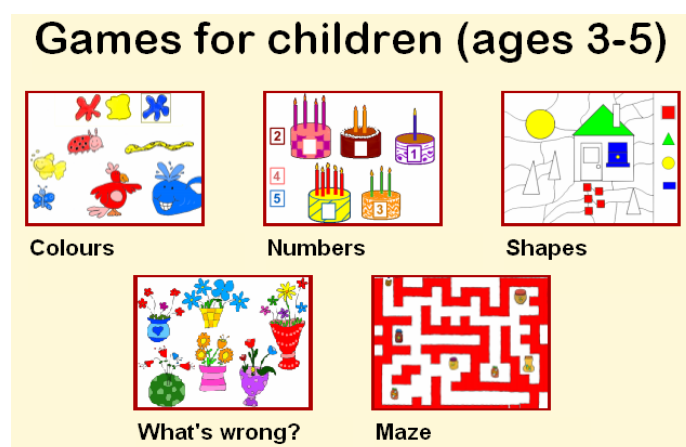


Figure 1. Choose the game

In our article we will describe some interesting programs for 3 to 5 years old children, which we made by transforming them from workbook activities.

From the very beginning our 4 years old son Thomas helped us. It seems that he understood how the computer can enhance the activities. He has seen how static activities from the workbook started suddenly moving and he was able to design colours of objects, give his voice to them or modify them. He was tirelessly colouring the pictures, he learned how to select them and move them to their proper place. Surprisingly, later when we worked with other activities in paper workbooks he was thinking how to transform them into computer activities, he even came with suggestions how the computer could check whether the child solved the assignments correctly.

Keywords

math, kids, computer games for kids, critical thinking

Motivation

3 to 5 years old children are given by the kind of curiosity and learning ability, which older children lack. We must certainly support these attributes by various types of activities. The most traditional way is to use workbooks, which allow the children developing their skills of using pencils and crayons. Such workbooks are often made by reputable authors and their applicability is tested by psychologists. The pictures are drawn by children's illustrators, so these workbooks are often not only suitable for children but also very appealing to them.



Figure 2. Workbooks for children
taken from <http://matthewsmediallc.com/items/books-4-kids/list.htm?1=1>

We have several such workbooks at home. Our son especially likes the Gold Stars series of workbooks because of its colourful and beautiful pictures. It has also an original idea of sticking stars to pages, which were well done by the child.

Nowadays many households own computers. The child sees how the parents are using the computer for their work and/or leisure. So it is very natural to seek for computerised activities for children, too. There are tons of activities, which their authors claim to be suitable for preschoolers. Can we believe them? We can compare them to our favourite workbooks. Or we can try to create computer activities, which resemble the activities in our favourite workbooks.

We tried to select such activities, where the computer can help making them more interesting and appealing. In the same time we wanted to make a set of activities covering several important skills and knowledge of preschoolers - distinguishing colours and shapes, counting up to 5, spotting differences and similarities, recognising the otherness. From the computer skills we focused especially to improving the coordination of eyes and hands when working with the keyboard.

Preparing the pictures

My son Thomas is 4 years old. He is already able to control the mouse, drag pictures around the screen, using the arrows on the keyboard. He learned all this by using the computer previously. He has been interested about the computer from his age of 3, so he already knew the graphical editor. He used it for creating his own very abstract paintings and drawings or he coloured pictures that we downloaded from the Internet.

To implement our new activities we needed at first to draw some pictures. We understood that the good activities attract children also because they are colourful and their pictures of animals, people and everyday things are realistic and childish at the same time. I drew them in the graphical editor RNA (<http://www.logo.com/rna>). Our son often stood beside me and pushed me

to finish drawing. He often indicated that the recently drawn line is not correct. So I had to use the undo ability of RNA quite often to make the pictures look well.



Figure 3. Thomas at work

Thomas understood quickly that he was not able to draw the pictures precisely enough. But he could colour them. So he was working with unbelievable patience, he blended colours until he made the right one. He coloured alone nearly all flowers in the *What's wrong?* and *Colours* activities.

We started with pictures for one activity. Then I transferred them into Imagine Logo and added the program so that the activity started to work according to my and Thomas's taste.

Programming

To implement the activities we chose Imagine Logo (<http://www.logo.com/imagine>). Not only because we know it very well, but also because it offers programming means, which are especially suitable when creating these kinds of activities.

Colours

The most well-known activities from children's workbooks are probably those for teaching the child to recognise colours. Children see colours every day, but it is not so easy to name them and to tell one colour from another one. The task of the child in this activity is to select a colour blot and then click to a picture of the same colour.

After preparing the pictures of blots and animals I was able to start programming. In the first phase, when defining the classes of objects used in the game, there is no visible result for a quite long time. My son started to be impatient and urged me. He wanted to see at least something working...

I created two classes: one of them is `Blot` and it represents the blots and the other, named `myPict` represents the coloured pictures. The `onLeftDown` event of the class `Blot` changes the frame of all blots to first one (which shows the blot not selected) and then changes the frame of the actually clicked blot to the second one (selected):

```
ask allOf "Blot [setFrame 1] setFrame 2
```

We must also create a variable named `frameMode` with value `true`. Otherwise it would not be possible to set the frames manually.

The `onLeftDown` event of `myPict` class must check whether the user clicked to a picture of the correct colour. In order to make testing simple we name the yellow blot `myYellow` and all yellow pictures will have names, which are different only by adding another character `myYellow1`, `myYellow2` etc.

When we added all yellow pictures, we found out that we must think out what should happen if the child clicks to a correct picture and what should happen if the click is incorrect. My son suggested that the computer will say "Yes" if the click is correct and "No" otherwise. He suggested the English words because they have some elementary English education in the kindergarten and he has already seen games, which react like this. Although in Imagine Logo it would be possible to make the speech synthesiser say the words by a computer's voice, it is more fun if Thomas can record his own voice for this.

Recording voice is quite easy in Imagine Logo. Just push the Multimedia button on the toolbar and in its drop-down menu select Wave sound. Then in the popping-up window just push its Create new button. You will get the standard Windows Sound Recorder application and you can start recording. After several trials (some trials were too loud or too silent or some noise was recorded along with the voice) Thomas was able to record the words Yes and No into files `Yes.wav` and `No.wav`.

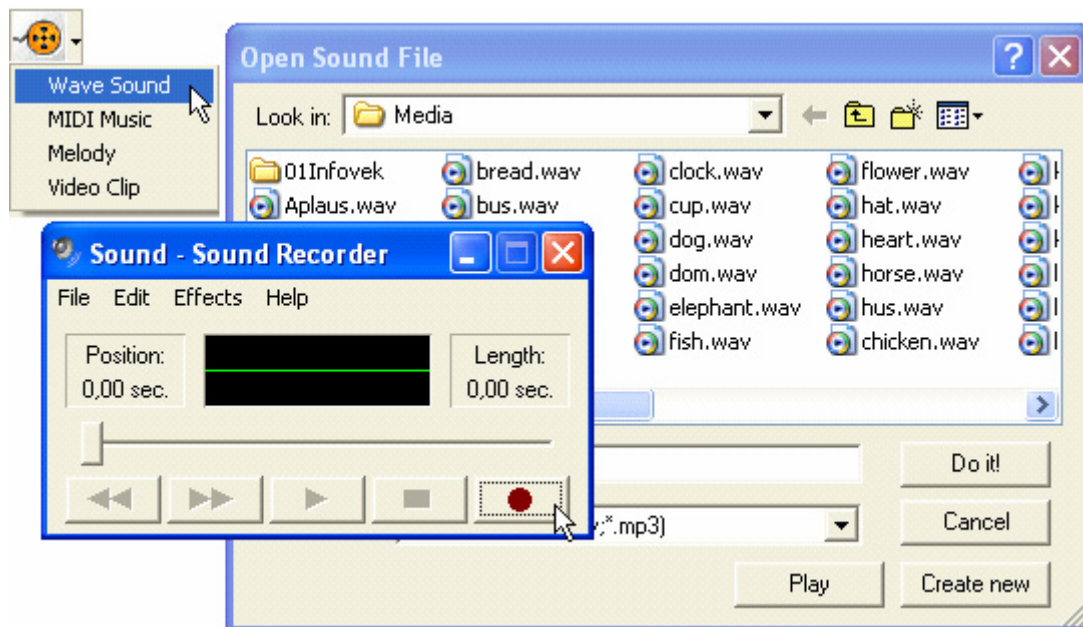


Figure 4. Record your voice

So the final version of `onLeftDown` event of the class `myPict` is:

```
ifElse (ask bl myName [frame]=2) [playW "yes] [playW "no]
ask bl myName [setFrame 1]
```


Numbers

A four years old child should be able to count up to 5 as well as know the numerals 1 to 5. We chose an activity where the task is to assign a numeral to a birthday cake, which has the same number of candles.

After creating the pictures, I started again with two classes. This time I named them `Cake` and `myNumber`. `Cake` is rather simple – it needs only define one variable `pen` with value `pu`.

The main piece of code lies in the `onLeftUp` event of `myNumber` class. Its effect is that the numeral being dragged either touches-down to the cake (if it is the correct numeral) or jumps off the cake to its home position:

```
ifElse (last last overlapped) = (last myName)
  [setPos ask overlapped [pos]] [home]
```

Note that testing is again based on the fact that the name of the cake and the name of the corresponding numeral are "similar". In this case their last character is equal (it is the numeral itself): The turtle representing the digit 1 is named `t1` and the cake with one candle is named `cake1`. The last two things to set for the class are two variables. One is `autoDrag` having value `true` and another is `pen` having value `pu`.

When the activity starts we want to rearrange the cakes so that the child could not remember each cake by its position on the screen and force it to take care even when doing the activity for several times in a row.

```
to start
  repeat 10 [
    let "mT1 pick allOf "cake
    let "mT2 pick allOf "cake
    let "p ask :mT1 [pos]
    ask :mT1 [setPos ask :mT2 [pos]]
    ask :mT2 [setPos :p]
  ]
end
```

Shapes

The task is to spot, identify and colour basic geometrical shapes (squares, circles, triangles and rectangles), which a scene consists of. Each type of shape should be coloured to the colour that is prescribed for them by one such object lying outside the scene at its border.

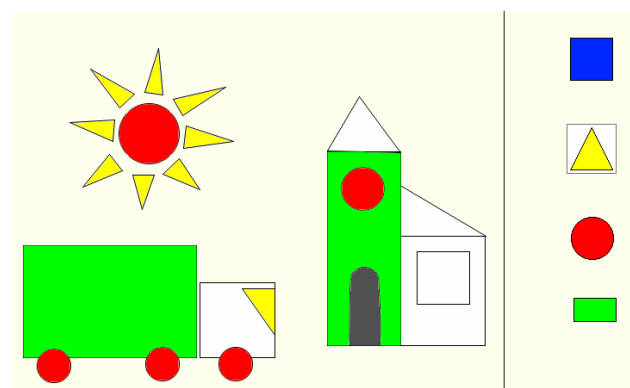


Figure 5. Can you spot another triangle?

This activity is in its structure nearly identical with the activity *Colours*. The objects on the border of the picture should behave in the same way as blots do in *Colours*. It means that they must

select themselves when clicked and deselect all other objects of the same kind. The picture consists of objects of the other kind. When they are clicked they must play a sound, then check if they were clicked correctly (i.e. they are of the same shape as the clicked prototype. If so then change their shape to a coloured one.

What's wrong?

It is a well-known puzzle to find a picture which does not belong to a group with other pictures. It can be as simple as an activity for preschoolers or as complex as an adult IQ test.

When creating the activity for such young children, the most important point is to find good pictures. They must show things, which are well-known to children and they can understand after a while which one does not fit into the group without lengthy explanations. Our picture shows several vases with flowers. Each vase contains one flower, which does not belong to the others in the same vase.

This is a kind of one purpose activity because it is not possible to mix the positions of flowers before the activity starts - the size and shape of vases and size and shape of flowers' leaves does not allow it. So we have only a background picture and one class of turtles named **Wrong**. Turtles of this class (shaped as flowers) are placed to all the places where the background picture contains a flower not belonging to that vase. The class **Wrong** has several variables: **shape**, **frameMode** and **transparentClick** (both set to **true**). It has also an event **onLeftDown**, which only changes the frame of the turtle to indicate that you spot correctly the wrong flower.

So the last point to finish the program is to place objects of class **Wrong** to the places where they belong to.

This project can be easily adopted for other kinds of objects. You just need to replace the background picture and to place instances of the class **Wrong** to correct places.

Maze

All the activities up to now contain one fixed assignment (even if we can mix things a bit as you have seen in the *Numbers* activity). The end-user cannot create its own versions of the assignment without some knowledge of Imagine Logo and an ability to draw new pictures.

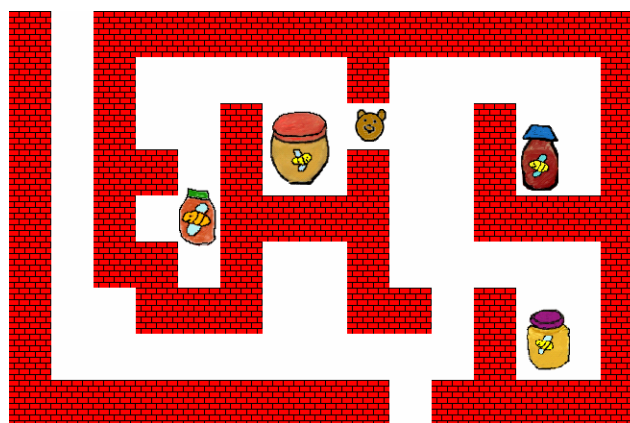


Figure 6. The Maze microworld

This activity is more flexible. It allows not only solving the task (collect all items in the maze and find the exit) but it also allows the child to create its own assignments (a maze with objects to collect positioned into it).

On the first page of this project there is a prepared maze. The player can move a bear inside the maze by using the arrow keys on the keyboard. The task is to collect all bottles with honey and then exit the maze.

When our son played the game he suggested that the computer should say something when the bear finds a bottle of honey. He remembered that recording had been very easy so he started to search for the correct button...

Later Thomas came with another suggestion: It should be not possible to exit the maze until the bear collected all the bottles. He suggested that the exit should be closed by a door and open only after collecting the last bottle. Another improvement was to forbid the bear to leave the maze by the same door by which he entered the maze.

The second page of this project contains a simple microworld, which allows the child to build a maze around the bear.

Clicking to an empty place you build a wall there. Clicking on a piece of wall it disappears. This way you can build a simple two dimensional maze. At the top of the page there are the bottles with honey. They can be dragged to some places in the maze.

The page has an event `onLeftDown`. It creates a new instance of the class `Wall`. It is important to position the walls to positions divisible by the sizes of the wall's picture:

```
new "Wall
[ pos ([22 24] + list
  44 * ( div first mousePos 44 )
  48 * ( div last mousePos 48 ))]
if ask lastName [overlapped] <> [] [eraseObject lastName]
```

Also the class has its event. It just deletes the piece of wall which has been clicked on:

```
eraseObject myName
```

The bottles of honey are simple turtles having corresponding shapes, their pen is up and their `autoDrag` feature is switched on.

Conclusion

When creating individual programs we tried to implement activities from colourful workbooks for preschoolers of age 3 to 5 in the simplest possible way. The *Maze* microworld has been created as a specifically computer activity which should help the child to orientate himself on the two-dimensional surface of a computer screen.

We have also shown that Imagine Logo is a highly suitable tool, which allows to implement several activities with identical or just slightly different code. It enables a parent (who can program in Imagine Logo in some extent) to create several interesting activities for his/her child with very little effort. Each activity can be changed by changing only the pictures.

The child took part in the process of designing the programs. And it is very important to mention that the child was able to work differently with the program than with the same kind of activity in the workbook. He was able to prepare and colour the pictures, record sounds, suggest effects indicating good or wrong solution of the tasks.

The computer allowed us to make the activities more vivid by adding movements, animations and sounds. Imagine Logo has proved to be nice towards such kinds of activities as well as towards parents and children, who want to spend interesting and useful moments together next to the computer.

There are many similar activity books on the market. A parent, who is able to program, can find plenty of activities, which are destined to be re-implemented into the computer in the way, which we outlined in this paper. The computer can increase their attractiveness by including more variability, interactivity, animation and sounds.

References

- Patilla, P. (2004) *Hviezda v matematike pre deti od 3 do 4 rokov*, Slovart, Bratislava, ISBN 80-7145-864-3
- Patilla, P. (2002) *Gold Stars Math Ages 3-4*, Paragon, Bath, ISBN 0752594664
- Tomcsányiová, M. (1999) *Logo programs for our 3 years old daughter Janka*, Eurologo99: Proceedings of the 7th European Logo Conference Eurologo'99, Sofia 1999, pp. 350-355, ISBN 954-9582-03-5.
- Kalas, I. and Blaho, A. (2001) *Object metaphor helps create simple Logo projects*, Eurologo2001: Proceeding of the 8th European Conference. Wien Österreichische Computer Gesellschaft, 2001, pp. 55-65, ISBN 3-85403-156-4
- Van den Daele, Ch. (2002) *Čáry-máry žlté čiary*, Editions Caramel, Bratislava 2002, ISBN 80-85327-96-1
- Macková, S. (2003) *Tak i tak*, časopis Silentium, Bratislava 2003
- Kauka, R. (2005) *Macko Pusik*, časopis pre deti, Magnet-press Slovakia, s.r.o., Bratislava
- Stoppard, M. (1992) *Test your Child*, Dorling Kindresley Limited, London 1991, ISBN 80-85186-46-2
- Nováková, M. (1999) *Škôlkárova zima*, Nomi s.r.o. Košice 1999, ISBN 80-88958-17-2
- Gibson, R. (1997) *I can draw animals*, Usborne Publishing Ltd. London, ISBN 80-89083-12-9
- Tomcsányiová, M. (2001) *Niekoľko aktivít v prostredí Imagine*, Didinfo 2001, pp. 171-175, ISBN 80-8041-02364-8
- Tomcsányiová, M. (2001) *Imagine – nová hračka pre rodičov s deťmi*, Poškole 2001, Praha 2001, pp. 171-175, ISBN 80-01-02364-8

Creating games in Imagine for pre school

Adriana Sobreira Torres, adriana@escolaparque.g12.br
Escola Parque, Rio de Janeiro, RJ, Brazil

Abstract

This paper reports an experience developed at Escola Parque, a fundamental school at Rio de Janeiro, Brazil. It is a private school, with 1600 students from 1 ½ to 18 years old. It's a constructivist school that values student's activity, cognitive development and cooperation.

In February 2006, pre school moved into a new building, and it was no more necessary to share computer labs and other resources with older children. We started to plan the Educational Informatics to age group 2-6, and we wished computers to be completely integrated to the routine, the pedagogical characteristics and the projects developed at school. Our research on available educational software brought many problems and unsatisfactory results, as we did not want to use tutorials and closed software, neither show lots of stereotyped images to our children. We realised that most of the games and activities would not fit our aims.

Thus we began programming our own activities in Imagine. And while more games and activities were being prepared, the whole group of teachers began to take part in this process. We naturally adopted (even without thinking of them) some guidelines and patterns that are present in the daily work on the games and activities: Do not use adult drawing for "motivation"; Give the children the opportunity to manipulate object, build and test hypothesis, without pointing out every mistake; Naturally mix concrete manipulation with many kinds of representation; Use the children drawings and writings to create activities to other children.



Figure 1. Children's drawing in school decoration and in Imagine activity

Now, one and a half year after this process began, we realise that the possibility of building our own games created a new conception in the group, and informatics has become a powerful tool for teachers.

Keywords

Logo at pre-school, Logo programming, In-service teacher training

1. Our problem – choosing computer activities to Pre School

Escola Parque is a private school, with 1600 students from 1 ½ to 18 years old, in Rio de Janeiro, Brazil. It's a constructivist school that values student's activity, cognitive development and cooperation.

Until 2006 the school had a single building, and small children shared the computer lab with older children. Children at the age group from 4 to 6 had one informatics lesson a week, with the classroom teacher and an informatics teacher. Younger children had eventual access to computers. In February 2006, the pre-school moved into a new building, specially planned for small children. By that time, I was invited to plan educational informatics to this age group. My task was to create a completely new conception to ICTs: plan how many computers we should have, where they should be installed, select the programs we would install, define the routine for teachers and students to use informatics resources.

We have taken some important decisions: we wanted to use computers in everyday work and to spread the equipment all over the school premises. We wanted computers to be part of the routine, as color pencils, books, paper and posters. We wished computers to be completely integrated to projects developed at school, many of them are focused on literature books we use to adopt from the earlier ages.



Figure 1: Computers in the pre-school library

When we began the research of educational software, we realized it would be very difficult to reach our targets. There is a lot of educational software being sold in Brazil, many of them to age group 5-6, and a few to age 2-4. But a very poor amount would be useful for our school. Many of them only allow children to give correct answers, there is no opportunity to free manipulation, and children mistakes or hypothesis are discouraged by "sorry, try again" messages. Many software use stereotyped images. It's hard to find beautiful software, with interesting information, and intelligent tasks. And, of course, they were not designed to our projects. A few software would fit our aims, but they are, in general, very expensive and can be used to a poor range of activities.

Thus we began to create our own activities, programming in Imagine. We began building some simple games for children to use, while we continued looking for Internet games and free download software.

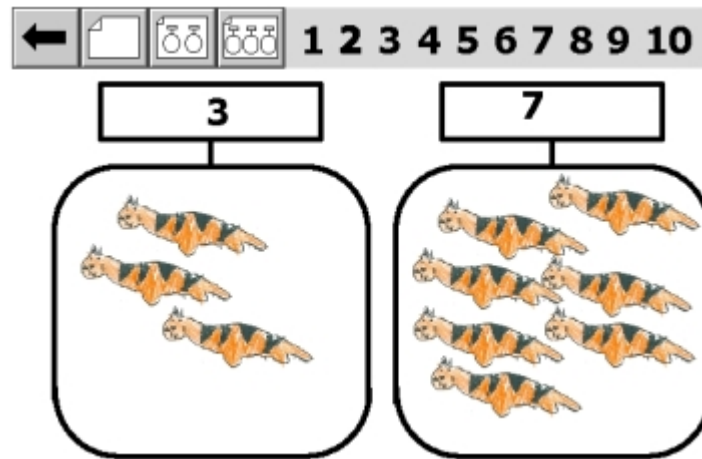


Figure 2: Imagine activity “10 tigers”, based on a literature book, transferred from paper to computer, using a child’s drawing.

2. Parameters to consider

Our experience in informatics to pre school education was very limited. But we had already made many decisions, adopted guidelines, made theoretical choices and developed many good experiences both about informatics and pre school methodology, that should be considered while planning informatics to this age group.

2.1 Our pre school methodology and its influence in ICT decisions

We wanted to use computers, digital cameras, scanners and data shows integrated to pre-school routine. Some important parameters in our approach to pre pre-school education have deep influence in our software choices.

We work with small children, in the pre operational stage. The language is being quickly developed, as other representational skills. Working with representation is one of the most important targets to pre-school. Children need to manipulate concrete material and to live physical situations. After these experiences, we suggest children to represent situations and objects in many different languages, using symbolic games, make-believe playing, drawings and other graphic representation, developing oral language and introducing written language. Computers should be a tool for children to represent objects, actions and situations, keeping a deep relation with the real activities and concrete material used.

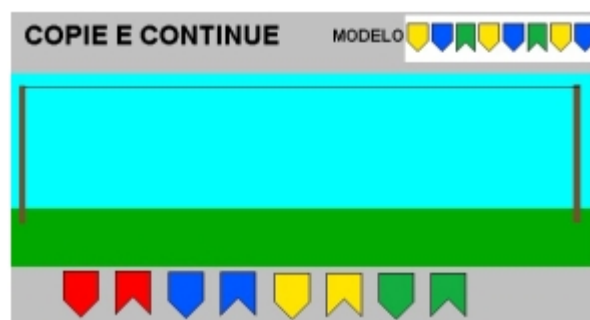


Figure 3: Imagine activity “Order the Flags”. Re-creates series of colors and shapes, after decorating the school for a traditional Brazilian party.

The computer is a concrete object to manipulate as well as a tool to represent, and can be useful in different stages in representational process.



Figure 4: Imagine game “Dice running”. While some children play with real dices, others play on computer.

We avoid using adult drawing or any other kind of “external motivation” for children. We believe that the tasks, the challenges and the goals of activities are, by themselves, enough motivation. For instance, we do not use known characters of pretty animals to create a nice impression on work sheets. They are not necessary, and generally show stereotyped and non-significant images. Teacher images in daily work are very functional, not decorative. To decorate our school we use children drawing.



Figure 5: Children drawing in school decoration and on a classroom wall.

Other adult drawings and images offered to children are not models to be copied, but beautiful things to be admired, and they should have nice artistic or expressive quality. The quality of illustration is an important choice criterion to literature books.

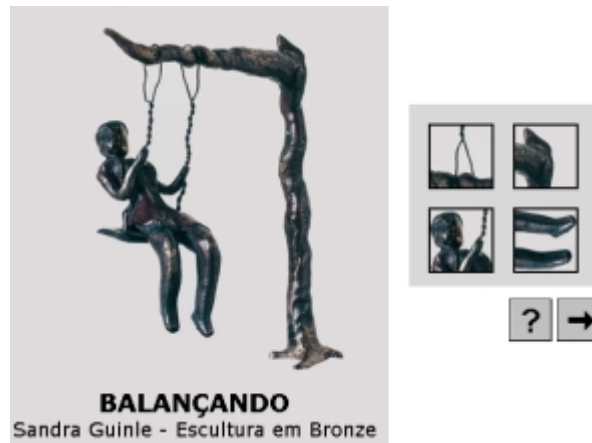


Figure 5: Imagine game “Sandra Guinle”. Puzzle with images of sculptures made by a Brazilian artist, showing many traditional games and playing.

We do not use tutorials or didactic books, because we want to let children manipulate objects, build and test hypothesis, without pointing every mistake. The school program is based upon researches on cognitive development, focused on mathematics, written and verbal language and science thinking.

2.2 Using computers at our school

We use computers at school since 1989. All our plans on ICT are based on curricular integration and cognitive processes. We think about computers in school as powerful tools, to any level or age group.

Tools to **learn** – with a computer students can get information about many subjects. But getting information does not mean significant knowledge. Giving lots of information to passive students is meaningless. On the other hand, computers are very interactive, and while taking their own decisions, students can be really active learners.

Tools to **create** – computers are incredibly versatile and malleable. It is a tool that gathers many tools, using different languages: text, sounds, animations, photographs, graphs and its combinations. Computers allow children to express themselves, gathering knowledge to creativity in authoring process.

Tools to **communicate** and **collaborate** – in small groups or long distances, quality, speed and diversity of communication are increased by using computers. Collaborating in different work teams and playing different roles is very important for us.

Tools to **think** – children are very active while interacting to computers. They can always change, increase, and review what they have done. They can try, test hypothesis, and evaluate “computers reaction” to their actions. It’s a powerful tool to be aware of their own thinking.

This approach is implemented in our work with older children and we wished to apply this in pre school education.

3. Our solution: creating our own activities

We haven’t made any plan or established a strategy to solve our problem, and we continued looking for good ideas, activities and software. I had no experience with this age group, so I began observing the pre school routine. Then I realized many activities could be represented on computer, by creating in Imagine games analogous to concrete activities.

The classroom teachers had no experience with educational informatics. Some of them are frequent users of computers, others are eventual users. But they started to observe children while they played with the games. They quickly had many suggestions to improvements and

variations to the games. Some teachers realized they could ask me to program something they wanted, and began to ask questions such as “Is it possible to do something about fairytales?” or “What can I use for my children to free writing, instead of Microsoft Word?”

Step by step, many teachers and the pedagogical staff began thinking of new possibilities, specific versions to different ages, in re-creating usual paper activities to computer, and analyzing each activity to decide whether we should or not give feedback to children.

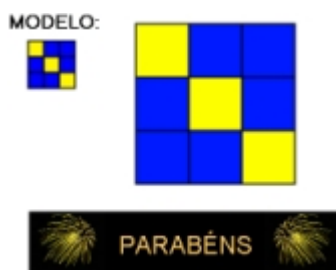


Figure 6: Imagine game “Nine squares”, suggested by 3-4 years old teacher. Previous version was difficult for children (needed dragging with mouse) and gave no feed-back.

A simple incident stimulated this process: A teacher had promised her class she would print a book with their drawings, as a gift for children day. Each child draw a person, divided into 3 parts. The parts could be combined creating fun characters. But this teacher has exceeded her printing quota. She was thinking on how to solve this problem without disappointing her class. Our solution was to create a computer game, published on our web page, for children to play at home. Parents and children loved the “gift”, and suddenly everybody realized we could use programming as a new solution to many problems, demands and creative ideas in school.

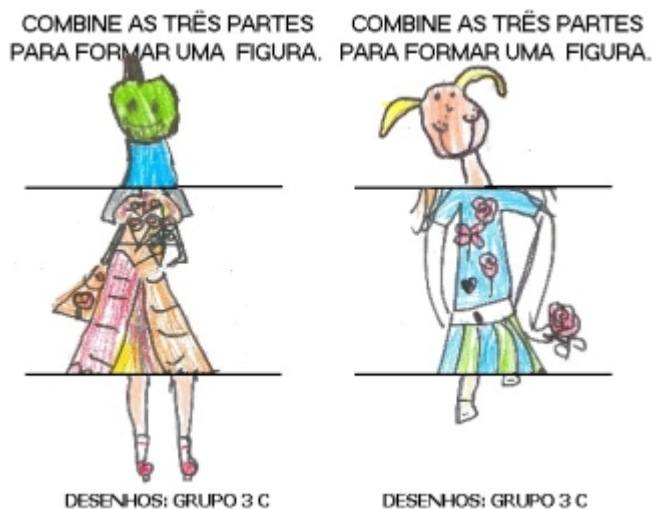


Figure 7: Imagine game “Surprise Gift”, to combine children drawing divided into 3 parts.

Now we are working in each specific teacher requests, such as “I want a simple program to work with tables and graphs, because we use to vote for many decisions on our classroom”, or “Can you build a matching game for children to learn the difference between R and RR (in Portuguese, they make different sounds).



Figure 8: Imagine activity “Graphs”, to create tables and graphs.

Other informatics resources, as well as imagine games, were adapted to school routine. For instance, the whole class plans a collective text, while teacher writes on the black board. Now we use a data show to this activity. Digital cameras are used to insert images to many work sheets, for instance, the registration of an excursion outside school.

The teachers’ team and pedagogical staff now think of computers as tools for their work, which can be used to reach very specific goals, in a very personal way.

Conclusion

Having a simple and accessible programming tool has allowed us to build a lot of activities, and during this process many school actors (classroom teachers, informatics teachers, pedagogic coordination, and, in a different way, children) were involved, because we wished to fully express our pedagogy in the way we use computers, and all these actors have their own contribution to this process.

I had never worked with such small children before, and I made many mistakes. In the beginning, I had a very intuitive feeling of what should be done. On the other hand, teachers had a deep knowledge about children and their development, but very poor experience in using computers, and could never imagine that they could make such a large number of decisions.

The collaboration among this big team has generated many important results:

1. About 50 games and activities being used at school, integrated to the curriculum.
2. Conscientious decisions and explicit statements about aims, aesthetic patterns and pedagogical features we wished to adopt.
3. A new conception about informatics at school, and new practices in routine.
4. A plan to the following years. Today we know what we want to do with our pre school children using the computers, and there are still lots of things to do.

References

- Valente, J. A. (1998) *Análise dos diferentes Tipos de Software Usados na Educação*, NIED-UNICAMP – In III Encontro Nacional POINFO – MEC, Pirenópolis;
- Vieira, F. M. S. (1999) *Avaliação de Software Educativo: Reflexões para uma Análise Criteriosa*, <<http://www.edutec.net/Textos/Alia/MISC/edmagali2.htm>>

Procedural building-up of geometrical object in concepts and strategies of primary school pupils

Dana Tržilová, trzilova@pf.jcu.cz

Dept of Mathematics, Faculty of Education, University of South Bohemia

Jiří Vaníček, vanicek@pf.jcu.cz

Dept of Computer Science, Faculty of Education, University of South Bohemia

Abstract

The article concerns the research of the geometry perception teaching of primary school pupils by means of computer application involving turtle graphics. Authors deal with problems and the research of LOGO environment usage for creation of ideas of dividing and assembling the complicated geometrical figures at the primary school pupils. The main goal was to encrypt which strategies do children use by discovering which part of given shape is capable to draw by applying an algorithm built from commands of several given types.

The application called Obkreslovačka (Encircler) has been created in the Imagine Logo for testing in schools. Turtle painting commands have been limited only to STEP, LEFT and RIGHT which only allow encircling the rectangular shapes. Some operating commands for the cycle have been added. Writing the turtle program by the commands typing has been replaced by the arrangement of command buttons due to the age of tested pupils. They had to pass several tasks to fill a given figure. They could create and use the turtle program to encircle the closed rectangular shape and to pave the rest of the figure by triangle-shaped tile objects. They were forbidden from filling the whole figure only by paving without using the turtle program.

The course of research tries to catch in what manner and what strategy a pupil divides the figure into parts created by procedural way and how to discover the symmetry of the shape parts and its reflection in the written program, what help gives the elementary geometrical shapes from which the figure is possible to compose. Based on these tests results a new set of tests has been prepared for the next testing in some South Bohemian schools, in the classes from three to five grades. The results of the main tests will be presented at the conference.

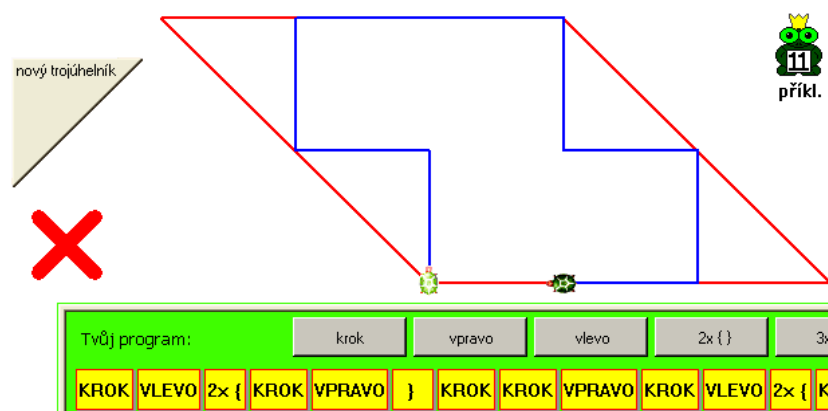


Figure 1. The environment of the testing application of Encircler

Keywords

geometry, primary school, building-up of geometrical object

Introduction

The contribution of micro-world Logo for geometrical imagination development and perception of space and shapes (turtle coordinates, creation of geometrical shape by commands for movement of an object defining the shape) is really well known. The modeling and drawing in the environment of turtle graphics is contributory to the teaching geometry based on the activity of a pupil and also to the expansion of the tasks repertory being available for the teacher. This leads to the more precise pupil conceptions creation.

Our concern is to find out how primary pupils are able to perceive geometrical figure as divided into two parts of various nature: a) part created by the procedural process (in terms of turtle graphics commands) and b) part paved with given basic tile objects. That was the question how pupils will discover which part of a figure can be drawn procedure-aided and in which method pupils discover this shape in each specific task and what strategies to solve such tasks will be created by them. Another question was how pupils form the appropriate procedure to create the shape.

We have started from the requirements of Czech and Slovak experts who declare the geometry teaching in primary school is based on the pupils experience obtained during solving various problems. For example František Kuřina says: "Geometry in primary school has not even a preparatory deductive character but it is markedly operational." (Hejný and Kuřina, 2001) Hejný argues similarly: "Experimentation is a basic and irreplaceable way to obtain geometrical experience. Pupils in primary schools have a natural ambition to investigate world by their own activity" (Hejný et al., 2004). Kuřina worked out a theoretic analysis of primary pupil geometrical experience and expressed four didactical principles that should form didactical structure for geometry teaching in primary schools (Kuřina, 2001). The presence of these principles is commented in the described investigative project.

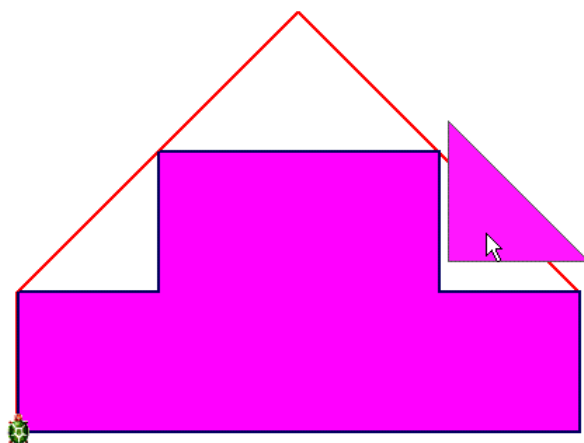


Figure 2. Procedural part of the figure and the rest paved by triangle tile object

Space Partition

It is applied when introducing the new notions into geometry (line divides a plane into two parts). Pupils meet the space partition principle from the babyhood. The space partition principle is included in that part of the project when a given figure is divided into the parts according to the way of creation, i.e. into the part that can be modeled by the use of the set of turtle graphics commands, and into the part that can be paved by predetermined basic tile objects.

Space Filling

The fact that the figure besides its sides also contains other points of the space is emphasized by the space filling principle. The areas of geometrical figures can be measured by filling the space. Calculations form traditionally a part of geometry teaching but it is also important to teach the pupils to observe and to guess. Although the assignment of testing tasks is to encircle a given figure the pupil fills the shape with objects created by procedure or paves the tile objects.

Construction

Constructions mean not only solving the geometrical tasks by means of drawing but also various ways of geometrical figures representation (for example construct a building from two, three, four bricks of the children building set). Many questions concerning the ordinary life also have a constructional character (determination of the journey from the place A to the place B). Modeling the figure by use of writing the program brings the possibility of a deeper understanding geometrical figure attributes. Above all the use of cycle leads to the idea of symmetry and its discovering both in shapes of parts of objects and in shapes of their perimeter lines.

Movement in the space

The idea of the direct movement and experience with it is good developed at a normal child aged of six years in our conditions. It can be checked on the tasks in which we search the traces on the plan. The movement in the space can be interpreted as the imagination of turtle movement when encircling the geometrical figure. Here the procedural construction of figure (or procedural shape) means the shape created by commanding, by means of which a given figure will be encircled (figure 2).

Main research objectives

Aspects of the process of polygon conception understanding

- How affects the computer and turtle graphics use a deeper comprehension of the polygon conception
- Understanding of the polygon as a part of the plane (it is not an outline demarcating it, but it is also created by the points from inside)
- propaedeutics of the symmetry conception as one of the polygon attributes

Geometrical transformation from a dynamical point of view

- paving of triangle tiles, propaedeutics of the geometrical transformation conception (congruency, translation, rotation, axial reflection)
- how will a procedural part of a given figure be found (i.e. the ability to find a figure of a specific shape in the given figure, particularly the shape of rectangle, which they are able to model by means of given set of motion commands) in a given polygon
- which shapes seem to be simple and which difficult and why to children
- which strategies to discover a procedural created shape will be chosen

Testing environment

An application called Obkreslovačka (Encircler) to test pupils in schools has been created in the Imagine Logo environment (figure 3). We have chosen this environment thanks to its three advantages: Imagine Logo naturally involves turtle graphics, enables to access object-oriented (Blaho and Kalaš, 2001) and is accessible to Czech schools.

The commands of turtle painting have been limited only to KROK (STEP, go ahead a constant distance), VLEVO (LEFT, turn left 90 degrees) and VPRAVO (RIGHT, turn right 90 degrees) because primary school pupils are not familiarized with an angle conception and miss the required skills to work with angle size and the research has been intended for schools without any prior experience with Logo. So the limited commands of turtle graphics only allowed encircling the rectangular shapes put together from vertices of a base grid of a graphic screen.

Some steering commands for cycle REPEAT 2x, REPEAT 3x, REPEAT 4x and brackets for defining lists of repeated commands were added to the commands controlling the turtle movement. Of course, to nest cycle commands was possible.

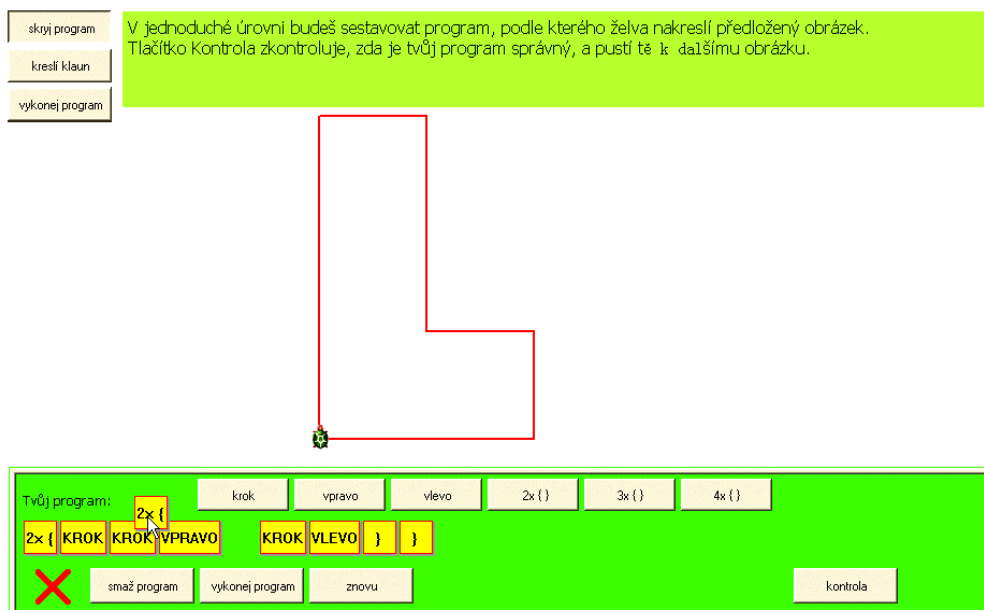


Figure 3 – The environment of the testing application, the level No. 1

To encircle the given geometrical shape in the Encircler environment was the typical task of testing. That meant to fill the given figure or its part with a rectangular polygon created according to the “turtle program” made by the tested pupil. In case of a turtle return to a start-up position after executing the program, an inside of the delineated polygon was coloured and could fill the whole object or its part.

Some given figures have not only the rectangular shape, but there are some skew line segments, so it is impossible to fill the whole figure only by using the turtle program. In this case, the user was allowed to add some other tile objects to the figure to fill gaps in the given polygon. These tile objects had exclusively the shape of right-angled and isosceles triangle with legs of the length of STEP, so that the two triangles could fill a basic square of the grid together.

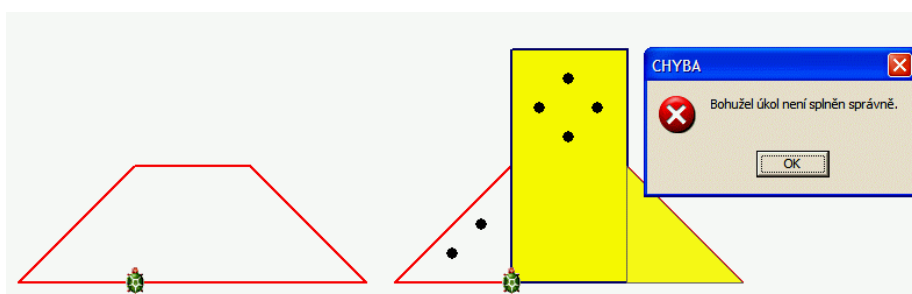


Figure 4. A settings of the task and a result of children solving check – there is one tile object missing and the turtle program is wrong.

The user could move such objects in this manner: to drag and drop through the grid, to turn right 90 degrees by a right-click, to remove it. It was not allowed to place two triangles in such position to fill a basic grid square. Thereby we prevent the pupil from paving the whole shape of given polygon by these tiles without creating a program.

Due to the age of tested pupils, writing of the turtle program by typing the commands was replaced by arrangement of command buttons. The user could modify his program by dragging buttons and could try and check it visually at his discretion; he could send his work to be checked only if he was sure that the whole figure was filled. After the request to check, computer checked whether the pupil filled a given figure correctly and any triangle-shaped tile object does not overlap the polygon made by the turtle; eventually it marked some mistakes and returned the task to be corrected by the user (figure 4). The tested person had the unlimited number of attempts to correct his solving, but he was allowed not to finish the task after the first unsuccessful check and start another task.

Testing tasks levels

The testing tasks have been divided into three groups varying by a difficulty of the given objects:

- The lowest level included only figures of simple rectangular shape made of base grid squares, e. g. rectangles of different shapes, shapes of L and U letters and bigger squares with startup positions on the vertices or the centres of the sides (the startup position is a place where a turtle is at start of executing the pupil's program). This level served mainly as a mean to familiarize with the application environment and the type of testing tasks. We used the same shape (e.g. square) several times with other starting point, which produced a couple of new tasks. We did not interest in the correct writing of the program encircling the object because our main point of interest was how children could divide given shape into the parts described above. In other words, we interested how children could find rectangular shape in the given object which was able to be drawn by the given turtle commands. So we could prepare more different situation looking like similar only by moving starting point of encircling on the same object.

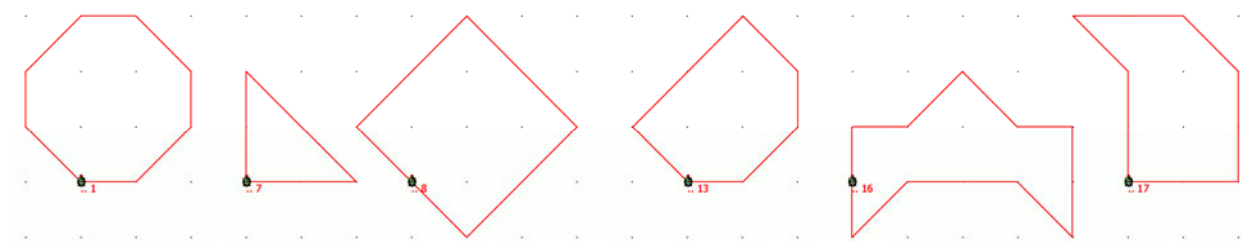


Figure 5. The second level shapes – in these shapes children had to discover a hidden part to encircle it by the turtle program.

- The shape of the setting figures used in the second level involved horizontal, vertical and transversal sides with a slant angle of 45 degrees, e. g. rhombs and rhomboids, big right triangles, octagonal shape of the traffic sign Stop, square staying on its top, a pinwheel etc. In this level, the first time pupils used tile objects to fill the figure. Each task had only one solution of a question - which part of a given figure is procedural (i. e. which part is creatable by a turtle program) - though algorithms and pupils programs leading to this solution were different (and were not evaluated). This level was pivotal and the most important for the research because it was the first time when pupils decided which part of the shape was procedural and which could be made by paving of triangle tile objects.
- The third level of the game was the hardest one. The tasks involved figures and environment of the same type as the second one, but the shapes of the figures were more difficult to make a turtle program for description of their procedural part. Only simple widening command buttons to a program row would lead to too long programs with a risk of overlapping a given length of the row. The pupil had to use more complicated structures, i. e. nesting cycles, or non-typical traces of depicting the turtle polygon, which had not to be always convex or coherent.

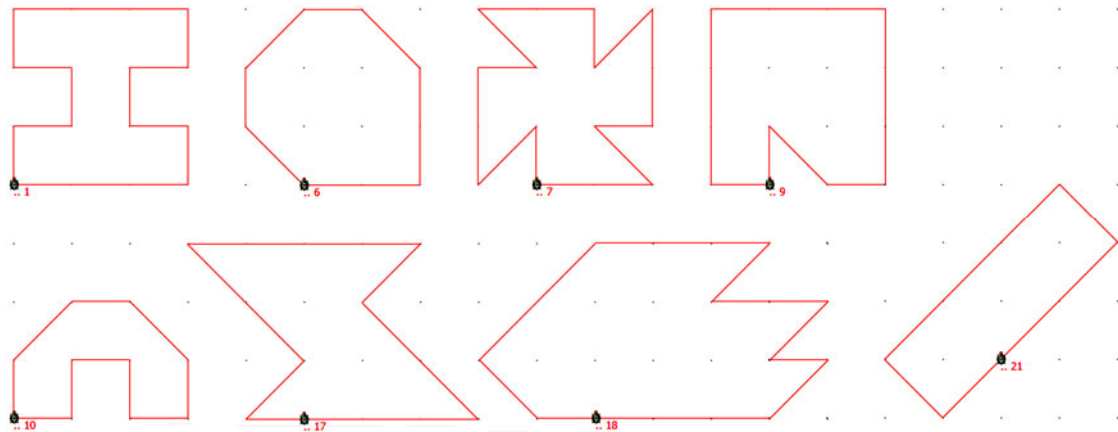


Figure 6. Shapes used in third level of the test.

Outlets and their analysis

After finishing each task level, following data used to the latest analysis of each tested pupil work have been saved:

- the given figure of task settings, the turtle program, the polygon painted by the turtle and position of each used triangle tile object have been saved after each request to execute the turtle program
- the same data have been saved after each request to check the task solution
- pupil answer to the question about the area of the given object has been saved when transitioning to a new task

The testing application enabled to show the same situation on the screen which was visible in the time of any request of the pupil to execute or check what is useful for the latest analysis of children process of the task solution creation. An examiner could see the same situation as the pupil whenever the pupil asked turtle to depict the program (e. g. when he was not sure or before checking his solution) and could check both the program and the shape of the polygon painted by the turtle. It enabled to find typical mistakes made by children and their strategies when encircling the given figure.

Course the test

The test consisted of an introductory training and three levels of the “game” in the Encircler environment. At each level, the tested pupil had to pass five tasks to fill a randomly chosen figure from a set of shapes for this level. Each task was finished with one question about the area of the given figure couched in squares of base grid or in the amount of triangle tile objects. The reason for including this question in the test was both checking the estimation in case of manipulated objects and a small relaxation before the next task and the extension of concentration time of the tested person.

Prior to the first test level pupils worked at the level of “training”, at which they could create simple turtle programs without any specific figure to encircle. The main purpose of the training was acquainting pupils with the environment of the Encircler application and introducing them to the turtle graphics and geometry by a simple way of using experiments and manipulations. This tutorial took about 20 minutes and the tutor had to lead and motivate pupils with the instructive tasks (to paint a square, to try to use the cycle buttons, to create stairs, to return the turtle to the start-up position) and help them individually.

After introducing to the environment, pupils have already worked without any help at the other levels of the “test game”. The tutor only checked the situation and tried to catch any information which the testing application could not have recorded. Tutors notes might be very useful for the later completion of the application. No limit has been determined for the time of the test.

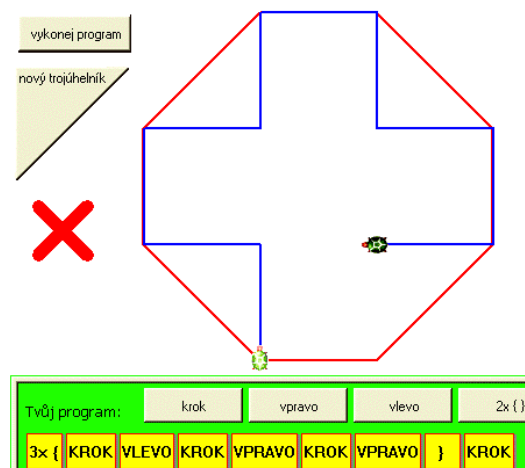


Figure 7. The second testing level of the test. The turtle creates the procedural part of the figure according to the program below

The objective of the first level (encircling the figures only by the turtle program use) was to introduce tested persons to the course of the test, type of testing tasks and to improve their skills to create the turtle program. Only the second level was pivotal from the research point of view. Pupils were examined how they divide a given shape to the part creatable by turtle painting and to additional parts created by paving the tile objects. The third level observed whether the pupils were able to divide more difficult figures and how they created programs for such a difficult procedural shape of the polygon.

The pretest – debug environment checking

The first objective of the pretest was to confirm the testing application in school conditions and the organization of the test. The second objective was to find out if qualitative data obtained within the testing are sufficient for the analysis of the way of pupil thinking and the observing his/her strategy.

The pretest took place in the primary school ZŠ Mladé, the suburban school in České Budějovice with pupils from fourth grade. These pupils have never worked in the environment of Logo before. 14 pupils were present within the pretest. They were divided into seven groups which appeared to be suitable especially in the first part of test – during the training and in the first level of test. The time period of the test was less than two school hours, i.e. approximately 80 minutes. The test was interrupted by the break with regard to the low age of pupils.

It also proved that random choosing of shapes from the set of figures creates (even with a big care of choosing) the non-comparable conditions for various pupils in particular at the first task. The result was the necessity to create a strict sequence of testing tasks even though it will be with regard to a various speed of pupils at the expense of the objectivity, because pupils of this age tend to advise in the event that they see the same task which they have already solved at their neighbor, even though they are not asked for it.

Observing the pupils in the second level of the pretest resulted in the conclusion that pupils approximately after two experiments moved to the strategy first to pave tile objects to a figure (some pupils immediately from the first task, one group at the last one). In this way they used the computer for drawing a rectangle as a rest of figure after paving tile object and for the ensued rest they wrote a program which was drawn by the turtle (figure 9). This meant for the following elaboration of the application the necessity to extend saved data concerning information, if at the beginning of each task a pupil first created a program or paved tile objects.

The first testing results show us the main goals for further improvement and arrangements of the testing application Encircler:

- it is necessary to add another testing level in which pupils are first allowed to pave tile objects to the figure only after finalizing the turtle program without any possibility to change it later
- to reduce the excessive check of a created program we ponder to add any form of counting the requests of the pupils (we also hope that pupils will be hereby motivated to make less attempts without thinking during creation of the program)
- to add information on the overlapping the maximal allowed length of the turtle program to the test record, which can be a trigger to shorting process of the programs by using cycles
- to add information to the test record whether the pupil started a new task solving by writing the program or paving a tile objects to the figure to get the required rectangle for turtle painting).

The test

Based on the pretests results a new set of tests has been prepared for the next testing in the classes from three to five grades. As the strategy “first to complete paving tile objects and then to write a program” showed to be a winner among the tested pupils, we suppose in the newest version of the Encircle a creation of another extra level of test in which the pupil will have to create first a program and then he/she will be allowed to pave tile objects. We suppose the pupils will be forced to think more abstract. This level was placed between second and third level.

The main tests took place in sixteen South Bohemian primary schools during two month, May and June 2007. Two hundred pupils were present within the test, 113 girls and 87 boys. Tests were realized by couples of teacher students who had been prepared at the faculty for leading initial training and for special technical and pedagogical situations.

The test proved that pupils are able to follow students’ instructions and manage to command the application without any marked difficulties that could misrepresent the outcomes of test by the fact that disability to command the application would have been the reason for failure during the testing.

Outcomes of the test

Our testers referred that children which all hadn’t programmed before were more kind and skilful than were expected. After preliminary evaluation of the test records and tester’s messages next findings can be framed:

Strategy of polygon creation

Some pupils verified their program too often, especially in the initial level and especially girls. After each change they verified the program that way that they let turtle draw an outline. Some of them evoked the feeling that they used a method try-error: they chose randomly any new command and added it at the end of the program. They verified their program immediately and then erased last button or left it. A typical problematic example was to turn the turtle in a concave angel of rectangle (figure 8). While observing them we have found that a part of pupils who seldom used verifying their program by the drawing the outline helped itself with the illustration of supposed turtle motion by a finger on the monitor.

The fact that girls were much more cautious is probably conformable with other researches, from which follows, that girls have not so direct relationship to the computer. There is a question, if it causes in virtue of the medium of computer or to the environment of the application and the character of the exercises.

It appeared that pupils didn’t understand their programs well because they corrected longer parts more often than it was essentially necessary. Another well used strategy was to delete the program and to begin creating it again but for the inverse turtle way through a figure. It seemed

like children supposed that opposite direction allowed turtle to reach the opposite side of the figure more easily. Children may have understood the tasks as not to encircle but to do through.

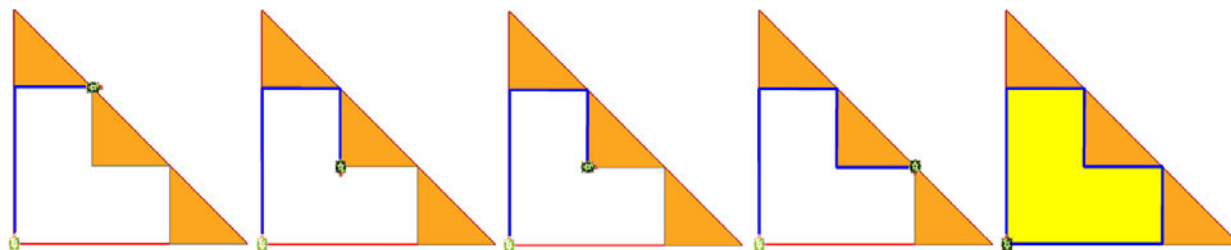


Figure 8. A recorded process of a turtle polygon creation. Maruška was a little cautious and checked it after any changes of the program.

The total feeling of the main test was that pupils have bigger difficulties to write a program than to discover division of a figure into a procedural and a tile object part. We could see many times that tile objects were dislocated correctly but the pupils were not able to write a program that fills the remaining rectangle shape. It appears that a longer time period within which the pupils will handle the turtle and write a program will be required before allowing the pupils to solve tasks of the main test. This can be supported with a longer and goal-directed training and a longer first level of testing.

The extra level of the test in which pupils have first to create a program and then to add pave tile objects caused big problems. The most of the children did not divide given shapes correctly even if they had a feedback. There was visible that even after several attempts drawn shapes did not get closer to the correct shape.

According to the achievement, we can divide pupils into three frictional groups. Beside expected groups of very successful and unsuccessful children one group more appeared. These children first work very slowly, with mistakes and a lot of checking attempts. About in the middle of the second level of the test they probably got familiarize with the environment because they start to work more quickly and most of the tasks were solved at once and without mistakes.

Cycle and symmetry

The analysis of the most difficult level of the test resulted in the finding that pupils begin to use a cycle for notation of program only when they are forced to shorten their program. Although the advantages of such notation were demonstrated to them in the introduction of work with Encircle the pupils give commands in a sequential consecutive manner. The restriction of the program length caused difficulties only in the most complicated level where the shapes, which were impossible to encircle by mere consecutive giving the commands, were included. Therefore the impulse for the structure program use was not the effort to simplify the program or to make it understandable but the effort to shorten it.

The pupils got themselves into the above mentioned situation unexpectedly even in more simple level when they needlessly drag out a program by composing the nonfunctional sequences of a program (for example LEFT RIGHT LEFT and etc.). Then because of the abbreviation they were adding a cycle only to the last part of a program instead of the whole program control.

Some pupils used the cycle only for repeating one command (REPEAT 3x [STEP]). Other pupils that inserted more commands into a cycle (for example ... REPEAT 2x [STEP LEFT STEP RIGHT] ...) could find a repeating part of a figure or the repeating sequences of program. The observation resulted in the finding that the approach described as the second one was prevailing.

Some pupils discovered the connection between the central symmetry and the program which was all written as a cycle (for example REPEAT 2x [STEP STEP LEFT STEP LEFT]). These

pupils tried to discover symmetry in further pictures and to use a cycle as a general approach to write a program for such kind of picture. One group discovered axial reflection at one picture but left the task without any trying to find another solution because it was not able to write a program as cycle. It looks like that they misinterpret this attribute also for axial reflection.

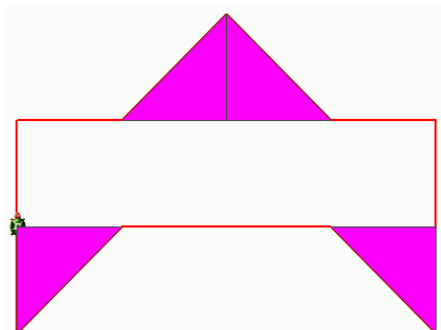


Figure 10. A strategy of paving tile objects first to get the shape of turtle painting.

Conclusion

The testing results show us the main goals for further improvement:

- Most of tested pupils have a problem with finding procedural part of the given shape. One of the future aims could be to discover which types of geometrical shapes are easier (and good for initial training of dynamical geometrical imagination) and which of them are more difficult.
- There is an interesting point, how children get the same object with different starting point of the drawing. We haven't got any records where pupils automatically use prepared program used in the task before in the case of two tasks with the same object and different starting point. But the research was not targeted to this question. Therefore it could be interesting to allow children to set the initial direction and place of the turtle.
- It could be useful to arrange used figures of the first and second testing level not only firmly according to the difficulty, but to the rate of repeating of the outline fragments and to the symmetry of the figure.
- Some other direction of the research could go to applying of other grid than rectangular. How difficult could be shapes drawn into a triangular grid for the pupils?
- It is necessary to train tutors for testing so that they are able to serve the application and computer in a technical manner and to lead the initial training and introduce children to the environment of the testing application. They need to be more supplied with the schemes for the turtle painting (i.e. to place turtle at the specific position, to encircle symmetrical figures, to create non-coloured figure though the turtle finished its program at the start-up position) and a methodology how to explain the technology and advantage of using cycles in a program to pupils. They need to repeat the commands and explanations more often.

References

- Blaho, A. and Kalaš, I. (2001) *Object Metaphore Helps Create Simple Logo Projects*. In Proceedings of EuroLogo 2001. Edited by G. Futschek. Linz, August. Srpen. pp. 55 – 65
- Hejný, M. and Kuřina, F. (2001) *Dítě, škola a matematika: konstruktivistické přístupy k vyučování*. Portál, Praha
- Hejný M., Novotná J. and Stehlíková N. (2004) *Dvacet pět kapitol z didaktiky matematiky*. Pedagogická fakulta UK Praha
- Kuřina, F. (2001) *Geometrie a svět dětí (o vyučování geometrie na prvním stupni)*. Pedagogické centrum, Hradec Králové

ICT Appropriation by Digitally Excluded Communities: the role of the learning context

José Armando Valente, *jvalente@unicamp.br*

Department of Multimeios and Nied, Unicamp & Ced, PucSP

Abstract

One of the key aspects of living in the knowledge society is the ability to acquire and process information. This should be done by everybody, regardless of their physical, social or economic status. They should be able to use adequate tools such as the information and communication technologies (ICT). In this context, how can ICT be appropriated by functionally illiterate or socio-economically disadvantaged people, by the elderly or by handicapped individuals? This article describes three examples of ICT appropriation by individuals from low income communities and considered digitally excluded. These examples show that the learning context created was the key element to enable them to learn how to use IC technology and to apply it in their working situation. This work is based on the concept that I have called “contextualized constructionism”, based on Papert’s constructionism and Freire’s ideas that state that the more the learning process is related to the interest and the situation in which the learner lives, the better the chance of him/her of understanding about the content and thus, of getting involved in the educational activities.

Keywords

Digital exclusion, technology in education, learning process, low income community, information and communication technology

Introduction

The verb “to appropriate” has its roots in Late Latin *appropriāre* which means “to make one’s own”. The modern use of this word according to The America Heritage Dictionary has added some additional qualities to the appropriation process to mean “to set apart for a specific use” or “to take possession of or make use of exclusively for oneself, often without permission”. In this paper the use of the term “ICT appropriation” is related to the way people incorporate ICT into their lives, so they can do things that are meaningful to them. ICT becomes an integrated part of the activities people develop, a tool that is mastered and people feel they own it.

Even though the work that I did in the beginning of the 1980s using computers, particularly Logo, with physically handicapped children was not about computer appropriation, the way in which the computer was used and its role in these students’ lives is an excellent example of what appropriation means and what technology can do once it is appropriated. This work started as a proposal to study whether Logo could be “an information prosthetic for the handicapped” (Papert & Weir, 1978; Weir, 1981). It was developed with children physically handicapped by cerebral palsy and as the project progressed we learned that not only Logo help these children intellectually but that the computer could become a “scratch pad” for them.

Cerebral palsied children have a disorder of movement and posture due to permanent but non-progressive lesions of the brain which may affect motor areas as well as areas of the brain that support specific intellectual functions. However, these children’s motor impairment makes understanding and evaluating their cognitive deficiencies quite difficult. It is difficult to create interesting and challenging activities that these children can perform in order to evaluate their intellectual abilities. The computer provided the means with which these children could perform a series of activities such as writing, drawing, solving problems, keeping notes and doing calculations. Thus the computer became for them what I called a “scratch pad” (Valente, 1983, p. 21).

With the computer these children developed activities that helped them to improve their understanding of concepts in geometry and mathematics; acquire knowledge about problem solving such as planning, setting up goals, and debugging their plans; and improve their writing skills. The computer also provided a means for the researchers to understand a series of principles related to the creation of effective learning environments. One of the most interesting of these principle was how and when to intervene in these children’s activities to help them to overcome their difficulties without taking control away from them. The learning context and the child’s interest were more important for the effectiveness of the learning process than a teaching plan developed *a priori* and imposed by the teacher onto these children.

Today ICT is not confined to the computer realm and its dissemination in our society makes the appropriation process a major obstacle for many individuals to be able to function and to work. For the handicapped child, who could not use traditional objects to express his/hers ideas, it was quite normal that the computer became an important tool to master and to make one’s own. At that time, when computers were not disseminated in the society, to be able to cross the digital divide was a major effort however it made a big difference since for these children it was a matter of being part of the “normal” world. Today the situation is similar for the normal individual who has to cross the digital divide in order to be able to fully participate in society. The difference is that ICT is much more than just the computer, it is part of practically every activity we do and it has to be mastered by everyone. The digital divide becomes a real issue, and how to overcome it is a major challenge for the present society specially if we consider countries such as Brazil in which 75% of the population is considered functionally illiterate; or if we consider the global mobility of people who immigrate to developed countries and have to face the problem of mastering the culture and the language. Thus the question of how to overcome the digital divide is not a question of only underdeveloped countries but it is affecting everybody regardless of their physical, social or economic status.

In this paper I describe three examples that are part of a study we are conducting to understand how ICT can be appropriated by people living in socio-economically disadvantaged communities. These examples show that the learning context created was the key element in order for them to be able to learn how to use ICT and to apply it in their working situation. The pedagogical approach used was based upon what I have called the “contextualized constructionism” concept (Valente, 1999) which combines Papert’s constructionism (Papert, 1986; Papert, 1992) and the contextualization as proposed by Freire (1975).

The location where the study is taking place

This study is part of the **Healthy Community Program (Programa Comunidade Saudável)** which has as its objective to use scientific knowledge to benefit society in general and, more specifically, disadvantaged populations. This Program supports interdisciplinary projects, integrating health, education and economically sustainable actions according to the specifications proposed by the World Health Organization. This Program is being developed in the Amaraís Region, one of the poorest areas in the city of Campinas, and in Pedreira, a city nearby Campinas. Both cities are located in the state of São Paulo, Brazil. The Program is the result of a partnership involving a non-governmental organization, the Institute for Special Research for Society (*Instituto de Pesquisa Especiais para a Sociedade – IPES*); several departments at the University of Campinas (Unicamp); and secretaries from the municipal governments of the cities of Campinas and Pedreira (Martins & Rangel, 2004).

The ICT appropriation study is carried out by establishing, in each of these communities, a Nucleus of Work and Research (Núcleo de Trabalho e Pesquisa – NTP) to develop research-action projects for the improvement of living conditions in the community. These NTPs are formed by residents of the community, by professionals from social organizations, and by professors and students from Unicamp. ICT has been used as part of almost every activity developed in these communities and the examples show that the appropriation takes place in different forms and in different situations such as how the community painted the computers with different colors, the use of ICT by community health agents, and by community handicraftsmen.

Colorful computer laboratory

Appropriation of ICT in some situations means to give a distinctive touch to the physical object being appropriated. This is what has happened with the computers installed in one of the ICT facilities set up in the community. They are painted in different colors, with drawings so the computer laboratory has a personal touch; it is colorful, it has a feel of being a playground, different from the impersonal and serious black or beige computers in most of the laboratories we have in business or in schools.



Figure 1. View of the computer laboratory and a detail of drawings painted on the cabinet

ICT access is provided to the community through the installation of a Telematic Center for Continuing Education (Centro Telemático de Educação Continuada - CETEC). In the Amaraís Region in Campinas two of these Centers were created, and each one is equipped with a local

computer network connected to internet, two printers, a scanner, a digital camera and a video recorder. Some of the computers installed are provided by the project, specially the ones that are used as the network server. The other computers are donated by banks or by local companies in the community. These donated computers need maintenance provided by community members who were trained to be able to do basic computer repairs. In general, from 10 donated computers it is possible to get 6 to 8 computers working properly and the remaining computers are used for spare parts.

These computers are unassembled, their parts fixed, their cabinets cleaned and painted by community members, as shown in the sequence of pictures in figure 2. These tasks are part of what is called a metarecycling process, very common among the institutions that are set up to disseminate ICT to socially disadvantaged communities. The activities in these institutions are based upon the free software spirit, as described by The Association for Progressive Communications (APC) (www.apc.org/english/index.shtml).



Figure 2. Sequence of pictures showing the metarecycling process developed by the community members who work or use the ICT facility set up in the community

These tasks involve community members, especially adolescents who can learn basic principles of computer maintenance. Also this process of metarecycling gives the community a sense of the ICT ownership. They have the feeling that these computers belong to them. The community members take care of the CETEC and we have no problem with users damaging the facility or stealing computer parts.

Use of ICT by community handicraftswomen

Nazareth Evangelista dos Santos, known as Donaⁱ Nazareth, lives in one of the neighborhood, located in the Amarais Region, in Campinas. She was trained to be a Health Community Agent in 2001 and, as part of her job as a health agent, she had to visit each community family, thereby becoming a familiar face in the community. Gradually she became a community leader and recently was honored by the students of a local school who invited her to talk about the work she is developing in the community, as shown in figure 3.



Figure 3. Dona Nazareth visiting a community family and at the local school talking to the students

In the Amarais region there is a large incidence of illiteracy and Dona Nazareth started a program combining two activities. Dona Nazareth helps these people to learn to read and write and in exchange, they help Dona Nazareth in the production of handicrafts which are sold and the money is reverted to help the community and to support her alphabetization program.



Figure 4. Sequence of pictures showing Dona Nazareth working with handicraftswomen from the community and the products these groups has produced.

Dona Nazareth, together with her group of handicraftswomen, noticed that there is a great interest for this type of work in the community and through her visits as a health agent she has gotten to know several craftsmen and women. In general this work is informal although it has become an important source of income for the families. However, since this work is not organized, these artisans have difficulty buying raw material in order to reduce their production costs. In addition, they are unknown in the wealthy circles and therefore need an intermediary who has access to the fancy boutiques in the city. However, these intermediaries buy the crafts at an extremely low price and sell them at much higher price to the boutiques. The community craftswomen feel that this markup is too high and that they are being taken advantage of.

The solution that Dona Nazareth has proposed is to create a cooperative so these workers, as an organization, could buy and sell their products at a much better rate. In order to convince them to join the cooperative, she has started a campaign going around her neighborhood visiting and talking to these craftsmen about the advantages of organizing themselves as a cooperative or as an association. Besides the price argument she has talked to them about having their products shown on a web site so their work could gain important visibility. With the help of Unicamp researchersⁱⁱ she has learned to use a digital camera, to take pictures or short films, and to download this material into the computer at the CETEC, as shown in figure 5.

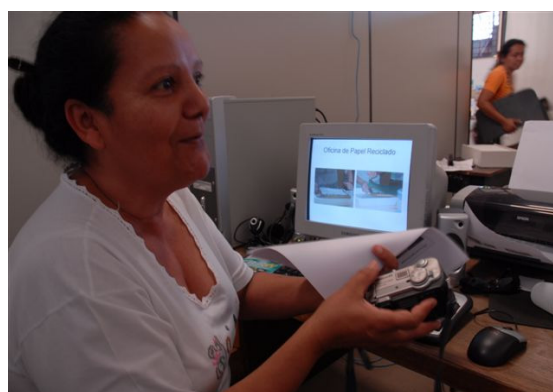


Figure 5. Dona Nazareth learning how to use the video camera and downloading her pictures from the digital camera to the computer at the CETEC.

Dona Nazareth has also learned how to prepared PowerPoint presentation using the pictures. These presentations are given to the community members, especially to the craftsmen, who are invited to the CETEC to discuss the ideas for the cooperative organization (figure 6). At these opportunities she shows them the pictures and talks about the development of the cooperative website. The fact that these artisans can see the picture of their product projected on the screen Dona Nazareth has created very powerful and convincing situation for mobilizing these workers to set up some kind of organization.



Figure 6. Dona Nazareth giving presentations to the community members at the CETEC.

The next goal that Dona Nazareth has set for herself is to learn how to create a website. She is receiving help from Unicamp researchers and students who work in the project and she is very excited about giving worldwide visibility to the community handcrafts.

Use of ICT by community health agents

Community health agents are people who live in the community and are trained to keep a close relationship between the families in the community and health centers set up in the region. These agents know the families, visit them with certain regularity and can either provide the necessary health support needed, refer a sick person to the health center or report incidence of diseases in particular areas to the health center. Fifteen (15) health agents from Barbim neighborhood in Pedreira participated in this study. They were between 20 and 45 year old, all of them had completed their secondary education, and seven were continuing higher education, five at the college level and two at technical school. They were not using any kind of ICT prior to the project, although five had a computer in their house and four had an e-mail.

An ICT facility was set up at the community health center consisting of five computers connected in a local network linked to the internet, a printer, a scanner and a digital camera. The work with these agents regarding the use of ICT was done by two graduate students from Unicampⁱⁱⁱ who

were interested in studying the ICT appropriation process. The content related to health issues in the agents' activities was provided by the health professionals working at the health center or doctors from Unicamp who were part of the NTP. These agents used the ICT facility to develop several activities such as writing reports; preparing multimedia presentations to show and discuss issues related to health problems with community members, such as high blood pressure and sexual transmitted diseases; developing a web site to disseminate their work; and setting up spreadsheets to organize their work and to help them to process data in order to make decisions. The health agents and the students had a face-to-face encounter once a week and the rest of the week they received technical support from the students through TelEduc, a distance education platform developed by Unicamp (Rocha, 2002).

The work with spreadsheets is particularly interesting since these agents had, as part of their activities, to keep track of the families visited. To do so they kept records and notes in exercise books where they registered data for their personal control. Also they had to fill out forms with information about children, pregnant women, and the elderly and about the incidence of diseases such as diabetes, arterial hypertension, and tuberculosis. These forms were given to a health professional working at the health center who fed the computer based "Information System for Basic Referral (Sistema de Informações da Atenção Básica – SIAB), maintained and controlled by the State Secretary of Health.

Based upon the experience these agents were having with collecting, registering and working with information, they began to discuss with the graduate students the possibility of developing a tool that would be easy to use and would help them to systematize their data. In this way they could keep better control of their family visits and could process the information in order to be able to make local decisions. Also they expressed a strong desire to learn about SIAB.

The SIAB system was developed for the DOS system, its structure is very rigid, and the form used to feed it look like a spreadsheet. The experience with SIAB and the discussions the agents had with the researchers led them to use MS Excel. This way each agent could develop a spreadsheet and structure the data according to a personal working style, and the specificities and necessary control for each community area. The idea was to facilitate the task of collecting and retrieving data about the families visited so each agent could have a better relationship with the families, could exchange more accurate information among health agents and keep the health center better informed.

As a result of this work each of the 15 health agents produced their own unique spreadsheet. The spreadsheets varied according to their visual design and structural organization of the data. Figure 7 shows three related spreadsheets: a general one (the biggest) and two smaller ones. In the general one the health agent organized his patients by address, according to the trajectory he travels to visit the families under his responsibility. Each day, before leaving for his visits he would print out this spreadsheet, fill it up by hand as he visited each family and at the health center he would update the digital version on the computer. This health agent decided to create two separate spreadsheets, one for each particular group at risk, hypertension and diabetes respectively. In the general spreadsheet the agent indicated the group at risk a patient may belong to (G.RISCO column).

MICROÁREA 4																			LISTA DE VISITAS															
AGOS/SET 2004																																		
ALTOS DE SANTANA																																		
Rua XXXXX			Endereço	CT NASC.	G.RISCO	C<1	C<5	HA	ID	F	DM	MF	NEO	ES	AD	26	27	28	29															
S/N	peessoa		Endereço	21/08/2027	38-39-54				38	39	54																							
	peessoa		Endereço	17/12/1931	15-39						39	15																						
Nº4	peessoa		Endereço	21/07/1954																														
	peessoa		Endereço	20/05/1976	43																													
	peessoa		Endereço	30/03/1983																														
	peessoa		Endereço	04/06/1984																														
Nº 78	peessoa		Endereço	01/08/1956																														
	peessoa		Endereço	27/10/1963	43																													
	peessoa		Endereço	16/01/1993																														
Nº 98	peessoa		GRUPO DE RISCO																															
	peessoa		HIPERTENSOS																															
	peessoa		Nº NOME	DN	IDADE	ENDEREÇO	Nº	ULT. CONS.	PRÓX. CONS.																									
	peessoa		1 pessoa	21/08/2027	77	Rua X		s/nº																										
S/N	peessoa		2 pessoa	24/02/1942	62	Rua X		50																										
	peessoa		3 pessoa	23/10/1946	57	Rua X		507 c																										
	peessoa		4 pessoa	10/08/2028	76	Rua X		637 b																										
	peessoa		5 pessoa	06/08/1935	69	Rua X		638 h																										
Rua YYYYYY																																		
Nº9A	peessoa		6 pessoa	GRUPO DE RISCO																														
	peessoa		7 pessoa	DIABÉTICOS																														
	peessoa		8 pessoa	Nº NOME	DN	IDADE	ENDEREÇO	Nº																										
	peessoa		9 pessoa	1 pessoa	17/12/1931	73	Rua X		s/nº																									
	peessoa		10 pessoa	2 pessoa	24/02/1941	63	Rua Y		nº 50																									
Nº9B	peessoa		11 pessoa	3 pessoa	10/08/2028	76	Rua Y																											
	peessoa		Endereço	4 pessoa			Rua Z		s/nº																									
	peessoa		Endereço	5 pessoa	19/02/1954	50	Rua Z		s/nº																									
	peessoa		Endereço	6 pessoa			Rua Z		s/nº																									
Nº50	peessoa		Endereço	7 pessoa	02/08/1961	43	Rua Z		s/nº																									
	peessoa		Endereço	8 pessoa	20/08/2029	74	Rua W		s/nº																									
	peessoa		Endereço	9 pessoa	26/08/1947	57	Rua W		s/nº																									

Figure 7. General spreadsheet showing the organization of data according to community members' address and spreadsheets of group at risk, hypertension and diabetes

This particular health agent was very reluctant to learn MS Excel since he found the program very complicated. Gradually he was able to master it and became one of the more knowledgeable members of the group about this particular software and about ICT in general. He played a very important role among his colleagues.

The process of incorporating spreadsheets in these agents' working routine was very slow and gradual. It took them almost three months to organize the data from about 8.000 families. During this task the agents learned a great deal about MS Excel, and in order to accelerate data entrance they learned about features such as how to use filters. Also, besides the lack of practice with the computer, the agents were afraid to loose their data if they made a mistake. This lack of confidence made them hold onto their exercise books and this practically doubled the time it took them to up date their files.

Today the spreadsheet is totally integrated into these health agents' work. This has made their work much more reliable and better organized, facilitating their action in the community. They have learned to process the data by using different representations such as bar or pizza graphics. This has helped them to interpret their data so they can see patterns and tendencies, which have helped them to make local decisions or to alert the health system if they encounter any abnormality. Also, not only have they learned about spreadsheets but they have learned about their own work as well. The process of producing the spreadsheets required them to make the strategies they used explicit and to learn about medical content. The new organization of the data helped them to identify information that was not clear before. This required them to research about public health issues in order to account for the information found.

How ICT appropriation is possible

The majority of researchers working with the digital inclusion agree that it is not enough to just provide a socially disadvantaged community with access to technology. Besides this access it is

necessary to create opportunities for the community to incorporate ICT into the activities they develop and thus to be able to appropriate them (Sorj, 2003; Silveira, 2003). The best practices in this sense are the ones that see the community members as active producers of content for the community and thus are able to produce change in the community (Pinkett, 2001). Warschauer (2003) has emphasized the fact that we should give more attention to the construction of a community structure that gives support to the process of ICT appropriation, rather than only providing hardware and software to the community.

The provision of support to the community in our case has been done through the creation of the NTPs which are formed by the community members, by professionals from social organizations, and by professors and students from the university. However, the strategies we created to work with the community are based on what I have called the contextualized constructionism approach (Valente, 1999). Papert's constructionism idea is related to the opportunity for knowledge construction that can happen when the learner uses ICT tools to develop activities that produce a concrete object (Papert, 1986; Papert, 1992). The contextualized aspect is related to Freire's ideas, showing that the more this product is related to the interest and context in which the learner lives, the better the chance of him/her of understanding about the content and of getting involved in the educational activities of producing this product (Freire, 1975).

In fact, the ICT are creating circumstances for people to express themselves as a whole, and not only the cognitive but the affective and social as well. The resources to explore the aesthetic aspects and the possibilities of forming networks of people interacting face-to-face or via internet have facilitated the exploration of these other human being dimensions, forcing us to continuously rethink our role as learner, the role of technologies in this process and our conceptions about learning, especially when done with the help of the ICT (Valente, 2003).

Also our approach takes into consideration the fact that we are working with adults, who according to the andragogy theory formulated by Knowles (Smith, 2002; Knowles, 1990) states that adults are self-directed human being, they have accumulated a lot of experience that becomes a resource for learning, their learning interest is oriented towards social aspects, and they want to see an immediate application of what they learn. These principles are aligned with Freire's contextualization concept.

These three concepts imply that an effective learning situation should not be based on formal lecture about ICT. The learning situation created in a CETEC is centered on: A learning environment that fosters knowledge construction according the learners' context;

- The use of ICT as tools for the learners to develop practical tasks and to interact with others;
- The intervention of people who can function as specialists to help the learners when it is necessary.

Thus all the tasks that were mentioned here were proposed by the community members and were decided on based upon discussions that the specialists had with the community members. Also, the specialist provided help when necessary and always intervened in the learning situation with the understanding that it was important to respect individual differences in terms of interest, learning style, and the knowledge each person had about ICT and about the task being developed.

In this context ICT appropriation by the community member was possible because they were learning about several concepts in the process of developing the particular activity: there were personally relevant concepts involved in the specifics tasks, varying from how to fix computers to knowledge about topics such as hypertension, diabetes etc; there were concepts about ICT since a community member used different software to develop various types of activities; and there were concepts about how to work cooperatively since they had to learn from one another or to organize themselves in order to be more effective.

Conclusion

The pedagogical approach we are taking in working with digitally excluded communities is based upon the idea of providing people with a more contextualized learning experience in terms of their background, their work, their interests, and their learning style. In this context, community members are learning about ICT and how to use ICT in activities related to their work or personal interest. Certainly this is much more than just providing access to the technology.

The examples discussed here show that these community members are incorporating ICT into their lives so they can do things that are meaningful to them. The ICT appropriation is happening in different ways. Physically, when community members paint the computers so they become very distinct from the computers we have. Socially, when the ICT are used to give visibility to the handcrafts they produce, and showing that the community can be more effective if its members act as an organization. And intellectually when health agents can make better decisions because ICT can provide a better representation of what these agents do.

In this context these community members can see that the IC technologies work for them. ICT appropriation becomes important, even though it is not an easy task. Each community member had to accomplish a lot. However, getting involved in this appropriation process means mastering the tool of the knowledge society and crossing the digital divide.

References

- Freire, P. (1975). *Pedagogy of the Oppressed*. New York: The Seabury Press.
- Knowles, M. S. (1990) *The Adult Learner. A neglected species* (4e). Houston: Gulf Publishing. First appeared in 1973.
- Martins, J.P.S. and Rangel, H.A. (org) *Campinas no Rumor das Comunidades Saudáveis* (pp. (209-218). Campinas, SP: IPÊS Editorial.
- Papert, S. (1986). *Constructionism: A new opportunity for elementary science education*. A proposal to the National Science Foundation, Massachusetts Institute of Technology, Media Laboratory, Epistemology and Learning Group, Cambridge, Massachusetts.
- Papert, S. (1992). *The Children's Machine: rethinking school in the age of the computer*. New York: Basic Books.
- Papert, S. and Weir, S. (1978). Information prosthetics for the handicapped. *Artificial Intelligence Memo nº 496*. Massachusetts Institute of Technology, Cambridge, Massachusetts.
- Pinkett, R.D. (2001). Redefining the Digital Divide. Available in: tcla.gseis.ucla.edu/divide/politics/pinkett.html
- Rocha, H.V. (2002). O Ambiente TelEduc para Educação a Distância Baseada na Web: princípios, funcionalidades e perspectivas de desenvolvimento. In: M.C. Moraes (Org.) *Educação a Distância: Fundamentos e Práticas*. Campinas: NIED-Unicamp, p. 197-212. Available in: www.nied.unicamp.br/oea.
- Silveira, S. A. (2003). Inclusão digital, software livre e globalização contra-hegemônica. In: S.A. Silveira and J. Cassino (Orgs). *Software livre e inclusão digital*. São Paulo: Conrad.
- Sorj, B. (2003). *Brasil@povo.com: A luta contra a desigualdade na Sociedade da Informação*. Rio de Janeiro: Jorge Zahar Ed; Brasília, DF: Unesco.
- Smith, M. K. (2002) 'Malcolm Knowles, informal adult education, self-direction and anadragogy', *the encyclopedia of informal education*. Available in: www.infed.org/thinkers/et-knowl.htm.
- Valente, J. A. (1983) Creating a Computer-Based Learning Environment for Physically Handicapped Children. *Technical Report 301*, Laboratory of Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Valente, J.A. (1999). Teacher Training: different pedagogic approaches. In J. A. Valente (Org) *Computers in the Knowledge Society*. Campinas, SP: UNICAMP/NIED. Available in www.nied.unicamp.br/oea/pub/livro1/livro_ingl.zip, chapter 6.

Valente, J.A. (2003). How Logo Has Contributed to the Understanding of the Role of Informatics in Education and its Relation to the Learning Process *Informatics in Education*, Institute of Mathematics and Informatics, Vilnius, Vol 2, Number 1, p. 123-138.

Warschauer, M. (2003). *Technology for Social Inclusion: rethinking the digital divide*. Cambridge, MA: MIT Press.

Weir, S. (1981). Logo as an information prosthetics for the handicapped. *Working Paper, nº 9* Division for Study and Research in Education, Massachusetts Institute of Technology, Cambridge, Massachusetts.

ⁱ Dona is the feminine version of Don and is used associated to the name of a person, especially women, to show respect.

ⁱⁱ I thank Maria Cecília Martins from Nied at Unicamp for helping in this work.

ⁱⁱⁱ I am particularly grateful to Carla L. Rodriguez and Lia Cristina B. Cavellucci for carrying out this work.

MicroWorlds Potential for Creating Educational Software for School

Alla A. Vitukhnovskaya, alla@onego.ru

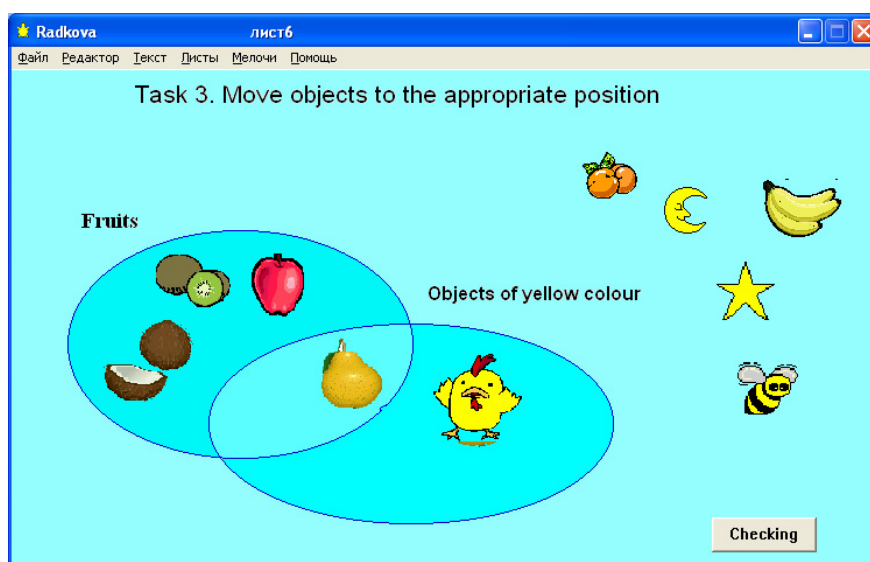
Dept of Physics and Mathematics, Karelian State Pedagogical University

Abstract

In the department of Elementary education Karelian State Pedagogical University within the framework of specialization "Teacher of mathematics and informatics into 1 - 6 classes" the two-level process of instruction in the Logo programming language is being carried out. During the first stage (in the special course "Basics of Programming in logo Language") the bases of Logo programming language are studied. The purpose of the second special course - "Information and communication technologies in elementary education" - is forming the skills of would-be teachers in the field of design and developing the complex designs - educational software for primary school (ES) with the use of Logo programming language. In the process of creation of projects the students become acquainted with the technology of the creation of projects, they generalize and systematize knowledge, obtained earlier:

- the means of the conclusion of information, organization of dialogue, change of personnel, etc are revealed;
- compare the different methods of obtaining of one and the same result;
- select the optimum method of achievement of the objective as a result of search and research activity students they create individual projects.

In the article the ideas and the practical decisions of the students, who found embodiment in the finished projects from mathematics, informatics, biology and to other subjects of the course of our long-standing work on the course, are generalized. The formulated author's theses are illustrated by the fragments of the programs, developed by students (example of the assessment frame from the program on the informatics).



Keywords

Educational Software; Primary School; MicroWorlds; Technology of Educational Software creating; Would-be informatics teachers

About the course “Information and communication technologies in Elementary education”

In the department of elementary education Karelian State Pedagogical University within the framework of specialization the "teacher of mathematics and information theory into 1 - 6 classes" is achieved the two-level process of instruction in the Logo programming language is being carried out. During the first stage (in the special course "Basics of Programming in Logo Language") the bases of Logo programming language are studied. The purpose of the second specialty course - "Information and communication technologies in elementary education" - it appears formation the future teachers of skills have in the region of design and developing the complex designs - educational software for primary school (ES) with the use of Logo programming language. According to specialists Logo programming language is the training language of programming. However, in our opinion, it contains great potential, which makes it possible to use it as the professional language of programming. In particular the means of the Logo language prove to be effective with writing of pedagogical software, and first of all, training programs for the students of elementary school and 5 - 6 classes. Why have we chosen pedagogical software? This is caused by the following circumstances:

1. ES for the elementary school must have the specific special features: comfortable interface, convenient navigation, the high degree of the clarity of training material, and for the students to desirably ensure the possibility of fulfilling various actions. All these special features can be successfully realized by the means of the language of Log (in particular, in the Micro Worlds environment).
2. In order to achieve the desired result and to create a good project, it is necessary for the students to implement search activity and to manifest and to develop systematic skills: to analyze, to compare, to assume optimal solutions.
3. Our students - the future teachers of information theory in the elementary school - with the development PPS already project the educational environment, in which will be used their program. Sometimes students succeed themselves in using a result of their training activity during the pedagogical practice on the lessons in the school.
4. On the object lessons, where the programs, written in the language of Log, are used, children work in by familiar by it to environment, which makes their activity more useful more comfortable and more interesting. However, the possibility of organizing the integrated lessons on the information theory and the corresponding object is created for the teachers. In the process of studying the course the students become acquainted with the requirements for pedagogical software for 1 - 6 classes, with the types of pedagogical software and the stages of their development. Each student appears in two roles: methodologist on one of the training objects and the programmer. As the methodologist he studies particular topic on the selected subject area, selects the content and develops the scenario, on basis of which the algorithm and the program are created. In the process of the creation of projects the students become acquainted with the technology of the creation of projects, they generalize and systematize the knowledge, obtained during the study of the Logo language:
 - they reveal the means of the conclusion of information, organization of dialogue, change of personnel, etc.
 - compare the different methods of obtaining of one and the same result (for example, output to the screen of text or graphic object)
 - select the optimum method of achievement of the objective (for example, program or tool house; programming one or other object or another and others.) All enumerated problems are discussed in the studies. As a result of search and research activity the students create individual projects. In the article the ideas and the practical decisions of the students, who found embodiment in the finished projects from mathematics, information theory, biology in the course of our long-standing work on the course "Information and communication technologies in

elementary education", are generalized. The formulated author's theses are illustrated by examples - frames from the programs, developed by students.

Educational Software for School

We view *educational software for school*¹ as an application created to realize particular educational tasks, having content of a definite school subject and being aimed at interaction with a schoolchild (1). The main ES purpose is utilizing in teaching process. According to educational tasks solved there are a few types of Educational Software:

- teaching programme
- training programme
- knowledge control programme (control or testing programme)
- computer textbooks
- computer taskbooks
- laboratory practice aids (Bashmakov & Bashmakov (2003))

In the present article we are going to overview only the first three ES types. Let us, above all define them.

Teaching software – Educational Software for basic training in one or a few units (topics) of a curriculum. Training is implemented by means of concentrating on small information units and providing feedback on every step.

Training software are modelled for repeating and securing schoolchildren's knowledge and skills. The child is provided with more or less wide *range of tasks* in a particular topic, whereas a constant control and assessment (in a "reply" form), aimed at educational skills correction is supplied.

Knowledge control computer system is aimed at defining a schoolchild's proficiency in a particular subject, course, unit or topic and his assessment, based on established skill requirements (Bashmakov & Bashmakov (2003)).

Most generally the process of ES elaboration may be viewed as a sequence of the following steps:

- development of a programme project, resulting in elaboration of a scenario
- a programme realization
- a programme expertise

An Educational Software scenario (project) is a detailed description of a purposeful child's interaction with the computer with the use of a natural language. A scenario is a sequence of frames linked together following each other. The following frame types can be defined: title, information, instruction, navigation (menu), control and frame - reply. Each frame represents the contents on the screen and leads to the next ones linked to it.

¹ Different term – educational computer tools (1).

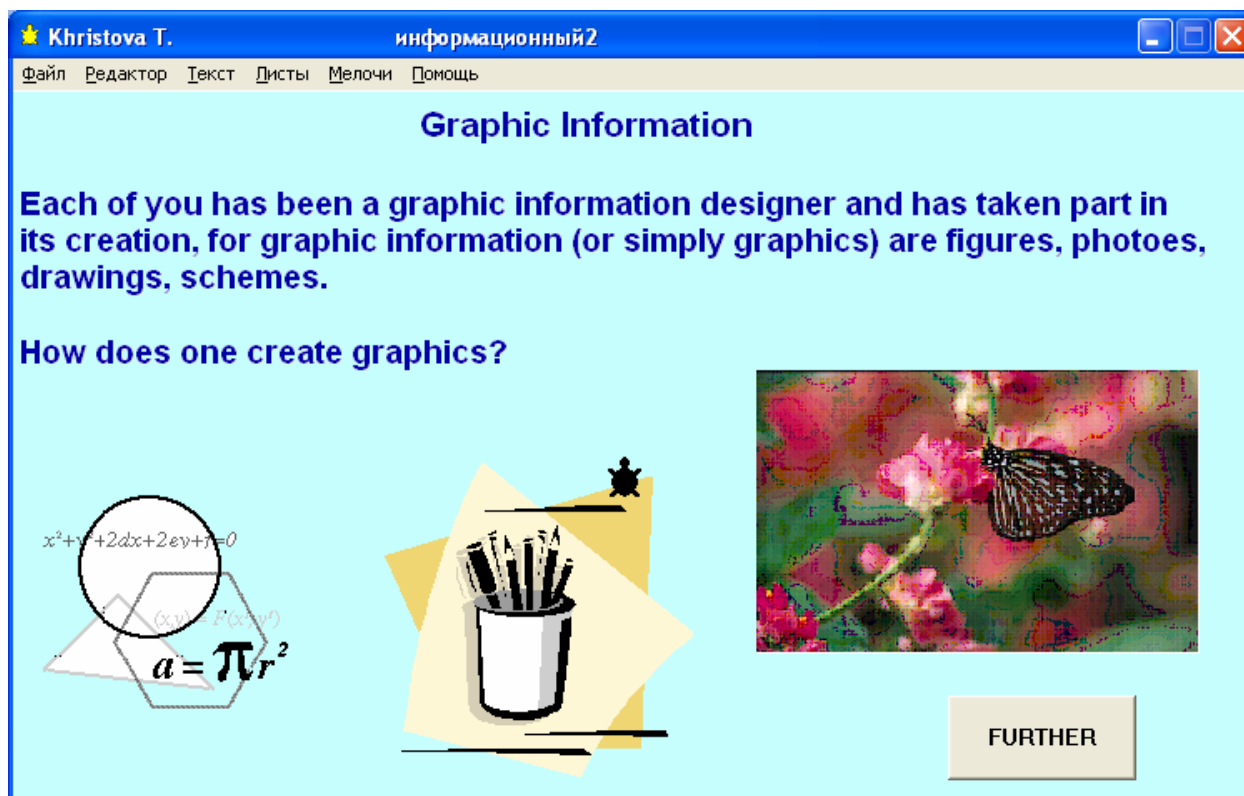


Figure 1. Sending and receiving common and private characters (style: Figure caption)

Structure of educational software for school

Educational software is elaborated on a scenario basis and, naturally, follows its logics. Each programme, in spite of its contents and purpose consists of necessary number of blocks (modules).

1. Screen output of information on the programme and authors.
2. Screen output of educational information (some new study information, tasks, exercises, questions, reference information, etc.)
3. Answer input
4. Check correctness of the given answer
5. Input of a reply – reaction to the quality of the answer given (message on correctness or mistake)
6. Assessment in control programmes

In the article the results of our research are being illustrated by the examples taken from the programmes created by the students. Below there are examples of frames taken from different programmes – educational, training and assessing. In the next part of the article ways of realization of every module by the Logo Language resources are to be analyzed.

Output of the information on the screen

While working with ES a schoolchild deals with information all the time – text, numbers, graphic objects, sound signals. Correlation of various types of information in the programme depends on the type of the programme, the subject, the topic, stage of studying process, school children's age and type of the Frame. Thus, in ES in local lore study graphic information (dynamic as well) will naturally prevail, and in software in the Russian and Foreign Language – text or audio information. Assessment programmes in any subject consist mainly of text and numerical data: tasks, exercises, questions.

Output of the information on the screen is actually carried out in all Frame types, however according to kinds of information to be output and ways of its output to the screen Title Frames, Assessment Frames and Frames-Replies may have a significant difference. (See Fig.1 for example)

Let us view various types of information for output on the screen and possible ways of creating it by means of Logo Language in MicroWorlds.

Text information

In Logo Language programmes text is put out on the special window (normally the text one), signal or dialogue one. The text window is known to be created in two ways: “manually”, with the help of the mouse or command **newtext**.

The text may be put out in two ways:

- with the use of integral text editor.
- by programming – (command print)

Text editing and processing is appropriately carried out with one of the following ways:

- with the use of the Text Editor commands
- with the use of programming (by using primitives of text processing)

In the process of ES elaboration the authors solve the problem of the way of text window creation and text output. If it is crucial that while implementing the programme the text is seen by the user, the programming way of text output is to be used. If a ready text-Frame is required, it is much more convenient to print it and process beforehand. While working at the programme it is often more interesting and convenient to see a message (reply, comment, assessment mark) not in the static text window, but in the signal window, which pop-up is accompanied with a sound. The announce command is used for this.

The signal window, which is put out by the command *inform*, is often used in Frames-Replies. In Assessment Frames another way of text output on the screen is used – command *question* and sensor *answer*. Command *question* opens the dialogue window, where the question (the input parameter of the programme) is printed and suggests to print the answer. After the answer has been put in, its text is saved in the sensor *answer*.

Graphic information

In ESS for Elementary school graphic information is most important – it allows to get educational principle of visual teaching, enforces motivation to study.

Ways of output of graphic information on the screen

1. Use of Graphic Editor for Bit-map Graphics
2. Programming (use of group of commands and sensors of the Logo Language, called the Turtle Graphics)
3. Import of graphic objects from files and the Internet.

Graphic Editor allows to create static objects and is most often used while making background and “non-processible” (not requiring manipulation, processing) objects in various programme Frames. *Programming* is convenient for elaboration of some graphic objects (for example, regular geometric figures), but mainly – for realization of dynamic graphics (creating simple moving objects and animation). Both of these ways allow merely put graphic objects out on the screen, i.e. make them visible, but, unfortunately, they do not let children operate with them, take some actions over them. One of the effective ways to represent graphic as well as text and numerical information on the screen – in order to manipulate them further – is the Turtle having the necessary form (picture, letter, word, number). We are going to scrutinize this way below, while analyzing ways of data input.

Organization of Dialogue

One of the ESS requirements is interactivity – the way of providing dialogue-like interaction between a child and the programme. The child should see (or hear) the task question or demand, enter the answer to it or solve the task and then wait for the system's reply to the answer given. Control of comprehension is carried out at all stages of studying process, while introducing the new information as well as during revising, so means of feedback must be projected in all types of ESS. MicroWorlds environment offers significant opportunities for “schoolchild – computer” dialogue realization:

1. Creation of the procedure may be treated as the simplest means of dialogue realization. Body of the procedure may include reply to the answer, if the answer coincides with the procedure's name.
2. Programming of objects (the Turtle, button, colour)
3. Working with the text window as the object
4. Command question and sensor answer. Command question opens the dialogue window, where the question (the input parameter of the programme) is printed and suggests to print the answer. After the answer has been put in, its text is saved in the sensor answer.

It is worth while providing conditions for forming skills to solve *various adequate* tasks with different objects in educational activity of young school students. Taking this into account the possibility to make different actions for choosing the correct answer without being restricted by the mouse click (as is often the case) should be provided in ES. Children's answer input in ES for School may be realized by making various actions with text and graphic objects.

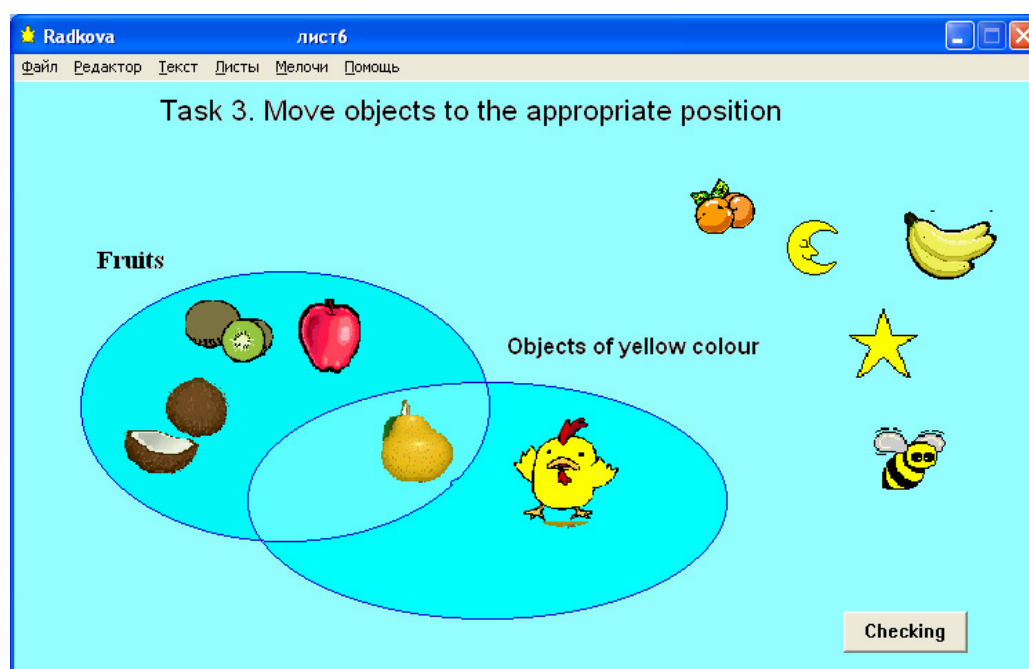


Figure 2. Assessment frame. The answer is put in by means moving the Turtles. After the answer is put in a click on the Check button. Assessment is carried out by comparing colour of the margin under the active Turtle with the sample number, corresponding with the real colour (50, 120, 10)

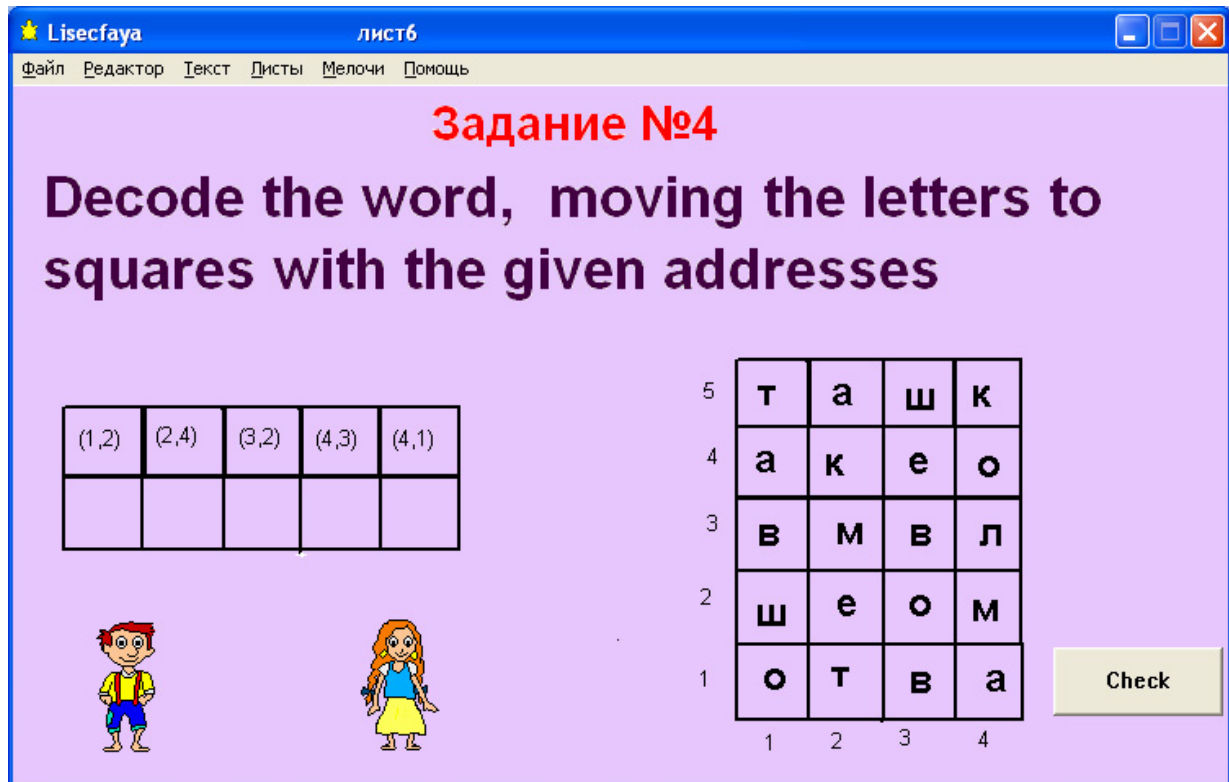


Figure 3. Assessment frame. The answer is put in by moving the letter-shaped Turtles. To start the programme the Check button is to be pushed. Assessment is carried out by comparing of the active Turtle's co-ordinate in axis x y , which is measured with the help of sensor $xcor$, with the pattern co-ordinates

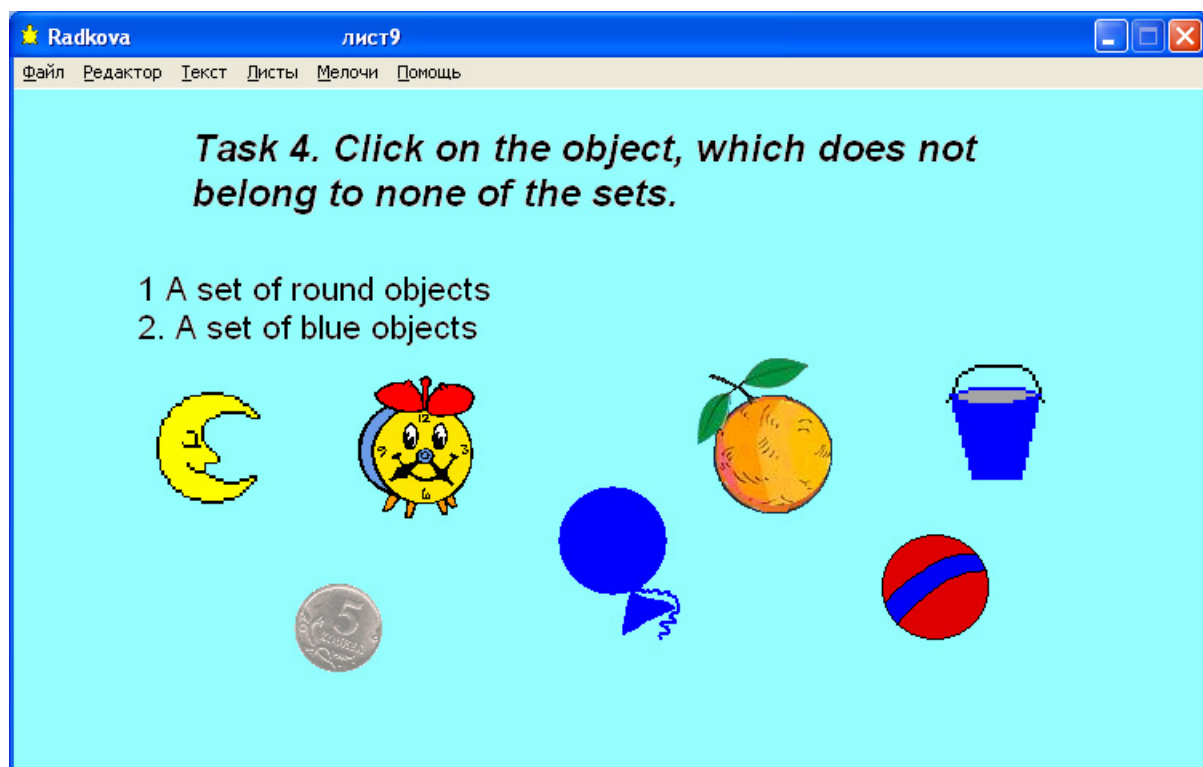


Figure 4. The answer is put in by clicking on the Turtle. Simultaneously the programme is started and the given answer is assessed

Ways of answer input

1. Clicking the programmed object (the Turtle, button, colour) lets starting the process at once and see the programme's reply on the screen. (Fig. 4).
2. "Manual" moving of the Turtle as a graphic object, word, symbol, number (see Fig.2, 3).
3. The Turtle manipulation (for example, changing its shape, colour)
4. Working with text:
 - text input (definite symbols, words, numbers) into the text or dialogue window (see Fig. 5).
 - text formatting (deleting, inserting, replacing part of a text)
 - symbols formatting (change of colour, size, outline)
5. Working with graphic images for example, deleting, filling in.

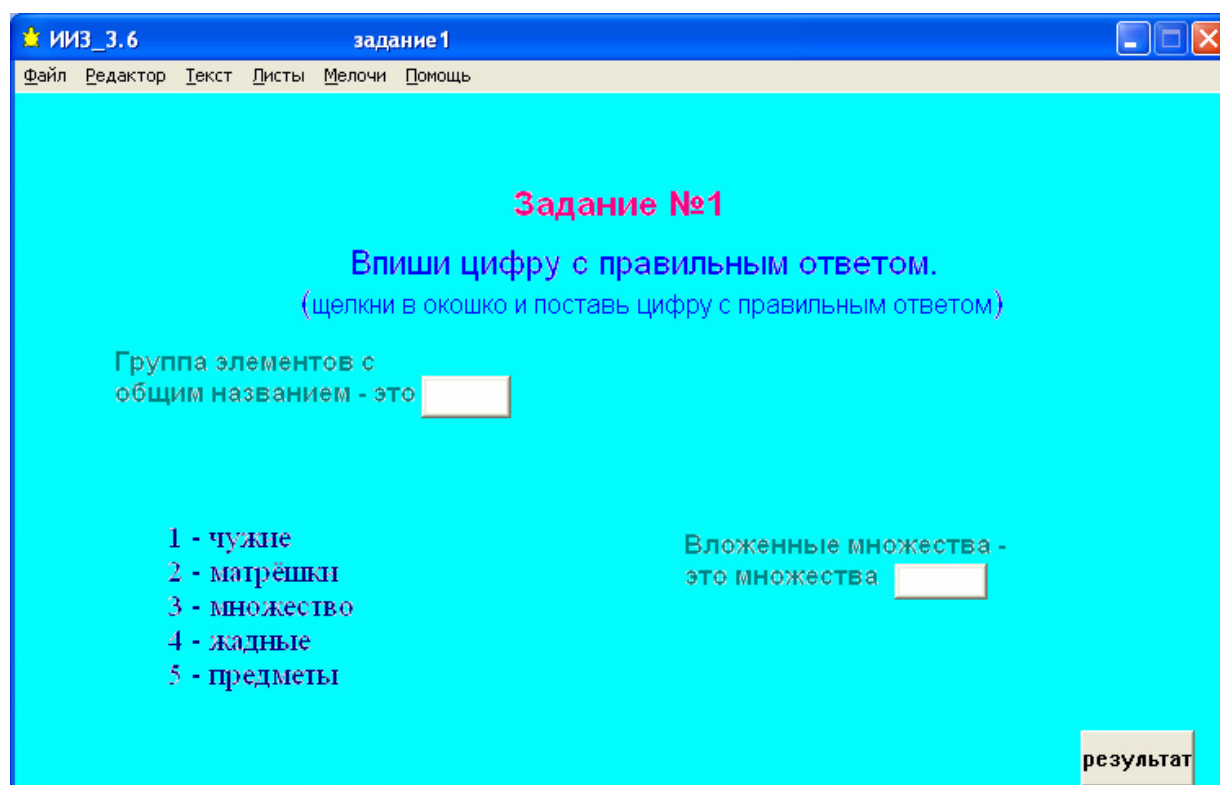


Figure 5. Assessment frame. The answer is put in the text window. After the answer has been put in the Result button is to be clicked

Let us thoroughly view the use of the turtle as the most "interactive" object.

1. In case the Turtle has been programmed the answer input may be realized by clicking on the Turtle (see Fig. 4).
2. The Turtle can be easily *moved* to the given area (see Fig. 2, 3), and its parametres may be measured with the use of specific sensors. Thus, for example, co-ordinates of the active Turtle may be measured with the use of specific sensors (*xcor*, *ycor*), and the colour of the marked its place position – with the sensor *colorunder*. The numbers (measurement results) may actually be viewed as children's answers which will be compared to pattern data of corresponding parameters.
3. The Turtle's most useful feature is a possibility to give it various forms – this is especially important for Elementary School. Thus, it is possible to attach to the Turtle not only shape of a picture, but also a number, a letter, a word. (see Fig. 3).

As shape an imported picture, created in the other programme (MS Word, Paint etc.) may be used.

The Logo language allows to realize the dialogue as a menu. For realization of menu in Microworlds the following objects may be used: buttons, programmed Turtles, and programmed colours.

The Right Answer Input Check-Up

Check up of the right answer is carried out by means of comparing answers with the pattern answers. Each answer is either clicking the specific object (chosen by a schoolchild) or co-ordinates of the Turtle having been moved by him or colour of the square, which a child had moved the Turtle on to, etc. For checking up the right answer programmed objects are often used. On a *Programme Page* necessary procedures are created, names of which are then written in instructions for these objects. The procedures include commands *if* and *ifelse*, which contain logical expressions with the sensors registering data on current parametres of the Turtle. The measurement results are compared to sample data (for example, colour number under the active turtle or its co-ordinates). In case a child's answer is put in by means of the Turtle's manipulation or text put in, for the programme start the use of a programmed object is required.

Reply on the Answer Input

Replying to the answer input the programme put a corresponding answer out. In education and training programmes in case the answer is wrong the mistake is analyzed, schoolchildren are recommended to revise the theory, solve a task again. (see Fig.6). In assessment programmes the credit number is displayed and then the next question is put out. Replies may contain both text and graphic objects, be accompanied with sounds. Replies output is carried out by various ways: into a signal window with the use of command "Inform" or into a text window (into a new one or cleared up from the old text one). All replies may be stored on a separate Page beforehand and be put out in when necessary.

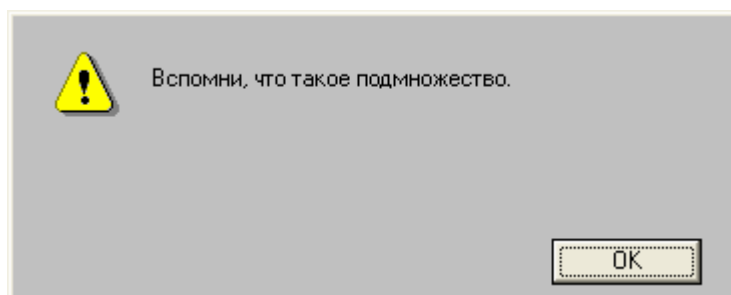


Figure 6. Reply on the given answer in a training programme. Is put out in the signal window

As mentioned above for a user, working with the programme it looks like a sequence of Frames following each other (images on the screen). One and the same Page with new information; the other one, stored beforehand or a signal window may be a Frame. The Frame change (change of the image on the screen) may be implemented by various means either by the user, or by the programme automatically. The programme opening from the required Page is carried out with the use of *startup* procedure.

Conclusions

At classes in the framework of the course «Information and communication technologies in Elementary education» students – would-be informatics teachers in 1st-6th years of Primary

Education Department work at development of individual projects. They realize the whole process of creating teaching and training programmes for Elementary School:

- analysis of a topic of Elementary School subject (Science, Mathematics, The Russian Language, Informatics)
- designing a programme
- a programme realization
- a programme expertise

As a result every student creates a scenario and a fragment of a teaching programme in each of the subjects of the Elementary school curriculum. Students analyze various resources of the Logo Language, assess them, and choose most appropriate ways of interaction realization, knowledge and skills control, use of the interface. ES for School has been elaborated as a project and is actually a programme in the Logo Language and a sequence of enumerated Frames linked to each other. Objects, graphic images, texts are created in the Frame, results of specific commands realization and complete programmes are displayed. The objects are programmed where necessary.

References

- Bashmakov A. & Bashmakov I. (2003), *Development of Computer Textbooks and Education Systems*, Moscow, 616.
- Boronenko T., Ryzhova N. (1999), *Methods of teaching of Computer Science (particular methods)*: Textbook. S-Petersburg, 89.
- Vitukhnovskaya A. (2003), *Basics of Programming in Logo Language (in MicroWorlds Environment): training aids*. Petrozavodsk: Karelian State Pedagogical University, 94.
- Vitukhnovskaya A. (2005), *Logo for the Would-be Teachers of the Computer Science Elementary Course*. EUROLOGO 2005. Proceedings of the Tenth European Logo Conference. Warsaw, august 2005. - Warsaw, 2005. P. 245 – 256.

Logo Nanoworlds

Michael Weigend, michael.weigend@uni-muenster.de

Institut für Didaktik der Mathematik und der Informatik, University of Münster, Germany

Abstract

Logo nanoworlds are small educational environments for Logo programming that focus on the mediation of intuitive models of informatics concepts. These are coherent Gestalt-like mental concepts, which are based on experience and which are subjectively certain (Fischbein 1987). Typical features of Logo nanoworlds are

- Visualisation of activity defined by a Logo programme
- Only a few Logo syntax elements are applied in each nanoworld
- Simple ad hoc understandable interactive elements
- Focus on certain intuitive models in each nanoworld
- Diversity of different intuitions in different nanoworlds
- Explication gaps
- High speed: a visit to a nanoworld takes just a couple of minutes

Two prototype nanoworlds are presented, which both adopt a delegation model for procedure calls. In *Turtles on the Beach* the player constructs (possibly recursive) procedures that produce graphical patterns. The editing is done by manipulating “puzzle pieces” with code fragments on the screen. In contrast to conventional Logo Turtle environments the programme is executed by a team of Turtles. Each Turtle on the screen is attached to one procedure call. This model is supposed to facilitate the comprehension of recursive programmes. The second example is called *Turtles in the Printery* and is about recursive procedures containing print statements.

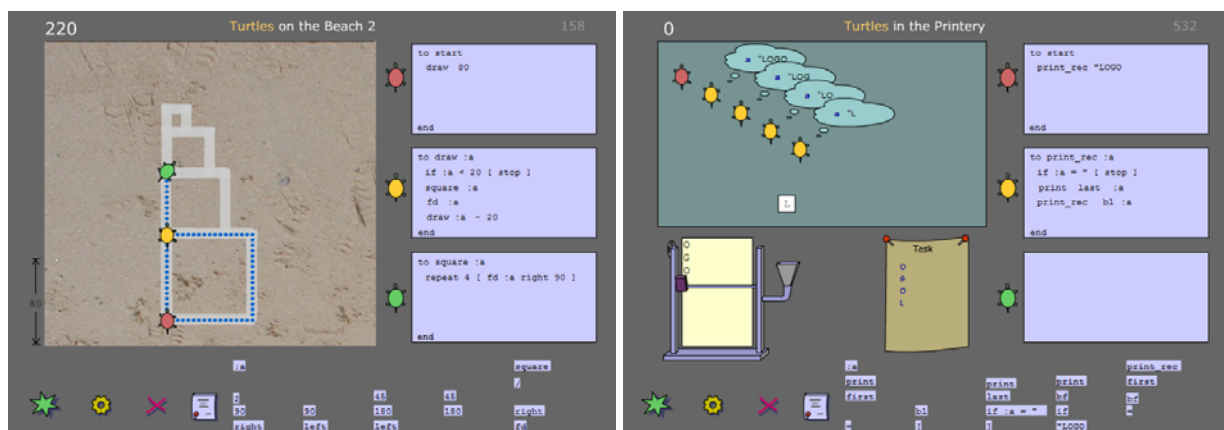


Figure 1. Screenshots from the nanoworlds "Turtles on the Beach" and "Turtles in the Printery"

Visiting a nanoworld is comparable to exploring an exhibit in a scientific museum or an art gallery. Classroom activities around nanoworlds include observing details of programme visualizations, comparing different intuitive models, bridging explication gaps and finding connections between visual representations and programming constructs.

Keywords

turtle; Logo; nanoworld; microworld; intuition; visualization; mental model; procedure

Intuitive Models and Programming - from Microworlds to Nanoworlds

When we try to understand the semantics of a computer programme or explain it to someone else we use intuitive models. According to Fischbein (1987) these are coherent Gestalt-like mental concepts about the world, which are based on experience. Intuitive models are self evident and are subjectively accepted as certain. Well known examples in the field of informatics are

- a box containing a value modelling a variable (container model),
- a factory accepting data and producing new data modelling a function,
- a railroad track with a switch modelling the "flow of control" during the execution of an if-statement.

Since representations of intuitive models are from different domains to the modelled target, they can be regarded as metaphors. While the term "intuitive model" refers to some rather abstract mental concept (which is not directly observable), the term "metaphor" focuses on the (observable) linguistic phenomenon of communicating ideas. There exist different metaphors (verbal descriptions, static pictures or movies) representing the same intuitive model. Sometimes they are invented ad hoc during a discussion and forgotten later, when the transported concept has been understood. Thus metaphors are somewhat volatile but intuitive models are not. They are persistent (Fischbein 1987, diSessa 2001), which means we never forget them, even if we have learned better, more scientific concepts meanwhile. Intuitions might be subconscious but still exert some tacit influence on our thinking and perception of the world. Sometimes they lead to misconceptions (see for instance Bonar & Soloway 1985) - thus it is necessary to know their limits of application. Intuitive models are rooted in domains of experience. The container model for variables is intuitive, because we are familiar with storing things in labelled boxes, drawers, envelopes and all kind of containers.

Among the intuitive models, which are adopted by the traditional Logo Turtle (Papert 1980) are

- the concept of a soldier, obeying commands without questioning and doing nothing unpredictable,
- the concept of bipedal walking in discrete units of length (steps),
- the concept of rotation around an axis.

Note that the Logo Turtle is not a model of a real turtle but an artefact combining different intuitive models of movement from different domains of experience. The step concept (adopted in the forward command) only makes sense in a bipedal world. For a creature with four legs a single step of one leg does not necessarily mean movement of the whole body. Except on parade grounds living things change the direction of movement by making a bend instead of turning on the spot, a concept adopted in the right command. But this kind of motion can be observed, when you turn your head, open a faucet or move a camera on a tripod.

The Logo Turtle is a virtual entity acting in a virtual world, thus forming a microworld, which is nowadays a multimedia world with sound and colourful pictures since Logo Computer Systems Inc. (LCSI) released MicroWorlds in 1993 (www.microworlds.com). In a more general sense a microworld is a learning environment that consists of a carefully designed coherent set of elements and rules that determine their behaviour (Schulmeister 2002). The major aim is to provide a framework for "creative exploration", which means to create new or reconstruct existing systems with the given expressive means of the microworld (see for example Stager & Einhorn 2003). In Logo-based microworlds intuitive models related to programming are used for construction and not really reflected upon or questioned. It is assumed that the player discovers the meaning of language constructs and programming principles just by using them and watching the results on the screen. Consider a Lego construction kit, which can be regarded as a non-computer-based microworld. When children build bridges and houses with Lego bricks, they just use the material to carry out their ideas. In this process they usually do not care much

about the bricks as such, the chemical composition of the plastic or the phenomenon that the bricks stick together so tight. To find out more about the bricks they would need a different environment, a science lab, where they could try to burn bricks, treat them with acid or measure the friction between two bricks using a dynamometer.

The "bricks" of a Logo microworld are basically the syntactical elements of the programming language. When the aim is to understand the underlying informatic concepts (like recursion, procedure call, parameter passing and so on) in depth there is some need for special learning environments which are like science labs for bricks.

I call this type of environment nanoworld, since it is much smaller than a constructive microworld. Each nanoworld does not cover the whole Logo syntax but focuses on just a few language elements and some intuitive models of programming principles connected to them. Whereas one and the same microworld may be useful for months or even years of constructing and learning, students should visit many different nanoworlds, each one for a couple of minutes, thus getting in touch with a wide range of intuitive models from different domains of experience.

Visiting a nanoworld is comparable to exploring an exhibit in a scientific museum or an art gallery. Visitors spend just a few minutes with an exhibit. They do not need long explanations about how to use the presented artefacts. Interactive components – if there are any – are very simple and understandable in seconds. Visitors want to get inspired, be surprised, emotionally touched and excited, grasp some new ideas. Typical activities are rather to observe and to discover things than to construct something useful. But what does this really mean? Let us steal some ideas from a real life gallery, the Ludwig Museum of Modern Art in Cologne, Germany. Like most institutions of that kind, it offers guided tours for children. A presentation of the sculpture "Nana", made by the French-American artist Niki de Saint Phalle depicting a big Afro-American woman dancing, includes activities like this: At the beginning the children describe details that are eye-catching, strange and different. Then they try to pose like the sculpture and say how they feel, when they are in this posture. This way they gain some understanding of the visual expressive means the artist has used.

The Traditional Logo Turtle and Intuitive Understanding of Recursive Procedure Calls

This section is about a shortcoming of the Logo Turtle regarding its power to explain recursive procedure calls. Procedures are special functions that do not return any data respectively return an empty object. A main feature of the Logo Turtle is that it supports the mental model of one actor executing a programme. The Turtle is the one and only active entity in the system. It listens to commands and executes them. Some of the operative knowledge – like the capacity of moving forward a certain number of steps – is inherited. During its lifetime a Turtle can learn additional commands. Thus in the dramatic framework of the Turtle microworld, programming is considered as some kind of teaching knowledge. Contemporary environments like MicroWorlds (LCSI) may allow several Turtles to run parallel. Nevertheless, in each thread it is one Turtle which is responsible for the execution of the whole programme text.

Logo programming based on Turtle graphics has been used for several educational purposes including the teaching of recursion. The educational value is that students see the effect of their programme *at once* and can check its logical correctness by comparing the picture drawn with the expected result.

Good graphical tasks for recursive Turtle procedures are carefully designed so that the picture the turtle has to draw has an "obvious recursive structure" like the arrangement of squares in figure 1. Thus the learner can adopt preconceptions about recursion – like periodicity, self-similarity or gradualism (Levy & Lapidot 2001) – and find connections to formal descriptions of movement. In that way – but only in that way – the understanding of recursive algorithms is supported. There is no visual component that focuses on the execution of recursive procedures

itself. The internal mechanics remain invisible. Students have to find a mental representation by themselves. The system offers no help in this respect.

Figure 1 is a screenshot from a session with Microworlds EX (LSCI). The recursive procedure `to draw` is part of the knowledge of the one and only one turtle on the field. Within the dramatic framework of Microworlds EX this knowledge is stored in its backpack. The procedure is an example of end recursion; the recursive call is the last statement in the body.

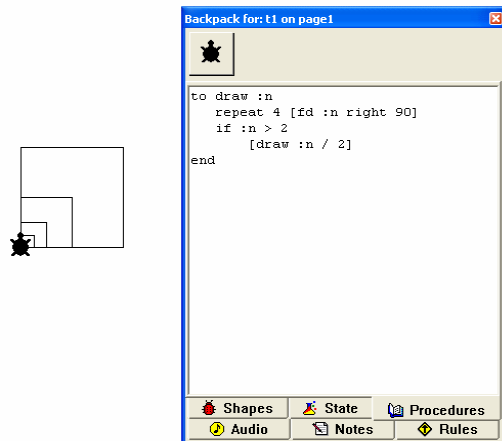


Figure 1. Screenshot from a session with Microworld EX. Working area with turtle t_1 and its backpack containing a recursive procedure.

End recursive procedures can efficiently be interpreted using the concept of repetition. In this case the last statement is read “do the same thing again but use a smaller number”. This concept is easily understood because it matches the (tacit) assumption of just one actor.

From every day life, we are familiar with doing things again but in a slightly different way. Imagine Tina solving a mathematical problem and calculating a number. At the end she checks whether the result is correct. If this is not the case she assumes that she made a mistake and starts again, but this time she is a bit more careful. Note that the test and the possible self-request at the end are part of the solving procedure. But the phrase “do it again” implies that there is some coherent activity, which already has been completed and thus can be repeated.

The intuitive concept of self-request is only appropriate for end-recursive procedures. Here the sequence of commands, which can be thought of as one holistic activity associated to the procedure, has already been executed before the recursive call takes place.

If the recursive call is not at the end of the procedure body but in the middle (embedded recursion), this model leads to erroneous interpretations. In this case the one and only active entity assumed has to store its current state. When the activity caused by the recursive call has been finished, this entity restores the state before the call and executes the next statements. But the storage of states is not part of the classical Turtle world and is therefore not visualized in any way. Moreover storing and restoring a state is something quite unnatural. Since human beings can move in time only in one direction from past to future there is no way to re-gain a state one has been in at some time in the past. Therefore there is no direct anthropomorphic representation of such an event.

Dicheva & Close (1996) asked students to predict the output of recursive Logo procedures.

Logo procedure	correct solution	incorrect prediction
<pre> to pattern3 :w if empty? :w [stop] print :w pattern3 bf :w print :w end </pre>	<pre> LEGO EGO GO O O GO EGO LEGO </pre>	<pre> LEGO EGO GO O O </pre>

Table 1. Recursive Logo procedure, correct and wrong expected screen output after the initial call `pattern3 "LEGO` (Dicheva & Close 1996)

Table 1 shows a procedure with embedded recursive call, the printed output after the initial call

```
pattern3 "LEGO
```

(correct solution) and a wrong prediction. Assuming a one-actor-self-request model, the wrong answer (right column) can be explained the following way: First the Turtle prints the string LEGO on the screen. At the recursive call, the execution of the command sequence is cancelled and started again (same actor). But this time the Turtle processes the original string without the first letter (EGO). This goes on until the string consists of only one letter. Then the recursive call

```
pattern3 bf :w
```

is skipped (because it has no effect) and the final print-statement is executed once, which outputs the letter o a second time.

Delegation Model

The delegation model emphasizes the difference between the definition and the call of a procedure. The definition of a procedure is knowledge how to do something. The Logo syntax supports this concept: A procedure definition starts with the key word `to` followed by an identifier that should be a verb (e.g. `to draw`). This suggests that a procedure definition is some kind of explanation of the meaning of a verb.

The second notion is that of an active entity. When we say “procedure A calls procedure B” we assume that procedure A and B are interacting entities. They are in a certain state in each moment of their existence and they can do things. Calling a procedure is regarded as a social phenomenon. It is seen as delegating a task to another entity which is able to execute the required procedure. This entity might already exist (somewhere out of sight waiting for a call) or is generated at the moment of the call. During a programme run there may be many active entities executing the same procedure but each of them is in a different state. When actor A calls a new actor B it waits until B has finished its activity and then continues.

Delegation of tasks is a well known phenomenon in everyday life. If the sink in the kitchen is broken we call a plumber to fix it. He is a trained specialist and uses his knowledge to solve the task. Usually there is some hierarchy of responsibility. At the top there is some entity responsible for the whole task. In our example this might be Jim, the owner of the apartment, where the sink problem occurred. He calls a plumber who himself might delegate some tasks to his assistant and use materials produced by different persons. Thus in the end, many actors are involved in the solution of this task. Still Jim feels responsible for the whole process and later tells his girlfriend that he had to fix the sink in the kitchen.

The delegation model is appropriate to explain recursion. A recursive procedure call can be modelled the same way as any other (non-recursive) procedure call. The calling entity A delegates a task to a new entity B, which just happens to be of the same kind as A. Instead of storing its state the calling entity just waits and remains in the same state until the called entity has finished its work. The concept of waiting is much more familiar and mentally easier to handle than the concept of storing and restoring.

It should be mentioned that the delegation model does not fit the paradigm of object-oriented programming (OOP). In an OOP world objects are the only active entities. They can execute more than one operation (method) and a function call is interpreted as a message to an existing object which has been generated before.

In the following sections two prototype nanoworlds are presented, that focus on the delegation model of procedure calls. They are published in the WWW and accessible for everybody (www.creative-informatics.de/tp).

Example 1: Turtles on the Beach

In *Turtles on the Beach* there are a maximum of three types of turtles with different colors (red, yellow, green). Each turtle can execute just one procedure. While walking on the beach the turtles put blue stones in the sand. The task is to lay a track of stones along a given way. The player has to complete procedure definitions on the right-hand side. At the bottom of the application window there are small fragments of Logo code that can be moved with the mouse into the editor fields. This “puzzle”-technique reduces the probability of syntax errors, which usually take much time to debug. Novices make many programming errors that result from slips rather than from misconception (Anderson & Jeffries 1985). Therefore the time-consuming debugging of this kind of error does not have much educational value and should be avoided if possible. When the editing of the procedures is finished, the player can drag a red turtle onto some place on the working area (the “beach”) and start the execution of the programme. When the player is satisfied with the result, she or he clicks on the diploma-icon. The system checks the solution and adds some points to the score. After clicking on the “New”-button a graphical pattern (consisting of thick grey lines) appears on the beach, defining a new task. The first task is for free, but each of the following tasks costs a certain number of points (according to the level of difficulty). When it is solved correctly the player gets twice the number of points attached to the task. It is possible to skip tasks but this means the loss of points. The player has got 10 minutes to get as many points as possible.

The difference to the usual Turtles microworld is that the activity is distributed among several entities. The red turtle is responsible for the solution of the whole task and starts the execution of its procedure. Every time, when one of the self defined procedures is called, a new turtle is generated and put on exactly the place with exactly the same rotation as the calling turtle. The new turtle is a helper, solving some subtask. When it has finished its job, it takes the calling turtle to its current position and vanishes. So instead of one turtle we have a team of interacting turtles. Since a procedure can be called recursively several times there may be several turtles of one type (colour) on the beach.

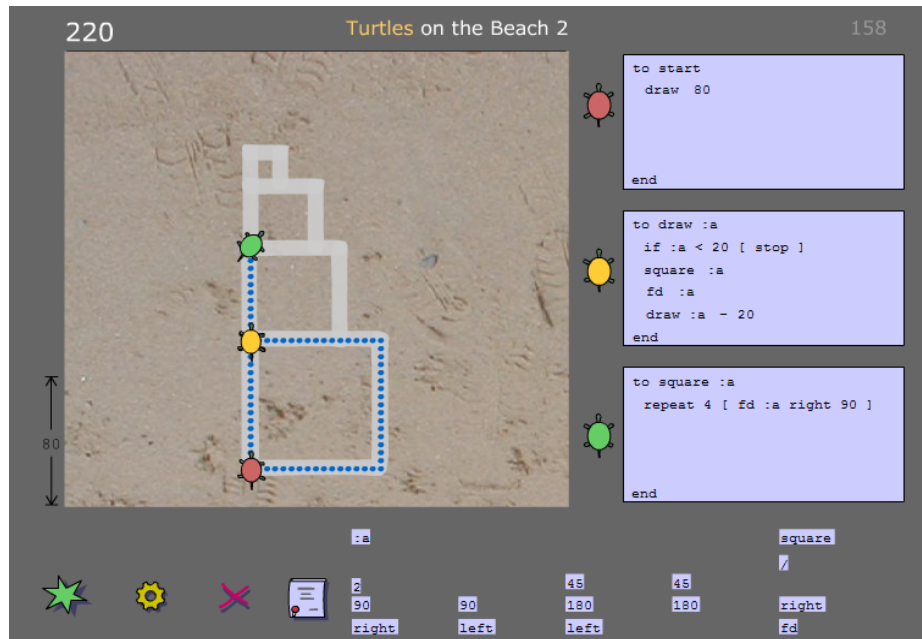


Figure 2. Screenshot from *Turtles on the Beach 2* (Weigend 2007)

Example 2: Turtles in the Printery

In the nanoworld *Turtles in the Printery* (see fig.3), the player has to programme a recursive procedure that prints a text which is given on a piece of paper with the headline “Task”. A red turtle (executing the procedure `to start`) can be dragged onto the working field and takes a position in top left-hand corner when the mouse button is released. Each time the procedure `to print_rec` is called, a new yellow turtle appears on the working field with a thinking bubble (like in a comic) containing the value of the argument. When a turtle executes a print-command, a card with a string written on it moves from the active turtle to the printer which processes this string and prints it in the next line.

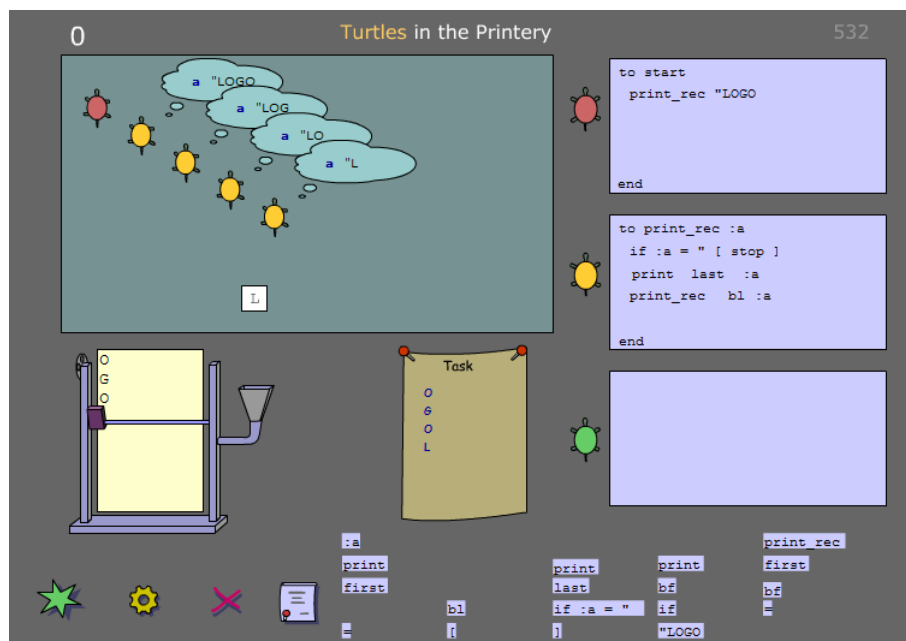


Figure 3. Screenshot from the nanoworld “*Turtles in the printery*” (Weigend 2007).

There are three different kinds of actors in this nanoworld representing three different programming concepts. The turtles are volatile and each one represents the execution of a single procedure call. In contrast, the printer is a persistent entity that does not vanish after having finished a task. It has all the qualities of an object as in the OOP paradigm. The printer accepts messages and processes them and it has exclusive access to the sheet on which the output is printed. Only the printer knows the current position of the printing head. The third type of active entities are the cards, which are sent by the turtles initiate printing. Each card finds its way to the printer thus representing the addressing and messaging aspect.

Explication Gaps, Consistency and the Gestalt of intuitive Models

“Question: Why do most civil servants wear glasses? Answer: So that they don’t stick their pencils into their eyes when they’re falling asleep.” In Germany at least, where most people think that civil servants really do not work hard, everybody understands the joke. Just imagine the awful scene: Somebody who looks really bored and tired is sitting at his desk with a pencil in his hand, scribbling something in a file, getting slower and slower. Finally he drops off and his head falls onto the desk top ...

The joke is funny because it does *not* explicate the possibility of getting hurt by sleeping during work. Explication gaps in verbal or visual descriptions are very common in human communication. They make a story or explanation more interesting and increase the dynamism by leaving out the obvious. It is virtually impossible to describe a real life process completely because there is no end to all the details that could be taken into consideration.

But computer programs are complete descriptions of processes. One of the difficulties novice computer programmers have to cope with, is to fill the explication gaps that are left by informal descriptions of algorithms. Bonar & Solway (1985) observed that novices tend to use a programming language like a natural language and fail to explicate certain things.

One way of coping with the difficulty of understanding programme text is visual tracing systems like the Jeliot family (Moreno & Myller 2003). They use a consistent set of metaphors (e.g. pointers, boxes) to represent variables and execution frames of function calls and show all the changes that take place within these components during a programme run.

Automatic programme visualizations of this kind are useful for debugging and similar purposes. Complete information about the system is essential for debugging because the programmer is searching for a logical error which is “hidden” somewhere and which can only be detected for sure, if access to *all* internal system changes is guaranteed. But a complete visualization is very complex, confusing and exhausting to follow. It does not provide an holistic, intuitive model of the idea of the programme. It can be *used* for explanations in some teaching processes but (because of its complexity) it is not an explanation in itself. In this respect nanoworlds are a bit different since they provide a reduced, simplified view into the system. They focus on certain aspects and ignore others. For beginners, who have just started to understand the basic principles of programming, this might be an advantage. Visiting a nanoworld includes filling explication gaps by imagination and reflecting simplifications inherent in intuitive explanations.

Compared to a system like MicroWorld EX the two nanoworlds presented might provide a richer visualization of the *internal* machinery of a running programme. But there are explication gaps. For instance

- the passing of arguments in a procedure call,
- the return of control to the calling turtle, after a turtle has finished its job,
- the evaluation of terms

are not visualized.

In Turtles in the Printery the argument of the procedure call is shown in a thinking bubble. This makes it easier to understand the computation of the printed strings. In Turtles on the beach

there are no thinking bubbles. This explication is not necessary because the arguments are documented permanently in the length of the dotted lines which the turtle produces. This example illustrates that the necessary degree of explication depends on the kind of programme to be explained.

Nanoworlds are products of creative topic-centred didactical design. To illustrate the difficulty of *automatizing* programme explanation, let me refer to some earlier research. In 2006 I asked 16 German informatics students to evaluate different animations visualizing the in-place-sorting of an array of numbers applying the straight selection algorithm. They were shown the following programme in Python syntax and had to decide which animation they would use to explain its execution to someone else.

```
s = [10, 4, 1, 3]
for i in range(len(s)):
    for j in range(i+1, len(s)):
        if s[j] < s[i]:
            s[i], s[j] = s[j], s[i]
```

Figure 4 shows screenshots from the animations. The majority (9 out of 16) preferred the fourth model. In contrast to the other visualizations it does not explicitly mention the two variables *i* and *j* which are indices of the actual two elements that are to be compared. These two elements are indicated as partly pulled out of the box. On the other hand this successful model contains an additional fantasy element, which is not directly represented in the programme and therefore could not be generated by a visualization tool: The part of the array, which is already sorted, is covered by a glass box labelled with “OK”. During the sorting process the sorted section grows until at the end the whole array is under a glass box and sorted.

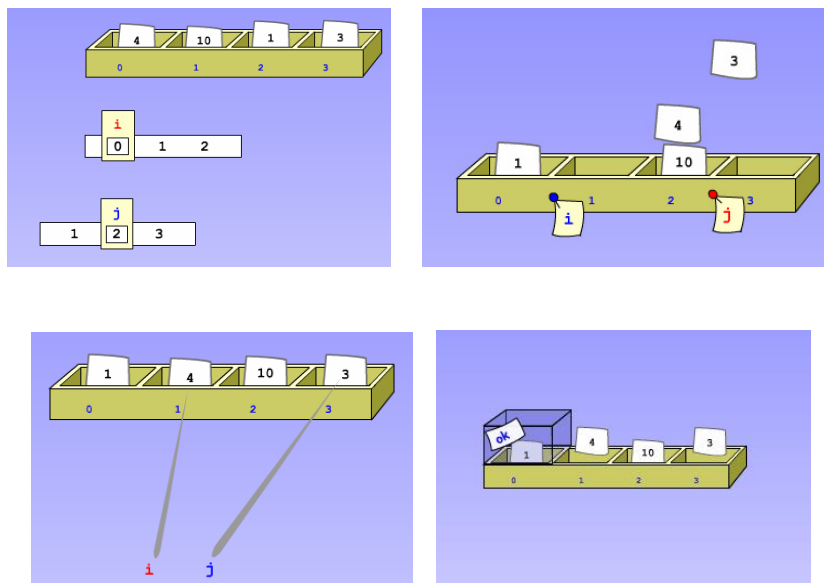


Figure 4. Screenshots from animations visualizing the straight selection sorting algorithm.

The price for greater richness and explaining power is a lack of consistency between different nanoworlds related to the same topic. In software development environments (including microworlds) that are used for extensive constructive activity (programming something really big and complex), consistency is essential, because it increases the efficiency of usage. The user has to learn just a few concepts and can apply them again and again. But nanoworlds are designed to provide the opportunity for basic insights into programming *as such*. The differences between nanoworlds may serve as a stimulus for reflecting the underlying concepts. Additionally one can observe inconsistencies within one and the same nanoworld (like different types of actors instead of just one type). They are sometimes necessary to provide a coherent Gestalt.

How to use Logo nanoworlds for Teaching

Nanoworlds facilitate experience with different intuitive concepts. These should be reflected on or at least the students should be made aware of them. Thus, when a nanoworld is used in an informatics class, it might be a good idea to talk about some issues after a period of individual play. Questions to be discussed may relate to

- simple observations ("What happens to a turtle when it has finished its job?"),
- comparison of different concepts ("What is the difference between the turtles in *Turtles on the Beach* and the turtle in your usual Logo environment?"),
- explication gaps ("How does a turtle in *Turtles on the Beach* know the size of the square that it draws?" or "How does a red turtle know that a yellow turtle has finished its job so that it can continue?"),
- connections between visual elements and programme constructs ("What are the cards that move to the printer in *Turtles in the Printery*?")

Logo nanoworlds visualize mental concepts that are useful for understanding and creating programmes. A way of elaborating these concepts is to use the dramatic framework of a nanoworld (including explication gaps) for a role play that simulates the execution of a logo programme. The following example should be played on a sunny day outside on the school yard, which should be asphalt so one can see wet spots. You need a bucket filled with water, a sponge and for each student a sheet with some simple Logo procedure like in *Turtles on the Beach*. One player – say Tom – is given the initial task (procedure call), and receives the bucket and sponge. When moving forward a certain number of steps a player squeezes the wet sponge to produce a track on the ground. To execute a (possibly recursive) procedure call, Tom selects another player – say Jenny. Jenny positions herself on exactly the spot, where Tom (the calling entity) was standing a moment before. Tom passes the bucket and the sponge to Jenny and tells her the argument (a number of steps). Now Jenny moves according to the procedure, which she is assigned to, and eventually engages other players. When she has reached the end statement, she asks Tom to come to her and returns bucket and sponge to him.

Note that during the role play explication gaps of the nanoworld have to be bridged. For the transfer of control (bucket and sponge) to another entity and the passing of arguments the students have to invent appropriate activities themselves.

References

- Anderson, John R.; Jeffries, Robin (1985) *Novice LISP Errors: Undetected Losses of Information from Working Memory*. In *Human-Computer Interaction*, 1, pp. 107–131.
- Bonar, Jeffrey; Soloway, Elliot (1985) *Preprogramming Knowledge: A Major Source of Misconceptions in Novice Programmers*. In *Human-Computer Interaction*, 1(2), pp. 133–161.
- Close, John; Dicheva, Darina (1997) *Misconceptions in Recursion: Diagnostic Teaching*. In *Proceedings of the Sixth Eurologo Conference "Learning and Exploring with Logo"*, pp. 132–140.
- Dicheva, Darina; Close, John (1996) *Mental Models of Recursion*. In *Journal of Educational Computing Research*, 14(1), pp. 1–23.
- diSessa, Andrea A.: *Changing Minds. Computers, Learning, and Literacy*. Cambridge, Massachusetts (MIT Press) 2001.
- Fischbein, Efraim (1987) *Intuition in Science and Mathematics*. Dordrecht Boston Lancaster Tokio (Reidel).
- Levy, Dalit; Lapidot, Tami; Paz, Tamar (2001) „It's just like the whole picture, but smaller“: *Expressions of gradualism, self-similarity, and other pre-conceptions while classifying recursive phenomena*. In *Proceedings of the PPIG*, Bournemouth UK, April 2001. Edited by G. Kadoda, pp. 249–262.

Moreno, Andrés and Myller, Niko (2003) *Producing an Educationally Effective and Usable Tool for Learning, The Case of the Jeliot Family*. In Proceedings of the International Conference on Networked e-learning for European Universities, Granada, Spain.

Papert, Seymour (1980): *Teaching Children Thinking*. In *The Computer in School: Tutor, Tool, Tutee*. Edited by R. Taylor, New York (College Press), pp. 161–176.

Schulmeister, Rolf (2002) *Grundlagen hypermedialer Lernsysteme*. Oldenbourg Verlag, Munich.

Stager, Gary and Einhorn, Susan (2003) *Exploring with MicroWorlds EX*. LCSl, http://www.lcsi.ca/pdf/microworldsex/exploring_projects.pdf

Weigend, Michael (2005) *Intuitive Modelle in der Informatik*. In Proceedings of INFOS 2005. Edited by Steffen Friedrich, pp. 275–284.

Weigend, Michael (2007) *Gallery of Logo Nanoworlds*. URL: www.creative-informatics.de/tp