Logo in Australia
Ten Years On

Editor: Anne McDougall

Computing in Education
Group of Victoria

Logo in Australia

Ten Years On

Papers from the Conference,
Logo in Australia: Ten Years On,
held at Normanby House,
Clayton, Victoria,
by the
Computer Education Group of
Victoria,
in August 1985.

Editor: Anne McDougall

Published by

Computing in Education
Group of Victoria

Statewide Resources Centre
217 Church Street
Richmond. Vic. 3121

National Library of Australia
Card Number and ISBN 0 9586879 2 7

Printed by Publicity Works

Logo in Australia: Ten Years On

Preface

This collection of papers is selected from those presented at the conference Logo in Australia: Ten Years On, held at Normanby House at Monash University in Clayton, Victoria, under the auspices of the Computer Education Group of Victoria (as it was then named), in August, 1985. The conference, as its title implies, was held to mark ten years of Logo work in Australia - ten years since the initial importing of a magnetic tape version of Logo for the PDP-11 minicomputer by Scott Brownell of the Education Department of Tasmania, and the pioneering work of Sandra Wills with Logo in Tasmanian primary schools.

For various reasons not all of the papers from the conference have been included here. The conference participants were provided at the time with a photocopied set of the papers. Since there was not a published Proceedings as such, a number of the presenters subsequently submitted their papers to journals and elsewhere for formal publication. In some cases I have not been able to obtain copyright clearance for this material in time for the printing deadlines for this volume. In a couple of others, where authors have changed jobs and/or moved interstate I have been unable to trace them to obtain information about the copyright status of their papers. I wish to thank Sandra Wills, Tony Adams, David Squires, Royston Sellman, Carolyn Dowling, Valerie Clarke, Tony Jones, Peter Carter, Lesley Tan, Pauline Adams, Sue Chambers and Liddy Nevile for permission to include papers in this volume.

Why would the CEGV and OzLogo publish, eleven years later, papers from a conference held in 1985? Firstly, 1996 marks twenty-one years, the "coming of age", of Logo in Australia; it seems an appropriate time to look back, to see how far Logo work here has developed in that time, to take stock, and to plan ahead building on the achievements of previous work. I have some concern that in these times when, for many people and institutions, it seems important to be "the first" with the latest innovation, earlier work in areas of educational computing is almost deliberately ignored in order to claim pioneer status in whatever the latest technique or

development is. As a result, many of our endeavours become re-inventions of wheels already at least partly perfected by others, superficial or repetitive due to our failure to build on previous work.

Secondly, I find that many of these papers still provide "a good read", and address many issues that are as important today as they were in 1985. Yes, they have dated in some details. Those of us who remember BASIC hard-wired into every desktop computer smile as we read Peter Carter's severe references to students' work showing "traces of thinking from that other language" with "procedures 50 - 60 lines long". And I think that today words like "gender stereotyped" have replaced the "sex stereotyped" of the eighties. But how important it is for us to understand, from papers such as Tony Adams' or that of David Squires and Royston Sellman, the "small-w" microworld ideas that preceded the naming of the latest version of Logo: MicroWorlds! I was struck, on re-reading the Squires and Sellman paper, by how many of the features we now see in StarLogo they might have developed had they had access to a sufficiently powerful development environment. Carolyn Dowling's ideas about examining computer languages as a stimulus for reflection about natural language are as interesting and useful today as they were when she wrote her paper. Tony Jones and Peter Carter describe two quite different but successful ways of addressing pre-service teacher education needs; few would claim that problems in this area are adequately solved even today. And so I could continue.

I commend to you the writings of these "ten years on" pioneers of Logo work in Australia. I hope they will prove a useful resource for researchers and practitioners alike.

Anne McDougall
September, 1996

Contents

Doodle Design Debug: Process vs Content Issues in Classroom Computing

Sandra Wills

A version of a paper originally presented to the Second Annual NSW Computers in Primary Schools Conference, Macquarie University, September 1983.

Unfortunately, current classroom computer usage tends to reflect and reinforce traditional teaching methods rather than harnessing the full power of the computer to enable students and teachers to learn in ways that were not previously possible. This paper examines three more recent examples of computer usage that go beyond drill and practice, showing a concern for the *process* of learning not just the *content.*

Student use of the computer as a tool for word processing, data base enquiry, and Logo projects is analysed with particular emphasis on the problem-solving processes of doodling, designing and especially debugging.

A quote from the British Microelectronics in Education Project: "HARDWARE without SOFTWARE is just JUNK but SOFTWARE without good TEACHING is just NOISE!"

The important ingredient for success in computer education is how we as *teachers* integrate computing into the curriculum. It is not our role to become programmers or hardware geniuses. Our role is to evaluate computing from an educational stance, not only with regard to the educational value of the *content* of the software but also with regard to the value of the *process*of teaching and learning the software employs.

The *how* of learning is as important as the *what.* The NSW Education Department document on the "Aims of Primary Education"

reinforced this process vs content approach when it put forward *communicating, expressing* and *investigating* as major aims. The introduction of computing in schools does not change these aims. It should complement them.

Table I provides one perspective on how computers can be used in schools. For the purposes of this paper I wish to compare and contrast "learning *from* computers" and "learning *with* computers".

In this introductory phase that most schools are going through, the two major uses of computers appear to be BASIC programming and simple computer assisted instruction (CAI), and particularly CAI in the primary schools.

With CAI the machine is used to teach the student some topic such as Mathematics, English, Geography. I'm sure you have all seen examples of the CAI that represents the worst aspects of our teaching methods forcing the student to try guessing that one pre-ordained answer the teacher/ machine is expecting rather than encouraging them to think, reason, and solve problems for themselves.

CAI concerns itself with the *content* of a subject, transferring the teacher/author's knowledge to the student. Although CAI has many advantages such as immediate feedback, individualised rates of learning, and perhaps automatic marking and record-keeping, the *process* of learning is really no different from what happens in normal classrooms. It automates what could be done with a one-to-one teacher-student ratio.

Since one-to-one is an impossible dream and since most teachers would be quickly bored if their talents were only employed in a drill approach to teaching, then CAI certainly has a place. However it is not the whole story. Just as drill under-utilises a teacher's talents, it also under-utilises a computer's power. And it under-estimates the capability of our students, relegating them to a role similar to rats in a Skinnerian maze.

Seymour Papert on CAI

CAI was perceived as modelling a good teacher (e.g. patience, feedback, drill) but then we always perceive new technology in terms of the old. For example automobiles were first known as horseless carriages.

Table 1

HOW CAN WE USE COMPUTERS IN SCHOOLS?

A. SUPPORT SCHOOL ADMINISTRATION
- library
- office
- inter-regional communication

B. SUPPORT STUDENT LEARNING

learning ABOUT computers
- computer awareness
- computer science
- commerce and business studies

learning FROM computers

computer assisted instruction (CAI)
- drill and practice
- tutorial learning WITH computers
- word processing
- simulations
- data base building and enquiry
- computer aided drafting
- music making
- electronic blackboard
- spreadsheet packages
- survey analysis
- logo as a tool

CAI is merely using "bright new gadgets to teach the same old stuff in thinly disguised versions of the same old way".

Jacques Hebenstreit on CAI

There is generally a lack of freedom for students in CAI programs. For example, unlike a text book, they cannot browse, skip sections at their own discretion, or correct previous answers at a later time should their understanding become clearer.

The programs are "cleverly" written by the author/teacher to give the computer a "personality", which thus disguises the important role of people in the teaching process and furthermore falsely poses the computer as all-knowing and intelligent, contrary to the goals of computer awareness.

Thirdly, if CAI is supposed to model the "good teacher" then it also *replaces* the teacher, surely not a politic use of computers in the current age of teacher surplus.

Isaac Asimov on CAI

"Any teacher who *can* be replaced by a computer *should* be!" If you're teaching like a computer then you deserve to be replaced by one. Teaching is more than drill and practice, therefore educational computing should also be beyond drill and practice.

Doodle, Design, Debug

Beyond drill and practice is using the computer as a tool to help you do things that were not possible manually, and to aid you intellectually. It is not so much the content of what you are learning but how you do it. Perhaps the computer could enable us to learn in ways that were not possible before. Perhaps it enables us to explore the processes of problem-solving, that is, those skills which may be transferable between content areas.

Using the *process* perspective, let's analyse three examples. I believe that each can be used to highlight to the learner three important stages of learning and problem-solving. I've chosen to call these three stages: Doodling, Designing and Debugging.

Word Processing

A word processor harnesses the computer's memory capacity to enable us to store the words we write and to easily modify them without rewriting all the words again. We are "released" from the mechanics and tedium of hand-writing and are free to experiment more readily than we might otherwise be inclined if faced with the boring prospect of rewriting huge chunks of otherwise unchanged text.

Experimenting is Doodling. Free to doodle, without feeling that we are wasting precious time (and paper), any of our "doodles" can be correctly recalled without rewriting if we decide that the doodle might perhaps form the basis of a large piece of writing.

Designing. After doodling or instead of doodling, you might have a design or plan for what you want to write. Usually this takes the form of headings and sub-headings with which you intend to structure your piece of writing. With a word processor, you are

free to type up this structure and then insert your writing under any of the headings in whichever order your train of thought takes you. Word processing has been described as "power typing" but it's only that to a typist; to a writer, it's "power writing". At any time in your designing, you can revert to doodling, to experiment with order of paragraphs or style, or calling up previous doodles to investigate how they fit in. Put it all down as it flows then go back to tidy it up later.

That is Debugging. All Designs need modification as they develop past the original specification. Experienced writers know that writing involves constant revision and improvement. This process is emphasised in Donald Graves' process writing model, which is now feasible to put into practice with the availability of word processors. Good writing involves re-reading and thinking in order to iron out the "bugs". Word processing enables you to easily modify parts without altering the whole.

According to Graves this is the process of learning to write effectively, regardless of the content of the text. With word processing you are not learning writing from a computer, but you are aided in your writing skills development with a useful tool.

And it is the teacher's role to highlight and discuss these processes. Do not assume that giving a kid access to a word processor will magically turn him/her into a best-selling author. The processes of writing need to be consciously understood.

Information Handling

The First Fleet Convict Data Base is an example of data base software that is readily available to schools. Thirteen items of data about each of the 777 convicts who were on the First Fleet are stored on diskette and can be quickly accessed, collated, counted, and selected according to the request you type on the computer, if it is a suitably phrased request. For example: SURNAME = JONES will select in a matter of seconds all convicts with the surname of JONES. This is a lot quicker than looking it up in a book, unless of course you're lucky enough to have a book that already lists the convicts in alphabetic order of surname.

When first approaching the data base, users tend to doodle, that is experiment with isolated requests, just to see what will happen. As

confidence and familiarity grows, a design for a project may grow from one of the doodles. For example: CRIME = MURDER often causes great surprise when the computer draws a blank and this may spark off a concentrated investigation of why there were no murderers and which were the worst crimes for transported convicts. The learners will progressively debug the design, testing related requests of the data base until they find a satisfactory conclusion to fit their hypothesis.

The content of the data base can be anything from NSW bushrangers to scientific observations in Antarctica, or census records, but what is important are the methods the students employ to solve problems and learn for themselves from the data that is available to them. Rather than passively reading someone else's opinions on convicts, for example, they are actively researching the topic for themselves. And the teacher's role is to guide their understanding of search strategies and problem solving.

Doodle, design, debug . . . or rephrasing Papert "teaching children to *be* historians versus teaching children *about* history".

Logo

Logo as used in primary schools, focuses on the "turtle" and uses commands like:

```
FORWARD 60
LEFT 90
BACK 40
PEN UP
RIGHT 10
```

to make it draw on a screen (or if you're lucky enough to afford a robot-turtle, on paper).

This simple and concrete, yet powerful "microworld" enables the child to explore the content of such topics as:

- spatial concepts
- mathematics and geometry
- computer programming and computer awareness
- art and design

but if we follow the "doodle, design, debug" philosophy, it is also a vehicle for highlighting problem solving strategies.

The learner obviously begins by doodling, investigating the capabilities of the world, discovering properties of angle, distance, and measurement.

The teacher can direct and extend this important doodling phase by setting such projects as mazes (Figure 1) or turtle golf (Figure 2) presenting them on transparencies to fit on the screen if no robot-turtle is available.

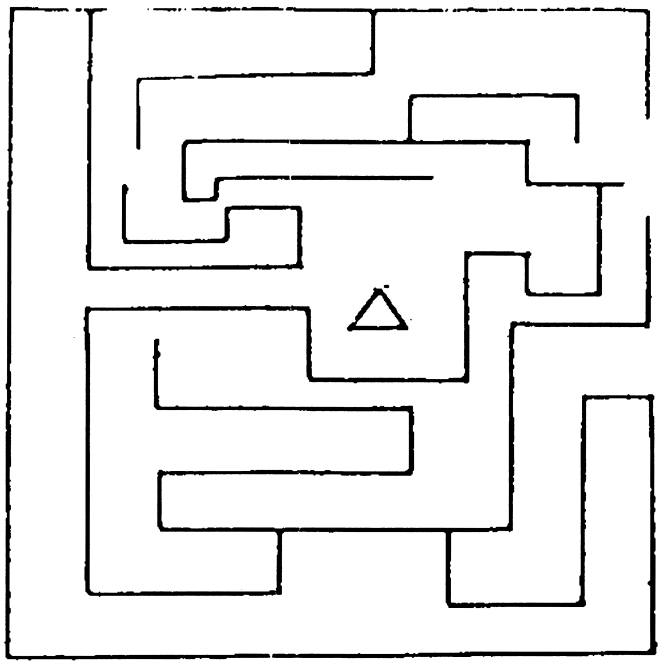
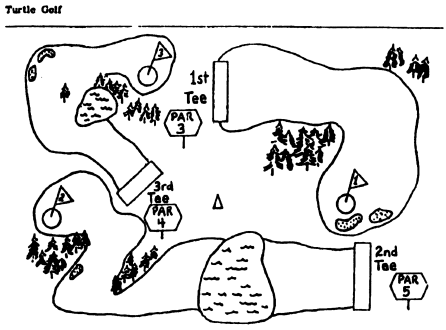


Figure 1



SCORE CARD Game 1		
Hole	Par	Strokes
1	3	
2	5	
3	4	
Total	12	

SCORE CARD Game 2		
Hole	Par	Strokes
1	3	
2	5	
3	4	
Total	12	

SCORE CARD Game 3		
Hole	Par	Strokes
1	3	
2	5	
3	4	
Total	12	

Figure 2, MECC 1983

Inevitably though the learner will want to make the turtle draw something. He or she will have a design in mind (Figure 3) and of course whenever there is a design there'll be a need for debugging (Figure 4).

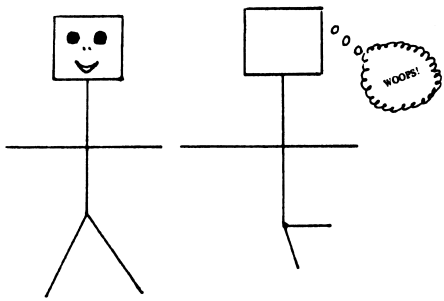


Figure 3 Figure 4

This will occur at every step along the way of learning with Logo, for example: the design in mind might be as in Figure 5.

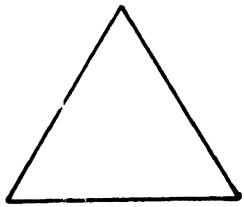


Figure 5.

Progressive debugging:

```
REPEAT 3 [FORWARD 80 LEFT 60
REPEAT 3 [FORWARD 80 LEFT 90]
REPEAT 3 [FORWARD 80 LEFT 135]
REPEAT 3 [FORWARD 80 LEFT 120]
```

Then doodle:

```
LEFT 60
REPEAT 3 [FORWARD 80 LEFT 120]
LEFT 60
REPEAT 3 [FORWARD 80 LEFT 120]
REPEAT 3 [LEFT 60 [REPEAT 3 [FORWARD 80 LEFT 120]]]
```

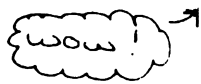


Figure 6. A three-dimensional design!

Notice the order doodle, design, debug is not pre-ordained! Doodling can be an interesting strategy at any stage.

The Ever-Popular "House with a Bug" Example with a Difference

If the Logo learner gets to the stage of storing procedures in the computer's memory and using these as building blocks in more complex designs, the need for designing and planning ahead carefully becomes even more crucial.

```
TO SQUARE
REPEAT 4 [FORWARD 60 LEFT 90]
END

TO TRIANGLE
REPEAT 3 [FORWARD 50 LEFT 120]
END

TO HOUSE
SQUARE
FORWARD 60
LEFT 30
TRIANGLE
END
```

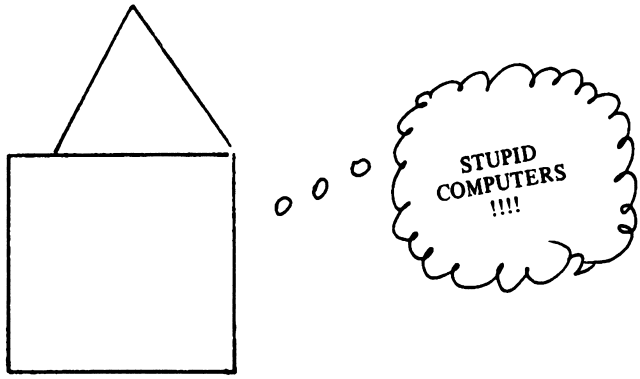


Figure 7.

But of course the best made plans . . . so into the debugging!

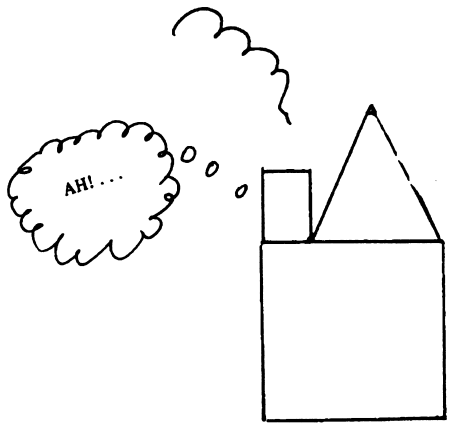


Figure 8.

Oh well, if the debugging is too hard, compromise, change the original design, Deplan rather than Debug! A cop-out? Or ingenuity?

And as for doodling with your building blocks, see Figure 9.

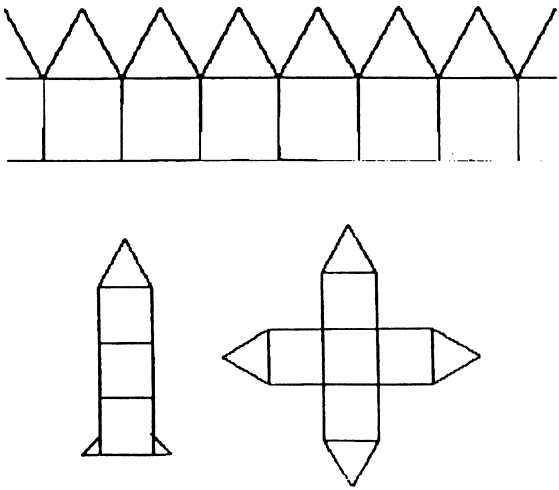


Figure 9

Logo is great for turtle geometry and drawing but it goes further than that. It is the role of the teacher to use those Logo projects not just to teach the content but to also highlight the problem solving strategies which may be generalised enough to transfer to other content areas.

What If?

Of course we always have been able to Doodle, Design, Debug using pencil and eraser/ "liquid paper"! The best thinkers have always understood the debugging process as an aid to solving problems. But the computer with its memory capacity now makes it even easier for us to think creatively, and perhaps opens up new avenues for many learners who may previously have found thinking laborious instead of an exciting challenge.

It enables them to ask "what if . .?" and seek to find the answer for themselves. A computer environment is unique in that it provides the learner with the flexibility to experiment, build, modify, save, combine and generalise.

Cynthia Solomon on Learning

"A different way of looking at learning and teaching emerges, one based on the Piagetian idea that even *young children have theories.* Thus teaching and learning are not a matter of being right or wrong but rather a process of *debugging*"

References

Education Department of NSW (1977) "A Supplement to the Aims of Primary Education in NSW".

Hebenstreit, J. (1983) "Computers in Education: the French Approach", unpublished paper presented to the WA Education Department, March 1983.

Papert, S. (1980) *Mindstorms.* Basic Books, New York.

Minnesota Educational Computing Consortium (1983) *Apple Logo in the Classroom.*

Solomon, C. (1982). Introducing Logo to Children. *Byte,* August, p.208.

Towards a Theory of Microworlds

Tony Adams

Information Systems Group
Royal Melbourne Institute of Technology

Sherry Turkle (1984) relates the story of Deborah, an adolescent described as withdrawn, lacking self-confidence, explosive and overweight, a child dependent and unhappy. Deborah, involved in a school Logo programme at the age of eleven was shown how to draw pictures on the screen by giving commands to the turtle. Turkle describes how at the beginning of the programme most of her time was spent trying to get the attention of the teacher and making little progress without the aid of others. Deborah changed when she restricted the power of her environment by placing an arbitrary rule that would allow her only one turning command, a right turn of 30 degrees.

> *Once she had her rule, Deborah got down to serious work. She drew flowers and rabbits and stars and abstract designs, everything built up of right turns of thirty degrees. 'I really liked my rule. It was neat. It was hard. I had to figure everything out. I thought about it all the time. I was the only one who had a rule.' Suddenly she found herself, perhaps for the first time, in a situation simple enough for her to control yet varied enough to allow for creative exploration. Weeks later, Deborah was the master of her restricted world and she began to dare to come out of it to experiment with less restricted geometry. Her mathematical learning had taken a leap forward.*
>
> (Turkle, 1984, p.143)

Turkle examines other microworlds

> *The issue of mastery has an important role in the development of each individual. For the developing child, there is a point usually at the start of the school years, when mastery takes on a privileged, central role. It becomes the key to autonomy, to the growth of confidence in one's ability to move beyond the world of parents to the world of peers. Later, when adolescence begins, with new sexual pressures and new social demands from peers and parents, mastery can provide respite. The safe microworlds the child master has built - microworlds of sports, chess, cars, literature or mathematical expertise - can become places of escape. Most children use these havens as platforms from which to test the difficult waters of adolescence. They move out at their own pace. But for some the issues that arise during adolescence are so threatening that the safe place is never abandoned. Sexuality is too threatening to be embraced. Intimacy with other people is unpredictable to the point of being intolerable. As we grow up we forge our identities by building on the last place in psychological development where we felt safe. As a result many people come to define themselves in terms of competence, in terms of what they can control.*
>
> (Turkle, 1984, p.207-208)

Turkle sees a microworld as a place where issues of mastery and control can be faced. She alludes to games such as chess, to sports, to areas such as literature and mathematics where a child (or adult) can immerse themselves in a bounded place, with rules to operate by and where exploration can take place. The computer is not an essential ingredient to such a microworld. Generations of children have immersed themselves in model railways and games based on complex rules such as Monopoly.

Definition of a Microworld

Papert describes a microworld as

> *A subset of reality or a constructed reality whose structure matches that of a given cognitive mechanism so as to provide an environment where the latter can operate effectively. The concept leads to the project of inventing microworlds so structured as to allow a human learner to exercise particular powerful ideas or intellectual skills.*
>
> (Papert, 1980, p.204)

Microworlds according to Papert are not new and could be implemented with many different psychologies of learning in environments that are not necessarily computer based. The new feature is the "availability of a technology for microworlds". He sees the microcomputer as providing a technological base that will allow the microworld to become a refined systematic and theory-based branch of education research. Papert argues that educators have only taken from Piaget's work what has been "implementable in traditional schools with traditional curricula" and that microworlds might provide an insight into Piaget's ideas that state "intellectual development does not always need explicit teaching".

This view is supported by Sewell and Rotherway (1985) who state that Papert's views on knowledge acquisition by exploratory learning are consistent with Piaget's constructivist hierarchical model rather than his stages theory which "under-estimates children's mental abilities".

Papert discusses students' understanding of Newton's laws and concludes that "after a year of college physics, most students cannot be said to understand the laws even if they have acquired skills in solving set problems". He further states

> *I maintain that a large part of the problem has to do with the difficulty of finding genetic stepping stones to Newton's laws. By this I mean something else in the mental world of the learner which will act as a precursor of her or his personal reconstruction of the Newtonian laws. Instead, the student is expected to make a leap to Newton that strains him in three ways. The first strain comes because he doesn't know anything else like these laws to which he can relate them. The second is that in fact they contradict much in his immediate experience of how the world works...The third comes from the 'passive' mode of learning: the new knowledge cannot be exercised in a personally involved and motivated way; it cannot be assimilated into living experience.*
>
> (Papert, 1980, p.206)

A Newtonian microworld is proposed by Papert that includes a structure that links turtle geometry, a precursor to the understanding of Newton's laws and a physics turtle that obeys Newton's laws. Papert comments that "the beginner is given a set

of down-to-earth but nevertheless *useable* ways to think about Newton's laws and the student has "lived through the experience of what it is like to control both" the geometry and the physics turtles.

Lawler (1984) discusses Papert's ideas of computer based microworlds as helping to tailor instruction more closely to Piaget's ideas of natural learning.

> *Learning is often a gradual process of familiarization, of stumbling into puzzlements, and resolving them by proposing and testing simple hypotheses in which new problems resemble others already understood. Microworlds are in essence task domains or problem spaces designed for virtual, streamlined experience...The appropriation of knowledge embodied in those experiences is made possible because the microworld does not focus on 'problems' to be done but on 'neat phenomena' - phenomena that are inherently interesting to observe and interact with. With neat phenomena, the challenge to the educator is to formulate so clear a presentation of their elements that even a child can grasp their essence. A well designed computer microworld embodies the simplest model that an expert can imagine as an acceptable entry point to richer knowledge. If a microworld lacks neat phenomena, it provides no accessible power to justify the child's involvement.*
>
> (Lawler, 1984, p.41-42)

Fischer (1981) defines a microworld as "a delineated piece of reality in which certain ways of acting and thinking work particularly smoothly and transparently without being disturbed by other components which the final skill will eventually require". Fischer and his co-workers have applied the language of computational learning environments (microworlds) to the teaching of skiing in an attempt to analyze the features of this highly successful learning environment to identify the principles to guide the design of computer based learning environments. He regards skiing as a success model in that it is an example of a complex skill that is learned by many people. In selecting the elements of a microworld the following uses are identified:

> *Makes it easier to begin learning a skill by creating the right entry points*

> *Accelerates the acquisition of a skill.*
> *Provides intermediate goals/challenges that are (and seem to be) attainable.*
> *Provides practice of the important subskills in isolation, allowing the common bugs to occur one at a time instead of in bunches*
>
> (Fischer et. al. 1978, p.121)

The paradigm Fischer and his co-workers based their work on they call "increasingly complex microworlds, in which students are taken through a number of stepping stones" to more complex environments.

> *The best example of a sequence of increasingly complex microworlds in skiing is the graduated length method (GLM). The student begins to ski on short skis over smooth and flat terrain. The short skis allow him to develop rhythm, to turn easily and to get up from a fall. The smooth and flat terrain limits his speed and reduces the danger and his fear. As the student gains abilities within these constraints, he is given slightly longer skis and he moves to steeper slopes until he is finally using full length skis on arbitrary slopes.*
>
> (Fischer, 1981, p.479)

A number of important features are identified.

Entry points

Entry points "get the learner started" and lead not to the final form of the microworld, but to one which is constructed from a simplification of the final skill.

Transient objects

Transient objects are "conceptual or physical entities which lead from one microworld to the next one...after the new level of experience is reached they are no longer needed". An example of a transient object provided by Fischer is the floor or robot turtle which "provides a transient object from a concrete world into the more abstract world" of TV images, procedures, programming and problem solving. These transient objects described by Fischer may themselves be microworlds.

Simplification

Fischer sees that simplification is possible in each of the major components of the learning process: the skills required, the equipment used and the environment. In terms of skills this may be beginning microworlds such as Papert's turtle geometry, or subskills that can be developed, perhaps in isolation, in which students can develop these subskills without having to deal with the interactions and side effects of the whole aggregate of subskills. A simplification of the equipment is given by Fischer's use of short skis or safety bindings which reduce the fear and eliminate the disastrous consequences of wrong behaviour and therefore support an active approach to exploring and mastering new environments. The following heuristics are identified.

- reduction of the number of subskills to get started,
- start with a transitional model close to the student's previous experiences,
- eliminate the danger of making irreversible mistakes,
- provision for decomposition of the task, and
- simplification of the task.

Groen provides a more formal definition of a microworld.

> *The basis of the theory is a formalization of Piaget's notion of structure...The most important aspect is that a structure is a set of states and transformations between states. An important property is that the transformations should be modular. In other words, they should be easily decomposable into chunks.*
>
> *A microworld is a (Piagetian) structure with certain additional properties. The two most important are:*
>
> 1. *A transformation can be undone to go back to the previous state;*
> 2. *There should exist mappings (in the precise mathematical sense of the term) to other structures that are representations of concrete actions in the real world.*
>
> (Groen, 1984, 9.51)

Using the above definition he identifies turtle graphics as a microworld, since the states are possible positions of the turtle on the screen, the commands are the transformations (which can be undone) and the procedures (the chunks) can be mapped onto the real world by pencil and paper (or body movement).

The idea of objects is also important to the design of a microworld. Turtle graphics is a form of object programming where communication is with an object that is completely defined by its state variables (position, heading, pen status, colour, etc.). Program commands communicate with the turtle to change these state variables, which are acted upon by side effect. The object nature of the turtle is clearly important to microworlds that make use of turtle geometry. The Newtonian microworlds described previously, Lawler's POLYSPI microworld (Lawler, 1984, p.43), Turkle's thirty degree microworld and others described in the literature operate by causing state changes to the turtle. Fischer's skiing microworld operates on state changes to a principal object (the skier) as well as subsidiary objects (ski poles, etc.).

Other non computer microworlds identified by Turkle such as the chess microworld rely on state changes to chess pieces. Chess pieces have a state (their position, colour, in the game or taken) and act according to a set of rules. The move command to a knight changes its state according to a set of rules (move forward one square and right two, for example). McDougall (1985) reports on the difficulty that she and Squires have had in discussing physics microworlds independent of the turtle; she attributes this to the lack of implicit objects in their thinking. This view is supported by Sewell and Rotheray (1985) who argue that in a Piagetian scheme, "knowing an object consists of acting upon it, manipulating it and discovering its properties". Lawler states that anyone who designs a computer based microworld should strive to represent all the state variables in a visible, obvious way. Doing so enhances the comprehensibility of the ideas embodied in the manipulation.

Clearly microworlds are not limited to Logo turtles. Gravina (1980) describes a microworld based on boxes of changing shapes and colours implemented in Smalltalk, an object oriented language.

The Essential Properties of a Microworld

From the above, the following essential properties of microworlds can be described:

- a powerful idea,
- a simplification that maps onto some part of the real world,
- a set of entry levels,
- a state description of the objects in the microworld,
- a set of commands that can transform the state of the objects and can be undone,
- an operating environment that contains a set of rules for transforming the state of
- objects, and
- a set of exit levels.

This description provides a specification for both computer and non-computer based microworlds. In terms of computer-based learning environments it will include the possibility of software such as simulations and work associated with approaches in the area of creative writing and exploratory learning with databases, including some current work with Prolog. It may be no accident that many working with Logo find a community of interest with teachers and researchers in these areas.

The importance of Logo is twofold in the development of a theory of microworlds. Firstly it is unique amongst computer languages and software used in education in that its learning model is explicit and based on a theory of cognitive development. This has led to Logo being used as a focus of research, discussion and practice about the use of computer based learning environments. Secondly its particular features, notably those of proceduralisation and extensibility provide two additional microworld capabilities of great power, these are:

- entry and exit points at different levels of abstraction, and
- decomposition of microworlds into other microworlds.

Lawler's POLYSPI microworld provides an example of a bounded domain which is simple in concept and lies within and adjacent to other microworlds. The power of Logo stems from the ability not only to develop and use POLYSPI, but also to dig into it as part of other experiences. He says of POLYSPI

> *The procedure...and its designs comprise a microworld. The objects of the microworld are all the designs that the procedure can generate, an engaging and extensive domain for exploration. More important, the designs are a class of 'neat phenomena' whose generation can be made comprehensible with the following set of ideas. First, the POLYSPI procedure provides a crisp model of variable separation: the three variables...are used once, and used differently...Second, the difference in relative potency of the variables (the impact of a unit change...) is obvious and striking. (ANGLE and then CHANGE are much more potent than DISTANCE). The POLYSPI microworld reveals the stepping of variables as a powerful idea. By stepping variables I mean identifying one variable as a dimension of examination and holding all other variables constant while the chosen one is varied incrementally...Piaget judged variable stepping to be an essential component of formal operational thought. The idea is a powerful one because it is almost universally useful; it is crucial to the process of scientific investigation. Within the microworlds of turtle geometry, the insights achieved are easily extended to a related microworld of INSPI.*

Fischer identifies in his skiing example the idea of increasingly complex microworlds which may be treated as transient objects (skills) towards the development of a final skill. In other words, depending on their existing skill levels students may start with different microworlds and move towards more advanced microworlds. Fischer sees this clearly as a number of microworlds linked together at different levels of abstraction in terms of the final skill. Squires and McDougall (1985) also see entry and exit at different levels, but in their analysis these levels are within a single microworld. The notion of decomposition of microworlds implicit in Lawler's example unites these ideas.

Non Turtle Graphics Microworlds in Logo

Logo has provided the opportunity for the development of other microworlds but only limited successes have been recorded. The Logo community has never come to terms with lists, they are just too hard to manipulate. Unlike turtle graphics they do not have a low entry threshold and for most applications a good working knowledge of recursion is required. It is more than a suspicion that many Logo educators simply avoid contact with recursion, an act that forever excludes them from list processing. Even when lists are well understood, the question of what to do remains unclear. The reason if not the solution remains straightforward. Howe (1975) talks about 'kits of parts', of providing primitive constructs that reflect the nature of the problem...rather than machine oriented syntax...We have hints of what these 'kits of parts' might be like. Abelson (1982) draws on artificial intelligence models and McDougall, Adams and Adams (1983) use some of Mike Sharples' ideas to create an interactive poetry kit. The essential notion is that appropriate primitives are provided in some area of interest. To date the work in this area has been too little and fragmented.

Groen (1984) states that language features such as lists do not by themselves constitute a microworld. Sharples (1985) comments that "attempts...to teach list processing to children and adults have not been successful. Learners who enjoyed and profited by turtle geometry were bored and confused by lists". He further comments that lists in Logo are based on the "elegant, abstract structures" of Lisp rather than the pragmatic features of Turtle graphics which act as instructions for "a notional drawing machine" and are based on state changes of the turtle which are readily observable.

Attempts to create microworlds based on lists have centred on using list processing as a precursor (or tool) for the development of the microworld rather than as a precursor to its use. McDougall, Adams and Adams (1984) create a data-base microworld. Sharples (1978) and McDougall, Adams and Adams (1983) create a poetry microworld. Sharples (1985) extends some of these ideas further to provide a phrasebook microworld. In each case they conform to the definition of a microworld. The data-base microworld can be used to illustrate this:

- a powerful idea that states that knowledge can be held in a data structure and is a simplification of both computer data base access and knowledge engineering,

- data base objects that have properties and have a current and observable state,
- a set of commands that operate on the data base objects, and
- a set of rules of Logo syntax, of the derived commands and of the use of the data base.

Conclusion

The development of a useable theory of microworlds is a necessary step in the evolution of computer based learning environments. To date Logo has been bound to inadequate machine environments. These environments have artificially restricted educators' perceptions and the development of microworlds outside of turtle graphics. There is evidence in the new implementations of Logo on the more powerful machines now available that developers see Logo as a microworld creation tool.

A theory of microworlds also provides an opportunity to establish a sound theoretical basis for the development of computer based learning environments independently of Logo. This may be the only opportunity for cognitive psychology to influence computer based learning.

References

Abelson, H. (1982) *Logo for the Apple II*, Byte Books, Peterborough, NH.

Fischer, G. (1981) "Computational Models of Skill Acquisition Processes" in Lewis, R. and Tagg, D. (eds.) *Computers in Education*, North-Holland, Amsterdam, p.477-481.

Fischer, G., Brown J. and Burton, R. (1978) "Aspects of a Theory of Simplification, Debugging and Coaching", BBN Report No. 3912, Bolt Beranek and Newman Inc., Cambridge, Mass, July .

Groen, G. (1984) "Theories of Logo" in Sorkin, R. (ed.) *Logo 84*, Pre-Proceedings of the National Logo conference, MIT, Cambridge, Mass, p.49-54.

Howe, J. (1975) "Artificial Intelligence and Education", in Hooper, R. (ed.) *Computer Assisted Learning in the UK*, Council for Educational Technology, London, p.295-317.

Lawler, R. (1984) "Designing Computer Based Microworlds" in Yazdani, M. (ed.) *New Horizons in Educational Computing*, Ellis Horwood, Chichester, p.40-53.

McDougall, A. (1985) Personal Communication.

McDougall, A., Adams, T. and Adams, P. (1983) *Learning Logo on the Apple II*, Prentice-Hall, Sydney.

McDougall, A., Adams, T. and Adams, P. (1984) *Learning Logo on the Commodore 64*, Pitman, Sydney.

Papert, S. (1980) "Computer Based Microworlds as Incubators for Powerful Ideas", in Taylor, R. (ed.), *The Computer in the School: Tutor, Tool, Tutee*, Teachers College Press, New York, p.203-210.

Sewell, D. and Rotheray, D. (1985) "Theoretical Influences on the Design and Implementation of Computer-Mediated Learning" in Duncan, K. and Harris, D. (eds.) *WCCE85 Proceedings*, North-Holland, Amsterdam, P.1019-1023.

Sharples, M. (1978) "Poetry From Logo", Working Paper No. 30, Department of Artificial Intelligence, University of Edinburgh.

Sharples, M. (1985) "Phrasebooks and Boxes: Microworlds for Language", in Duncan, K. and Harris, D. (eds.) *WCCE85 Proceedings*, North-Holland, Amsterdam, p.617-622.

Squires, D. and McDougall, A. (1985) "Computer Based Microworlds - a definition to aid design, publication pending.

Turkle, S. (1984) *The Second Self: Computers and the Human Spirit*, Simon and Shuster, New York.

Designing Computer Based Microworlds

David Squires
Royston Sellman

Educational Computing Section
Chelsea College
552 Kings Road
London SW10 0UA

Introduction

The Computers in the Curriculum Project at Chelsea College has been concerned with research and development in computer assisted learning for ten years. During the last year work has commenced on the production of microworlds which can be used to support science education. In this paper we offer a definition of the term microworld and describe a microworld we have developed using this definition.

There is an elusive quality about a software environment which makes it a microworld. In such an environment learners are able to access the power of the computer in a motivating and relevant way. Many essentially trivial pieces of software are claimed to be microworlds but if the concept is to realise its full potential in education there must be some general agreement on the essential features of a microworld and a significant number of examples which are relevant to a wide range of learners and teachers must be provided.

In order that we may develop legitimate examples we have adopted the following definition of a microworld:

A computer based microworld is formed out of the conjunction of

(i) some clearly stated primitives developed from a fundamental concept or concepts

and

(ii) a set of programming constructs. Using the programming constructs the learner must be able to manipulate the primitives to produce an extensible set of operations which can interact with each other.

The programming constructs such as recursion, repetition, conditionals and procedures are usually made available to the learner through a computer language. A major criterion when selecting a microworld language is the range and suitability of the constructs that it offers.

The fundamental concept concerned must be represented in a viable way. This can be done by using a concrete object as in turtle geometry where the turtle clearly represents the concepts of rotation and translation.

Figures 1 and 2 are intended to illustrate these ideas:

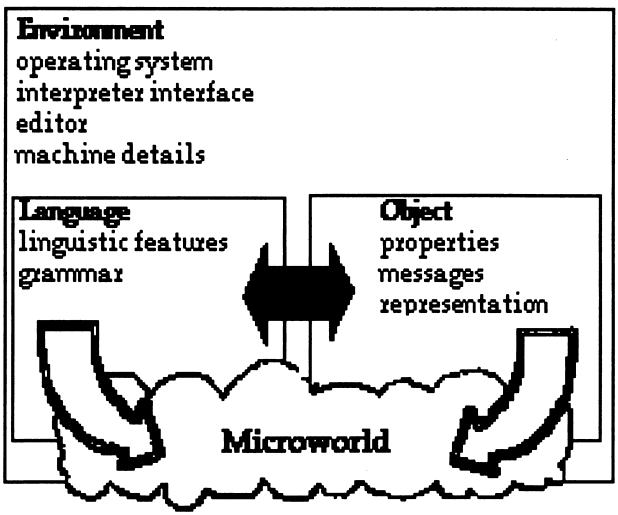


Figure 1. An operational definition of a microworld

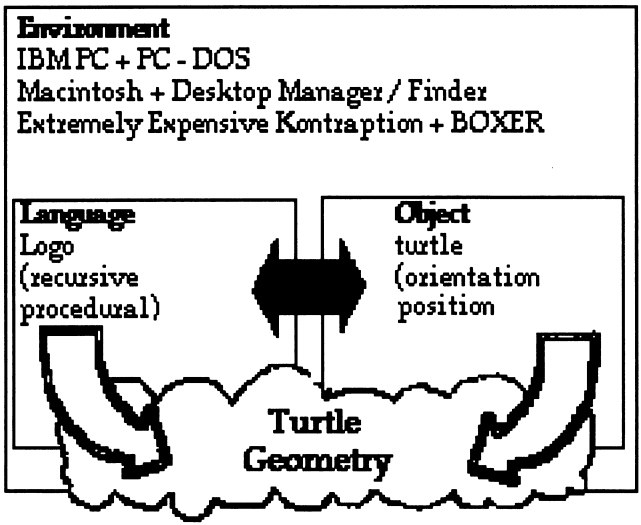


Figure 2 - Some examples of the definition

A 'conventional' turtle can be adapted to provide a way of representing other concepts by adding to the list of primitives which describe the turtle. By adding a balance primitive to this list we have produced a microworld based on equilibrium. We are using Logo to provide both the turtle and the programming constructs so that learners may explore this microworld.

A Balance Microworld

To implement a microworld around the concept of Balance the following initial tasks have been completed:

(i) We have written a small suite of procedures, mainly in Logo, intended to add to the list of primitives which define a conventional turtle. We call this new creature the Balance turtle.

(ii) We have devised methods of encoding the screen (the turtle field) as a list (of lists) of numbers. We call this object the screen/list.

(iii) We have written procedures which take the screen/list and represent it in ways appropriate to the educational context.

How Does It Work?

The Balance Turtle, when told to get going, examines nearby locations in the screen/list and follows a rule - the **Balance rule** - to choose a direction of travel. Sometimes the turtle may be unable to follow the Balance rule which could be upsetting for her if we didn't provide a second rule - the **Action rule** - which helps out in such cases. Incidentally these rules, which respectively inquire of and make changes to the turtle field, show why it is so convenient to represent it as a list and thus allow Logo's list processing features to be exploited.

Balance then, can both inspect and alter its environment according to rules. These rules can either be chosen from those we intend to supply or teachers could encourage learners to write their own.

A Simple Example

step 1

```
TO BALANCE
 BALANCE_RULE
 IF RULE_WAS_OK [BALANCE] {else} [ACTION_RULE
 BALANCE]
END
```

i.e. The turtle will follow BALANCE_RULE when she can, then follow ACTION_RULE then continue.

step 2

We need a screen/list. Lets say the list looks like Figure 3.

9	9	9	9	9	9	9	9	9	9
0	1	2	3	4	5	6	7	8	9
9	9	9	9	9	9	9	9	9	9

Figure 3 - A slope screen/list

step 3

Now say the balance rule is just a move towards the lowest nearby value in the turtle field. If we start the turtle off at the right hand side she will eventually find her way to zero. Now there is no lower value nearby, the balance rule cannot be followed and the turtle requires the action rule. Let's make that - Reverse the row you are in.

So now:

```
TO BALANCE
 FINDLOWEST
 IF FOUND_ONE? [BALANCE] {else} [REVROW ROW?
                                          BALANCE]
END
```

will produce a simple see-saw oscillator.

A Toolkit for the Microworld

To make the turtle work and be compatible with the ordinary Logo activities the learner may want to do, we have written a number of support procedures. These cover such things as:

Moving the turtle to a region in the screen/list.

Letting the turtle find out where she is in the screen/list.

Handling screen/list definition.

Simplifying list processing operations on the screen/list.

Giving a pictorial representation of the contents of the workspace.

Making the Microworld Accessible to Learners

Microworlds enable learners to work in a way which is appropriate to their own needs, experience and aptitude. Ideally learners need to enter a microworld at a level which suits them.

As software designers we are enabling entry at a number of different levels by providing procedures which encapsulate derived features within the microworld. In the Balance microworld, we will supply a number of balance rule procedures and several screen/lists representing configurations of varying complexity.

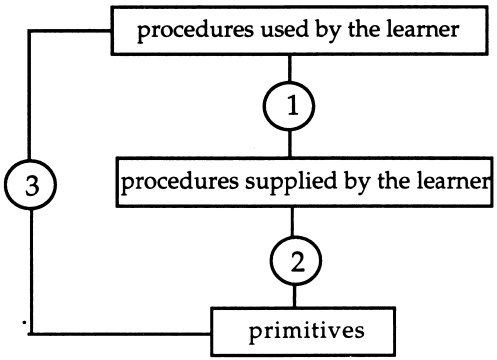


Figure 4. Entry levels to a microworld

Learners may use a supplied procedure in isolation or combine several of them to create new procedures. In this way the supplied procedures can be regarded as pseudo-primitives (it might be pedagogically sound to 'bury' them, that is make their contents invisible, using the facility now available in most Logos). This corresponds to route 1 above.

Of course, learners may just follow route 3 in Figure 4 and write their own procedures without reference to anything we supply.

Summary

The formulation of a basis for a well designed microworld involves:

(i) A clear definition of the microworld in terms of primitives derived from the concept(s) the microworld is based on, together with the identification of a programming environment which makes an appropriate set of programming constructs available.

(ii) The articulation (and documentation) of a set of derived procedures which enables a learner to enter the microworld at an appropriate level.

Logo and Language Development

Carolyn Dowling

Lecturer in Computers in Education
Institute of Catholic Education, Mercy Campus.

We are all familiar with the idea of Logo as a mathematical environment - an environment which facilitates the formation, development and exploration of mathematical concepts and intuitions, and of thinking strategies which, it is hoped, may be transferable to other areas.

These qualities of Logo have led to its being welcomed with enthusiasm by many teachers who have made a significant place for it in their classroom activities. There are, however, a number of other teachers who either reject Logo out of hand, or consign it to an obscure corner of the curriculum on the grounds that it is 'only about maths' or is 'just a thing for drawing pictures'.

There is a real problem here, in that something which is seen as being 'about maths' is regarded by much of the community, including teachers, who may themselves have suffered a rather sketchy mathematical education, as being intrinsically difficult and generally inaccessible to all but specialist mathematicians. This attitude can lead to a rejection of Logo, or, where it is used, to a nervous product-oriented approach reflecting the teacher's own limited mathematical experience and insight - attempts to dutifully reproduce a small subset of shapes, for example.

Likewise, 'drawing pictures' is perceived by many teachers as having limited educational value beyond the sphere of lower primary grades and art classes.

Are such teachers justified in their judgements of Logo? There is a sense in which they are. To present teachers with an open-ended

tool for learning and expect them necessarily to apply it in new and creative ways to a wide variety of learning situations is probably asking too much. Certainly there are teachers who will see the possibilities being offered and will take up the challenge, but for many the real-life limitations of time, energy, even their own abilities, will militate against them. They look around them, see Logo being used for maths and drawing, and that's that.

But are there other doorways into Logo which might make it more accessible to such teachers, and hence to their students? Language development, rightly or wrongly, is generally perceived as being a less esoteric field than mathematics. We may not all be able to multiply and divide effectively, but a functional level of language use is the heritage of virtually every human being. The catchcry, "every teacher is a teacher of English," is accepted as a truism (at least in English speaking countries!). Is Logo relevant to language development? If this could be seen to be so, then the hesitations of some teachers might well be overcome.

Logo is a language. It is a system for the symbolic representation, manipulation and communication of ideas, concepts, or whatever term one chooses. Being a system, it has rules which must be followed. In this sense all forms of mathematical expression are, of course, language, although the connection between mathematical and natural language is often not made clear to students, to their loss.

Logo, however, has features which closely resemble (albeit sometimes spuriously) natural language (such as the use of 'natural' terms for commands). This both makes the language easy to learn, and suggests implicitly that further analogies might be made which could shed light on the rationale of some aspects of natural language.

Logo is a language with a special purpose - it is designed to enable human beings to communicate with a computer. This aspect of Logo is likely to provide some interesting points of comparison with natural language, its uses and effects. Communication between person and machine and between persons through the intermediary of a machine is becoming increasingly prevalent in our society. The differences which this makes to the manner, content and effect of communication are subtle, but they do exist, and are already being investigated in some work environments. This is a new factor in language use, and an area where children might be led, through working with Logo, to the development of very important insights.

A further application of Logo in the area of language development is in the production of programs designed to manipulate elements of natural language in an enlightening way. This can include the use of Logo to carry out the computer application dearest to the heart of most teachers of language, word processing.

These then are the three categories in which I shall now discuss Logo as a language environment in more detail - Logo as 'just another language', Logo as a language for communicating with computers, and Logo as language for writing programs useful for language study and development.

The study of any language other than one's first natural language is important both for the extension of our ability to communicate, and for the light which is shed by comparison, whether explicit or implicit, upon one's first language. Indeed a large number of children who study a 'foreign' language at school gain little skill in communicating in a real-life situation in that language. For them the value of the exercise has been in a heightened awareness (of which they may or may not be conscious) of features of their own language such as the structure of sentences, verb forms, nuances of definition and so forth which they may previously have taken for granted and ignored as self-evident.

But there are still students for whom natural language in whatever form is too complex a medium for theoretical issues to be made clear. For such students, it is possible that familiarity with a more limited language environment such as Logo, simpler in structure and limited in initial vocabulary, might lead to fruitful insights into the significance of features of their own language, and thence into broader questions concerning language in general.

For example, the effects of sequencing of elements of language can be clearly seen by the youngest student when directing the turtle. This can be discussed in combination with extremely straightforward ('the man bit the dog') and more complex examples in the students' own language. In Logo as in natural language, failure to obey accepted rules of syntax generally results in misunderstanding, or in some cases, a lack of any meaningful understanding at all. (Compare 'I don't understand a word you're saying' with 'SYNTAX ERROR'!)

Why might the particular commands and syntax used in Logo have been chosen, rather than other possibilities? Apart from technical considerations falling into the province of computer science, questions such as, 'is it necessary to have both FD and BK and both LT and RT', are worth debating with quite young children. Are there words in our own language that we could do without under some circumstances?

What letters of the full commands are used as abbreviations in Logo? When do we use abbreviations in our own language? Are there rules which govern which letters we choose to use in the abbreviated form of a word?

Often the same result can be achieved on the screen with a different combination or sequence of commands. To what extent can we express the same thing in different ways in our own language?

In Logo, spaces must be left in particular places. Where do we leave spaces in our own language? What about hyphenated words?

Variables are widely used in Logo. How are variables expressed and understood in natural language. Dad, Jack, Mr. Smith and the Boss may be one and the same person - or are they?

In the area of meaning, definition and the creation of words Logo offers opportunities for exploration which are not available in real-life situations using natural language. The opportunities for the average person to create a new word, or to name something, are virtually non-existent after early childhood. Objects, events, ideas in the world, already have labels attached to them, and in the interests of effective communication a great premium is placed on the efficient and accurate acquisition of these by the child. Programming in Logo, with its analogy to the creation of new 'words', opens the door to a heightened awareness of words and their meanings. In this context, Logo programming can be linked with historical reconstruction of the possible origins of words in the child's own language, with the study of dictionary definitions, of changes in meanings of words, of the use of existing words to define new ones, of the naming of new inventions and a host of other related topics.

But Logo is not a natural language, despite its similarities. Not only did it not follow the same evolutionary process of development,

but, importantly, it is not a vehicle for communication between human beings. It is this difference which can be utilised both to familiarise children with the peculiar qualities of communication with and through a computer, and also to highlight many features of communication in natural language which are not present in using Logo.

Human beings are flexible and tolerant in their use of natural language. Not only this, but natural language is supported by a variety of non-verbal cues such as facial expression, gestures, intonation, pauses and so on. From an early age children are well aware that it is only teachers and anxious parents who demand precision in language use. The rest of the world is wonderfully willing and helpful at struggling to decipher the vaguest and most confused utterances. And as for spelling....If they say you got it wrong, they must recognise what you meant, so what's the problem?

Communicating with the computer is a very different matter. Lacking human organs of perception, the machine is limited to accepting the precise and literal meaning of a written message. Suddenly spelling really matters - a word spelt differently is a different word. Word order must be exact if a message is to be understood and acted upon.

Thus features of our communication in natural language can be highlighted in exercises and discussion comparing communication in Logo with natural discourse.

Computer awareness and the study of natural languages combine in the writing of interactive 'conversation' programs which cause the computer to appear to accept, absorb and re-use information in a way that seems at the very least commensurate with human intelligence. Without in any sense denigrating AI activities in their many forms and applications, it is important that children have at least some understanding of the processes that lie behind this apparently 'humanly intelligent' behaviour of the computer. To have children program the computer themselves so that it appears to be having an 'intelligent' conversation with the user (somewhat in the 'Eliza' mode) is both a valuable exercise in computer awareness, and a fascinating language development activity which focuses attention to a degree which is difficult to achieve in natural language alone, on the mechanisms of human conversation. In the

course of this type of programming activity, the wording of questions to evoke responses predictable within a certain range must be carefully considered. The range of likely responses places constraints on the next utterance, and so on. As with all activities where the computer is used in the classroom, the benefits of this exercise are multiplied when it is undertaken as a group activity.

As a programming language, Logo can be used by teachers and students alike to create opportunities for manipulating textual material in natural language. Some possibilities include the creation of templates designed to generate particular sentence structures, forms of verse and so on. The value of such exercises comes from comparison of the results achieved with what was expected, and from the subsequent processes of debugging as a result of further insight into what is actually required. ('Is this a proper sentence?' 'If not, why not?') Programs which make specific changes to text (such as altering the tenses of verbs), which produce a variety of 'cloze' type exercises from a piece of text and so on can be of great value, particularly if the programs are written by the students themselves in collaboration with the teacher, so that the underlying concepts are being creatively explored, rather than correct answers being sought.

The growing popularity of word processing linked with the process approach to writing has much in common with the affectionate regard in which Logo users hold their bugs, and the attractions of the debugging process. Simple word processing in the Logo Editor, particularly for younger children, does away with the need for additional (possibly costly) word processing packages, in addition to enabling integration of words and pictures. Programs can be written to add many of the features of 'proper' word processing packages.

In concentrating on these aspects of Logo, has the baby been thrown out with the bath water? Logo was, after all, conceived by its creators as an essentially mathematical environment. Does an emphasis on language contort its use to such an extent that more is lost than gained?

From my own experience, limited though it certainly is, I would suggest not. I referred earlier to the restricted view of mathematics held by many teachers as a result of their own learning experiences.

These limitations are reflected in the use many of them make of Logo with their students. In a number of cases I have observed, it is only when such teachers are liberated from the necessity of having to do what they think of as mathematics, that real mathematical exploration and development, (incidentally), takes place. With the focus firmly on language, the turtle can be permitted to waddle happily off into the unknown, often with the most exciting, unexpected and educational results!

References

The ideas discussed in this paper have been inspired by so many different sources that it is not possible to document them with any accuracy.

More detailed development of some of the teaching ideas may be found in

Nevile, L. & Dowling, C. (1983) and (1984) *Let's Talk Turtle: Teachers' and Parents' Edition*, Prentice-Hall.

Nevile, L. & Dowling, C. *How Turtles Talk: Teachers' and Parents' Edition*, Prentice-Hall, forthcoming.

Logo Tool Kits

Valerie A. Clarke

Division of Cognitive Science and Psychology
Deakin University
Victoria, 3217

Computer based learning in schools generally takes one of two distinctly different approaches (Buck & Kohn, 1985). The first one focuses on computer programming where the main concern is with creating programs. Emphasis is placed on learning the commands and syntax of the language. Student activities involve writing programs designed to develop programming skills, rather than to achieve goals that are intrinsically rewarding. The second approach is to use pre-written programs to achieve clearly defined ends. Generally, the emphasis is on mastering a particular content area (e.g. CAI programs) or developing particular skills (e.g. data querying). Usually the student is not expected to think about the program and, in fact, is often unaware of the language in which it was written. Programs are defined as "user-friendly" if the code is kept well hidden and the user is clearly instructed as to the specific response to be made (Barclay, 1985).

Such user-friendly, carefully designed instructional packages, where the environment is fully controlled by the computer are inconsistent with the Logo philosophy. Whereas this philosophy seeks to place the students in control of their own learning by providing powerful tools with which they can explore, develop and test ideas, most pre-constructed software places the computer in control of the learner, defining the responses which can be made, and in many instances prescribing which responses are "correct" and which ones are "incorrect".

An alternative method of introducing children to computers is to adopt the best features of both approaches by providing "black

boxes with lids that open" (Martin, 1985). Students are given a tool kit of procedures designed to both develop specific skills and provide programs which can be examined and explored. Initially the programs are treated as "black boxes". At this stage they are used by the students to learn particular ideas or develop particular sets of skills. For example, they may be designed to develop familiarity with the computer and a particular set of commands from a given programming language, as well as to develop a defined set of skills or mastery of specific concepts. Once the students are completely familiar with the use of the programs, and their understanding of the computing language has reached an appropriate level, the "black box" procedures can be opened, displayed, and studied. The students can examine the way the code was written, modify it to make it more consistent with their own needs or interests, extend it, or use ideas derived from it to develop other programs. As exploration of these tool kits continues, students develop an understanding of the construction of them, which, in turn, creates an interest in exploring the capacities of the language and its applications to other activities.

This method of learning is essentially formalising a style of learning which is evident throughout the non-classroom computer culture. Many computer addicts have their initial contact with computers through a wide variety of arcade games (Reinecke, 1983; Keisler, Sproull & Eccles, 1983, 1984). They play these games extensively until, becoming bored with the pre-constructed games, they seek ways to crack the code and modify original programs. Through code breaking and piracy, many computer users gain a fairly comprehensive understanding of the operation of both the hardware and the software.

However, only a selected section of society engages in this experience. Most arcade games are designed by middle-class, white, male programmers to appeal to their middle-class, white sons (Note 1) leading to the development of programs which appeal to males and create a male image of computers, computing and video arcades, as well as to the development of computing skills among this particular section of society (Kiesler, Sproull & Eccles, 1983, 1984). It is suggested that this male image can be neutralised, and the benefits of these learning opportunities retained, by providing similar learning situations designed to appeal to a wider audience,

and made available to a broader spectrum of students. This is achieved by designing tool kits which are equally appealing to both girls and boys, and which involve activities that can be integrated into the school program. Such a tool kit has been developed to introduce primary school children to Logo turtle graphics. A further tool kit is being designed to develop a number of skills relevant to the language curriculum.

Turtle Graphics

A set of games procedures was designed to achieve four specific objectives: to introduce kindergarten and lower primary school children to the initial Logo commands (e.g. FORWARD, BACK, LEFT, RIGHT, PENUP, PENDOWN, LOAD); to develop an understanding of the turtle's left and right turns; to practise estimating distances and angles; and to provide simple programs which the children could explore. Most of the programs involve a game in which there is a target (e.g. a lake, a house) and a randomly positioned turtle. The player attempts to reach the target using a minimum number of commands. Alternatively, the player moves the turtle around a set of objects, along a path or through a maze. Other games introduce negative numbers and cartesian co-ordinates (See Clarke & Chambers, 1985 a, b).

These games programs can be used at three levels. At the first level the game is presented as a "black box" procedure. The children load the procedure from a disk, display the instructions on the screen and play the game using the introductory Logo commands. At the second level they take the lid off the "black box" and look at the procedures which were used to create the game. They can modify the procedures, making simple changes to colours, shapes and instructions, or they can take ideas from more sophisticated games to develop the less sophisticated ones. It is not necessary for children to fully understand all aspects of the programming to make minor modifications. However, by successfully modifying existing procedures, children develop an interest in programming and feelings of competence and confidence in their own ability to use computers meaningfully. At the third level, children create their own new games. Drawing on the ideas underlying the pre-constructed games and the knowledge acquired from playing with these procedures, children create new games which are then played and studied by other groups of children.

This turtle graphics tool kit has been successfully used with primary school children and has achieved its initial goals. However, it is essential to develop tool kits related to other curriculum areas. The emphasis within Australian schools on "approaches to teaching Logo programming" (McDougall, 1985) and the focus on turtle graphics has created an image of Logo as a programming language "for kids". One solution to this problem is to develop further sets of "black box" procedures, using a wider range of the features available to Logo programmers and to develop activities which can be used in non-mathematical areas of the curriculum. Currently, a number of procedures are being developed which have application within the language and social science curricula. Three of these programs will be described briefly: LIBRARY, STORY and QUIZ (Note 2).

LIBRARY

LIBRARY was designed to be readily used by children without any previous computing experience. It demonstrates the use of a computer as a tool to create, extend and order lists of items, by providing a set of procedures which enables children to generate a list of the books they have read. The program consists of two parts: ADD.BOOK which is used for creating lists or adding to lists and LIST.BOOK which will display or print listing of the files. When creating a file, the user is asked to enter the title of the book, the surname of the author, the initials of the author and the number of pages in the book. The listing can be output in the chronological order of entry, sorted alphabetically according to the surname of the author or the title of the book, or sorted numerically according to the number of pages in each book. Using Apple Logo on a 64K Apple II computer, the current version of the program will record information on approximately 50 books, which is generally adequate to record a term's reading.

The program can be used in three ways. Firstly, it provides an opportunity for children to learn to enter data on a file and to update it when required. Files can be stored on individual students' Logo disks, or on a class disk using each individual student's name as a file name. Listings can be readily updated and printed in library classes or language sessions.

Secondly, printed listings can serve as a basis for class discussions of the use of computers as tools to store and sort data, the types of

information about books which might be included in such a data bank, the different ways in which listings may be sorted and presented, and the types of authors and titles which children are currently reading.

Thirdly, it can be used as a model program. Once children are familiar with using LIBRARY as a set of "black box" procedures, the lid can be taken off the "black box" and the procedures displayed or printed. Children with sufficient Logo programming experience can modify the procedures by adding or eliminating procedures to record different information about their books. For example, they might want to omit the number of pages and include bibliographical details such as the place of publication, the name of the publisher and/or the year of publication. Alternatively, they can use the ideas underlying the structure of the procedures to work out how to create other types of lists for other purposes. For example, they might like to develop a set of procedures which records and sorts the names, addresses and dates of birthdays of children from a particular grade or year level.

STORY

STORY is a simple word processor also written in Apple Logo, to run on a 64K Apple II computer. It was developed to overcome some of the disadvantages experienced in using the Logo editor directly as a word processor. Most Logo users are aware of the possibility of using Apple Logo as a simple word processor by typing the text into a procedure (e.g. TO STORY) and displaying the text using the PO command (e.g. PO "STORY"). Stories can be readily edited and extended by entering edit mode and using the full range of Apple Logo editing commands. Furthermore, by using CTRL-Q in conjunction with the space bar and/or the RETURN key, paragraph indentations and blank lines can be included.

However, there are two difficulties inherent in the use of this technique. Firstly, when new words are inserted into the text, the text is not reformatted, and as the children put it "words fall off the end of the line". Essentially, it is easy to correct minor errors or to add to the end of the text, but insertions create problems. The immediate solution is to work through the text and reformat it using the edit commands and the RETURN key, but this becomes rather tedious and frustrating, creating negative rather than positive

attitudes towards using computers as word processors. Secondly, every word included in the text is added to the contents, thus reducing the available memory, making it necessary to re-boot Logo after each child has finished using the computer.

These disadvantages were overcome by writing a STORY program which reformats the work and does not store the words from the text in the contents. STORY can also include procedures to engage and disengage the printer, enabling the text to be printed without printing the control commands. Thus the program retains the benefits of the facilities available in the Logo editor, whilst overcoming the limitations of relying solely on the editing facilities.

STORY has been used with lower primary school children to develop competence and confidence in using the Logo editing commands, at the stage when they are still using Logo turtle graphics in immediate mode. Through working with STORY these children broaden their Logo experience, recognizing that Logo is applicable to language activities as well as to mathematical activities. They also gain practice in the use of the edit commands before they learn to use procedures in their own programming. It is easier for children to learn to use the editing facilities whilst working with familiar text rather than less familiar programming code. Children who have had more Logo experience can take the lid off the "black box" to look at the procedures and develop an understanding of the way a computer program is written to format and print text.

QUIZ

QUIZ was designed to enable children to create a quiz containing any number of multiple choice questions, with each question having any number of alternative answers. The tool kit consists of six sets of procedures: CREATEQ, which is used to create the quiz; RUNQ, which runs the quiz and records each participant's separate responses and total score; PLAYQ, which plays the quiz without recording any results; EDITQ, which enables the quiz to be modified; and DELETEQ, which is used to delete either the results, the quiz or both the results and the quiz.

The QUIZ program can be used at three levels. At the first level the teacher designs a quiz which the children play, finding out what they do and do not know about a particular topic. At the second level, children use the program as a set of "black box" procedures

and write a quiz which other children in the class can play. In order to write a quiz, children need to conduct their own research into the chosen topic to create a range of relevant questions and to identify both correct answers and appropriate distracters. They also need to read the documentation carefully to learn the syntax and commands required to work with the program. At the third level, they can take the lid off the "black box" and look at the procedures which are used to make up the program. At this stage they can merely look at the procedures and try to understand the way they work; they can modify the procedures to change the structure or format of the input or output; or they can use the ideas contained in these procedures to develop other programs for different purposes.

Implementation

The use of such tool kits requires three elements: clear illustrations of the content applications of each tool kit, accurate and user-friendly documentation for each tool kit, and ready availability of disks containing the procedures (Bull & Cochran, 1985). At this stage the turtle graphics games are clearly illustrated, documented and available on the disk in the "Thinking with Logo" package (Clarke & Chambers, 1985a,b) whilst the language programs are currently being developed and tested in schools.

References

Barclay, T., (1985) Interacting, using Logo, in Logo: or writing programs where the student interacts as a Logo programmer, not just as a menu selector, *Logo 85 Pre-proceedings*, M.I.T., Cambridge, Massachusetts.

Buck, L.H. & Kohn, M.H., (1985) The Logo road from algebra to calculus and beyond, *Logo 85 Pre-proceedings*, M.I.T., Cambridge, Massachusetts.

Bull, G. & Cochran, P., (1985) Extending Logo: creating tools for teachers and clinicians, *Logo 85 Pre-proceedings*, M.I.T., Cambridge, Massachusetts.

Clarke, V.A. & Chambers, S.M., (1985a) *Thinking with Logo: Apple version*, Sydney, McGraw-Hill.

Clarke, V.A. & Chambers, S.M., (1985b) *Thinking with Logo: Commodore Version*, Sydney McGraw-Hill.

Kiesler, S., Sproull, L. & Eccles, J., (1983) Second class citizens? *Psychology Today*, March.

Kiesler, S., Sproull, L. & Eccles, J., (1984) Poohalls, chips and war games: women in the culture of computing, *Psychology of Women Quarterly*.

Martin, A. (1985)(1985) Black box programs with lids that open, *Logo 85 Pre-proceedings*, M.I.T., Cambridge, Massachusetts.

McDougall, A. (1985) Approaches to teaching Logo programming, In K. Duncan and D. Harris (eds.) *Proceedings of the 4th World Conference on Computers in Education*, N.Y., IFIP.

Reinecke, I., (1983) *Microcomputers*, Melbourne, Penguin.

Notes

1. Communication from Carol Edwards during "Breakfast with the Experts: Social Issues", Fourth World Conference on Computers in Education, Norfolk, Virginia, 1/8/85.

2. The procedures for LIBRARY, STORY, and QUIZ were designed by Val Clarke and programmed by Greg Heffernan, School of Science, Deakin University.

Teaching and Learning about Recursion

Anne McDougall

Faculty of Education
Monash University
Clayton

Papert (1980) argues that Logo will enable children to encounter powerful ideas by providing an environment in which children might discover and use such ideas in programming explorations and projects of their own. While this is being observed for many of the mathematical and other concepts suggested by Papert, I have not been able to find any reports of spontaneous discovery and subsequent comprehending use of one of Logo's most interesting and powerful ideas, recursion.

If a child does "discover" recursion in a Logo procedure, it seems generally to be in the situation where he or she has forgotten to exit the definition mode before typing the procedure's name to execute it, thus unwittingly inserting a recursive call at the end of the procedure. When the procedure is executed subsequently, the result is unexpected, and there is no error message to explain what is happening - unless the memory runs out and even then the explanation is not obvious to a child who does not already have a reasonably accurate mental model of how the computer handles recursive calls. Classroom studies (Noss, 1984; Kurland & Pea, 1984) have reported that teacher assistance is required for children to develop skill and understanding with recursion in programming.

Kurland and Pea point out that "In addition to understanding recursion, the child must understand the logic and terminology governing the language's control structure." (Kurland & Pea, 1984, p.2). Perhaps it might help a learner to separate the understanding of recursion itself from understanding the computer's way of

handling it. Kurland and Pea also draw attention to earlier work indicating that students' understanding of recursion is facilitated if they work with iteration first. However their own study found no subjects able to predict accurately the outcome of a turtle procedure involving embedded recursion, and they observed that "The children were fundamentally misled by thinking of recursion as looping. ...The most pervasive problem for all children was this tendency to view all forms of recursion as iteration." (Kurland & Pea, 1984, p.6).

It would seem then to be worth seeking ways of discussing recursion with children at first independent from the computer. Noss (1984) describes one such attempt, though he too notes some difficulty in avoiding "the redundancy of the idea in simple applications where a repeat loop would suffice" (Noss, 1984, p.111).

> *The provision of a concrete representation which went beyond infinite repetition was not always easy. One group of 3 fourth-years, for example were introduced to the idea of recursion by comparing it with learning to deal a pack (a large pack) of cards. All you really have to know is how to deal the first card:*

```
TO DEAL
HAND OVER TOP CARD
TURN TO THE NEXT PERSON ON THE RIGHT
DEAL.
END
```

(Noss, 1984, p.111)

A fortuitous discovery in a children's picture book led to the approach taken in my present work with recursion and children. This approach essentially involves an exploration with the children of a literature of recursive structures and processes away from the computer, mainly provided by pictures, stories and everyday situations. By this means children might develop mental models of recursion itself, before they attempt to combine it with computing ideas in recursive programming. The recursion examples should include plenty of the full or embedded type, so that children can appreciate that recursion is "more than" iteration: while recursion does involve an element of repetition, the repeated entity need not be an exact copy of the original (although it will be related to or derived from it). Further, in recursive processes there is an element of interruption and suspense which is not present in iteration.

The example that triggered the idea appears in the children's picture book, The Cat in the Hat Comes Back (Seuss, 1958). The Cat, confronted by a cleaning task, pauses after his first attempts and declares that he cannot do it alone. He then produces another smaller cat, Little Cat A, from inside his Hat. Little Cat A in turn needs help and lifts his hat to reveal Little Cat B, who subsequently and similarly produces Little Cat C, and so on. This continues until the appearance of tiny Little Cat Z, who has in his hat some special magic. At its appearance the story moves very fast and it is not clear what the order of events is. However it seems most likely that the magic triggers the action, and each cat in turn, in reverse order, does some cleaning and returns to its appropriate hat, until the task is done and only the original Cat in the Hat remains.

Clearly it is assumed by Dr. Seuss, and by the many teachers and parents who present this story and its wonderful accompanying illustrations to children, that the process described is within the understanding of very young children. And yet here we have a 26-level embedded recursive process.

Prompted by this discovery I set off among my children's books in search of other examples. The next I found was the cover illustration on a book called Humbug Rabbit (Balian, 1974) in which a rabbit is shown leaning back against a tree reading a book called Humbug Rabbit, on whose cover is a rabbit leaning back against a tree reading ... and so on - tail recursion visible to five levels. My attention was drawn by a colleague to The Mouse and His Child (Hoban, 1976), a book for older children in which the Last Visible Dog is sought on a dog food can label bearing a picture of a dog carrying a dog food can on whose label ... etc.

Armed with these examples as well as the tried and true "third wish" (Abelson, 1982, p.33), I began to talk about recursion with a 9 year old child, my daughter, who had at the time done no Logo programming. She seemed to recognise the idea as interesting, and soon suggested an example of her own, asking whether a Christmas tree decoration shaped like a Christmas tree would be recursion.

I do not present the following as a step-by-step curriculum for teaching about recursion. Rather it is a sequential narrative of incidents from which I have learned a lot about ways in which children might think about recursion.

After a couple of months during which the matter was not mentioned, the child drew my attention to a particular mark of musical notation, occurring in several consecutive measures in a percussion part she was rehearsing. The mark means to repeat for the present measure just what was played in the previous one. She suggested that each consecutive occurrence of the mark required her to go back through one more level before "finding" the actual notes to be played, and said that reading the part felt like recursion. From then she rather took over the idea, finding more examples and telling me about them.

During the next few months this child and her younger sister, a 6 year old (to whom incidentally I had not shown the Cat in the Hat example), suggested dozens of instances of situations which they either recognised as incorporating recursion or thought might do so. I shall list some of their examples here, as they might be useful for teachers seeking ideas for situations illustrating features of recursion for children.

Two further books with recursive cover designs were The Whizzkids' Handbook No. 3 (Eldin, 1983) and a song book called Mango Spice (Connolly et al., 1981). Other stories containing self referent or recursive incidents were The BFG (Dahl, 1982), About Teddy Robinson (Robinson, 1974), Winnie-the-Pooh (Milne, 1926), Hugh Lofting's Travels of Doctor Dolittle (Perkins, 1968), My Naughty Little Sister (Edwards, 1959) and a beautiful three-level version of the Three Bears story in Ten in a Bed (Ahlberg and Amstutz, 1983). Self referent pictures included a postage stamp on which was depicted an earlier stamp design, and a Japanese fan on which a Japanese lady holding a similar fan was painted. A patchwork quilt containing patches made from fabric printed in a patchwork design provided another example. The rings in the cross section of a tree trunk, a series of hollow wooden dolls of different sizes fitting into one another, a similar set of plastic beakers and a set of plastic bags of different sizes put one into another were examples illustrating the nested or embedded aspect of the children's developing model of recursion. Looking through a magnifying glass at a magnifying glass, and preparing party games for a doll's party which was itself a game were activities identified with recursion. Recursion was mentioned in discussions of big fish eating smaller fish which in turn eat even smaller ones, and a tree bearing an apple containing a seed from which a new tree will grow.

A television show in which the stars make a television program, and a ventriloquist with a doll, itself being a ventriloquist with a smaller doll were some examples the children drew to my attention while they were watching television.

At this stage both children could recognise recursion in structures and in processes. The younger one, now turned seven, could not describe or talk about recursion at all, despite having suggested many instances incorporating its features. When I asked the older one to write me something about recursion, she listed several examples and then said that it was "too hard to write about". She did once venture a description, saying that a spiral shape was recursive "because it goes on and on and in and in".

Although they were unable to describe or define recursion, the children began to create for themselves, either in reality or in fantasy, situations involving self reference and recursion. The 9 year old plaited fine strands of a doll's hair, grouped these in threes and plaited again, and then plaited three of the resulting plaits. She held mirrors up to mirrors, and on another occasion took a photograph of herself and the camera reflected in a mirror. She completed one day's entry in her diary with a note that she ended the day by writing in her diary. The 7 year old gave her teddy bear a tiny toy teddy to hold, and described imaginary situations such as putting the dolls' house inside the children's playhouse and moving that into our house, or taking a toy ferry boat on a ferry trip.

Meanwhile the children had begun learning Logo. Both learned to use the repeat command, and I deliberately avoided the use of tail recursion for iterative situations. The younger child has had no encounters with recursion in her Logo programming. The older one's first awareness of recursion in Logo occurred when she recognised it in a procedure we were copying from a book (McDougall et al., 1982). The procedure was a list processing one used in a message coding and decoding procedure.

```
TO GETCHAR :NUMB :LNAME
IF :LNAME = [] [STOP]
IF :NUMB = 1 [OUTPUT FIRST :LNAME]
OUTPUT GETCHAR :NUMB-1 BUTFIRST :LNAME
END
```

As she began to read the fourth line of the procedure she exclaimed "Hey! That's recursion. This is TO GETCHAR." On several subsequent occasions she similarly recognised recursive lines in ready-written procedures. However my efforts in explaining how these procedures worked met with very limited success, and she remained puzzled through two sessions during which I tried to help her write a turtle procedure involving embedded recursion.

I puzzled for some time about how to help a child who seemed to have a reasonably accurate model of recursion itself, to understand something of the ways in which recursive computer procedures work. I decided to try writing The Cat in the Hat in Logo. The procedure I wrote had Logo-like syntax, since the child was familiar with this, but contained some features which would not in fact run in Logo (such as the use of +1 to indicate the next letter in the recursive call).

```
TO HAT.CAT :CATLETTER
IF :CATLETTER = Z PRODUCE.MAGIC RETURN. TO.HAT STOP
HAT.CAT :CATLETTER+1
CLEAN.SOME RETURN .TO.HAT
END
```

And for this child this provided a break-through. We talked through this procedure and drew diagrams of the new copies of the procedure generated by the recursive calls. Perhaps her familiarity with the recursive situation being described allowed her to concentrate more clearly on the computing processes involved.

At the time of writing, the child has still not written independently a Logo procedure using recursion. However she is currently enjoying solving what she calls Logo recursion puzzles - reading Logo procedures that contain recursive calls, and predicting the outcomes.

I shall attempt to summarise what I have learnt from this work so far. Children who cannot define or describe recursion or write programs using it, may nevertheless be able to recognise, devise, and interpret recursive structures and processes. It is possible, and perhaps helpful, to present to children the idea of recursion away from the computer and independently of computer programming. Examples of recursive structures and processes found in pictures, stories and everyday situations in the children's experience are valuable for this. Avoidance of early use in programming of tail

recursion for repetition might avoid confusion with iteration in children's mental models of recursion.

Acknowledgments

I am grateful for the contribution of the children who participated in this work. My understanding of recursion in programming and of related teaching and learning issues has been much enhanced by collaborative work with Tony Adams.

References

Abelson, H. (1982) *Logo for the Apple II*, Peterborough: BYTE/ McGraw-Hill.

Ahlberg, A. and Amstutz, A. (1983) *Ten in a Bed*, London: Granada.

Balian, L. (1974) *Humbug Rabbit*, Nashville: Abingdon..

Conolly, Y., Cameron, G., and Singham, S. (1981) *Mango Spice*, London: A&C Black, 1981.

Dahl, R. (1982) *The BFG*, Jonathon Cape.

Edwards, D. (1959) *My Naughty Little Sister*, Harmondsworth: Puffin Books.

Eldin, P. (1983) *Whizzkid's Handbook 3*, Glasgow: Armada.

Hoban, R. (1976) *The Mouse and His Child*, Harmondsworth: Puffin Books.

Kurland, M. and Pea, R. (1984) Children's Mental Models Of Recursive Logo Programs, Technical Report No. 10, Center for Children and Technology, Bank Street College, New York..

McDougall, A., Adams, T., and Adams, P. (1982) *Learning Logo on the Apple II*, Sydney: Prentice-Hall.

Milne, A. (1926) *Winnie-the-Pooh*, London: Methuen.

Noss, R. (1984) Children Learning Logo Programming, Interim Report No. 2 of the Chiltern Logo Project, Advisory Unit for Computer Based Education, Hatfield.

Papert, S. (1980) *Mindstorms: Children, Computers and Powerful Ideas*, Brighton: Harvester.

Perkins, A. (1968) *Hugh Lofting's Travels of Doctor Dolittle*, London: Collins.

Robinson, J. (1974) *About Teddy Robinson,* Harmondsworth: Puffin Books.

Seuss, Dr. (1958) *The Cat in the Hat Comes Back,* New York: Random House.

Logo in Preservice Teacher Training: An Australian Experiment

Anthony Jones

School of Education
La Trobe University
Bundoora, Vic. 3083

Introduction

Preparing teachers to make effective use of computers in their classrooms has not turned out to be as easy a task as many teacher educators originally expected. For most teachers, an introduction to computing will involve moving into areas of content and technology with which they are unfamiliar. The problem is to provide sufficient knowledge and skills to enable the use of computers without overwhelming the teacher with too much unnecessary information.

1. The Australian Scene

1.1 Training of Secondary Teachers

The majority of teachers in post-primary schools throughout Australia receive their teacher training as an end-on course after they have completed academic or vocational qualifications. For example, a typical English and History teacher will graduate from university with a Bachelor of Arts degree, and then will enrol for a one year Diploma in Education at a university or college. Similarly, teachers of vocational subjects such as carpentry or typing will complete a period of study and work experience before they undertake any teacher training.

The Diploma in Education, the pre-service teacher training course for most Australian secondary teachers, attempts in one year to

take people with very diverse backgrounds and qualifications and to prepare them to operate efficiently in the classroom. It is not therefore surprising that the major interest of the trainee teachers is in practical matters - how do I control a class or design a curriculum or plan a lesson? Many Diploma in Education students never see computers being used in the subjects they are being trained to teach, either during teaching practice or in teaching methods classes.

1.2 Computing in Australian Secondary Schools

While it is true that in 1985 there will be very few secondary schools throughout Australia that will not have at least one computer, it is not true to say that computer education is widespread. In many schools there is no organised attempt to provide even the rudiments of computer awareness for every student. Far too often computing is still seen as the province of the mathematics department, and teachers of other subjects feel that they are not welcome when they try to use computers in their teaching.

The previous paragraph describes the worst aspects of the present state of computing in Australia. However there is much being done that is innovative and educationally excellent. The educational importance of computer education has been recognised by both the federal government and the various state governments. In Australia secondary education is constitutionally under the control of the states. Despite this, the present federal government has made available some A$18 million for the 1984-1986 triennium. It must be noted however that a 1983 report from a federal government commission stated that an amount of A$125 million needed to be spent over a five year period in order to develop a national computer education program.

2. Teacher Training

2.1 Preservice and Inservice

Over the past few years several writers have described various approaches to training teachers to teach about computers. Brooking describes a one year full-time retraining course being trialled in the U.K. Qualified teachers from a variety of teaching backgrounds were involved in a 'mini' computer science course. The course represented a very specialised form of inservice training, because only experienced teachers participated. The aim was to take trained

teachers and to retrain them so that they could return to the school system as competent teachers of computing (Brooking, 1983).

In the U.S.A. Taylor and Poirot have reported on the state of accreditation of secondary school teachers of computer science in 1983. The courses surveyed, and the curriculum the authors propose, are preservice rather than inservice, but relate only to the preparation of specialist computing teachers. Several other authors have described general computer education inservice training (for example Anderson, 1982, Bevis, 1982, McGee, 1983, Russell, 1982) or the preservice preparation of secondary computer science teachers (for example Kull and Archambault, 1984, Martin and Heller, 1984). For the teacher educator involved in training teachers to use computers there are some important differences between the requirements of preservice and inservice courses. Perhaps the major difference can be described as a lack of educational expertise among preservice teacher trainees. When conducting an inservice course for practising teachers, basic assumptions can be made about the teachers' experience with certain teaching techniques, knowledge of the organisation and content of curricula, and understanding of the capabilities of students.

Experienced computer educators will also know that there is usually a significant difference between the approach to computing of novice teachers and of experienced teachers. Some research into inservice education concludes that the content of the course must relate to the needs of the participants. (For example Spears, 1981 and Joslin, 1980.) McDougall, discussing approaches to teaching Logo as a programming language, suggests that teachers develop preferred methodological techniques because of particular curricular aims, the type of student being taught, as well as a number of other factors. (McDougall, 1985) Preservice trainees can not be expected to have developed their own perceptions based on practical experiences of classroom interaction between student and teacher.

While teachers attending a computing inservice will try to incorporate what they are learning into their individual models of schools and learning, this is not as noticeable at the preservice level. Preservice trainees appear to be influenced by memories of their own schooling and by their most recent university or college experiences. The latter usually seems to involve the concept of lecturing, and this is often the preferred mode of teaching adopted

by students at the start of a preservice course. Educational computing seems best learned in a non lecture situation, because among other things operational functionality in computing involves the acquisition of a number of skills that must be developed and practised by the learner. In practical terms this means that educational computing courses must include practical, or hands-on, experiences.

A number of reports have been published concerning the use of Logo as a medium for introducing teachers to computing as an inservice activity (for example Russell, 1983 and Fisher, 1984.) However there have been few reports of the use of Logo at the preservice level as an introduction to computer awareness rather than as one aspect of mathematics education. The remainder of this paper will consider some objectives of a preservice computing course for secondary teachers and will provide explanations and examples of how Logo is used in the introductory sessions.

2.2 Minimum Competencies

Over the past five years several groups have published recommendations about the knowledge and skills teachers should possess in order to effectively use computers in schools. In 1981 the Association for Computing Machinery, through its special interest groups on Computer Science Education and on Computer Uses in Education, published a report that considered many aspect of computer education in primary and secondary schools. One section of this report concentrated on teacher education, and proposed computing competencies for all teachers, additional competencies for teachers of computing subjects, and finally some subject specific competencies. As well as detailing particular competencies, topics of study for the various competencies were provided.

In Australia, the National Advisory Committee on Computers in its July 1983 report identified twelve groups of teachers requiring different sets of computing competencies. A list of appropriate skills and understandings was provided for each group. The report also detailed a list of fifty two recommendations, at least nine of which specifically related to teacher education in computing. (Commonwealth Schools Commission 1983)

As part of the research for the course, many practising teachers have been interviewed about their perception of teacher needs in computer education. This research is continuing, and during 1985 it is planned to question several hundred teachers, preservice teacher trainees, and teacher educators. The objectives selected for the course are based on a preliminary analysis of these interviews and the competencies determined recently by groups such as ACM and the Schools Commission.

3. Using Logo to Achieve Course Objectives

3.1 Objectives

In very broad terms the course currently being trialled aims to achieve competence in three major areas. The first aim is to enable all students to acquire the necessary skills and confidence so that they can successfully operate a microcomputer. It is hoped that the students will be capable of planning and teaching lessons using computers during their practice teaching rounds. One measure of the success or otherwise of the course might be the proportion of students who actually take formal lessons using computers in some way.

The second broad aim of the course is to introduce the trainee teachers to examples of software currently being used widely in Victorian secondary schools. In the introductory section of the course this will be a general overview, and will concentrate on software that has applications in several subject areas. Logo and word processing are two examples of the type of software to be considered.

Finally the course aims to introduce students to the fundamentals of software evaluation from an educational point of view.

3.2 Why Logo?

When computer educators plan introductory courses for teachers, there are several options available for the teachers' first experiences. The emphasis can be on the computer and its components and peripherals, on the keyboard and keyboarding skills, or on the software being run. Because of the non technical background of most students in preservice teacher education courses, a decision has been made to concentrate on software. As far as possible hardware and software concepts will be treated incidentally as they are required.

In Victorian schools computing is taught on a variety of computers, including Apple, Atari, BBC and Commodore 64. Because any of

these could be encountered during practice teaching, it is more logical to look for something more common than hardware. As the ability to run a full version of Logo is a requirement of any hardware nominated by the state Department of Education, Logo is common to all approved classroom computers.

Russell suggests that another factor that must be considered is the ability of whatever is used to provide teachers with the opportunity to reflect on what and how they are learning, and to later use these personal experiences with students. Computing is one of the few areas where teachers have the opportunity for such reflection.

The approaches to learning with Logo as outlined by Seymour Papert (Papert 1980) are different from traditional classroom practices. Teachers and student teachers who are introduced to Logo have an opportunity to experience learning as both a teacher and a student. "Learning and teaching are intertwined and become a process of developing debugging aids as knowledge gaps are discovered and filled in. (Solomon 1982 p.208)

While Logo has undoubted advantages, there are also some problems. When it is taught as an exercise in graphics or as a simple syntactical programming language, teachers often query Logo's educational value in the classroom. It is not possible to isolate the Logo language from the Logo philosophy and to still retain the essence of either. This means that teachers need to be motivated to explore Logo for themselves, rather than be presented with a recipe for teaching it.

Another problem that may occur in a Logo course for teachers is the "Logo wall" (Martin and Heller 1984). This is an intellectual barrier that separated the more concrete concepts of turtle graphics from the abstractness of recursion, number manipulation and list processing. For some teachers without a computing background the "Logo wall" can appear insurmountable.

3.3 Some Brief Course Details

The first hands-on session aims to introduce the students to the computer. Students are paired then given a Logo disk and a computer. Various hardware components are described, disk use and care outlined, and turning on and off demonstrated. After practising turning on and off and booting a disk, the students are introduced to the fundamentals of Logo and then left to explore movement, direction and screen size.

For the next session students bring in a blank disk and are provided with a System Master disk as well as Logo. Some disk operating system commands are demonstrated, and the students initialise their personal disks. Further exploration in Logo, initially looking for the solution to some simple problem, follows.

In subsequent sessions students proceed through a range of Logo activities. In all of these introductory sessions the practical activities are based on Logo. Away from the computers, discussion of the Logo activities is used to get students to think and comment on the way they are learning and how this might be relevant to teaching in general.

4. Conclusion

At this stage only a small number of teacher trainees has progressed through the course. Evaluative comments from these students have been very encouraging about the use of Logo as an introduction to computing. By the end of 1985 several hundred students will have taken the course, and a more searching form of evaluation will be carried out. However the initial results suggest that participants will leave the course with a favourable view of educational computing and in their personal capability of using a computer as one part of their teaching methodology.

References

Anderson, J. (1982) "New Directions in Teacher Education", *Computer Education* 41 June, pp. 9-10.

Bevis, G. (1982) "INSET - A Way to Meet an Urgent Requirement in Teacher Training". *Computer Education* 41 June, pp. 12-13.

Brooking, A.G. (1983) "The Problem of Producing Teachers with Computing Expertise within the School System", *SIGCSE Bulletin, 15*(3), September, pp. 13-19.

Commonwealth Schools Commission (1983) *Teaching, Learning and Computers*, Report of the National Advisory Committee on Computers in Schools, Commonwealth Schools Commission.

Joslin, P.A. (1980) *Inservice Teacher Education: A Meta-Analysis of the Research*, Ed.D. Dissertation, University of Minnesota.

McDougall, A. (1985) "Approaches to Teaching Logo Programming" in *Proceedings of World Conference on Computer Education.*

McGee, P. (1983) "The RSA Teachers' Certificates in computing", *Computer Education,* 44 June, pp. 25-26.

Martin, C. and Heller, P. (1984) "Teaching Logo to the Teachers", *Informatics And Teacher Training,* North-Holland, pp. 73-82.

Papert, S. (1980) *Mindstorms: Children, Computers and Powerful Ideas,* Harvester Press.

Russell, S.J. (1983) "Teachers and Computers: Reflections on Learning", *The Journal of Staff Development,* 4(2), November, pp. 101-107.

Solomon, C. (1982) "Introducing Logo to Children", BYTE August, pp. 196-208.

Spears, P.W. (1981) *Effective Elements of In-Service Programs as Perceived by Indiana Public School Teachers,* Ed.D. Dissertation Ball State University.

Stevens, D.J. (1984) "Microcomputers: An Educational Challenge", *Computing Education 8*(2), pp. 263-267.

Taylor, H.G. and Poirot, J.L. (1984) "A Proposed Computer Education Curriculum for Secondary School Teachers", *SIGCSE Bulletin,* 16(1) February, pp. 115-118.

'...In Four Easy Lessons': The Evolution of an External Studies Logo Course

Peter J. Carter

Magill Campus
South Australian College of Advanced Education

One of the tasks presented to me when I was appointed to the College was the writing of the Logo section of the course for the Graduate Diploma in Instructional Uses of Computers. Prior to this, students had been given a quick look at whatever version of Logo had been available, usually the Wollongong version, hardly sufficient to make people aware of the potential. Since the course was being offered externally, there had to be a structured course, and the materials had to be written.

The section of the course was to take one term, meaning that there would be four Modules. No decision required there. What went into the Modules would be a different matter.

After some thought, I decided on the following general scheme:

Module 1: Introduction, turtle graphics, editing, procedures and variables, loading and saving, bibliography.

Module 2: More graphics, IF...THEN...ELSE, recursion, with graphics examples.

Module 3: Local vs free variables, use of OUTPUT, logical operators, list processing.

Module 4: Special features: music, sprites, property lists etc.

With that, a reading of *Mindstorms* and some articles from *Turtle News*. A book such as Abelson's *Apple Logo* or *Learning Logo on the Apple II* by McDougall *et al* was to be strongly recommended.

It was my aim to move past the straightforward turtle graphics rapidly so that students could spend as much time as possible on the more distinctive and difficult aspects of Logo. There is a danger in such an approach, of course, since teachers in primary schools may never use property lists or complex list handling with their students, and will see little relevance in them. It is my view however, that they can be of more use in their classrooms if they have an understanding of those things, because if nothing else, the experience in solving problems and handling the syntax can help them assist their students. In my own experience I had known a Year 7 student who wrote no graphics procedures for the entire year, he was interested only in text handling and wrote conversational programs. I have no doubt he was not unique, and therefore regard it as essential that all teachers have some expertise in all aspects.

All Modules were to have Activities, which are not marked, and Tasks, which are. I resolved to have a range of tasks, at adult level, which would give the students plenty of scope for their imagination. To complicate matters, everything was to be useable with either Apple Logo or Commodore Logo. It was pointed out to me that external studies students had no one to turn to, they were isolated and therefore everything had to be spelled out in detail for them. That is not a view to which I subscribe entirely. One of the virtues of Logo is the element of the unexpected, and there is a danger of, as Abelson and diSessa put it, "pre-made (already discovered)" truths. I wanted students to experience both the joys of discovery and the need to think carefully with Logo.

As the writing progressed I gave drafts to colleagues in schools, asking them to try them and comment. Module 1 soon found its way into the hands of Year 7 students who apparently found it useful. There were no other comments. There must be a moral there.

Towards the end of 1984 Apple Logo II was released. Its minor quirks needed pointing out, but it was otherwise easy to incorporate. The promised BBC Logo was another matter. We had a prerelease version, courtesy of the Adelaide retailer, but virtually no documentation. The definitive implementation was promised 'Real Soon Now'. All the easy things, FD, BK and the like needed no working out, and there was enough documentation to use the editor, but of the list functions, the property lists, multiple turtles, and the other features, not a word.

By January 1985, with printer's deadlines approaching, life became hectic. There was no option but to go to press with very sketchy details of the BBC version. Fortunately only two students were to use it, they were close to each other and not too far from Adelaide. The first users were our third year students.

Module 1 was quite well received. On their evaluation sheets students wrote comments like: 'logical and easy to follow', 'well prepared and presented', 'good introduction for beginners'. That was pleasing, but their work frequently showed traces of thinking from that other language. The idea of separate procedures for separate tasks seemed new to many of them, even though they had all done some Pascal. One of the set tasks was to 'Draw a street', using the inevitable triangle/square house procedure. Wrote one student: 'What is meant by a street?'.

The second Module brought a greater range of reaction, particularly with the concept of recursion. One student was so frustrated by it he responded 'I find Logo rather tedious and unexciting!', and suggested that it be dropped from the course. Another was just the opposite: 'Logo is very exciting... I have enjoyed working through each exercise - sometimes frustrating but always enjoyable.' They must have been a patriotic group, as almost without exception they chose to draw an Australian flag, with mixed results. The influence of years of that other language was now really showing and there were procedures 50-60 lines long in some students' work. No one tried designing recursive procedures from scratch.

I knew that Module 3 would be the real separator, and sent out some extra examples with it. They found it heavy going though, and said so. 'Most frustrating', 'very chalenging' (*sic*), 'big leap from Module 2', 'I'm confused by local/free, OP'. One claimed to have spent 720 hours on it. Revision and expansion was clearly needed.

With the special features of the four versions Module 4 promised to be disjointed, and a sentence on its first page warned students of that. After reading comments I concluded that students don't read any more than the Assignment page: one Apple Logo user wanted to know what he was supposed to do for the Commodore Sprite and the Apple Logo II file exercises, while a Commodore user rang at least twice to ask how he was to use property lists. Despite that,

there were some good musical offerings and a couple of interesting sprite programs. One student wrote a program to translate English to Tagalog.

The second edition is now going out to second year students, who have the choice of Logo or Pascal. Since Module 3 had caused the most difficulties, it now deals only with list handling. The other items have been moved to Module 2 and aspects of graphics have been moved to Module 1. More examples have been included. Module 4, now that we have BBC manuals, includes work with its multiple turtles and other features.

The exercise has shown that most people take fairly readily to turtle and graphics, with the quality of their programming depending on their previous experience. It is with lists and recursion that difficulties come. That is no surprise. Anyone who has taught Logo has always realised this, and it points up the need for adequate support materials and tutoring. For internal students that can be readily arranged, for remote students it is difficult, and one of the reasons I have begun the newsletter *POALL*, through which I hope there will be considerable sharing of ideas and inspiration.

Girls and Computing

Valerie A. Clarke

Division of Cognitive Science and Psychology
Deakin University
Victoria

There is a growing concern that women and girls are not participating equally with men and boys in the benefits arising from the rapid introduction of computers into society. Within the workplace, women predominate in the unskilled data entry sections, doing the relatively boring, repetitive, low status tasks, whilst men predominate in programming, systems analysis and systems operation where the work is of higher status, more highly paid, more stimulating and more creative (Game & Pringle, 1983). If women are to have equal opportunities within the labour market, they need similar training to men. However, surveys show that women and girls are under-represented in formal training programs (E.O.C., 1983; Firkin, 1984; Watt, 1982) and in informal computing activities (Acorn, 1984; Muira and Hess, 1984).

The question arises as to why these sex differences are occurring. Essentially, it is argued that the problem stems from the cultural context in which computers are being introduced into society, rather than from any quality inherent in computers. The clearest message emerging from the computer culture is that computers are for men and boys, rather than for women and girls, that society sex-types computing as a male activity (see below).

The important effects of the social context in which computers are being introduced into schools were demonstrated by Clarke (1985a) who showed that girls in a single-sex primary school reacted to computers differently from children in a co-educational school. The children in the co-educational school sex-typed computing as a male

activity, such sex-typing being more evident at higher than lower grade levels, whereas the girls in the single-sex school sex-typed computing as a female activity. This sex-typing was reflected in both the children's general attitudes to computing and their knowledge of computing.

Sex-typing of computing, like all social attitudes, is learned. Four processes, which help to explain how this learning occurs, are modelling, direct experience, social reinforcement and generalisation from other pre-existing attitudes.

Modelling

Children are continually being provided with male role models operating computers. At the 1985 Australian Computer Education Conference all members of the official party present at the Opening Ceremony were male. Similarly, at the Opening Ceremony of the Fourth World Conference on Computers in Education, 1985, ten of the eleven members of the official party at the Opening Ceremony were male, and all speakers were male.

The image presented at these conferences is equally prevalent in the media and within the classroom. The models evident in illustrations in magazines (Lockheed & Frakt, 1984), on television programs and in computer stores (Kiesler, Sproull & Eccles, 1984) are predominantly male. Within school classrooms, computers are generally introduced by male teachers who have a personal interest in computers which they see as "machines for men and boys" (E.O.C., 1983, p.7), a bias reflected in their tendency to choose boys as "volunteers" to operate the machines (Fisher, 1984). At home, fathers and sons are more likely to use computers than are mothers and daughters (Acorn, 1984).

Direct Experience

Girls have less direct experience with computers. In relation to boys, girls are less likely to enrol in formal computer classes (E.O.C., 1983; Firkin, 1984; Watt, 1980), computer camps (Muira & Hess, 1984), and computer clubs (Acorn, 1984; E.O.C., 1983) and are less likely to own and use home computers (Lockheed & Frakt, 1984; Acorn, 1984). Within school classrooms, they are less likely to gain access to the computers. In mixed-sex classes boys demand, and receive, more than their fair share of resources, including books, space,

teacher attention and computers (Becker, 1981; Firkin, 1984; Fisher, 1984; Spender & Sara, 1980; Whyte, 1984). Within computer classes where there are usually considerably fewer computers than children, boys tend to be competitive and aggressive, rapidly deciding to move in and take over the machine whilst the girls tend to defer to the boys, preferring to avoid hostility and competition (Boss, 1982; Davidson & Johnson, 1985; Firkin, 1985; Fisher, 1984).

Due to their limited access to computers, many girls lack the opportunity to discover that computers are fun, easy to operate and as suitable for girls as for boys. Because they lack the direct experience more common among boys, girls are less likely to develop positive attitudes to computing and an interest in participating in formal and informal computing activities. This is partly a practical problem and partly a resource problem. Kiesler, Sproull & Eccles (1983) suggest that if girls are given the opportunity to have direct experience with computers, they respond enthusiastically. Similarly, Campbell (1983) reports that in schools where there are sufficient computers to enable each child to have individual access, girls and boys do equally well, and both girls and boys come before school and remain after school to spend their spare time at the computers.

Reinforcement

Generally, children are reinforced for showing appropriate sex-typed behaviours (Maccoby & Jacklin, 1974). Once computers are sex typed as male, boys are reinforced for developing an interest and expertise in operating them whilst girls are not so rewarded. If teachers and parents believe that computers are predominantly for men and boys, their classroom behaviour will subtly reinforce children who conform to the appropriate sex-stereotypes. The effects of teacher expectations are well documented by Rosenthal (1968), while Whyte(1984) demonstrates that many teachers are unaware of the extent to which they sex-type certain activities and the extent to which they unconsciously foster the development of these stereotypes in children.

Within the peer group, fairly young children have clearly defined views of appropriate behaviours for boys and girls and do not hesitate to express their approval or disapproval of a peer's interests

and actions. Generally, children from kindergarten to about year nine tend to remain relatively segregated for most activities. Once an activity or an area has been identified as a "male" domain, the girls tend to avoid it. Within most schools, by male selection and female default, the computer centre tends to become identified as a male domain (Lockheed & Frakt, 1984).

Furthermore, girls' social needs may also reduce their willingness to gain direct experience with computers. Boys and girls have different social needs and different patterns of interaction (Maccoby & Jacklin, 1974). Becoming competent at computing requires a considerable expenditure of time and effort, often working alone with a computer. Whereas boys enjoy working alone and tinkering with machinery and electronic gadgetry, girls prefer activities involving social interaction. Many girls are reluctant to give up opportunities for sharing activities with other girls or to jeopardise their popularity with their peers to spend time tinkering with a computer. The indirect costs of computer involvement are thus much greater for girls than for boys who receive peer group encouragement to develop their computing expertise and are readily rewarded for demonstrating their skill (Maccoby & Jacklin, 1974; Lockheed & Frakt, 1984).

The role of reinforcement in shaping attitudes within the school context partly depends on the age at which these stereotypes are evident. Although Alvarado (1984) argues that in the U.S. such sex-typing occurs in the early years of secondary school, English (E.O.C., 1983), Scottish (Hughes, Macleod & Potts, 1984) and Australian (Clarke, 1985) data show clear evidence of sex-typing by about the middle of the primary school.

Generalisation of Attitudes

The under-representation of girls in computing activities may be partly attributed to the association of computers with mathematics, science and technology. At a tertiary level, computing departments have often emerged from mathematics departments. In schools, computers are frequently introduced by mathematics teachers within the mathematics curriculum (Burt, 1983; Steele, 1984). Pre-existing attitudes to mathematics are being generalised to computing.

Generally, the literature pertaining to the problems experienced by girls in relation to mathematics, shows that society sex-types

mathematics as a male domain in which females are unwelcome and seen as having little to contribute. At a professional level, women mathematicians experience sex-based discriminations in their training, in their professional activities, and socially, due to the prevailing stereotyped attitudes (Ernest, 1976). Fewer women enter and complete tertiary mathematics majors or postgraduate courses (Ernest, 1976; Fennema, 1977). By secondary level, mathematics has already become sex-typed as a male activity (Fennema & Sherman, 1977; Fox, 1977), such sex-typing being continually reinforced by parents, peers, teachers and counsellors (Fox, 1977). Similar problems emerge in relation to science which is also an area generally less preferred by girls, (eg. Kelly, 1982; Kelly, Whyte & Smail, 1984).

Girls are not only seeing computing in the context of mathematics and science they are also seeing it as a form of technology, something that is mechanical, akin to the "tinkering activities" associated with boys rather than girls (Whyte, 1984). Although many teachers are seeing computers as "machines for men and boys" and computer manufacturers are stressing the technical aspects of computing (Reniecke, 1983), there is no need to understand the technical aspects of a computer to operate one. Few women are concerned with the insides of their T.V. sets or the "gadgetry" under the bonnets of their cars, but they watch television and drive cars. Papert (1980, 1985) emphasises the importance of procedural and systematic thinking in the operation of computers rather than a need for technical expertise. His current concern is to direct attention away from the technocratic aspects of computing and to focus on the use of computers as tools for developing skills and solving problems.

The Broader Social Context

The problems experienced by girls in relation to computing reflect the broader issue of sexism in society. Such sexism is maintained by the differential values placed by society on particular jobs, on particular activities and on certain knowledge domains. Generally, the activities and interests defined as "female" are devalued and treated as inferior to those sex-typed as male (Gilligan, 1982; Shaw, 1984). Similarly, through the development of a male-oriented language, women are continually reminded that their interests and activities are secondary (Spender, 1983). To some extent, the introduction of computing into schools using male-oriented

software and activities and within subject areas traditionally preferred by boys is reminiscent of the earlier educational discussions of the problems of "disadvantaged" groups in schools where disadvantaged groups were essentially those sharing cultural values not shared by middle class, white, Protestant males (Berger & Luckman, 1966; Young, 1971). Similarly, it reflects the continuing debate through the psychometric literature which endeavoured to "prove" that whites had superior abilities to blacks, or that middle class children were brighter than working class children, by demonstrating that these groups gained lower scores on tests designed to be consistent with the knowledge and experience of white, middle class American children (see Jenkins & Patterson, 1961; Karier, 1972).

Ability

Another approach to the gender issue is to ignore the social context or situational determinants of behaviour and attribute girls' lack of participation in computing activities to dispositional determinants of behaviour. This argument suggests that girls do not participate because they lack the interest or the ability to do so. By attributing the gender problem to characteristics inherent within girls, administrators, teachers and parents alleviate from themselves the responsibility of addressing the issue. However, data do not support this attribution.

For example, Clarke (1985b) has demonstrated that the social context, or situational factors have more effect on the acquisition of knowledge of computing and attitudes to computing than does general ability. Using three samples of children, boys in a co-educational school, girls in a co-educational school and girls in a single-sex school, it was demonstrated that there were no initial differences between these samples on measures of general ability, attitudes to mathematics, attitudes to science, attitudes to computing or computing experience. However, after a year's computing experience at school, the boys in the co-educational school *and* the girls in the single-sex school had gained significantly higher scores on a test of computer knowledge than did the girls in the co-educational school. Furthermore, the relationships of ability and attitudes to computing performance were different for each of the three samples. For the boys, scores on the computer knowledge

test related predominantly to general school ability. For the girls at the co-educational school, scores on the knowledge test related to attitudes to computing but were not significantly related to ability. For the girls at the single sex school, both ability and attitudes related to knowledge acquisition. Therefore, it is concluded that, for girls, attitudes to computing are a more relevant factor to consider in relation to their computing performance than is their general ability. These attitude and performance differences are important as the decisions to select elective curriculum areas are often reflections of small interests developed during earlier experience. Thus the challenge facing school administrators, teachers and parents is the challenge of providing a social context in which girls will develop and maintain positive attitudes to learning both with and about computers so they can perform as well as boys on evaluations of computing competence and take an equal place in the workforce and in society.

Strategies

Strategies for increasing the interests and participation of girls in computing related activities emerge from a consideration of the processes involved in the development of the perception of computing as a male activity. Four strategies are suggested.

Firstly, it is essential to provide equal numbers of male and female role models participating in all types of computing activities. This is more easily said than done as attention must be directed towards the media, commercial interests, parents and teachers. The introduction of one or two role models or the highlighting of a few selected female figures within the history of computing may have little effect as it is easy to treat these as isolated instances, as different, as people girls would rather not model than as people they wish to imitate. At a school level it is essential to involve *all* teachers, both male and female, in the integration of computers as tools across the curriculum. At the family level it is necessary to educate parents, mothers as well as fathers, in the nature and use of this new technology.

Secondly, girls need the opportunity to gain direct experience with computers. This might involve allocating times when girls have access to computers or designating certain computers as being for

girls. It also requires the development of activities and software which have intrinsic appeal for girls.

Thirdly, parents, teachers and peers need to encourage girls' interests in computing. Attention must be focussed on males as well as females to ensure they are both providing positive social reinforcement for girls engaging in computing activities.

Fourthly, and most importantly, the computer must cease to be a primary object of study and become a tool which is used to facilitate the achievement of goals. It must become part of the home (like the T.V.) and part of the classroom (like the blackboard), being used by those who need it, when they need it and ignored when not relevant to the task in hand.

References

Acorn, (1983) Mimeographed Computer Research Notes.

Alvarado, A.J., (1984) Computer education for all students, *The Computing Teacher, 11*, 8, 14-15.

Anderson, R.E., Welch, W.W. & Harris, L.J., (1984) Inequities in opportunities for computer literacy, *The Computing Teacher, 4*, 8, 10-12.

Becker, J., (1981) Differential treatment for females and males in mathematics classes, *Journal for Research in Mathematics Education, 12*, 1, 40-83.

Berger, P. & Luckman, T., (1966) *The Social Construction of Reality*, N.Y., Doubleday.

Boss, J., (1982) Sexism among the micros, *The Computing Teacher*, Jan.

Burt, G., (1982) Computer Education in Victorian State Secondary Schools: The results of the 1981 survey of the secondary computer education committee. *Com-3, 31*, 15-21.

Campbell, P., (1983) Computers and children; eliminating discrimination before it takes hold, *Equal Play, 4*, 1, 4-7.

Clarke, V.A., (1985a) Computing in a social context, *WCCE/85 Proceedings*, Norfolk, Virginia.

Clarke, V.A., (1985b) When attitudes count, *Logo 85 Conference*, Boston, Massachusetts.

E.O.C., (1983) Equal Opportunities Commission, *Information*

Technology in Schools; Guidelines of Good Practice for Teachers. Report prepared by the London Borough of Croydon.

Ernest, J., (1976) Mathematics and Sex, *American Mathematical Monthly, 83,* 595-614.

Fennema, E.H., (1977) Influences of selected cognitive, affective and educational variables in mathematics learning and studying. In *Women and Mathematics: Research Perspectives for Change,* N.I.E. Papers in Education and Work, No. 8.

Fennema, E.H. & Sherman, J., (1977) Sex-related differences in mathematics achievement, spatial visualisation and affective factors, *American Educational Research Journal, 14,* 51-71.

Firkin, J., (1984) *Computers in Schools,* Melbourne, VISE, October.

Firkin, J., Davidson, M. & Johnson, L. (1985) *Computer Culture in the Classroom,* Melbourne, VISE.

Fisher, G., (1984) Access to Computers, *The Computing Teacher, 11,* 8, 24-27.

Fox, L.H., (1977) The effects of sex role socialization in mathematics participation and achievement. In L.H. Fox, E. Fennema, and J. Sherman (eds.), *Women and Mathematics: Research Perspectives for Change.* N.I.E. Papers in Education, No. 8.

Game, A. & Pringle, R., (1983) *Gender at Work,* Sydney, George Allen and Unwin.

Gilliland, K., (1984) Equals in computer technology, *The Computing Teacher, 11,* 8, 42-44.

Gilligan, C., (1985) *In a Different Voice,* Harvard University Press, Cambridge, Massachusetts.

Gray, J., (1980) A competitive edge: examination results and the probable limits of secondary school effectiveness, *Education Review, 33,* 1.

Hughes, M., Macleod, H. & Potts, C., (1984) *The Development of Logo for Infant School Children,* A project funded by the Scottish Education Department; April 1983-March. Unpublished report.

Jenkins, J. & Paterson, D., (1961) *Studies in Individual Differences: The Search for Intelligence,* N.Y., Appleton-Century-Crofts, Inc.

Karier, (1972) Testing for order and control in the corporate liberal state, *Educational Theory, 22*, 154-180.

Kelly, A., Whyte, J. & Smail, B., (1984) *Girls into Science and Technology: Final Report*, GIST, Department of Sociology, University of Manchester.

Kiesler, S., Sproull, L. & Eccles, J., (1983) Second class citizens?, *Psychology Today*, March.

Kiesler, S., Sproull, L. & Eccles, J., (1984) Poolhalls, chips and war games: women in the culture of computing, *Psychology of Women Quarterly*.

Lockheed, M.E. & Frakt, S.E., (1984) Sex equity: increasing girls' use of computers, *The Computing Teacher, 11*, 8, 16-18.

Maccoby, E.E. & Jacklin, C.N., (1974) *The Psychology of Sex Differences*, Stanford, Stanford University Press.

Muira, I.T. & Hess, R.D., (1984) Enrolment differences in computer camps and summer class, *The Computing Teacher, 11*, 8, 22.

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

Papert, S., (1985) Computer criticism vs. technocratic thinking, *Logo 85 Theoretical Papers*, M.I.T., Cambridge, Massachusetts, pp. 53-67.

Reinecke, I., (1983) *Microcomputers*, Melbourne, Penguin.

Rosenthal, R., (1968) Self-fulfilling prophecy, *Psychology Today*, September, 46-51.

Are Computers Sex-Stereotyped by Four-Year-Olds?

Lesley E. Tan

Melbourne College of Advanced Education
Institute of Early Childhood Development
Kew, Victoria.

Primary school children have been found to regard computers as boys' activities. The attitudes of four-year-olds to computers is reported here. Children at two preschool centres were compared - at one centre a computer had been available for children to use for a term. Children made choices between different activities for themselves and for other children. Children with no computer experience chose computers more often for boys than for girls, but those familiar with the computer did not. This indicated that sex-stereotyping of attitudes to computers may be present in four-year-old children, but that giving them the opportunity to play with computers could help to eliminate this.

As the use of microcomputers becomes a common component of primary school education, concern is being expressed increasingly as to whether the opportunities computers provide are equally available to girls and boys (Moont, 1984). Studies in Victorian primary schools (Clarke, 1985) indicate that in the typical coeducational school, computing is already sex-stereotyped. Both boys and girls see computing as a boys' activity and one at which boys will perform better than girls. Furthermore, it was found that whereas the boys' performance related mainly to intelligence, for girls it was more dependent on attitude. This sex-stereotyping was already clear by year 3 in primary school and Clarke suggested that attempts to counteract or prevent this might begin at or before school entry.

Provision of a computer in preschool programs is not common and it is only very recently that preschool children have gained any awareness of personal computers. One United States study reported that among a group of 17 preschoolers under 5, the boys used the computer much more often than the girls, although there was no difference among the 15 five-year-olds (Beeson & Williams, 1983). Swigger, Cambell and Swigger (1983) found no difference between males and females in use of a computer in a preschool. Whether such young children tend to regard computers as more suitable for boys in a wider sense has not been investigated.

In conjunction with a study of the effects of playing with computers on certain aspects of preschoolers' development, it was decided to assess children's attitudes to computers in terms of their perceived suitability for boys or girls. A comparison could be made between children in two preschool centres. In one (identified here as Centre C) a computer had been provided for the children to use for a term, in the other (Centre N) no computer had been available.

The children were attending two outer suburban Melbourne preschools which were in a similar residential area and drew children from the same general socio-economic range. The general nature and organisation of the programs were similar, providing a wide range of indoor and outdoor activities and allowing children to choose what they did for most of the time. In Centre C, an Apple microcomputer was available on most days of most weeks during the second term. The (female) teacher and the mother-on-duty provided minimal supervision and instruction. The computer was set up to run a simplified (single key) version of Logo graphics (McDougall, Adams & Adams, 1982). It contributed to the program in the same way as many of the other activities provided (e.g. painting, blocks, puzzles, home corner). Children played with it as they chose, in some cases singly, in others with peers. On some occasions the computer was in high demand, at other times, not.

All children were interviewed in October (i.e. in the case of Centre C children, approximately seven weeks after the end of the period when the computer had been in the centre). At Centre C all available children were interviewed. This group comprised 23 girls (mean age 52.7 months, s.d. 3.7 months) and 15 boys (mean age 54.2 months, s.d. 3.1 months). None of the children in either group had

a computer at home, nevertheless they all knew that it was a computer when shown a photograph of one.

The children were interviewed singly and asked to make choices between photographs showing typical preschool activities (equipment only was shown; there were no children in the photographs). Three series of six choices were made. In the first the child chose which of the two he/she preferred. In the second and third series, the child was shown a photograph of a four-year-old child of the opposite (second series) or same (third series) sex and asked to choose between the activities for that child, thus "If you were helping this child choose, which one of these he/she would rather play with - which one would you choose for him/her?"

A set of 24 photographs was used: six showing male-typed activities (blocks, digging patch, firemen, hammering, Mobilo, train set), six female-typed (dolls, dress-ups, collage, threading, home corner and dolls' house), six not gender-typed (puzzles, books, painting, swing, water play, slide) and six showing different views of an Apple microcomputer. Considerable consensus exists between researchers about sex-stereotyped pre-school play activities (e.g. Maccoby & Jacklin, 1974, Garvey, 1977, Tan, 1983) and this enabled a set of choices to be designed which would allow comparison of boys and girls. The order of presentation of photos within types (male, female, neuter, computer) was randomised, but each of the three series involved the following six pairs: male / female, neuter / computer, female / neuter, male / computer, neuter / male, computer / female.

Each child had three opportunities to choose a computer for him or herself. Table 1 shows how often this occurred. The children at Centre C who were familiar with a computer chose it more often than the children at Centre N, and there was no difference between boys and girls. At Centre N girls chose the computer less than boys but the numbers are small and the difference is not statistically significant.

Table 1: Number of times computer was chosen against other activities by children choosing for themselves.

	Boys			Girls		
	n	(%)	Total	n	(%)	Total
Centre C	25	(52)	48	38	(55)	69
Centre N	12	(33)	36	8	(22)	36

When choosing for another child of the same sex, the pattern of choices was little different from that found when children choose for themselves. From these, an indication can be obtained of whether children regarded the computer as more suitable for one sex rather than the other.

Table 2: Number of times computer was chosen against other activities by children choosing for a boy and for a girl.

	Boys			Girls			Total		
	n	(%)	Total	n	(%)	Total	n	(%)	Total
Centre C, for boy	50	(52)	96	24	(35)	69	74	(45)	165
Centre C, for girl	20	(42)	48	61	(44)	138	81	(44)	186
Centre N, for boy	27	(38)	72	13	(36)	36	40	(37)	108*
Centre N, for girl	8	(22)	36	16	(22)	72	24	(22)	108

*Comparison between choices for boy and for girl by Centre N children: $x^2=5.65$, $p<.02$.

Considering children at Centre N, it can be seen that they were significantly more likely to choose computers for boys than for girls. This was not the case for the Centre C children, who were more inclined to choose computers for their own sex than for the opposite sex but did not show any overall tendency to choose computers for boys rather than for girls.

Looking at these results another way, it can be said that the experience of having a computer as part of a preschool program did not have a great affect on children's inclination to choose a computer as a suitable play activity for a boy (45% of Centre C children compared to 37% of Centre N children). However, considering choices made for girls, there was a highly significant difference between the two centres (44% of Centre C children compared with 22% of Centre N children, $X^2=13.5$, $p<.001$) and children familiar with a computer were much more likely to regard it as an appropriate play activity for a girl.

While any conclusions we might tentatively draw are based on data from only two centres and small samples, there seems to be no other comparable data from such young children. The results obtained indicate that four-year-old children with no direct familiarity of computers may already see computers as sex-stereotyped to some extent - boys' toys rather than girls'. However,

this was not the case for children whose preschool program had included a computer as one of the available activities over a period of some weeks. Clarke (1985) suggested that in order to prevent the idea of computers as a boys' activity becoming established during primary school years, appropriate action should be taken from the preschool period onward. This study supports that view. However, a comment on the aims of including computers in preschool programs should be made. The primary purpose of activities in an early education programs is to promote optimal development in a balanced fashion across all areas. In my view the countering of sex-stereotyping, while desirable, would of itself be only secondary to this.

References

Beeson, B.S. & Williams, R.A., (1983) The effects of gender and age on preschool children's choice of the computer as a child selected activity. ERIC report, ED233810.

Clarke, V. (1983) Radio interview, "Warm boot", ABC, Melbourne, 12 March.

Garvey, C., (1977) *Play*. London: Open Books.

Maccoby, E.E. & Jacklin, C.N. (1974) *The Psychology of Sex Differences*. Stanford, Calif: Stanford University Press.

McDougall, A., Adams, T. & Adams, P., (1982) *Learning Logo on the Apple II*, Melbourne, Prentice-Hall.

Moont, S., (1984) Will women be the dropouts of the information age?, Proc. Second Australian Computer Education Conference, Sydney.

Swigger, K.M., Campbell, J. & Swigger, B.K. (1983) Preschool children's preference for different types of CAI programs, *Educational Computer Magazine*, Jan/Feb, pp.38-40.

Tan, L.E., (1983) The effect of an older sibling on preschoolers' sex-role development. Unpublished.

Acknowledgment

The assistance of Pauline Adams with this project is very much appreciated.

Various Computers in the Kindergarten

Pauline Adams

This paper describes three different micro-computers and Big Trak being used in the pre-school environment. The micro-computers were an Atari 800, an Apple IIe and a TRS 80. A single key Logo was used on each computer. In 1980, I read Papert's paper called "Redefining Childhood" and found it exciting and stimulating, particularly the following:

> *Children seem to be innately gifted learners, acquiring long before they go to school a vast quantity of knowledge by a process I call 'Piagetian learning' or 'learning without being taught'. For example, children learn to speak, learn the intuitive geometry needed to get around in space, and learn enough logic and rhetorics to get around parents - all this without being 'taught'.*
>
> (Papert 1980, p.7)

As a teacher of children in these pre-school years, I believe I have a responsibility to provide the best possible learning environment for the children. This environment includes all the regular kindergarten activities, such as the blocks, home corner, painting, and play dough. This environment can be extended by including a micro-computer and using the computer language, Logo. Papert, in *Mindstorms: Children, Computers and Powerful Ideas*, describes the Logo environment as follows:

> *In most contemporary educational situations where children come into contact with computers, the computer is used to put children through their paces, to provide feedback, and to dispense information. The computer programming the child. In the Logo environment the relationship is reversed: The child, even at pre-school ages, is in control; the child programs the*

> *computer. And in teaching the computer how to think, children embark on an exploration about how they themselves think.*
>
> (Papert 1980, p.19)

When I introduced the Atari and the Apple, I wrote a program in Logo for the children to use single key commands to instruct the turtle. To instruct the turtle to move forward ten turtle steps, it is necessary to press six keys, the F, D, space bar, 1, 0 and RETURN. This is a lot to remember and to press in the correct order, it is much easier to press one key, F, to move the turtle the same distance.

I chose the single key F (FORWARD 10), R (RIGHT 30) and B (BACK 10) because their relationship to the full Logo command made them easy for adults to recall. Other keys used were W (CLEARSCREEN), C (CIRCLE), T (TRIANGLE), S (SQUARE), and A (NAME). All these keys are on the left of the keyboard, this limits the search area for the children. These letters were put on a card on the computer just above the keyboard.

In 1983, at the Monbulk Pre-School, I introduced Big Trak, a computerised vehicle, to the C Group. These children were four and five years old and came to kinder two days a week for two and a half hours.

Big Trak can be instructed to move forward, turn, flash lights, and sound a horn. When the instructions are given, the operator then presses "GO" and Big Trak carries out the instructions. More than one instruction can be given at a time. The children were shown which part to press and then proceeded to drive Big Trak around the room. Tunnels and roads were built with the blocks in the block corner and there was a lot of fun and exploration in driving Big Trak around. Over six weeks, there was one child in particular, who was fascinated and enthralled by Big Trak. He quickly became the tutor and taught many children which buttons to press and how many times. He was also very involved in designing the tunnels etc. After six weeks of constant use, the back axle broke. This is apparently a weakness in the design.

During this time, I took the Atari 800 to the A-Group. These children were also four and five years old, however they came to kinder four times a week for two and a half hours. This group of children had the computer for four sessions only. I included single key

commands to change the background colour. For this, I put coloured insulation tape on the keys for easy identification.

In 1984, I took an Apple micro-computer to the Greenwood Park Kindergarten for Term 2 with both groups of children. The Adams family Apple II was used to introduce the computer to the children, then an Apple IIe was hired from the Computer Education Group of Victoria.

The repeating function on the IIe keyboard made it difficult for the children to control the computer. After one session a technician from a local computer shop disabled it. Since then, other teachers have had a switch connected, so that the function can be turned on and off.

The computer was set up in an area next to the puzzle table and opposite the library and construction areas. It is important that the monitor does not face the window and reflect the outside scenery. The children were introduced to the computer two at a time and the names of the other children were recorded and they were called when it was their turn. The children crowded around at the beginning of the first session; however they were happy to come back later, even the next day, when they were reassured the computer would be back for more sessions. During the group time on the first day the Apple was put on the floor and the top taken off for all to see inside it. Each part was named, such as the monitor, disk drive, diskette and processor.

During the introductory sessions, I was able to sit with the children all the time, because the full-time teacher was present. This meant each child could have as much attention as he/she required. This changed while I was the relieving teacher. The assistant and willing mothers, who had on the job training and later attended a two hour workshop, assisted the children. When the full-time teacher returned, after three weeks, she set up the computer each day and assisted the children with the help of the mothers and the assistant and I called in whenever possible.

The first time a child used the computer, he/she was shown the A key. When this key was pressed the question "WHAT IS YOUR NAME?" appeared at the bottom, left side of the screen. The child then typed his/her name, e.g. KATE, initially with some adult help, however the children quickly learnt to type their own names. The computer responded with "HELLO KATE". The F key was then

shown to the child to move the turtle forward and the R key to turn it. The child first moved the turtle forward until it went off the screen. If the turtle is turned to a different angle, it will then move off the screen and reappear on a different part. This can be lots of fun filling up the screen with lines. When the child was ready to clear the screen, W was introduced. The next keys introduced were C to draw a circle, T to draw a triangle and S to draw a square.

Some children watched for a long time before using the computer. One child, when he came to his turn, pressed R, then T many times and made an interesting pattern. Another child moved the turtle within the screen and turned at right angles all the time. This required careful pressing of the keys and checking after each move to see if the turtle was heading the right way.

Other children learnt to use additional keys, such as L to turn left 30 degrees and U to lift the pen up and D to put the pen down, also much fun can be had using E to erase the lines drawn. Many children enjoyed typing letters and numbers. They quickly learnt that by pressing A, they could type in any letter, number or symbol until they pressed the RETURN key. For example, the following might appear:

QQQWWWEERRRTTYY123456789801902334ZZXX/;/;P QWE

Some children would try typing names and real words. After several sessions when every child had had as many turns as he/she wished, the demand for the computer fluctuated, from no child, to one or two children. Those that did use the computer, seemed to spend more time alone at the computer.

The computer was set up on a table as in an office. A typewriter and telephone were on the next table. Later an electric calculator was added on another table. Office clothes were put out to dress-up, also paper, pencils and a telephone book were added. The whole area was popular, the dramatic play stimulating, the language and interaction between the children very exciting. The children had no difficulty coping with the keyboard and finding the keys. One day the teacher forgot to put out the card with the letters on. The third child to use the computer came and asked for the card. The first two children had remembered which keys to use and did not need the card.

At the same time as the children were using the computer, the teacher was saving their work on a disk. These files can be printed out after the session and the teacher can see what each child has done.

In Term 3, I introduced the Tandy TRS 80 Color Computer to both groups of children at the Heathmont Baptist Pre-School. These children were four and five years old and attending four times a week for two and a half hours.

Included in TRS 80 Color Logo, there is a feature called Doodle Mode, which uses a single key to direct the turtle. The single keys used are the number keys on the top row of the keyboard, and there is an overlay to put on the keyboard. In the first session, the children were shown how to move and turn the turtle and then left to try it on their own. It is possible to include the child's name with his work. At the end of a session the child's work could be recalled to the screen. This was a most important feature in public relations with the parents.

In 1985, I took the Apple II to the Templeton Orchards Pre-School in Term 2 for six weeks. These children were just turning four years old attending kinder twice a week for two and a half hours. I used the single key commands and I did not have the same time to spend with each child. This did not seem to matter, as the children quickly taught each other. One day, I took a BBC and after a single "The computer is different", the children carried on as before.

All the children through their own efforts, started learning concepts of space, direction and distance. One of the most important results however, was the necessary planning and discussion that took place between each pair of children at the computer, development of communication skills and co-operation between each other.

I believe the value of the projects can best be measured through observation of the amount of time the children chose to stay at the computer, the instructions they were giving the computer, as well as the amount of communication between the child operating and the child observing. Most children spent between five and ten minutes with the computer at any one time, often returning later in the same session to try again, after others had had a turn.

The fact that the children chose when they would use it and when they wanted other activities seems to point to the possibility that with a wide variety of experiences available, children will self regulate their learning to their stage of development. The computer became another part of that learning process.

In each instance, the children were confident and capable with the computers. It was very exciting to see the learning taking place, the language being used and the co-operation between the children, however the most exciting happening is the children's complete lack of fear and apprehension.

There are parents and professionals who believe that micro-computers should not be used in the pre-school environment. There is a fear that the computer will de-humanise the child and turn him/her into a zombie, unable to think and be independent and unable to create. It is seen as a passive activity. The teachers who have used computers have all seen the active children, thinking, solving problems, talking and working together.

I believe children can and will use computers in the kindergarten as one of the many activities available to them. The children found computers to be fun, had very little difficulty with the keyboard and they certainly do not have the fears many adults have. In a report on a recent study in rural New South Wales, Alison Elliott and Neil Hall say:

> *Given the media hype that surrounds the microcomputer industry and the increasing popularity of microcomputers for everyday purposes it does not seem at all surprising that the small group of three, four and five year old children that we surveyed had absorbed so much about computers and felt so positive about using them.*
>
> (Elliott and Hall, 1985, p.31)

Acknowledgments

I acknowledge the support, help and co-operation of the directors, the assistants, the parents and children of the Greenwood Park Kindergarten, the Monbulk Pre-School, the Heathmont Baptist Pre-School and the Templeton Orchards Pre-School; the Computer Education Group of Victoria for the Apple computer; Mr. John Barber at Megatron and his able helpers for fixing, very promptly, the many and varied hardware problems; and Mrs. Lesley Tan, Senior Lecturer, M.C.A.E. Institute of Early Childhood Development, for professional advice.

References

Adams, T., Adams, P. and McDougall, A. (1984) *Learning Logo on the TRS-80 Color Computer*, Sydney, Prentice-Hall.

Elliott, A. (1984) "Microcomputers in the Pre-School? Thinking it Through", in Hughes, J. (ed.) *Computer and Education - Dreams and Reality*, Sydney, CEGNSW.

Elliott, A. and Hall, N. (1985) "Young Children's Perceptions of Computers", in Salvas, A.D. (ed.) *Communication and Change*, Melbourne, CEGV.

Gregory, D and McDonald, S. (1983) *QWERTY the computer*, Wildgrass Books, Carlton North, Victoria, Australia.

Lamplighter Teachers (1981) *Learning with Logo at the Lamplighter School*, Microcomputing, September.

McDougall, A., Adams, T. and Adams, P. (1982)*Learning Logo on the Apple II*, Melbourne, Prentice-Hall.

McDougall, A., Adams, T. and Adams, P. (1984) *Learning Logo on the Commodore 64*, Melbourne, Pitman Publishing Pty Ltd.

Nevile, L. and Dowling, C. (1983) *Let's Talk Apple Turtle*, Sydney, Prentice-Hall.

Nevile, L. and Dowling, C. (1984) *Let's Talk Commodore Turtle*, Sydney, Prentice-Hall.

Papert, S. (1980) *Mindstorms: Children, Computers and Powerful Ideas*, New York, Basic Books Inc Publishers.

Papert, S. (1980) "Redefining Childhood: The Computer Presence as an Experiment in Developmental Psychology", *Information Processing 80*, Amsterdam, North-Holland.

Turkle, S. (1984)*The Second Self, Computers and the Human Spirit*, New York, Simon and Shuster.

Using a Computer in a Pre-School

Lesley Tan

Senior Lecturer
MCAE-IECD

Pauline Adams

Pre-school teacher

Pre-school education is concerned with promoting optimal development of young children. Development is often conceptualised in terms of cognitive, social, emotional, linguistic, perceptual, motor categories. Such a fragmented view may be convenient for educational analysis, teacher training, etc., but it is essentially artificial - these areas are not independent. Pre-school education aims to promote the development of attitudes and skills which will be adaptive for learning and for everyday functioning not only in early childhood but as the child progresses on to adulthood. Experiences in the pre-school are typically supported by activities which include the following: painting, drawing, dolls cutting, pasting, threading, household-related materials, water and sand play, wheeled toys, carpentry, clay, music puzzles, table games, books, pictures, swing, jounce-board, climbing frame.

Within such a context microcomputer experience would seem appropriate - as with any other activity what is important is how the equipment is used rather than assigning the equipment any intrinsic value of its own. Blocks are fine for building, they would not be included if they were mainly used as weapons, or simply to give children familiarity with blocks.

In planning the program an appropriate *balance* between different types of activities is important, considering such aspects as:

- degree of structure or flexibility
- suitability for individual or group use
- degree of challenge, likelihood of competence
- opportunity for creativity, self-expression
- aspects of development likely to be promoted.

Criteria used in choosing equipment have been discussed by Irvine (1984) in relation to five categories under which they may appeal to the child:

- Challenge Appeal to senses, respond to child's actions; offer "ego appeal" mastery of skill
- Physical for dexterity, co-ordination, large or fine muscle skills; need to plan a sequence of actions
- Perceptual require visual or auditory discrimination, and memory; associated with symbolic representation, e.g. language
- Creative provide for alternative and multiple uses; opportunities for imagination, invention
- Problem-solving require co-ordination of senses, of objects in space and sequential planning; allow experimentation.

(Of course, factors such as robustness, safety, and physical suitability must also be considered.)

Before we look at microcomputer use within this framework we must remind ourselves that pre-school children are preliterate. They cannot be expected to read or write and their skills of alphanumeric recognition are fairly poor. The *teaching* of such skills is generally regarded as inappropriate in pre-school, though incidental learning would not be discouraged. Software of the CAL type and adaptations of Logo are available which do not require that users are literate and in this sense seem suitable for pre-school children. We may also note that computer speech on a relatively cheap Text and Speech synthesiser is now available with fairly good

intelligibility which will no doubt continue to improve. Thus the need to read message-type output may be easily bypassed.

CAL-type programs which aim to promote skills of shape recognition, matching on various visual properties, discrimination of size, of position and orientation in space are available and seem to be designed at an appropriate cognitive level. It might be suggested that such exercises based entirely on visual stimuli and requiring only a small stereotyped movement in response are an undesirable substitute for similar exercises using actual objects which the child manipulates (literally) (Barnes & Hill, 1983). While it is true that the experience lacks any tactile component and is thus less "concrete", such activities should not be seen as a substitute but rather as a complement, as another way of developing these cognitive skills.

In the area of auditory perception, this criticism could be made with even less justification. So far we have not seen software developed to teach the corresponding skills involved in recognizing, matching and ordering sounds of different pitch, volume, duration, or quality, yet this is an area where the microcomputer could provide structured experiences which have often not been provided in typical programs.

Another aspect of the program where it seems to us that the microcomputer *could* be really useful (but so far there is very little software of this type) is games. By this we mean "table games" which involve two or more players taking turns and co-operating or competing to reach some goal. Microcomputer based games offer advantages which include:

- convenience - no pieces to get lost, broken
- the chance element (dice, card-draw) is easily organized
- scoring, turn-taking is easily managed (built-in referee)
- flexibility - different ability levels easily accommodated with variations available.

Such games need not be totally screen-based, for those who are concerned about lack of "concrete" experience, off-screen props or models could be incorporated into the design, with the translation to them possibly providing valuable learning in itself.

How would activities of this CAL-type measure up against Irvine's implied criteria? They can certainly offer challenge and perceptual appeal. Opportunities for problem-solving and experimentation are only very limited within the circumscribed framework. The appropriate response from the learner (user) is predetermined, little dexterity or co-ordination is required, and opportunities for creativity and control really are not available.

This is not to say such experiences are inappropriate for pre-school. When properly managed, we believe children enjoy them greatly, showing this by long attention span, willingness to wait for their turns, as well as obvious interest in doing and discussing the activities. There are those who feel that the impact computers will have on everyday living requires early acquaintance with them - "computer literacy should begin at pre-school". For such people any interaction with a computer that has some educational value will be justified, but for others this is not enough.

CAL which is highly structured and focuses on repeated practice and *the* right answer, may not appeal to many early childhood educators. They will associate it with the learning machine, "Skinner's rats", approach to education, or at best with an orientation to facts rather than to skills and attitudes. Other educational software for young children appears to offer little that is not available from drawing, picture books or television (other than inferior graphics). Such software is unlikely to convince early childhood professionals to spend $1000 or so on a machine to provide that in their centres. (However, as we discuss later, it may be that many pre-schools will buy a microcomputer mainly, or partly, for administration, in which case it need not be wholly justified in terms of the developmental program).

We now consider Logo-based experience; the Logo language especially with graphics gives even very young children the opportunity to make the computer work for them. Within Papert's Logo philosophy, children develop not only skills but desirable attitudes towards computers, for example, that a computer is a man-made tool which people control to serve *their* purpose. The Logo commands need redefining for the average 4-year-old. To type FD space 50 space or return RT space 45 return CIRCLER space 25 is just too much, whereas typing F R C[1] is easy and allows the child

to think about what has happened as well as what he had to do. We have found children can easily learn to find the keys which correspond to several letters printed on a reminder card. A diagonal scanning strategy (which we observed a pre-school child develop and use) for finding particular keys has been taught to both children and adults and found to be very efficient. Some children need to be shown how to hit the keys - down-and-up sequence. We have found the repeated response to a key being held down on the Apple IIe is undesirable with young children. They may hold the key down without realising it - this hinders their appreciation of one-to-one correspondence between act and effect. It can also ruin their plans. For young children, we would recommend the disabling of this feature of the Apple IIe, which is easily arranged.

Within the single key command Logo system, differing degrees of complexity and many different options are available. The number of procedures available to the user can be varied from three (forward, turn, clear) upwards. Control from keyboard or games paddles is possible with the turtle moving at a chosen speed. Alternatively, the turtle can advance only on command. It is relatively simple to write programs (such as Hi, Shillingburg, no date, or Doodle mode on the TRS 80 Color computer) which make available options not only to move forwards/backwards, to turn left/right, to have/not have a trail, to erase a line or the last move and to draw different shapes, but also to change pen or background colour, and to save and recall previous drawings or sequences. This can lead on to the use of a previous drawing as an element in a repeated pattern or composite (subprocedure procedure relationship), though it is usually for the purpose of showing someone later what was done. Such productions are usually labelled with the child's name. Some 4-year-olds can spell (and type) their names already, others have been observed to learn rather quickly when rewarded by the recalling of their drawing (maybe many times over).

The manner in which different children explore the microworld of single key Turtle graphics varies greatly. Some begin with more interest in exploring the keyboard than in noticing the results on the screen. Others establish remarkably quickly a relationship between what they do and what they see and go on to plan out drawings. Some children have undertaken quite systematic investigations of sequences and relationships. For example, what

happens if the sequence ****2 is repeated (it makes an asterisk or star), and what happens if you draw over green with black, or draw with white on a white background. Both of these led other children who were watching at the time to contribute to an interesting discussion and develop the ideas this gave them in their own ways later.

Much 4-year-old play is still of the "parallel actions in a shared context" type; the "committee" approach to programming which teachers have reported with such pleasure in, say, 8-year-olds (e.g. Maxwell, 1984) could not be expected in pre-schoolers. Nevertheless, it is usual to have at least one or two children waiting for their turn around the computer - they frequently follow what is being produced and are eager to make suggestions or to discuss points of interest or problems that crop up. Thus the microcomputer activity promotes communication skills as well as other social skills like sharing and helping.

We have found as have others (e.g. Berg, 1984; Williams, 1983) that the appeal of the microcomputer activities varies between different children as is the case for other activities. After an initial first turn, some children seldom return, others want a turn every time it is available. Some known as "flitters" (moving quickly from activity to activity) show a long attention span at the computer, another child, normally quiet and shy, becomes animated as she participates in the discussion at the computer. The feeling of power and competence demonstrated by children who have driven the computer is very clear in some cases and especially rewarding for the teacher if this applies to a child thought to have poor self-esteem. We have not yet had the opportunity to use Logo with sprites or with music in the pre-school. Less open-ended programs in Logo such as mazes, shape-tracing, or spatial estimations can be readily made at a level suitable for 4-year-olds, but we have no time to discuss them here.

Let us now check our Logo single key systems against Irvine's criteria. Challenge is certainly evident with sequential planning, fine motor skills and perceptual appeal; Discrimination, memory and symbolic representation are an integral part of the experience. Opportunities for imagination, experimentation and problem-solving abound. They seem to be provided in a way which is rather

different from other activities and thus are likely to appeal to children who may not have taken opportunities that other media provide. For example, girls seldom play with floor blocks at many pre-schools but the computer so far is apparently not "gender labelled".

The microcomputer should be able to take its place among the other activities in pre-schools. The experiences it can provide are valuable and appropriate. But a computer is still a big expense, and most pre-school personnel are unfamiliar with them, and have only a distrust of Pacman-type games, very vague notions of CAL and no idea of Logo. The need for appropriate inservice and preservice computer courses is great, and we are happy to report that demand is considerable. There is also a need for research (Barnes & Hill, 1983; Brady & Hill, 1984). Are computers likely to provide something that pre-school children *should have* but are *not likely to get* otherwise? This may be about the most important question for some - but is it likely to be answered soon or at all? Has it been required that other activities in the pre-school program have their inclusion justified by rigorous research? Most pre-schools have easily $1000 worth of books, of which at certain times in the program no use at all is being made - so is the expense of a computer so special and different?

But there is another role for the microcomputer in pre-schools. Sessional pre-schools are normally very small administrative units, run by a voluntary committee and staffed by one trained educator and an untrained assistant. Within a typically tiny office, and without any office staff, the pre-school director must maintain waiting lists, enrolment, health and developmental records, produce notices, information sheets and various parent consent forms, keep track of library books, stock-taking of other permanent equipment and ordering of consumable materials - to give only a very general and incomplete outline! While a child care centre will have more staff and probably a different system of management, the administrative tasks will, in general, be fairly similar. In such contexts, the value of a microcomputer which can be used for word processing, producing and updating notices, etc., and as a data storage and retrieval system (using a filing package such as PFS) is obvious. No doubt the management (or committee) would also wish to take advantage of it for their records, production of agendas and minutes and for their accounts. It would not be surprising if the

personal computer is bought for these purposes in some cases anyway, and then - "well, since we have it anyway, maybe the children can play with it".

References

Barnes, B.J. & Hill, S. (1985) Should young children work with microcomputers - Logo before Lego? *The Computing Teacher*, May, 11-14.

Brady, E.H. & Hill, S. (1984) Young children and microcomputers: research issues and directions. *Young Children*, March, 49-61.

Burg, K. (1984) The microcomputer in the kindergarten. *Young Children*, March, 28-33.

Irvine, J. (1984) Play in day care. In Dixon, D. (ed.) (1984) *Handbook for Day Care*, Canberra. AECA.

Maxwell, B. (1984) Mathematics from Logo. Seminar presentation. Stevenage Teachers' Centre, Stevenage, Herts, 24 May.

Shillingburg, P. (no date) *Hi: a Pre-schooler's Introduction to Apple Logo.*

Williams, R.A. (1983) Dolls, Blocks and a Computer. Mimeo. Ball State University, Muncie, Indiana.

[1] Redefined procedure for the above

[2] Such explorations have given us food for thought about relative positions of particular keys in the system and means of labelling them, which we cannot discuss here.

Cognitive Components and Mechanisms Underlying Children's Acquisition and Transfer of Logo Programming Skills

Susan M. Chambers

Division of Cognitive Science & Psychology
Deakin University
Victoria

How do children think when they are writing Logo programs? Do their thought processes change as a result of Logo experience? What is the nature of the skills and knowledge acquired? These issues are fundamental for those concerned with the introduction of Logo programming to school-aged children. The issues are also basic to models of cognitive development since there is a need to define the cognitive processes underlying performance, the nature of the interrelationships of these processes, the factors which affect these processes and the effect skill in one domain has on performance on other tasks. Sternberg (1984, 1985) has proposed a Componential Theory of human intelligence and elaborated the model to provide a theoretical framework for considering the underlying cognitive processes and mechanisms which children may use when acquiring cognitive skills. The present investigation used Sternberg's theoretical framework for accounting for children's acquisition of computing skills.

In Sternberg's Componential Theory three kinds of information processing are proposed: metacomponent (M), performance component (P) and knowledge acquisition component (K) processing. M processing refers to the executive and planning processes used during a task: the recognition of the nature of the problem; selection of the components which will perform the task; choice of strategy for combining the processing of the two other

components; choice of a mental representation upon which the strategy and other components can operate; allocation of time resources between components involved in the task; solution monitoring; interpreting and acting on feedback concerning performance. P processing involves the processes used in the execution of the task: encoding information; inference of relations between aspects of the problem; mapping of higher-order relations by determining similarities and differences between two lower order relations; application of a known relation to a new domain; comparison; justification of selected answer; response. K processing is used to acquire new information: selective encoding or sorting relevant information from the irrelevant; selective combination of the selected information to make it interpretable; selective comparison to integrate newly encoded and old information. These three kinds of information processing are claimed to interrelate differently in novel and familiar tasks.

In novel tasks, global processing is used. M processing guides and directs P and K processing during task performance. This takes considerable additional resources. The relevant components for this task may be either absent or only weakly established leading to slow, error prone performance in which M processing dominates P and K processing. With increasing expertise local processing is used. Task knowledge is packed into local processing sub-systems, or production systems. These can be accessed as a whole. Once the M processing system recognises there is a relevant local processing system which might satisfactorily perform a task, control for task performance is passed to the local processing system until a result is returned to the metacomponent monitoring the solution. Local processing involves P, K and, if necessary, M processing. There is no hierarchical relationship. With increasing expertise, the expert may draw on a number of local processing systems in parallel and thus allow more time for efficient planning and executive processing. Expert performance should be distinguished from novice performance by the greater involvement of P and K processing based on task specific knowledge. Novice performance should exhibit M processing dominating P and K processing.

The Componential Theory provides a very plausible model of the cognitive processes and mechanisms which may underlie the acquisition of programming skill. So far, the model has not been

applied to considering programming performance. Nor, for that matter, has there been an empirical demonstration of the changing interrelationship of M, P and K processing as expertise develops. The present study attempted to operationalize the Componential Theory of cognitive development using programming as the task domain.

Programming is a complex dynamic skill involving several cognitive activities. Pea and Kurland (1983) have distinguished four sub tasks: understanding the problem, solution planning, implementation of the plan, and evaluation of plan and of its implementation. With the exception of the implementation sub task, the other sub tasks would be M processing functions in Componential Theory terms. Implementation is a P processing function. There is no specific role for K processing. There is no indication in Pea and Kurland's model how the sub tasks would interrelate dynamically during programming. An analysis of the programming task in the present investigation suggested that several measures might be used as a representative of M, P and K processing. Planning and style were used as measures of M processing, accuracy as a measure of P processing and mode of programming as a measure of K processing. Two programming tasks were used, one ON task (the program was written at a computer) and one OFF task (the program was written away from a computer). The use of two tasks allowed a predictor set based on M, P and K processing in one task to be formed to predict performance on the second task.

Level of expertise was expected to vary as a function of age, ability and programming experience. The participants were 312 children, aged from five years six months to twelve years two months. They participated in Logo sessions during two years. Post-testing included computing measures (two programming tasks and a computer knowledge test), four other cognitive tasks (Tower of Hanoi, Rotation, Typing, Block tasks) and several psychometric tests (Raven Progressive Matrices, Series, Memory span, Directions, Mazes and Patterns tests).

It was predicted that children with more expertise might engage local processing (mainly P and K processing) and children with less expertise would rely on global processing (M rather than P and K processing). The nature of the processing might vary with the programming task. If there was evidence of local processing it

was predicted that the processing would be based on task-specific rather than general skills and knowledge. Supporting evidence was found for all the predictions.

The issue of the nature of the skills and knowledge acquired and transferred as a result of programming experience can now be addressed. Are the skills and knowledge only domain specific as suggested by Howe (1980), or are general skills also acquired as proposed by Papert (1980)? The analysis reported above indicated that non-task specific skills played a role in M processing rather than P and K processing. This finding suggests that general, rather than task specific skills may be enhanced as a result of programming expertise, since these are the skills used across tasks. The nature of the transfer may also depend on the explicit/implicit nature of the instruction used. In the present investigation no explicit transfer training was attempted.

To determine the unique role of programming experience on children's performance in computing and non-computing tasks, Cohen and Cohen's (1983) method of hierarchical R^2 analysis by sets was used. According to Cohen and Cohen, an effect of experimental treatment can be claimed if the entry of the experimental measure leads to an independent change in R^2 and if there is no significant interaction effect of the experimental treatment measure and the covariates. The covariates selected were age in months, ability (Otis-Lennon School Ability score taken at the beginning of the study), school and test experience. The dependent measures were computing performance, performance on the four non-computing tasks, performance on the psychometric tests.

Evidence was found of enhancement of domain specific and general skills. Performance on all the computing measures were significantly predicted by programming experience. There was no enhancement of performance on the four non-computing tasks. This finding suggests that specific P and K skills for these tasks may be critical for their performance rather than general skill which may be acquired as a result of programming skill. The finding of particular interest was the enhancement effect found on several psychometric measures. There was a statistically significant enhancement for performance on the Memory span, the Directions, and the Pattern test. Further, a positive effect for performance on the Raven test was evident if the age range was limited to those

children less than twelve years and three months at the time of testing. These results are consistent with Papert's (1980) claim that Logo experience will encourage the development of combinatorial and systematic thinking.

The finding of enhancement of general skills as a result of Logo experience is of great importance both to educators and psychologists. Teachers and educators have been enthusiastic about the educational benefits of Logo. The present findings provide statistically significant evidence that there are grounds for the claims that Logo experience promotes the development of general skills as well as the learning of programming skills. Psychologists concerned with the way in which skills are transferred will be interested that the enhancement of general skills reported here occurred as a result of implicit rather than explicit instruction. These results are consistent with those of Berry and Broadbent (1985) who found that explicit instruction may interfere with the transfer of skills in complex tasks. The effect of the nature of instruction in skill transfer needs further exploration.

References

Berry, D.C. and Broadbent, D.E.B. (1985) "Interactive tasks and the implicit-explicit distinction". Unpublished paper.

Pea, R.P. and Kurland, D.M. (1983) *On the Cognitive Prerequisites of Learning Computer Programming*, (Technical Report No. 18), New York: Centre for Children and Technology, Bank Street College.

Sternberg, R.J. (1984) "Mechanisms of cognitive development: A componential approach". In Sternberg, R.J. (ed.) (1984) *Mechanisms of Cognitive Development*, San Francisco: Freeman.

Sternberg, R.J. (1985) *Beyond I.Q.*, Cambridge: Cambridge University Press.

Some Comments on Logo after Ten Years....

Liddy Nevile

Barson Research
Melbourne

It is ten years since Logo was first introduced to children in Australia. Yet yesterday, in attempting to convey some of the attributes of Logo to a group of teachers, I had the feeling I was attempting to incite gentle folk to riot, rather than offering a long-awaited alternative which could open new horizons for those people.

What has gone wrong? Why it taken ten years of hard work on the part of ever so many people to achieve so little? Is there really less to offer in Logo than the Logophiles believe? Has the progress been more significant than it appears from my point of view? What are the changes which Logo implies and what is the mechanism for achieving them?

Papert talked about the use of Logo as a catalyst for "mindstorms" in children's minds. He foresaw a day when children would have a more explicit view of epistemology as well as the objects of their learning processes as a result of having worked with Logo. Never did he expect Logo the programming language to deliver the goods, but he did imagine that a Logo environment could make a special contribution to the learning environment as a whole.

In this paper I wish to consider some of the factors which I believe have been responsible for the difficulties associated with Logo, and a report of some work undertaken in an attempt to smooth away some difficulties.

Papert considered that Logo itself was a tool for thinking. His idea has been well accepted, and learning to program in Logo is not

considered an end as much as a means to the creation of microworlds in which children will explore concepts and processes which may otherwise have been inaccessible to them. The role of these microworlds, and they include the whole set, many subsets, and extended subsets, of Logo primitives, was to provide children with an appropriate perspective and context in which to explore their fragmented knowledge.

Unfortunately there are still very few microworlds which have been created using Logo, and which have the power of the earliest examples (the dynaturtles of diSessa, the smelling worm of Abelson, and the turtle itself). Papert thought that children would create their own microworlds, and in a sense I believe they do, but the challenge is for teachers to produce microworlds to stimulate learning in areas where children will not spontaneously explore and learn. Noss has looked at the problems associated with creating a mathematical environment for children, Sharples, Goldenburg and Fuerzig at the creation of language microworlds, Bamberger at the possibilities with music, but nobody has found a formula, an explanation for the successful creation of microworlds.

Metaphorically, the turtle is the child (the user), and commands to the turtle represent instruction to the child. Soon however, this metaphor is abstracted by the child and the relationship with the turtle becomes direct. The charm of this is that children who were generally too young to have learned to abstract to this extent, have been supported by the special turtle environment to achieve this position. Children, and teachers have seen the results time and again, and simple turtle graphics are praised as having educational value for children.

Curiously, few teachers realise that the turtle is not a model of the child. They are too abstract from the original metaphor without noticing, and are prepared to accept, for example, the fact that the turtle's actions are difficult to model in real life, in a Newtonian world.

I use this example to draw attention to the power of the turtle as a metaphor. It is because the turtle is not a perfect model of the real world that so much exploration can take place in the turtle microworld which has implications beyond the particular screen context. If the turtle were represented on the screen as a real turtle, and commands caused it to move about in a 'real' way, (as could

be done with existing technology), the turtle microworld would be shallow. It would simulate the real world rather than be a metaphor for it; its accuracy would have been achieved at the price of its value as a stimulator of thinking processes.

Assuming that the goal of the educational activities is to alter the children's intuitive knowledge states, and not merely to provide them with a formal knowledge system which can be activated in certain settings, the role of the microworld looms large.

In the early days of Logo Fischer wrote about the progressive use of microworlds which would sustain a child for the necessary time, and then become obsolete and be discarded in favour of a more sophisticated one. He warned of the danger of microworlds which incorporated features which may have to be unlearned at a later stage in the child's development, but encouraged the use of props so long as they were required. Examples of successful everyday microworlds were those associated with learning to ski, ride bicycles, etc.

An important feature of these microworlds is that they are open-ended. There is not one lesson to be learned from each stage, but an accumulation of skills is expected to motivate the child's progress from one microworld to another. Nor is it true that there are not aspects of the various microworlds which do not hold true for the complete learning process: not needing to take account of the speed of the bicycle is true for the stage when learning to pedal to propel the bicycle is taking place, but when the stage of learning to ride down hills is reached, speed becomes very important.

This consideration of Fischer seems to suggest that there are some essential pieces of knowledge which should not be introduced at one stage merely to be unlearned at another stage, but does allow for the introduction of pieces of knowledge which will later be displaced. Weir has made the point that it is the fundamental ideas of a topic which should be the primitives within an educational microworld so that the learner has the facility to get on with 'messing-about'. She does not see these primitives necessarily as models of the real world however. In fact she says the "the whole point has been to make the artefact different in principled ways by isolating the basic components of the activity so as to enhance learning". She uses as an example the separation of the turn and forward in the turtle geometry microworld.

Well, where are the microworlds? The ingredients of microworlds which I have been attempting to identify are obviously needed before microworlds can be built. An understanding of a topic must be so proficient on the part of teacher that the primitives chosen will be required as knowledge (p-prims in diSessa's terms, privileged mental schemas in Weir's), the metaphors chosen will need to stimulate the necessary response in the learner so that the interaction between the learner's p-prims and the microworld will activate processes which will in turn advance the state of the learner's knowledge, and for this the teacher will need to know the p-prims which the learner will bring to the microworlds. The mere modelling of a concept does not constitute this sort of microworld.

Let us turn now to two aspects of Logo programming which are difficult to teach, and see if there are any implications from what has been said which may explain why they are so difficult. First, it should be noted that list-processing and recursion (the two aspects) are essential skill/concepts for even the satisfactory use of Logo, and that the failure of the majority of Logo users to acquire them must limit the successes of Logo.

Let us assume that learning through the use of microworlds is the goal for the teaching of list-processing and recursion, and also that the microworlds should be open-ended and extensible.

In many cases, teachers try processing lists as a way of explaining the processes. For example, using a procedure which writes a given word or sentence in reverse order is typical. Generally the learner can follow what happens, and is willing to undertake similar activities. It is rare however, to find Logo users who then spontaneously use this process for some other purpose. More commonly, they will accede that they 'know how to do it' but will find other activities more pressing for this time. The same is true of the use of recursion other than tail recursion. In many classes learners are shown how the recursive procedure will be executed by the computer, and therefore what to expect, and are even invited to play about with some standard procedures which produce pleasing fractals, but few of those learners leave the class and take real understanding with them.

It has been an interesting challenge to try to produce microworlds for messing about with list-processing and recursion. Consistent

with what has already been said, metaphors for list-processing and recursion have been sought. Identifying the 'basic components' and finding ways of separating them have been time consuming tasks. A brief report of these endeavours follows. It should be noted that the target learners in the situations below have been adults at this stage.

In the case of list processing, it seems that many adult learners do not have a satisfactory concept of strings of words and lists constituting a list. One teacher asked what a list was, and has difficulty reconciling a 'computer's list' with a shopping list. This problem is obvious when teachers are learning to format text with a word processor too, so one identifiable primitive was to be the list itself.

These adults often do not understand very well that Logo operations and commands can be combined to form other operations and commands (early on Papert identified procedurality as a necessary primitive). For a combination of this type the metaphor of 'rule' was chosen.

The simple procedure following was then suggested:

```
TO PROCESS :LIST :RULE
RUN :RULE
PROCESS BF :LIST :RULE
END
```

with the stop-rule introduced only when it had been seen that the procedure ran through the list until there were no more elements (IF EMPTY? :LIST STOP), and where a typical rule was expressed as follows:

```
MAKE "RULE [MAKE "WORD :NEWLIST FIRST :LIST]
MAKE "NEWLIST []
```

with an accompanying

```
MAKE "LIST [A E I O U]
```

The result of using the procedure would be:

```
AEIOU
```

It is not necessary, or even useful always, to write this type of procedure this way, but separating out all the activities in this way does give the learner a greater chance to understand the process

involved.

It is also advantageous to start list-processing in this exaggerated way, because the next step towards understanding embedded recursion follows very simply.

If the given procedure is re-written thus:

```
TO RECURSE :LIST :RULE
IF EMPTY? :LIST STOP
RECURSE BF :LIST :RULE
RUN :RULE
END
```

the result of running it would be:

```
UOIEA
```

This theme is easily developed as a microworld for recursive messing about. If the basic (or primitive) procedure becomes

```
TO R :LIST
IF EMPTY? :LIST STOP
RUN :RULE1
R BF :LIST
END
```

then other procedures can easily be built on this model:

```
TO RR :LIST
IF EMPTY? LIST STOP
RUN :RULE1
R BF :LIST
RUN :RULE2
R BF :LIST
END
```

```
TO RRR : LIST
IF EMPTY? :LIST STOP
RUN :RULE1
R BF :LIST
RUN :RULE2
R BF :LIST
RUN :RULE3
R BF :LIST
END
```

and so on.

If the :RULE1, :RULE2, and :RULE3 etc. are carefully chosen, the representation of the procedure's execution can be most enlightening. Examples already used include melodies, patterns, a clock-face with concentric bands of colour which change as an indicator moves around the edge, and controlled animation is anticipated.

Now suppose the stop rule is changed and becomes either

```
IF COUNT :LIST = 1 OP FIRST :LIST
```

or

```
IF COUNT :LIST = 1 OP LAST :LIST
```

It can be seen that the stop rule will cause the remaining element of the list to be included before the process stops. Now the first procedure could be re-written

```
TO PROCESS :LIST
IF COUNT :LIST = 1 OP LAST :LIST
OP WORD FIRST :LIST PROCESS BF :LIST
END
```

and would produce

```
AEIOU
```

The need for the rule and the creation of an empty list before the process can start have both disappeared. The recursive process reflects the self-referential aspect of the rule: every element is processed by being processed on to the already processed elements.

As diSessa explains, the programmer chooses the limiting event, the result of the first event, and the connection between that event and the others, and leaves the rest of the work to the computer.

It is believed that a procedural microworld of this type will help many teachers come to understand how the 'clever recursive procedures' are written. It is by no means clear that this microworld can be created easily.

Once these skills are gained, not only will the programming power of those teachers increase enormously, but they will be in the position to use the programming techniques they have experienced to tackle topics outside the programming domain. A good example is the elusive topic which is outside this domain, but can be metaphorically connected to it - the use of inductive thinking techniques.

The difficulty inherent in the development of any microworld is very apparent in these examples. A model of recursion is not the same as a metaphor for it; the same is true for list processing.

The traditional 'metaphor' for list-processing and recursion is the little man model. It has been proposed that in Boxer the activities

of the little man will be taking place on the screen, and therefore the problem of recursive programming should disappear. I suspect that all the explanations of the watch-the-process type do not help beginners understand recursion. They believe what they see, and feel sure they could carry out a procedure by hand with a pencil, but they do not intuitively bring forward the process as a strategy for achieving a goal.

Carroll and Mack describe a metaphor as "a crystal seed disturbing a saturated suspension to precipitate the formation of a growing crystal". They state that if a metaphor is appropriate, the learner will be stimulated to become active in the learning role, and to construct a model of what is to be learned.

I conclude by restating the difficulty of the problem with Logo: not only do the teachers (or those who create the microworlds) need to know the subject domain sufficiently well to be able to decompose it, they must do this in a way which activates the learner and facilitates the construction, by the learner, of a progressively more suitable model of the concept/skill being 'taught'.