MASSACHUSETTS INSTITUTE OF TECHNOLOGY
MEDIA LABORATORY

EPISTEMOLOGY AND LEARNING GROUP

E&L Memo No. 7
September 1990

Software Design as a Learning Environment

Idit Harel
Seymour Papert

Abstract

This article describes a learning research called the Instructional Software Design Project (ISDP), and offers a Constructionist vision of the use of computers in education. In a Logo-based learning environment in a Boston inner-city public school, a fourth-grade class was engaged during one semester in the design and production of educational software to teach fractions. Quantitative and qualitative research techniques were used to assess their learning of mathematics, programming, and design, and their performance was compared with that of two control classes. All three classes followed the regular mathematics curriculum, including a two-month unit on fractions. Pre- and post-tests were administered to the experimental and control groups. The evaluation revealed greater mastery of both Logo and fractions as well as acquisition of greater metacognitive skills by the experimental class than by either control class. Selected results from several case studies, as well as an overall evaluation are presented and discussed. Using ISDP as a model project, a Constructionist vision of using technology in learning is elaborated. The ISDP approach of using Logo programming as a tool for reformulating knowledge is compared with other ways of learning and using Logo, in particular the learning of programming per se in isolation from a content domain. Finally, ISDP is presented as a way of simultaneously learning programming and other content areas; and the claim is made that learning both of these together results in better learning than if either were learned in isolation from the other.

Software Design as a Learning Environment

Idit Harel
Seymour Papert
Massachusetts Institute of Technology
Cambridge, MA 02139

Abstract

This article describes a learning research called the Instructional Software Design Project (ISDP), and offers a Constructionist vision of the use of computers in education. In a Logo-based learning environment in a Boston inner-city public school, a fourth-grade class was engaged during one semester in the design and production of educational software to teach fractions. Quantitative and qualitative research techniques were used to assess their learning of mathematics, programming, and design, and their performance was compared with that of two control classes. All three classes followed the regular mathematics curriculum, including a two-month unit on fractions. Pre- and post-tests were administered to the experimental and control groups. The evaluation revealed greater mastery of both Logo and fractions as well as acquisition of greater metacognitive skills by the experimental class than by either control class. Selected results from several case studies, as well as an overall evaluation are presented and discussed. Using ISDP as a model project, a Constructionist vision of using technology in learning is elaborated. The ISDP approach of using Logo programming as a tool for reformulating knowledge is compared with other ways of learning and using Logo, in particular the learning of programming per se in isolation from a content domain. Finally, ISDP is presented as a way of simultaneously learning programming and other content areas; and the claim is made that learning both of these together results in better learning than if either were learned in isolation from the other.

OVERVIEW

This article has a double intention: It adds to the description and discussion of an experiment that formed the centerpiece of Harel's doctoral dissertation (Harel, 1988), and it uses the discussion of this particular experiment to situate a general theoretical framework (developed over the years by Papert and his colleagues) within which the experiment was conceived. The experiment will be referred to here as the "Instructional Software Design Project" (ISDP), and the theoretical framework as "Constructionism" (e.g., Papert, 1990).

The ISDP experiment involved studying a class of fourth grade students. Each student worked for approximately four hours per week over a period of 15 weeks on designing and implementing instructional software dealing with fractions. A narrow description of our intention in doing this is that we wished to turn the usual tables by giving the learner the *active* position of the teacher/explainer rather than *passive* recipient of knowledge; and in the position of designer/producer rather than consumer of software. This idea is in line with Constructionism's use of "building," "constructing," or "knowledge-representing" as central metaphors for a new elaboration of the old idea of learning by doing rather than by being told ("Constructionism" rather than "Instructionism").

The usual passive view of integrating computers into education supports *Instructionism and Technocentrism* (Papert, 1987). ISDP, like all projects at Paper's Epistemology and Learning Group, attempted to change this approach by giving children the control over their learning with computers. Children were the agents of thinking and learning—not the computer. Our view is: Computers cannot produce "good" learning, but children can do "good" learning with computers.

Does wood produce good houses? /f I built a house out of wood and it fell down, would this show that wood doesn't produce good houses?... These...questions ignore people and elements that only people can introduce: skill, design, aesthetics... (Papert, 1987. p. 24).

It ought to be equally obvious that people*are the agents when it comes to thinking and learning, not computers. People use computers to do things. If we were to say anything meaningful about the thinking and learning involved, then we should look at what people are doing with computers, and not at what "the computer" is allegedly doing to them. For in reality, there is no such thing as "the computer" in general—only specific uses of computers in specific contexts... With a passive view of education, we open the door to technocentrism when we speak about the computer as an "educational tool"... It should not be an* "educational *tool, but just a tool. Like other tools, it allows us to do things we couldn't do before, or more usually, to do some things that we could do before better (Falbel, 1990, pp. 2-3).*

Building on the computer (or with the computer) a piece of instructional software about fractions is discussed here as a privileged way for children to engage with fractions by constructing something personal. In this, it may overlap educational techniques that employ materials such as cuisenaire rods, fraction bars, or pattern blocks. But constructing software goes far beyond the physical manipulations involved in using such materials. To the adage "you learn better by doing," Constructionism adds the rider, "and best of all by thinking and talking about what you do." Without denying the importance of teaching, it locates the important *directions of educational innovation* less in developing better methods of teaching than in developing "better things to do and more powerful ways to think about what you are doing" (e.g., Papert, 1971a, 1971b).

The key research question is to determine what kinds of things are "better." In this paper we focus on attributes such as *appropriability* (some things lend themselves better than others to being made one's own); *evocativeness* (some materials are more apt than others to precipitate personal thought); and *integration* (some materials are better carriers of multiple meanings and multiple concepts).

We see several trends in contemporary educational discussion such as "situated learning," and "apprenticeship learning" (e.g., Brown, Collins, & Diguid, 1989; Collins & Brown, 1987; Suchman, 1987) as being convergent with our approach, but different in other respects. Two features will be discussed here as giving specificity to Constructionism in relation to this essentially synergistic body of literature. The first is our emphasis on developing new kinds of activities in which children can exercise their doing/learning/thinking. (Turtle Geometry is one example. ISDP is another.) The second is our special emphasis on project activity which is self-directed by the student within a cultural/social context that offers support and help in particularly unobtrusive ways. ISDP provides us with insights into the unique ways in which constructing instructional software generates and supports personal reflection and social interaction favorable to learning.

In elaborating the Constructionist vision we take the time to dissipate misunderstandings by contrasting it with derivatives of Papert's early work that radically miss its epistemological essence. In particular, we emphasize the fact that ISDP has little to do with the idea that learning Logo is in itself either easy or beneficial.

WHAT WAS ISDP?

Context

ISDP was conducted as part of a larger project to study the uses of computers in elementary schools. Project Headlight, as it is called, is based in an inner city public school, the Hennigan

School, in Boston. Only *one third* of Hennigan students, with children from first through fifth grade, participate in Headlight. (The experimental ISDP class and control class C1, which did daily programming in Logo, were both part of Project Headlight. Control class C2 was not). As at many Boston public schools, the majority of the student population at Hennigan is Black and Hispanic, and in most ways the school is quite conventionally structured. A major purpose of Headlight was to gain understanding of how a computer culture could grow in such a setting. One feature that is not typical in Hennigan is its building, which dates from the early seventies when there was a fad for "open architecture." When we first saw the school its architectural features were virtually unused, but we viewed them as an opportunity to reinforce our open-ended educational philosophy through the design of the space. We saw the *physical environment* as a very important factor in shaping a learning culture. These open spaces allowed us to bring the technology closer (physically and conceptually) to students and teachers; to integrate the computer activities with the regular classroom activities; and to facilitate movement and action around the computers; to reinforce communication and information-sharing regarding computer based activities across grade levels and among teachers.

In Headlight there is no long hallway leading into one classroom called the "Computer Lab" where children take their weekly "Computer Literacy Class." Rather, there are two large open areas (the "Pods") housing four large circles with 100 computers, and each pod is surrounded by 6 classrooms. At Headlight, children use computers at least one hour a day, for working on their different computer projects, as an integral part of their homeroom learning activities.

In Headlight there is virtually no use of "ready to use software" and little emphasis on learning *about* computers and learning programming as ends in themselves. The students learn programming but programming is a means to different ends, which we conceptualize as entering a new learning culture—developing new ways of learning and thinking.

Our vision focuses on using technology to support excellence in teaching, in learning, and in thinking *with* computers— technology as a medium for expression. We particularly eschew naive views of the computer as replacing (in the guise of improving) some of the functions of the teacher. Headlight students are encouraged to tackle exceptionally complex problems and work on exceptionally large-scale projects in a culture where they have a great responsibility for their own learning. They are able to work individually and collaboratively in a variety of styles where the differences are reflected in gender, ethnicity, cognitive development, and in the individual personality of the teachers as well as in the personality of the learners (see also, Goldman Segall, 1989a-b; Harel, 1986,1988,1989a-e; Motherwell, 1988; Resnick, Ocko, & Papert, 1988; Resnick, 1989; Sachter, 1989; Turkle & Papert, 1990).

ISDP Procedures

During the period of the ISDP project, one of the "pods" in Headlight was turned into a software-design studio, where 17 fourth-grade students worked on constructing personally designed pieces of instructional software; the only requirement was that they should "explain something about fractions" to some intended audience. Before they started their software design work, the students were interviewed individually and were tested on fractions and Logo programming. Presenting herself as a researcher and a "helper," Harel explained to the students that they were not being graded, but were involved in a new kind of activity which she wanted to observe, evaluate, and report on for the benefit of others. Students were encouraged to think of themselves as collaborators in the project and its data collection.

ISDP was open-ended, but somewhat more structured than the other Headlight projects. It included a series of activities that all the experimental students performed. Each working day, before going to the computer, the students spent 5 to 7 minutes writing their plans and drawing

their designs in their personal Designer's Notebooks. Then, they worked at their individual computers for approximately 45 to 55 minutes. They implemented their plans and designs, created new ones, and revised old ones. When they wished, students were allowed to work with friends, help each other, or walk around to see what other students were doing. At the end of the ISDP daily period, students saved their daily Logo files on a diskette. In their Designer's Notebooks, they then wrote about the problems and changes of the day (related to Logo, fractions, instructional design, teaching, etc.) and sometimes added designs for the next day. The students had full freedom to choose which concepts they wanted to teach (within the domain of fractions), how to design their screens, what the sequence of their lesson should be, and what instructional games, quizzes, and tests to include, if any. In short, the Project was open-ended in terms of what the students chose to design, teach, and program. The only two requirements were: (1) that they write in their Designer's Notebooks before and after each working session; and (2) that they spend a specific amount of time at the computer each day. The purpose of this second requirement, regarding time limitations, was to allow the project to fit into the schedule of the class and of the school. This requirement also made it possible to estimate and draw generalizations about what students could accomplish in a project of this kind, within time periods that could fit into the regular schedule of any class or school in the future.

Several "Focus Sessions" about software design, Logo programming, and fraction representation were conducted in the classroom during the project. In the first session, Harel briefly introduced and discussed with the students, the concept of instructional design and educational software. Together—the children, teacher, and Harel—we defined the meaning and purpose of instructional software, and briefly discussed a few pieces of software with which the students were familiar. Harel showed the students her own designs, plans, flowcharts, and screens from various projects she had worked on in the past. She also passed among the students the book Programmers At Work (Lammers, 1987) and asked them to look at notes, pieces of programs, and designs by "real" hardware or software designers and programmers— such as the people who had designed the Macintosh, PacMan, Lotus 1-2-3, and others. In this first session the students also received their personal diskettes and their Designer's Notebooks (see Appendix), and we discussed the ways in which they should and could be used during the project.

Other Focus Sessions encouraged the students to express themselves on issues such as the difficulties of specific concepts and on how they might be explained, represented, or taught. For example, in two of these discussions, we hung two posters, one on each side of the blackboard. On one poster we wrote, "What is difficult about fractions?" and on the other, "What screens and representations could be designed for explaining these difficult concepts?" We asked the students to generate ideas for both posters simultaneously.

Other discussions focused on specific Logo programming skills. For example, in some of these short sessions about programming, the teacher, the researcher, or one of the students, could stand next to one of the computers that were in the classroom or in the "computer pod," in front of the whole class or a group of students, and explain how to use REPEAT, IFELSE, variables, etc. The students could take notes on such concepts and programming techniques in their notebooks, or go directly to their computers and write a procedure which included that new programming technique or concept.

In addition, the fourth-grade students/ designers worked with third graders from another Headlight class, who visited the ISDP class once a month, for the purpose of trying out ("evaluating") the students' pieces of software as they were developed. The fourth graders gave the third graders "demos," and then, different pairs of children were engaged in discussing different aspects of the software projects: some were teaching/learning fractions; some were teaching/learning Logo programming; some discussed design issues; and so forth. A great deal of teaching/learning through socializing went on during these sessions. However, the actual teaching was not as important as the fourth graders' feeling that they were working on a real product that

could be used and enjoyed by real people. It reinforced the "thinking about explaining things to others" during their product development, and it placed them in the role of epistemologists.

The teacher and the researcher (Harel) collaborated and actively participated in all the children's software design and programming sessions during the project: walked around among the students, sat next to them, looked at their programs, helped them when asked for, and discussed with them their designs, programming, and problems in a friendly and informal way. In general, there were no specific plans for the Project's sequence, or for our presentations and focus discussions; rather, they were initiated by the teacher or by the researcher "as needed," at times when they were relevant to the children's work or problems, or according to the children's requests.

To summarize, the children's daily activities resulted in 17 different pieces of instructional software about fractions—one product for each child in the experiment— and 17 personal portfolios consisting of the plans and designs they wrote down for each day's work, and the pieces of Logo code they had programmed, as well as their written reflections at the end of each session on the problems and changes they had dealt with that day.

To our pleasure, we observed that students worked with great intensity and involvement, over a period of four months, on a subject that more often elicits groans or yawns than excitement—namely fractions. What seemed to make fractions interesting to these students was that they could work with them in a context that mobilized creativity, personal knowledge, and a sense of doing something more important than just getting a correct answer.

ISDP Atmosphere

Procedures answering to the descriptions in the above section could be carried out in very different atmospheres but would then, from our point of view, constitute radically different projects. It is therefore appropriate to devote some space here to capture the particular ambiance of this project.

The ISD environment was marked by the deep involvement of all participants. There were interactions and reciprocal relations among the students, teacher, researcher, members of the MIT staff, and sometimes visitors—all of whom walked around the computer-area, talked together, helped each other, expressed their feelings on various subjects and issues, brainstormed together, or worked on different programming projects individually and collaboratively. Knowledge of Logo programming, design, and mathematics was communicated by those involved. Children, much like the adults in this area, could walk around and observe the various computer screens created by their peers, or look and compare the different plans and designs in their notebooks.

Young students were developing knowledge and ideas with out workbooks or worksheets, working within a different kind of a structure. They became software designers, and were representing knowledge, building models, and teaching concepts on their computer screens. They were thinking about their own thinking and other people's thinking—simultaneously—to facilitate their own learning. The following "snapshot" briefly illustrates the atmosphere of this noisy, flexible, and productive learning environment.

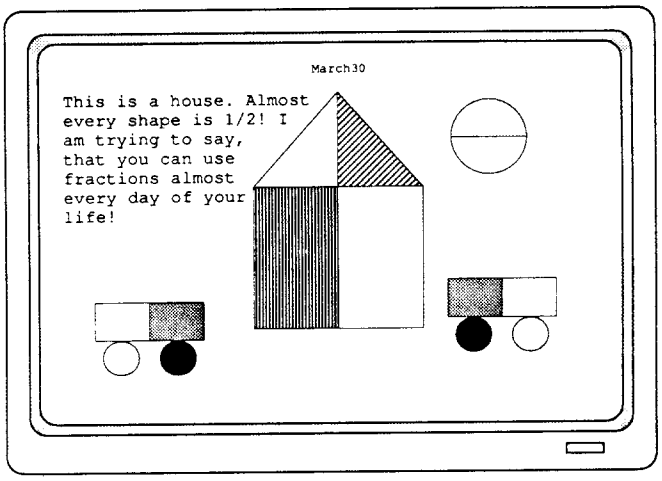
Debbie is swinging her legs while sitting at her computer and programming in an apparently joyful way. To her right, Naomi is busy programming letters in different colors and sizes. To her left, Michaela is engaged in programming and debugging a screen that shows a mathematical word-problem involving fractions, comparing thirds and halves by using a representation of measuring cups that are filled with different amounts of orange juice and water. She is very involved with her design, typing with one hand on the keyboard while her other hand is moving and touching the figures on her computer screen. A few computers away,

the teacher is trying out Tommy's program, giving him feedback on one of his explanations about "what mixed fractions are." In the background, Charlie is walking around the other computer circle, holding his Designer's Notebook in one hand, and chewing on the pencil that is in his mouth. He suddenly stops next to Sharifa's computer. He chats with her for a moment, presses a key or two on her keyboard, and observes Sharifa 's designs as they appear on her computer screen. After looking at her Logo code, moving the cursor up and down on the screen, he calls out, "Hey Paul, come see Sharifa's fractions clock!" The noise and movement around Michaela and Debbie do not seem to bother them at all at this moment. Now Naomi, who sits next to Debbie, has just completed the "title screen" for her software, which reads: "We/come To My Fractions Project! by Naomi." She is stretching her arms while moving her head to the left and to the right, looking around to see "what is new in her friends' programs. She then stretches towards Debbie's computer, and asks her to show her what she is doing. . .

Debbie shows Naomi her programming code. "It's a long one," she says, running the cursor down the screen, very proud of the 47 lines of code she has programmed for her "HOUSE" procedure. She then gets out of the programming editor to run her program, which impresses Naomi, who moves her chair even closer to Debbie's computer. In a quiet and slow voice, pointing to the pictures on her screen, Debbie explains to Naomi. "This is my House Scene. All these shapes [on the screen] are one-half. In the house, the roof has halves, the door has two halves, and I will add to this scene two wooden wagons and a sun. I'll divide them into halves too... The halves [the shaded parts] are on different sides [of the objects]. You can use fractions on anything. No matter what you use... Do you like the colors?" Their conversation goes on and on.

The idea of representing halves on the different sides of the objects, the objects being "regular human things" in a real-life situation, is Debbies's. In her final version of the teaching screen, there will be an explanatory text accompanying the pictures on the screen which says "This is a house. Almost every shape is $^1/_2$! I am trying to say that you can use fractions almost every day of your life!" Debbie is the only child in her class who has designed such a screen. She is very clear about why she designed it: to teach other children that fractions are more than strange numbers on school worksheets. As she discovered, fractions can be all around us: they describe objects, experiences, and concepts in everyday life.

Debbie has painted half of each object a different color, and left the other half blank. The house half is painted in light blue, the roof half in orange, the sun half in yellow, the door half in red, the wagon half is red, etc. While Debbie is working on this, the only advice she asks of her friend Naomi is about the colors: "Do you like the colors?" Naomi, who has adopted a different design strategy for her software, tells her, "It's nicer if all the halves are in the same color." They negotiate it for a minute or two. But Debbie doesn't agree "No. It will be boring." Naomi and Debbie continue to work on their projects with the computer keyboards on their laps..[1]



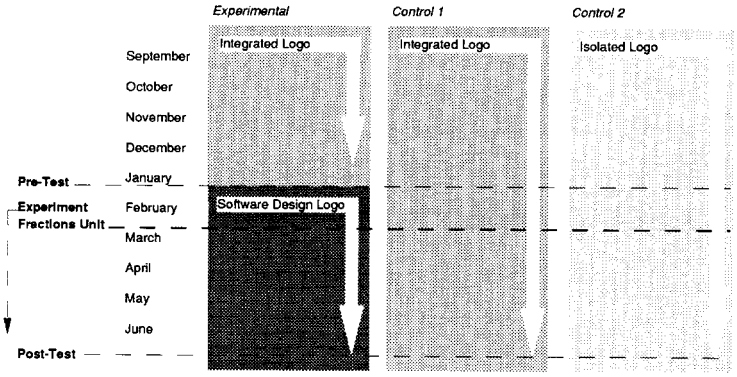
EVALUATION OF ISDP

The Evaluation of ISDP was designed to examine how students who learned fractions and Logo through the ISDP differed from students who learned fractions and Logo through other pedagogical methods. Three fourth-grade classes from the same inner-city public school in Boston were selected for this evaluation. One class, from Project Headlight (N = 17), was involved in the ISD Project (Experimental Class). Control Class 1, or C1 (N = 18), studied fractions only in their regular math curriculum and programmed in Logo as part of Project Headlight. Control Class 2, or C2 (N = 16), studied fractions in their regular math curriculum, was not part of Headlight, and programmed only once a week in the school's "computer laboratory."

Experimental Design

In January 1987, all three classes were pre-tested on specific skills and concepts in fractions and Logo. Thereafter, one of the classes participated in the four-month ISDP experiment. All 51 pupils were then tested again in June on their knowledge of fractions and of Logo (see Figure 2).

Using the set of pre-tests, it was established that no significant differences existed between the experimental and the control children's knowledge of fractions and Logo before the experiment began (Harel, 1988, 1989e). Four months after the pretests, by using a similar set of post-tests, the ways in which these students differed in their knowledge and understanding of fractions and Logo were investigated in



detail. In addition, during the project the researcher and the teacher conducted careful observations and interviews with the experimental students, and assessed (by the use of case study methods and videotaping) the development of the students in the ISD Project.

Many research questions could have been raised concerning the ISDP experiment, since it involved many variables within a complex pedagogical situation. However, for the purpose of this study, the objectives and questions were narrowed down to two main sets of assessments:

1. an assessment of the experimental children's knowledge of basic fraction concepts; and
2. an assessment of the experimental children's knowledge of Logo programming concepts and skills.

The "experimental treatment" integrated the experimental children's learning of fractions and Logo with the designing and programming of instructional software. Since the experimental students and the C1 class had equivalent, though differently-styled, exposure to Logo (i.e., both classes were part of Project Headlight), it was an open question whether participation in ISDP would result in greater Logo knowledge, but one naturally expected both of these groups to exceed class C2 in this area. With respect to fractions learning, the experimental group had additional (but not formal) exposure to fractions concepts through ISDP, so that improved performance of the experimental class was expected in this area as well, but the assessment sought to determine whether this was in fact true and, if so, what the nature of the improvement was. As will be seen in the next sections, the assessment uncovered some surprising results, more finely textured than these general surmises.

Within the fractions domain, emphasis was placed on children's ability to translate between various modes of fractional representations. This aspect has been shown to be a crucial part of rational-number knowledge, and particularly difficult for young children (e.g., Lesh & Landau, 1983; Behr, Lesh, Post, & Silver, 1983; and others). But standard school tests were also used, which concentrated on students' use of algorithms. In Logo, the evaluation investigated the children's knowledge, use, and understanding of programming commands, instructions, and operations. More specifically, it assessed whether the students from the experimental class knew and understood more programming commands and operations such as REPEAT, IFELSE, SETPOS, variables, and inputs in their projects, and became better at these skills, than the students in the two control classes. The evaluation also investigated whether the experimental students could understand, implement, debug, transform, optimize, and modify someone else's programming code better than the students from the control classes. Finally, the evaluation assessed whether the experimental students were able to construct Logo routines for someone else's design or picture and were better at this than the students in the two control classes.

Given the breadth of the learning experience and the mixed methodology of the assessments—including the extensive case studies of several students (i.e., examination of the children's progress, Designer's Notebooks, finished products, interviews with participants during and following completion of the project), as well as the more formal pre- and post-tests—it was possible to trace in detail the microgenesis of Logo and fractions skills and concepts, exploring different approaches taken by the experimental students with different personal and learning styles (see, e.g., Debbie's Case in Harel, 1988, pp. 7~245; and the Appendix in Harel's paper in Journal of Mathematical Behavior, 1990a), as well as to draw inferences concerning their acquisition of metacognitive skills.

The experimental design of ISDP and the analysis of its results we present here raise methodological issues for education research. Most acutely, these concern the question of what *kinds of rigor* are appropriate.

A simplistic position would maintain that the highest standard of rigor is always required. But we argued elsewhere (e.g., Papert, 1987) that this can sometimes result in an analog of the complementarity principle in physics, stronger formal rigor sometimes being obtained only at the cost of thinner results. Thus Harel (1988) adopted different kinds of rigor for different aspects of her work, and we will do likewise in this article.

The first results section demonstrates with statistical rigor *that learning* took place: the ISDP subjects learned *quantitatively measurable skills* in the programming and in standard school domains. The section that follows illustrates some aspects of the in-depth investigations into *what and how* they learned, going beyond test scores to obtain qualitative insights into the changes that occurred in students' thinking about fractions, and the dynamic of the process that lead to those changes. Finally, a discussion section follows, where we discuss *why* the students learned what they learned.

RESULTS

Quantitative Results from ISDP

The "thinnest" and most formally rigorous part of the analysis shows that the subjects in the experiment did improve in their ability to perform on standardized quantitative tests of performance in their work with fractions (as presented in the following subsection). Here the solidity of the results derives from the existence of a large established body of data on how students perform in such mathematics tests (e.g., Behr et al., 1983; or Lesh et al., 1983). We also present some quantitative data to show that the ISDP subjects did learn much more about Logo programming than the subjects in the two control groups (as presented in the subsection about Logo results).

Results from the Fractions Post-Tests of the Three Classes

All the teaching of fractions, for all the three classes, was conducted for two months, during regular math lessons only and following the city-wide curriculum and traditional teaching methods (see Figure 2). The experimental class was not provided with any additional formal instruction on fractions, although we note that the representations of fractions in the context of instructional design was discussed in a few informal Focus Sessions. (More information about the characteristics of the pupils, teachers, and their math curriculum is available in the dissertation and the Appendix of Harel, 1988.)

Table 1. Average Percentage Correct of Pre- and Posttests on Fractions Knowledge

	Fraction Knowledge	
Treatment	**Pretest (%)**	**Posttest (%)**
Experimental Class	52	74
Control Class 1	54	66
Control Class 2	47	56

The post-test included 65 multiple-choice questions. Out of these, 60 were taken from the Rational-Number Project (RN Project, Lesh et al., 1983 pp. 309-336). The remaining five were designed by the researcher and included word problems and construction of representations. Of the 60 RN Project questions, 43 were given to the students in the pre-test, then again in the post-test. As examples, Table 1 shows the children's average percentages of correct answers on the fractions

pre- and posttests; Table 2 shows the table of results for the Two-Way Factor Analysis of Variance with repeated measurement for the fractions pre- and post-test scores; and Figure 3 shows the interaction diagram of the two main factors. In general, the difference in pre- and post-test scores of the students from the experimental class was almost twice as great as that achieved by the students from class C1, and two-and-a-half times as great as that of class C2.

Table 2. Two-Way Repeated Measurement Analysis of Variance
(The results account for the unequal sample sizes of Factor A).

Source	*d.f.*	*F*-Statistics
A (Groups)	2	15.31**
Subjects between Samples	48	
Within Subjects	50	
B (Pre-Post)	1	110.99**
A × B	2	8.29**
B × Subjects	48	
Between Subjects	51	
Total	101	

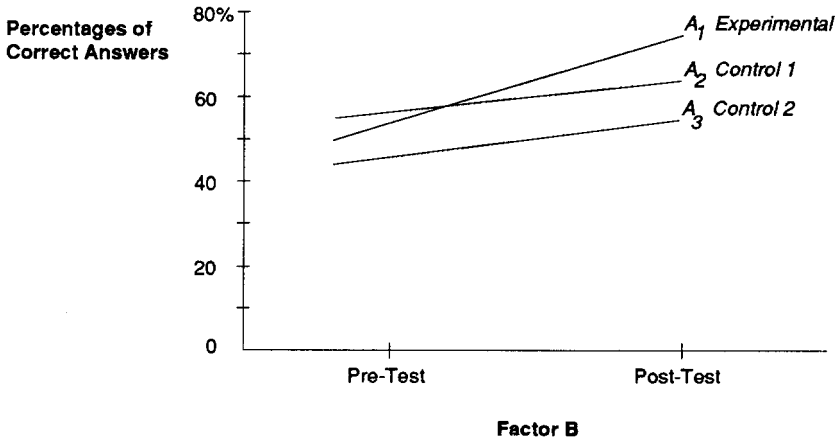


Figure 3. Interaction Diagram of the Two Main Factors in the Analysis of the Pre- and Posttests Scores

Results from the More Difficult Questions on the Fractions Test

We gave specific attention to the analyses of the most difficult translation modes between rational-number representations that the students had to carry out in the test. Some of these translations were the most difficult for students of all ages in previous studies, and were equally so for all students in the present study's pre-tests. In the post-tests however, these translation modes were still relatively difficult for the control students, but dramatically less so for the experimental students. Let us consider an example. Lesh et al. considered question 50 to be so complex that it was not given at all to the fourth graders in the RN Project, only to sixth, seventh, and eighth grade students (Lesh et al., p. 326). To answer this question, the students had to translate a pictorial representation into a written (verbal) representation of a fraction.

Table 3. Contingency Table Statistics, Comparing Performance of the Study Sample with the Performance of the Background Sample (Lesh et al., 1983, Average of Grades 6–8)

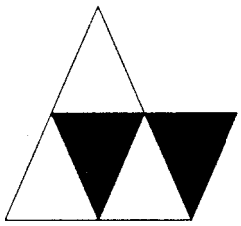
	(%) Correct in Study Sample	(%) Correct in Background Sample
Experimental Class	66	33
Control Class 1 & Control Class 2 (Average)	29	33

$\chi^2 = 33.49$

Question 50 presented students with a polygonal region representation, with a numerator that was higher than 1, a denominator, a representation of a rational number lower than 1, in a discrete object that included a perceptual distraction (i.e., one part was "outside" the triangle area). In order to choose one of the options, the students had to (1) translate the given picture into symbols or words (two fifths are shaded in), (2) read the question again and realize that the question referred to the denominator of the shaded fraction, and (3) find the correct answer, which was b. Option a is confusing because it is written like a spoken symbol and includes "relevant" numbers—five and thirds. Option b is confusing because it does not mention "fifths," but rather "five" (the denominator is "five"). Table 3 shows the scores in their percentage of correct answers for question 50.

The ISDP students scored twice as high on question 50 as did the control students, and twice as high as the sixth to eight graders from the RN Project. The Chi Square analysis shows that the differences of frequencies are highly significant.

50) What is the denominator of the fraction that tells us what part of the picture below is shaded?



a. five-thirds b. five c. three d. two e. not given

Figure 4. Question #50

Perhaps there is some "transfer" from Logo programming experience at work here. Decomposing a given picture into its geometrical components is a common process in Logo programming, and a skill students usually acquire in their ongoing programming experiences. What Lesh et al. (1983) and Behr et al. (1983) consider as a "perceptual distraction" (i.e., the one

little triangle that was "outside" the big triangle area) was probably not at all a distraction for the students who looked at the picture with "Logo eyes" and decomposed it into its five geometrical components.

Another example is Question 42. It involved a translation of pictorial into symbolic representation (see Figure 5). This question, number 42, was the 13th most difficult of the 18 asked in this subset. It was the 44th most difficult in the whole set of 60 questions given in the RN Project to students from fourth through eighth grades (Lesh et al., 1983, p. 323). It included a discrete object representation in which the represented rational number was less than one; moreover, parts of this object were not congruent and were visually distracting. Table 4 shows the scores (given as percentage of correct answers) on this question according to the children's division into math groups (see Harel, 1988 for the detailed description of the math groups).

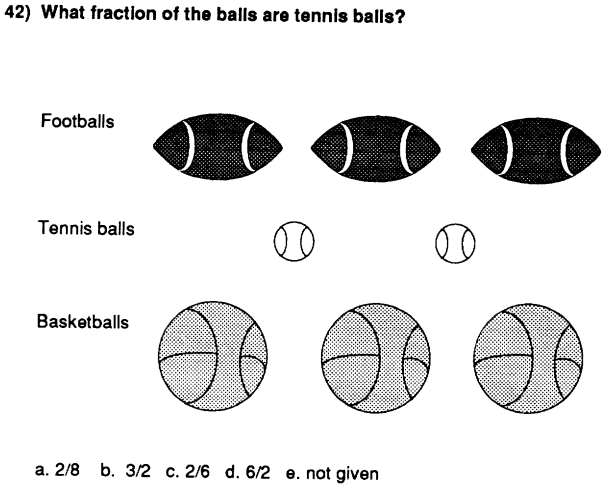


Figure 5. Question #42

As seen in Table 4, none of the high-math experimental students made any mistakes. The medium-math experimental students scored like the high-math students in the two control classes. The experimental class as a whole scored 100% better on this subset as the students in the RN Project, and 14 percentage points better and 27 percentage points better than class C1 and class C2, respectively. Table 5 shows that the Chi Square analysis of differences of frequencies is highly significant.

Table 4. Percentage of Subjects Responding Correctly to Question #42, by Treatment and Mathematical Ability

	Mathematical Ability		
Treatment	**Low (%)**	**Medium (%)**	**High (%)**
Experimental Class	50	72	100
Control Class 1	40	68	72
Control Class 2	20	50	72

Table 5. Contingency Table Statistics, Comparing Performance of the Study Sample with the Performance of the Background Sample (Lesh et al., 1983, Grade 4)

	(%) Correct of Study Sample	(%) Correct of Background Sample
Experimental Class	74	36
Control Class 1, Control Class 2 (Average)	53.5	36

$\chi^2 = 42.62$

Results from Standard Boston Public-Schools Math-Tests

In addition, all the pupils were tested in math, as part of their end-of-year public school series of "referenced tests." This mathematics test included 40 multiple-choice questions. The average number of incorrect answers was 5.06 incorrect answers per child in the experimental class, 6.27 per child in class C1, and 9.45 per child in class C2.

Of the 40 questions, six were specifically on fractions ordering and equivalence, four on decimals, four on measurements of distance and time that required the use of fractions, and one on understanding geometrical shapes (i.e., this was the subset of 15 questions directly related to rational number concepts, their representations and computation). The average number of incorrect answers to this subset of 15 rational-number questions was 1.60 per child in the experimental class, 3.16 per child in class C1, and 4.62 per child in class C2.

Several conclusions can be drawn from analyzing these results. The first is that the experimental students, in general, did much better on the entire conventional school test than the two control classes. The second conclusion is related to the children's incorrect answers in the rational number concepts subset of this test. In the experimental class, only 29% of the incorrect answers in the whole test (40 questions) were incorrect answers about rational-number concepts. But in both class C1 and class C2, approximately 50% of the incorrect answers were on rational-number concepts. This shows the superiority of the experimental class on rational-number knowledge in particular—as measured by this standard test. Table 6 shows the proportion of incorrect answers in this rational number subset to the whole test.

Table 6. Contingency Table Analysis, Comparing the Proportion of Rational-Number Subset to Whole Test in the Boston School Math Test between the Experimental and Control Classes (of incorrect answers)

Experimental Class (%)	Control 1 Class (%)	Control 2 Class (%)
29	51	48

$\chi^2 = 6.631$

The third conclusion is related to "transfer." By subtracting the average of incorrect answers on the fractions subset from the average of incorrect answers on the whole test, we can examine the children's average of incorrect answers to all the non-fractions questions: for the experimental class, 5.06 - 1.60 = an average of 3.46 incorrect answers per child on non-fractions questions; for class C1, 6.27 - 3.16 = 3.11; and for class C2, 9.45 - 4.62 = 4.83. The differences between the experimental class and class C1 are not significant here, but the differences between these two

classes and class C2 are. This finding is interesting because it might be that the experience of Project Headlight students (experimental class and class C1) with Logo programming contributed to their general mathematical ability.

SAMPLE RESULTS FROM THE LOGO POST-TESTS

In the Pencil-&-Paper Log Test the students were asked: *"Please list all the Logo instructions and commands that you know and use—in column A; then, write an explanation and give an example for each one— in column B."* The results for this question were divided into two major groups of findings. The first are simple findings that relate to how many instructions and commands each child actually listed. The second relate to the children's understanding of the meaning and functions of these commands and instructions in the Logo language. Table 7 represents the differences between the students in terms of how many Logo commands, operations, function keys, control keys, etc., they listed in the post-test. The number in each slot shows the average number of commands and instructions the children from all three classes listed and explained. The advantages of the experimental students over the students from the two control classes become clear from examining this table.

Table 7. Contingency Table Analysis, Comparing the Average Number of Listed Logo Commands between the Experimental and Control Classes

Experimental Class	Control 1 Class	Control 2 Class
25.6	12.0	8.3

$\chi^2 = 10.8426$

The students were also evaluated on the quality of their definitions and examples for each of the items they had listed. In class C1 no one was evaluated as "Very Good," whereas in the experimental class, three students who wrote over 40 commands and instructions, and four who wrote over 30, and gave very good examples and definitions of each, were evaluated as "Very Good." No one was evaluated "Low" or "Very Low" in the experimental class. However, four students in class C2 were evaluated as "Low" since they listed fewer than five commands and instructions and did not provide examples or definitions for all or most of those.

We also tested the children's ability to analyze given programming code and "execute" it on paper. A long, linear Logo code composed of short strips of Logo primitives was given to the students, and the students were asked to draw the graphics. This task required that students read the given linear code, comprehend it, understand its flow of control, build a mental model of what the computer would do when each of the lines in this program was executed, and draw the picture accordingly, step by step.

Many researchers in the field of programming distinguish between writing a linear program and a modular program. These researchers consider a linear program as one which emphasizes the generating of effects without any consideration and understanding of the inner structure of the code (e.g., Papert, 1980; Papert, Watt, diSessa, & Weir, 1979; Carver, 1987; Soloway, 1984; several researchers in Pea & Sheingold, 1987; and others). On the other hand, a modular program emphasizes elegant and efficient programming, and is accompanied, they claim, by a higher-level

of understanding of programming in general, and of the programming language characteristics in particular.

Our results show that students who had written linear as well as modular programs during their process of learning to program were better able to understand and correctly execute this confusing linear program. The students in C2, who only knew how to write linear programs, were not able to solve this problem accurately unlike many of the ISDP students. We should note that ISDP students often introduced structure (i.e., subprocedures and functional naming) into their programs only after a long period of purely linear programming, and only when they themselves decided it was necessary; it was not imposed on them from the outside. They learned to introduce structure, modularity, and elegant coding when they themselves realized the need for it in maintaining their *long* programs, in adding new parts to them, or in re-using (instead of re-writing) certain subprocedures in several places in their programs.

Another interesting aspect of these results came to view in the "number of trials" category. Many of the ISDP students tried more than once to draw the picture on paper, and finally found the right solution; but the students in the control classes who had gotten it wrong in their first trial were apparently not motivated or determined to try again or to find the right solution. Many of them simply wrote "I don't know how to do it," and went on to the next task on the test.

Finally, we mention that on a "Debugging Task" given to the students on the computer, the ISDP students were faster at identifying the bugs, locating them, and then re-evaluating the program in order to create an output that corresponded perfectly with the original goal given to them. The data in Table 8 shows the results for "Tasks 1 and 2" on the computer, which required that the students run a given bugged program, analyze the features of the resultant graphics, identify the discrepancies between them and the desired graphics, enter the Logo code on the computer, locate the different bugs causing the discrepancies, fix the program on the computer, and add the corrections on the program that were written on the paper.

Table 8. Results from the Debugging Task

	No. of Bugs Found and Fixed	Identify & Fix Bugs in *Computer* Prog.	Identify & Fix Bugs in *Paper* Program	Average Time for Solving 1 & 2
Experimen. class $n = 17$	16—all bugs 1—one bug	17 children—yes 100% succeeded	17 children—yes 100% succeeded	15 min per child
Control 1 $n = 18$	9—all bugs 4—one bug 5—none	13 children—yes 5 children—no 70% succeeded	8 children—yes 10 children—no 44% succeeded	35 min per child
Control 2 $n = 16$	2—all bugs 4—one bug 10—none	6 children—yes 10 children—no 37% succeeded	2 children—yes 14 children—no 12% succeeded	55 min per child

Table 8 speaks for itself. The superiority of the ISDP students over the other pupils is clear, as is that of class C1 over class C2. Table 9 shows a Chi Square Analysis of these results.

Table 9. Contingency Table Analysis, Comparing the Number of Bugs Found in the Debugging Task between the Experimental and Control Classes

	Experimental Class (%)	Control 1 Class (%)	Control 2 Class (%)
Bugs found:			
2	94	50	13
1	6	22	25
0	0	28	62

$\chi^2 = 138.92$

In addition to the above quantitative results we made a number of qualitative observations about the children's debugging strategies. For example, the first thing all the ISDP students did was to change the HT (Hide Turtle) command at the very beginning of the procedure, to ST (Show Turtle), so that they could follow the turtle as it executed the code. On the other hand, the first strategy that most of the students in class C2 and many in class C1 used was to copy the program given to them on paper (in sub-task 2) into the Logo Command Center and execute it line by line. This strategy worked well until they reached the REPEAT statements, which were written on more than one line. Then, the students got confused because the program still did not work, though they were sure that they had located a bug. Instead of trying a new strategy, these students then erased everything and started to copy the procedure into the computer in "direct mode" again, which resulted in the same thing happening again, and so on.

In "Tasks 3 and 4 on the computer," the students were asked to optimize the code given to them in Tasks 1 and 2, and make it clearer and shorter. In order to solve these sub-tasks, the students had to cease operating on the individual command level, and start thinking in a procedural mode, using REPEATs, procedures, and inputs. To summarize these results, the experimental students were more flexible and attempted to explore a greater variety of ways for producing the same Logo drawings. They understood and reached a more modular level of code, and many of them tried to use repeats, sub-procedures, and variables. The experimental students also performed significantly better than the control students on the three other items of this test, covering use of inputs, modification of procedures according to specific requests, and prediction of results of short but confusing graphics programs (see Harel, 1988; 1989b. d, e).

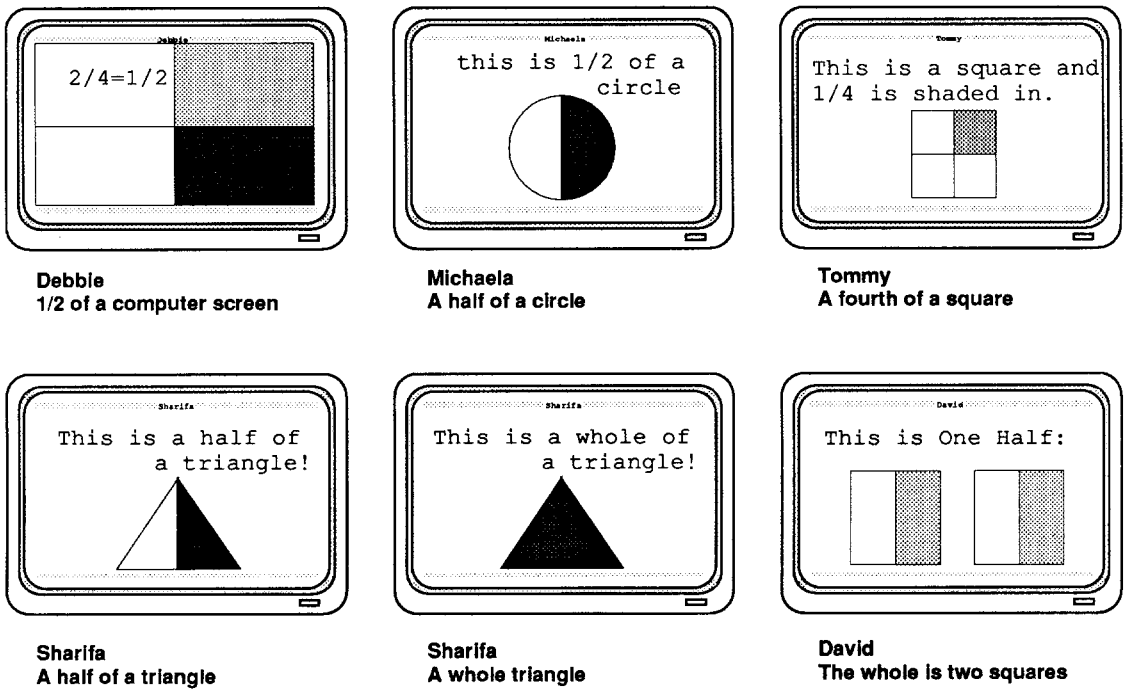
Interestingly, all the ISDP students, who had already performed much better than the control students in the similar pencil-&-paper tasks, performed even better when using the computer. But the students from class C2 got more confused at the computer, and performed less well than they had on the pencil-&-paper task. Class C1 was somewhere in between: the high-math students, like those from the ISDP class, performed much better at the computer, and the medium- and low-math students performed similarly to those from class C2— far less successfully than they had in the pencil-&-paper task.

Similar trends were found in the results of the Logo post-tests and in the Fractions post-test: the ISDP students consistently scored higher than the other two classes; but class C1 usually scored higher than class C2. Also, the high-math students from class C1 made up a special group. They were never as good as the high-math ISDP students, but most of the time they were as good as the medium-math ISDP students. Their scores in the fractions test were often higher than those of the students from the RN Project, and stood out from those of the other control students. What does this mean? It seems as though only the high-math students in class C1 strongly benefited from Project Headlight experience with respect to the pictorial-to-symbolic translation of fractions. This was probably due to their programming expertise, which contributed to their ability to

translate picture representations into written ones, and vice versa. This phenomenon requires further investigation. It is an interesting one, since it suggests a correlation between the children's level of understanding and involvement in Logo programming, and their ability to understand different representational systems.

QUALITATIVE RESULTS ABOUT *WHAT AND HOW* THE STUDENTS LEARNED

Thicker descriptions than "getting better at" fractions or Logo in the school's terms were derived from an analysis of a large body of qualitative data derived in three ways: formal interviews, preservation of students' work, and observations of process. The 51 students in the experimental and control groups were interviewed before and after the ISDP experience. The ISDP students' work was preserved in Designer Notebooks and in computer files showing the state of their software projects at the end of each day. In addition to direct daily observations by the researcher and teacher, videotape made in two modes gave many opportunities for micro-analysis of behaviors: in one mode the video camera was carried by an observer and directed at interesting events, in the other it was placed in one position on a tripod for an entire session and simply allowed to run. These sources of data allowed us to see subjects discovering new ways of talking about fractions and relating to fractions spatially and kinesthetically as well as linguistically and conceptually (e.g., Harel, 1 990b).



Debbie
1/2 of a computer screen

Michaela
A half of a circle

Tommy
A fourth of a square

Sharifa
A half of a triangle

Sharifa
A whole triangle

David
The whole is two squares

Figure 6. Some Children's Initial Representations on the Computer

The interpretative nature of such conclusions required rigor that is different in kind from statistical analysis that checks whether or not the probability of differences in scores could be due to chance. But it is the richness of observation obtained from so many different sources that yielded a coherent sense of the development of individual subjects as well as of shared

developmental trends, and this gave us confidence in our conclusions that we could not have obtained by any other means. To appreciate this coherence in full it is necessary to refer to finer textured case studies published elsewhere (Harel, 1988, 1990a). Here we focus on four issues which we label as *development of concept, appropriation of project, rhythm of work,* and *cognitive awareness and control*.

Development of Concept. Under the rubric development of concept we analyze the movement from rigidity, particularly, and isolatedness toward flexibility, generality, and connectedness. In the initial interviews questions such as "What is a fraction?" or "When you close your eyes and think about fractions what images do you have?" or "Can you give me an example of a fraction?" revealed several aspects of particularity. There was particularity in the use of particular rational numbers (usually one half or one fourth) as prototypes. Most strikingly there was particularity of restriction to the spatial: A fraction is a part of something, and "something" means something physical or geometrical. Of course children from an early age use fraction words linguistically to refer to parts of other kinds of entities, such as time ("half an hour" or "I am eight-and-three-quarters") and money ("a quarter"). But in the interviews they very seldom seemed to connect such usages to a general notion of a fraction. When specifically prompted to look for fractions in a real calendar or clock, subjects gave answers referring to the squares on the calendar or shapes on the clock face. One student even referred to the pattern strap-watch-strap as analogous to the numerator, the slash, and the denominator in the school representation of fractions! And even within the spatial there was a high degree of particularity in choosing examples that happened to coincide with those one expects to meet in school books: "a fraction is a half a pie" or "a fraction is like an apple or an orange divided in the middle." When asked to draw a fraction most commonly they would draw a circle or a square, divide it vertically (not necessarily equally), and shade some parts. In some cases the degree and rigidity of the particularity bordered on the bizarre. For example, Debbie was committed to the idea that a fraction is the right shaded part of a circle divided by a vertical diameter. When asked whether the unshaded part of the circle is a fraction, she said, "No. It's not a fraction. It's nothing." Such tendencies were also seen in the choice and modes of representation of fractions in the very first examples of computer screens made in the experimental students' software projects.

All this changed dramatically in the course of the project. The content of the software as well as the post-interviews revealed a widening diversity of kinds of examples and representations among the ISDP students. Even more significantly, there was often a conscious—indeed, one might well say philosophical, recognition of the achievement of greater generality. In Figure 7 we show a few examples of some children's further representations. Although it is difficult to capture the colorfulness and playfulness of those animations in this static black and white medium, the children's general ideas, their diversity, and complexity is captured here.

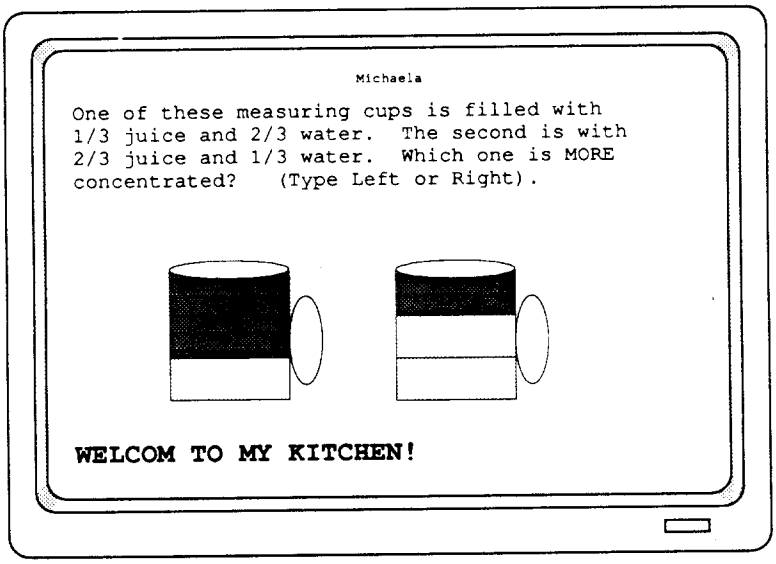


Figure 7a. Michaela's Kitchen Scene

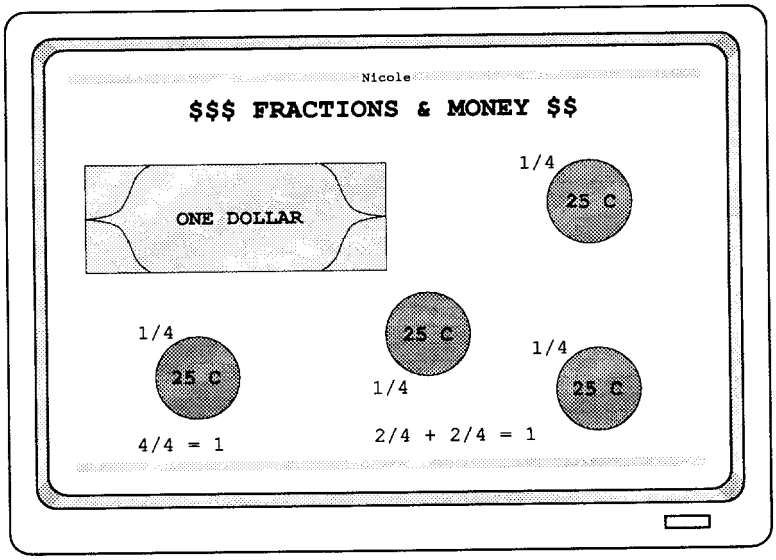


Figure 7b. Nicole's Money Scene

Consider Debbie again. After a whole month of explaining about fractions—by creating a representation showing a half of her computer screen, and different geometrical shapes divided into halves and fourths—Debbie discovered something. Her discovery was expressed in her choosing to teach an idea of a different, more "philosophical" nature than how to cut a shape into thirds or how to add a third and a half.

She chose to explain that, "there are fractions everywhere... you can put fractions on anything." To teach this idea, Debbie designed a representation of a "house, a sun, and two

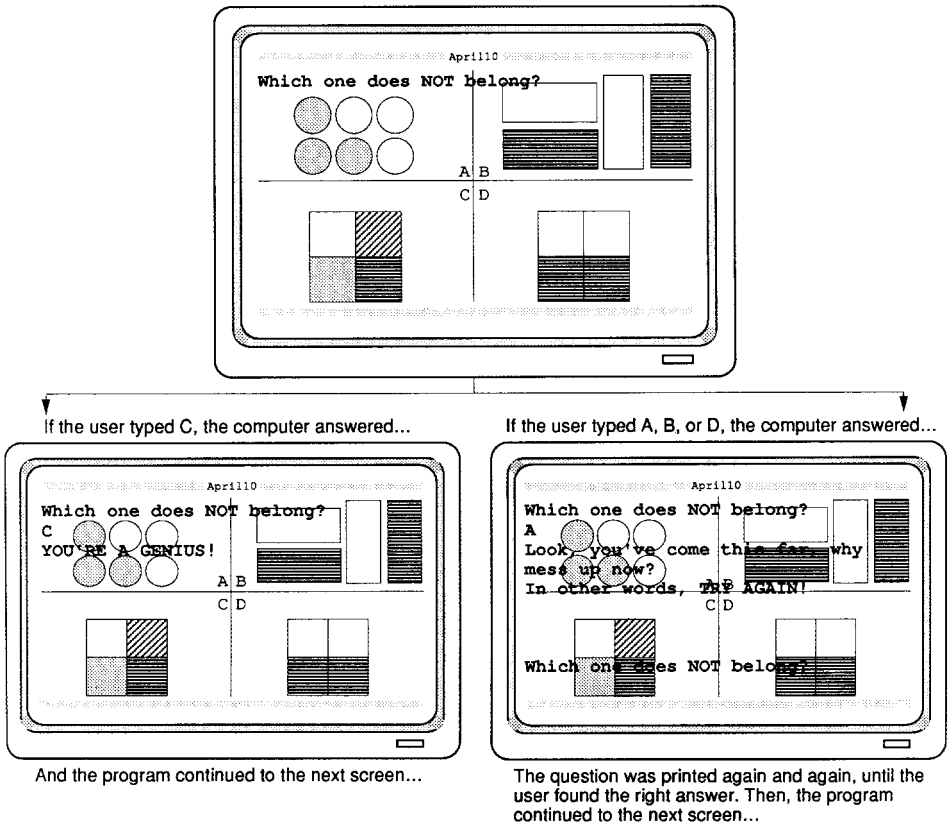


Figure 7c.

wooden wagons" (see Figure 1). She worked very hard on implementing this representation using some quite complex Logo programming code (see Harel 1988, pp. 118-140 for a detailed description of her lengthy and complex programming process and her work on this particular screen).

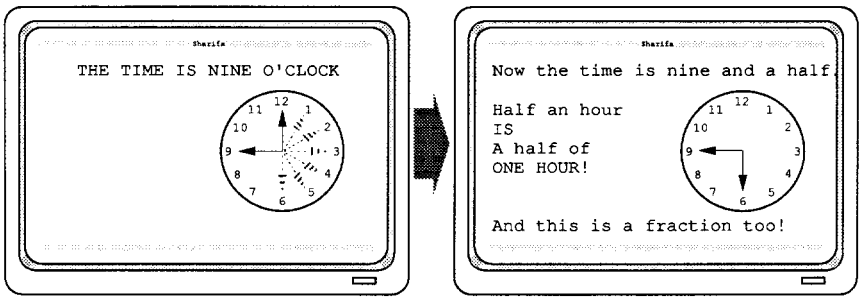


Figure 7d.

Debbie was not alone. A few weeks later, Tommy's House appeared, and then Paul's. The idea that it is important to teach others that "fractions are everywhere," and that one could "find fractions in regular human things" was spreading around the Design Studio.

Michaela and Sharifa, who used Debbie's software and received her full set of explanations about it, also chose to teach the same principle, but in another way. Sharifa selected to represent

fractions by using a clock, teaching her users that “Half an hour is a half of ONE hour!” Her enthusiasm in announcing to the world that, “half of an hour is a fraction too!” (and her use of exclamation points) is evidence for the philosophical importance of the breakthrough as she experienced it. Michaela chose to teach this principle through using a representation of “two measuring cups filled with different quantities of orange juice, water, or flour—depends on the fraction...” Later she confessed, “I found so many fractions in my kitchen ... I told my mom about it too...”

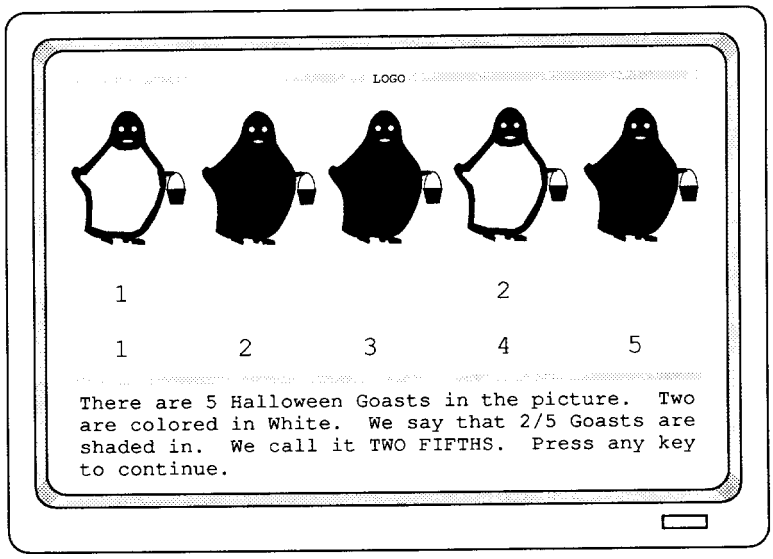


Figure 7e.

These observations are consistent with the ways in which ethnographers such as Scribner and Lave (1984) have demonstrated the separation of school knowledge of mathematics from practical, everyday knowledge. But we note something further that has a disquieting as well as an educationally hopeful aspect. The disconnection seems to be well entrenched within both the *practical* and *everyday* side and on the school side, as shown for example by the fact that Sharifa had to discover a connection between “half an hour” and “half an apple.” On the other hand, she did make the discovery, and did so without explicit or directive prompting by adults. Similarly, we see clear evidence that many students do not see “a quarter” as related to either the 1/4 in their school worksheets or the “25% off” in a store's sale pricing.

Our conjecture is that “disconnection of knowledge” must not be seen primarily as a limitation of “schoolish” knowledge but rather as a universal characteristic of how knowledge develops, first as “knowledge in parts” (to use Andrea diSessa's phrase) and then by the unifying effect of control mechanisms such as those described by Lawler in *Computer Experience and Cognitive Development* (1985), by Minsky in *Society of Mind* (1986), and by Papert in *Mindstorms* (1980).

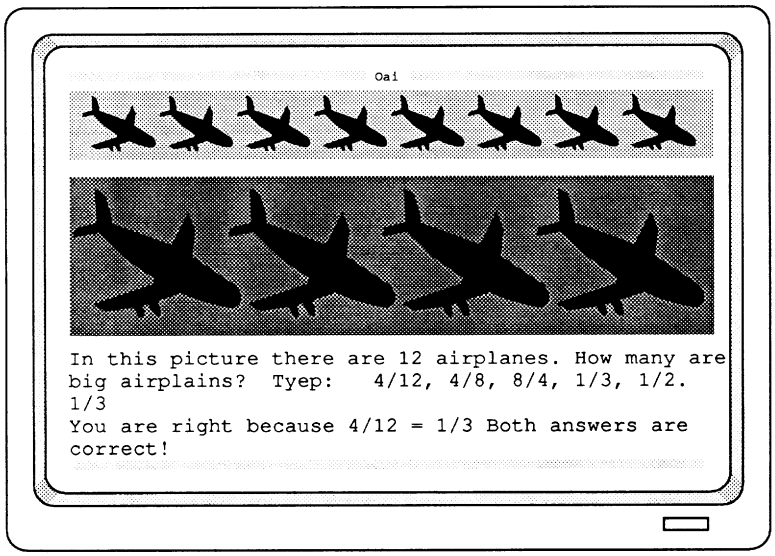


Figure 7f. Pay attention to the options Oai gave to his users (4/12, 4/8, 8/4, 1/3, 1/2). He probably had an idea of what could be their problems with understanding this representation. He programmed it so both the answers 4/12 and 1/3 would receive the same feedback ("you are right . . ."). He also had an explanation for the users for *why* their answers were incorrect.

Appropriation of the Project. Our second rubric, appropriation of the project, refers to observations about a shift from a reluctant, impersonal, and mechanical mode of working to a growing personal engagement, assertive individuality, and creativity. Debbie's case once more illustrates this process. Her initial response to the project was globally very negative. She simply did not want to develop software about fractions. In the culture we tried to maintain, she was allowed to hold back but gradually began to succumb to generalized social pressures. So by the end of the second week, she was beginning to put fractions on her computer screen. But what she put up was still a direct reflection of the stereotyped model of fractions she had derived from math class. However, a new process was also beginning. We would say that she was "working through" her ideas (and no doubt her feelings too) about fractions. It took her approximately a month to achieve her break-through. Now she had an individual philosophical position which she pursued with something of a missionary zeal. She had given herself the task of leading the rest of the world to her discovery.

Time Frame and Rhythm of Work. This category appears to be an essential element of the process of appropriation. Switching in and out of projects in the fragmented time of the regular school, simply does not provide the conditions for personal appropriation and expression of personal intellectual style. Observations in the ISDP also show the importance of pace in the student's rhythm of daily work and in the radical differences in individual style of work, action, and thought. Analysis of videotapes set up to run continuously at fixed places show a pattern of work in striking contrast with the regular school notion of "efficient time on task." In the videos, we do see periods of intense concentration. But we also see periods in which students' attention is elsewhere: sometimes looking at a neighbor's work, sometimes engaged in play, chatting, and interactions that have no discernible connection with the project. Is this an "inefficient" use of time? While we did not measure this with any rigor, it appears to us that the rhythms of work adopted by the individual students have an integrity that contributes to getting the job done and especially to getting it done creatively. And in making this assertion we feel supported by such ethnographic studies as Bruno Latour's (1987) description of the ways in which engineers and scientists at work

mix “serious” talk about the problems in hand with intrusions from everyday life and personal concerns.

Metacognitive Awareness. In this rubric we describe in what ways ISDP encouraged children's metacognitive awareness (i.e., children's thinking about their own thinking), their cognitive control (i.e., planning, self-management, and thinking about these processes), and their metaconceptual thinking (i.e., children's thinking about their own knowledge and understanding of concepts).

Through the project the students developed *problem-finding skills*. For four months, students involved themselves in discovering problems they wished to solve. No one specified the problems for them; rather, they were the ones in charge of deciding, for example, what was difficult about fractions, what screens to design to explain fractions, what Logo procedures to create and how, etc. Students also developed an awareness of the skills and processes needed to solve the various problems they posed. The Designer's Notebooks, as another example, required that children design and think about their screens on paper. Their initial drawings and plans demonstrated that they were not very aware of either the programming or the fractions knowledge and skills needed to accomplish their designs; however, as the project progressed they rarely came up with a design they could not manage in Logo. They also had to be aware of their target users' knowledge of fractions so that they could make the representations they had created on the computer comprehensible to them. Not only did children become *aware of strategies* to solve a problem at hand, they also learned to *activate* them. The Logo post-tests, for example, showed that the experimental children were able to optimize, modularize, and debug Logo procedures better and faster within given time constraints.

Over the course of the project, children developed the ability to discard inefficient designs, plans, and solutions and to search for better alternatives. In other words, they developed *cognitive flexibility*. During the project they learned to adjust their cognitive efforts to match the difficulty of the problem. They would often begin to implement their designs in Logo, but when they realized that too much effort was needed to accomplish a simple or “unimportant” design, they stopped working on it and moved on to a screen that was more crucial for their software or decided to redesign the screen that was giving them problems. As a result the ISDP students were not rigid in their solution processes in the Logo posttests, and did not stop working on difficult problems (unlike many of the control children who simply answered “I don't know”), but kept trying until they found the solution.

Another thing they learned was how to *control distractions and anxiety*. In this project (and in Project Headlight in general), children worked in an open area next to their classroom. Different children worked on different problems, with other children, teachers, and visitors often walking around. Children learned to keep their attention focused on the problems they were working on, and to resist being distracted by external stimulation. They also learned to control their anxiety when a problem was difficult. Post-tests showed that Project Headlight children (both ISDP and C1) did better in avoiding anxiety, focusing efficiently on the problems given to them and not letting external interference distract them from their thinking and writing.

The community supported a *practice of continual evaluation:* Children evaluated their own and each other's performance every day when they ran their software and made entries in their Designer's Notebooks, and when they looked at other children's software—sometimes making suggestions or borrowing ideas. They were constantly relating their current performance and implementation phases to the general goals of the task and making appropriate changes if the result was too slow or unclear.

The students learned to *monitor their solution processes*. Since they were in charge of their own learning and production, they knew that when they had a problem or difficulty they could look first to themselves for a solution. They developed self-reliance and faith in their thinking.

Finally, the students *became articulate* not only about general planning and specific design tasks, but about the subject domain as well. They talked, thought about, and actually related to fractions, both during their involvement in the project and in the interviews and tests that took place afterwards. From their point of view, it was having to teach and explain fractions to someone else that caused them to embrace it so thoroughly because, as they said, "how can you teach it if you don't know it yourself?" Much like professional educational-software producers, who gain deeper understanding of the topics involved in their software by thinking of ways to build explanations and graphical representations for their future software users— the experimental children, through teaching and explaining, also gained an awareness of what fractions were or of what they knew and did not know about fractions. To give some examples of students' metacognitive expressions, here are four related quotes from the post-interviews.

Andy: *"It's supposed to be for littler kids, right? But to program it so they can understand it, you have to be sure that you know what you are talking about. 'Cause the teacher has to know more... You don't know how the other kid will react to it and all of that... it was really hard to get it so they will like it... Always to think about and imagine that you are small, right, and how would you like it?!"*

Naomi: *"It is hard to teach. You have to have a pretty good understanding of something, so you'll be able to explain it well to others... and a lot of times it's really hard to understand what's happening with these fractions..."*

Debbie: *"You have to show them fractions and explain, little by little. To program the scenes, so they will learn how to do fractions, and what they did wrong... then, someone can listen to you, to the computer, I mean, and understand."*

Paul: "It's hard to tell someone else that doesn't'' know about fractions how to do these things. So I program this software for them, to help them understand it... But I have to think a lot *about what I really know and how to show it on the computer, and how to explain it. And at the end, how to test them about it."*

DISCUSSION: WHY DID THEY LEARN?

The simplest description of the ISD experiment reads like a "treatment" type of experiment: These subjects did something particular (made instructional software) for so many hours (close to 70 hours of work). In fact, the situation is vastly more complex than anything that could be sensibly described as "changing one variable while keeping the everything else constant" because there were too many particulars involved. To make their pieces of software, the students used particular computers (IBM PCjrs) and a particular programming language (LogoWriter). The project included focus sessions where the specific content of fractions was discussed in a particular way—informally and compared with school classes, briefly. The project took place in a particular part of the school with a particular "computer culture." And during the ISDP the culture developed further in a particular way, with particular customs of interaction, attention, mutual help, secrecy, humor, and so on. The students and their teacher were aware of having a unique relationship with the experimental staff. They reacted in particular ways to the presence of video cameras, question-askers, and note-takers.

One can raise innumerable conjectures about the "real" source of their learning about fractions, for example. Did the simple fact of spending some 70 hours programming representations in Logo contribute to the results? Was the "moral climate" in the project largely responsible? Or the fact that the teacher felt she was part of something important or simply different? Some such conjectures, or

aspects of such conjectures, we can, and do, try to check by studying control groups. But there are far too many of them to treat in a rigorous way.

What can be said with some certainty is that we created a *total learning environment* in which some impressive learning took place. Teasing out the contributions of particular aspects of the environment is not a reasonable goal for any single well-defined experiment. Understanding will come through a process of gradual accumulation of many projects and of a great deal of theory building (e.g., Kafai & Harel, 1990; Jackson, 1990; Resnick, 1989). What we can do here is to share our own intuitions and, as part of the larger scientific enterprise, to formulate and discuss some conjectures concerning these intuitions of ours.

In the following sections we speculate that improvement in performance might be affected by factors related to the *affective side* of cognition and learning; to the children's process of *personal appropriation* of knowledge; to the children's use of *LogoWriter;* to the children's constructivist involvement with the deep structure of fractions knowledge (namely, *construction of multiple representations)* to the *"integrated-learning"* principle; to the *"learning by teaching"* principle; to the *power of design* as a learning activity.

However, the main point we would like to make here is that each one of these conjectures, when considered alone, would only give very partial information about why ISDP took the form and yielded the results that it did. Only by considering them together, and by speculating about their interrelations, can we take a step towards understanding the holistic character of Constructionism in general and of ISDP in particular.

The Affects of Affect

From certain Instructionist points of view (e.g., Papert, 1990) one could see a paradox in the results obtained here. Here are a few examples from the ISDP students' test scores: Debbie scored 51% correct on the fractions pre-test and 84% on the post (33% difference); Casey scored 55% on pre and 83% on post (28% difference); Rachel, 55% on pre- and 87% on post (32% difference); or Oai, 55% on pre- and 97% on post (42% difference). Debbie's, Oai's, Casey's and other children's ability to work with fractions, improved considerably from working on a project that was entirely self directed, gave them no "feedback" in the form of marking responses right or wrong, gave them very little guidance or information about fractions. How could worrying about whether "fractions are everywhere," to take Debbie's concern as an example, lead to greater ability to do school problems in manipulating fractions?

The "obvious" explanation, which nevertheless surely has more than a little truth, is that the students developed a better attitude towards fractions, perhaps even came to *like* fractions. We recall that Debbie was initially reluctant to have anything to do with such stuff but ended up with enthusiastic missionary zeal. One does not need any complex theory of affectivity to conjecture that she might therefore be more likely to engage her mind with fractions both in the regular math class, so that she would learn more of what she wanted to teach, and in test situations, so that she would score more.

Pursuing the idea that Debbie changed her "relationship with fractions" leads into an area where the line between the affective and the cognitive becomes hard to maintain (e.g., Turkle, 1984; Turkle & Papert, 1990). We see something happening that is analogous to the development of a greater intimacy in relationships with people. Debbie becomes willing to take more risks, to allow herself to be more vulnerable, in her dealings with fractions. As long as fractions-knowledge was teacher's knowledge regurgitated, she was emotionally safe; the risk of poor grades is less threatening than the risk of exposing one's own ideas.

Our view of people like Debbie is strongly colored by the sense that when they allow themselves to tap into personal knowledge, they allow knowledge about fractions to become connected with the personal sides of themselves. We conjecture that improvement in performance is related to the extent to which the students respond to a problem about fractions by "digging around" in their own stocks of knowledge as opposed to trying to follow set procedures. We note that this point could be formulated in Scribner's (1984) language by saying that their thinking about fractions shifts from scholastic intelligence, characterized by rigid, inflexible, externally imposed methods, to practical intelligence characterized by the use of multiple, flexible, and personal methods.

The Importance of Situatedness

The idea, though not the word, is an important theme in the development of Logo-based Constructionism (Ackermann, 1990; Papert 1980, 1984a-b, 1987). In this spirit we attribute the fluency with which our subjects work with fractions to the fact that this knowledge is situated in computational microworlds, much as Jean Lave's weight watchers benefit from the supportive consequences of the fact that fractions are situated in the micro-world of the kitchen. A similar example is how Michaela was able to grasp fractions' significance in the context of using cooking tools for representing fractions. An even more striking example is provided by Sharifa, who got a grasp on the fractional nature of time through support from an overlap between the way the clock face represents fractions as angles, and the way in which the Logo turtle (which by then was familiar to her) does something very similar.

In that sense, our observations are consistent with those of Lucy Suchman, Jean Lave, and John Seeley Brown about "situated knowledge." Like these researchers, we are strongly committed to the idea that no piece of knowledge stands and grows by itself. Its meaning and its efficacy depend on its being situated in a relation to supporting structures. However, we attach more weight than we think those writers do to the Society of Mind metaphor (e.g., Lawler, 1985; Minsky, 1986; Papert, 1980) which would allow the situating of knowledge in internalized, mental environments to act in much the same way as situated in external, physical environments. Looking at the performance of Sharifa from this point of view we would say that her work with the computer enabled her to bring together *in her thinking* mutually supportive internal microworlds, in this particular case, microworlds of clock-time and of simple fractions.

The Contribution of Logo

There is a body of literature that addresses the question whether "programming" in general or "Logo" in particular can induce cognitive effects, and if so to what extent. In this sense, Logo would be seen as a causal factor in the improvement of fractions-knowledge or cognitive skills seen in our study.

But Papert (1987) has used the term "Technocentrism" to warn against simplistic forms of this question. In different contexts the import of the phrase "learning Logo" can differ so greatly that the question borders on meaninglessness. Nevertheless, in the particular context of the ISD Project, where Logo was not isolated from a total context, and where students programmed intensively and extensively, one can meaningfully *begin* to ask how various features of Logo contributed to the success of the children's work.

At least one important contribution of Logo in this study was *indirect*—having less to do with acquiring cognitive skills than with mastering a subject domain— learning how to program and

using Logo enabled these students to become more involved in thinking about fractions knowledge.

But we do think that Logo, because of its structure (or ISDP, because of the unique way it used the structure of Logo), had a direct affect. Sheriff's ability to see the analogy between the clock and the turtle is one example of these affects. Our conjecture here, stated in its most general form, is that the structure of Logo brings students into direct and concrete contact with issues of *representation*—in the case of ISDP, representation of the specific object of study, fractions; and more generally, with the representation of objects, projects, structures, and processes in terms of subprocedures, LogoWriter pages, and other computational structures.

It is relevant to note that much of what the ISDP students did could in principle be done by other methods, such as using pencil and paper to draw representations, or using physical manipulatives of various kinds (for representation construction). This might seem to make the contribution of Logo quite incidental. But in practice, we find it implausible that traditional media could equal the ease with which Logo allows students to save and connect concepts and their different representations, and especially how it allows them to develop and modify such representations over long periods of time. Even more important, working in Logo on one's own machine, in a culture where that's what everybody else is doing, reinforces the learner's contact with his or her personal knowledge that is expressed in a real product—a piece of software—that can be used and re-used by oneself or others, changed, modified, and grow with the knowledge of the learner and of the culture. Logo facilitated this ongoing personal engagement and gradual change of knowledge; and at the same time, it also facilitates the sharing of the knowledge with other members of the design studio, and it allowed learners to continue and build upon their and others' ideas and comments very easily. Logo facilitated communications about the processes and acts of cognition and learning.

Of course we do not maintain that only Logo could do this. Surely, many new media will develop that can do it better. But looking carefully at the features of Logo that contribute here, and the ways it was used in the ISDP context, will be of use in guiding such developments (e.g., Harel & Papert, 1990). Pursuing such issues requires much further research. However the research that will elucidate them is not well guided by the kind of questions that have often been posed in the literature, such as "Does Logo have such and such a cognitive effect" but rather "Can Logo be used to amplify and support such and such a direction of children's intellectual development, or such and such a change in a learning culture. "

The Deep Structure of Rational-Number Knowledge

Whereas most school work touches only on the surface structure of rational-number knowledge, we believe ISDP puts students in touch with the deep structure.

Elementary-school children's processes as well as difficulties in learning fractions and understanding their representations have been well documented. Unlike whole numbers, the meaning of which students largely come to grasp informally and intuitively out of school, learning the rational-number system is confined almost exclusively to school. Because rational-number concepts and algorithms are so difficult for so many pupils, they figure prominently in school curricula from the second grade on, mainly in the form of algorithmic tasks and the working out of specific well-defined mathematical problems. Even so, several national assessments have found that children's performance on fraction ordering and computation was low and accompanied by little understanding (see the discussion of this topic in Harel, 1988; 1989 ibid, 1990a). This is particularly unfortunate because fractions are ideal tools for learning about number systems and representational systems in mathematics.

We see the understanding of the rational-number representational system as a privileged piece of knowledge among the other pieces of rational-number knowledge. Representations form part of the deep structure of rational-number knowledge, whereas algorithms put students in touch with only the surface structure (e.g., Janvier, 1987; Lesh & Landau, 1983).

Logo can be a direct route to this encounter with the deep structure, enabling students to explore the concept of fractions through various on-screen representations of their own devising. In ISDP, this process was catalyzed by setting students the task of creating good pedagogical aids for other students, in the course of which they thought to create fractions representations in such forms as money, food, or clocks, as well as geometric shapes, and to accompany them with symbolic or verbal explanations, they thought would be helpful to their target audience.

By becoming designers of instructional software, the students gained distance and perspective in two senses. In the first place, they were dealing not with the representations themselves, but with a Logo representation of the representations. Moving between representations was subordinated to programming good examples of representations. Secondly, the students programmed, not for themselves, but for others. They had to step outside and think about other children's reactions. The depth and creativity of such an experience contrasts with the rote, superficial quality of what typically occurs when a student is put through the paces of an externally conceived sequence of learning.

In summary, ISDP recast fractions learning in essentially three ways:

(1) it emphasized more involvement with the deep structure (representations) over the surface structure (algorithms) of rational-number knowledge;
(2) it made fractions learning simultaneously incidental and instrumental to a larger intellectual and social goal, that is, having students think about and explain what they think and learn, in an interactive lesson for younger children; and
(3) it encouraged both personal expression and social communication of rational-number knowledge and ideas.

The "Integrated Learning" Principle: Learning More Can Be Easier Than Learning Less

It must be admitted that there are certain problems with integrating instructional software design activity into a school's curriculum. Software design is a time-consuming and complex enterprise for a teacher to handle, and it is not yet clear how it can fit into the average class schedule. Also, at the present time, it is not very clear which school subjects would lend themselves best to this process of learning (e.g., Jackson, 1990; Kafai & Harel, 1990).

But knowledge about computation (such as programming) and the sciences of information (involving control over one's own processing, metacognition, and information construction) has a special character in this respect because it has a reflexive synergistic quality—it facilitates other knowledge. In ISDP, the learning of fractions and the learning of the complex of skills (programming, design, etc.) encompassed in the phrase "software design" did not compete for time; rather we maintain that each took place more effectively than would have been the case had they been taught separately.

The reflexive quality of information science offers a solution to the apparent impossibility of adding another component to an already full school day. If some knowledge facilitates other knowledge, then, in a beautifully paradoxical way, more can mean less!

The idea that learning more science and math necessarily means learning less of something else shows a wrong conception. If these domains are properly integrated into individuals' knowledge and into learning cultures, they will be supportive, not competitive with other learning. We believe in the possibility of integrating science, mathematical concepts, art, writing, and other subjects and making them mutually supportive. We also believe that in ISDP this principle of integration—which meant that young students learned fractions, Logo programming, instructional designing, planning, story-boarding, reflection, self-management, etc. all at the same time and in a synergistic fashion—greatly contributed to the results.

Special Merits to Learning By Teaching and Explaining

As educators or teachers, producers, computer programmers, software developers, or professional people in general, we are rarely encouraged to draw on our own learning experiences in order to better understand the reasons, purposes, and processes of learning and teaching our subject matter. Too often we tend to forget what was really difficult for us to understand, or why one learning experience was more or less valuable for us than others in the course of our own intellectual and professional development.

It has been observed by students and educators in our group as well as by many "experts" that the best way to learn a subject is to teach it. Let us consider for a moment, experiences that are common to professional people in all fields in the course of their everyday work or professional training. Teachers, for example, often remark that they "finally understood something today for the first time" when a student asked for an explanation of something he did not understand. Some of our friends (professional computer programmers) at MIT have told us that they "really" learned how to program when they had to teach it to someone else—or when they were involved in a real, complex, long, and meaningful programming job. Many university professors choose to teach a course on the theory of topic of their research while they are actually working on it; so that the process of teaching and discussing their work with students, will enable them to clarify and refine their own ideas and theories. And it certainly seems to be the case in the educational software field, that the people who are having the most fun, and are learning the most, are the software designers and programmers. With most educational software today, especially the drill-and-practice kind, the users rarely gain *deep* understanding of the concepts taught, unless the software is supplemented by instruction and explanations from a good teacher. But the designer, who spent a long and intensive period of time designing, learning, and thinking of ways to build explanations and graphical representations for given concepts (even for the simplest form of educational software), has probably mastered these concepts and gained a much deeper understanding of them than they were able to convey in the software product itself.

The intellectual benefit of generating one's own explanations have been stressed by a number of theorists. Piaget, for example, has argued that higher level reasoning occurs in a children's group in the form of arguments. These arguments, according to Piaget, help children construct and internalize ideas in the form of thought. Such observations prompted Piaget to conclude that the very act of communication produces the need for checking and confirming one's own thoughts (e.g., Piaget, 1953). Furthermore, in the Child's Conception of Space (1967), Piaget emphasizes how difficult it is for young children to decenter— that is, to move freely from their own point of view to that of another, in either literal or metaphorical senses. Increasing communication develops the child's ability to decenter, and to come closer to an objective view of the whole. The process of decentering, says Piaget, is fundamental to knowledge in all its forms.

Among contemporary researchers, Brown, for example, has done many studies to elucidate the ways in which explanatory processes, as part of reciprocal teaching activities, motivate learners and encourage the search for deeper levels of understanding and subject mastery. Brown characterizes these explanatory-based interactive learning environments as ones that push the learners to explain and represent knowledge in multiple ways and therefore, in the process, to comprehend it more fully themselves. The interactions could be supported by computers, teachers, or other learners (e.g., Brown, 1988).

Hatano and Inagaki (1987) also argue that comprehension and interest is enhanced where students have to explain their views and clarify their positions to others. In the process of trying to convince or teach other students, they explain, "one has to verbalize or make explicit that which is known only implicitly. One must examine one's own comprehension in detail and thus become aware of any inadequacies, thus far unnoticed, in the coordination among those pieces of knowledge." Their studies demonstrate how persuasion or teaching requires the orderly presentation of ideas, and better intra-individual organization of what one knows. It also invites students to "commit" themselves to some ideas, thereby placing the issue in question in their personal domains of interest (Hatano & Inagaki, 1987, p. 40).

Fourth-grade children seldom have such opportunities. Peer teaching or reciprocal teaching can be used to take a small step in that direction. We feel that ISDP took a much larger step.

Designing For Learning

In Knowledge as Design, Perkins (1986) discusses in detail the instructional philosophy that supports the creation of a design environment for learning, arguing that the act of designing promotes the active and creative use of knowledge by the learner—the designer. In the designing process, Perkins says, the problem's meaning is not given by the problem itself; rather, the designer imposes his own meanings and defines his own goals before and during the process. The goals and the sub-goals may change over that period of time, and keeping track of these changes is a central interest when the design task is not for the purpose of "getting it right," but is instead aimed at producing something useful through the use of creative and critical thinking.

Schön's work (1987) is also relevant to this theme. He is interested in how different designers (e.g., architects) impose their own meaning on a given open-ended problem, and how they overcome constraints (created by themselves, or given as part of the problem they solve) and take advantage of unexpected outcomes. This interactive process requires high-levels of reflection and develops the ability to "negotiate" with situations in "as needed," and creative ways.

What is the difference between programming as such and designing a piece of instructional software? How does it relate to the "knowledge as design" framework?

A "computer program" is an independent entity consisting of a logically arranged set of programming statements, commands or instructions, that defines the operations to be performed by a computer so that it will achieve specific and desired results. We use the term "instructional software design" to refer to the building of a computer program that has a specific instructional purpose and format—much more is involved than mere programming. In this context, the lessons constructed by children were composed of many computer procedures or routines (i.e., isolated units) that were connected to each other for the purpose of teaching or explaining fractions to younger children. A unit of instructional software is a collection of programs that evolve through consideration of the interface between product and user. The instructional software must facilitate the learning of something by someone.

Designing and creating instructional software on the computer requires more than merely programming it, more than merely presenting content in static pictures or written words, more than managing technical matters. When composing lessons on the computer, the designer combines knowledge of the computer, knowledge of programming, knowledge of computer programs and routines, knowledge of the content, knowledge of communication, human interface, and instructional design. The communication between the software producers and their medium is dy*namic*. It requires constant goal-defining and redefining, planning and replanning, representing, building and rebuilding, blending, reorganizing, evaluating, modifying, and reflecting in similar senses to that described by Perkins and Schon in their work.

In terms of the programming end of it, software designers must constantly move back and forth between the whole lesson and each of its parts, between the overall piece and its subsections and individual screens (e.g., Adelson & Soloway, 1984; Atwood, Jeffries, & Polson, 1980; Jeffries, Turner, Polson, & Atwood, 1981). Because of the computer's branching capabilities, the designer has to consider the multiple routes a user might take, with the result that the nonlinear relationship between the lesson's parts can grow very complex. Moreover, the producer needs to design interactions between learner and computer: designing questions, anticipating users' responses, and providing explanations and feedback—which require sophisticated programming techniques. Finally, the child-producer who wants to design a lesson on the computer must learn about the content, become a tutor, a lesson designer, a pedagogical decision-maker, an evaluator, a graphic artist, and so on. The environment we created in ISDP encouraged and facilitated these various processes, and therefore we believe, contributed to the results.

SUMMARY AND CONCLUSIONS

This paper had a double intention: to describe ISDP, and to situate this particular project in a general theoretical framework called Constructionism. ISDP offered a realistic and comprehensive model for our constructionist vision of education in general, and for the use of computers in education in particular. It also offered a model for the kinds of research that we find insightful and beneficial to our understanding of learning and development, thinking, teaching, education, and the use of computers to facilitate these processes.

We described how the participant ISDP class, comprised of 17 fourth-grade students, integratively learned mathematics, design, and programming, etc. in the course of using LogoWriter to develop pieces of instructional software for teaching third-graders. We illustrated various aspects of our evaluation—quantitative and comparative results, as well as qualitative ones. Our evaluation showed that the ISDP students achieved greater mastery of both Logo and fractions as well as improved metacognitive skills than did either control class. The ISDP approach of using Logo programming as a tool for reformulating fractions knowledge was compared with other approaches to using Logo, in particular the traditional learning of programming per se in isolation from a content domain, and was also compared with other approaches of learning fractions. The ISDP experiment showed that simultaneously learning programming and fractions was more effective than learning them in isolation from each other.

The ISD Project recast fractions learning in essentially three ways:

(1) it emphasized more involvement with the deep structure (representations) over the surface structure of rational-number knowledge *(algorithms);*

(2) it made fractions learning instrumental to a larger intellectual and social goal, that is, having students think about and explain what they think and learn, in an interactive lesson designed for younger children; and
(3) it encouraged both personal expression and social communication of rational-number knowledge and ideas.

We emphasized the fact that ISDP had little to do with the idea that learning Logo is in itself either easy or beneficial. We asserted that in different contexts the import of the phrase "learning Logo" can differ so greatly, that the question borders on meaninglessness. Nevertheless, in the particular context of the ISD Project, where Logo was integrated into a total context, and where students programmed intensively and extensively, one can meaningfully *begin* to investigate the question of how various features of Logo contributed to the success of the children's work.

We found that Logo facilitated the ongoing *personal engagement* and gradual evolution of different kinds of knowledge; and at the same time, it also facilitated the *sharing* of that knowledge with other members of the community, which in turn encouraged the learners to continue and build upon their own and other people's ideas. In short, Logo facilitated communications about the processes and acts of cognition and learning. We do not maintain that only Logo could do this. But looking carefully at what specific features of Logo enhanced individual cognition and social learning can help guide us in future technological developments. And indeed, ISDP provided us with many insights—cognitive/developmental as well as technological—into what kinds of learning tools we want to develop for constructionist learning.

We mentioned that the ISDP should not be viewed as a "very controlled treatment" type of experiment. The pedagogical situation was quite complex, and one could formulate innumerable conjectures about the "real" source of the experimental children's learning. We concluded that

ISDP allowed us to create a total learning environment in which some impressive integrated learning took place.

It was beyond the scope of this study to single out the contribution of the individual aspects of that environment. In our view, a more complete understanding of this learning process can come through an integrative and accumulative process of experimentation and theory-building (and there are several projects of this kind within our Group at the Media Laboratory, e.g., Harel, 1990c). This article is also intended as a contribution to that process, in which we shared our conjectures and the bases on which we formulated them. We hypothesized, for example, that improvements in performance among ISDP students could have been affected by factors related to: the affective side of cognition and learning; the children's process of personal appropriation of knowledge; the children's use of LogoWriter; the children's constructivist involvement with the deep structure of fractions knowledge; the integrated-learning principle; the learning-by-teaching principle; and the power of design as a learning activity.

However, the main point we wanted to make here was that each one of those conjectures, when considered alone, would give only very partial information about the meaning of the results. By considering them together, and by speculating about their interrelations, we are endeavoring to make use of the very kind of holistic approach—to knowledge and cognition, and to the development of learning technologies—that we believe informs and characterizes Constructionism in general, and ISDP in particular.

Acknowledgments: The research reported here was conducted at Project Headlight's Model School of the Future during 1987-88 as part of Idit Harel's Ph.D. Thesis at the MIT Media Laboratory; and was supported by the IBM Corporation (Grant # OSP95952), the National Science Foundation (Grant # 851031-0195), the MacArthur Foundation (Grant # 874304), the LEGO Systems A/S, and the Apple Computer Inc. The preparation of this paper was supported by the National Science Foundation (Grant # MDR 8751190). The ideas expressed here do not necessarily reflect the positions of the supporting agencies.

We are deeply grateful to Linda Moriarty, who made an essential contribution to the Project and to the research ideas reported here. Over the years, many other teachers in Project Headlight at the Hennigan School contributed indirectly, but very importantly, to the work. We thank all the students and teachers of Headlight—without whom this project would not have been possible.

We thank Aaron Falbel and Beth Rashbaum for their editorial assistance, and other members of our Epistemology and Learning Group for their contribution in their inspiring discussions of the ideas presented in this paper. We thank Yasmin Kafai for her help in the preparations of the statistical tables.

REFERENCES

Ackermann, E. (1990). "From Decontextualized To Situated Knowledge". In I. Harel (Ed.). Constructionist Learning: A 5th Anniversary Collection Of Papers. Cambridge, MA: MIT Media Laboratory.

Adelson, B., & Soloway, E. (1984a). "A Cognitive Model Of Software Design. '''Cognition and Programming Project, (Research Report No. 342). New Haven, CT: Yale University.

Adelson, B. & Soloway, E. (1984)b. "The Role Of Domain Experience In Software Design." Cognition and Programming Project, (Research Report No. 25). New Haven, CT: Yale University.

Atwood, M.E., Jefferies, R., & Polson, P.G. (1980, March). "Studies In Plan Construction I And II." (Tech. Report No SAI-80-028-DEN). Englewood, CO: Science Applications Inc.

Behr, M.J., Lesh, R., T.R., & Silver, E.A. (1983). "Rational Number Concepts." In R. Lesh & M. Landau (Eds.), Acquisition of Mathematics Concepts and Processes. New York: Academic.

Behr, M.J., Wachsmuth, I., Post, T.R., & Lesh, R. (1984). "Order and equivalence: A clinical teaching experiment." *Journal of Research in Mathematics Education, 15 (5), 323-341.*

Brown, A.L., Bransford, J.D., Ferrara, R.A., & Campione, J.C. (1983). "Learning, Remembering, And Understanding." Handbook of Child Psychology: Cognitive Development, 8, New York: Wiley.

Brown, A.L. (1984). "Reciprocal Teaching: Comprehension-Fostering And Comprehension Monitoring Activities." *Cognition and Instruction, 1(2), 117-175.*

Brown, A.L. (1988). "Motivation To Learn And Understand: On Taking Charge Of One's Own Learning." *Cognition and Instruction, 5(4), 311-322.* Hillsdale, NJ: Erlbaum.

Brown, J.S., Collins, A., & Diguid, P. (1989). "Situated cognition and the culture of learning." *Educational Researcher, 18(1), 32-42.* AERA.

Carver, S.M. *(1987).* Transfer Of Logo Debugging Skill: Analysis, Instruction, And Assessment. Unpublished doctoral dissertation, Carnegie-Mellon University: Pittsburgh, PA.

Chipman, S.F., Segal, J.W., & Glaser, R. (1985). Thinking and Learning Skills. Vol. 1 & 2, Hillsdale, NJ: Erlbaum.

Collins, A. & Brown, J.S. (1987, April). "The new apprenticeship." Paper presented at the American Educational Research Association, Washington, DC.

Falbel, A. (1990). "The Computer As A Convivial Tool." In 1. H a re I (E d .) . Constructionist Learning: A 5th Anniversary Collection Of Papers. Cambridge, MA: MIT Media Laboratory.

Goldman Segall, R. (1989a, February). Videodisc Technology As A Conceptual Research Tool For The Study Of Human Theory Making. Unpublished manuscript. Cambridge MA: Media Laboratory, MIT.

Goldman Segall, R. (1989b, November). Learning Constellations: A Multimedia Ethnographic Description Of Children's Theories In A Logo Culture. Unpublished Ph.D. Thesis Proposal. Cambridge MA: Media Laboratory, MIT.

Harel, I. (1986, July). "Children As Software Designers. An Exploratory Study In Project Headlight." Paper presented at the LOGO '86 International Conference, Cambridge, MA: MIT.

Harel, I. (1988, June). Software Design For Learning: Children's Construction Of Meaning For Fractions And Logo Programming. Unpublished doctoral dissertation, Cambridge, MA: Media Laboratory, MIT.

Harel, I. (1989a, April). "Tools For Young Software Designers." Proceedings of the Third Workshop of Empirical Studies of Programmers (ESP Society). Austin, TX.

Harel, I. (1989b, June). "Software Design For Learning." In W.C. Ryan (Ed.). Proceedings Of The National Educational Computer Conference *(NECC).* The International Council of Computers in Education, Boston, MA.

Harel, I. (1989c, September). "Software Designing In A Learning Environment For Young Learners: Cognitive Processes And Cognitive Tools." Proceedings Of The Friend21 International Symposium On Next Generation Human Interface Technologies. Tokyo, Japan .

Harel, I. (1989d, September). "Software Design For Learning Mathematics." In C. Maher, G. Goldin, & R. Davis (Eds.), Proceedings Of The Eleventh Annual Meeting Of Psychology Of Mathematics Education. Rutgers University, New Brunswick, NJ: Center for Mathematics, Science, and Computer Education .

Harel, I. (1990a). "Children As Software Designers: A Constructionist Approach To Learning Mathematics." *Journal of Mathematical Behavior, 9(1), 3-00.*

Harel, I. (1990b). "The Silent Observer And Holistic Note-Taker: Using Video For Documenting A Research Project." In I. Harel (Ed.) Constructionist Learning: A 5th Anniversary Collection Of Papers. Cambridge, MA: MIT Media Laboratory.

Harel, I. (1990c). (Ed.). Constructionist Learning: A 5th Anniversary Collection Of Papers. Cambridge, MA: MIT Media Laboratory.

Harel, I. & Papert, S. (1990). "Instructionalist Products Vs. Constructionist Tools: The Role Of Technology-Based Multimedia In Children's Learning." Cambridge, MA: MIT Media Laboratory. (Paper in progress) .

Hatano & Inagaki (1987). "A Theory Of Motivation For Comprehension And Its Applications To Mathematics Instruction." In T.A. Romberg & D.M. Steward (Eds.), The Monitoring Of School Mathematics. Background Papers. (Vol. 2.): Implications From Psychology, Outcomes From Instruction. Madison, WI: Center for Educational Research.

Jackson, I. (1990). "Childrens' Software Design As A Collaborative Process. An Experiment With Fifth Graders At Project Headlight. In 1. Harel (Ed.). Constructionist Learning. A 5th Anniversary Collection Of Papers. Cambridge, MA: MIT Media Laboratory.

Janvier, C. (Ed.) (1987). Problems In Representation In The Teaching And Learning Of Mathematics. Hillsdale, NJ: Erlbaum.

Jefferies, R., Turner, A.A., Polson, P.G., & Atwood, M.E. (1981). "The Processes Involved In Designing Software." In J.R. Anderson (Ed.), Cognitive Skills And Their Acquisition. Hillsdale, NJ: Erlbaum.

Kafai, Y., & Harel, I. (1990). "The Instructional Software Design Project: Phase II." In 1. Harel (Ed.). Constructionist Learning: A 5th Anniversary Collection Of Papers. Cambridge, MA: MIT Media Laboratory.

Lammer, S. (1987). Programmers At Work. Redmond, WA: Microsoft Corp. Press.

Latour, B. (1987). Science In Action: How To Follow Scientists And Engineers Through Society. Cambridge MA: Harvard University Press.

Lawler, R. W. (1985). Computer Experience And Cognitive Development: A Child Learning In A Computer Culture. West Sussex, England: Ellis Horwood Limited.

Lesh, R., & Landau, M. (1983). Acquisition Of Mathematics Concepts And Processes. New York: Academic.

Minsky, M. (1986). Society Of Mind. New York: Simon and Schuster.

Motherwell, L. (1988). Gender And Style Differences In A Logo-Based Environment. Unpublished doctoral dissertation, Cambridge MA: Media Laboratory, M IT.

Papert, S. (1971a). "Teaching Children Thinking." (Al Memo No. 247, and Logo Memo No. 2). Cambridge, MA: MIT.

Papert, S. (1971 b). "Teaching Children To Be Mathematicians Vs. Teaching About Mathematics." (Al Memo No. 249 and Logo Memo No. 4). Cambridge MA: MIT.

Papert, S., Watt, D., diSessa, A., & Weir, S. (1979). "Final Report Of The Brookline Logo Projects. Part I And II." (Logo Memo No. 53). Cambridge MA: MIT.

Papert, S. (1980). Mindstorms: Children, Computers, And Powerful Ideas. New York: Basic Books.

Papert, S. (1984a). "Microworlds Transforming Education." Paper presented at the ITT Key Issues Conference, Annenberg School of Communications, University of Southern California, Los Angeles.

Papert, S. (1984b). "New Theories For New Learnings." Paper presented at the National Association for School Psychologists' Conference.

Papert, S. (1987). "Computer Criticism vs. Technocentric Thinking." *Educational Researcher, 16(1),* 22-30. AERA.

Papert, S. (1990). "An Introduction To The 5th Anniversary Collection." In Harel, I. (Ed.). Constructionist Learning: A 5th Anniversary Collection Of Papers. Cambridge, MA: MIT Media Laboratory.

Pea, R.D. & Sheingold, K. (Eds.) (1987). Mirrors of Mind: Patterns of Experience in Educational Computing. Norwood, NJ: Ablex.

Perkins, D.N. (1986). Knowledge As Design. Hillsdale, NJ: Erlbaum.

Piaget, J. (1955). The Language and Thought of the Child. New York: New American Library.

Piaget, J. & Inhelder, B. (1967). The Child's Conception of Space. New York: W.W. Norton.

Resnick, M., Ocko, S., & Papert, S. (1988). "Lego, LOGO, and Design." *Children's Environments Quarterly, 5(4),* New York: Children's Environments Research Group, The City University of New York.

Resnick, M. (1989, April). "LEGO/Logo. Learning Through and About Design." Paper presented at the American Educational Research Association. San Francisco, CA. Appeared in Harel, I. (1990). (Ed.). Constructionalist learning: A 5th anniversary collection of papers. Cambridge, MA: MIT Media Laboratory.

Sachter, J.E. (1989). Kids in Space: Exploration into Children's Cognitive Styles and Understanding of Space in 3-D Computer Graphics. Unpublished Ph.D. Thesis Proposal. Cambridge, MA: Media Laboratory, MIT. A short version of this proposal appeared in Harel, I. (1990). (Ed.). Constructionalist Learning: A 5th Anniversary Collection of Papers. Cambridge, MA: MIT Media Laboratory.

Schon, D.A., (1987). Educating the Reflective Practitioner. San Francisco: Jossey-Bass.

Scribner, S. (1984). "Practical Intelligence." In B. Rogoff & J. Lave (Eds.), Everyday Cognition: its Development in Social Context. Cambridge, MA: Harvard University Press.

Soloway, E. (1984). Why Should Kids Learn to Program? (Cognition and Programming Knowledge Research Report No. 29). New Haven, CT: Yale University.

Suchman, L. (1987). Plans and Situated Actions. The Problem of Human Machine Communication. Cambridge, MA: Cambridge University Press.

Turkle, S. (1989). The Second Self: Computer Power and the Human Spirit. New York: Simon & Schuster.

Turkle, S., & Papert, S. (1990, in press). "Epistemological Pluralism. Styles and Voices Within the Computer Culture." *Signs Journal*. Chicago, IL: The University of Chicago Press. Also appeared in Harel, I. (1990). (Ed.). Constructionalist Learning: A 5th Anniversary Collection of Papers. Cambridge, MA: MIT Media Laboratory.