# SOFTWARE DESIGN FOR LEARNING: CHILDREN'S CONSTRUCTION OF MEANING FOR FRACTIONS AND LOGO PROGRAMMING

**BY**

**IDIT RON HAREL**

B.A. in Psychology and Philosophy. Tel-Aviv University, Israel. August 1982.

Ed.M. in Interactive Technologies and Education. Harvard University, Cambridge, MA. June 1984.

C.A.S. in Human Development. Harvard University, Cambridge, MA. June 1985.

Submitted to the Media Arts and Sciences Section in partial fulfillment of the requirements for the degree of

**Doctor of Philosophy in Media Technology Arts and Sciences**
**at Massachusetts Institute of Technology**

**JUNE 1988**

Signature of the Author [illegible] **Idit Ron Harel**
Media Arts and Sciences Section

Certified By [illegible] **Seymour Papert**
Professor of Media Technology
Dissertation Supervisor

Accepted By [illegible] **Stephen Benton**
Chairman
Departmental Committee for Graduate Students

# SOFTWARE DESIGN FOR LEARNING:
**CHILDREN'S CONSTRUCTION OF MEANING FOR FRACTIONS AND LOGO PROGRAMMING**

BY
IDIT RON HAREL

Submitted to the Media Arts and Sciences Section in partial fulfillment of the requirements for the degree of
**Doctor of Philosophy in Media Technology Arts and Sciences**
**at Massachusetts Institute of Technology**
**June 1988**

## ABSTRACT

In this thesis I assess and describe a four-month long experiment called the Instructional Software Design Project. In the first two chapters of this thesis, I propose and investigate new conditions for learning fractions, Logo programming, metacognitive skills, and software design; I analyze in detail one model case of a child's seventy-hour-long problem-solving enterprise; and I describe the day-by-day learning and cognitive processes involved as this child and her peers learned fractions and Logo programming through planning, designing, programming, and evaluating their own pieces of instructional software in the view of using it to teach pupils from a lower grade. In the third chapter, I describe the Evaluation of the Project, and how the children involved in the Project became superior in their mathematical and programming knowledge and expertise to two classes who learned fractions and Logo by methods other than instructional software design.

The Case Study and the results from the Evaluation demonstrate that fourth-grade children, placed beside one another to work on a common software-design project, over a long period of time, were strongly motivated, and established sequences of learning and development that were an interesting mixture of Piagetian and Vygotskian processes, and of Papert's and Perkins' perspectives. The Project resulted in different children's establishing different stage sequences in organizing and implementing their long-term projects. Several local models of one child's software design processes were constructed. The use of Logo to design instructional software was an unusually effective way for learning Logo programming. The learning of fractions through designing instructional software on fractions involved children in creating and translating multiple representations for fractions, and enhanced their mathematical understanding of what fractions are. The children's ongoing work during the Project also seemed to enhance their metacognitive awareness, cognitive control strategies, and metaconceptual knowledge of the topics involved.

Finally, several questions for further research are discussed in this context--for example, what microgenetic phenomena could be further investigated in such an extreme learning environment, what other topics could be learned by elementary-school children through the method of instructional software design, and what design tools or computer technologies could be developed and used in this learning process.

Principal Thesis Supervisor:
**Seymour Papert,** Professor of Media Technology
Director of the Epistemology and Learning Group
Media Technology Laboratory
Massachusetts Institute of Technology

**THIS THESIS WAS SUPERVISED, REVIEWED, AND ACCEPTED BY THE FOLLOWING DOCTORAL DISSERTATION COMMITTEE:**

[illegible]

**Seymour Papert**
Professor of Media Technology
Director of the Epistemology and Learning Group
Arts and Media Technology Laboratory
Massachusetts Institute of Technology

[illegible]

**David N. Perkins**
Co-Director of Project Zero,
Associate of the Educational Technology Center
Lecturer on Education and Senior Research Associate
Harvard Graduate School of Education
Harvard University

[illegible]

**Sheldon H. White**
Professor of Psychology
Former Chairman
Harvard Psychology Department
Harvard University

To David, my best friend of thirteen years and a wonderful husband of almost nine, who has made this work possible by his great love and support to me and to our two splendid children.

To my father Yehuda, may he rest in Heaven, who I believe must be up there watching all of this happening: for being a wonderful father to me for eight years and three months, and for being with me and loving me from afar for the last 21 years. There is probably no one who would better be able to enjoy this thesis than you, because you were so involved in education, and in creative and innovative research and writing that were valuable and influential during your own time. You were the one who taught me from age zero how to be independent, productive, creative, and always in charge of my own learning, thinking, and doing. Here is one result of your wonderful genes, your education, and your strong inspiration. I wish you were here.

To my father and to my husband I dedicate this work.

## ACKNOWLEDGEMENTS

Many people have taught me throughout my lifetime, and many have helped and guided me while I worked on this thesis--from its origins to the final draft. I can only thank a few of them by name, but I would like to express my deep thanks to all the people with whom I worked and by whom I was inspired and guided over the years.

The Media Technology Laboratory at MIT has always provided me with a highly stimulating intellectual environment, with many talented people and many exciting state-of-the-art projects to learn from. So stimulating was this environment that I also had to learn how to not involve myself in many of these projects, how to stop generating ideas, and how to concentrate my attentions on one thing only--my own thesis; so rich and diverse was this environment, that it was often very difficult to find more than one person who was doing or was fully interested in the kinds of things I wanted to do. The intellectual perspective that underlies this thesis is that educators should provide learners with a stimulating and rich environment as well as with a great deal of personal and intellectual support, although learning really takes place when the learner is in charge. This intellectual perspective is probably very familiar to many of my friends and colleagues at the Media Lab, whom I want to thank for discussing my ideas and research with me, and for helping me at times when I was desperate for help and advice from a human being, rather than from a computer.

I particularly wish to thank my professor, Seymour Papert, who by accepting me into his research group at the Media Lab, and by allowing me to collaborate with him in Project Headlight, opened many new windows in my thinking, learning, and professional growth. He inspired me throughout, and still does so to this day. Many of his lectures, our own discussions, and our Group's conversations of children's cognitive and social development, education and learning, computer science, programming, and artificial intelligence were invaluable and helped shape and re-shape my thinking, so that I finally came up with this Project which I love and believe in.

I also want to thank another group of people at "that other place" up along the Charles River, which happens to prove something that only a few MIT people believe: that other creative and productive lives exist outside of MIT. There is almost nothing I could have done on my own at MIT without the skills and knowledge I gained while studying for my Masters Degree at Harvard. I came to MIT after studying at Harvard and working with Judah Schwartz and other researchers from the Educational Technology Center, from the Special Telecommunication Department of Channel 2 (with which it was affiliated at that time), and from the Human Development Department. From them I learned a great deal about conducting research, cognitive development, interactive technologies, software design, and videodisc production. I am deeply indebted to them for their help and advice, and for their allowing me to live in both worlds and maintain close working relationships with people and projects at both places.

Among the Harvard people with whom I worked, I wish to give special thanks to David Perkins and Sheldon White. David Perkins, whose work and ideas I was inspired by constantly, was the most systematic supporter and advisor during my last six years as a graduate student in Cambridge. He was there for me whether I was at Harvard or at MIT, was constantly available to discuss my research, and worked very hard at "pulling me up in my zone of proximal development" by his wonderful questions and insights. Sheldon White taught me that there are no limits to human thought...Whatever I thought and wrote was never "crisp" enough for his very demanding taste. He taught me how to be more critical and careful in my analyses or syntheses of theories and data. For this I am deeply grateful.

Seymour Papert, Sheldon White, and David Perkins invested a great deal of their time in reading and commenting on all my drafts. I never ceased to marvel at the constancy and abundance of their insightful and detailed comments. I am deeply grateful for their criticisms and reviewings and hope that they believe in my thesis as much as I believe in their theories and ideas. Sylvia Weir, Edith Ackermann, Sherry Turkle, Brian Harvey, Judy Sachter, Marlene Kliman, Ricki Goldman Segall, Maria Vignals, Mitch Resnik, Terry Tivnan, Alan Kay, Sharon Carver, Jim Kaput, Eliot Soloway, and Roy Pea contributed their ideas and commented in detail on mine while I worked on my pilot studies and on this thesis. For this I wish to express my gratitude.

I would also like to express my gratitude to Mario Bourgoin and Harry Nelson for their tireless help in Logo programming and other computer-related and technological issues throughout the Project; to Jacqueline Karaaslanian and Stephanie Hobart for providing administative support and more. I thank all the members of the Epistemology and Learning Group and those from the other Research Groups at the Media Lab., as well as all my colleagues and friends from outside of MIT who worked with me during the years and read and commented on my drafts.

I would like to thank all the teachers and pupils of Project Headlight and the principal of the Hennigan School, with my special thanks to Linda Moriarty, Joanne Ronkin, Carla Rios, and to their 51 pupils for their enthusiastic participation in my experiments. Linda Moriarty was a wonderful person to collaborate with on my long and complex Project. She was so much on top of all the theoretical and practical issues of the Instructional Software Design Project and helped me in its implementation throughout. I learned a great deal from her on what learning in the public schools of today is all about, and what a great art teaching is. I thank her pupils for sharing their ideas and thoughts with me, for allowing me to ask them all these "strange questions," to look at their private computer files and writings, and for doing exactly what they did and in the way they really wanted to do it: from them I learned a great deal about children's learning, problem-solving, cognitive processes, and about their inner worlds.

I wish to express special thanks to Eyal Salei, who was the research-assistant dream of any academician--so bright and organized, and so very precise in our intensive data-encoding processes; and to Marie-Claire Cournand, who was very helpful and creative in the ways she taught me how to write and revise, struggling through this thesis, and editing and re-editing it in such a wonderful way that really helped my ongoing thinking and writing.

My family and friends have continually provided support and encouragement. My husband David never quite understood how someone could be so involved in writing 400 pages about "one thing." Yet he was very supportive and loving, which gave me the power and will to keep up with the hard work. My children Anat and Ron constantly asked me: "How come they give you so much homework in your school, mommy?" It was very difficult to convince them that nobody was giving me this homework and that I chose to do these things because I wanted and loved to; "So if you love it so much, how come you are so pale and tired?" they asked. I thank them and my husband very much for forgiving me my terrible habits during the writing of this thesis--not spending enough time with them, and so often being "pale and tired." I also wish to thank Julie and Yossi Harel, Maryrose and Eyal Ron, and especially my sister Tali and my mother Rachel who proved that love and support could come in huge quantities and with very special qualities from overseas by long-distance phone calls, and could help a great deal. I thank my mother and all my family so much for their support and for believing in me.

Finally, I wish to express my deep gratitude to the institutions that supported my studies and research financially: the IBM Corporation (Grant # OSP95952), the National Science Foundation (Grant # 851031-0195), the McArthur Foundation (Grant # 874304), the LEGO Systems A/S, and the Apple Computer Inc.

**Idit Ron Harel**, Massachusetts Institute of Technology.

Cambridge, Massachusetts. May 13, 1988.

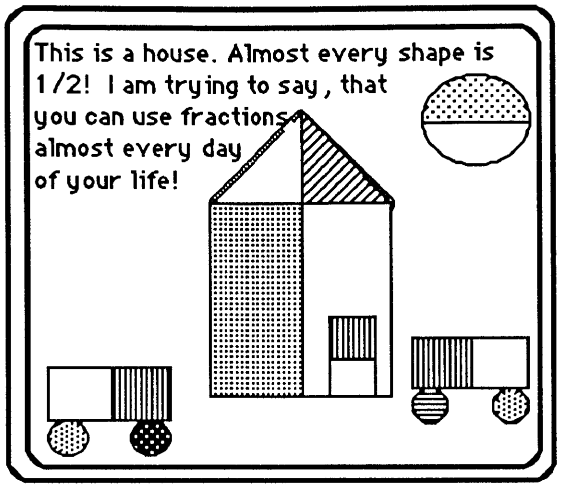
## TABLE OF CONTENTS

# OVERVIEW OF THE THESIS

*"You Can Put Fractions On Top Of Anything,*
*You Can Use Fractions Almost Every Day of Your Life!"*
**(Debbie D., Fourth-Grader. March 23, 1987)**

Consider Debbie, a plump nine-and-a-half-year-old black girl, round-faced with open mouth and large eyes, swinging her legs while sitting in front of her computer and programming. Next to her sits another girl, Naomi. She stretches towards Debbie's computer, and asks her friend 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 her forty-seven lines of code for the "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, using "regular human things" in a real-life situation, is Debbie's. In her finalized screen, she will add an instructional sentence that will appear with the pictures on the screen and say: "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. Fractions can be all around you; one can use fractions in many everyday experiences. As Debbie says, "You can use fractions almost every day of your life."


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's final "House Scene"**

Debbie has painted each half in a different color. The house half is painted in light blue, the roof's outline in red, and the roof half in orange; the sun half is yellow, the door's outline is four different colors, the door half is red, the wagon half is red, and the wagon wheels are also painted in different colors. 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 with the computer keyboards on their laps...

The "House scene" that Debbie is showing her friend is part of a long piece of software she has designed and programmed for teaching basic concepts of fractions to third graders, as part of her work in the Instructional Software Design Project. Debbie and Naomi are both sitting in front of their own computers, each involved in her own way in the same job. For the purpose of their own learning, thinking, and cognitive development, Debbie, Naomi, as well as their 15 other classmates, are each involved in creating an instructional software about fractions, working on it for one hour every day for approximately four months. In the course of this Thesis we will explore the day-by-day learning and cognitive processes involved as these fourth-grade children learn and discuss the planning, designing, programming, evaluating, and modifying of their own pieces of instructional software.

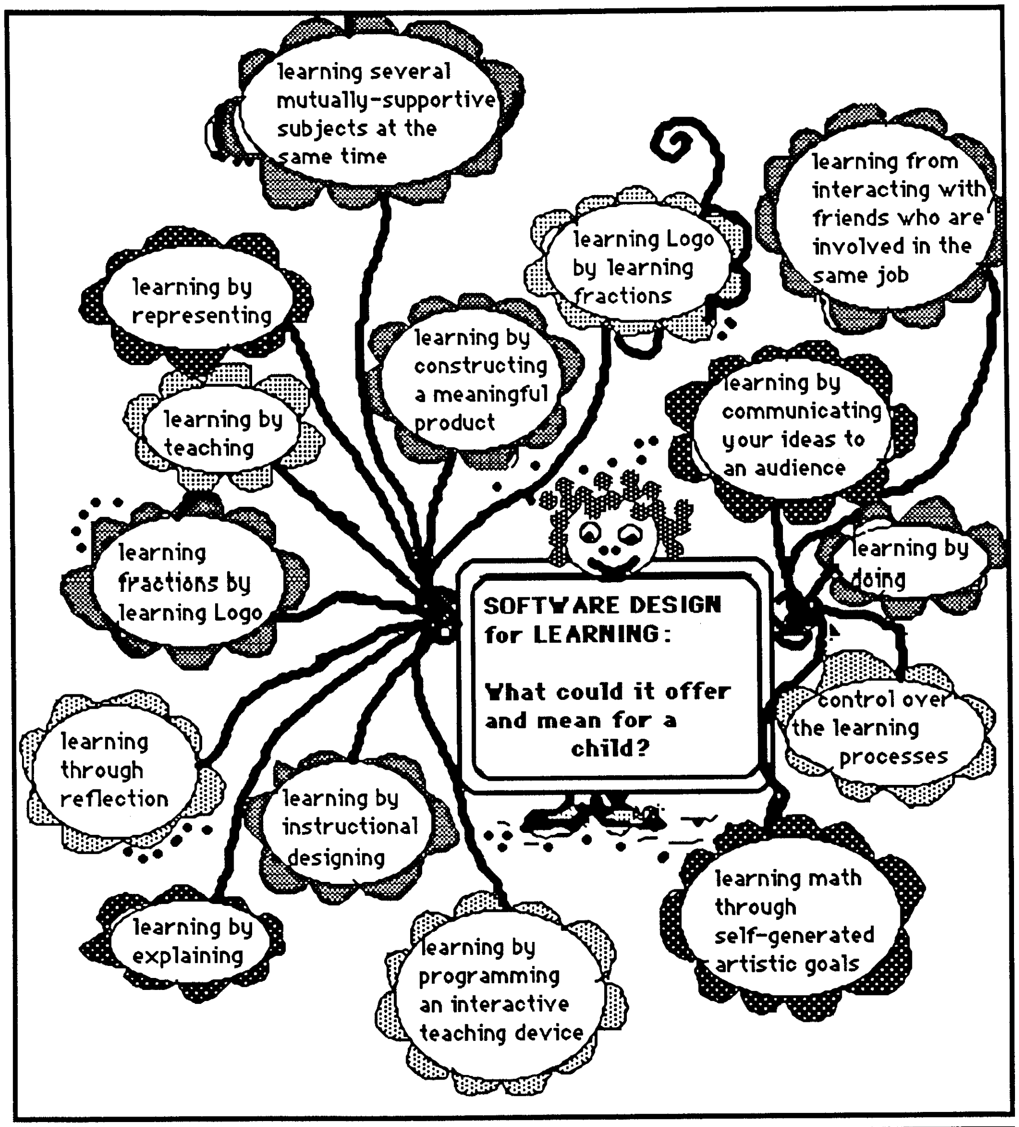
**Chapter I** of this thesis is divided into three parts. **Part 1** describes the general purposes of the study, the research site, and the conditions for the learning that took place in the course of an experiment called the Instructional Software Design Project. **Part 2** places the entire study in its intellectual landscape. It presents the rationale and motivations for conducting the research, the issues in question, the ways in which the research questions related to previous findings on the learning of fractions and Logo, and how the research method, questions, and objectives were derived from three different but related sources: 1) the experiential motivations for the study, or the way in which common professional adult learning experiences motivated me to create similar learning experiences for young children; 2) the theoretical motivations for the study, or the way in which the theory of constructionism and other learning and cognitive theories led me to create a model learning environment, and to investigate children's minds as they learned and thought in this particular environment; 3) the technical motivations for the study, or the way in which previous studies and their findings, as well as the issues raised by the existing research in the fields of Logo programming, fraction-knowledge acquisition and understanding, and software designing and programming, motivated me to conduct a study that aimed at changing the conditions for learning, with the goal of solving some of the problems reported by that research literature. **Part 3** of this chapter describes the research design and method, the rituals and activities involved in the Instructional Software Design Project, in order to provide the reader with a better sense of what it was like to participate in the Project. The Project's Evaluation procedure is described, as well as the pupils involved, their teachers, their math curriculum, the researcher's role, the instruments used, and finally, the data-collection and analysis techniques. In short, the three parts of this first "meta-chapter" are meant to create in the reader's mind a sharp mental model of the points being addressed in the study, and the ways in which the researcher went about investigating and analyzing them.

**Chapter II** of this thesis presents one Experimental girl's learning, designing, programming, and thinking processes in the Instructional Software Design Project. A detailed documentation of Debbie's day-by-day microgenetic processes over a four-month period is offered so that the reader will be able to follow Debbie's complex processes of constructing her Fractions Software step by step, her learning of fractions and Logo through these processes, her relationship with her interactive product as it developed, as well as with her peers and their products. By following Debbie, the reader will understand the reasons for some of the results that were reached, and which are presented in Chapter

III. Several examples of other children's processes and products of software design are interlaced into this chapter in order to provide the reader with a broader concept of the richness of the learning environment that was created in the Instructional Software Design Project.

**Chapter III** presents the results of the Pre- and Post-Tests and Interviews that were conducted before and after the Experiment took place with 51 fourth-graders. 17 of these made up the Experimental class (i.e., the instructional software designers), while the other 34 children were part of the two other Control classes. This chapter attempts to familiarize the reader with the results of the experiment and its influence on children's minds and educational practice in general. The reader will come to realize that the children of the Experimental class were able to master fractions and Logo with greater understanding and depth than those in the two Control classes.

Finally, **Chapter IV** of this thesis presents integrative discussions of the first three chapters. It presents the ways in which the thesis succeeded or did not succeed with respect to its major purposes and goals, as well as a number of ideas for further investigations in these areas. In the last section of this chapter, basing myself on the research findings, I shall outline my preliminary ideas for an integrative Software-Design-and-Production Tool for young children's learning.


learning several mutually-supportive subjects at the same time
learning from interacting with friends who are involved in the same job
learning Logo by learning fractions
learning by representing
learning by constructing a meaningful product
learning by teaching
learning by communicating your ideas to an audience
learning fractions by learning Logo
learning by doing
SOFTWARE DESIGN for LEARNING: What could it offer and mean for a child?
control over the learning processes
learning through reflection
learning by instructional designing
learning math through self-generated artistic goals
learning by explaining
learning by programming an interactive teaching device

The reader is invited to construct any imaginary relationships, interactions, and overlaps among these many flowers.
The work presented here aimed at generating, implementing, collecting data about, and analyzing many of those relationships It aimed at exploring what these conditions, taken together, meant for fourth-grade children. It aimed, furthermore, at exploring in what ways these conditions could change thinking, learning, and understanding of fractions and Logo programming on individual and group levels.

# SOFTWARE DESIGN FOR LEARNING

# CHAPTER I:

# THE STUDY AND ITS PURPOSE

*"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..."* (Naomi L. fourth grader. May 5, 1987).

*"It's supposed to be for littler kids, right? But [in order] 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 <u>you</u> like it ?!"* (Aaron K. fourth grader.June 12, 1987)

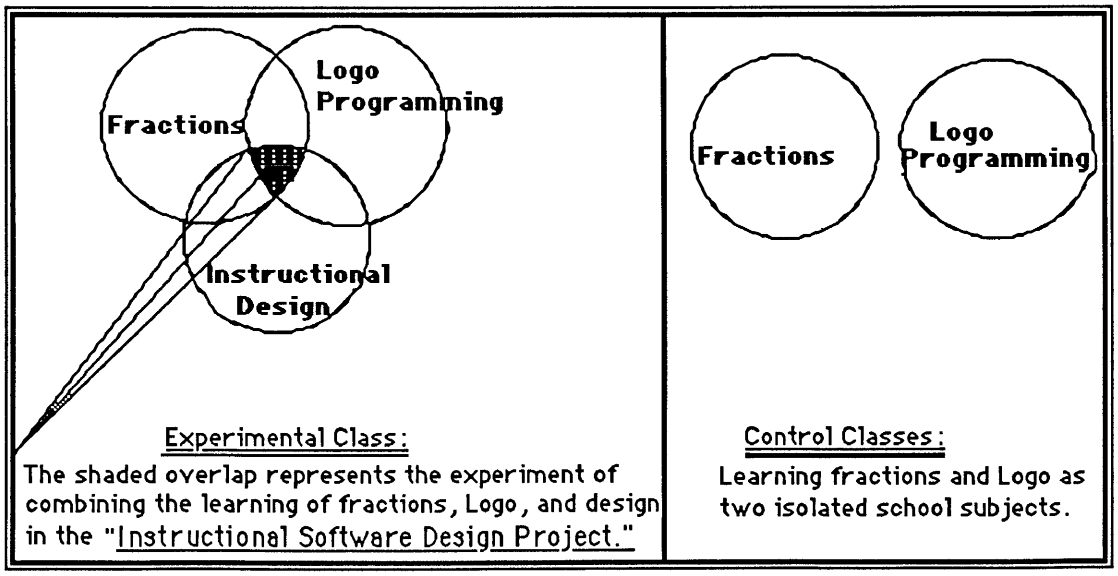
*"You have to show them fractions and explain, little by little. To program the scenes, so they will learn how to do fractions, and what they did wrong...then, someone can listen to you, to the computer, I mean, and understand."* (Debbie D. fourth grader. April 28, 1987).

*"It's hard to tell someone else that doesn't know about fractions how to do these things. So I program this for them, to help them understand it...But I have to think a <u>lot</u> 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."* (Ben S.fourth grader. May 22, 1987).

## CONTENTS OF CHAPTER I:

## PART 1: DESCRIPTION OF THE EXPERIMENT, ITS GENERAL PURPOSES, AND THE RESEARCH SITE

The aim of this thesis is to assess and describe an experiment called the Instructional Software Design Project. The Project involved fourth-grade children in the learning of fractions and Logo by their designing a piece of instructional software every day, for four months, at their own pace and using their own ideas. In this thesis I analyze the day-by-day processes these children generated, constructed, or followed, what they gained through this Project, and how these children differed in predictable ways from others who learned fractions and Logo by methods other than software design. The following diagram shows the differences between the Experimental approach that was assessed in the Instructional Software Design Project, and the regular approach of learning fractions and Logo as two isolated subjects.



Experimental Class:
The shaded overlap represents the experiment of combining the learning of fractions, Logo, and design in the "Instructional Software Design Project."

Control Classes:
Learning fractions and Logo as two isolated school subjects.

### 1.1. Overview of Research Purposes

The in-depth studying of children's minds as they learned fractions and Logo through instructional software design (the shaded overlap represented on the left side of the above diagram) had several purposes. Some of the more general goals in conducting this experiment were: **1)** To construct a <u>holistic picture</u> using both Vygotskian and Piagetian perspectives and the learning theories of Papert and Perkins, of what it meant for a child to learn and be involved in an open-ended, complex, and unusually long problem-solving process related to designing and computer programming. The study presented here did <u>not</u> attempt to focus on the kinds of well-defined problems that are usually given to subjects during one observed session; instead, this study used instructional software design as a <u>vehicle</u> for analyzing a child's total learning in a rich and complex environment, over a long period of time, analyzing an activity that involved many variables (many of which could be considered ill-defined variables), and analyzing what was <u>learned</u> through this complex open-ended solution process (but not necessarily focusing just on the solution process itself). **2)** To understand how children can learn several subject matters and skills at the same time, and how the learning of one can contribute to the other. **3)** To experiment with one prototypical activity of children who are learning through instructional software design, to assess what major changes this activity can offer to educational practice, and to judge whether or not it reveals any information that might lead to changes in the teaching and learning approaches of fractions and Logo in elementary schools. **4)** To investigate in what ways this activity encourages children's learning and understanding of the Logo programming language. **5)** To assess in what ways this activity encourages children's learning and understanding of the rational-number repersentations system. **6)** To investigate in what ways this activity encourages children's cognitive awareness (i.e., children's thinking about their own thinking), their cognitive control (i.e., planning, reflection, self-management, and thinking about these cognitive processes), and their meta-conceptual thinking (i.e., children's thinking about their own knowledge and understanding of concepts). **7)** To gather data about the processes of learning through Instructional Software Design on <u>several levels</u>: the day-by-day microgenetic process of an individual's software design sequences in creating an interactive teaching device; the process of the whole class participating in this Project; and the way in which these learning processes compared with those of other children who learned through different, more or less conventional approaches.

This set of general goals and purposes makes it clear that the present study will not be formal in the sense of using or producing precise models of mechanisms of mind. My analyses and inferences from the data are too diverse to permit the creation of functioning models of children's learning, cognitive development, or even software design. I hope to create a comprehensive model of the Project itself, to illuminate certain aspects of children's learning and cognition, of the learning tools they used in this context, of the "learning culture" that was created in the Project, as well as to make certain advances in understanding a child's cognitive-psychological front. The Project was a construction for me as well as for the children and their teacher. I created this learning environment while participating in it myself in the same way the children did, and I constructed the meanings for this environment and for the children's learning within it as I wrote this thesis--much like a detective--in an attempt to relate the different pieces of knowledge, functions, behaviors, processes, and products to one another. In other words, I went through a microgenetic process myself in implementing, analyzing, and writing this study, and did not have well-defined models for implementation or interpretation in advance. What I have produced is: 1) a large body of data that could be used as a springboard for creating such models through better understanding of children's learning and thinking processes in this rare pedagogical situation, and 2) strong evidence for supporting the success of the case under discussion, the Instructional Software Design Project, as a powerful vehicle for children's learning and cognitive development, and for the in-depth studying of children's minds as they learn, think, and produce under these "extreme" and complex conditions.

## 1.2. The Research Site: Project Headlight

An inner-city elementary public-school in one of Boston's low SES communities was the site of my study. One third of this public school, with children from first through fifth grade, of which approximately 40 percent were black, 40 percent Hispanic and 18 percent white or Asian, had been participating in Project Headlight (Papert, 1985,1986, 1987). As part of my work at Project Headlight, I implemented this Instructional Software Design Project, conducting my investigation in one fourth-grade classroom in this school, in the Project Headlight Area, during a four-month period. Earlier, during the Spring of 1986, a pilot study had been conducted with fifth graders of Project Headlight, the findings of which revealed the need for further investigations and a more systematic study of the Software Design Project. (See Harel, Children As Software Designers: An Exploratory Study in Project Headlight.1986, MIT).

The Instructional Software Design Project was the project I created within MIT's Project Headlight, and was strongly influenced by its educational philosophy, as well as an integral part of it. In the following paragraphs, I shall briefly describe Project Headlight and its educational goals.

During the 1985-86 academic year, a collaborative project involving Boston Public Schools, the MIT Media Technology Laboratory, and the IBM corporation laid the foundations for a model school of the future. Today (1988) we are in the third academic year of the running of Project Headlight, and it is now being funded by other organizations such as the National Science Foundation (NSF), the Apple Computer Inc., the Lego Company, the McArthur Foundation, and others. Project Headlight was originated and mainly inspired by Seymour Papert, Professor of Media Technology at the Massachusetts Institute of Technology in Cambridge. It was designed in anticipation of a near future in which technology would be used far more extensively than anything seen today in schools. It anticipated, for example, a system of free access to computers that might eventually involve two computers per student--one at school and one at home. In the model school, the number of computers introduced at the outset was one per three students; and it is to be hoped that this number will gradually increase as more money comes in, and as new ways are found to integrate the computers into the educational life of the school. A committee of MIT scholars and school department officials selected this specific inner-city public school as the site for Project Headlight, after inviting proposals from all of Boston's Public Elementary Schools.

Although the project uses technology extensively, it is not defined as a "technology project," but rather as an education project. It explores new approaches toward learning and teaching in the context of a school that is rich in technology. Its educational goal is open classrooms, centered around students, integrating the learning of several subjects, and using computers as tools for learning these subjects.

Project Headlight currently operates as a "school-within-a-school," comprising about one-third of the School's students and teachers. The two hundred and fifty participating students are in grades one through five, and are divided into Advanced Work Classes, Regular Classes, Bilingual Classes, and Special Education Classes. The participating teachers learn about computers and the Project's educational methodology during several three-week-long summer workshops, which, since the summer of 1985, have been repeated each year at progressively more advanced levels and according to the teachers' needs and interests. In addition, there are regular in-service meetings once every week or two weeks. Operational decisions are also made in these regular meetings, which are attended by the school's teachers and by the principal of the school, as well as by researchers from the MIT Media Lab. Among the sixteen participating teachers, five teach Bilingual Classes in Spanish, three (including two of the bilingual teachers) are Special Education teachers, four teach Advanced Work Classes in English, and five teach Regular Classes in English. The Project uses approximately 120 IBM PCjr computers in classrooms, in teachers' homes, and in the open areas next to the classrooms (see next page for Project Headlight's floor plan). Most of the computers are connected to each other in a local-area network.

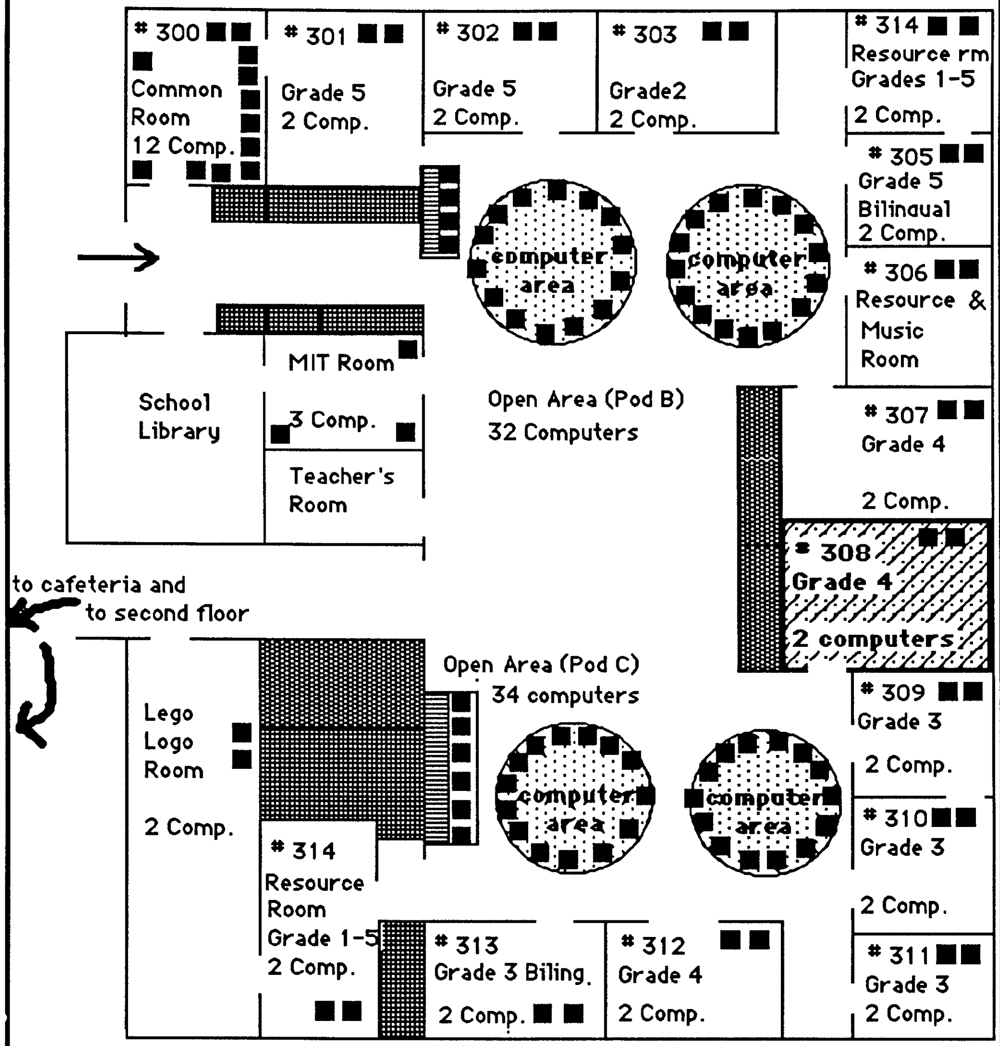
The MIT staff participates in several ways. They support and train the teachers, are responsible for the instruction of teachers and students in computer skills such as Logo, word processing, using the network, etc. They teach some special courses, conduct a number of projects within several classes (the Instructional Software Design Project was one of these), and work towards integrating the computer into the basic curriculum. Finally, MIT researchers conduct an extensive program of observations and documentation of the many projects that have been implemented at the school, and of the teachers' and students' progress.

On the whole, the goals for Project Headlight's first year were to create its technological and conceptual infra-structure, build a team relationship among the various participants, and develop projects and systematic methods of observations. The goals of the second and third years were similar, and many more projects integrating the computer into the basic curriculum were implemented by the teachers and MIT staff. However, the MIT staff concentrated far more than previously on creating among teachers and students a degree of mastery over computers and their use that might support the growth, over a certain period of time, of an educational environment rich in computers.

The diagram on the following page shows Project Headlight's floor plan and the location and number of the computers. Other first and second-grade classrooms (besides the ones indicated here) participate in the project even though they are located on other floors of the school, and are not shown in the floor plan, which represents the center of Project Headlight.

**Project Headlight: Floor Plan & no. of Computers**

(As of 1986-1987 School-Year)



▨ The Experimental Class: Instructional Software Design Project

▤ ◌ Tables for the computers in circles or along the walls.

■ ■ ■ Each black square represents one IBM PCjr computer.

▩ Bathrooms, or other non-used spaces

Total Number of Computers in Project Headlight = 110 Computers
(15 other computers are in teachers' homes).

## 1.3. Description of the Instructional Software Design Project

**Seventeen fourth-grade children placed beside one another worked on a common Project, one hour a day for four months.**

In the context of the Instructional Software Design Project, 17 fourth-grade children (all in one class) each designed, programmed (using the new LogoWriter programming language), and evaluated a collection of interactive screens (an interactive lesson) to teach third graders about basic concepts of fractions. Each child designed and programmed his piece of software for one hour each day, for approximately four months. On the whole, the children spent close to 70 hours each in this problem-solving enterprise, designing and working on implementing their software.

**The Instructional Software Design Project was open-ended but included a series of rituals and activities.**

The Project was <u>open-ended</u> but included a series of activities or routines that all the Experimental children followed. Each working day, for 5-7 minutes, the children wrote their plans and drew their designs in their personal Designer's Notebooks before going to the computer. (The Designer's Notebook is a notebook I created especially for this Project and will be described in detail later; see Appendix C for a sample.) Then, after completing their writing and drawing of plans and designs, the children worked at their own computers for approximately 45 to 55 minutes. At the computer, the children implemented their plans and designs, created new ones, and revised old ones. When they wished, they were allowed to work with their friends, help each other, or walk around to see what other children were doing. When the computer time was over, each child saved his (or her) daily files on a special diskette and went back to the classroom. In his Designer's Notebook, he then wrote and reflected on the problems and changes he had made that day, and sometimes added plans and designs for the next. The children had full freedom in choosing which concepts they wanted to teach, how to design their screens, what the sequence of their lesson should be, and what testing to include, if at all. In short, the Project was open-ended in terms of what the children 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 the time limitations in using the computer, was to allow the Project to fit into the schedule of the class and of the school. This requirement made it possible to estimate and draw generalizations about what children could accomplish in a project of this kind, designed to fit easily into the regular scheduling of any class or school in the future.

**Each child used his personal Designer's Notebook for designing and drawing screens, story-boarding, planning, and reflecting on changes and problems.**

The first requirement, the writing in the Designer's Notebook, was very important for the Project and for children's cognitive development in general. We did not tell the children what to write, how much to write, what or how to plan or draw, or how to reflect or make changes. However, we explained to them that, like professional software designers, they would enjoy keeping track of their ideas and changes, since this could help their implementation, their concentration, their not losing good ideas from one day to the next, and so on. If either of these two limitations became a problem for these children, it was the time requirement at the computer. The children always seemed to be frustrated about having to stop work and leave their computers, and often requested more time. In addition, the writing in the Designer's Notebook was a problem at first, since the children were not accustomed to handling in writing such a routine of planning, note-taking, and reflecting.

Let us consider for a moment the rituals and skills involved in children's using the Designer's Notebook in the process of designing and programming their software. This routine was essential to the Project, and its importance was clear to the teacher as well as to me. The rationale behind teaching children to plan, reflect, and take notes had been documented in several theories and rigorous experiments reported in educational psychology literature; more specifically, in the literature on learning metacognitive, cognitive control, and other related thinking skills (Nickerson et al., 1985; Chipman et al., vol 1 & 2, 1985; Brown A. L. et al., 1983). In none of these experiments, however, were the children required to plan and reflect in writing every day for as long as a period of time as they were in this Project (which went on every day for approximately four months). It was therefore expected that, in this context, new insights would be revealed about children's development and abilities in acquiring these executive processes and cognitive control skills.

Writing in their Designer Notebooks during the course of the Instructional Software Design Project was clearly a process that the children had to "get used to." After approximately ten days into the Project, the children realized its importance and made it their own. The length and structure of this Project, the fact that the children were in charge of their own learning throughout the Project, and the complete integration of these cognitive skills into other kinds of learning, were the crucial factors in encouraging these skills among children (rather than implementing a direct or explicit instruction of them). The Designer's Notebook became a personal and important tool for the children, and made them use these skills extensively and grow aware of the benefits of keeping track of their own planning, note-taking, and changes. Moreover, they realized that they did not need to implement what they wrote unless they wished to; they realized that the Notebook facilitated their thinking of "new" ideas while still implementing "old" ones; that going back and forth in their Notebooks to "old" drawings and notes was beneficial to them and very useful in their programming processes.

**Several "Focus Sessions" about software design, Logo programming, and fractions representing were conducted in the classroom during the Project.**

Several short classroom discussions (10 to 15 minutes each) were conducted during the period of the Instructional Software Design Project. In the first one, I briefly introduced, and discussed with the children, the concept of instructional design and educational software. Together, we defined the meaning and purpose of educational or instructional software, and briefly discussed a few pieces of software that the children were familiar with. I showed the children my own designs, plans, flowcharts, and screens from various projects that I had worked on in the past. I also passed among the children the book *Programmers At Work* (1987) and asked them to look at the notes, pieces of programs, and the designs of "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 children also received their personal diskettes and their Designer's Notebooks, and we discussed the ways in which they should and could be used during the Project.

During the other Focus Sessions we concentrated on issues such as the difficulties of specific fraction concepts and the children's ideas 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 OR REPRESENTATIONS COULD BE DESIGNED TO EXPLAIN THESE DIFFICULT CONCEPTS?" We asked the children to generate ideas for both posters simultaneously. After long lists of ideas were created, the children could copy all of the listed ideas, or parts of them, into their Designer's Notebook; or, if they wished, they could use these ideas as inspiration for generating new ones. Other discussions focused on specific Logo programming skills. For example, in some of these short "focused sessions" about programming, the teacher, the researcher, or one of the children, could stand next to one of the computers that were in the classroom, in front of the whole class or a group of children, and explain about how to use REPEAT, IFELSE, variables, etc. Again, the children could take notes on such concepts and programming routines in their Designer's Notebooks, or go directly to their computers and write a procedure that included that new programming skill or concept.

**The teacher and the researcher collaborated and actively participated in all the children's software design and programming sessions during the Project.**

The teacher and I collaborated, and were present during <u>all</u> the Software Design Project working hours. We walked around the children, sat next to them, looked at their programs, helped them when needed, and discussed with them their designs, programming, and problems in a friendly and informal way. In general, we had no specific plans for the Project's sequence, or for our presentations and focus discussions; rather, they were initiated by the teacher or by me at times when they were relevant to the children's work or problems, or at the children's request. (More information about the Project's learning atmosphere, objectives, and procedure, will be described later in this thesis. See also Harel's Pilot Study, "Children As Software Designers." MIT, July, 1986).

**Seventeen personal software-design-portfolios were created.**

Finally, these daily activities resulted in seventeen products (i.e., seventeen different pieces of instrictional software about fractions), and seventeen personal portfolios--one for each Experimental child--consisting of their daily sets of written plans, designs, the pieces of Logo code they programmed, and their written reflections on problems and changes for each working day.

## PART 2: THE RESEARCH RATIONALE, THEORETICAL BACKGROUND, AND EDUCATIONAL PHILOSOPHY

Three motivating factors came into play in the rationale for conducting this Instructional Software Design research. First, some of my own learning experiences, as well as those of certain professionals, incited me to create similar professional-like learning experiences for young children. Secondly, Piaget's theory about the characteristics of cognitive development, Papert's constructivist learning theory, Vygotsky's social vision of cognitive development, and Perkins' theory of knowledge and its construction through design, led me to create a model learning environment, and to investigate (by using these and other related theories) the children's minds as they learned and thought in this particular environment. Finally, The research literature and previous findings in the fields of Logo, fractions, and software design incited me to conduct a study that was aimed at changing the conditions of these subjects' learning, as well as at changing researchers methods of studying children as they learned, thereby attempting to solve some of the problems reported by that research literature. These three motivators are briefly described in the following sections, **2.1.**, **2.2.**, and **2.3.** When considered together, they form the rationale for implementing the Project and for conducting research on children's cognition, design processes, problem-solving, and learning fractions and Logo through software design.

### 2.1. The Experiential Motivators for the Study

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, within our special domains, in order to better understand the reasons, purposes, and processes of learning and teaching in our fields. 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 learning and professional development. I believe, however, that a self-experiential rationale, and all the ordinary personal experiences we, or our friends and colleagues, have gone through, offer strong heuristic reasons for the things we do--especially in education, since we are dealing with our own and other people's personal and professional education throughout our lifetime.

My learning experience as an elementary and high-school student, my professional work, and my graduate studies revealed to me two major things. The first was that I learn most effectively from teaching or explaining something to another person. The concepts, articles, or papers I remember and understand best are the ones I had to present and explain to others in a seminar, rather than the ones that required my completing worksheets or taking conventional tests. In the summer of 1984, for example, I took a three-month-long intensive course in Harvard's Computer Science Department, "Computer Science and Its Applications Using Pascal Programming." It was almost impossible to fully master all the programming concepts and applications presented in that "crash" course, although I did spend at least 6 hours a day programming in addition to the lectures and exercises, which were given by excellent instructors. However, it was only after I was asked to give a two-month-long introductory course to <u>other</u> graduate students at the Harvard Graduate School of Education that I realized what I <u>did</u> understand about programming in Pascal, what I did know, and especially what I <u>did not</u> understand. It was through teaching and explaining the material to others, preparing handouts, drawing diagrams, creating exercises and tests, and answering the students' questions that I really came to understand the material myself. When I was the teacher--rather than the learner--I spent a great deal of time thinking why something was difficult and how I could clearly explain it to others, or what might be a good exercise to master such and such a programming concept, and so on. I realized that the hands-on experience I had as a tutee, together with the kind of thinking I did as a tutor or "explainer," helped me to acquire a deeper understanding of the material.

The second thing I experienced as a learner is in some ways related to the first: it was my experience as one of the <u>instructional designers</u> for "Seeing The Unseen," a science videodisc for middle-school students--my work as part of the Harvard Technology Group (ETC, 1986). My group was involved in designing interactive lessons about basic scientific concepts and processes. I realized that the amount of understanding and knowledge I had gained on the scientific concepts and skills involved in that videodisc was probably greater than that of any future user of that videodisc system. My struggling with the raw materials, my discussions with the group, my thoughts about different ways of presenting a concept, the selections of video-clips and representations, the production of instructional sentences and feedbacks, my decisions about the sequence of screens, the design of graphic displays, and so on--forced me and my group to learn, understand, and become involved in all aspects of the content and skills presented in the videodisc.

In brief, whenever I was engaged in a learning activity that combined 1) my action upon the materials, the manipulation of it, and the building with it, and in addition, 2) communication, constructing an argument, and representing, teaching, or explaining my ideas about a given concept to others--I was also able to assess, confirm and broaden my own knowledge and understanding of these ideas. What made a learning experience most valuable to me was the combination of "learning through doing" with "learning through communicating," as well as the working independently on ideas I generated, and sharing my ideas (and products) with others as we all worked towards accomplishing a challenging, meaningful, and complex task over a long period of time.

For the time being, I will leave aside such issues as the learner's motivation, intentionality, or interest in a specific content or specific process. Let us instead consider for a moment, and in the same way, the experiences of certain professional people in their everyday work or professional training. For example, teachers have occasionally told me that they had "finally understood something today for the first time" when a student asked for an explanation of something he did not understand. Some of my friends (professional computer programmers) at MIT have told me 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 or topic of their research while they are actually working on it; through their processes of teaching and discussing their work with their students, they also identify and revise their own ideas and theories.

We can also find similar examples in professions that require even more precise skills and expertise. All Israeli fighter pilots, for instance, are required to serve for two years as instructors in the Air Force Academy as part of their intensive training. They come to the Academy as instructors only one year after they have finished their own training in the Academy, because the Air Force believes that these pilots gain new perspectives and understanding about flying by flying with, establishing relationships with, and teaching less experienced cadets how to fly. In fact, the pilots/instructors are re-evaluated while actively working at the Air Force Academy, and in many cases, through their teaching, they show greater ability than they did as cadets in the Academy. In such cases, after gaining and demonstrating their new expertise as pilots during their period as instructors, they are later asked to fly a more complex jet (switching, for instance, from a helicopter to a Phantom, or from the latter to an F-16), and to be part of a higher-level squadron from the one they served in before coming to teach at the Academy.

To take other examples from different fields, TV producers have told me that the amount of things they learned about a topic or about production, while involved in a TV production, was sometimes far greater than for their target viewers. Furthermore, it seems to me that in the educational software field, the people who are having the most fun, and are learning the most, are the software designers and programmers. Most educational software today, especially of the drill-and-practice kind, is only used once or twice by the student. The software users rarely gain deep understanding of the concepts taught, unless the software is supplemented by the quality of their teachers' instruction and explanations. But the designer, who has 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 deeper understanding of them.

To summarize, the experiences described above led me to conduct an experiment that would provide children with similar conditions for learning as those of the MIT computer programmers, Israeli fighter pilots, university professors, TV producers, or professional software designers. These people gain expertise and learn concepts and skills by actually experiencing and exercising them in long-term, professional, meaningful, and complex contexts; they acquire a deeper understanding of their knowledge and of their professions by communicating their knowledge to less experienced people; they learn about their own theories by teaching them; they learn about flying by instructing less experienced pilots how to fly; they learn about production by producing; and they learn about a topic by designing a videodisc or a piece of software for it.

## 2.2. The Theoretical Motivators for the Study

The theoretical perspectives, ideologies, and images of learners, such as they are seen in the theories of Piaget, Vygotsky, Papert, and Perkins, were strong motivating factors in my creation of a model learning environment for elementary school children (i.e., the Instructional Software Design Project), and in my construction of the image of the learner, whom I placed and studied within this complex environment. The reasons for combining these theories, the ways in which I created the Instructional Software Design learning environment, and the image of the learner whom I also attempted to study by using these theories, are briefly described in the following pages.

On the whole, the Instructional Software Project aimed at integrating the social aspects of learning (e.g., Vygotsky, 1962, 1978; Perkins, 1986), with the individualist and constructivist aspects of learning and cognitive growth (e.g., Piaget, 1976, 1955, 1972; Papert, 1980, 1986). It aimed at studying children's learning and cognitive development as a total activity, of which some aspects could be influenced by the "scaffolding" of a guiding adult, a helpful peer, or a probing researcher, while others could be a result of the child's interaction with his learning tools, his own thoughts, or his own spontaneous inventions and constructions.

For Piaget, the "truly" psychological development of the child as opposed to his school or family development, is spontaneous in nature:

> "...I will above all stress the spontaneous aspect of development, though I will limit myself to the purely intellectual and cognitive development...the development of intelligence itself--what the child learns by himself, what none can teach him and he must discover alone; and it is essentially this development which takes time...it is precisely this spontaneous development which forms the obvious and necessary condition for the school development..." [Piaget,*The Child and Reality*, 1973, pp. 2-4).

In describing the formation of the cognitive stages, Piaget places central emphasis on constructivism (1973). Although recent theories of mechanism of cognitive development emphasize different aspects of the Piagetian cognitive stages and constructivism, they generally maintain a similar developmental picture (e.g., Case, 1984, 1985; Fischer, 1984; Sternberg, 1984). Papert (1980), for example, constructs his model of the learner by using Piagetian cognitive theory, artificial intelligence theories on the mechanisms of the mind, and Turkle's (1984) personality research on the different social and affective facets involved in mathematics, computation, and science education. Above all, he emphasizes a constructivist image of the learner (Papert, 1980, 1986; Minsky, 1986). Similar to Piaget, Papert views learning as a child's construction and reconstruction of knowledge rather than as transmission of knowledge from a teacher to a child. Learning means the inventing and the formulating of concepts and rules by progressing through the Piagetian stages of cognitive development (Papert, 1980)--an active process of doing and thinking of what one does, how one does it, and what one feels about it at different times in one's intellectual development or at different stages within the problem-solving process. Papert sees learning as particularly effective when placed in the context of a rich activity, which the learner experiences while constructing a meaningful product such as a piece of art work, a story, or a research report. However, for various reasons related to his background and to his beliefs about the future of society, Papert places a great deal of emphasis on learning while constructing a computer program, or some kind of a functioning machine (Papert 1980, 1986; for a recent application of his theory of constructionist learning, see Resnick et al., Lego-Logo research, 1986, 1987).

Although Papert bases himself on Piaget's major ideas on intellectual development, on the mind's psychological structures and mental operations, and on the different stages in a child's cognitive development, he differs from Piaget when considering the "objects" in a child's life. Although his work is based on Piaget's rigorous experiments on children's cognitive development (for example, on the conservation experiments, see Papert's Principle of learning, in Minsky 1986), Papert is far more involved in the processes of <u>learning</u> than Piaget, and creates and emphasizes richer learning environments than does Piaget in his experiments. While accepting the principles of Piagetian cognitive progress (i.e., from the sensorimotor period to the preoperational period, to the concrete operational period, and to the formal operational period), Papert is mainly interested in the role of <u>self-constructed, interactive</u> objects (including computerized objects) in children's cognitive development, and in how these objects might affect the children's ways of knowing, thinking, learning, socializing, and their (Piagetian) progress in cognitive development. Papert emphasizes that Piaget's theoretical investigations have been on the mind's <u>internal events</u>, but that his own perspective, although based on Piaget, is more interventionist than his:

> "...My goal is education, not just understanding. So in my work I have placed a greater emphasis on two dimentions implicit but not elaborated in Piaget's own work: an interest in intellectual structures <u>that could develop</u> as opposed to those that actually at present do develop in the child, and the design of learning environments that are resonant with them [i.e., with the intellectual structures that could develop by the intensive, meaningful and widespread use of Logo. Therefore, Papert believes that cognitive development will eventually be quite] different in computer-rich cultures of the future. If computer and programming <u>become a part of the</u>

daily life of children, the conservation-combinatorial gap will surely close and could conceivably be reversed: Children may learn to be systematic before they learn to be quantitative" (my emphasis, Papert, *Mindstorms*, 1980, pp. 156-176).

This particular perspective of Papert's on the important role of self-constructed, interactive objects in children's cognitive development, learning, and thinking--and the idea that in many children's and adult's learning and problem-solving situations one could observe the recapitulation of the different Piagetian stages of cognitive development, or even observe new forms of Piagetian progression (or of "decalage")-- strongly influenced my thinking on the implementation and assessment of the Project.

In a slightly different way, Vygotsky (1978, 1962) also looks at the psychological processes of the individual when engaged in "activities," "tasks," or "cultural events" that are far more people-oriented, intrapersonal or communication-oriented, complex, and stimulating than in Piaget's experiments. Vygotsky places a great emphasis, far more than Piaget, on the role of language, and on the integrative relations between speech and intelligence, or between speech and cognitive development:

"...the independence of intelligent action from speech [i.e., Piaget's theoretical position] runs contrary to our own findings, which reveal the integration of speech and practical thinking in the course of development...A child's speech is as important as the role of action in attaining any goal...Speech and action are part of one and the same complex psychological function, directed toward the solution of the problem at hand...The more complex the action demanded by the situation and the less direct its solution, the greater importance played by speech in the operation as a whole. Sometimes speech becomes of such vital importance that, if not permitted to use it, young children cannot accomplish the given task...language enables children..to overcome impulsive action, to plan a solution to a problem prior to its execution, and to master their own behavior...The cognitive and communicative functions of language then become the basis of a new and superior form of activity in children..." (my emphasis, Vygotsky, *Mind In Society*, 1978, pp. 24-30).

In my Project here described, these ideas were implemented according to a slightly different method: children's were asked to think about communicating and explaining their knowledge of fractions to younger children; they were also required to use language intensively in their planning, designing, and reflection processes for their Designer's Notebooks. These language-based or communication-based tools were expected to affect children's learning and cognitive development in the present context.

Most of the studies in the Vygotskian framework (e.g., Wertsch, 1985; Rogoff & Wertsch, 1984; Brown A. L. et al., 1983; Collins & Brown J. S., 1985) focus on the role of language in learning and cognitive development, and on its role in the child's "moving" within the zone of proximal development (the region of skills that lies between the child's independent cognitions and inventions and his functioning with socio-lingual support). These researchers believe that intelligence develops as a collective activity, jointly accomplished by a child and by an adult through the use of language, and emphasize that for effective development and learning the joint socio-lingual intellectual action should occur even <u>before</u> a child functions intelligently on his own. Rogoff & Wertsch, for example, quated Vygotsky whose position was that

> "Instruction is only good when it proceeds ahead of development. It then awakens and rouses to life those functions which are in a stage of maturing, which lie in the zone of proximal development. It is in this way that instruction plays an extremely important role in development" (Rogoff & Wertsch, *Children's Learning in the 'Zone of Proximal Development'*, 1984, p.3).

Stressing this social vision of cognition and learning and the highly integrative relations between language and intelligence, Vygotskian researchers focus exclusively on the important role and processes of scaffolding (by the use of language), the role of adult mediation in children's learning, or the imitation and internalization processes in children's cognitive growth. These researchers usually do not collect data or describe, even as a possibility, the learner as someone who sometimes acts as a <u>constructive and efficient independent inventor</u> and builder of his own knowledge, <u>without</u> the help of the "scaffolder" adult. The social-vision studies mainly concentrate on describing the interesting and complex processes of adult guidance, what types of interactions between the adult and the learner result in what types of learnings, and what learning could result by imitation and internalization of culturally appropriate kinds of behavior.

The Vygotskian theorists offer us several important pedagogical ideas, but no technical sub-structure related to the mechanisms of internalization or of the mind in cognitive development. For my part, while I believe in Piaget's constructivism, in his theoretical sub-structure, and in his theory of the mechanisms of mind and cognitive development, I also feel that Vygotsky offers an important addition to the Piagetian theory. In other words, I believe that his views on the human, people-oriented, cultural and lingual aspects of cognitive development are of great importance, and could be used in conjunction with Piagetian perspectives.

To my knowledge, no problem-solving studies have so far provided us with a detailed description of a learning or cognitive process by systematically using and analyzing a <u>combination of both perspectives.</u> This is why, in the Project here described, my aim was 1) to create an environment that offered children <u>both</u> an opportunity for individualistic constructivism and for intensive use of language in a social and cultural interaction, and 2) to study and analyze all of these aspects of learning in children.

The notion of intentionality is important to the images of the learner constructed by Vygotsky and Papert. Both see the learner, rather than being a creature of experience, as one who chooses which particular experiences to "enter," or construct. They also try to capture and interpret the learner's intentions in particular learning situations. However, the particular mechanisms that generate, select, and organize what gets through the senses into the mind, and the manner in which the learner makes sense of these experiences and uses them in other learning situations, are slightly different for each theorist. Vygotsky stresses the role of language and the role of scaffolding through the use of language, while Papert, as previously stated, stresses the relationship between the learner and the interactive objects he constructs. Both Papert and Vygotsky see culture as the context for human cognitive development. Papert (unlike many of his interpreters), always urges concentrating attention on culture, rather than on technology (Papert, 1985). However, Vygotsky, much more than Papert, emphasizes the acquisition of culturally appropriate types of behavior as a process of interaction between children and adults, in which the adults guide or mediate the children's learning processes (Vygotsky, 1978, 1962; Wertch, 1985). In this sense, Papert places greater emphasis on the nature of the activity (the Logo programming culture, for example) as the mediator and cultural guide. Vygotsky describes the adult as leading the child on, ahead of his development, while Papert believes that when technological constructivist-based activities become accessible and widely used by society, they could be the major leading factors in the child's learning and cognitive development from the concrete to the abstract. Both emphasize the role of dialogue in problem-solving: the former emphasizes dialogue with a knowledgeable adult, and the latter, dialogue with the computer.

I think that both aspects of learning are important: the child's dialogue with the computer, or with the interactive objects he invents and constructs--as well as with his teacher, his peers, and with the various people, objects, knowledge, and other support systems of his culture. Bruner (1985) gives an excellent summary of these aspects of learning:

> "The world is a symbolic world in the sense that it consists of conceptually organized, rule-bound belief systems about what exists, about how to get to goals, about what is to be valued. There is no way, none, in which a human being could possibly master the world without an aid and assistance of others for, in fact, the world *is* others. The culture stores an extraordinarily rich file of concepts, techniques, and other prosthetic devices that are available...[These devices] require for their use certain fundamental skills, notable among them the ability to use language as an instrument of thought--natural language, and eventually, such artificial languages as mathematics, Polish logic, Fortran [or the Logo language],...and especially written languages...It is a matter of using whatever one has learned before to get higher ground next. What is obvious and, perhaps, "given" in this account is that there must needs be at any given stage of voyaging into the zone of proximal development a support system [internal and/or external] that helps learners get there..." (Bruner, Vygotsky: A Historical and Conceptual Perspective. In Wertsch, J. V. (Ed.), *Culture, Communication, and Cognition.* 1985, pp.32-33).

Beyond cognitive deveopment, all of the above theorists' ideas on learning and thinking are based on exceptional learning situations that correspond best to the ideal of each one. They talk about the need for a "support system" or "mediators," but use different types of mediation. They discuss epistemology, abstractions, reflections, concrete and formal knowledge, or zone of proximal development, but use different terminology and describe their processes in slightly different ways. Piagetians, for example, think that children must first construct their own knowledge, and that on their own (when "ready"), they might occasionally reach or "touch" their own zone of proximal development. Vygotskians, on the other hand, believe that children could do this most effectively with the guidance of a knowledgable adult. I believe that one should consider both these aspects in the child's process of learning and in his "moving" in his zone of proximal development, rather than choose one over the other.

In addition, these theorists place an emphasis on "conceptual tools," yet each speculates about different kinds of tools--interactive, self-constructed, technological, lingual, or cultural--depending on his own culture, beliefs, and interests. In the present study I offer a learning situation that combines and studies the child's interactions and learning through the use of a combination of these conceptual tools.

In fact, Perkins' (1986) analytical framework and his theory of knowledge and its construction suggest, to my mind, an interesting combination of the Piagetian, Papertian, and Vygotskian perspectives. His "Knowledge As Design" theory on the <u>design character</u> of knowledge, of constructing one's own knowledge, and of the giving and receiving of knowledge, strongly influenced me in the implementation and assessment of my Project.

Perkins combines epistemology and social and cognitive psychology in a theory of constructivism that is not just based on what is happening in the learner's head; rather, what is constructed must be done so socially--in an overt way--before, at the same time, or after it is constructed internally. If we consider the design character of knowledge, we realize that the building of one's own knowledge should take place internally by one's own construction and designs--as well as externally, through being scaffolded by a knowledgeable adult who also uses the design framework in his mediation and guidance of the child's learning.

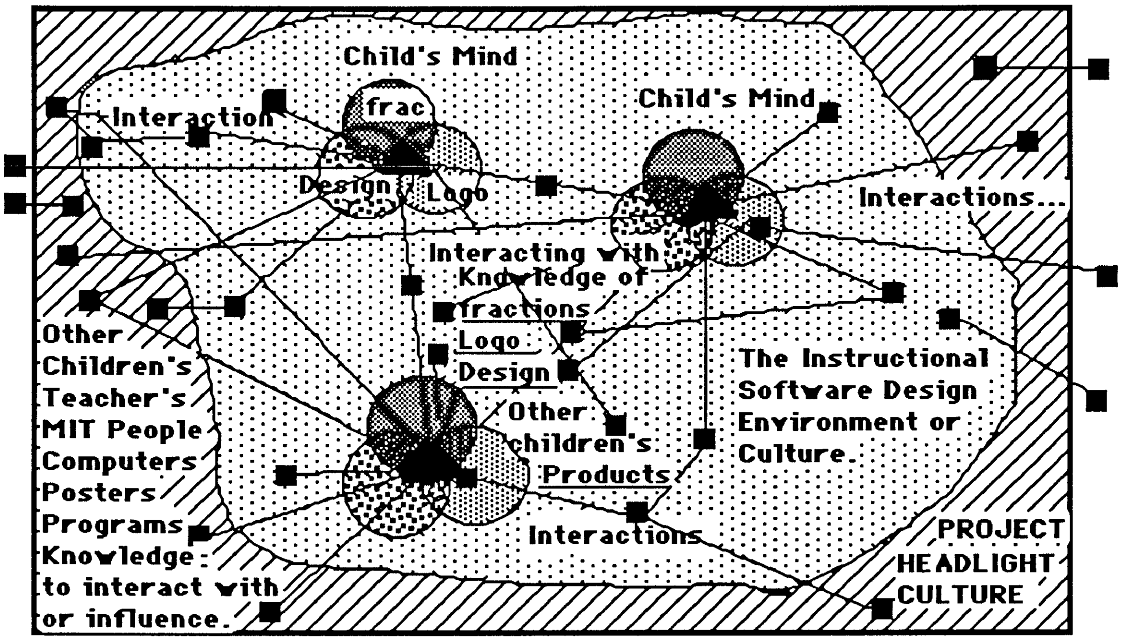
In this way, Perkins' social vision is not purely Vygotskian, therefore requires that the relationships between the adult and the learner be of a constructivist nature. The adult's role in the learning process is in allowing the child to construct his designs (i.e., his pieces of knowledge) but also in guiding and advising him in this construction by offering an appropriate design-based learning environment, or design-based intellectual tasks and discussions. Learners can attain a new level of insight when the learning highlights the constructed and constructive character of knowledge. A knowledgable adult, a helpful peer, or a particular design-tool are equally essential in this framework, in that they probe the learner and prompt in him the purpose, structures and arguments of pieces of knowledge (or designs) at times when he is not able to construct them effectively on his own. Similar to Papert, Perkins also places a great emphasis on the role of complex products in the child's cognitive development and learning. Perkins believes that the child ought to learn large, complex, and meaningful chunks of information and should not be involved in isolated exercises, or learn subject matters as isolated and non-related entities. Designing and producing complex products naturally focus the child on the inter-relations between different components of any system of knowledge.

Taking the idea of the learner's intentionality very seriously, I attempted in my Project to provide children with an environment in which they could choose to enter complex cultural learning experiences of design and imitate and use the help of an adult or peer when needed, but could also choose not to interact with society at times when they were involved and satisfied with their own constructions and inventions of designs. Allowing the children to use language intensively, and to combine and move back and forth from a social learning situation to an independent one, was a necessary thing for discovering what children could learn or sort out on their own, and how they did it--as opposed to what and how they would learn through their interactions with their culture. In other words, for my Instructional Software Design Project, with all of its related rituals and activities, I tried to integrate the ideas of a "conceptual support-system," to use Bruner's terminology, of a "dialogue" (Vygotsky's term), of "constructionism" (Papert's term), and of "knowledge designing" (Perkins's term). I experimented with specific new conditions of one type of idealized learning, based on the synthesis of the above theorists' image of the learner, and was inspired by their views on the most effective learning processes. In reality, according to Papert's more recent writings constructionism is a synthesis of these views.

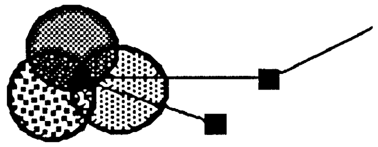
> "Constructionism is a synthesis of the constructivist theory of developmental psychology [Piaget's theory], and the opportunities offered by technology to base education for science and mathematics on activities in which students work towards the construction of an intelligible entity rather than on the acquisition of knowledge and facts without a context in which they can be immediately used and understood. A central feature of constructionism is that <u>it goes beyond what is usually called "the cognitive" to include social and affective facets of mathematics and science education</u>... Constructionism goes beyond (while) including hands-on...but the fact that [several] children are working to make something, and especially the fact that they are making something they believe in, adds extra dimensions" (my emphasis, Papert, *Contructionism: A New Opportunity for Elementary Science Education*, p. 8).

The Instructional Software Design Project aimed at allowing each child to learn concepts in fractions and Logo through his or her (concrete and abstract) <u>own construction</u>

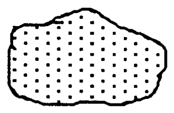
of an interactive teaching device--at the same time allowing this process of building and abstractions to be influenced by the child's interactions and conversations with the peers who were around him, doing the same job and involved in similar processes. These relations, dialogues, and interactions between one child's spontaneous inventions and thoughts and his environment or "culture"--people, objects, knowledge and thought--are depicted in the following diagram:


Child's Mind
Child's Mind
Interaction
frac
Design
Logo
Interactions...
Interacting with Knowledge of fractions
Logo
Design
Other children's Products
The Instructional Software Design Environment or Culture.
Other Children's Teacher's MIT People Computers Posters Programs Knowledge to interact with or influence.
Interactions
PROJECT HEADLIGHT CULTURE

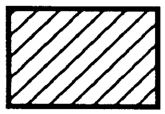
In the following paragraphs I shall briefly explain the meaning of this diagram, and what major ideas and processes it attempted to capture.



The black-shaded overlap at the center of the three circles represents the child's own constructions and creations, as well as his state of mind while he is in the process of integrating the knowledge or the learning of fractions, Logo, and design into the Instructional Software Design activity (with all of its routines, rituals, and processes). It indicates the "requirement" that each child design and implement a piece of software to teach third graders about fractions. The interactions between children's minds are shown in the diagram by straight lines ending in a square node, representing the complex relations between 17 minds involved in the same job, and their knowledge, objects, and products. In other words, these nodes stand not only for the relationship between the internal and external knowledge of each child, but for the relationship between all the Experimental children and the other people, objects, and knowledge forming their "culture."



This shaded "area" represents **the culture created within the Experimental class as a whole.** It is the environment of each child at the time he was working on his Instructional Software Design Project. It includes interactions, connections, and reciprocal relations among many of the following: other objects such as computers, computer tools, Designer's Notebooks; other people such as other children, the teacher, the researcher (myself), and members of the MIT staff (such as Harry Nelson, Mario Bourgoin, and Professor Papert)--all of whom walked around the computer-area, talked together, helped each other, expressed their feelings on various subjects and issues, or worked on different programming projects; and finally, knowledge of fractions, Logo, and design as communicated by those involved, or knowledge inherent in other people's projects (which the children could observe simply by walking around and looking at the various computer screens or the different plans and designs in the Designer's Notebooks).



This "area" represents the **culture of Project Headlight** as a whole. As mentioned before, the Instructional Software Design Project was my work within Project Headlight, and was very much inspired by Project Headlight's philosophy and routines. This area represents, among other things, the Project Headlight children and teachers from other classes who were working on other computer projects. The Experimental children interacted with them before and after each computer session, and sometimes throughout each school day. This area also represents the thoughts and actions of other teachers from Project Headlight, other people from the MIT staff, and visitors (of which there were many throughout the year).

In addition, the Project Headlight "area" and its culture (and whatever else it might represent) are surrounded and interact with other "areas" (the rest of the school and its cultural scope), and yet another (the children's homes), and another, and another--*ad infinitum.* Each child brings with him different experiences, feelings, and knowledge from the schoolyard, the school cafeteria, library, field trips, as well as experiences from home and family, and from conversations with family members and friends outside of school. A relevant example is the child's experiences with fractions from stores, from restaurants, movies, games, etc.

This thesis does not at all attempt to investigate these complex relationships between individuals and their different cultures (although it takes into account their existence in the child's life). The diagram is also very limited in terms of capturing the inter-relations between the different cultures that each child lives in and with. The diagram's main purpose is to highlight how the conditions for learning in the Instructional Software Design Project integrate Piaget's, Papert's, Vygotsky's, and Perkins' theories of learning. In other words, while the children built their knowledge of fractions, Logo, and software design within this Software Design Project, some specific interactions were created between the various kinds of knowledge, designs, people, behaviors, objects, technologies, feelings, and products.

Finally, the above diagram, as well as the present study, also attempt to describe how a child, as an instructional software designer, builds his knowledge of fractions, Logo, and instructional design by a recursive process--one might even say a recursive dance--going back and forth between at least three different modes: **1)** self-awareness of his own knowledge (i.e., meta-conceptual thinking, for example, "What do I know about fractions?" or "Can I program this thing in Logo?"); **2)** his thinking about how another child might come to know it (for example, "What screen should I design to clearly explain how 3/6 equals 1/2?" or "Should I use the same or different colors to show that all the objects on my screen are one half?"), and **3)** his thinking about what other people, such as the other children in his class or his teacher, think or do about these same concerns regarding fractions, Logo, or design (for example, "Let's see what Debbie did today," or "Gee, Naomi's game about halves and fourths is really cool, and really difficult to program" or "My teacher's explanation of this fractional algorithm was quite confusing" or "Why did Idit (the researcher) ask me that question about my screen?").

During this highly interactive and recursive process, which is described above as the Instructional Software Design learning acitivity, the child, who is at the same time the learner, designer, and computer-program evaluator, comes full circle and becomes aware of his own process of learning of a given concept, of the structure of a specific piece of knowledge, and how it relates to other forms of knowing such as a teacher's way of teaching, a friend's knowledge, another child's piece of software, or the conventional final fractions test. And in terms of his cognitive development, the child, in this way, will be able to break away from rigidity, concrete and literal thinking and become more fluent, flexible, and better able to grasp formal knowledge and create complexities.

## 2.3. The Logo programming, Fractions, and Software Design Research as Motivators for the Study

In the following three subsections I briefly review previous findings from the research on learning computer programming (in Logo), rational-number concepts, and software design. The methodology and findings of previous studies in these three fields, and the questions raised by them, make it clear that new conditions for learning were needed in these domains, as well as new kinds of investigations of children's learning (in this case, Logo programming and basic rational-number concepts). I will later argue not only that the integrative Instructional Software Design Project was a different and more meaningful way to teach children fractions, Logo, and metacognition, but also that the Project suggested a new context for studying children's processes of learning, cognitive development, in addition to assessing their mastery in these domains.

### 2.3.1. The Research on Logo Programming

The difficulties of Logo for elementary school children have been well documented (Pea & Kurland, 1984, 1987; Pea, 1984, 1987; Kurland & Pea 1983, 1987; Carver, 1986, 1987; Heller, 1986; Perkins et al. 1985; Salomon & Perkins, 1986; Clements, 1984,1985; and others). However, although programming (in Logo and Basic) has been found to contain many difficult constructs for children to master, one strong reason why it will probably remain in the curriculum of many elementary schools is that many educators and parents believe that computation is an important piece of knowledge, and that computer programming is an important skill for all children in our technological society.

Many claims have been made about what <u>knowing, learning, and understanding</u> Logo programming means to young children, and what <u>cognitive changes</u> learning to program in Logo is expected to bring. Inspired by Papert's original ideas (1980), by Feurzeig, Horwitz, and Nickerson (1981), and by Pea and Kurland (1984, 1987), I shall here summarize the different versions of their claims--but with some modifications, eliminating and adding a few points. The following is a list of ideas on what the process of programming could mean for children, and what they could learn in this process:

* The very first and most general claim about what learning to program in Logo means was provided by Papert (1980), who said that programming in Logo would first and foremost result

in reformulating knowledge; in rethinking and relearning "old" domains in new and different ways so that these "new" ways would actually become new domains of knowledge (e.g., new ways for understanding and learning geometry). This claim is extraordinarily important. We must realize that very early on in its history, Logo was proposed as a learning tool for learning other things, as a functional tool for acquiring and manipulating other kinds of knowledge, as an "educational philosophy," or as a unique "learning experience," rather than as a set of concepts and skills to be mastered in isolation, for their own sake.

* On that same general level, Papert also claimed that once Logo was extensively and widely used in society, as well as integrated into many school subjects (e.g., like word processors today), the particular learning experiences inherent in Logo would bring major changes in learners' attitudes towards learning in general, in their self-images as learners, in their self-esteem, and in their course of cognitive development.

* Through Logo programming, children could experience and learn various kinds of thinking; among them are creative, innovative, and constructive thinking, as well as rigorous thinking and precise expression through recognizing the need to make their innovative and intuitive assumptions and thoughts precise and explicit, since computers can only function within precise algorithms and routines.

* Programming could enhance children's self-consciousness and literacy with respect to processes of solving complex problems in general.

* Through programming, children could experience and learn the art of heuristics, which are intuitive or explicit approaches to problems that can be useful for solving problems in various domains (e.g., planning, finding a related problem, exploring alternative solutions, or solving the problem by decomposing it into parts).

* In their process of programming, children could experience and learn the general idea that the debugging of errors is a constructive activity and an integral part of the learning.

* Through programming, children could learn that simple and small procedures can be invented to be used, for instance, as building blocks, for gradually constructing solutions to large problems.

* In their process of programming, children could learn that there is rarely a single "best" way to do something, but different ways that have comparative costs and benefits with respect to specific goals.

* On a much more specific level, knowing how to program means an understanding of basic concepts such as procedures, variables, functions, conditionals, and transformations, all of which are used extensively in programming.

* As in the relationships between knowing reading and writing, knowing how to program means knowing how to generate code as well as how to read, understand, and interpret one's own or someone else's code.

* As in the relationship between free writing and writing according to well-defined constraints and guidelines, knowing how to program means both generating code in an open-ended free way and according to pre-defined constraints and requirements.

These are the major claims and ideas in the field about programming, its learning and understanding, and its effects on thinking and development in general. A review of the major research questions regarding children, programming, and cognitive development, reveals that questions--such as what difficulties children encounter while actually learning how to program, how to debug computer programs, how to modularize programs, or how to use inputs, variables, conditionals, recursion, or lists, or what difficulties they encounter in transferring this knowledge into other domains and tasks (e.g., Carver,1987; and Pea and Kurland,1984)--have been very much emphasized in the field during the last decade. Much less emphasized is the idea of Logo as a tool for reformulating knowledge, for manipulating and relearning other kinds of knowledge, or for changing children's self-images as learners and thinkers and their attitudes towards school learning in general (e.g., Turkle, 1984; Weir, 1986). My review of the literature has revealed to me the need to clarify and re-examine several issues related to children's learning and understanding of Logo programming.

Many researchers over the last decade stated that young children find it difficult to learn Logo in the first place, or to pursue a richer route into programming and other metacognitive and meta-learning skills of various kinds (i.e., other more general thinking skills such as planning, note-taking, explaining, representing, or reflecting). Researchers in the field, basing themselves on these very limited findings and many other educational practice constraints, and strongly inspired by studies on the effects of human-interactionism, explicit instruction, and the importance of the adult-mediated scaffolding processes in the child's learning--have sought "better" instructional techniques and more sophisticated teaching methods for Logo programming *per se*, hoping that the development of "better Logo Courses" would result in the learning of programming and its transfer (e.g., Carver, 1987; Perkins et al. 1985, 1987; or several articles in Pea and Sheingold, Eds., 1987).

To my mind, some of the reasons for many of the pessimistic findings in the research on children's learning and understanding of Logo programming, or on children's programming and their cognitive development, were partially and possibly related to several limitations in the studies themselves, and not necessarily to the children's cognition, learning, and problem-solving abilities with Logo. Piaget, for example (whose cognitive theories strongly influenced the creation of Logo and its educational philosophy) argues that a child's knowledge of something results from his own progressive constructions in wide and meaningful contexts, and that each time one prematurely teaches a child something which he could have discovered and constructed for himself, the child is prevented from inventing it and therefore from understanding it completely. With this quite radical and rather "Logoish" perspective in mind, I here suggest that one limitation of the previous studies might be the fact that the children did not program intensively or extensively, and were not given the chance to explore and experience Logo in a wide variety of contexts over long periods of time. Other related limitations of previous studies might be their involvement with programming for the sake of programming, their not providing children with meaningful contexts and tasks for programming, and their failure to integrate the programming process into the learning of other subjects.

Several attempts have been made to conduct more holistic and constructivist studies about young children and Logo (e.g., Papert et al., 1979, 1986, 1987; Lowler, 1985; Turkle, 1984; Weir, 1986). These attempts were the ones that mainly influenced me in my study. However, to my knowledge, Logo programming has only rarely been studied systematically as a child's total and meaningful learning activity; instead, it has too often been studied as Logo for the sake of Logo, rather than integrated into a larger context of the child's learning and analyzed as such.

In my Project I also wanted to assess various more specific questions regarding fourth-grade children's learning and understanding of basic concepts of Logo programming, such as procedures or modularity, variables or inputs, repeat or recursion, conditionals, and other Logo commands and operations. I also aimed at investigating the children's ability to understand processes in programming such as generating codes, debugging codes, and manipulating and modifying codes, written by themselves or by other people. However, I was not interested in investigating the learning of Logo for the sake of learning Logo, as an isolated piece of knowledge; neither was I interested in a microcausal type of investigation of what method of explicit instruction would result in the transfer of specific Logo skills into other specific domains (e.g., Pea & Kurland's study of transfer of planning skills, 1984; or Carver's study of transfer of debugging skills, 1987; or Perkins et al, Meta-Course, 1987). Instead, I used Logo as a tool for reformulating and manipulating other kinds of knowledge (e.g., instructional design, software production, or rational-number representations, ); I assessed the learning of Logo as one aspect in the child's whole learning situation; and investigated whether or not children who learned programming by designing and programming instructional software for others on fractions would become better programmers than those who, meanwhile, were learning programming in more limited situations (such as the ones reported by the literature).

It is therefore clear that one aim of my study was to investigate whether or not a high level of expertise in Logo could be a result of using Logo for learning other things, in the processes of designing and constructing long and complicated instructional programs over a long period of time. The placing of Logo programming into a broader and particularly meaningful context, which might encourage children to follow up through constantly designing and re-designing, planning and re-planning, reflecting, fitting sub-parts into a whole, fixing, evaluating, and modifying--was the condition under which children were expected to acquire sophisticated Logo programming skills, as well as other kinds of content knowledge.

An interesting note might be added here about the benefits of programming software in general. In one of the first Logo studies, Pea & Kurland (1984) found that the level of Logo learning was low among the subjects of their experiment (after approximately 30 hours of programming, once a week, in a computer lab), and that "the promised" cognitive and metacognitive effects of Logo programming on other kinds of planning tasks were insignificant, or did not exist at all, for various reasons listed in that paper. However, in the course of their studying and differentiating levels of programming skill development, Pea and Kurland observed an interesting phenomenon that is relevant to the study presented here. It has to do with the learning and understanding of programming concepts by <u>people who develop and program software</u>:

> "It is at this level of programming sophistication [software development] that we would expect to see most extensive evidence for cognitive transfer. The student can distance himself or herself from the low-level coding aspects of program generation, to reflect on the phases and processes of problem solving involved. The issue of programming which the student is concerned with at this level -- issues of elegance, optimalization, efficiency, verification, and style -- begin to transcend low-level concerns with program execution, and may lead him or her [the software developer] to consider wider issues. [The programmer]...has to take the audience fully into account, a skill that has wide applicability in many other domains such as writing" (Pea & Kurland, *On the Cognitive Effects of Learning Computer Programming,* 1984, 137-168; republished, 1987, 147-176).

To summarize, the general premise of the present study is that most of the experimental and technical literature of Logo over the last decade described how **1)** many programming skills were cognitively problematic for young children, and that **2)** the impact or effect of programming on other cognitive skills (this process, in the literature, is called transfer), had yielded a few positive and many negative findings. However, my intentions in this study were quite different from those of the researchers in the other studies: I wanted to use Logo in a very broad and complex context, where the programming activity was integrated into a larger context of software design and development (in the domain of fractions), and where extensive practice, meaningful objectives, and a greater amount of time "on task" were available in the process of learning to program. Only then could I assess whether this approach would result in a great number of positive findings on children's ability to <u>learn</u> the Logo programming language, achieve <u>cognitive gains</u> of various kinds, and better <u>understand other content knowledge</u> which was integrated into their learning of Logo.

### 2.3.2. The Research on Fractions

The difficulties of fractions for elementary school children have been well documented (Carpenter et al., 1976; Behr et al, 1983; Peck & Jencks, 1981; Post, 1981; Post et al., 1985; Kaput, 1987; Janvier, 1987; Tierney, 1987). However, although fractions and their representations are difficult constructs, they have been and will remain an important subject in elementary school curricula. Fractions are, in fact, the first number system that is introduced to children almost exclusively in school (i.e., children experience the whole-number systems much more intensively outside of school). Because fractions are so difficult for so many pupils (partly because the whole-number system is so very dominant in children's life and conflicts with the rational-number system), they figure prominently in the curriculum each year from the second grade on; still, several National assessments of children's mathematical achievements have found that children's learning and performance with fraction-ordering and computation were low and done with little understanding (Tierney, 1987). A short review of research questions regarding children's understanding of fractions, the major misconceptions they may have, the typical difficulties they encounter, as well as the researchers' analyses and suggestions for ways to overcome these conceptual and operational difficulties, can be found in the work done by the

Educational Technology Center (ETC) at Harvard in1985, in Lesh and Landau's *Acquisition of Mathematics Concepts and Processes* (1983), and in Janvier's *Problems of Representation in the Teaching and Learning of Mathematics* (1987).

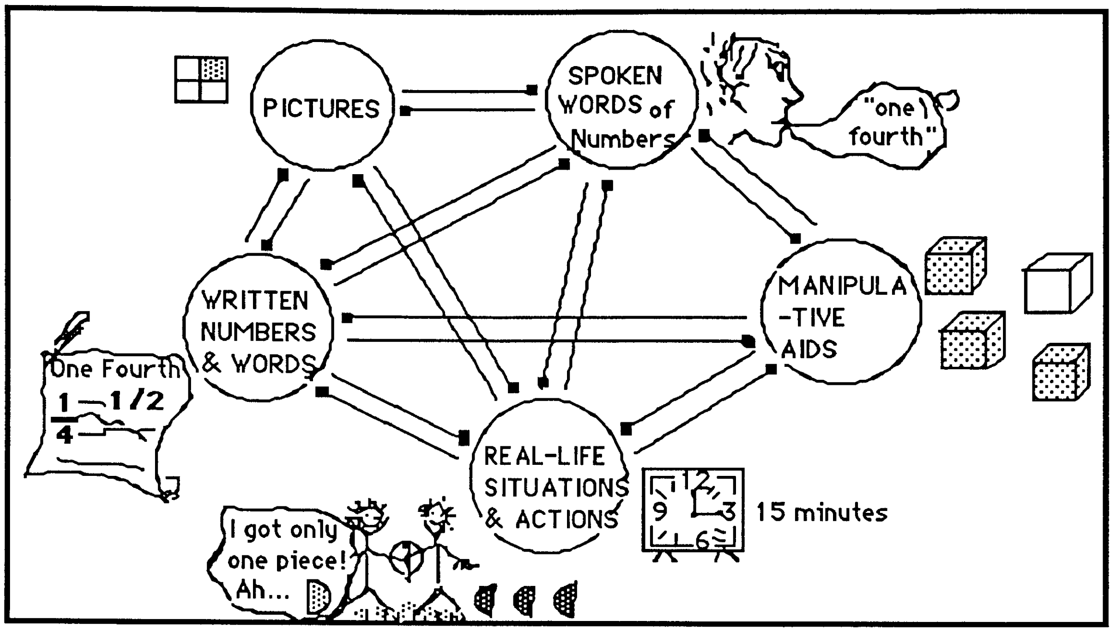
Fractions are ideal tools for learning about number systems and representational systems in mathematics. In the domain of fractions and the difficulties they present for young children, one of the goals of my Project was to investigate children's learning and understanding the system of representations of fractions, and their ability to translate different representations of fractions. These systems of representations will be described in the following paragraphs; but first, I shall briefly describe the rational-number system and what it might mean for a child to know it.

The rational-number system is one of the most sophisticated systems familiar to elementary and middle-school children. There are many rational-number **sub-constructs.** They can be perceived as islands of rational-number knowledge in the child's mind. Examples of these sub-constructs are: ratio, number-line, part-whole, operators, rate, decimals, and percentages. According to the theory of Lesh, Landau, and Hamilton (1983), and Behr, Lesh, Post and Silver (1983), a child's rational-number conceptual model contains: **1)** Within sub-construct networks (or within-concept networks): when the child considers the rational sub-construct of percentages, for example, he can compare percentage values, add or multiply them, etc. This means seeing each rational-number sub-construct as a network that includes many components, interrelations, and operations between these components. **2)** Between sub-construct systems (or between-concept systems): when the child considers the sub-constructs of percentages and decimals, for example, he can compare their sizes and manipulate, translate, or operate on these quantities. This means seeing the whole rational-number conceptual system as the super-structure that includes several sub-constructs and the interrelations between these sub-constructs. In other words, the between-concept systems have three components: **a)** different rational-number sub-constructs (e.g., part-whole fractions, ratios, decimals, etc.) and their networks; **b)** links between those networks (i.e., the "sameness" or "distinctness" of rational-number sub-constructs); and **c)** operations, which, among other things, make it possible to transform a given rational number into different forms.

According to this theory, the relationship between the two systems (between-concepts and within-concepts) is reciprocal: the between-concept systems derive some of their meaning from within-concept networks, and vice versa. For example, the translation of a simple fraction (1/2) into a percentage (50%) gives the fraction a meaning that includes the idea of proportion (one-half is like fifty to a hundred).

The relationship a child perceives when considering a rational-number concept is simplified and restricted by the characteristics of that particular concept's topology. For example, when considering a number-line, the child will tend to notice and use properties such as "betweenness" or "distance" in a subset of a set of numbers; whereas a child using a "ratio" interpretation will not use the properties of betweenness, and will think in this context of rational numbers as extensions of whole numbers. In other words, when the child "visits" a particular rational-number "island," he will be sensitive to the properties and characteristics of that island. Several of these islands (or within-sub-construct networks) are considered to be quite intuitive, and some children might be more familiar with, and think more easily according to the "style of" one conceptual network (e.g., part-whole), while at the same time they may be less familiar with another, or feel uncomfortable thinking according the style of another (e.g., decimals). However, for most children, the between-sub-construct systems associated with rational numbers are "poorly organized and unevenly formalized" (ibid., p.269). In other words, the whole rational-number structure and its parts derive some meaning from each other. The role of the representational system is crucial in this relationship, since these networks and systems are manifested in different kinds of representations and translations among these representational systems. For example, a child can represent 40% in a picture, or in numbers; he can also represent an operation such as 4%*0.5=? by symbols, pictures, or beads.

When we say that a child understands the mathematical system of rational-number concepts (sub-constructs), we partly mean that we expect the child to be able to move with flexibility from one "island" to another, connect and differenciate between each island's characteristics and properties, or express the same fractional ideas using several representations: sections of circles or rectangles, words, money, food, or time. These kinds of connections are facilitated by understanding the rational-number representations system. A possible model for the translation processes between rational-number representations is depicted in the following diagram (I have reconstructed this diagram basing myself on the models of Lesh et al., 1983; and Behr et al., 1983):



**An interactive model of translations between some of the more general modes of rational-number representations**

A major focus in previous assessments of children's development of rational-number concepts has been the role of manipulative aids such as Pattern Blocks (e.g., Harrison, 1972), Fraction Bars (e.g., Bennett, 1981), and Cuisenaire Rods (e.g., Davidson, 1977), etc. In these experiments, such materials were used to facilitate the acquisition of rational-number concepts, as the child's understanding moved from the concrete to the abstract (Davidson, 1977). However, the psychological analysis done by Lesh et al. (1983) showed that manipulatives were just <u>one way</u> of presentation in the large representational system, and that the other modes of representation (the symbolic, written, spoken, or real-life situations that are presented in the above diagram) also played a very important role in different children's styles of thinking, and in their acquisition and use of these concepts. Different materials and activities were found to be useful for making models of different situations, and no single manipulative aid was found to be the "best" for all children, for all rational-number situations, or for translating fractional representations (Behr et al., 1983).

Lesh et al. (1983) tested their subjects on different "translation modes" among representations of fractions. In their large-scale testing program (as part of their "RN PROJECT") they found that some translations were more difficult for children to process than others (they are listed below from the <u>easiest</u> for children, type number 1, to the most <u>difficult</u>, type number 9):

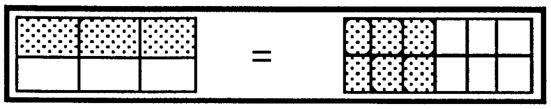
**Level 1.** Translating word representation into word representation (for example, three-sixths equal six-twelfths: this was the easiest translation for the children in the RN Project conducted by Behr et al. 1983, and Lesh et al., 1983).

**Level 2.** Translating symbols into symbols (for example, 3/6 = 6/12).

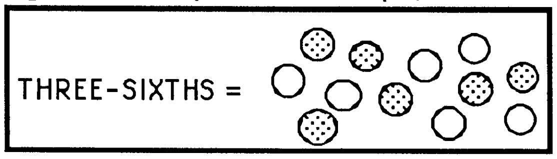
**Level 3.** Translating symbols into words (for example, 3/6 equal six twelfths).

**Level 4.** Translating words into symbols (for example, three-sixths = 6/12).

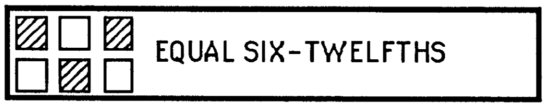
**Level 5.** Translating a picture into a picture, for example,



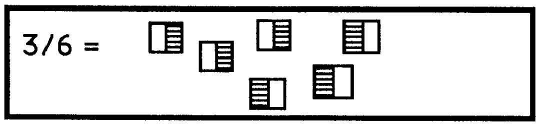
**Level 6.** Translating words into a picture, for example,



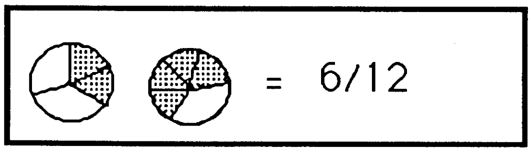
**Level 7.** Translating pictures into words, for example,



**Level 8.** Translating symbols into pictures, for example,



**Level 9.** Translating pictures into symbols (the hardest for children of ages 9-14),



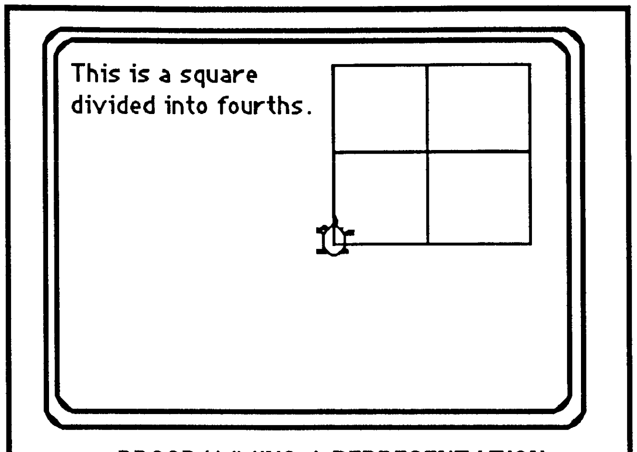
The Instructional Software Design Project consisted in children's working on all the levels presented in the above diagrams. It consisted in children's working <u>between</u> these representational systems, <u>combining</u> the different islands of rational-number knowledge (i.e., connecting or translating between two different rational-number sub-constructs such as 50% and1/2), and <u>creating</u> and <u>translating</u> several representational modes (i.e., designing a screen that combined both graphical and written representations for the same fraction, 1/3). Contrary to previous projects, it no longer attempted to identify the "best" manipulative aid for the fractions-learning process. It emphasized, however, that while working with the computer, children could decide, by themselves, which representations they wanted to create and manipulate. Since any one of the above representations (in the two diagrams above) could be programmed, children could choose which ones they felt comfortable with, and which ones they wanted to learn, work with, or combine.

A child, for example, can program in Logo a simple picture of a circle region divided into fourths, and, using different flashing colors, shade in two of these fourths (which will blink on and off) in order to show a representation of two-fourths. The child can add to this picture the written words "two-fourths," which is, in fact, the translation mode of Level 7 above (i.e., translating pictures into written words). One can add to this picture another one--a large round clock with an animation of the clock's big hand moving slowly from the number12 to the number 3--and write, "this is one-fourth of an hour," then move the hand from number 3 to the number 6 and write, "this is another fourth of an hour"; or write on the screen, "one-fourth of an hour is 15 minutes, two-fourths are 30 minutes" (i.e., this is a translation of the pictorial representation of time or clocks into words, but it is also a representation of a real-life situation and its translation into fractions). A child can also program another picture showing a one-dollar bill, and under it, four quarters (coins). Two of these quarters might be highlighted in different colors, or might be animated to "walk" around the screen and "sit" beside the written words "two-fourths of one dollar." Countless other examples for fractional representations can be programmed in Logo. We can briefly imagine a variety of representations, from pizzas to gears, from musical rhythms to body movements. The list is endless, but I invite the reader to imagine a few, and can guarantee that most of them could be programmed in Logo. These simple examples are, in fact, taken from several of the Experimental children's projects; other more complex representations can be programmed as well. These examples are provided here to show the richness of the Logo environment for creating various kinds of static or animated fractional representations (or various rational-number subconstructs), as well as its challenging characteristics for young children in terms of creating connections or relationships among the different representations.

When using the computer (programming in Logo), children had the opportunity to be involved in <u>representing these representations</u>. In the process of programming in Logo, they could represent clocks, but did not need to actually manipulate real ones (at least, not as part of the experiment). They could represent chocolate bars or money within a "store scene", but did not actually work with these real-life objects or situations. Some children could choose to represent blocks or beads, but did not actually use these manipulative aids; others could simulate spoken symbols in giving instructions on the computer, and in combining pictorial representations or sound when needed. Although the Project made no use of manipulative aids or other structured processes for moving from the concrete to the abstract, the computerized Logo environment remains rich in its potentially available representations (i.e., the representations are not there, on the computer, but the child can create procedures for these representations). Furthermore, Logo lends itself to combinations of several representations at the same time and on the same screen. In other words, the Logo programming code itself becomes a new kind of representation--the writing and the execution of a piece of Logo code being a procedural representation of a pictorial representation. To give an example, a very simple picture of a square region divided into fourths, could be represented in a simple Logo procedure in the following way:

```
TO HALF.SQUARE :NUM
HOME
REPEAT 4 [FD :NUM RT 90]          [drawing a square]
FD :NUM/2 RT 90                   [going to the middle of the left side of the square]
FD :NUM                           [drawing a horizontal line in the middle]
RT 90 FD :NUM/2 RT 90 FD :NUM/2   [going to the middle on the bottom side of the square]
RT 90 FD :NUM                     [drawing a vertical line in the middle]
PR [This is a square]
PR [divided into fourths]          [representing it in words by printing an
                                      instructional statement on the screen]
HOME
END
```

While this procedure is being executed, the child is able to observe the dynamic and procedural movement of the Turtle on the screen as it follows the Logo instructions given by that child. The execution of this particular procedure will result in the following picture:



PROGRAMMING A REPRESENTATION
ON THE COMPUTER SCREEN.

A child can draw a square by hand and divide it into fourths, which will result in a similar picture. However, when the same child programs this picture in Logo, he must describe his actions exactly, step by step. In other words, saying "I am drawing a horizontal line in the middle" is different from saying "I gave instructions to the Logo Turtle to go forward :num/2." In the Logo version, the child's involvement with the specifications of the representation is deeper and much more intense. In order to take the Turtle anywhere, the child must be aware, for example, that in order to characterize this type of representation as a fraction, the parts of the whole must be equal. The "equality of the parts" idea is manifested in the child's instructions to the Turtle: "Turn RIGHT 90 degrees, and then Go FORWARD :NUM/2," which will result in the Turtle's turning right and going forward for half of the length of one of the square's sides. In the hand-drawn representation, the child can use his sense of balance in dividing the line into two pieces. However, in using Logo, the child must understand what dividing the line into two equal pieces means, or dividing a shape into halves. Therefore, the above procedure becomes a programming representation of the pictorial representation. A child can read this Logo code alone (without the printed statement or a view of the final picture) and say: "Aha! This shows four-fourths."

Another example (not in Logo) of the potential richness of a relatively different set of instructions and commands used in computer graphics can be seen in Sachter's (1987) exploratory work. In her system, children were required to manipulate 3-D objects on the screen by giving specific instructions which were in themselves representations of the 3-D objects and of their future rotated movements or scaling.

Sachter studied eleven-year-old children from Project Headlight, and among other things, investigated children's development of spatial abilities and their understanding of such concepts as object rotation and scaling while in the process of using her system, giving specific instructions to it, and observing the execution of these instructions. With regard to my previous arguments, dealing with the contribution of the writing of Logo representations for fractions to a child's understanding of the structure of certain fractional representations or of the relationships between representations, some of Sachter's findings were similar to mine: the children who were involved in direct manipulation of the 3-D objects on the computer screen, and who wrote precise and complex instructions and commands, eventually performed much better on other pencil-and-paper spatial tasks than those who did not use her system, and therefore were not able to develop accurate mental representations of the objects' rotations or scalings.

The questions in the "Rational-Number Concepts" test and in the "What Is A Fraction?" interview (Appendix B) were selected for these very reasons. It was assumed (according to the Harel's Pilot Study, July 1986) that no one in the Experimental class would be able to design and program more than a maximum of fifteen representations during the course of the Instructional Software Design Project. In the process of learning fractions in the Project, the children did not use any manipulative aids, cover a very wide range of rational-number sub-constructs, or translate or manipulate a large number of representations. However, the Post-Tests included a large number of rational-number sub-constructs, their multiple representations (in the pencil-and-paper tests), and a variety of manipulative aids (in the interview). These were in fact assessing some kind of transfer.

In summary, my hypothesis was that the Experimental children would do well on these tests, not because they actually learned from these materials, but because--through designing and programming instructional software on fractions, creating representations of fractions through Logo, and explaining and teaching about fractions and their representations--they had the opportunity to think about, reflect upon, revise and experience (although on a very small scale) the principles involved in the relationship between the different rational-number sub-constructs and their representational systems. It was clear that the literature on fractions also revealed the need to investigate the following questions: Would designing and programming representations for fractions, and creating graphical displays for equivalence and operations of rational numbers, influence children's thinking and learning about representations of fractions and the relationships between them? Would representing and explaining fractions to another person help children to overcome their own difficulties in understanding fractional representations in the ways that had been reported in the other studies? And if so, in what ways?

### 2.3.3. The Computer as a Tool for Instructional Design

This thesis also focuses on the computer as a medium for children's learning through building and explaining. There are several reasons why the computer is an outstanding medium for learning through instructional designing, as well as for investigating children's processes of designing and producing instructional software. For the sake of my discussion, I shall clarify how I differenciate between the words "software," or "instructional software," how they differ from "computer program," and what I mean when using these words in this context.

A "computer program" is an independent unit, or 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. In computer science, a program is sometimes called a "computer routine," that is, a routine written in a computer language that controls the operation of the computer.

Sometimes in computer science, "software" simply means a "computer program." By "software," computer scientists often refer to the computer languages themselves, or to larger computerized routines that control larger-scale operations of a computer. However, I shall differenciate between two major types of computer programs: 1) system programs, and 2) applications programs. The "application programs" are the ones that I will consider here as "software," or even as "software packages." A software package (or and application program) is a set of programs that can be purchased for use with a specific computer to perform various duties, such as accounting, payroll, statistical analysis, word processing, spelling, drawing, animation, etc. There are also educational software packages, or educational games and lessons written to assist human learning of various kinds, such as logical and mathematical skills, reading and writing skills, inquiry and scientific concepts and skills, or artistic skills.

In the present context, "instructional software designing and programming" means the making of an educational/instructional piece of software. In other words, it means the making of an interactive instructional system on the computer, or the building of a system that has an instructional purpose and the format of an interactive lesson. In this context, the instructional system is constructed over a long period of time, and is composed of many computer programs or routines (i.e., isolated units) that are connected to each other for the purpose of teaching or explaining something. Furthermore, unlike most computer routines or programs, instructional software is a collection of programs designed while seriously considering the human interface. The instructional software must facilitate the learning of something by someone--a real person.

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 dynamic. It is a constant planning and replanning, representing, building and rebuilding, blending, reflecting, reorganizing, evaluating and modifying. Software designers must constantly work back and forth between the whole lesson to its parts, between the overall piece and its sub-sections and individual screens (Adelson & Soloway, 1984; Atwood & Polson,1980; Jeffries et al.,1981). Because of the computer's branching capabilities, the designer has to consider multiple routes each user might take. The usual "beginning-middle-end" lesson format does not hold, and the non-linear relationship between the lesson's parts can grow very complex. Moreover, while using the computer, the producer can design interactions between the learner and the computer: designing questions, anticipating users' responses, and providing explanations and feedback. The 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 (Steinberg, 1987).

Let us consider for a moment how a child learns in a traditional classroom and how he might learn through designing and programming instructional software. The psychology of instructional software designers is different from that of learners in a regular classroom. In a regular classroom, the teacher gives a lecture, asks a question; only one person answers, while the others "sit and listen." The teacher has control over the ideas and the actions involved in the children's learning. On the other hand, instructional software design is a complex, active, and time-consuming enterprise. It requires that software designers invest a large amount of time in learning to program, create, and implement their _own_ ideas and explanations about the subject matter involved. They do not passively "sit and listen," but are _personally_ involved in their learning/teaching enterprise, and take pride in it. _They_ are the ones who make it happen.

Perkins's theory of Knowledge as Design (1986) provides another rationale for creating an environment where children learn concepts through their processes of design. Perkins discusses in detail the instructional philosophy behind creating a design environment for learning, arguing that it promotes the active and creative use of knowledge by the learner--who is also the designer. In the designing process, the problem's meaning is not given by the problem itself; rather, the designer imposes his own meaning 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 aimed at learning and developing thinking skills. Schon's work (1987) is also relevant to this framework. There too, the author is interested in how different designers (architects, for example) 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.

In the process of learning, and when educational practices are at issue, the research questions change. The rare existing literature on the processes of software design or software engineering in no way attempts to investigate what the designers _learn_ through the process of software design. For example, Nickerson's valuable work (1986) focuses, among other things, only on guidelines for "good" software design, and rests on the assumption that an expert software designer is specifying these guidelines. Simon's relevant research (1973) sketches out a theory of the psychological processes involved in reaching a solution for a design task, but only in the context of discussing the distinction between well-structured and ill-structured problems (e.g., p. 190).

Other studies are more directly related to the _processes_ involved in software design. For example, Guindon and Curtis (1988, *Control of Cognitive Processes During Software Design: What Tools Are Needed?* ) collected verbal protocols of professional software designers, and revealed "three major design-process control strategies" (p.1-5). However, the problem given to their three subjects was still in the well-defined problem framework, "a lift-control problem, that requires their designers to design the logic to move N lifts between M floors, given a set of constraints." The designers were given two hours "to produce a solution that was in a form and a level of detail that could be handed off to a competent system programmer to implement" (ibid). Although the title of this study included the words "processes of software design," this study, in fact, only revealed narrow information about experts' control over the logic design of a complex computer routine, but none on software designing and programming *per se*, or on what these experts learned through the process about the relationship between their designs and their implementation. In a similar way Adelson and Soloway (1984) studied the way expert designers solved the problem of creating a computerized mailing system. Their observations, as well as others of the same kind (e.g., Guindon, 1987; Guindon, Krasner, & Curtis, 1987; or Jefferies, Turner, Polson, & Atwood, 1981), described the very specific cognitive processes of people who were more or less expert designers and performing very logical and well-defined tasks over a short period of time. In fact, very few generalizations could be drawn from the work done by these researchers that might be strongly relevant to my study; furthermore, besides the collection of verbal protocols and the use of videotaping, few research methods and models existed that might have been directly adopted for the Instructional Software Design Project, since the problem given to my subjects was open-ended (though not ill-defined!) and was presented in the following manner: "Design a program that teaches a younger child about fractions." The children/designers' processes toward solving the problem took approximately 70 hours of programming, and within this long and complex context each child established his own route, goals, and subgoals, which could not be predicted in advance, or later analyzed by the researcher--in the sense of "good" or "bad" designs, or "expert" as compared to "novice" approaches.

There are certain drawbacks to implementing instructional software design activity in a school's curriculum. Software design is a time-consuming and complex enterprise, and it is not yet clear how it can fit into the average class schedule. Also, it might cause problems in children's learning of other subjects in the school, and, at the present time, it is not very clear which school subjects would lend themselves best to this complex process of learning. However, my goal in the Instructional Software Design Project was to experiment with one topic (fractions), and make it possible for children to learn through designing and programming instructional software much as they would do in a professional environment. I do not think children can produce "better" quality software than adults, but I believe they could be involved in the process of designing and producing instructional software--for their own learning, their own pride, their own fun. I am not suggesting that we have to stop producing educational TV programs, videodiscs, or computer software for children. What I am saying is that it might be an error to think that the instructional software design process is not suitable or appropriate for children. It is in fact the contrary. In this Thesis I shall not only examine the ways in which children can have fun designing and producing a piece of software, but analyze in detail the ways in which they can learn content and skills through it. Like adults in a professional environment as instructional designers or computer programmers, young children, through instructional software design, can also learn concepts and skills that may be difficult to master in other learning situations. In addition, learning processes of instructional software design also offer major changes in the conditions for learning among young children in the ways described by Papert (1986), as quoted below:

> "Knowledge about computation [ such as programming] and the sciences of information [such as control over one's own processing, metacognition, and constructing of information] have a special role in changing education. Such knowledge is important in its own right. It is doubly important because it has a *reflexive* quality--it facilitates other knowledge...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 means necessarily learning less of something else shows a wrong conception of the integration of these subjects into knowledge and cultures. They should be **supportive** of the other learning. It should be possible to integrate at the same time, blocks, learning of science, mathematical concepts, art, writing, and other subjects. **If two pieces of knowledge are mutually supportive it might be easier to learn both** [at the same time] **than to learn either alone**" [Papert's emphases, *Constructionism: A New Opportunity for Elementary Science Education*, 1986, p. 2).

In my Project, with its approach of learning through designing and programming instructional software, these new conditions meant that young children would learn fractions, Logo programming, instructional designing, planning, story-boarding, reflection, self-management, etc.--all at the same time and in a synergistic fashion. In the context of the Project, and through the use of the computer, I wanted to integrate different kinds of knowledge and disciplines because they were mutually supportive of one another and could contribute to each other while the child was in the process of learning them.

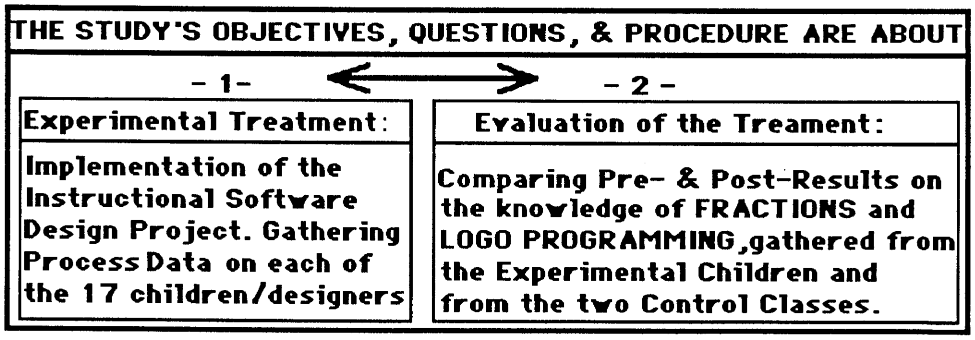
In a recent colloquium (on March 30, 1988) at the Media Technology Laboratory at MIT, William Mitchell, Professor of Architecture at the Harvard Graduate School of Design and Principal of the Computer Aided Design Group in Los Angeles quoted the novelist E.M. Forster, who once asked: "<u>How do I know what I think until I see what I say?</u>" Mitchell emphasized this quote since "it expresses very well the relation of an artist or a designer to his or her medium." He also said that "representations do not merely record, they give shape to thoughts and mediate the profits of speculation and exploration...Different representational media, according to their structures, do this in different ways. Thus when a medium is introduced into a traditional thinking or design process, that process is likely to change." In his work, Mitchell explores the ways in which the use of computer-aided-design (CAD) technology changes the traditional architectural design processes. In a very similar spirit, I explore in this thesis the ways in which the computer and Logo programming, and their use for designing fractions software, change children's processes of learning, thinking, representing, programming, as well as their understanding of fractions.

In Part 3 of this chapter which is following, I shall present the research design, describe the selected pupils and the reasons for selecting them, and outline the different objectives and questions that guided me **1)** in the implementation of the Instructional Software Design Project, **2)** in the investigation of the cognitive processes of the individuals who participated in the experiment, and **3)** in the investigation of the learning of the Experimental class as a whole. I shall also describe the research procedure and its instruments, as well as the data analysis techniques, and how this data will be presented in Chapter II and Chapter III.

## PART 3:
## RESEARCH DESIGN AND METHOD

This study was designed to systematically focus on: **1)** The learning, cognitive processes, and experiences of each child who designed, programmed, and evaluated a piece of software about fractions during a four-month period. **2)** The ways in which the software design process contributed to the Experimental children's learning of fractions and Logo; in other words, comparing the Experimental children's learning with the learning of the children in two Control classes.

The sections that follow are organized accordingly. In section **3.1.,** entitled "The Experimental Treatment," I shall describe my objectives and procedure for the implementation of the Project in the Experimental class, and the investigation of each child's cognitive and learning processes during the Instructional Software Design Project. In section **3.2.**, entitled "The Evaluation of the Treatment," I shall describe my objectives and procedure for the Evaluation of the Project: comparing, Pre and Post, the Experimental class with the two Control classes. The general concerns of this study as a whole are represented in the following diagram:



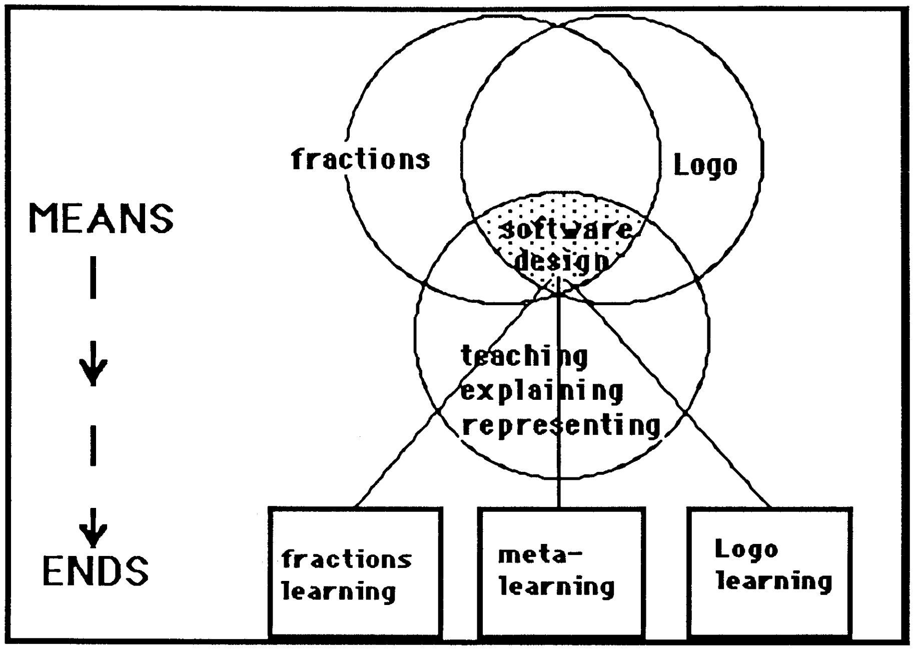
THE STUDY'S OBJECTIVES, QUESTIONS, & PROCEDURE ARE ABOUT

- 1- ⟷ - 2 -

| Experimental Treatment: | Evaluation of the Treament: |
|---|---|
| Implementation of the Instructional Software Design Project. Gathering Process Data on each of the 17 children/designers | Comparing Pre- & Post-Results on the knowledge of FRACTIONS and LOGO PROGRAMMING, gathered from the Experimental Children and from the two Control Classes. |

### 3.1. The Experimental Treatment:
### The Instructional Software Design Project

As stated previously, the experiment called Instructional Software Design Project aimed at interlacing children's learning of basic rational-number concepts and Logo programming in a software design activity. It was assumed that through this integrative approach, represented previously in the diagram in Section 2.2. and simplified in the shaded overlap of the diagram shown below (the means), the Experimental children would learn (the ends): basic concepts of fractions, programming concepts and skills in Logo, and thinking skills such as self-management, reflection, planning, revising, and representing (which come together under the term "meta-learning" in the diagram).



For this purpose, I selected a fourth-grade classroom (N=17) composed of 6 black children (4 girls and 2 boys), 2 Oriental children (1 girl and 1 boy), 1 semi-Indian boy, and 8 white children (3 girls and 5 boys). This particular class was selected for various reasons. The teacher of this class had participated in the studies I had conducted in 1985-1986, when she was teaching a fifth-grade class (e.g., Harel, July 1986). She was one of the most active teachers in Project Headlight, and was excited and willing to participate again, in the Spring of 1987 with her fourth-grade pupils, in this long and complex Project. This teacher and I had enjoyed a fruitful relationship since 1985; furthermore, she believed in the main ideas embodied in the Instructional Software Design Project, and we were both interested in working together on this Project. These were major factors in the selection of this class because it was important, in this kind of project, for the teacher to fully participate, invest a great deal of time and thought into it every day, and constantly collaborate with me on all levels, including the management of this learning environment, the extensive and intensive interaction with the pupils, data collection, and many other aspects of implementation. In addition, most of the pupils in the selected class were new to the school and had never used a computer until September 1986. We thought that the Instructional Software Design Project offered an interesting situation for these pupils to learn about the computer, Logo, and fractions. The teacher was aware that "fractions and Logo are very difficult for so many children," and that "fourth-grade children do not have deep understanding of both." The software design activity, according to her, "could offer a great change" within this class's learning approach; because of her experience in similar projects the year before, she believed that "designing and programming instructional software to teach about fractions could help many of these children to overcome many learning difficulties," such as those reported by the literature on Logo and fractions (previously described in section 2.3.1. and 2.3.2. in Chapter I of this Thesis).

### 3.1.1. The Experimental Treatment's Objectives and Procedure

My objectives during the Experimental Treatment were twofold: **1.**to implement the Instructional Software Design Project within the selected fourth-grade classroom; and at the same time **2.** to assess the day-by-day cognitive processes and learning achievements that resulted from the children's designing and programming instructional software. In other words, I was involved in implementing the Project, inspiring the teacher and her pupils, being there during all the working sessions, talking to children and working with them; at the same time, I was involved in collecting data about the on-going microgenetic processes of each individual of this class during the duration of the Project.

The children collected most of the process-data for me. In order to achieve some of the goals of this study (i.e., such as the children's meta-learning of various kinds), I wanted the children themselves to keep track of their own work; and much like professional software designers, to have the chance of constantly reflecting upon the different stages of development of their products. In short, I wanted the children themselves to follow, be aware of, and appreciate their own learning and design processes in the same way and at the same time as I was following and studying their processes.

The implementation of the Instructional Software Design Project meant involving children in learning fractions and programming through the teaching of fractions via the use of Logo, and in designing and programming interactive screens for teaching about fractions in their own way for a long period of time. In other words, my goal was to implement software design for children as a long-term process of creation and revision towards a complex product, in a similar way to the working methods of professional software designers; but the emphasis was on the learning that took place during these processes.

In order to accomplish these objectives, the children were asked to work on their Projects for one hour a day, four times a week, for four months. This group of children designed instructional fractions software in addition to their regular math curriculum. Throughout the Project I was a source of inspiration, a facilitator, an intervener (question-asker), a resource, a manager of the learning environment, and sometimes a teacher (conducting small sessions with teacher and children). Eventually, part of the management and facilitating was taken over by the teacher and her students (see also Harel's Pilot Study, 1986). Several of the short (5-15 minutes) classroom presentations or discussions (called "Focus Sessions") were initiated and conducted by the researcher or by the teacher. These discussions and instruction were initiated according to the children's needs, problems, and concerns, without any strict pre-established plan.

For both educational and research purposes, I wanted to create a learning environment in which design, or instructional design, as well as programming processes, would be integrated and overt; to create an environment in which metacognitive and meta-conceptual thinking were facilitated and explicit to the greatest possible extent (i.e., very much in the spirit of Perkins, 1986, Knowledge as Design approach). Finally, besides implementing the Project, I also had in mind a set of objectives for the assessment of the cognitive and learning processes of the individuals (i.e., I wanted to gather information for the individual case-studies). I wanted to examine the processes involved in learning through software design, both on individual and classroom levels; and to create records of the design and cognitive processes that would benefit the child, the teacher, and the researcher at the same time.

I followed and observed the software designers every day while they were working on their programs. Besides these observations, protocols of the children's processes were gathered by videotaping individual children while they were working at the computers and participating in the classroom discussions. In addition, on-line, daily records of the design, programming, and learning processes were kept by the children themselves. After each working period they saved their daily working files from the computer in a special diskette, which resulted in approximately 60 on-line files per child. Every day, before and after each working session, the software designers' daily ideas, plans, story-boards, and reflections were written by them in their Designer's Notebooks. Their daily computer files, the things they wrote in their Designer's Notebooks, the teacher's comments, and the researcher's daily observations and conversations with the children constructed a "holistic" picture of each child's progress in creating his or her product.

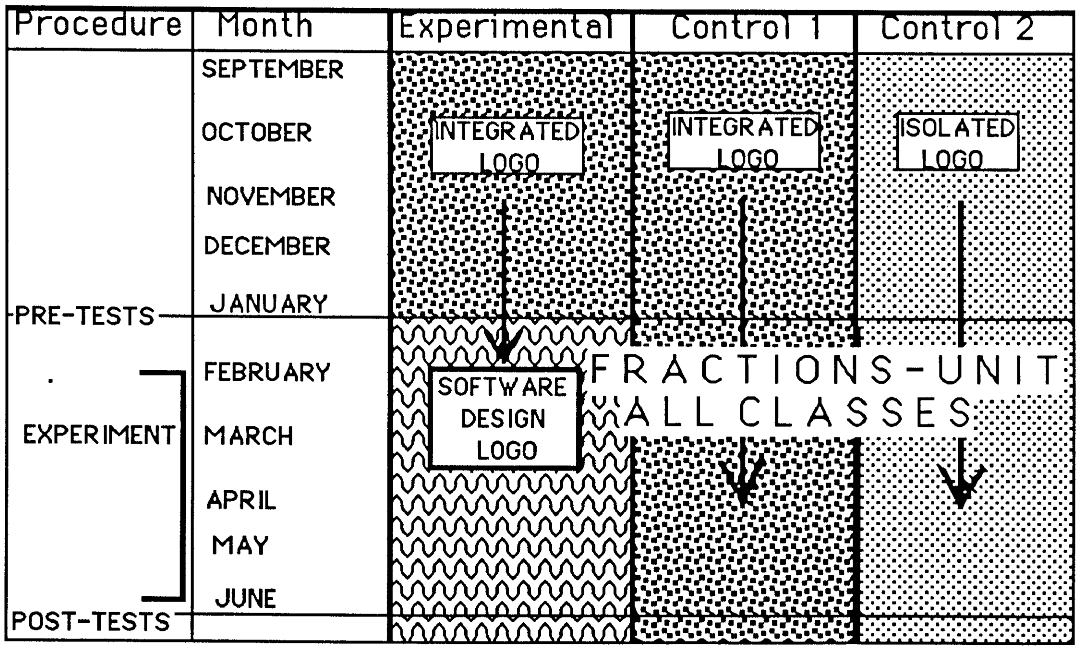
The objectives of the Designer's Notebooks were twofold. First of all, they were created for educational purposes: to encourage metacognition such as self-management, reflection, and planning, and to foster children's screen designs and reflections in writing. Secondly, the Designer's Notebooks were created for research purposes: to accumulate records on each designer's process, plans, drawings, changes, problems, etc. These notebooks included such headings as: "My Plans For Today," with design grids and writing space, "Problems I Had Today," for the children's reflections and ideas for changes; "My Script," with story-boards, "Notes," etc. (See Appendix C for sample pages from Designer's Notebooks).

### 3.2. The Evaluation of the Experimental Treatment: Comparing the Experimental Class with Two Control Classes

The general aim of the Evaluation was to find out whether software design could be considered as a process, or tool, as a means of learning concepts and skills in several domains at the same time. Within this context, instructional software design was used, among other things, for integrating the learning of fractions and Logo. The evaluation of the Experimental Treatment made it possible to go beyond the documentation of the learning and cognitive processes of the individuals in the Experimental class, and to find additional and more systematic evidence of whether or not these software designers had mastered the domains integrated into the Project.

Moreover, the Software Design Project was not taught to all the children of the Experimental class in a unified way. On the contrary, its structure was open-ended, and resulted in many different artifacts (see also Harel's Pilot Study, 1986). It was therefore very important to test the Experimental class before and after the experiment was conducted, in order to find out whether the children had mastered the specific concepts in fractions and Logo, or had only found "fun and excitement" in an unusual project. My study attempted to go even further: I wanted to demonstrate not only that the Experimental children were able to master these subjects, but show how, by designing software for teaching fractions, they might find different ways to work with problems given to them within these domains.

A comparative study was conducted with the Experimental class by means of two other classes. In order to gather information about the growth of the Experimental class as a whole, and to understand the effects of the Software Design Project on the Experimental class' learning of fractions and Logo, these three classes (N=51) were pre-tested and interviewed on their knowledge of fractions and Logo during the month of January. After this, one class (N=17), was provided with the Instructional Software Design Project (the Experimental Treatment described above), which meant intervening in the learning and teaching strategies for the fractions and Logo programming curricula. The two other classes (N=18; N=16), were not provided with the Experimental Treatment, and continued to study math and programming in their traditional way. This procedure is shown in the following diagram:



This diagram shows the different Logo learning approaches used in the three selected classes before and during the experiment (these three approaches to learning Logo will be described in the following sections). It also shows the Experimental design. The three classes were pre-tested during January; they began their "Fractions Unit" (their regular math curriculum) in March, and worked on it until the end of April; the Fractions Software Design experiment was worked on by one class only, and lasted from the end of February to the middle of June; the Post-Tests and Interviews were conducted during the last three weeks of June. Via the assessment of the Control children's knowledge of fractions and Logo, and the Pre- and Post-Tests and Interviews with the selected 51 pupils, the software designers were compared with other children who had learned fractions and Logo and (according to the school's tests) had also mastered their concepts, though not through software design.

### 3.2.1. Description of the Pupils

The three classes were selected for several reasons related to their **1.** unified grade-level and school, **2.** unified mathematics learning approaches, and **3.** various Logo learning approaches. These reasons are further described below.

**1.** Grade Level & School, Ethnicity & Gender. All the selected subjects were fourth graders from the same inner-city Public School in one of Boston's lowest SES communities. According to the City of Boston, these three classes were considered Advanced Work Classes (A.W.C). Two of these were "regular A.W.C.," and the third was a "bilingual A.W.C." These fourth-grade children had not been part of an A.W.C. in third grade, and were selected to be part of the fourth-grade A.W.C. according to the following: 1) the children's evaluations and grades given by their teachers in third grade; 2) their final grades in the City of Boston's Public Schools' examination in math, reading, social studies, etc., at the end of the third grade; 3) their parents' motivation to place them in advanced work classes, since all parents had to fill applications and attend interviews after the children's third-grade teacher had recommended that their children apply for A.W.C.

The three selected classes were almost identical in their proportion of boys and girls: the first class (Experimental) consisted of 8 girls and 9 boys; the second (Control-1), of 11 girls and 7 boys; and the third (Control-2), of 9 girls and 7 boys. In terms of their ethnicity, the Experimental class was very similar to the Control 1 class: the first (Experimental), consisted of 6 black, 2 Oriental, 1 semi-Indian, and 8 white children; the second (Control-1), consisted of 8 black, 3 Oriental, 6 white children, and 1 Hispanic child. However, the third class, the Spanish bilingual A.W.C, consisted of 16 Hispanic or semi-Hispanic boys and girls. Since the regular mathematics and programming curricula and testing were conducted in English only and regularly in all three classes, it was assumed that the fact that one of these classes was Spanish bilingual would not affect the results of this particular study. The analysis did not attempt to focus on trends related to gender and ethnicity in the results of this study. The results of the Pre-Tests indicated that all the children were on a very similar level of understanding fractions and Logo before the experiment started.

The results from the fractions Pre-Tests were also compared with those of the children in the Behr et al. National Study (1983). This comparison showed that the three selected classes were not very advanced in their knowledge of fractions; therefore the children in the present study would provide us with sufficient evidence for generalizations and conclusions about low, medium, and high-level students, and not just A.W.C pupils.

2. Mathematics Learning Approaches. All the subjects studied math every day and during the same class period. Each of the classes' teachers taught a math group: one teacher taught the low math group, another the medium group, and the third the high group (i.e., the low-level children from the three selected classrooms studied together, as did the medium and high-level children). The math classes of all levels were handled in a traditional fashion, where the teacher taught and the students completed assignments and exercises from given worksheets, and were usually tested once a week. The three teachers (all females, two white and one Hispanic) had received their teacher-training in the United States, had been teaching for several years in this Public School, and were very similar in their teaching techniques and procedures. All three were quite informal in their relationships with their pupils. They tended to be creative, and tried as much as possible to provide their pupils with small projects in the course of their teaching. They shared their curricula, updated each other on their pupils' progress and difficulties and, according to them, because of the Boston Public School Referenced Tests, they covered almost the same material in most subjects, especially in math, writing, and reading. They continued to do the same while they were teaching the Fractions-Unit, which lasted from March through April, 1987.

The three teachers tested their pupils at the beginning of the year, and also considered their pupils' math-test results of the previous year, so as to divide the pupils into math groups. The division of the pupils into particular math groups is shown in the following diagram:

**The selected classes divided into their math levels:**

| | Experimental | Control 1 | Control 2 |
|---|---|---|---|
| Low Math | N = 4 | N = 5 | N = 5 |
| Medium Math | N = 5 | N = 6 | N = 5 |
| High Math | N = 8 | N = 7 | N = 6 |
| Total in Class | N = 17 | N = 18 | N = 16 |

This division of the children into three math-levels quite consistently corresponded with the children's results in the Pre-Tests (e.g., all the low-math children scored low in the Pre-Tests, and the high-math scored high). This division will therefore be used for comparing the children's learning and understanding levels as well as their scores in the various Post-Tests.

3. Logo Learning Approaches. The three selected classes began to learn Logo programming during September of 1986 (four months before the research had begun). Three Logo learning approaches were assessed and compared for this study. For this purpose, I named them Integrated-Logo, Isolated-Logo, and Software-Design-Logo.

In the beginning, two Integrated-Logo classes and one Isolated-Logo class were selected. During the experiment, one of the Integrated-Logo classes switched into the Software-Design-Logo approach (Software-Design-Logo was only one aspect of the Treatment). The two other classes remained constant in their Logo learning approaches (as Integrated-Logo and Isolated-Logo) throughout the experiment, acting as two different Control classes. In this context, it is important to note that all three classes remained constant (between the Pre- and Post-Tests) in their regular mathematics curriculum. There were no changes of any kind in the traditional Fractions-Unit. However, the children from the Experimental class thought about fractions, designed and programmed screens about fractions, and discussed fractions with one another during the Instructional Software Design Experiment. No didactic teaching of any additional information was given during their software working sessions. It was therefore possible to investigate exactly how much the Experimental children were able to learn by their construction of an interactive teaching device about fractions. The general characteristics of each approach of learning and teaching Logo programming, the reasons for their existence and for naming them thus are described in the following paragraphs.

**Integrated-Logo:** Two Integrated-Logo classes were originally selected for this study. They were part of MIT's Project Headlight (see Appendix A). In general, children in the Integrated-Logo classes programmed from 45 to 60 minutes a day, five days a week, in an open area next to their classes. The teachers of the Integrated-Logo classes who were participating in Project Headlight had been and were being trained by an MIT staff (see Chapter I, Section 1.2). They implemented a project-oriented approach toward programming, which meant that no worksheets or exercises were given by the teachers. Instead, children worked on several Logo programming projects that were integrated into a specific curriculum, such as science or literature. It was the first year in Project Headlight for these two classes, and their first year in learning how to program or use the computer. Therefore, it was assumed that at the point of starting the experiment, their programming skills would be equal and almost as low as those of the third Isolated-Logo class (which is described below). The Integrated-Logo teachers did not grade their students in Logo, since each child's work was different from the others'. The teachers used a "soft" evaluation system for the children's project, usually relying on the child's level of involvement in his work, his investment of time and thought, or his amount of learning of new Logo skills.

**Isolated-Logo:** One Isolated-Logo class was selected for this study. Its pupils, like those in the Integrated-Logo classes, began to program in Logo for the first time in September 1986. However, they used the computer only for 30 to 45 minutes a week, in the school's computer room. The pupils in the Isolated-Logo class neither integrated their learning of programming into the curriculum, nor worked on meaningful "projects." Instead, they learned how to program by completing short programming exercises and assignments given by the computer-room coordinator. This Isolated-Logo class was learning Logo as part of a "Computer Literacy" program--the most common approach used today in elementary schools across the United States. Their teacher was not involved in the computer sessions, which were handled by the computer-room coordinator at all times. The children were graded by their teacher according to the exercises they had completed.

**Software-Design-Logo:** This Logo learning and teaching approach was one aspect of the Experiment as a whole; it was strongly inspired by, and very similar in nature to the Integrated-Logo described above. Software-Design-Logo took Integrated-Logo one step further. The major differences between Integrated-Logo and Software-Design-Logo were in the purpose, structure, meaning, characteristics, and length of the "projects."

Software-Design-Logo consisted in children's using Logo for the purpose of designing and programming instructional software. It involved the children in programming something for others, rather than just for themselves. It required that the children think about a target audience, and construct a program that would work for another, even younger, person. The Integrated-Logo projects (in Project Headlight) usually lasted from a few days to three weeks, whereas the Software-Design-Logo Project lasted four months. It was assumed that the length and complexity of the Project would be an important factor in children's learning how to reflect on, revise, modify, or maintain their programs. The children in Software-Design-Logo had their minds on other issues besides "a program that works." They worked on their Project while thinking about other people, about screen designs for teaching, about interactivity and feedbacks--much like professional software designers. In short, they were involved in a rich, meaningful, and complex task, working towards designing and programming a "real" product for "real" people.

### 3.2.2. The Evaluation Objectives and Questions

Many research questions could have been raised concerning the Experimental Treatment, since it involved so many variables, as well as a very integrative and complex pedagogical situation. However, for the purpose of this Thesis, my objectives and questions for the Experimental Treatment's Evaluation were narrowed down into 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 interlaced the Experimental children's learning of fractions and Logo with the designing and programming of instructional software. It was assumed that the Treatment probably would change the Experimental children's approaches and abilities in their learning of fractions and Logo, and that there would be a smaller change in those of the Control children's. Therefore the research questions were divided into two groups: The first group explored the Experimental children's learning and understanding of specific fractions concepts, and compared them with the Control children's learning of fractions; the second explored the Experimental children's learning and understanding of specific Logo programming skills, and compared them with the Control children's learning of Logo. These questions had been designed to provide detailed documentation on the progress made by the Experimental children in comparison with the individual Control children. In other words, they did not attempt to document the individual learning processes and cognitive development of each child from the Control classes in as much detail as for those of the Experimental class.

The questions related to the Experimental children's **knowledge of fractions** were the following: Would they be generally better at translating fractional representations? More specifically, what type of translation would they master more readily in relation to the children in the Control classes? (i.e., would the Experimental children grow more proficient than the others at translating picture representation into symbols, words into pictures, symbols into words, pictures into pictures, etc?). The theoretical framework for this last question on translations of fractional representations was mainly based on the work done by Behr, Lesh, Post, & Silver, and by Lesh, Landau, & Hamilton for their NSF "Rational Number Concepts Project" (Lesh & Landau, Eds. 1983, pp. 91-126 and pp. 263-343 retrospectively). Furthermore, would the Experimental children relate to fractions in a more personal way, be better at talking about fractions, defining a fraction, and describing fractions in real-life situations? Would they be better at representing fractions by using real-life objects such as blocks, pegs, rods, or play dough? At manipulating these objects and answering questions while using them? At basic fractional operations, such as addition, subtraction, multiplication, or division? On the whole, would the children who learned fractions in a non-conventional way, and in a rare pedagogical situation, also get higher scores on the regular public school tests? More specifically, would the children who spent a lot of time "playing" with fractions at their own pace, and constructing their own representations of fractions, be able to pass the school's or the city's regular exams?

In order to compare the Experimental children with the Control children on their **learning of Logo programming**, I attempted to answer a number of questions. These questions first of all explored the differences between the Experimental class and Control class 1, since both classes spent the same amount of time programming every day and were both part of Project Headlight. Secondly, they explored the differences between the Experimental class and Control class 2--the latter using the computer much less frequently, as part of the school's conventional Computer Literacy program. The questions were the following: Would the Experimental children use, know, and understand many programming commands and operations? Would they be better at understanding, implementing, debugging, and modifying someone else's programming code? At understanding and using the Logo REPEAT command? At understanding and using variables and inputs? Finally, would they be better at constructing codes for someone else's design or picture?

## 3.3. Data Collection and Analysis

As stated previously, my inquiry combined **1)** an educational intervention, i.e., the design and implementation of The Instructional Software Design Project; and **2)** a systematic investigation of that Project. The investigation of the Project integrated several research techniques: the creation of Individual Case Studies about the processes of software design of the Experimental children; and the Pre- and Post-Tests that compared all the Experimental software designers with children who learned fractions and Logo according to different pedagogical methods.

A large amount of data was gathered during the course of the study. In the following chapters of my thesis (Chapter II and III), the research questions will be answered in two ways: through describing in detail the step-by-step progress (the microgenetic processes) of one child from the Experimental class (Chapter II); and through comparing the Experimental class as a whole with the Control classes (Chapter III).

I will describe my data-collection and analysis techniques in the context of each of my Thesis chapters (Chapter II and III). In general, qualitative and quantitative data were used for the Case Study, and for comparing the Experimental children with the Control children.

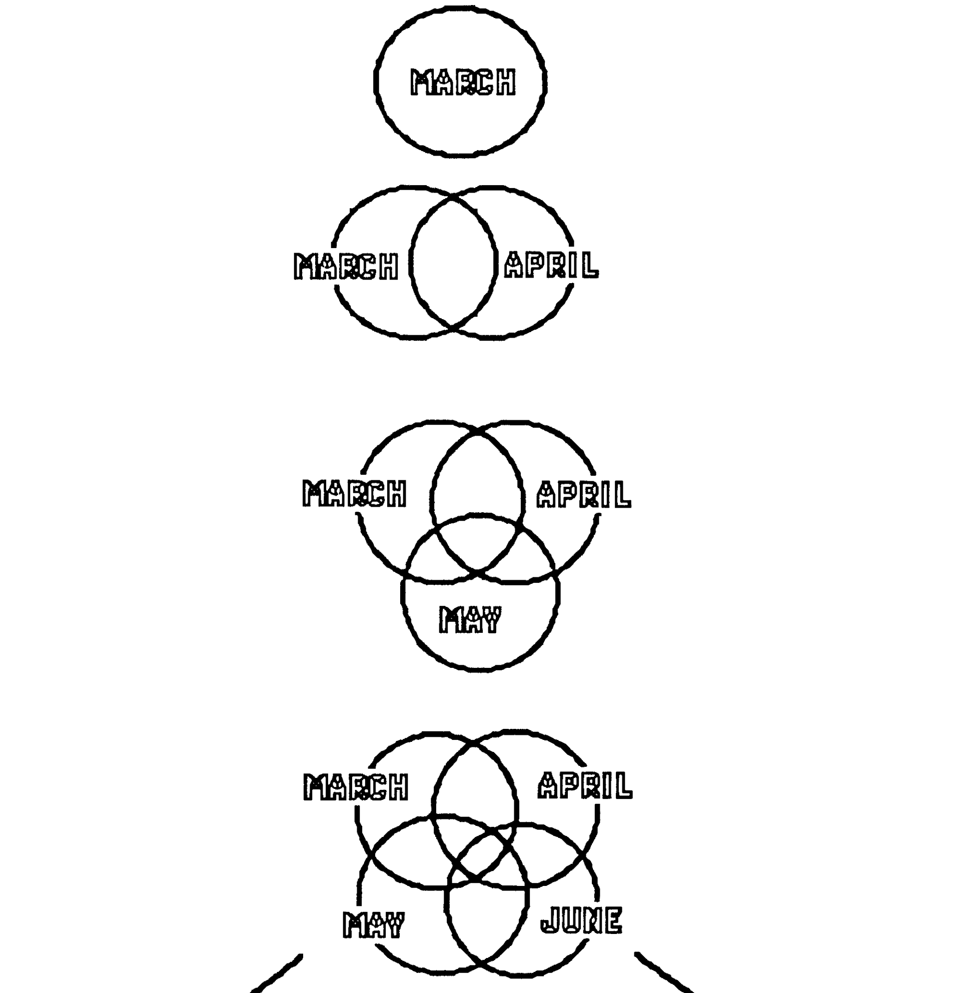
In neither chapter will I place an emphasis on microcausality, in other words, there will be no strong and direct correlations between an "X" aspect or variable of the Treatment and an "$X^1$" aspect of the result. This is because the present study describes a rare pedagogical situation in which learning was considered to be a total activity; all the different kinds of learning were incorporated into each other, with emphasis on the contribution of each to all.

In Chapter II, entitled Debbie's Case, this point will become clearer. The reader will see the profound interaction between all the different facets of Debbie's learning throughout the Project; he will also realize that Debbie's learning processes were often too complex for it to be possible to do more than speculate on a specific and discrete cause for each effect or outcome. In Chapter III, entitled Results from Pre- and Post-Tests, I shall analyze the Logo and fractions results separately. This may create the impression of a microcausal-type investigation, but the reader must remember that it is quite artificial, and was done strictly for clearness of presentation.

# SOFTWARE DESIGN FOR LEARNING

# CHAPTER II:

# DEBBIE'S CASE



*TO HALF ...TO CALF ... TO HOUSE ... TO TWO ...TO QUESTION1 ... TO FRACTIONS TO WAIT.FOR.USER... TO IGNORE ...TO QUEST ... TO INTRODUCTION...TO BIG..TO SUPER ...TO EXAM ... TO INTRO ...FRACT2 ... TO EQUIVALENT ... TO MORE ... TO UNDERSTANDING...TO OKAY ... TO TOPO ... TO TALK.. TO FIX...TO TOGETHER...TO WARNING...TO FIX1...TO FIX2... TO EQUIV... TO ADDINGFR...TO ADD...TO SUBFR ...*

## INTRODUCTION

### 1. Who Is Debbie?

Debbie is a shy black girl, socially insecure and painfully aware of her excessive plumpness. She lives in a small apartment in Roxbury, one of Boston's lowest SES communities. Both her parents work, and she has one younger sister. In February 1987, at the time the Project started, Debbie was nine and a half years old and in the fourth grade.

In the six lessons I observed before the Software Design Project began, I did not perceive Debbie as an active member of her class. She did not participate very much; classes would often go on and on without her contributing anything to them. Unless the teacher asked her to give an answer or express her ideas, she would rarely volunteer to do so. On only a few observed occasions did she raise her hand to answer a question. Most of the time she looked sleepy or bored, and very reserved. In the schoolyard, during recess, she would stand and look around, or chat with another girl, feeling perhaps uncomfortable, because of her size, about running around and playing games like her peers. The overall impression I got from my observations, together with the teacher's information, was of an unhappy, passive, shy, and insecure girl who never seemed to get excited about anything and was just getting through another day in school.

Debbie took some time to grow accustomed, get started, and involve herself in the Instructional Software Design Project. It also took her a while to get used to me. In her relationship with me, and in her attitude towards the Project, Debbie went through what I describe as two phases: 1) the first ten days of the Project, her confused phase; and 2) the deeply engrossed phase, in which she was completely involved and successful in the Project.

**Phase 1:** During the first ten days, while Debbie's peers were getting started on the Project, Debbie could not seem to do the same. She kept writing in her Designer's Notebook: "Tomorrow I'll start my Fraction Project..." Also, unlike the other children, she did not happily welcome me when I entered the classroom. She did not jump up like the others and say "Come see what I did yesterday!" In fact, she did not let me see anything she did at all. She was very reserved. She acted skeptical about her ability to deal with this large, new, and demanding Project. She complained, "Do I really have to do this?", looking rather upset, and said to me: "How?" Her teacher declared that this was Debbie's "regular attitude," and described her as "an average student having quite good writing skills but relatively low skills in math, definitely not the star of the class," who had "difficulties with some social aspects" in the life of the classroom.

The best summary of "who Debbie is" is in fact provided by her own reflections in writing. On March 31, when she was well into the Project, Debbie stopped working on her program for ten minutes; instead, she left her Fractions Program, opened a new Logo Page, and wrote two very personal poems. The same day she told me: "Logo is good for many things, even for animated poems about myself." In these poems Debbie confirmed my first impression that she was a depressed child. Originally, I felt very uncomfortable about presenting these two highly personal poems in this context; but I finally decided to quote them here, since they are worth more than a thousand of my own words about who Debbie is. She wrote her two poems using the following Logo procedures:

```
TO NO        [ Poem No. 1]
HT PR [This life is stupid to live in, nobody to love or care for you!
No one to drive you places and someone to see that you don't get hurt.
No one, no one to care or worry about you!]
HT SETC 1+RANDOM 14 REPEAT 20 [REPEAT 4 [RT 90 FD 30] RT 18]
END
```

```
TO GROW      [Poem No. 2]
HT PR [Every day I grow and grow. In every way I see my self grow.
I grow to see other things, I grow to like, I grow to hate.
No matter what I see I grow!]
SETC 1+RANDOM 14 PD REPEAT 15 [REPEAT 360 [FD .2 RT 2] RT 20]
END
```

The structures of the two procedures are very similar. Each of them includes hiding the Turtle (HT), writing a statement about herself (PR [.....]), then creating a simple animation. I interpret the first poem or procedure as Debbie's reflection on her anger about her "stupid" and lonely life (i.e., "This life is stupid to live in...No one to drive you places or someone to see that you don't get hurt. No one, no one to care about you!"). The animation in this procedure is a design of a circle of squares that appears in one color, and is selected randomly by the computer as one reads the poem. The second poem or procedure can be interpreted in two ways. First, as a child's reflection on growing (in age) in general. Secondly, and more specific to Debbie's problem, as a child's reflection on growing fat. The animation in the second poem, together with my knowledge of Debbie's awkward appearance, convince me that in this second poem Debbie was expressing her feelings about her excessive plumpness. The second animation is a circle that is growing and growing and growing as one reads the poem on the screen. Because of her particular personality and the type of relationship we had by then established in the Project, I felt that it was inappropriate to question Debbie about these two poem procedures.

In Chapter Three of *The Second Self*, Sherry Turkle (1984, pp. 93- 136) describes the case of Tanya, who through her "world of words" established strong relationships with the computer, and later, through the computer, established relationships with her peers and teachers (ibid., pp. 122-126). Like Debbie, Tanya felt comfortable with the computer when she used it for writing, for expressing something personal and painful (in her case, writing stories about classmates she had been afraid to speak to). Turkle believes that Tanya's appropriation of the computer through language, through using it to write very personal stories, was her process of "making things her own." Turkle observed many children and adults using the computer as an "evocative object" that encouraged very deep self-reflections. Evocative objects, says Turkle, "provoke" a discourse in everyday life on topics that are usually more compartmentalized:

> "The computer sits on many boarders; it is a formal system that can be taken up in a way that is not separate from the experience of the self. As such, it may evoke unconscious memories [feelings and emotions] of objects [or events] that lie for the child in the uncertain zone between self and not-self...Psychoanalytic theorists call these objects 'transitional' because they are thought to mediate between the child's closely bonded relationship with the mother and his or her capacity to develop relationships with other people who will be experienced as separate, autonomous beings" [Turkle, *The Second Self*, 1984, pp. 118-119].

Debbie also seemed to also appropriate the computer (and the Project as a whole) through her "world of words" and self-expression. Moreover, the fact that she could use the same object (Logo, or the computer) for personal and painful poems, and at the same time, for learning mathematics, probably helped her to establish strong relationships with mathematics, with the Project, and with the other objects or people that were involved in the Project.

**Phase 2:** It took Debbie some time to get engaged in the Software Design Project and to feel comfortable with its rituals and routines, and completely in charge within its open-ended structure. It also took her a while to establish a relationship with me, and "let me into" her private programs and ideas. But it seemed that once she got into it, she grew very involved in it, and unlike some other children in her class, she never "got stuck" (i.e., I did not perceive the fact that she took short breaks to write her poems or other procedures that were not related to the Project as "getting stuck" in any way.) During what I call her the second phase, I did not see Debbie in a situation of "not knowing what to do," or dissatisfied and upset about what she programmed and implemented; on the contrary, she kept generating many new ideas and was eager to implement them; in other words, I saw her appropriation of the computer through her using it for teaching and explaining, for designing interesting screens, and for writing her personal poems. Debbie, who was usually so shy, quiet, and not the "star" of her class, became, during the Project, independent and also known for her good ideas. She came to feel creative and successful because other children wanted to see or play with her software, and gave her positive responses: "I love it Debbie!," or "This is fresh!", they would say, then ask her to teach them how to do things: "How on earth did you make these colors change?" I think the Project was a very new and different learning situation for someone like Debbie. She went on and on with her designs and programs, and unlike many other children, never actually said or wrote in her Designer's Notebook that she was "done."

Debbie's second phase is described in this case study. It is important to note that my collection of data includes several quite different, sometimes even more successful or more interesting model cases than Debbie's (in terms of the children's screens and final products, software design processes, or their progress in learning about fractions or Logo programming). However, for reasons related to educational practice, cognitive, learning, and problem-solving research, I selected Debbie's case for this Thesis. In the following paragraphs I shall point out some of these reasons to the reader, since they are the very themes or theoretical issues that are repeatedly highlighted throughout the analysis and presentation of Debbie's case.

**A.** One theme that will be highlighted here is Debbie's particular personality, and her attitude towards school learning and socializing in general. Debbie is a model case for an average, inner-city public-school fourth-grader: a young black girl, very reserved, not very happy, in fact somewhat depressed, who usually gave a strong impression of being rigid and reluctant to try anything new. She was not academically or socially successful, but through the Project she grew to feel better about herself, opened up, and interacted more freely with the people around her. Through the Project she was "discovered" by her peers and teachers, and quite often received positive responses about her work from them. Her teacher said to me at the end of the Project: "...I realized her rather sophisticated programming skills, her excellent ideas for teaching [fractional concepts], and her interesting representations...really, I never thought of Debbie as someone who was able to do all these wonderful things."

In general, Debbie did not have the "ideal" personality for participating in a type of an experimental investigation that required from children a great deal of collaboration with the researcher and a constant sharing of their ideas, thoughts, knowledge of various kinds, designs, and programming problems. (The reader will be able to realize this in reading my conversations with Debbie which are reported throughout the Case; I had to constantly play games with her, and find ways to make her answer more than "Because," or "I don't know.") But, to my mind, it was all the more important to gather information on someone like Debbie because she was not at all the typical child whom researchers would spontaneously choose to study or interact with during their pilot studies, their educational innovations, or their short-term clinical investigations. In fact, one of my colleagues who also conducted short-term research at Project Headlight told me that she had got very negative responses from Debbie when first beginning her research, and had consequently decided not to work intensively with her. Because of her time constraints, my colleague felt that she would benefit much more, and gather more detailed information, by working with children who were more enthusiastic, articulate, and open-minded towards the new projects she was conducting (therefore better able to express and share their ongoing thoughts, ideas, and processes). But we must remember that many children of Debbie's type exist in classrooms; and if we believe in correlations between the child's personality and her cognitions, we should gather detailed information on such children's ways of learning and thinking in complex projects of this kind (complex not only because of the academic, but the social factors involved). We should not concentrate solely on children who are eager, open-minded, and enthusiastic about trying new things, and therefore make it easy for various researchers to interact and work with them.

**B.** Another important theme that I shall stress is Debbie's writings in her Designer's Notebook. Debbie planned and designed screens in her Designer's Notebook in greater detail than many other of her classmates. She eventually came up with some plans and screen designs that were complex both conceptually and from the point of view of programming. The Designer's Notebook was another "evocative object," to use Turkle's (1984) terminology, in Debbie's activities and thinking throughout the Project. Debbie's relatively high writing skills and her love for drawing helped her to appropriate the Project through the use of language and artistic expression; it also helped me to gather a great deal of data about the problems she had in math and her processes of thinking and learning about fractions and software design. As described previously in Chapter I, the children's writings in their Designer's Notebooks were an important element in their own processes of learning, and at the same time in my data collection. And Debbie, who loved writing and drawing in general, and in her Designer's Notebook in particular, planned, drew, and made detailed notes in writing on her processes of software design, programming, and fraction learning. This helped me a great deal in the construction of her case study, and also suggests some kind of a correlation between a child's writings and her development in mathematical thinking about fractions. It was probably a situation in which a child's high skills in "X," contributed to the improvement of her low skills in "Y." It becomes even more interesting since, in educational practice, the "Y" (math skills) is often considered to involve a very different set of skills than "X" (writing).

C. Debbie is also an interesting case in the context of Piaget's general characteristics of cognitive development--more specifically, his theory of the underlying patterns of thought during the different stages of cognitive development. In her processes of thinking, designing, programming, and learning throughout the Project, Debbie moved according to Piaget's progression scheme overall, and also within each step or stage of her software design. In her thinking and actions during the Project, Debbie moved back and forth from being attentive to limited and static amounts of information, to considering several aspects of a situation simultaneously; she moved from concrete thought to more formal thought; from rigid thinking that focused, for example, on one dimension of a programming problem, to more fluent and dynamic thinking related to several dimensions of her computer programs (we shall see this aspect through comparing her work during March, to the ways in which she later reorganized her software and programmed new super-procedures during June). She shifted from narrow and rigid actions to more flexible actions, so that she could grasp several aspects of a situation at once; and though she began by constructing very simple plans, designs, and screens, she was later able to create more complex plans, designs, and screens, moving back and forth from simplicity to complexity. In general, Debbie learned to break away, to free herself of rigidity and acquire more flexibility within the Project's length and breadth (across the board)--in her translating and combining representations of fractions, in her Logo programming techniques, in her planning and reflecting in the Designer's Notebook, and in her attitude towards the Project as a whole.

**D.** Another theme that will be highlighted is related to the differences between the Vygotskian and Piagetian aspects of learning. Debbie's learning suggests a combination of both. She is both a strong case for constructivist learning, and for social-cultural learning. In other words, there were several situations in which Debbie learned from her peers or from the adults around her, imitated her friends' ideas and designs, was affected by my questions and interviews, and was strongly inspired by her culture; however, side by side with those situations, she also spontaneously came up with her own ideas independently of her friends, or shifted from rigid to more flexible thought without the help of an adult. These two different aspects of learning (i.e., imitating as opposed to inventing, or being guided by an adult as opposed to being guided by her own intentions), created an interesting learning pattern that repeated itself in Debbie's work throughout the Project. Vygotsky's perspective on the role of language in children's learning and thinking will be emphasized as well throughout Debbie's Case.

**E.** Finally, another aspect in Debbie's learning that will be discussed is her exceptional progress in learning fractions and Logo through the Project. The Pre-Test scores indicate that Debbie's abilities in Logo and Fractions before the Project began were average or low (she was a member of the medium-math group, and was one of the lowest pupils in this group); however, her scores at the end of the year after the Project ended (in the Post-Tests), were of a high-math child, often at the top 10 percentile of all the 51 tested children. In the Post-Tests she constantly scored higher than all the medium or low-level children in the Experimental and Control classes, as will be described in detail in Chapter III. Debbie's final high scores are particularly interesting because, before the Project started, the teacher had described her as someone who was low in math skills but quite good in writing skills. After the Project ended, Debbie progressed dramatically in her acquisition of high math skills as well. This year, in fact (1987-1988), Debbie became a member of the high-math group of her fifth grade; according to her teacher's reports, she has made great progress in math. These findings will be discussed in the context of, and in relation to the other themes, A, B, C, D, and E presented above. It seems that the combination of: changes in her attitudes towards learning and towards new and open-ended projects; positive responses she received from her teachers and peers about her software and ideas; her overall joy and satisfaction during the Project and feeling positive about herself afterwards; her particular writing, designing, and learning processes throughout the Project; her being able to work on her own as well as with other people during the Project; and her particular cognitive progress of breaking away from rigidity to generating and understanding complexities--should be all equally considered as important factors when looking at the results from her Post-Tests. For these reasons and others, which remain to be explored and described in this chapter, I selected the story of Debbie as a model case for demonstrating a child's potential learning of fractions and Logo by the method of instructional software design.

## 2. The Data Collected for Debbie's Case

The information about Debbie's day-by-day development in the Instructional Software Design Project was gathered in the same way as for the whole Experimental class. This has already been described in Part 3 of Chapter I of my Thesis. In the following subsections I briefly summarize the research techniques, data-collection instruments, and analysis that were used in the construction of Debbie's particular case.

**Pre- and Post-Data:** Information about Debbie's knowledge of fractions (using two written tests and an interview) and Logo programming (using a written test and several computer tasks) was gathered before and after the Software Design Project. To evaluate Debbie's case, her scores are compared with those of the 16 other children in her class (the

Experimental class), and of the 34 children from the two Control classes. Emphasis is placed on comparing Debbie with her medium-math peers from both the Experimental and Control classes. (The rationale and objectives behind this Pre- and Post-Testing procedure are described in detail in Chapter III. Samples of the tests are in Appendix B.)

**Process Data:** The information about Debbie's cognitive and learning processes throughout the Instructional Software Design Project includes 50 Logo files, which were gathered almost daily on the computer, and 45 sets of plans, designs, and notes, also gathered in her Designer's Notebook. Debbie's Designer's Notebook files and Logo on-line files, which were used for this case, include her notes, designs, plans, and pieces of Logo programs for the following days:

| | | |
|---|---|---|
| In March: | 4, 5, 10, 11, 12, 13, 16, 19, 20, 23, 24, 26, 27, 30, 31 | (15 daily files) |
| In April: | 1, 2, 6, 7, 9, 10, 13, 14, 27, 28, 30 | (11 daily files) |
| In May: | 1, 4, 5, 7, 11, 14, 18, 19, 20, 22, 26, 27, 29 | (13 daily files) |
| In June: | 4, 6, 12, 13, 16 [the school year ended on June 23] | ( 5 daily files) |

(The rationale and objectives behind this process-records procedure are described in detail in Part 3 of Chapter I. Samples pages from Debbie's Designer's Notebook and selected Logo files are in Appendix C).

**Videotapes:** Debbie, like several other Experimental children, was videotaped seven times during the period of the Software Design Project--on several occasions when she was working at the computer, and other times with her whole class, in classroom discussions. In general, I was very involved in the classroom discussions, and in working with all the Experimental children at their computers when they needed some kind of help. So, on some occasions, in order not to miss any ongoing information about Debbie, I used the video camera as a further recording instrument. After the experiment was completed, I reviewed and transcribed the tapes about Debbie. In other words, these tapes were used as raw data on Debbie in the same way as her Designer's Notebook or daily Logo on-line files.

**Researcher's Notes:** In addition to all the above I took notes on Debbie's progress and concerns on certain days, and conducted short interviews (informal conversations) with Debbie every once in a while--when something captured my interest, or when she asked me questions regarding her design, Logo programming, etc. No specific format was used for writing these notes.

**Information from the Teacher:** On several occasions I interviewed the teacher of the Experimental class about Debbie. Most of our conversations were informal, taking place during lunch in the teacher's room, or in the classroom when her pupils were in Gym or in Swimming. Through these interviews, I gathered information about Debbie's background, ability, character, and progress. No specific format was used in these interviews. Most of our conversations were short and consisted in my asking the teacher questions such as: "Tell me about Debbie," or "What do you think about Debbie's last screen?" or "Why do you think Debbie was behaving this way yesterday?" or "Did you work with Debbie on using a REPEAT in her Half.Fraction procedure?"

The five preceding major data collection techniques were used for building Debbie's individual case study. For the purpose of writing Debbie's story, various kinds of information about her were integrated and analyzed, using the videotapes, the researcher's notes, the teacher's comments and reports, the Pre- and Post-Tests and Interviews, and the various records of the software design process (i.e., Debbie's writings in the Designer's Notebook and her saved Logo files).

## 3. The Structure of Debbie's Case

In this chapter, I analyze Debbie's experience within the Instructional Software Design Project during the four-month period between the tests I conducted before and after the Project. The case follows in detail her cognitive processes and learning of design, fractions, and Logo through her designing, programming, evaluating, and modifying her piece of software to teach a third grader about fractions. My inquiry of Debbie's microgenetic processes is systematic, and is based upon a combination of: **1)** Process-Information, which assesses Debbie's day-by-day evolution of software design, and the way in which her work relates to those of her peers from the Experimental class; and **2)** Pre- and Post-Information, which assesses Debbie's progress during the period between January (Pre-Tests) to June (Post-Tests), relates her results in these tests to the different stages and steps of her software construction, and compares Debbie's learning of fractions and Logo with that of the other Experimental software designers, as well as that of the children who learned the material through different instructional approaches.

My focus is on Debbie's processes of design, her relationship with her own program, her relationship with the other children's programs, and her learning through the different stages of her program construction. However, the accounts of how this Project contributed to her learning are also checked against the results of her tests and interviews, and those of other children's. Debbie's level of learning and ways of learning will be compared with those of the other software-designers in the Experimental class, her peers in her math group, in the other math groups, and in the two Control classes. The information about Debbie's processes of software design will be presented in this chapter. The results from the Pre- and Post-Tests will be briefly presented here, but only when they are relevant to the discussion of Debbie's learning and her design and programming processes. However, her Pre- and Post-Results will also be presented at length in Chapter III ("The Results from the Evaluation"), Section 2.3.

In this chapter, I shall divide the information gathered on Debbie's processes of software design and programming into several main sections, based on <u>her own</u> natural segmentations in her software design, construction, and modifications. Usually, she herself announced her breaks in her Designer's Notebook, for example:

> "Today I'll finish my old fractions project. I'll start my new one about equivalent fractions."
> [Or:] "I finished my House Scene. It works. I'll start my Scene about Thirds tomorrow."

However, as her program grew long and complex, her plans and reflections in the Designer's Notebook became complex as well. Her breaks or segmentations were not as obvious anymore, because she took care of several aspects of her software simultaneously, for example, on April 7 she wrote:

> "Today I will do something else. I can't fit any more work on my fractions page. So I'm gonna do a new page called fract2. I'll do equivalent fractions on fract2. I might do some more poems. If I have time I'll do much more stuff on other pages, that I've learned."

In this case, I had to work more like a detective, and if Debbie actually worked on equivalent fractions on that day, I included her work in the Equivalent Fractions segment, and the work she did on other screens was included in the sections about them. The following are Debbie's stage sequence in organizing and implementing her long-term work:

**MARCH:**

1) GETTING STARTED WITH ONE-HALF

2) THE "FRACTIONS COLLECTION": THE DESIGN THAT WAS NEVER IMPLEMENTED

3) MULTIPLE REPRESENTATIONS FOR HALVES: A FIRST STEP

4) THE HOUSE SCENE: "FRACTIONS ARE EVERYWHERE" (or, THE SECOND STEP IN MULTIPLE REPRESENTATIONS FOR HALVES)

5) TWO-THIRDS: "ONE OF THE FRACTIONS THAT TEACHERS USE MOST OFTEN AS EXAMPLES FOR TEACHING THEIR STUDENTS"

6) THE FIRST ATTEMPT TO CONNECT THE PARTS: CREATING A SUPER-PROCEDURE

**APRIL:**

7) CREATING THE SOFTWARE'S OPENING SCREENS: DEBBIE, RALLY, AND TERRY

8) DEBBIE'S FIRST SOFTWARE EVALUATION WITH BIBBY, A THIRD GRADER

9) CREATING A NEW LOGO PAGE, "FRACT2": DEBBIE MAINTAINS STRONG RELATIONSHIPS BETWEEN THE NEW KNOWLEDGE AND OLD IN HER SOFTWARE AS WELL AS IN HER MIND

10) DESIGNING REPRESENTATIONS FOR EQUIVALENT FRACTIONS

**MAY:**

10.1) IMPLEMENTING ONE REPRESENTATION FOR EQUIVALENT FRACTIONS.

10.2) ANOTHER SCREEN FOR EQUIVALENT FRACTIONS

10.3) THE "SESAME STREET" SCREEN: DEBBIE AND NAOMI

11) THE "ADDITION ASSIGNMENT"

12) CREATING MORE EXAMS, "GOING OVERS," AND WORKING ON THE USER-FRIENDLINESS OF THE SOFTWARE

13) THE SUBTRACTION SCENE

**JUNE:**

14) FINISHING UP: AN OVERVIEW OF DEBBIE'S FINAL PIECE OF SOFTWARE

Together, these sections reveal most of the existing information about Debbie's day-by-day processes of software design and her ways of learning fractions and Logo through that process. It is important to note that these segmentations and their sequences are <u>unique and specific to Debbie</u>. There are no two children in the Experimental class with the same identical sections in terms of their actual number, content, or order. The segmentations of each of the seventeen Experimental children's software-design processes revealed that these children generated, constructed, and followed different routes in their processes of software design and in their learning of Logo and fractions through the Project. This interesting phenomenon will be described in detail in a forthcoming paper (Harel,1988, in progress), entitled: "Individual Differences in Children's Processes of Instructional Software Design".

Like the Software Design Project itself, Debbie's case is presented in a process-oriented fashion--much like a documentary movie on movie-making. I present Debbie's thoughts and actions in their context over a period of time, trying to capture Debbie's ways and sequences of working with fractions and programming, and showing how what she did relates to what her peers did at that time. The reader becomes familiar with Debbie's step-by-step process of creation, and only later will be given possible interpretations of Debbie's actions, and a view of her final product as a whole. When Debbie started the Project, she did not know how her software would look in the end, or what the parts of her final product would actually be. It would thus be misleading to present Debbie's software from the top (the final product's contents and objectives) to the bottom level (the product's actual elements), or from the whole to its components.

In other words, the presentation of the main sections in the Case follows the sequence of Debbie's processes of creation. These sections are conceptual and procedural in the way they describe her internal concepts and ongoing thoughts about fractions, her external conversations with her peers, and her instructional designing or programming at a given time and place in the process. The 14 sections do not necessarily correspond with the structure of the screens in Debbie's final product. In this way of presentation, we can assess Debbie's conceptual development within the Project through the evolution of her screen designs, the complex inter-relations between her software's parts, her pedagogical considerations, and her fractional representations.

To give an illustration of this, I do not start by describing her "Opening" and "Introduction" screens, which first appear when the software is running. These screens were created during April, almost two months after the Project started, and after she had completed other screens. To take another example, she designed her screens about equivalent fractions in April, then left them aside, and implemented them in May. I describe these screens in their chronological order and in the context in which they were created.

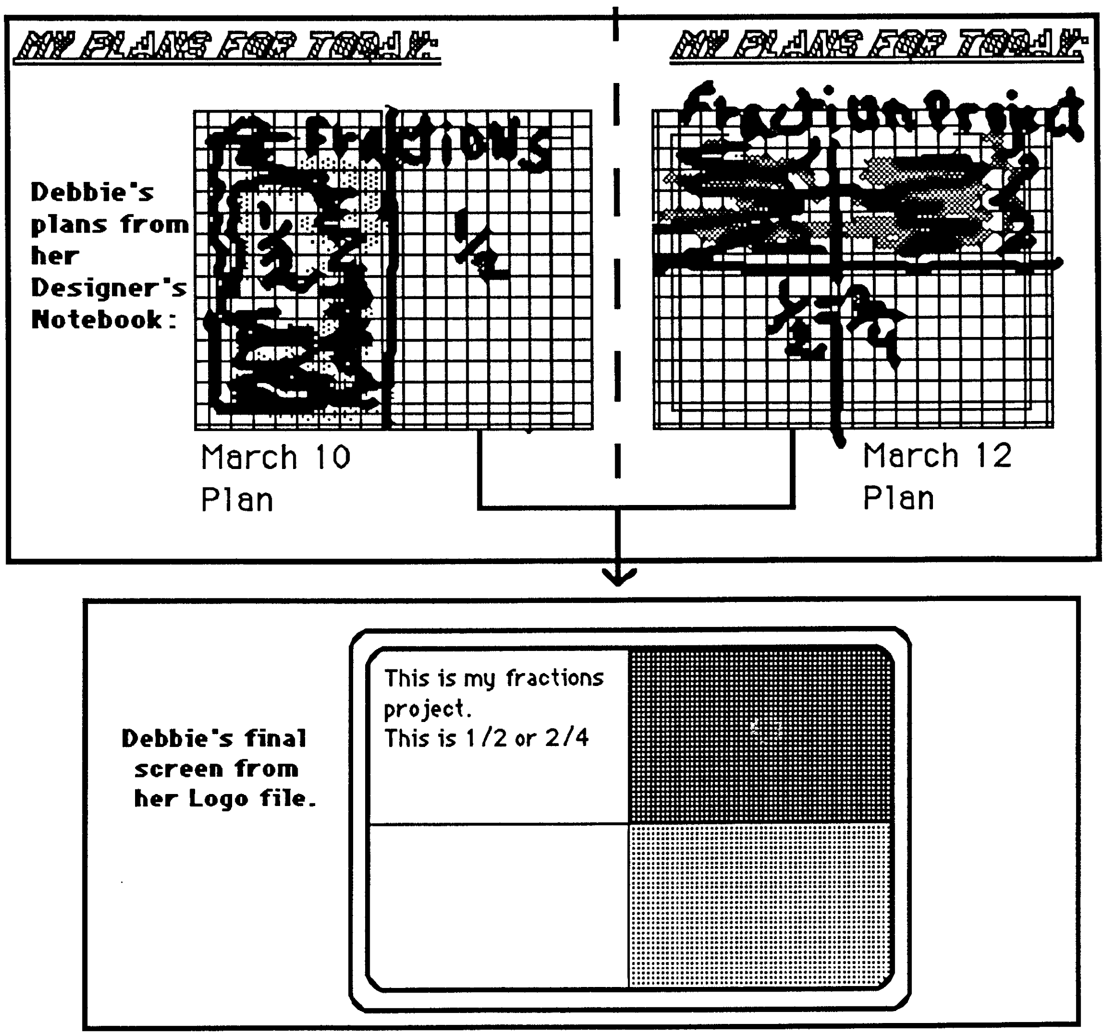
Their meaning and purpose are derived from their place in Debbie's process of creation, not just from their place in the final piece. I was more concerned with Debbie's processes and with the evolution of her knowledge and designing techniques than with the order of her screens in the final piece of software.

## DEBBIE'S SOFTWARE DESIGN PROCESSES

MARCH..MARCH..MARCH..MARCH..MARCH..MARCH

### 1. GETTING STARTED WITH ONE-HALF

Debbie's sketches of March 10 and March 12 show the evolution of the very first screen she designed. Two versions for the first screen were drawn by hand in her Designer's Notebook. The two designs were combined on March 16, and programmed into what became her first "teaching unit," a screen showing **1/2=2/4**. Although it was created first, in the final piece of software, this screen appears after the "Opening" and the "Introduction" screens that Debbie composed two months later.



This is my fractions project.
This is 1/2 or 2/4

**Debbie's Logo programming code for her first screen**

[with my explanations of it in the bracketts]:

```
TO HALF
SETC 9 FD 250                  [cuts the screen horizontally]
PU HOME RT 90 SETC 1           [goes back home and changes the pen color]
PD FD 320                      [cuts the screen vertically]
PU SETPOS [40 45]              [goes to the center of the top-right fourth]
LT 90 SETC 4 PD FILL           [fills it in red color]
PU SETPOS [40 -45]             [moves to the center of the bottom-right fourth]
PD SETC 3 PD FILL              [fills it in pink color]
PR [This is my fraction project]   [prints instructional sentence at the top]
PR [This is 1/2 or 2/4]
PR []                          [skips a space]
PR [By Debbie]                 [prints her name]
END
```

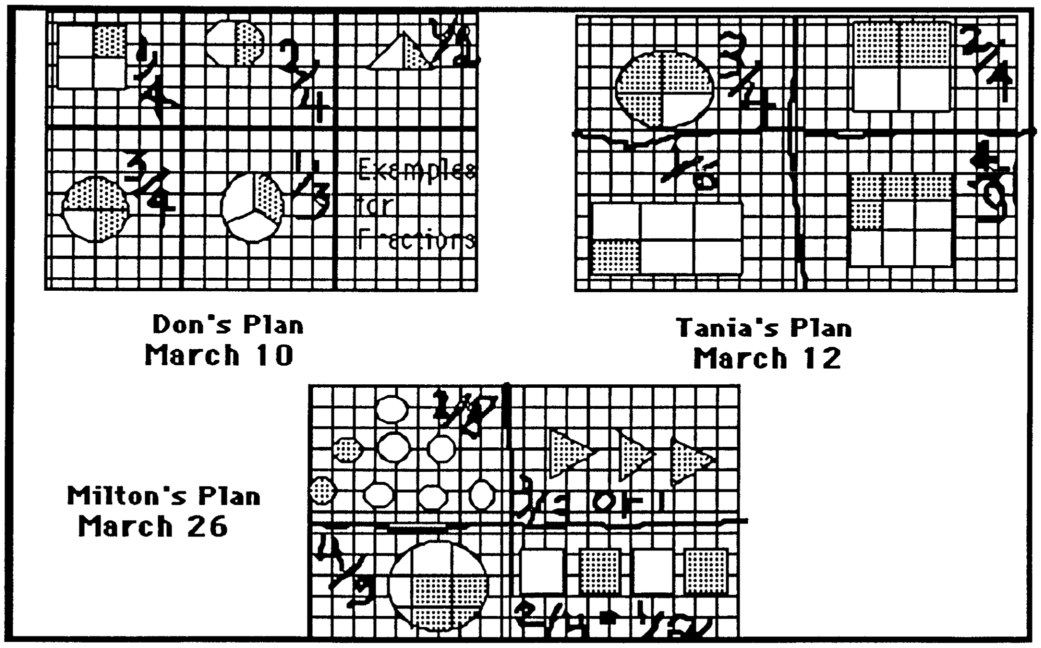
Debbie's procedure TO HALF is quite simple. She cuts the screen horizontally using FD 250, and vertically using FD 320, which causes the Turtle to wrap around the screen and draw the two perpendicular long lines. She changes the color of the pen (SETC) before each line and before filling in the colors for each of the two-fourths. In order to fill in the top and the bottom fourths, she uses the command SETPOS (which stands for SETPOSition), the primary tool for Cartesian graphics in Logo programming. SETPOS requires one input consisting of two numbers (i.e., SETPOS [X Y]). This list of two numbers moves the Turtle to the point on the screen represented by the two coordinates. For example, in SETPOS [40 45], x=40 and y=45. If the pen is down (PD), the Turtle draws a line on the screen as it moves to the point x,y or 40,45. If the pen is up (PU) the turtle moves to the required point on the screen without drawing a line as it moves.

A note should be made here on the children's learning of SETPOS. A few days before the Software Design Project began, we explained to some children what SETPOSition was, and how to use it. Many children designed their first screen with lines dividing it, to show examples for fractions. They could use FD (FORWARD) or BK (BACK) as a strategy for dividing the screen or moving the Turtle to any location on the screen, or they could use SETPOS with PU (Pen Up)or with PD (Pen Down). SETPOS requires understanding and calculating of the Cartesian coordinate points, and therefore takes longer for beginners in Logo to understand.

Debbie felt rather comfortable with SETPOS: she used it for her very first screen, and used SETPOS to move and place the Turtle on the screen throughout her entire project. In addition to the Logo skills that are apparent in Debbie's first and simple screen design, this particular design, with the colored fourths on the right side of the screen, also suggests Debbie's concept of halves and their representations. I shall discuss in detail Debbie's concept of halves, and her evolution in understanding and representing halves in sections 2, 3 and 4.

## 2. THE "FRACTIONS COLLECTION": THE DESIGNS THAT WERE NEVER IMPLEMENTED

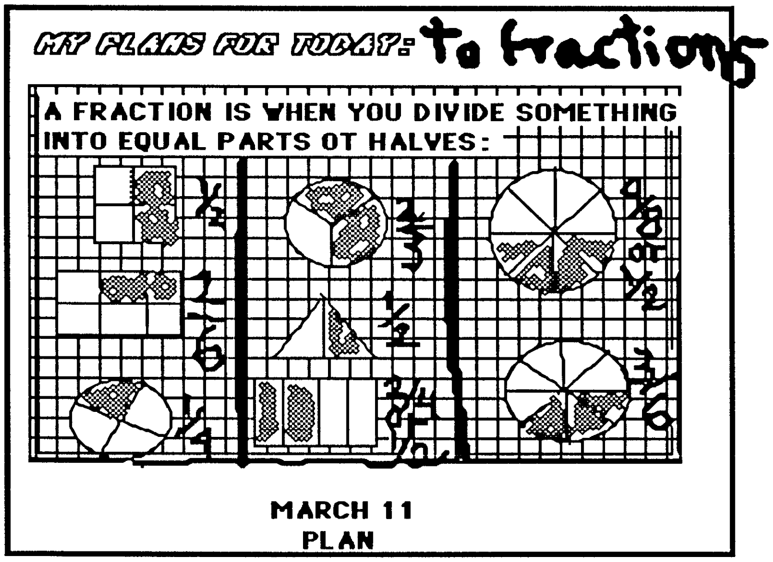
Many of the experimental children began their projects by showing on the computer screen a simple collection of basic fractional representations. The most popular representations were of the half-fourth-eighth family, or the third-sixth-ninth-twelfth family, as represented by the three children's plans in the diagrams below:

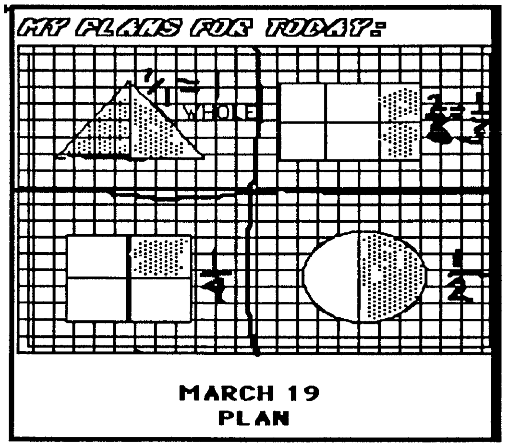


Don's, Tania's, and Milton's plans represent the children's typical getting-started approach, which I call "the collection designs." The children who started with "the collection designs" began their projects by designing a screen divided into four or six parts.

In each of the screen's parts they represented one fraction. Most of the fraction representations were done by dividing individual and familiar shapes (such as a circle, a square, a triangle, or a rectangle) into parts, some of these being shaded in. Another, more advanced representation (according to Lesh et al., 1983) was done by using a group of equal shapes, such as the group of eight little circles in Milton's design, where two of the circles were shaded in to show the fraction two-eighths.

As shown previously in Section 1, Debbie did not begin with this approach of representing a collection of fractions on the screen. Rather, she began by representing one large half using half of the computer screen. However, she saw the "collection designs" all around her in the Designer's Notebooks and on the computer screens of her peers. Judging from the next drawing in her Designer's Notebook, it seems that Debbie decided to adopt her class' idea of representing a collection of fractions. The following are the designs that Debbie drew in her Designer's Notebook on March 11 and 19:


MY PLANS FOR TODAY: to fractions
A FRACTION IS WHEN YOU DIVIDE SOMETHING INTO EQUAL PARTS OT HALVES:
1/2
2/3
4/8
1/2
2/6
1/2
1/4
3/4
3/6
MARCH 11
PLAN



There are two interesting design concepts in these hand-drawn pictures, made on March 11 and March 19. The first is Debbie's process of simplification of the design. She planned a busy screen on paper at first, on March 11, then "cleaned it up" on March 19, with the purpose of creating a design that would fit on the small screen of the IBM PC jr. As a designer, Debbie was aware of the "busy design" issue. My notes indicate that her reason for changing the design was "I will never be able to fit it all on one screen...too many things to see." The practical issue hidden in Debbie's reasoning was her difficulty, in terms of Logo, in creating so many little representations on one screen; the other, more sophisticated, design issue was that even if she could fit all these representations on one screen, they would still be cluttered and visually confusing for her future user. The second design concept that is of interest is the instructional sentence Debbie wrote at the top of her hand-drawn design of March 11: "**A fraction is when you divide something into equal parts or halves.**" The rational-number "half" had a special identity in Debbie's mind, as had been indecated in her Pre-Interview when she said that a fraction was the action of dividing something into halves.

The fraction "half" has a special identity for young children in general (Kieren & Nelson, 1978; Behr et al. 1983; ETC, 1985). Researchers report on children's notion of a half being a "special kind of a whole" and a special kind of an operation (see the Halving Operator, Smith, 1987 and ETC 1985). The half is the most "primitive" fraction, and an "easy" one for children to deal with. From a very young age children can partition quantities into halves quite accurately (ibid.). In my study, I also found among children a definite familiarity with the "halfness" concept. All the 51 children I pre-interviewed chose to show a half of something (blocks, pegs, pictures) as their first "good example for a fraction." Although the children were definitely familiar with the notion that "a half is a fraction," some of them had several misconceptions regarding those halves, which I shall discuss later.

90% of the Experimental children started their Software Design Projects by representing halves or equivalents of halves in one way or another. Perhaps their involvement with a large number of representations for halves resulted in their overcoming their misconceptions. Debbie is an interesting case because of the way she overcame her misconceptions about the concept of halfness. She worked with halves and explained about them for several weeks. Her later screens better illustrate her concept of halfness.

Debbie did not implement the two "collection designs" of March 11 and 19. Perhaps this is because she was so involved in working with halves only, and because she had so many ideas on how to teach about halves; at that time, she did not yet feel a need to move towards representing (or teaching about) other kinds of fractions. We will see later that throughout the month of March she created different screens with a variety of representations for halves.

In a more traditional teaching context, Debbie would never have been allowed to experiment with halves for so long, or to dwell on them for a whole month. But I realized that Debbie needed that time and was not ready to move on and deal with other fractions, even through the "collection design" technique. Although some of her peers were moving on, she decided to continue to play with halves only. We shall see later that the fact that she spent so much time working with halves did not decrease her understanding or her ability to work with other kinds of fractions or operations. In fact, she gave good answers in the Post-Tests in questions involving other fractions or more complex operations than equivalences of halves. Debbie explored the properties of fractions by spending a lot of time with these halves--the fractions she felt most comfortable with. White (March 1988, personal communication) interprets this phenomenon as Debbie's realizing the need for a "home base." He emphasizes that in many other learning situations one could observe children's creation of an "entry space," as the space "they feel most comfortable to live in when they encounter new and relatively unknown situations." In creating this "home base," children, in fact, create a very "secure" space for their exploring the new properties of an activity or a domain that is relatively new to them. In this way, through her intensive involvement with halves throughout the month of March and even afterwards, Debbie in fact gained control over the new properties and characteristics of the materials she was working with: rational-number concepts and Logo-programming concepts and skills.

## 3. MULTIPLE REPRESENTATIONS FOR HALVES: A FIRST STEP

In her Pre-Interview towards the end of January, Debbie defined a fraction in the following way: **"A fraction is a half. Yeah...A half is a whole. A half is a fraction."** When she was asked to construct representations for any fraction using blocks, play dough, rods, or pegs, she constantly built representations for halves, touched the "shaded" part of the shape she had constructed, and said: "This is a half, this is a fraction." When pointing to the object, I asked Debbie,

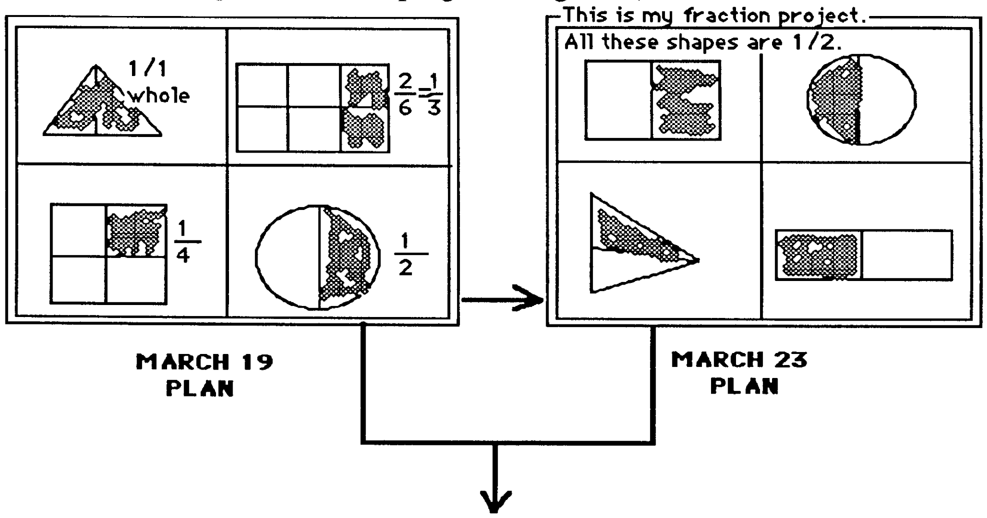
**I**: "And how do you call this [unshaded] part?"

**Debbie** answered: **"Um...this is nothing. Only the shaded part is a fraction...A half"**

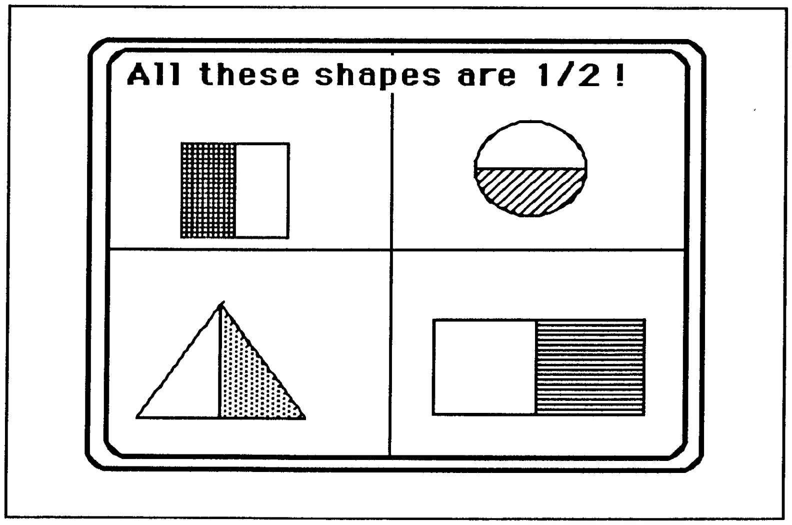
At a later stage of designing her software, Debbie, pointing at her drawings in her Designer's Notebook, told me,

**Debbie:** "You see, I made it like this. To show other children. **But both sides are a half.** It doesn't matter which one [side] is shaded in. I flip it [the shaded sides of the objects]. In each shape I shade a different side. So it doesn't say that the other side [unshaded side] is not a half."

The two following drawings from her Designer's Notebook determine the evolution of Debbie's screen of multiple representation for halves. On the next pages we see the final implementation of her design on the computer, and the final Logo procedure (with my interpretations on the right side of the programming code).



March 23 and 24 IMPLEMENTATION



**<u>Debbie's Logo Programming Code for the above screen:</u>**

```
TO CALF
HT SETC 1+ RANDOM 14                    [sets the pen color randomly]
PD FD 200                               [draws a vertical line across the screen]
PU HOME
RT 90
SETC 1+RANDOM 14                        [sets the pen color randomly]
PD FD 320                               [draws a horizontal line across the screen]
PU HOME
PU SETPOS [30 50]                       [moves Turtle to position of circle]
SETC 12 PD REPEAT 360 [FD.3 RT 1]       [draws a circle on the top right]
RT 90
PD FD 33                                [divides the circle in half]
PU SETPOS [45 40]
PD FILL                                 [shades in the bottom half of the circle]
PD HOME
PR [ALL THESE FRACTIONS ARE 1/2.]       [prints the instructional statement]
SETPOS [-60 50]                         [moves the Turtle to position of square]
SETC 1+RANDOM 14
PD REPEAT 4 [LT 90 FD 40]               [draws the square at the top left]
PU LT 90
FD 20
LT 90
PD FD 40                                [divides the square in half]
RT 90
FD 10
RT 90
PU FD 5 PD FILL                         [shades in the left half of the square]
PU HOME
SETC 1+RANDOM 14
SETPOS [35 -70]                         [moves Turtle to position of rectangle]
PD REPEAT 4 [FD 50 RT 90 FD 100 RT 90]  [draws a rectangle at the bottom right]
```

```
RT 90 FD 50
LT 90
PD FD 49                      [divides the rectangle in half]
PU RT 90
FD 25 RT 90
FD 10 PD FILL                 [shades in the right side of rectangle]
PU HOME
SETC 1+RANDOM 14
PU SETPOS [-105 -55]          [moves the Turtle to position of triangle]
RT 45
PD FD 50                      [creates a triangle]
RT 90
FD 52
RT 135
FD 73                         [positions the Turtle for cutting the triangle into
                               two equal parts]
REPEAT 2 [RT 90]
FD 36
LT 90
PD FD 35                      [divides the triangle]
PU SETPOS [-55 -40]
PD FILL                       [shades the half at the right side of the triangle]
END
```

This procedure already demonstrates how fast Debbie progressed in terms of her programming in Logo. In Section 1, we saw her short and simple Logo procedure for the half of the computer-screen; and here we see a much longer and relatively more complex procedure for the second, multiple representations for halves. In the following subsections I compare and interpret Debbie's plans of March 19 and March 23 (above) and her implementation of these plans in the above procedure TO CALF; I also interpret her processes of revising the CALF procedure while learning how to use a new Logo command, RANDOM, and developing new perspectives about halves and their representations.

## 3.1. Rhyming Procedure Names

Debbie named "TO CALF" the procedure shown above, which represents four different shapes divided into halves. I asked her why she had chosen the name "CALF."

**Debbie answered:** "...Because there are...[the shapes] all showing different ways of halves. I already have a procedure [called] "HALF" that does one large half or two-fourths on the screen [the first representation she had made]. I couldn't name this one HALF too. I had to find another name. This name is cute, 'cause you read it in the same PAGE [these two procedures appear on the same Logo Page one after the other]. First, TO HALF, and then, TO CALF. HALF, CALF. I like it like this. Because they [the two procedures] are almost the same, but they're different. The names [of the procedures] also are almost the same, but different."

This phenomenon of naming procedures in this way is interesting for at least two reasons. First, Debbie wanted the two names to sound similar yet different. Secondly, she wanted the procedure names to sound "cute together." Debbie realized that both procedures represented halves in two different ways; so she also decided that the procedure names should also correspond. The words HALF and CALF not only resemble each other, but also rhyme with each other.

Debbie liked writing poems. She even wrote several poems while she was working on her fractions software. (I have already described two of these personal poems in the Indroduction to the Case). At the beginning of the Software Project she approached her Logo code as if she were writing little fraction poems or Logo poems. She thought it was "special." When I discovered these procedure names in her program and asked her about it, she smiled and seemed proud of her "invention." She thought the procedure names were not only "cute together" ("read it! read it!" she said), but also appropriate for the conceptual similarity they represented.

One day in May (two months after the CALF-HALF implementation), I asked her why she no longer made procedure names rhyme.

**Debbie:** "Well, I started doing lots of other things for my fraction project. My program is very long now. I have to remember many procedures, and names too. It's hard to find things if you don't remember their names, and what they do [according to their names]."

Debbie felt she had to change her approach in order to remember where the procedures were, and what they did.

Naming is a central concept in programming (Papert, 1980). "Good" programmers tend to write readable programs with meaningful variable or procedure names, especially when the programs are long, so that debugging and revising will be easier to do and remain under the control of the programmer (Carver, 1986; Soloway, 1986). Through her process of composing a longer piece of software, Debbie felt that it was important to name things in a meaningful way so that she would be able to find where things were, and remember what they did. For functional reasons, she stopped using this rather charming rhyming strategy. This could be interpreted as Debbie's metacognitive awareness of the need for a strategy switch; she was flexible enough to adopt a new naming strategy.

### 3.2. Screens With Random Colors: Taking The Target Users Into Consideration

Before describing how Debbie discovered the idea of randomming colors, I shall first explain how most of the Logo skills were learned in the Software Design Project.

One child would learn something new (such as REPEAT, IFELSE or VARIABLES), and use it in his or her program. Other children would see something "special" on that child's screen, without knowing how to "make it happen," and would want to use it in their own programs. After this, they would seek to learn how to use this "trick" in their Logo programs. In most cases, a good teacher, such as the teacher I was working with, chose this time to thoroughly explain this Logo command to all the children during the last five minutes of the computer session.

Debbie was not the leading programmer of her class. On most occasions, she was not the first one to learn a new Logo skill, use it in her program, or "spread" it around her class. However, the following anecdote describes how, through Debbie's design-needs, she and other members of her class learned about the Logo operation RANDOM. (RANDOM is an operation that takes one input, a positive integer. The output from RANDOM is a non-negative integer that is lower than its input. RANDOM 3, for example, selects one of three choices, and outputs either 0, 1, or 2; RANDOM 5 selects one number out of 0, 1, 2, 3, or 4.)

One day in March, after the CALF procedure had worked for several days, Debbie came to me and asked,

**Debbie:** "How can I make it [the objects on the screen with the four shapes divided into halves] in different colors, so each time [you run the procedure] you get different colors?"

**I:** "What do you think?"

**Debbie:** "Well, I think I can do it [the same procedure] many times with different colors. I can do procedures CALF, CALF1, CALF2. It's too much work...But, do <u>you</u> know how to do it? [did I have a "trick"?]"

In that part of the school, the MIT people were known to know about such Logo tricks. It was just a matter of knowing what you, the child, wanted to do, and when. Judging from her question, Debbie was ready to learn about RANDOM. So she was the first one in her class to learn that the commands "SETC 1+RANDOM 14" would set the Logo Turtle's pen coloring in a random way. Each time this line of instructions is used, the computer selects one number out of 14, adds 1 to it (so that there will be no black or white selections), and outputs that color on the needed spot on the computer screen (each number represents a color, and there are 15 colors available in that version of Logo). This is a rather complex Logo operation to understand. Debbie seemed to grasp it (at least within this particular context). She was overjoyed with her creation and "invention." From then on, she used SETC 1+RANDOM 14 whenever she used colors. She began to use this RANDOM "trick" extensively. Her peers who saw or played with her software became aware not just of how beautiful it was, but also <u>how</u> to use it, and how they could change the screen colors randomly as a <u>design strategy.</u> The idea of "randomming" the colors was spread to almost the entire class.

**As Debbie said:** "Now, they [her future users] can use my fraction project many times. They will not get bored. They will pay attention to what I am showing them!"

## 3.3. Summary: Learning Something About Fractions (halves) and Logo (Random) Through Thinking About Teaching

The above anecdotes on Debbie's modifications of her CALF procedure demonstrate two essential aspects of the Software Design Project. **1)** Fractions learning and Logo learning were interlaced in Debbie's process of creating an instructional screen for someone else; and **2)** Debbie's own learning was "spreading around" among her peers, resulting in her peers' learning. In other words, Debbie learned a new "trick" in Logo (RANDOM) that made her screen look special. When she ran her modified procedures, they were visible on her computer for her peers to be inspired by. From then on, many of her peers adopted the new "trick" Debbie used (SETC 1+RANDOM 14) in their own procedures.

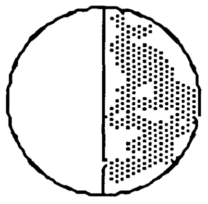
## 3.3.1. What does Debbie's CALF procedure reveal about her processes of overcoming her misconceptions about representing fractions, and about what a fraction is?

By that time, Debbie wanted to teach mostly about halves in her software. In the case of her screen, which I described previously (her procedure TO CALF), we can see the first obstacle she had to overcome in order to correct her misconceptions. She taught the concept of "halves" by showing 4 geometrical shapes divided into halves: a square, a circle, a rectangle, and a triangle. I will hypothesize here that her plans and her drawing in the Designer's Notebook showed her awareness of the problems she had initially had with halves, and wanted her users not to have.

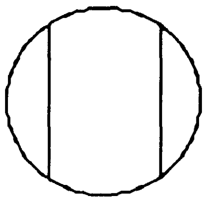
Debbie originally had a particular concept of a half (and of a fraction in general). This was reflected in her talking about halves or fractions in the Pre-Interview, and in her first screen designs. **1)** In her Pre-Interview, Debbie was not aware that the parts of a divided geometrical and symmetrical region must be equal in order to be a fraction in all cases. **2)** She thought that if a shape was divided into two parts, and one of the parts was shaded in, the other "empty" part was not a half. (In other words, she had to overcome her misconception that only the shaded part was a fraction). **3)** In the Pre-Interview, Debbie also insisted that a fraction was like a square divided into two parts, where only the right shaded half was the fraction.

Debbie's conceptions were typical of many fourth-graders I interviewed before the project began. In order to understand Debbie's development within the Software Project, we have to look first at her answers to my questions in the Pre-Interview:

**1)** In the Pre-Interview she drew a circle and divided it into two parts, saying, as she drew: "I divide it in the middle, right here. Here, this is a half, a fraction." However, immediately afterwards, when I asked her to draw another fraction, she drew another circle and divided it into three parts, saying: "I divide it here, and here, [pointing and counting] one, two, three fractions. Threes. Um...Thirds." She drew and divided the second circle into three parts, but was not aware that her "thirds" were not equal. This is represented in the following diagrams:

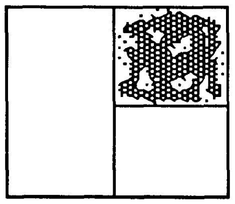


First Drawing:
Debbie's drawing of a half,
dividing the circle
"in the middle."



Second Drawing:
Debbie's drawing of a circle
divided into "one, two, three fractions. Thirds."
She was not aware of the fractional "equal parts schema".

I then asked her to draw an example for one third using a square. She first divided the square into two halves. Then she divided the right half into two. She shaded in the top part [fourth] and said: "and this is a third, [pointing at the three parts and counting] one, two, three, and one is shaded in. This is one-third."



Third Drawing:

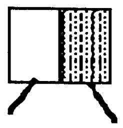
**Debbie:** "This is one-third."

**I:** "Is the big part a third too?" [pointing to the left half]

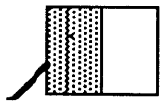
**Debbie:** "Yes. One-third."

(Note: We can also see her quick adaptation to my use of language: when she drew the thirds in the citcles she called them "One, two, three fractions...Threes." A minute later, after I asked her to give an example for one third by using a square, she clearly said: "This is one-third").

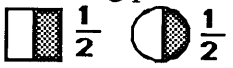
**2)** In the Pre-Interview, I showed Debbie a drawing of a square divided into halves. The right half was shaded in. I asked her what the fraction in the picture was. She touched the shaded part and said: "This is a half." I asked her: "And what is this other part?" She said: "This is nothing." I said: "This is nothing?" Debbie answered: "Yes. It's not a half. It's not a fraction."



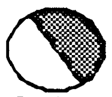
3) In that same Pre-Interview, I showed Debbie another identical square divided into two halves, but the shaded half was on the left side. I pointed to the left shaded half and asked her: "Is this a half?" She said, "Not really. Uhm...Yes. But you have to turn it kind of," as she was rotating the square. And when the shaded half appeared on the right side again she said, "Yes. It's a half."


Q: Is this a half?"
Debbie: "Uhm.. Not Really. But you have to turn it kind of."

One way of interpreting the source of Debbie's notion of "the fraction must be on the right side of the shape" is through an analysis of the fourth-grade regular school worksheets about fractions. My analysis of the worksheets used in the selected class's math curriculum indicated that most of the shaded parts in shapes were on the <u>right side</u> of the shape, as is shown in the following pictures:



The researchers Lesh, Landau, and Hamilton (1983, p. 310) have also found that up to 7% of the children they tested from grades 4 to 8 have problems identifying a half with a slightly rotated circle such as:


**What Is This Fraction?**

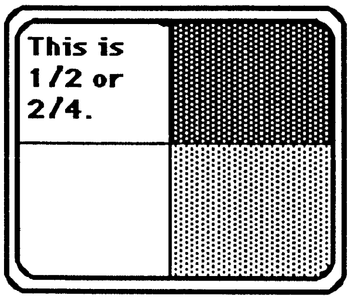
At the end of the Pre-Interview, I also gave Debbie the above rotated circle-region picture and asked her which fraction the picture showed. In order to answer my question, she rotated the paper to the right until the middle line was perpendicular, and only then said "a half."

Debbie's actions and thoughts in this situation could suggest several interpretations.

One is simply that her previous experiences with representations of halves were limited to representations of areas that were cut vertically. Another intrepretation could be given here by using Piaget's perspective on children's pre-operational rigidity, and their not being able to focus in a flexible way on two aspects of the situation at once, or on a situation that is slightly different from their most dominant experiences of the past (i.e., Debbie was probably mostly exposed in the past to a circle divided into two parts by a vertical line, rather than by a diagonal line). However, another interpretation might be that a vertical symmetry-detector in her sense of perception was preventing her from seeing that the picture showed a representation of a half, without actually rotating the picture. The literature on symmetry perception and feature-detection (e.g., Olson, 1975; Beiterman, 1986) describes a similar phenomenon (i.e., that children have a hard-wired, low-level, perceptual symmetry-detector). In fact Olson found that young children develop the right-left symmetry (which he calls low-level detection) much faster than the top-bottom symmetry. The fraction half, in the way it is represented using circle or square regions divided vertically, might be the easiest for children to recognize and analyze, thus it is considered to be an intuitive symmetry. For this reason, in their processes of learning about representations of fractions (or halves), children might have to learn how not to use the "vertical-only" symmetry-detector; rather, they must become more flexible and start using other, more sophisticated, perceptual detectors instead. However, if this aspect of perception is a hard-wired feature in children's minds, it might also be true that they first have to learn how to rotate the shape in their head to match it with their symmetry detectors of halves. This interpretation suggests that Debbie was sufficiently flexible in answering this question to not say: "No. It is not at all a half." Rather, at the time of the interview, she found a way to recognize that the non-shaded half was indeed a representation for a half; she detected the symmetry, but could not rotate the circle mentally--she could only do it physically.

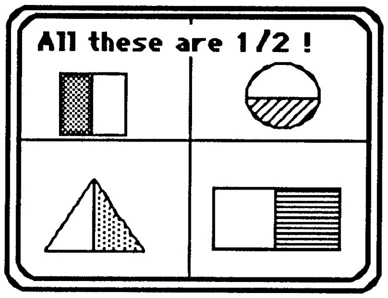
In summary, knowing this information about Debbie's conceptions of halves before the Project began, such as her strong beliefs that "the shape must be divided vertically in order to show a fraction" and that "a fraction must be on the right side of the shape," and knowing that she generally thought that "a fraction is a half" and was not aware of the fractional "equal parts schema"--make it easier to understand why she designed her first screen in one particular way, and, more important, why she designed her later screens in another.

Debbie's very **first** computer-screen (illustrated in Section 1 above), represented her idea that a "fraction is the right shaded side of a square." This is shown in the diagram below:



MARCH 16

However, in her **second** computer-screen (section 3), made two weeks later on March 24, we see that the shaded halves were located on different sides, top, bottom, right, or left of the 4 different shapes. She also wrote the following sentence at the top of the screen: "All these shapes are 1/2!"



MARCH 23, 24

Through wanting to teach about halves, Debbie constructed her own representations for other children. Inspired by the short discussions we had in the classroom, she was probably involved in a process of thinking about "what is really difficult for me to understand--that I can make easy for others."

I now suggest that 1) Debbie's warnings such as "Pay attention, all these shapes show 1/2!", and 2) her designing a screen for other children and flipping shapes and their shaded parts, were two important steps towards overcoming some of the above misconceptions about fractions (i.e., not being aware of the equality of the parts schema; thinking that the non-shaded part in a shape was not a fraction; and thinking that the shaded part must be on the right side in order to be a fraction). And while she was involved in her learning of fractions, she also learned new Logo skills. Debbie's involvement with fractions representations was a result of her wanting to program her designs. On most occasions, she had many ideas about what she wanted to design or teach, but not the knowledge of how to do it (i.e., she was not sure how to implement her designs in Logo).

Debbie, in fact, continued to work on these concepts in her fourth screen, the one she called the "HOUSE SCENE," where she implemented the same idea of multiple representations for halves using more complex, sophisticated, and real-life examples such as a house, a few wagons, or a sun. This screen is described in Section 4 below.

In the present Section 3.3.1., and in Section 3.3.2. which follows, I describe how two pieces of knowledge were mutually supportive of one another (i.e., fractions and Logo programming), and what were the consequences of Debbie's learning both pieces of knowledge at the same time, rather than separately.

3.3.2. What does Debbie's CALF procedure reveal about her processes of learning new Logo programming skills? Designing her third screen with the four shapes partitioned into halves did not require extremely sophisticated programming skills. Initially, she did not use REPEAT for all the shapes she created. She used REPEAT 360 for her circle, and REPEAT 4 for her square. She was also very quick at answering my questions about how to make them larger or smaller. By that time (March 23), she had full control over her circles, squares, and the SETPOSITION command used for placing her objects the way she wanted them on her computer screen. However, she could not see that, for example, a rectangle needed only REPEAT 2, instead of REPEAT 4 [FD 50 RT 90 FD 100 RT 90]; she only discovered this later on in the project. At that time, she could still not figure out how to draw a triangle using REPEAT. In fact, she had difficulty understanding how to program a triangle in the first place. In her Designer's Notebook, Debbie, reporting the work of March 23, wrote: "I had one problem. It was making a perfect triangle. Otherwise I had no problems....I made one change because it messes up [her procedure]. So I did it over and over. Finally I asked Cassania [a class mate] and she helped me. She showed me how to do a perfect triangle...I finished that project" (i.e., the screen with the four shapes divided into halves).

It was Cassania, then, who helped Debbie understand "how to do" a triangle. And Debbie programmed it using a string of commands of FD's, RT's, and SETPOSitions. When she had finished programming her procedure, it was beautiful. It worked. On that day she said that she had "finished that project". (Debbie called each procedure, or each screen, "a project". She wrote about "my old project," meaning procedures she had completed; and "my new project," meaning procedures being planned in her Designer's Notebook, which had not yet been implemented on the computer).

Other Logo learning situations reported in the literature have documented the problems and difficulties young children have with programming in Logo (e.g., Pea & Kurland, 1984; Perkins et al, 1985; Heller, 1986). Most of the children in these studies usually programmed smaller programs; and when they were finished, and their programs were working, the process of programming was ended. But Debbie did not end here. In the context of the Instructional Software Design Project, she ran her procedures every day she logged in, in order to refresh her memory about what she had accomplished the day before. She also ran her programs every time she wanted to add new screens for her software. It was only after several days of running her working procedure CALF that she actually found the need to change something in it.

Children in the other situations reported by the above literature rarely went back to "old" projects to revise them, which apparently resulted in very little Logo learning. Debbie had to go back as part of her process of creating and fitting in new parts to her enterprise. Eventually she added RANDOM for the colors, changed her rectangle procedure to REPEAT 2, and re-created the triangle using REPEAT instead of the string of commands. Moreover, she went beyond a "program that works." She was thinking about how to capture her users' attention. She told me that she was thinking about "how to make the CALF procedure less boring." She wanted to vary the colors on her screen so that her users would be more interested, and so that, as Debbie said, "they [would] learn better." Through thinking and expressing her ideas about how children should learn, Debbie learned a new Logo skill. As described previously, Debbie wanted to change something on her screen (colors), but did not know how to program it. That was when she began to learn SETC 1+RANDOM 14. She did not think about the color issue when she was planning her screen in her Designer's Notebook. On paper, there were no colors or interactions with the colors to make her think about this issue. Her first goal was to make her paper plan work on the computer. Only after she had run it on the computer for several days did the idea of varying the colors occur to her. Only then did she realize that a revision was necessary. Thus, RANDOMing the colors became a meaningful concept which she needed to learn in order to accomplish the designs she had in mind.

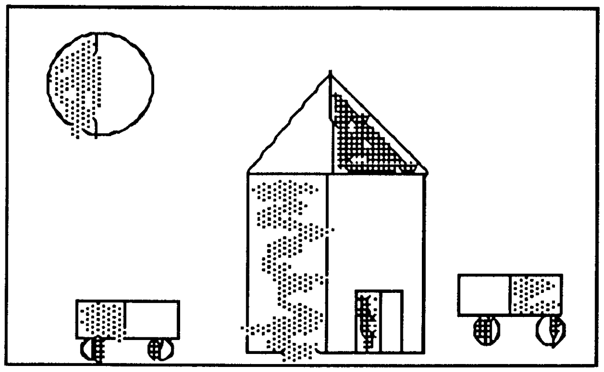
I have placed a great deal of emphasis here on this stage of Debbie's learning. The reason is that Debbie's activities, as I have described them in the previous paragraphs, should ideally occur as often as possible. (Can we compare such a moment with those experienced in a computer lab, with a computer coordinator standing in front of a class and lecturing about the importance of RANDOM and its meaning for a programmer?) At this stage of Debbie's software design, she was highly motivated to learn and use RANDOM for her coloring of the shapes, or REPEAT for constructing these shapes--both being quite difficult programming constructs to grasp. She learned and used her new skill relatively fast because it was meaningful to her and gave her a useful solution for the designs she had in mind. And all this came about because she was trying to find ways for the user to feel, in her words, "less bored."

## 4. THE HOUSE SCENE: "FRACTIONS ARE EVERYWHERE"

On March 23 Debbie wrote her plans in her Designer's Notebook:

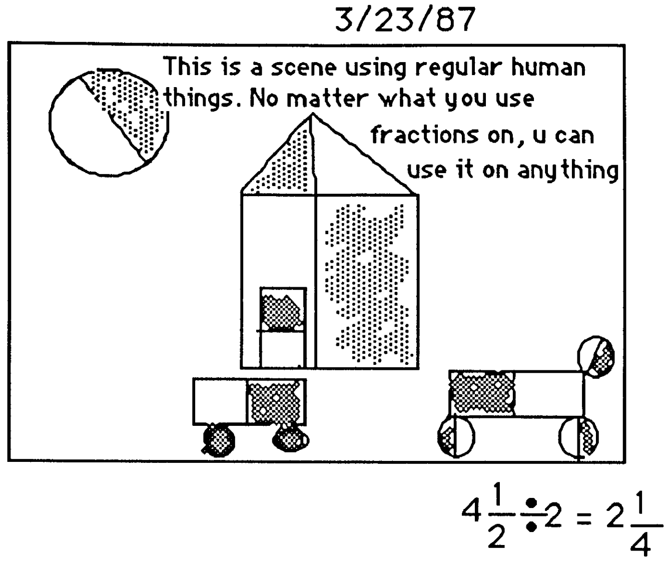
"I think I have to finish my old fractions project [the above computer screen in Section 3, with the 4 shapes divided into halves]. I am going to make a scene, where every shape is shaded half. When I'm finish I'm going to do 2/3 for my project & so on. The [current] scene is a sun, a house, made of triangles, rectangles, & squares. I'm going to have two little wooden wagons. The wagons are made of one rectangle and two circles each. It looks like this:"

**THE HOUSE SCENE** [HAND-DRAWN PLAN NUMBER 1]:



On that same day, before she went to her computer, Debbie quickly sketched another screen for the same "scene" on another piece of paper:

THE HOUSE SCENE [HAND-DRAWN PLAN NUMBER 2]:


3/23/87
This is a scene using regular human things. No matter what you use fractions on, u can use it on anything
$4\frac{1}{2} \div 2 = 2\frac{1}{4}$

[The numerical calculation at the bottom of the picture is part of Debbie's original design.]

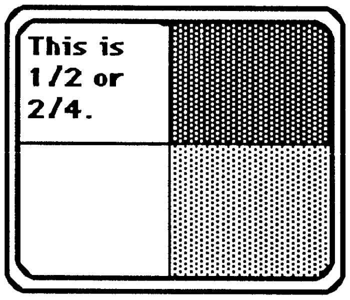
Putting these two plans in the context of Debbie's designs prior to these two plans reveals three interesting transformations. They are described in the following subsections.

## 4.1. The Three Transformations In Debbie's Designs

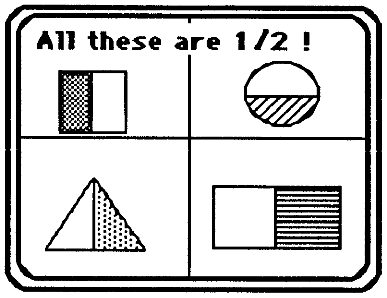
**1)** The first transformation was her shifting from rigid and simple geometrical designs: the first screen divided into four parts showing the half of the entire computer screen, or the second screen divided into four parts in which four shapes (circle, square, triangle, and rectangle) were divided into halves--into a relatively more complex thematic/narrative design that showed a sun, a house, and two wagons. She even described the wagons as "wooden wagons." She shifted from a "hard" geometrical design to a "soft" story-like design.

**2)** The second transformation was her describing the plan as a "House Scene." It was the first time she had used the word "scene" for her designs. The scene was, in her words, a "regular human scene." By that stage in her software-design process, she had decided to represent fractions using everyday objects, in real-life situations. I asked her "Why?" several times, and each time she answered, "I don't know." However, she would later explain to her users that "No matter what you use fractions on, you can use it on anything." Her words "..you can use fractions on anything" [my emphasis] are interesting because they reveal her idea that fractions can be used--super-imposed--"on top of" any object. For Debbie, fractions were not part of the object, but rather could be put on anything. (In fact, by that time, Debbie did not really use fractions for "anything"; she only used shapes and discrete objects. However, other children in her class represented fractions using liquids, time, or word problems about distances. I assume that her own experience outside of school, and her peers' representations, influenced her thinking that "you can use fractions on anything"). These first two transformations were interpreted as such through analyzing the evolution of Debbie's screen designs. This evolution is shown in the following screens.

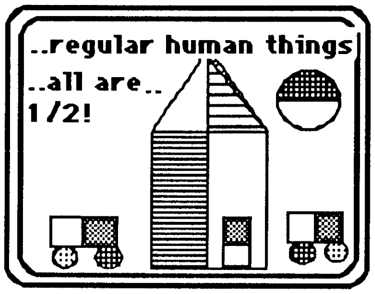
**Computer Screen number 1:**



**Computer Screen Number 2:**

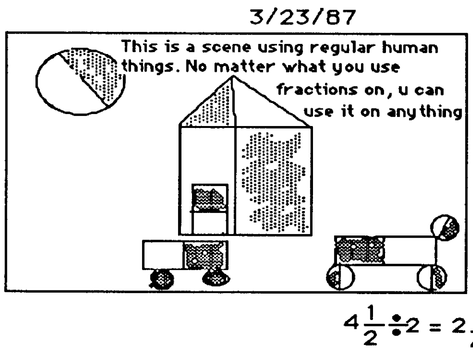


**Computer Screen Number 3:**



Debbie used simple geometrical designs at first, and later created a much more complex thematic "scene." In addition, she began by designing a single representation for a half in one geometrical figure, then shifted into multiple representations of halves using four different geometrical figures; later she shifted into multiple representation of halves in a real-life situation. Since the analysis of all the 17 children is not completed yet, I am not quite sure whether or not these transformations can be characterized as a typical trend of the children's evolution of screen designs, or whether these transformation are deeply rooted in all children's conceptual changes, rather than just in Debbie's. These important issues require further investigation.

3. The third transformation was interpreted from Debbie's plan number 2 for the House Scene, which is replicated below:


3/23/87
This is a scene using regular human things. No matter what you use fractions on, u can use it on anything
$4\frac{1}{2} \div 2 = 2\frac{1}{4}$

Debbie's second plan captured my interest as I walked around the classroom that day. I was struck by her attempt to attach a symbolic operation to her picture. I saw Debbie writing the following numbers under her hand-drawn picture:

$$4\frac{1}{2} \div 2 = 2\frac{1}{4}$$

I asked Debbie: "What are the numbers under the picture?"

**Debbie:** "If you count all the halves in the picture you get [pause] four wholes and one half."

She counted the halves on plan number 2 as she answered my question. She put two fingers on the two wagons halves and said "one;" then she put her two fingers on the wheels of the right wagon, and said "two;" she placed two fingers on the half-door and the half-house, saying "three;" then two fingers on the half-roof and the half-sun, saying "four;" finally she put her pinkie on the half of the steering wheel of the wagon on the right and said "and a half." I was even more fascinated when she continued,

**Debbie:** "Well, I don't think I'll use it. It's not really working..."

**I:** "What's not really working?"

**Debbie:** "The wooden wagons work, the wheels work...but the sun and the roof together do not work. Not really."

**I:** "What do you mean?"

**Debbie:** "I don't know...I am not going to use it."

**I:** "Why do you divide it into 2 here?" [she wrote 4 1/2 divided by 2 equals 2 and 1/4].

**Debbie:** "I don't know...I just tried to ask them questions..." [by "them" she meant "the users"].

**I:** "Try to use it..."

At that point, I did not want to distract her too much, realizing that she was very eager to go to her computer to program her designs. I assume that Debbie was attempting to translate her picture into symbols. She drew real objects, and, using her words, put halves on them. Then she felt a need to shift from her concrete real-life picture to a more formal symbolic operation.

There was no evidence for inferring that Debbie was imitating anyone when she came up with this idea. I doubt that she had ever been exposed to a real-life situation where all the real-life objects were divided into halves and a symbolic operation was attached to the situation. She was not imitating any of her peers, since she was the only one to come up with the idea of representing this particular real-life situation. For that reason, I do not interpret this moment of learning, or Debbie's three transformations, in a Vygotskian (1978) perspective. I suspect that Debbie did not produce this design by communicating with, or imitating an adult's action or idea; she did not internalize something from "the intellectual life around her" (Vygotsky, 1978 pp. 84-91). In this particular situation, she was not asked "leading questions," and was not in any kind of an "apprenticeship" ( in the way described by J. S. Brown & Collins, 1985), or "reciprocal relations with a knowledgeable adult," professional software designer or fractions expert (in the way described by Palincsar & A. Brown, 1984).

My assumption here is that Debbie's ideas for constructing a "House Scene" and her attempt to translate it into formal symbols appeared from her own mind. I interpreted this as an example for a rather creative learning moment that happened in a child's mind, independent of the people around her, but perhaps dependent on the task and the materials she used.

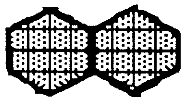
I have placed an emphasis on this issue because during the Instructional Software Design Project Debbie did inspire other children, and on several other occasions was inspired by others. During the Project, some kinds of learning that occurred were influenced by the "culture"--peers, teachers, MIT people, the nature of Logo, and the Designer's Notebooks; while other kinds of learning occurred due to the child's own mind, through his or her own constructing, thinking, representing and designing. During the course of this study, I tried, as much as possible, to identify with the children when they were inspired by their culture or when they came up with their own ideas. On most occasions it was difficult to tell which was which. Nevertheless, it seems to me that Debbie's House-Scene example--her original design as well as her attempt to translate it into symbolic operation--whether or not it was implemented in the end, can be seen as a model for a creative idea that resulted from one child's mind working independently of the environment, in a true Piagetian sense:

> "Piaget has sometimes labeled his position *constructivism*, to capture the sense in which the child must make and remake the basic concepts and logical-thought forms that constitute his intelligence. Piaget prefers to say that the child is inventing rather than discovering his ideas...The ideas in question do not preexist out there in the world, only awaiting their discovery by the child: each child must invent them for himself. By the same token, since the ideas have no a priori external existence, they cannot be discovered by a simple exposure [e.g., such as in imitation]; rather they must be constructed or invented by the child. Thus, Piaget's book dealing with the growth of the concepts of object, space, time, and causality...is not called *The Discovery of Reality*, but *The Construction of Reality in the Child* ." (Gruber & Voneche, *The Essential Piaget*, 1977, pp. xxxvi-xxxvii).

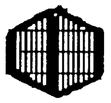
### 4.2. Debbie's attempts to attach symbols to the addition of real-life objects did not work. But did she learn to add fractions of other objects?

Why did Debbie abandon her idea of attaching symbols to the "House Scene" in her final implementation at the computer? Perhaps after she counted all the half-objects in her picture, she realized that it worked for the wagons, meaning that the two halves on the different wagons were more or less equal in size and could therefore be added into one whole.The two halves of the wheels of the right wagon were also equal, and could therefore be added into one whole. However, the half of the door was not equal to the half of the house, nor was the half of the sun equal to the half of the roof. Debbie seemed to have realized that "the wagons work, the wheels [work too]...but the sun and the roof <u>together</u> do not work," meaning that, their parts were not equal and could not be added into one whole. This was a very sophisticated perception of the properties of fractions, more specifically, the addition of fractions.

In the Post-Interview in June I assessed whether Debbie had indeed understood that adding fractions assumes adding equal parts, and whether or not she had understood the role of a common denominator. As part of the long Post-Interview (35-40 minutes), I brought out a box of Pattern Blocks (e.g., Harrison, 1972) and arranged two sets of the blocks on the table. One set consisted of two six-sided yellow shapes. The other set consisted of two four-sided red shapes. I told Debbie that I considered each set of blocks as a whole. This is how they looked on the table:



One whole composed of 2 yellow (six-sided) blocks.



A second whole composed of 2 red (four-sided, trapezoidal) blocks.

I asked Debbie: "One child told me that I can take half of the yellow shape, and half of the red shape and write: 1/2 + 1/2 = 1 ...What do you think?"

**Debbie:** "No. You can't."

**I:** "Why? This other child told me that that is true."

**Debbie** insisted: "No. This is wrong."

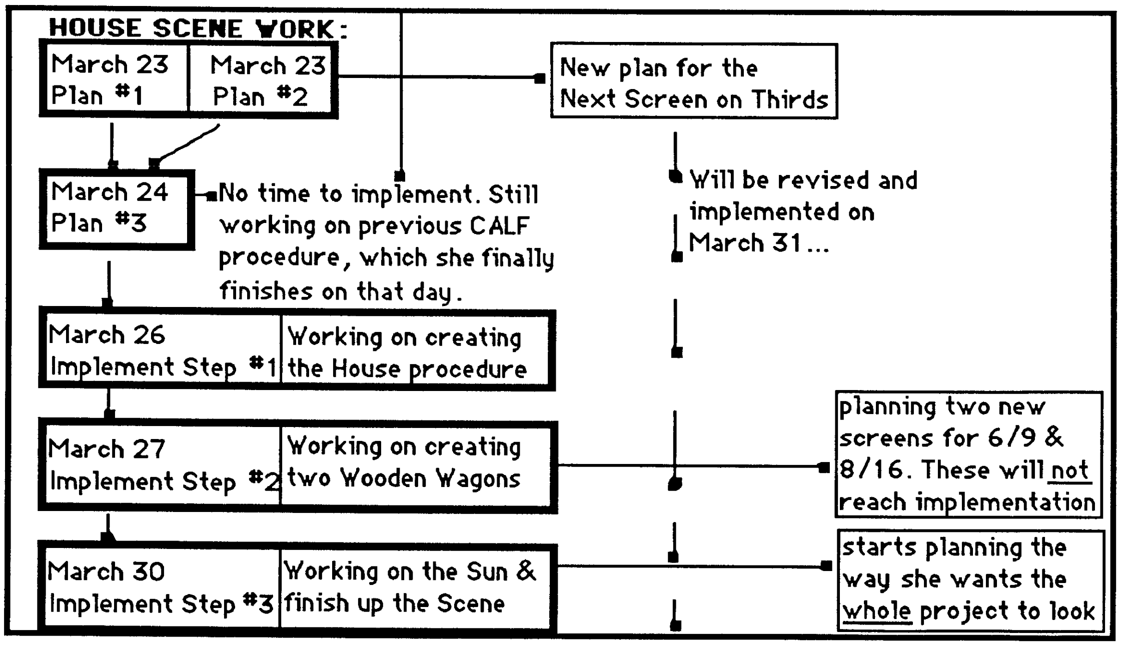
She took one half of the yellow shape (one yellow block), and one half of the red shape (one red block) put them together, and said "They are not equal. You can't do that." She then took two more red blocks from the box, and attached them to my original set of red blocks, so that the two shapes (the red and the yellow) would be equal in size (2 yellows = 4 reds). Then she said: "I'll do it like this. Now they are equal." Debbie had in fact created the common denominator (2 yellows+4 reds), in her process of adding the shapes. It is important to mention that approximately 90% of the <u>Control</u> children did not realize what Debbie realized in this segment of the Post-Interview. And many of them were children who had performed better than her in the Pre-Interview.

Let us go back for a moment to Debbie's answers to my questions about the symbols she attached to her design. As I illustrated before, when I asked Debbie why she divided 4 1/2 by 2 in her notes under plan number 2 for "House Scene," she replied: "I don't know...I just try to ask <u>them</u> questions." <u>What did she mean by this?</u> My assumption is that she was designing her "House Scene" to teach her users something. She also thought about a suitable test she could create using that same picture. Her test was not supposed to be pictorial, but symbolic. Perhaps she wanted to show her student a picture, and then ask a question which required an operation of symbols somehow related to her picture (she does it later with her 2/3 picture). At that time Debbie's knowledge about addition of fractions was quite minimal. My short interview with her demonstrated that she did have some knowledge about the properties of fractional addition; but she could not verbalize it to herself, or to me, and she was not able to revise her plans for using the picture and the symbols so as to test about addition of fractions. In fact, she never dealt with this issue again in the rest of her Software Design Project. Later in her project she created a screen for showing addition of fractions, but she used a different design approach (not a real-life situation).

In the following subsections Debbie's processes of planning, designing, revising, and implementing in Logo the House Scene are described.

### 4.3. Debbie's Processes of Planning and Programming the House

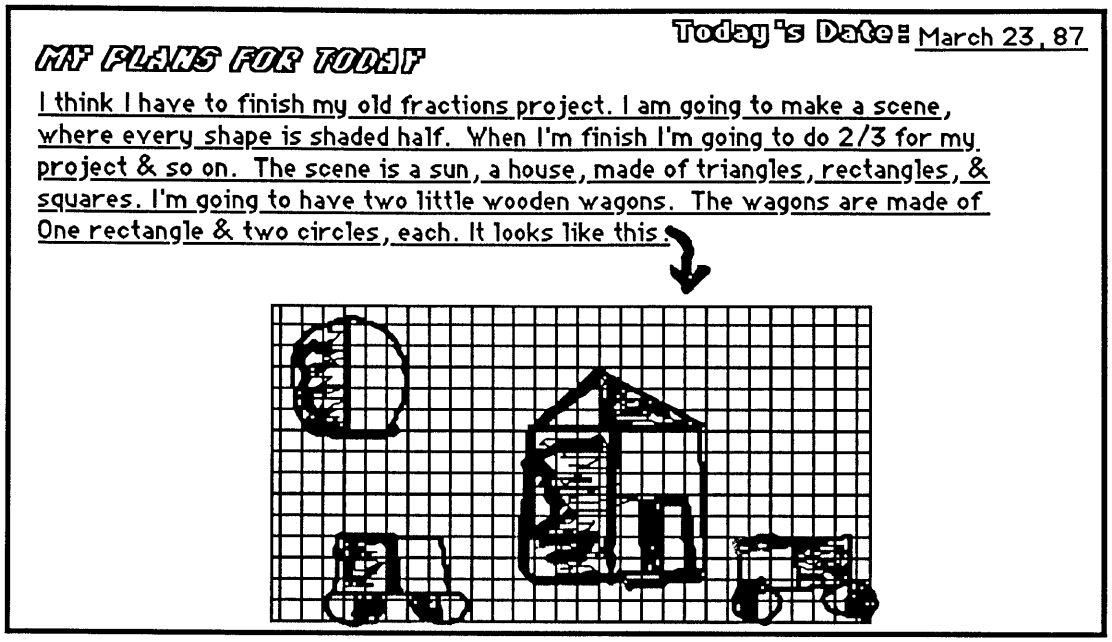
Debbie started planning her House Scene on March 23 and modified her plans on March 24 even before trying to implement them on the computer; this she began to do on March 26, and she continued with her implementation on March 27 and March 30. On March 30, she finally finished her House Scene. Debbie's intensive work on this scene--generating the designs, then modifying them, and later implementing and remodifying them--took her much longer than the other screens she designed and programmed during the Project. The general structure of her processes of designing and constructing the House Scene is depicted in the following diagram:


HOUSE SCENE WORK:
March 23 Plan #1
March 23 Plan #2
New plan for the Next Screen on Thirds
March 24 Plan #3
No time to implement. Still working on previous CALF procedure, which she finally finishes on that day.
Will be revised and implemented on March 31...
March 26 Implement Step #1
Working on creating the House procedure
March 27 Implement Step #2
Working on creating two Wooden Wagons
planning two new screens for 6/9 & 8/16. These will not reach implementation
March 30 Implement Step #3
Working on the Sun & finish up the Scene
starts planning the way she wants the whole project to look

**The overall structure of Debbie's processes of constructing the House Scene.**

Each box in the above diagram represents a phase in Debbie's process, and corresponds with each one of the following subsections. Replicas of Debbie's original drawings and written reflections from her Designer's Notebook are presented in each subsection, in order to capture Debbie's evolution of thinking about the House Scene; and her Logo programming code is presented when relevant, in order to show Debbie's ways of implementing her ideas and her programming processes.

## 4.3.1. March 23: Plan #1 from Debbie's Designer's Notebook

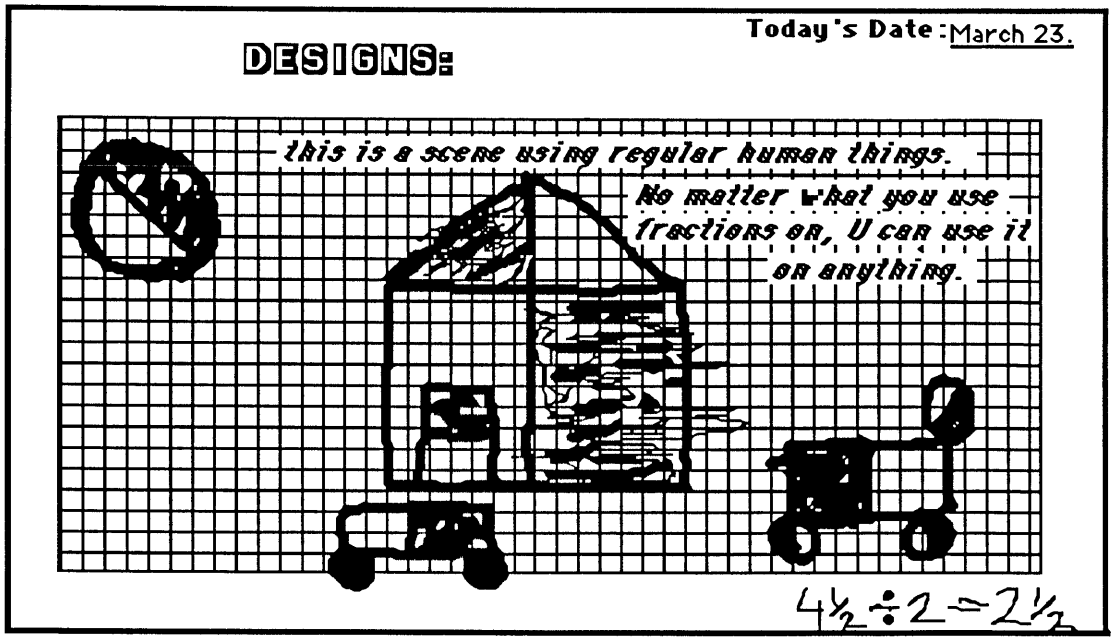


Today's Date: March 23, 87

MY PLANS FOR TODAY

I think I have to finish my old fractions project. I am going to make a scene, where every shape is shaded half. When I'm finish I'm going to do 2/3 for my project & so on. The scene is a sun, a house, made of triangles, rectangles, & squares. I'm going to have two little wooden wagons. The wagons are made of One rectangle & two circles, each. It looks like this.

## 4.3.2. March 23: Plan #2 from Debbie's Designer's Notebook



Today's Date: March 23.

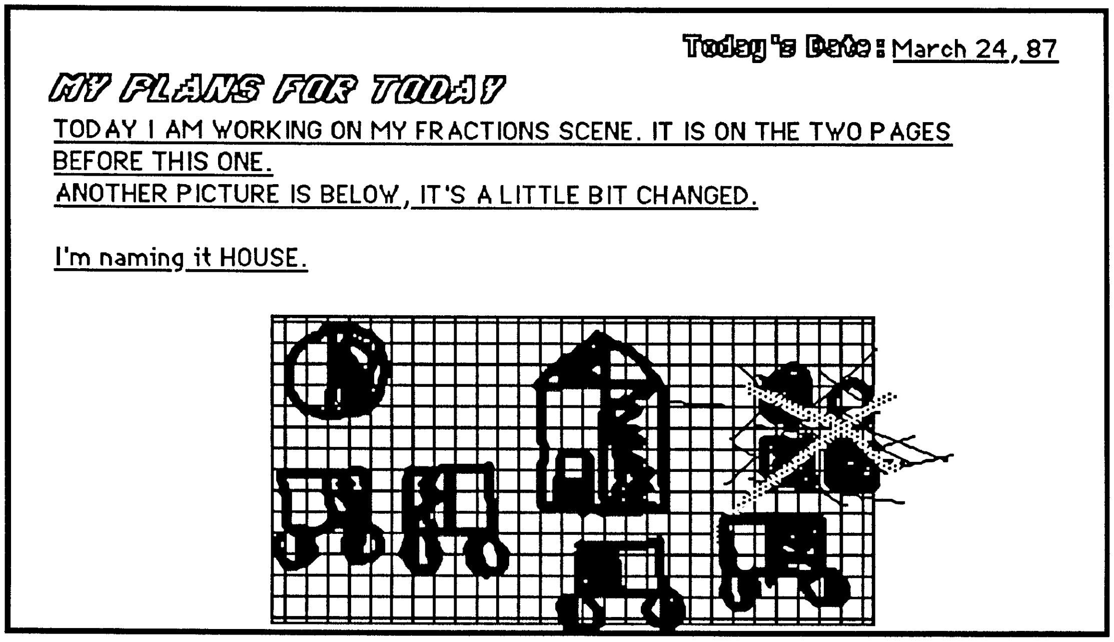
DESIGNS:

this is a scene using regular human things. No matter what you use fractions on, U can use it on anything.

$4\frac{1}{2} \div 2 = 2\frac{1}{2}$

The analysis of Debbie's instructional-designing transformations as shown in the above two plans fron March 23, her ideas about the fractional representations that are included in these plans, and her attempts to attach symbols to the real-life pictorial representation, were presented in Sections 4.1., and 4.2.; therefore, they will not be repeated here. In addition, on March 23 Debbie was still working on her CALF procedure and on that day she did not manage to start working on the House Scene.

### 4.3.3. March 24: Plan #3 from Debbie's Designer's Notebook



Today's Date: March 24, 87

MY PLANS FOR TODAY

TODAY I AM WORKING ON MY FRACTIONS SCENE. IT IS ON THE TWO PAGES BEFORE THIS ONE.
ANOTHER PICTURE IS BELOW, IT'S A LITTLE BIT CHANGED.

I'm naming it HOUSE.

Debbie really wanted to start working on the House Scene on March 24. Before going to the computer, she created another design in her Notebook: "...Another picture is below, it's a little bit changed." I asked her why she had slightly modified it that day, and she answered: "The more wagons I have, the more halves I can show." She then paused for a second or two, and continued: "And the more halves I have, the....they can be on top of more sides."

As stated in the previous sections of this thesis, at that point in her instructional-design process Debbie was very involved in showing children that the halves could be on any side of the shape (one recalls the misconceptions she originally had in thinking that the half must be on the right side of the shape, and that the "empty" side was not a fraction). Debbie was in fact saying: The more wagons I create, the more representations of halves on the different sides of the wagons and wheels I will be able to show. She was probably thinking that these ideas would become much clearer to her users if she used as many examples as possible for the same phenomenon.

On that day, rather early on in her process of planning, Debbie also stated in her Notebook her plan to name this scene "house." She wrote: "I am naming it HOUSE."

**I** asked her: "What will you name house?"

**Debbie:** "The procedure, of course."

**I:** "Which procedure?"

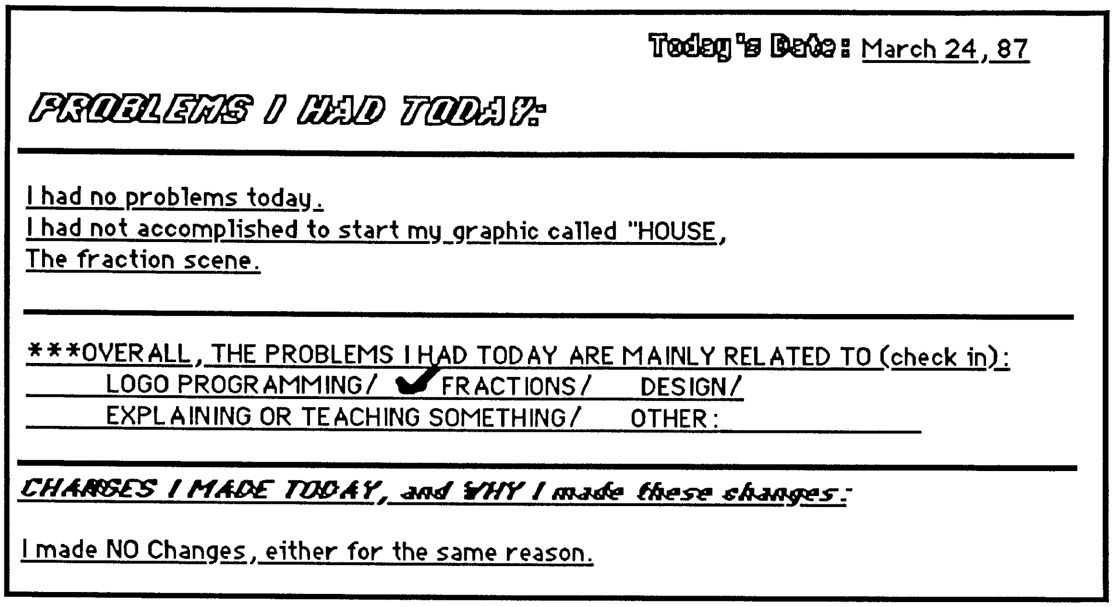
**Debbie:** "The House Scene procedure."

**I:** "Why house, and not wagons, or halves, or sun, or whatever?"

**Debbie:** "...'Cause it's a scene about a house. And...[pause] the house is the biggest too."

I asked Debbie these questions about her naming process for two reasons. First I was interested in knowing whether or not she would give a name for each part of the whole picture (i.e., To house, To sun, To wagon, etc.), and later create a super-procedure for all the objects shown in this scene. Secondly, I wondered why she had chosen a name that had no correspondence with the fractional representations it actually contained. One must remember that this was her third procedure in her Fractions Project, and that she had named her previous procedures HALF and CALF, using a rhyming strategy, and showing the purpose of the procedures (i.e., representing halves) by their names. However, Debbie's answers to my questions revealed that she had chosen to name her <u>entire</u> future procedure "house," and at that time was not yet planning to create sub-procedures for each of the objects or a super-procedure for the whole scene. She chose this name because it was for the House Scene, in which the house was the "biggest" and most dominant object. In this way, by choosing this name, she emphasized the fact that she was actually designing a real-life, "regular human scene" of a house and its surroundings. In addition, her thoughts about naming demonstrated that, during the process of planning in writing, she was already thinking about the Logo's framework of programming, that is, about the implementation of her designs in a Logo procedure.

However, as on March 23, Debbie was again involved in finishing up her previous CALF procedure on March 24, and due to the computer-time constraints, she did not manage to start her House Scene on that day as well. At the end of the computer time, after she had finally finished the CALF procedure, she went back to the classroom and wrote in her Notebook:



Today's Date: March 24, 87

PROBLEMS I HAD TODAY:

I had no problems today.
I had not accomplished to start my graphic called "HOUSE,
The fraction scene.

***OVERALL, THE PROBLEMS I HAD TODAY ARE MAINLY RELATED TO (check in):
LOGO PROGRAMMING/ [✓ FRACTIONS/] DESIGN/
EXPLAINING OR TEACHING SOMETHING/ OTHER:

CHANGES I MADE TODAY, and WHY I made these changes:

I made NO Changes, either for the same reason.

In the Reflections Forms in their Designer's Notebooks (i.e., the above "Problems I had Today" Form), the children would briefly describe whatever problems they had had that day. Whenever Debbie managed to solve her problems during the computer-time period, she would say, much like her peers, "I had no problems today." In general, the children reported their problems for one of two reasons: 1) when they got stuck with an unsolved problem at the end of a session; or 2) when they had a problem during the session that they could not solve by themselves and had to be helped by their teacher, myself, or their peers. This phenomenon--what children perceived as "problems" or defined as "problem-solving"--requires further investigation. It seems to suggest that the children's definitions and perceptions of these issues were slightly different from those of researchers involved in conducting studies about "children's problem-solving."

Another interesting issue related to this particular Reflections Form, and other similar ones, was Debbie's tendency to check "FRACTIONS" for the following item, in which the children were asked:

OVERALL, THE PROBLEMS I HAD TODAY ARE MAINLY RELATED TO (check in):
__LOGO /_X_FRACTIONS / ___DESIGN / ___EXPLAINING OR TEACHING SOMETHING /
__OTHER:______________________________________________________________.

It is important to note that all the other items on the Reflections Forms were open-ended. This was the only well-defined item, and had been designed by the researcher in order to capture the children's general feeling about what their problems related to, or what the main issues in each of the Instructional Software Design working-sessions were. If they wished, they could check none of the options, check only one, check a few, or all of the options. One day, I asked Debbie why she almost always checked in "FRACTIONS" as her answer to this item. I was fascinated by her answer:

**Debbie:** "Because I don't have any other problems. My problems are in fractions."
**I:** "What do you mean?"
**Debbie:** "I don't have problems with Logo."
**I:** "Really"
**Debbie:** "I mean...[pause] I know how to solve it in Logo but not in fractions."
**I:**"Aha..."
**Debbie:** "I mean that...I have to learn about fractions. I still have to learn A LOT!"
**I:** "And what about Logo, and designing your screens, and teaching third graders?"
**Debbie:** "Well...[pause]...It's not that I know it already, but <u>I can</u> think about it. I can...I usually know what to do. And with fractions it's much more trickier. I am thinking about fractions all the time. Besides, this is my Fractions Project...I do things about fractions...I do other things too. But I like explaining or teaching, there's no problem with that one. I have some problems with Logo, sometimes, but usually I can...Well, Naomi helped me once, and Don helped me once, and you helped me a few times...But I have [problems with] Logo <u>because</u> I've problems with fractions. [smiling] Do you understand?!"

I think I did understand...and I suggest here that Debbie's interesting answer showed her general concept of the Instructional Software Design Project as a whole. <u>What did it really mean for her</u>? For her, it seems that the Project as a whole did indeed mean "fractions." It also meant teaching and designing and programming and having fun, but above all--"fractions." My data indicate that Debbie was usually satisfied with her abilities in designing, drawing, writing, and teaching. She also felt satisfied about her involvement with Logo programming, and she did not perceive her occasional need for help in Logo as a problem. This may be a result of the "culture" and people's general attitudes towards Logo in that part of the school (see Section 1.2. "The Research Site: Project Headlight," in Chapter I). It also might be because she perceived that having problems with Logo was an integral part of the "deal". On another occasion she told me: "There is no one here that doesn't have problems with Logo. No one. You have bugs all the time but you fix them...to make your programs work."

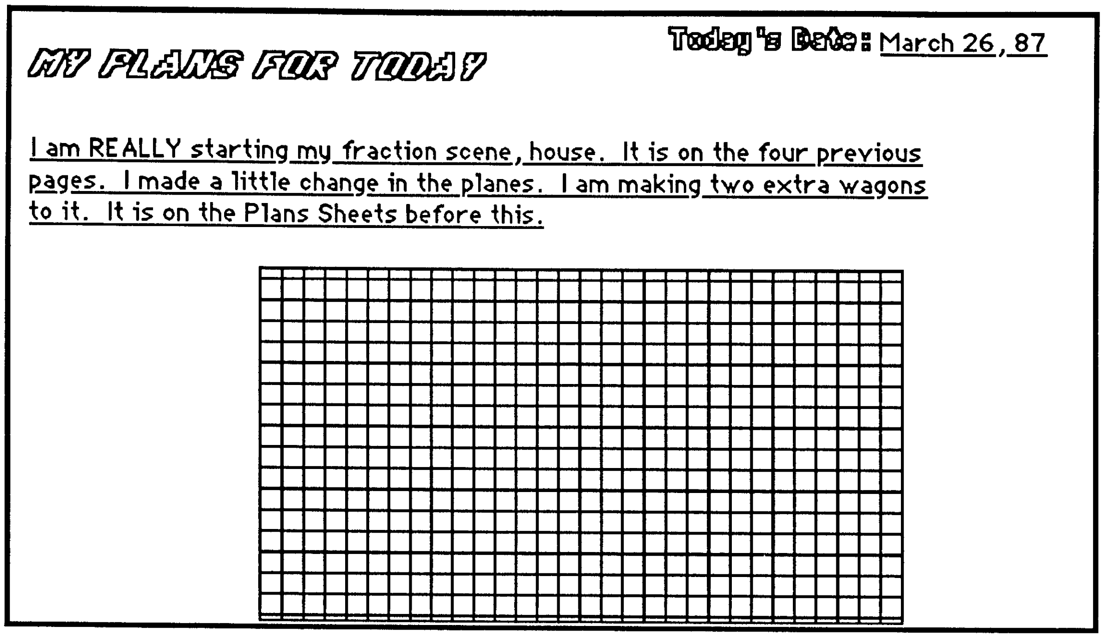
Her thoughts about the relationships between the Logo and fractions problems--that her ongoing difficulties with Logo related directly to (or were caused by) those with fractions--are very interesting. By saying what she said, Debbie revealed her constant conflict (and her metacognitive awareness of that conflict) of having many grandiose plans, and of usually wanting to design a complex screen about fractions that could not be easily implemented in Logo (due to her relatively limited skills in programming) and therefore caused difficulties in Logo as well: "But I have problems with Logo <u>because</u> I've problems with fractions."

In addition, Debbie's motivations behind each of her screens usually were: her thinking about her own problems with fractions, her thinking about what representation of fractions she should create, and her thinking about how to teach third-grade children about fractions. Moreover, in her regular math curriculum, Debbie was not very secure about her knowledge of fractions. She was not in the high-math group, and had not been able to pass any of the conventional math tests without making several mistakes, therefore getting average to low grades. She probably did not feel as satisfied about her fractions knowledge as she did about about designing, teaching or programming (also, there were no tests on them). All these reasons, taken together, probably explain why she only checked in "fractions" in the Reflections Forms as her constant and main "problem" within the software design process.

### 4.3.4. March 26: Implementation, Step #1

On March 26, Debbie finally managed to start implementing the House Scene. In her Designer's Notebook, she did not create any new plan, but referred her imaginary reader to her plans from the "four previous pages" of the Notebook. In her plans of that day, March

26, she also summarized the changes she had made on March 24: "I am making two extra wagons to it." The following is a replica of Debbie's planning Form of March 26 from her Designer's Notebook:


MY PLANS FOR TODAY

Today's Date: March 26, 87

I am REALLY starting my fraction scene, house. It is on the four previous pages. I made a little change in the planes. I am making two extra wagons to it. It is on the Plans Sheets before this.

Debbie started working on her House Scene on March 26. She worked in the Logo direct-mode for a while, drawing the house and roof in different colors and sizes, placing them at different points on the computer screen. She then checked the list of commands accumulated in the Logo Command Center, deleted several lines, changed a few inputs for colors or sizes, and used the appropriate function keys for copying what she wanted to put in her procedure (i.e., with the Function 1 and the arrows keys she marked what she wanted to copy, with the Function 2 key she cut the marked command lines, flipped the Page, and with the Function 4 key pasted the lines into the Editor). She then created a procedure by adding a procedure name: "TO HOUSE" at the top of the list, and the word "END" at the bottom.

"Let's see..." she said to herself while flipping the Page and typing HOUSE at the Command Center. The shapes looked rather disorganized on the screen, to say the least. Debbie spent a few seconds looking at the picture on the screen (probably analyzing it), and entered the Editor again. She used a rather sophisticated strategy for organizing her procedure and for detecting where the problems were: she deleted some interrelated FDs and BKs, and more important, she added "PU HOME" at the beginning of each "chunk". In other words, she divided the long procedure by typing "PU HOME" in six places: **1)** at the beginning of the whole procedure, before the roof and the house outlines were drawn; **2)** between drawing the house outline and drawing the line dividing the house into two halves; **3)** between dividing the roof into halves and shading the left half of the roof in orange; **4)** between shading the roof half in orange and drawing the door; **5)** between dividing the door into two halves and shading of the top door half in red; and **6)** at the end of the procedure. (Three other PU HOME were there already for moving the Turtle to desired spots on the screen. She did not change those).

The following is Debbie's procedure with my interpretations of the code (in brackets). I underlined the PU HOMEs Debbie added just before the end of that session to show the way she herself "chunked" her procedure into its meaningful units.

| | |
|---|---|
| **TO HOUSE** | **[First Step, March 26]** |
| pu home | |
| setc 4 | [sets the pen color into red] |
| pu setpos [-65 25] | [positions the Turtle at the top left corner of the house] |
| rt 45 | [the angle needed to draw the left side of the roof] |
| pd fd 100 | [draws the left side of the roof] |
| rt 90 fd 100 | [turns right and draws the right side of the roof] |
| rt 135 fd 140 lt 90 | [turns right and draws the bottom of the roof, which is also the top ofl the house] |
| setc 1 | [changes pen color into blue] |
| fd 100 | [draws the left side of the house] |
| setc 13 | [changes the pen color into pink] |
| lt 90 fd 140 | [turns left and draws the bottom line for the house] |
| lt 90 setc 10 | [turns left and changes the pen color into green] |
| fd 100 | [draws the right side of the house] |
| pu home | |
| pu setpos [10 0] | [moves the Turtle to one point on the middle line of the house] |
| setc 11 | [changes pen color into light blue] |
| pu bk 75 | [moves back to the bottom line of the house] |
| pd fd 100 | [draws a line that divides the house into two halves, in light blue] |
| pu home pd fill | [fills the left half of the house in light blue] |

```
pu setpos [10 25]     [moves to the middle of the top line of the house, or bottom of roof]
setc 12               [changes pen color into orange]
pd fd 68              [draws an orange-colored line that divides the roof into two halves]
  pu home
pu setpos [20 35]     [moves to the middle of the right half of the roof]
pd fill               [fills the right half of the roof in orange color]
  pu home
pu setpos [25 -75]    [moves to the bottom right of the house for creating the door]
setc 5                [changes the pen color into purple]
pd fd 60              [draws the left side of the door]
setc 9                [changes the pen color into blue]
rt 90 fd 30           [turns right and draws the top of the door]
rt 90
setc 4                [changes the pen color into red]
fd 60                 [draws the right side of the door]
  pu home
pu setpos [25 -40]    [moves to approximately the middle of the door]
pu bk 3               [corrects Turtle position in the middle of the left side of the door]
rt 90 pd fd 30        [turns right and draws a line that divides the door into two halves]
pu setpos [36 -33]    [moves the Turtle to the middle of the top door half]
pd fill               [fills the top door half in red]
  pu home
END
```

In fact, at the end of the computer time on March 26, I sat with Debbie and asked her what each of the lines in her Logo code meant. She observed how I wrote the code with her explanations on my piece of paper (she spoke slowly, as though aware that she was collaborating with me on my data collection). Her explanations were very precise and demonstrated her understanding of what the different commands in Logo were doing, as well as the relationships between them. On this occasion, I also asked her whether there was another way to draw the house, the roof, and the door, other than drawing them line by line.

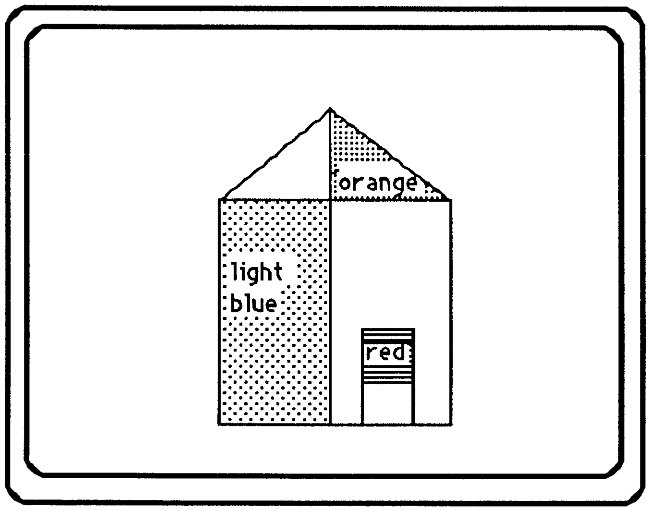
**Debbie:** "Yes. This is a rectangle (pointing at the house), a triangle (pointing at the roof), and another rectangle (pointing at the door). But I do it like this 'cause...you see...I changed the colors."

**I:** "What do you mean?"

**Debbie:** "Each line is in a different color. It's nicer. Red line, green, pink, blue line [pointing at the different lines on the computer screen]. I could not use Repeats here and all of that, because of these colors."

I find this an interesting point because an outsider could have entered into this particular situation, read Debbie's House procedure and said: "After all this time she invested in programming, this girl still does not know how to use REPEAT for creating simple rectangles or a triangle." Or "Look at her spaghetti code...with no sub-procedures, etc." But having worked with Debbie so intensely, and knowing her previous procedures, I was aware that by that time she knew how to use REPEAT and what using REPEAT meant in programming. Debbie <u>chose</u> not to use REPEAT in this design because she wanted to program each line in a different color. Her answer to my question (my question did not include the words "REPEAT" or "sub-procedures") also demonstrated that she clearly understood my question and its purpose, but "could not use REPEATs here and all of that, because of these colors."

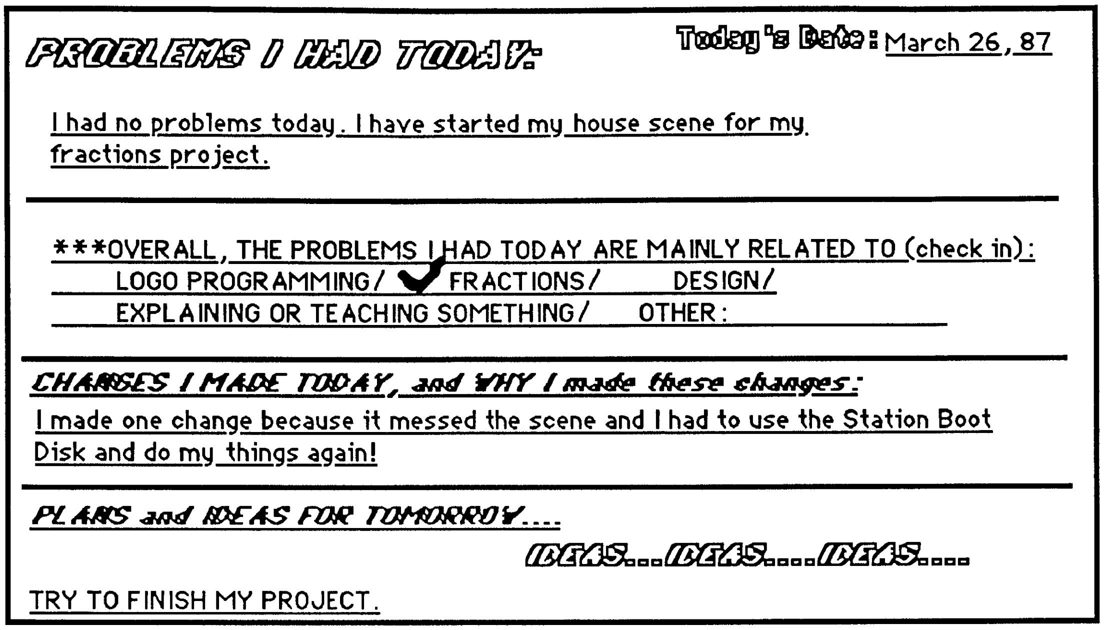
**At the end of computer time on March 26, Debbie's screen of the House Scene looked like this:**



This screen was very colorful (and cannot easily be depicted in the limited black and white medium that is now used for describing it): the roof's outline was red; the line that divided the roof into two halves and the roof's left half were orange; the left side of the house was blue; the right side of the house was green; the bottom side of the house was pink; the top side of the house was red; the line that divided the house into two halves and the shaded half on its left were light blue; the left side of the door was purple; the top side of the door was blue; the right side of the door was red; and the line dividing the door into two halves and the top door half were red as well.

What role did this process of playing with colors play in Debbie's learning? Debbie was really involved in a technical and mathematical project, and we can also recall that her teacher had originally described her as someone "not very interested in mathematics," and "quite low in math skills." However, in the context of this Project she did not seem to perceive her work as personally irrelevant because of its being mathematical and quite technical. Observing her planning, designing, drawing, and implementions during the period of the House Scene might, in fact, make us think that Debbie was not involved in a math project at all, but rather in a visual arts project. The computer and Logo offered Debbie something she could not refuse: dealing with artistic goals, with drawing, with colors, and with combining colors. It is another example for a child's exercising mathematical skills by using the technology for artistic creation. Turkle (1984), Weir (1987), and Papert (1986), among others, discuss these very issues quite often: how the fact of using a piece of advanced technology changes the child's relathionship with things so technical or mathematical. Turkle, in her research, concentrated on describing cases of children who progressed quite deeply into programming but with artistic motivations. These findings seem to be relevant to Debbie's case as well.

**Finally, at the end of the computer time on March 26, Debbie went to the classroom and wrote in her Designer's Notebook:**



PROBLEMS I HAD TODAY: Today's Date: March 26, 87

I had no problems today. I have started my house scene for my fractions project.

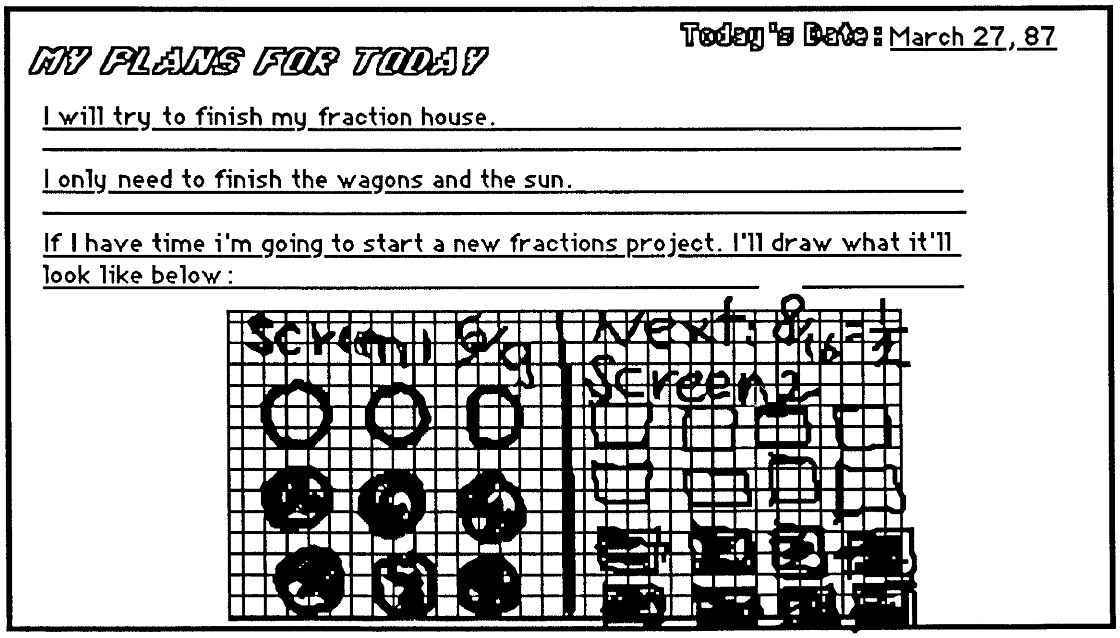
***OVERALL, THE PROBLEMS I HAD TODAY ARE MAINLY RELATED TO (check in):
LOGO PROGRAMMING/ [✓ FRACTIONS/] DESIGN/
EXPLAINING OR TEACHING SOMETHING/ OTHER:

CHANGES I MADE TODAY, and WHY I made these changes:
I made one change because it messed the scene and I had to use the Station Boot Disk and do my things again!

PLANS and IDEAS FOR TOMORROW....
IDEAS....IDEAS....IDEAS....

TRY TO FINISH MY PROJECT.

## 4.3.5. March 27: Implementation Step #2

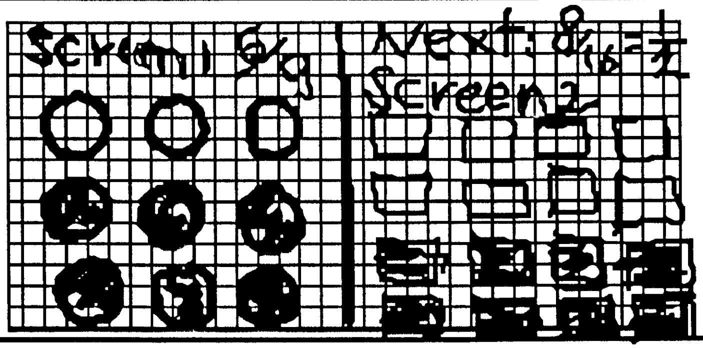


MY PLANS FOR TODAY Today's Date: March 27, 87

I will try to finish my fraction house.

I only need to finish the wagons and the sun.

If I have time i'm going to start a new fractions project. I'll draw what it'll look like below:



In the March 27 entry, we can read that Debbie attempted to finish up her House Scene and start work on the two new screens shown on the grid above. She divided the grid in her Designer's Notebook into two parts. Each part represented one screen. The first screen on the left was designed to show 6/9, while the second screen on the right was designed to show the equivalence of 8/16 = 1/2. However, she never managed to accomplish these two screens. When she logged into her computer, the first thing she did was run the procedure HOUSE she had created the day before. She smiled while looking at the computer screen. She ran the same procedure three more times while silently looking at her screen and mumbling something to herself. When I saw her doing that I approached her and asked her what were her plans for that day.

**Debbie:** "Did you see what I did yesterday?"

**I:** "Of course. I was sitting next to you, for some of the time, remember?"

**Debbie:** "Oh..Yea...But look..." [She pointed at her House picture on the screen].

**I:** "I love it. Great job Debbie! And what do you think you'll do now?"

**Debbie:** "Uhm...To change little things here and there and to finish it."

**I:** "Change what things?"

**Debbie:** [Instead of answering me, Debbie entered the Logo Page and deleted a few lines from her Logo code] "I don't need these now," she said.

**I:** "Don't need what?"

**Debbie:** [long pause, deleting something...typing something else]..."This."

Debbie deleted most of the PU HOMEs she had added the day before. She also deleted the spaces and "shrank" the procedure "so it would not take up too much space on the Page." The following is the result of her modification of the house procedure on March 27:

```
TO HOUSE        [Second Step, March 27]
pu home setc 4  pu setpos [-65 25] rt 45 pd fd 100 rt 90 fd 100 rt 135 fd 140 lt 90
setc 1 fd 100 setc 13 lt 90 fd 140 lt 90 setc 10 fd 100  pu setpos [10 0] setc 11
pu bk 75 pd fd 100 pu home pd fill pu setpos [10 25] setc 12 pd fd 68
pu setpos [20 35] pd fill pu setpos [25 -75] setc 5 pd fd 60 setc 9 rt 90 fd 30
rt 90  setc 4  fd 60 pu setpos [25 -40] pu bk 3 rt 90 pd fd 30 pu setpos [36 - 33] pd fill
pu home
END
```

After doing the above, Debbie typed "house" in the Command Center, and was satisfied to see that her changes did not ruin the procedure and that the executed picture remained the same. In direct mode she started programming one of the wooden wagons at the bottom left of the computer screen. Her finalized wagon consisted in the following code:

```
pu setpos [-150 -40]                 [moves the Turtle to the bottom left of the screen]
pd repeat 2 [fd 30 rt 90 fd 60 rt 90] [draws a rectangle, the body of the wagon]
pu home pu setpose [-155 -50]        [moves the Turtle to the place of the left wheel]
setc 1+ random 14                    [random selection for the wheel's color]
pd repeat 360 [fd .2 rt 1]           [draws the left wheel]
pu home pu setpos [-120 -40]         [moves the Turtle to the middle of the wagon]
setc 4 pd fd 30 pu rt 90 fd 20       [draws a red line that cuts the wagon into two halves,
rt 90 fd 10 pd fill                  and moves the Turtle to shade in the right half]
pu home
pu setpos [-145 -50] pd fill         [moves the Turtle into the left Wheel, shades it in]
pu home
setc1+random14
pu setpos [-90 -50]                  [moves the Turtle to the right side of the wagon]
pd repeat 360 [fd .2 lt 1]           [draws right wheel in a randomly selected color]
setc 4 lt 90 pu fd 10 pd fill pu home [shades in red the whole right wheel]
```

We can see that Debbie decided to shade in the wheels completely, each in a different color, instead of shading only half of each. On that issue she said: "It's easier [in terms of programming]. They are each [each wheel] shaded in a different color, so it's like a half too." In other words, due to programming constraints, Debbie decided, during the implementation phase, to change her original plans (of shading a half of each wheel), and instead, to shade each wheel in a different color. In this way, she considered the two wheels together to be a whole, so that each whole wheel, being in a different color, became the half of the whole.

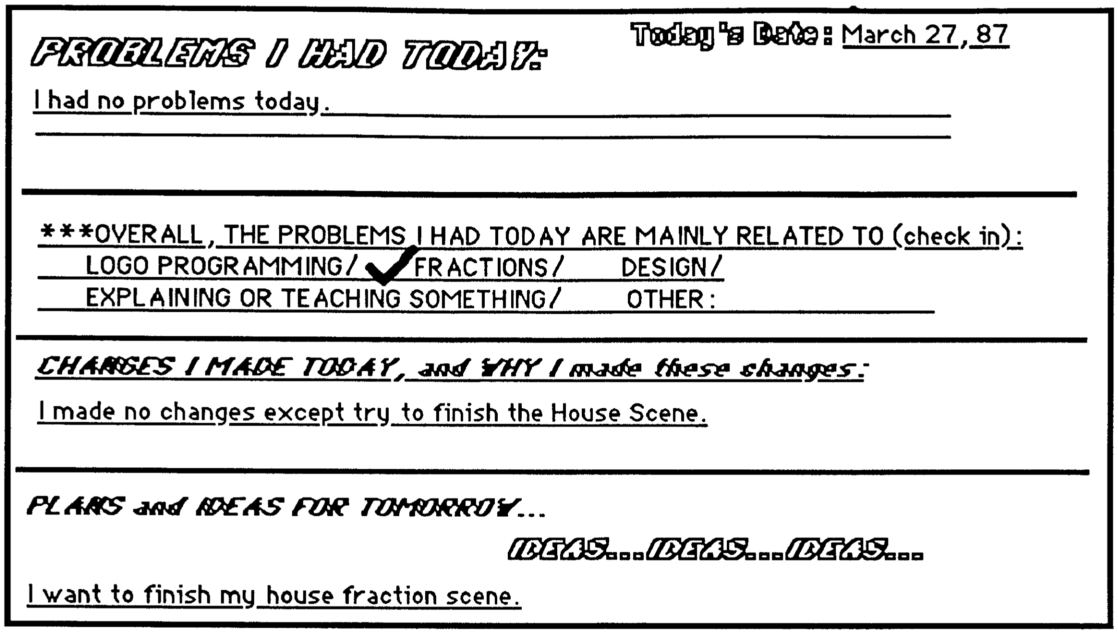
After working on this for a while, and after finalizing all the POSitions, REPEATs, SETColors, the sizes of the wagon and its wheels, Debbie tried her procedure line by line several times. Originally, she had written REPEAT **4** [fd 30 rt 90 fd 60 rt 90] as her commands for the rectangle that represented the body of the wagon. This is a typical mistake among young programmers, who often transfer their knowledge of how to do a square--Repeat **4** [fd 30 rt 90]--into their programming of a rectangle.

I do not think that Debbie shared this misconception, but was probably just careless in typing the commands. Also, her previous procedures and her Logo Post-Tests indicated that she understood that concept well. However, after the teacher commented on the fact that "REPEAT **2** would be a more accurate way of doing that," Debbie said, "I know..I know" and quickly changed the rectangle commands into REPEAT **2** [fd 30 rt 90 fd 60 rt 90]. We can also see how Debbie used SETPOSitions for almost all of the positionings of the Turtle on the screen. This time, she used the PU HOME strategy to place the Turtle at the center of the screen before each SETPOS, which helped her thinking about and figuring out the different x and y coordinates. In addition, she was accurate in her circles commands, which she could draw both clockwise and counterclockwise.

Finally, she copied the following lines of code from the Command Center, using the appropriate function keys, and pasted them in her "house" procedure:

```
TO HOUSE   [Third Step, March 27]
pu home setc 4  pu setpos [-65 25] rt 45 pd fd 100 rt 90 fd 100 rt 135 fd 140 lt 90
setc 1 fd 100 setc 13 lt 90 fd 140 lt 90 setc 10 fd 100  pu setpos [10 0] setc 11
pu bk 75 pd fd 100 pu home pd fill pu setpos [10 25] setc 12 pd fd 68
pu setpos [20 35] pd fill pu setpos [25 -75] setc 5 pd fd 60 setc 9 rt 90 fd 30
rt 90  setc 4  fd 60 pu setpos [25 -40] pu bk 3 rt 90 pd fd 30 pu setpos [36 - 33] pd fill
pu home pu setpos [-150 -40] pd repeat 2 [fd 30 rt 90 fd 60 rt 90] pu home
pu setpose [-155 -50] setc 1+ random 14  pd repeat 360 [fd .2 rt 1]
pu home pu setpos [-120 -40] setc 4 pd fd 30 pu rt 90 fd 20 rt 90 fd 10 pd fill
pu home pu setpos [-145 -50] pd fill pu home  setc 1 +random 14 pu setpos [-90 -50]
pd repeat 360 [fd .2 lt 1] setc 4 lt 90 pu fd 10 pd fill pu home
END
```

She ran the procedure to see if it worked, and it did. The computer time was over. That day, Debbie wrote in her Designer's Notebook:



PROBLEMS I HAD TODAY: Today's Date: March 27, 87

I had no problems today.

***OVERALL, THE PROBLEMS I HAD TODAY ARE MAINLY RELATED TO (check in):
LOGO PROGRAMMING/ [✓ FRACTIONS/] DESIGN/
EXPLAINING OR TEACHING SOMETHING/ OTHER:

CHANGES I MADE TODAY, and WHY I made these changes:

I made no changes except try to finish the House Scene.

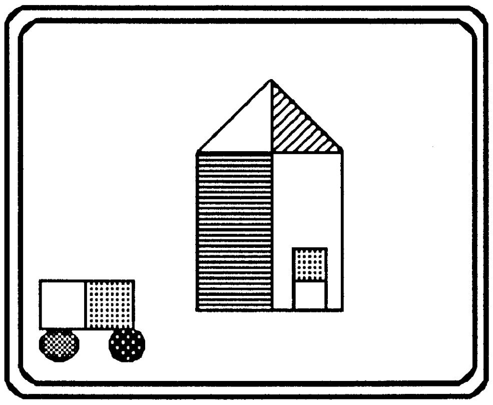
PLANS and IDEAS FOR TOMORROW...

IDEAS...IDEAS...IDEAS...

I want to finish my house fraction scene.

Debbie did not report on changing her original plans about the representations of halves using the wagon wheels (i.e., shading the whole wheel instead of a half of each wheel). She wrote: "I made no changes except try to finish the House Scene."

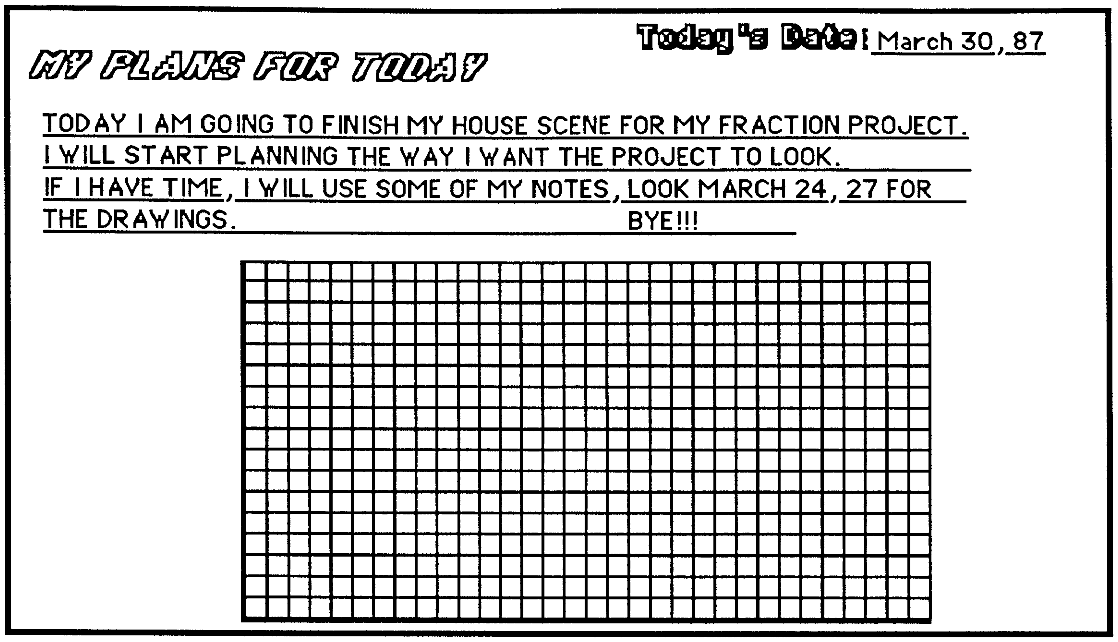
**The following diagram represents how Debbie's House Scene looked at the end of the computer time on March 27:**



**The House Scene on March 27**

**After Implementation Step #2**

### 4.3.6. March 30: Implementation Step #3



MY PLANS FOR TODAY

Today's Date: March 30, 87

TODAY I AM GOING TO FINISH MY HOUSE SCENE FOR MY FRACTION PROJECT.
I WILL START PLANNING THE WAY I WANT THE PROJECT TO LOOK.
IF I HAVE TIME, I WILL USE SOME OF MY NOTES, LOOK MARCH 24, 27 FOR
THE DRAWINGS. BYE!!!

On March 30, Debbie was planning to finish the House Scene. She also wrote about her attempts to start planning the way she wanted the project to look.

**I asked her**: "Which project, Debbie?"

**Debbie:** "My Fractions Project."

**I:** "And what do you mean by planning the way you want it to look?"

**Debbie:** "I have three things now. HALF, CALF, and HOUSE. I have to check how they look together."

I did not want to bother her anymore because she was very eager to finish the House Scene, and started to work on the second wagon while I was talking to her. But this was the first time Debbie had ever mentioned anything about planning how the project would look as a whole, and how her screens might look together. In fact, she dealt with this issue at the end of the following day's computer-time period, on March 31; then, she realized the need to make decisions about the order of her procedures (she had five of them by then) and finally created a super-procedure.

On March 30 Debbie also wrote: "If I have time, I will use some of my notes." By this I think she meant to use her notes from March 23, 24, and 27 that were related to the designs she planned to accomplish after the House Scene was completed. We can see that Debbie intended to make a full use of her Designer's Notebook, and indeed did use it to its maximum. She planned "new" designs while still involved in the implementation of "old" ones, and referred to previous plans and drawings several times during the process. She enjoyed planning screens in advance and did not seem to perceive this as a waste of time. On the contrary, she did it because she thought she would eventually go back to her plans when the time came to implement them.

While working at the computer, Debbie programmed another wooden wagon in a very similar way to her first wooden wagon, of March 27. She also succeeded in programming the sun, and adding an instructional sentence at the top of her screen. By the end of March 31, Debbie's House Procedure was the following (with my interpretations in the bold characters in brackets):

```
TO HOUSE   [Third and Final Step, March 30]
  [THE HOUSE, THE DOOR, AND THE ROOF:]
pu home setc 4  pu setpos [-65 25] rt 45 pd fd 100 rt 90 fd 100 rt 135 fd 140
lt 90 setc 1 fd 100 setc 13 lt 90 fd 140 lt 90 setc 10 fd 100  pu setpos [10 0] setc 11
pu bk 75 pd fd 100 pu home pd fill pu setpos [10 25] setc 12 pd fd 68
pu setpos [20 35] pd fill pu setpos [25 -75] setc 5 pd fd 60 setc 9 rt 90 fd 30
rt 90  setc 4  fd 60 pu setpos [25 -40] pu bk 3 rt 90 pd fd 30 pu setpos [36 - 33] pd fill
  [THE LEFT WOODEN WAGON:]
pu home pu setpos [-150 -40] pd repeat 2 [fd 30 rt 90 fd 60 rt 90] pu home
pu setpose [-155 -50] setc 1+ random 14  pd repeat 360 [fd .2 rt 1] pu home
pu setpos [-120 -40] setc 4 pd fd 30 pu rt 90 fd 20 rt 90 fd 10 pd fill
pu home pu setpos [-145 -50] pd fill pu home setc 1 +random 14
pu setpos [-90 -50] pd repeat 360 [fd .2 lt 1]  setc 4 lt 90 pu fd 10 pd fill
  [THE RIGHT WOODEN WAGON:]
 pu home pu setpos [85  -45] pd repeat 2 [fd 30 rt 90 fd 60 rt 90]
rt 90 fd 30 lt 90 fd 30 pu lt 135 fd 15 pd fill pu home
setc 1+random 14  pu setpos [80 -55] pd repeat 360 [fd.2 rt 1] setc 4 pu rt 90 fd 10
pd fill pu home pu setpos [155 -55] pd repeat 360 [fd .2 rt 1] setc 4 lt 90 pu fd 10 pd fill
 [THE  SUN:]
```

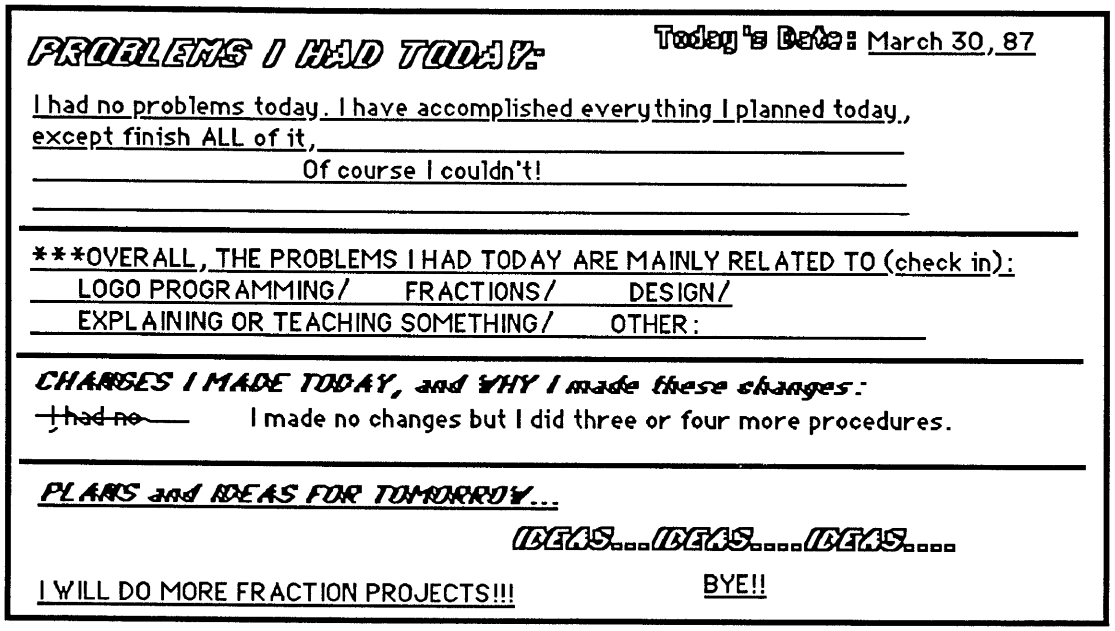
```
pu home  pu setpos [95 65] setc 14
pd repeat 360 [fd .3 rt 1] rt 90 pd fd 35 pu home pu setpos [115 75] pd fill
  [THE INSTRUCTIONAL SENTENCE:]
pr [This is a house. Almost every shape is 1/2!  I am trying to say, that you
use fractions, almost every  day of your life!]
END
```

We can see that Debbie changed her plans regarding the right wagon in the same way she changed her plans for the left one (i.e., shading each whole wheel in a different color instead of shading a half of each). The right wagon had the left half of its body shaded in, and the left wagon its right half. In other words, she did not change her plans to show halves on different sides of the objects.

In the final version of the House Scene, Debbie also changed the position of the sun so that it would be on the right side of the screen (it had been on the top left side of the screen throughout all of her plans). I assume that she changed the position of the sun because of the way Logo printed the instructional sentence. The Logo PRINT command prints whatever is in brackets at the top left of the screen, and Debbie probably did not want to manipulate the printing and move it to another spot on the screen. It was easier, in terms of programming, to change the SETPOS of the sun's position.

The instructional sentence she printed at the top of the screen, was also different from that of the original plan of March 23. On March 23 she planned to write: "This is a scene using regular human things. No matter what you use fractions on, you can use it on anything." However, in the final version of the House Scene of March 30, she decided instead to write: "This is a house. Almost every shape is 1/2! I am trying to say, that you use fractions, almost every day of your life!" I perceive the two versions of the instructional sentence as very similar in terms of their overall instructional message. I do not know why Debbie decided to choose the second instead of the original version. Perhaps she felt it would be more appropriate for her future third-grade users, or maybe that it better represented what she really wanted to teach by that screen. I was not able to gather information on this.

Debbie managed to run the finalized House Procedure twice before the computer time was over. She insisted that her teacher and Naomi, who sat next to her, look at it. Her teacher was very impressed and asked Debbie to print the screen (create a hard copy) so they could hang it on the class Logo-Ideas Board. Debbie also called her friend Cassania, who sat a bit further, and asked her to look at the House Scene as well. Cassania walked over to Debbie's computer with two other children from the class, a boy and a girl. They were all very impressed, crying: "Debbie this is cool!" "Debbie this is the freshest!" "Neat, Debbie, neat!" On March 30, at the end of the computer time, Debbie returned to her classroom and wrote in her Notebook:



PROBLEMS I HAD TODAY: Today's Date: March 30, 87

I had no problems today. I have accomplished everything I planned today, except finish ALL of it,

Of course I couldn't!

***OVERALL, THE PROBLEMS I HAD TODAY ARE MAINLY RELATED TO (check in):
LOGO PROGRAMMING/ FRACTIONS/ DESIGN/
EXPLAINING OR TEACHING SOMETHING/ OTHER:

CHANGES I MADE TODAY, and WHY I made these changes:
~~I had no~~ I made no changes but I did three or four more procedures.

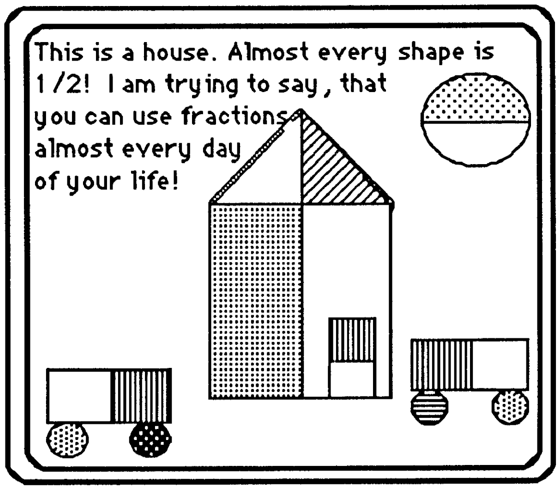
PLANS and IDEAS FOR TOMORROW...
IDEAS....IDEAS....IDEAS....

I WILL DO MORE FRACTION PROJECTS!!! BYE!!

Debbie wrote that she had finished the House Scene, but had not accomplished all of it. I think she was referring to her plans of March 24, and 27 to create four wagons in the scene. Apparently Debbie was aware of her misimplementation of part of her original plans (i.e., her having decided not to create four wagons, only two: one on the right side of the house, the other on the left). I never asked her why she had made that decision. I assume it was because she really wanted to finish the House Scene, so as to go on with the rest of the Project and start working on new screens.

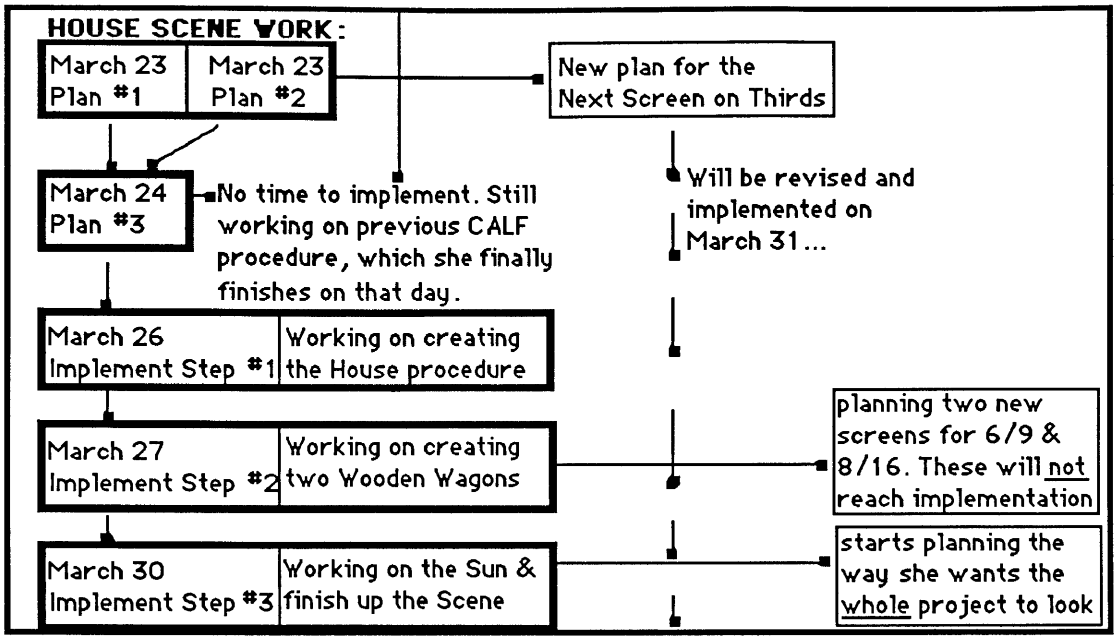
Debbie also wrote (in the above Reflections Form): "I made no changes but I did three or four more procedures" [my emphasis]. What did she mean by this? One possible interpretation is that she did not understand the meaning of the word procedure, or that she misused it in her writing. However, knowing Debbie's knowledge of programming and her previous uses of procedures and sub-procedures, I suggest interpreting her words in another way. I believe that although she programmed the whole scene in the one long procedure that she named "house," in her mind she viewed each of its parts as a separate unit. In other words, on that day she added the right wagon procedure (one part), the sun procedure (another part), and the instructional sentence procedure (a third part). To my mind, Debbie's written sentence above (i.e. "I did three or four more procedures") represents her modular way of thinking. Actually, she did not use sub-procedures for each of the objects in the House Scene, but she definitely thought of each object as a separate part or, in her words, a procedure.

**At the end of the computer time on March 30, Debbie's House Scene was completed, and looked like this:**


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!

**The House Scene, Final Version March 30**

As a summary, let us examine again the general structure of her processes of designing and constructing this Scene. It is flowcharted in the following diagram:

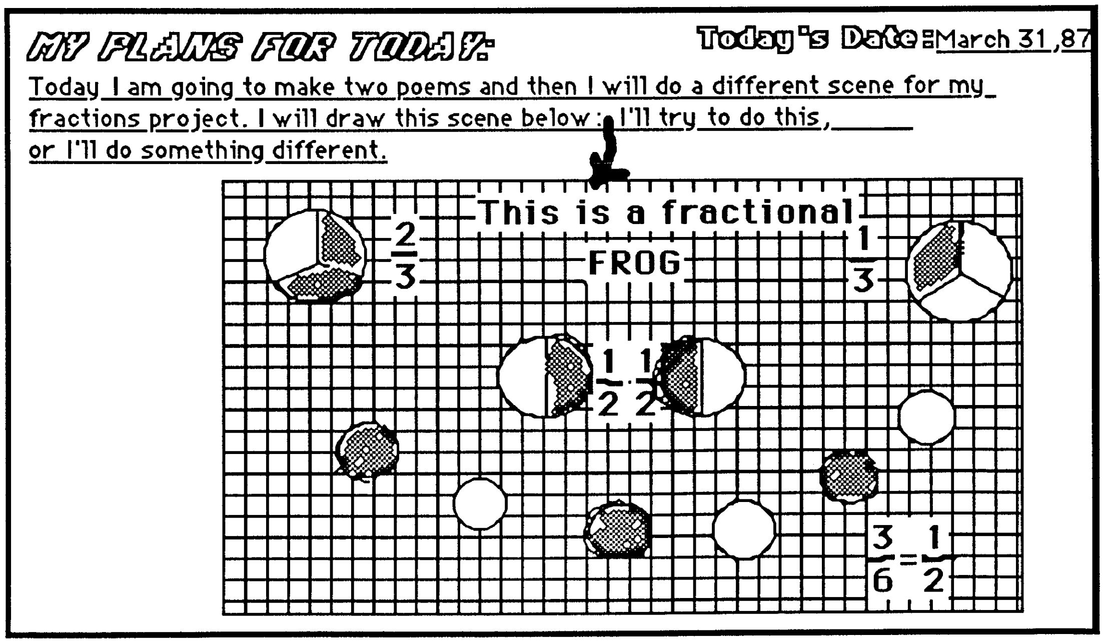

HOUSE SCENE WORK:
March 23 Plan #1
March 23 Plan #2
New plan for the Next Screen on Thirds
March 24 Plan #3
No time to implement. Still working on previous CALF procedure, which she finally finishes on that day.
Will be revised and implemented on March 31...
March 26 Implement Step #1
Working on creating the House procedure
March 27 Implement Step #2
Working on creating two Wooden Wagons
planning two new screens for 6/9 & 8/16. These will not reach implementation
March 30 Implement Step #3
Working on the Sun & finish up the Scene
starts planning the way she wants the whole project to look

## 5. TWO-THIRDS: "ONE OF THE FRACTIONS TEACHERS USE MOST OFTEN AS EXAMPLES FOR TEACHING THEIR STUDENTS"

Debbie's goal after she finishing her "House Scene," was to create a representation for two-thirds. We have already seen that on March 23, while creating the plan for the "House Scene," she was also planning one step ahead and thinking about the screen she would do after the "House Scene" was completed. These were her notes from March 23:

"I think I have to finish my old fractions project [the third screen she had created, with the 4 shapes divided into halves, see Section 3 above]. I am going to make a scene, where every shape is shaded half. When I'm finish **I'm going to do 2/3 for my project & so on**" [my emphasis on her plan to work on the 2/3 screen, see the process diagram in Section 4.3].

On March 31, when her "House Scene" was finally completed, she remembered her old plan to create a representation for thirds. On that day (March 31), she drew in her Designer's Notebook the following plan:


MY PLANS FOR TODAY: Today's Date: March 31, 87

Today I am going to make two poems and then I will do a different scene for my fractions project. I will draw this scene below: I'll try to do this, or I'll do something different.

Debbie called this design "A FRACTIONAL FROG." According to her, she did not plan in advance to draw a frog, but it looked like one after she finished drawing it, so she named it this way. Also, Debbie emphasized the idea of shading the "different sides" of the halves or thirds (for reasons described in Sections 3 and 4), showing that, in her words, "it doesn't matter which side of the shape is shaded in--all the parts, whether they are shaded or not, can be fractions."

The structure of Debbie's design of March 31 can be considered in three parts. **Part one** is the two top circles divided into thirds, the right circle showing 1/3, the left circle 2/3 (Debbie thought these two circles were the Frog's ears). **Part two** is the two middle circles showing halves (i.e., the Frog's eyes). In the right circle (i.e., the "right eye") the left half is shaded, and in the left circle (i.e., the "left eye"), the right half. **Part three** is what she called "the Frog's necklace," the six small circles at the bottom, representing 3/6 = 1/2.

I find this design interesting for several reasons. "Part one" and "part two" are still in the mode of "showing fractions," whereas "part three" is Debbie's first step towards representing <u>equivalence.</u> This screen's design is based on the idea of transitions between representations of fractions. She "connected" the concept of thirds with the concept of sixths through her representation for halves (i.e., the Frog's eyes). In this design she did not explicitly connect the thirds with the sixths (i.e., she did not show that 2/6 = 1/3). But my interview with her, on that day, indicated that she definitely connected these thirds and sixths in her mind:

**I:** "Why did you put all these things in one design?"

**Debbie:** "...Because sixths can be thirds...and sixths can be halves..."

**I:** "What do you mean?"

**Debbie:** "You can <u>put</u> two sixths <u>on</u> one third." [my emphasis]

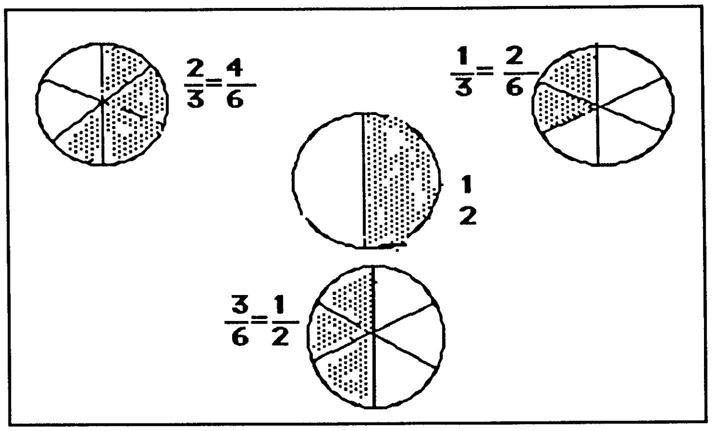
**I:** "Why did you choose not to show that here?"

**Debbie:** "..'cause I wanted to show something else here. That you can <u>put</u> three-sixths <u>on top of</u> one-half too. Everybody knows that 2/6 = 1/3" [my emphasis].

[Debbie's notion that one could <u>put</u> fractions <u>on top of</u> something, has already been discussed: when Debbie and I talked about her "House Scene," she said that "you can put fractions on anything"].

I assume that Debbie had decided not to show, in this particular design, that 2/6 = 1/3, since "everybody knows that" (in other words, <u>she</u> knew it). By that time, the equivalence between thirds and sixths existed in her mind, and she did not have problems responding to questions I asked her about this equivalence. She probably thought that the fact that "two-sixths equal one-third" was easy to understand, although the concept that three of those sixths could also equal one-half is more difficult to grasp.

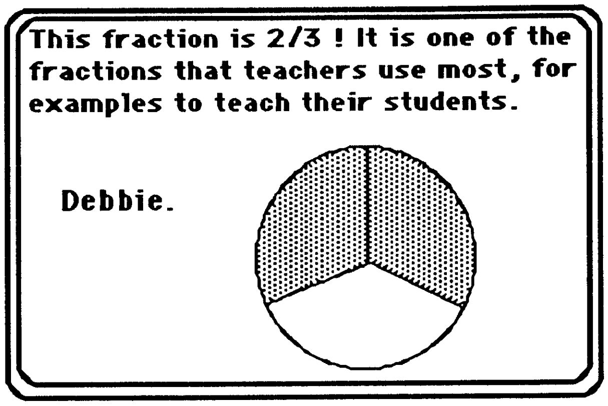
The sizes of the different fraction representations were not consistent in her design. Considering what she wanted to teach, it would have been better to represent the thirds, halves, and sixths on the same scale. The picture shown below summarizes what Debbie really wanted to teach. I drew it according to her explanations in her interview with me:



$\frac{2}{3}=\frac{4}{6}$
$\frac{1}{3}=\frac{2}{6}$
$\frac{1}{2}$
$\frac{3}{6}=\frac{1}{2}$

**This is <u>my interpretation</u> of what Debbie really wanted to teach via her plan of March 31.**

Debbie, however, did not implement the whole "Fractional Frog" design of March 31. Instead, she implemented <u>one part</u> of this design, the Frog's right ear, for reasons unknown to me. Using Logo, she programmed one big circle divided into thirds and shaded two of those thirds. **Here is what Debbie implemented on the <u>computer</u> on March 31:**


This fraction is 2/3 ! It is one of the fractions that teachers use most, for examples to teach their students.

Debbie.

The "instructional sentence" Debbie wrote on that screen suggests her awareness of three things: **1)** thirds are generally more difficult fractions; **2)** since thirds are difficult, teachers use them most often as examples to teach their students; and **3)** now that she herself is teaching, she is using thirds like all other teachers. The following conversation records the first time I explicitly saw Debbie perceiving herself as a teacher:

**I:** "Why do you tell your users that teachers use this fraction most?"

**Debbie:** "Um...'Cause it's a hard one. And I'm the teacher here. So I use it. [Long pause. She keeps on typing]...I think I'll make a test for it too."

**I:** "A test?"

**Debbie:** "Yes. You'll see it later. When I'm done. [Long pause. She continues to talk to me while typing] I'll show it [the representation of thirds] to them [to the users], and then, I'll take numbers [2/3, and other numbers], and ask them, later, to answer what fraction it [the picture] is."

Debbie generated a new plan while she was working on the computer and programming the representation for two-thirds. It was the first time she had decided to use a picture (a fractional representation) in two different contexts. She planned to use it once as an example, accompanied by one specific text, then in a testing procedure, with a different text. In order to program the "testing procedure," she went through three steps of learning new Logo programming concepts. This process is described in the following Section **5.2.**, but before describing how Debbie actually programmed her screens (i.e., the example and the testing), I shall first describe in Section **5.1.** what it means for a child--in terms of Logo programming and instructional design--to write an interactive quizzing or testing procedure.

## 5.1. The Programming ideas behind Logo Quizzing Procedures

On March 27, according to the pupils' needs and requests, we conducted a ten-minute classroom discussion on creating interactive screens (i.e., quizzing procedures). In this short discussion, the teacher showed the children how to create a procedure that asked the user a multiple-choice question, took the user's input (i.e., answer), and gave the user feedback for his right answer, or another for his wrong answer. Different children suggested different approaches for questioning and feedbacks; they expressed their ideas and feelings about what they considered "appropriate" ways for asking questions--giving hints, providing the user with an option to "try again," when to "allow" the computer to give the right answer, etc. During those ten minutes, the children were learning and discussing two major issues simultaneously: **1)** Logo programming concepts, such as how to use conditionals, conditionals evaluation (i.e., the Logo possibility to choose between two expressions to evaluate), using IFELSE as an operation, how to create a space in the computer memory for Variables (i.e., for accepting the user's INPUTS), how to use READLIST or READCHAR, how to use recursion, and how to format all the above in Logo (i.e., where and why to use brackets, parentheses, quotation marks, or colons); **2)** instructional design concepts, such as how to ask questions, what pictures to use if at all, what explanations or hints to include, what feedbacks to give, the "user-friendliness" of their programs, etc.

Let us focus for a moment on the Logo programming concepts included in a basic "testing procedure" similar to the one most of the Experimental children used in their Fractions Programs.

In general, when IFELSE is used as an operation, it requires three inputs. The first input must include either a TRUE or a FALSE expression, the second and third inputs must be lists containing Logo expressions, instructions, and commands. If the first input's value is true, then the output of IFELSE is a result of evaluating the second input; if the first input's value is false, then the output of IFELSE is a result of evaluating the third input. As an example, let us follow this simple procedure "TO QUIZ":

```
TO QUIZ
PR [2V3 + 2V6 = ?   Please choose one answer: 3V6, 4V9, 1, 4V18]
                        [Logo prints the above multiple-choice question on the screen]
NAME READLIST "USER.ANS
                        [the user types an answer which Logo "reads" and saves in a special memory
                        slot called "USER.ANS". The programmer has specified the slot's name]
IFELSE (:USER.ANS=[1])
                        [Logo compares the value of USER.ANS with 1, if the user's answer is "1";
                        then Logo executes the second input which is the next line of code,]
[PR [That's right! Greate job! 2V3 + 2V6 = 6V6 =3V3 = 1]]
```

[This is what Logo prints (or does) if the the value of USER.ANS = 1 is TRUE]

**[PR [Well, I guess you must try that one again. Think about how many sixths equals 2\3, or how many thirds equals 2\6...Try again...] CT QUIZ]**

[This is what Logo executes as the feedback, if the value of USER. ANS = 1 is FALSE; it prints a statement, clears the text on the screen, and calls the procedure QUIZ (itself) again]

**END**

In this model procedure, Logo first printed a question, then "read" the user's input, saved it, and later evaluated the user's answer; if the user's answer was, for example, "1," Logo printed on the screen the feedback for the "true" value (that is: "That's right! Great job!"). However, if the user's answer was something other than "1," then Logo printed the second feedback, cleared the text on the screen (CT) and called the procedure again (QUIZ).

This short procedure, in fact, includes within it several complex programming concepts and among them is the idea of *recursion*. A recursive procedure is a procedure that calls itself as a sub-procedure. In general, "Recursion is an idea that is very simple once you understand it, but which many people have trouble learning about" (Harvey, 1985 p. 70; see also Kurland and Pea, 1985, "Children's Mental Models of Recursive Logo Programs"). There are several different ways to approach recursion (e,g., Harvey 1985, pp. 54-120, pp. 142-160). In the above procedure, for example, I included the recursive call as the last instruction. This is called a *"tail recursion."* There is a lot to say about the meaning of *tail recursion* and other, more complex types of recursion. Several researchers have studied children's and adults' misconceptions and understanding of the different types of recursion; however, it is beyond the scope of this section to review the typical problems children and adults had with the idea of recursion and with recursive procedures. Debbie did not demonstrate any misconceptions regarding *tail recursion* (the only type of recursion she knew and used), and we will see later that she used it quite often in her Fractions Project.

## 5.2. Programming The Representation For Two-Thirds

Debbie's code for the large circle divided into two-thirds was as follows:

```
TO TWO
PU HOME
SETC 1+RANDOM 14                 [the computer randomly selects a pen color]
PD REPEAT 360 [FD 1 RT 1]        [creates a big circle]
PU SETPOS [60 5]                 [positions the Turtle in the middle of the circle]
SETC 1+RANDOM 14
PD FD 53                         [draws lines for dividing the circle into thirds]
PU BK 53
RT 135 PD FD 57
PU BK 57 RT 90
PD FD 62
PU BK 62
RT 45
PU FD 20
SETC 1+RANDOM 14
PD FILL                          [shades in one third]
PU BK 40
PD FILL                          [shades in the second third]
PU HOME
WAIT 100 PR [THIS FRACTION IS 2\/3 ! IT IS ONE OF THE FRACTIONS
TEACHERS USE MOST FOR EXAMPLES TO
TEACH THEIR STUDENTS.]           [prints the instructional statement]
END
```

This was Debbie's first step in programming her representation of two-thirds on March 31. It took her 25 minutes to write this procedure, try it out, and fix it. She then decided to create her testing procedure, which she called **QUESTION1**. Using her notes from the classroom discussion (i.e., of March 27), and with my help, she flipped into the

Logo editor and wrote:

```
TO  QUESTION1
PR [IS THIS FRACTION, 1\/2, 3\/4, 2\/3, OR 1\/26 ?]
NAME READLIST "ANSWER
IFELSE (ANSWER =[2\/3])
[PR[GREAT!!]] [PR[PLEASE TRY AGAIN] CT WAIT 30 QUESTION1]
END
```

This procedure prints a question at the top of the screen, asking the user: "Is this fraction 1/2, 3/4, 2/3, or 1/26?"; the user types in an answer; Logo then "reads" the user's input (i.e., his answer), and saves it in a memory slot, which Debbie named for that purpose, "answer." If the user's answer is "2/3," Logo prints "GREAT!" and the procedure is over (or done). If the user's answer is different from "2/3," Logo prints "Please try again," remains within the same procedure but calls it again (i.e., prints the question all over again). Debbie meant to attach this procedure to her TWO procedure; the following paragraphs describe how she did it.

As soon as Debbie finished typing the TO QUESTION1 procedure (with my help), and before she even executed it, she entered her TO TWO procedure, ran the cursor down to the bottom of that procedure, and added another line of code before the END line (she did this without my help--her actions were videotaped, for I was not sitting next to her at that moment). I underlined and *italicized* the line she added:

```
TO TWO
•
•
.                                          [she did not change anything here]
WAIT 100 PR [THIS FRACTION IS 2\/3!  IT IS THE FRACTION TEACHERS USE MOST,
FOR EXAMPLES TO TEACH THEIR STUDENTS.] WAIT 150
CT QUESTION1                     [she added a "WAIT 150," left the picture on, cleared
END                              the text, and called her testing procedure QUESTION1]
```

Only after she had completed her changes in the TO TWO procedure did Debbie execute her modified procedure. She called me and wanted me to see it running. She was thrilled to find that it worked the way she wanted it to: "This is exactly what I wanted," she said. She evaluated it by asking Naomi, who was sitting beside her, to try it. It worked for Naomi as well.

### 5.3. Debbie's Poems

On March 31, Debbie also wrote in her Designer's Notebook about her plans to write two poems. At this point, having finished the programming of the procedures that showed two-thirds and tested the user about it, Debbie stopped working on her program for 10 minutes; instead, she left her Fractions Program, opened a new Logo Page, and wrote two very personal poems. She told me that day: "Logo is good for many things, even for animated poems about myself." These two poems, their content, procedures, animation, and meaning, have already been discussed in my introduction to Debbie's Case, therefore need not be repeated here. I refer the reader back to the section called "Who Is Debbie?" in the introduction to this Chapter.

## 6. THE FIRST ATTEMPT TO CONNECT THE PARTS: CREATING A SUPER-PROCEDURE

Towards the end of the "computer time" on March 31, Debbie left her poems aside and went back to work on her Project. Debbie had five separate procedures in her Fractions Project. The order in which she created them was the following:

1) **"TO HALF,"** designed and programmed on March 9, 10, and 11 and completed on March 16.
2) **"TO CALF,"** designed and re-designed from March 19 to March 23, and completed on March 24.
3) **"TO HOUSE,"** designed and worked on from March 23 to March 30.
4) **"TO TWO,"** designed and programmed on March 31.
5) **"TO QUESTION1,"** also completed on March 31.

In order to execute and check how all the procedures looked together, Debbie had to type, in the Command-Center, one procedure-name after the other, but only one at a time. On this particular day, she did not yet have any super-procedure that called for all the procedures to appear one after the other. As we see, on March 31 Debbie's project started to be, in her words, "really long." It was the first day in which her project became really meaningful to her as an entity. It took Debbie <u>one full month</u> to realize the need for a super-procedure.

Researchers in other studies (e.g., Pea & Kurland, 1983; Heller, 1986; Carver, 1986; Perkins et al., 1985; and others) have reported on children's writing non-modular codes (i.e., the spaghetti-code phenomenon). They found that children did not <u>understand</u> the meaning or purpose of a super-procedure, and rarely <u>used</u> super-procedures or sub-procedures in their programs unless they were very explicitly instructed to do so, such as is the case in Carver's (1987) study.

For this particular case, I offer Debbie's programming process as a demonstration of how important it is to let children program more extensively and intensively before one conducts research about their understanding and use of modularity, sub-procedures, and super-procedures in their programs. As is stated in the literature, children with minimal exposure to Logo, or with minimal explicit instruction in it, do not show much learning, understanding, or knowledge of how to use super-procedures and sub-procedures.

All the children in the Instructional Software Design Project, much like Debbie, created super-procedures and sub-procedures at one point or another during their process of creation. As we saw, Debbie, for example, created five sub-parts before she decided to create a super-procedure. I suggest that the structure of the Software Design Project affected in some ways her process of modular programming. Debbie first created and thought of each screen or representation as an independent part, with a specific name and role. Only after discovering that several procedures existed and worked, did Debbie realize that she had "enough parts" to bring them together into a whole--her super-procedure.

On March 31, towards the end of the computer session, I saw Debbie creating a super-procedure for the first time. I asked her why she had created one at all, and why then.

> **Debbie:** "Now I have enough things [parts] to put together, HALF, CALF, TWO, QUESTION1, and HOUSE. I am tired of typing these [procedure-names] all the time."

I interpret her two reasons for creating a super-procedure, which she named TO FRACTIONS, in the following way: 1) conceptual--"I have enough things to put together" into a whole, or in other words, enough things she could consider and call a whole; and 2) practical--"I'm tired of typing" these procedure names over and over again. It seems as though she learned about the purpose and meaning of a super-procedure through experience. In short, I think it is necessary to give children enough time and experience so that they will have opportunities to learn, understand, and use certain powerful programming ideas such as modularity or subparts-superparts relations.

Debbie worked on her super-procedure in three steps. Connecting the parts was not a very easy job for her. I observed how long it took her to work out the problem of how to connect the parts, and in what order. First, she wrote in the Logo editor the following super-procedure:

```
TO FRACTIONS [the first step]
HALF
CG CT CC
CALF
CG CT CC
TWO          [which includes QUESTION1]
CG CT CC
HOUSE
CG CT CC
END
```

### 6.1. Changing the order of the parts

Although Debbie created the procedure TO TWO after the procedure TO HOUSE, she decided to put TWO before the HOUSE in her super-procedure. I asked her why. Debbie answered: "Because it's [the procedure TWO] easier than the house scene [the procedure HOUSE], and also the house scene is nicer." She considered the two-thirds representation to be less complex, and not as "nice" as the "House Scene." So she did not attach much importance to chronological order in her creation of these procedures. Instead, she re-evaluated the design features of the procedures and paid attention to issues such as the complexity of the representations and their appearance.

## 6.2. Connecting the sub-parts while thinking about the target users

When Debbie executed the FRACTIONS super-procedure for the first time, the different screens were switching from one to another with too much speed. This happened because of the CC (Clear the Command Center), CG (Clear the Graphics), and CT (Clear the Text) commands she wrote after each of the procedures. In order to solve this problem, she first tried to eliminate the CC CT CG lines, but this did not work the way she wanted it too: "Oh! No!" she said. The screens were drawn one on top of the other, and the Turtle was not necessarily at the HOME position at the beginning of each procedure, which resulted in chaos. Her second step towards a solution was to add a series of WAIT commands after each procedure, before the CC CG CT line.

```
TO FRACTIONS [the second step, adding WAIT 150 between the picture and its clearing]
HALF WAIT 150
CG CT CC
CALF WAIT 150
CG CT CC
TWO   WAIT 150
CG CT CC
HOUSE WAIT 400
END
```

Other children in the class had similar problems while working at connecting the subparts into a whole (super-procedure). Debbie solved this problem by adding WAIT between screens before clearing them, as is shown in the above Logo code.

A few days before, another child, called Don, had asked me how to write a program that instructed the user to "press any key to go on." He told me that "The same WAIT 100 is not good for all the children," meaning, that it might be "too slow for some children or too fast for others." Don wanted to give more control to the users, allowing them to advance in his program at their own pace. He told me that he had seen "that trick in one of

Harry's programs." (i.e., Harry Nelson, a member of the MIT research and support staff). It was not clear to me what Don meant, and I asked Harry about it.

As a result, on March 31, Harry Nelson came into the classroom, where the children, their teacher, and I were gathered; he remained there for 10 minutes, and showed the children how to create a procedure that presented something on the computer screen and instructed the user to press any key to go on to the next screen. Whatever key the user pressed, the program continued. Harry's procedure looked as follows:

```
TO  WAIT.FOR.USER
TYPE [PRESS ANY KEY TO CONTINUE]  [types a message at the bottom of the screen]
IGNORE READCHAR                   ["sends" the character that was typed by the user
 END                               as the input for the IGNORE procedure]

TO IGNORE :KEY                    [the procedure IGNORE takes one input, a character,
END                                but does nothing with it, so the program goes on]
```

At the end of the day, after Harry's demonstration, Debbie rushed to her computer and added the new procedure to her project. All the children learned about "WAIT.FOR.USER," but Debbie was one of the first children to implement it in her super-procedure.

### 6.3. A Summary of Debbie's Activities on March 31

Debbie accomplished a great deal on March 31. Her processes of design and programming on that day demonstrate both a clear case for "cultural learning," and for "individualist constructivist learning." In other words, we can see how her own innovative activities (e.g., her creating a screen for thirds, as well as the instructional sentence, or her connecting the five procedures in a super-procedure) were interlaced and influenced by what was happening within her "culture" at that time (e.g., her learning about "quizzing procedures" in Logo, or about the "Wait.For.User trick").

The diagram in the following page attempts to capture the sequences and complexity of Debbie's activities on March 31 and the history of her planning and designing, various kinds of learning, and implementation phases.

**A trace of Debbie's planning, programming, and learning on March 31 :**

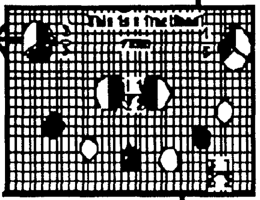
working on the House Scene

**March 23** ..."When I'm finish I'm going to do 2/3 for my Project..."
**March 24** ....
**March 26** ...
**March 27** .....
**March 30** ..."I will start planning the way I want the Project to look....."

**March 31 :**

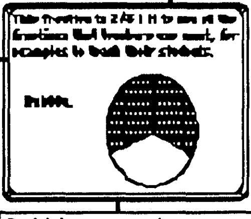
When Debbie finished her "House Scene" she remembered her general plan to do something on thirds. On March 31 she decided to create a design for this plan in her Notebook:

Plan, March 31 :



Debbie implements only a small part of her original design

Implementation, March 31 : Creating the procedure "TWO."



While she implements "TWO" Debbie composes an "Instructional Sentence" which captures the instructional purpose of her screen.

Creating the procedure "QUESTION1"

Debbie generates a new plan while she works on her screen for thirds: She decides to use the picture in a "testing procedure" which she creates and calls "Question1."

When Debbies finishes her procedures TWO and QUESTION1, she remembers her plan to think on the "way I want the project to look."
She has five separate parts: HALF, CALF, HOUSE, TWO, and QUESTION1, which she decides to connect together in a super-procedure.

Debbie learns about the Wait.For.User "trick."
She implements it right away in her super-procedure.

```
TO FRACTIONS
HALF Wait.For.User CG CT CC
CALF Wait.For.User CG CT CC
TWO Wait.For.User CG CT CC
HOUSE Wait.For.User CG CT CC
END
```

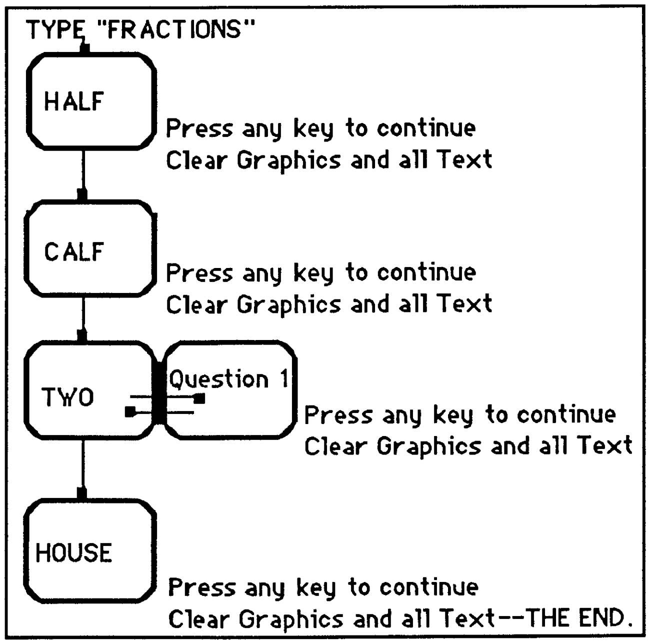
At the end of March 31, Debbie creates this super-procedure.
She places TWO before HOUSE, because "Two is easier than House, and House is nicer than Two."

<u>**At the end of the day (March 31) Debbie's program looked as follows:**</u>

**TO FRACTIONS** [the third step, adding the WAIT.FOR.USER sub-procedure]

```
HALF   WAIT.FOR.USER
CG CT CC
CALF   WAIT.FOR.USER
CG CT CC
TWO    WAIT.FOR.USER
CG CT CC
HOUSE  WAIT.FOR.USER
CG CT CC
END
TO  WAIT.FOR.USER
TYPE  [PRESS ANY KEY TO GO ON]
IGNORE READCHAR
END
TO IGNORE : KEY
END
```

**The structure of Debbie's software by the end of the computer time on March 31:**


TYPE "FRACTIONS"
HALF
Press any key to continue
Clear Graphics and all Text
CALF
Press any key to continue
Clear Graphics and all Text
TWO
Question 1
Press any key to continue
Clear Graphics and all Text
HOUSE
Press any key to continue
Clear Graphics and all Text--THE END.

## 7. CREATING THE SOFTWARE'S OPENING SCREENS: "QUEST," "FRONT," AND "INTRUDUCTION."

So far we have seen how throughout the month of March Debbie created five different screens: the first HALF screen--a large square filling the whole screen, with one big half (composed of two-fourths) shaded in on the right side of it, representing 1/2=2/4 ( this Chapter, Section 1); the geometrical CALF screen--four different shapes, a rectangle, a square, a circle, and a triangle, each divided into halves, teaching that "all these shapes are 1/2!" (this Chapter, Section 3); the thematic HOUSE screen--a house, its roof, and its door, each divided into halves, two wagons and their wheels divided into halves, and a sun divided into halves, teaching that "fractions are all around you" and that "you can use fractions almost every day of your life" (this Chapter, Section 4); the TWO screen--one big circle divided into thirds, with two of the thirds shaded in, explaining that thirds are difficult fractions "that teachers use most, for examples to teach their students"; and the QUESTION1 screen, which used the latter two-thirds representation and asked the user which fraction was represented in the picture, teaching him how to translate a pictorial representation of thirds into a symbolic representation of thirds (this Chapter, Section 5). On March 31, Debbie connected her different screens and created a super-procedure which she named FRACTIONS, so that a software user could type the word "fractions" in the Logo Command Center and run the software (this Chapter, Section 6).

At this point, Debbie realized the need for an opening screen. Debbie's various steps in planning, designing, implementing, and modifying the introduction to her software are presented in the following subsections. To summarize briefly, Debbie, on April 1, constructed her procedure QUEST as her original opening screen. Her instructional design approach is presented in the context of those of her peers'. Influenced by her peers, Debbie, on April 6, designed in her Notebook another introductory screen, which she called FRONT. During its implementation phase Debbie decided not to finish this particular design; instead, she went back to her Quest procedure and modified it. In addition, at the end of the computer time on April 6, she created yet another new procedure for her introduction which she named INTRODUCTION.

In Section **7.1.** I describe Debbie's implementation of QUEST and analyze how it relates to what two other children created for their opening screens. In Section **7.2.** I describe Debbie's design of the FRONT, which was strongly influenced by one of her friends' designs for the opening screen; in this Section I also explain Debbie's decision not to finish this particular design. In Section **7.3.** I describe Debbie's new INTRODUCTION procedure, and show how she went back to her procedure QUEST and modified it. On April 1 Debbie added her procedure QUEST to her super-procedure. However, on April 6, she replaced QUEST with her new INTRODUCTION procedure. She then created another super-procedure which she named BIG and put QUEST within it; her reasons for doing this are also described in this Section.

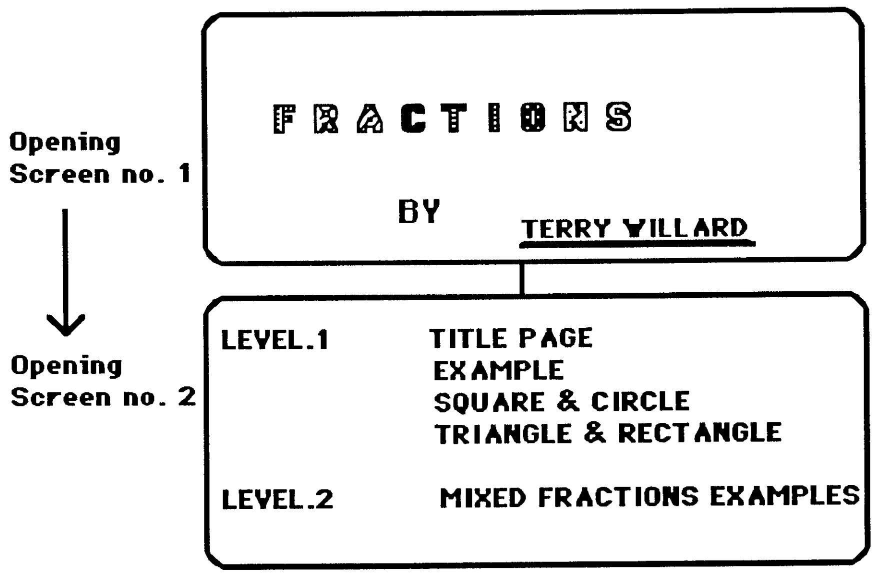
### 7.1. "QUEST": Designing and Implementing the First Procedure for the Software's Opening Screen:

On April 1, Debbie suddenly decided to design and implement her opening screen. So far, although she has often generated ideas in advance for "new" forthcoming screens while still working on an "old" screen, we have never seen her try to create an opening screen. April 1 was the first time she wrote anything in her Designer's Notebook about creating an introduction:

> "Today I am going to work on my INTRODUCTION, IN FRACTIONS. Then I'm going to do another graphic for it. P.S., There's a bug on your head! April Fool..."

It was Debbie, completely on her own, who decided to create the opening screen at that particular time (rather than earlier), and where to place it in the context of her process of software design. By then, however, many other children in the Experimental class had already designed and implemented an introduction or an opening screen earlier on in their process of software design; the idea of creating an opening screen therefore existed in the classroom long before Debbie decided to create one, and probably influenced her decision about it.

A first brief example of one approach to the design of an opening screen will be taken from the software of Terry, a tall, cheerful boy with blond hair and large blue eyes, who was in the high-math group. Terry was popular among his peers and teachers, who considered him to be one of the brightest pupils in the class. Very early on in his processes of software design, Terry created an opening to his software that was composed of two colorful screens: the first read: "FRACTIONS by Terry Willard," in which each letter was painted in a different color, his name was labeled in yellow, and the background was painted in blue; the second screen for his opening was a menu, or a table of contents for his software as a whole, in which he labeled the words in yellow over a red background. Both screens are shown in the following diagram:



**Terry's Opening Screens, March 19**

As another example, let us briefly follow another girl, Rally, who during the computer sessions sat only two seats away from Debbie. Rally was a black girl from the high-level math group, known by her peers and teachers to be very sociable and successful in everything, including writing, drawing, Logo, and mathematics. In general, Rally, a "top-down" thinker much like Terry, also followed a different route from Debbie's in her processes of software design, had different ideas about fractions and teaching, and used quite a different style in implementing these ideas.

Rally planned to create an opening screen as early as the second week into the Project. On March 10, in one of her first plans in the Designer's Notebook, Rally wrote:

"I will start on my fractions project. I will draw different fractions and tell what fractions they are. I selected this because it seems very easy. I want to teach them different fractions. I might make opening page, Welcome To Fractions."

However, although Rally worked on another screen (about fourths) for the following 10 days, she constantly mentioned and drew plans for an opening page in her Designer's Notebook. For example, on March 12 she wrote that her plans for that day were to

"make questions for the kids about fractions, see the design on the next page. [I will] try to make an opening page that will say Welcome To Fractions [and she drew the opening screen on a special grid in her Notebook]." On March 19 she wrote: "Today I will definitely make the opening page...[and drew it again]."

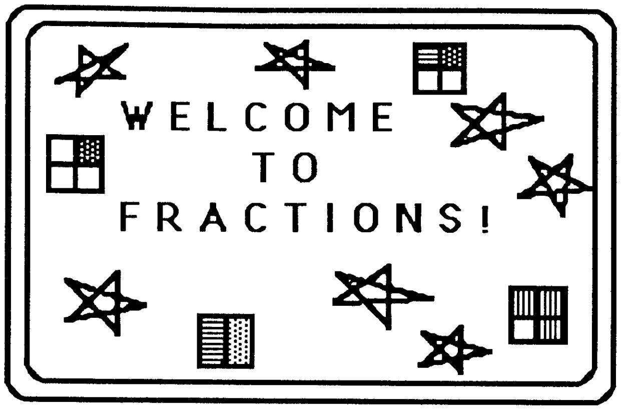
On March 23 Rally wrote:

"[My plans for today are to] do my opening page and if I have time, I will make stars around it [she drew it again with the stars]." On March 26 she wrote: "I will finish making designs around my opening page. I will also make shapes that look like fractions and stamp them around it."

At the end of the computer time on March 26, Rally finally wrote in the Reflections Form in her Notebook:

"I finished everything!!! I had no problems."

By that time (March 26) Rally's first instructional screen showed four squares of the same size, placed evenly on the screen from right to left; and in each of them she shaded one-fourth, two-fourths, three-fourths, and four-fourths, respectively. In addition to that screen, she also had an opening screen which read: "WELCOME TO FRACTIONS!" It had originally been designed in many colors, and was accordingly implemented with many little stars and small representations of fourths spread all over the screen. In short, Rally was involved for a long time in thinking about, and implementing her rather detailed opening screen. She programmed it using the Logo Writer Shapes Page for creating a shape for each character, the stars, or the little fraction representations, and then she used the Logo regular command for placing the shapes and connecting them on the computer screen. Rally presented her software users with this opening screen:

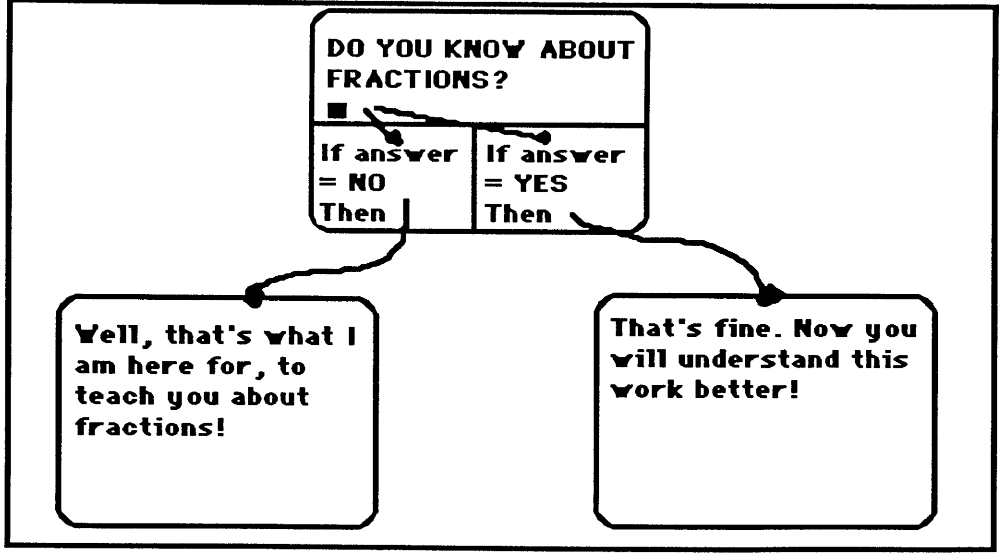


**Rally's Opening Screen**

In a completely different style, Debbie, in her opening screen, used white plain text over a black background, and asked her users a question using the following Logo procedure:

```
TO QUEST
PR [Do you know how to do fractions?]
name readlist "answer
ifelse :answer = [no] pr [Well, that's what I'm here for, to teach you about fractions!]]
[pr [That's fine. Now you will understand this work better!]]
END
```

**The structure of Debbie's procedure QUEST is depicted in this diagram:**


DO YOU KNOW ABOUT FRACTIONS?
If answer = NO Then
If answer = YES Then
Well, that's what I am here for, to teach you about fractions!
That's fine. Now you will understand this work better!

### 7.1.1. The three differences between Rally's and Debbie's approaches to instructional software design

Three differences between Rally's and Debbie's approaches to instructional software design can be interpreted from the preceding information (Terry's instructional design approach and implementation style will be here considered equivalent to Rally's).

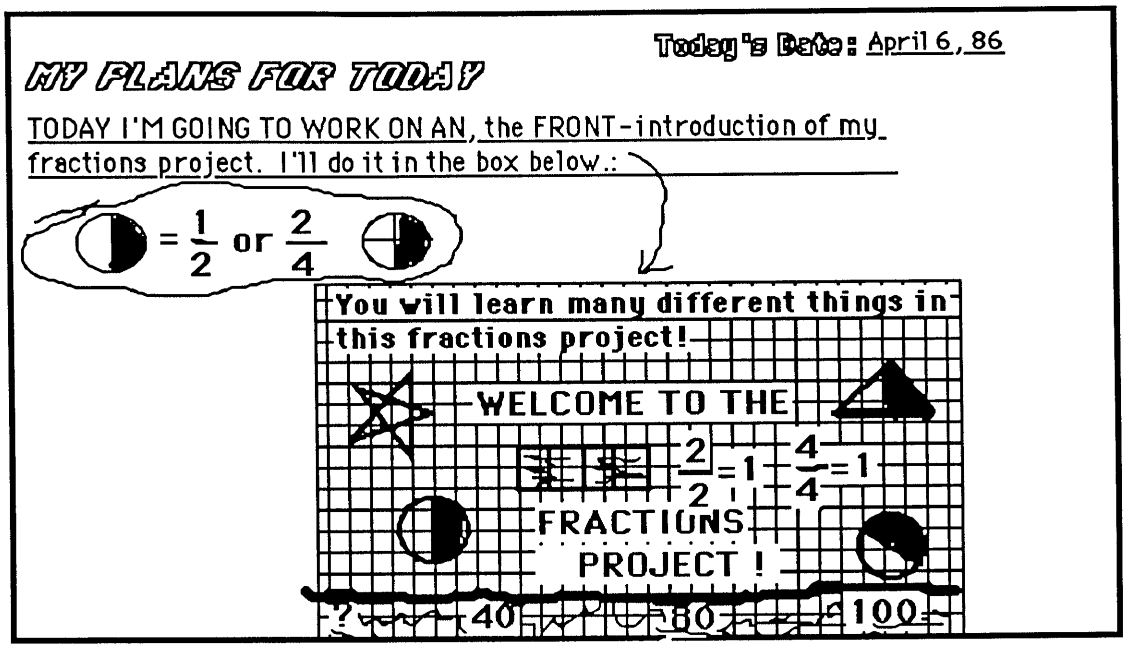
**1)** First, Rally, who seemed to be constantly thinking in a top-down style felt a need, first of all, to plan the opening screen for her program; whereas Debbie, who was probably more of a middle-out thinker, did not feel the need to do this until she had a finished entity to create an opening screen for. (Other examples for the top-down, middle-out, and bottom-up styles of thinking can be found in Turkle, 1984; Papert, 1980; and Weir, 1986.) **2)** The second difference between Rally's and Debbie's approaches for designing their opening screens shows two uses of the computer as an instructional medium. Debbie's opening screen took advantage of the computer's interactive capabilities and was designed as a question, much like a teacher's opening of a lesson in a classroom, or the opening of a conversation between a teacher and a student before the start of a teaching session (i.e., "Do you know about fractions?"); Rally's opening screen, on the other hand, was more like the opening page of a book about fractions, or a cheerful first screen in a movie about fractions (i.e., "Welcome To Fractions!"). Both screens were instructionally designed to motivate the user, each in a different way: the user in Rally's scenario was presented with a very colorful, attractive, and inviting first screen; the user in Debbie's scenario was presented with a rather plain, text-only screen, but was also required to be immediately active and reflective in his process of learning. **3)** The third difference suggested by each of the opening screens was in their presentation of the purpose of each software. Rally's screen invited the user into the world of "fractions," giving the impression that fractions were entities one could visit. In Debbie's opening screen, if the user typed "NO, [i.e., I do not know about fractions]," the program (which represented Debbie's voice) answered: "Well, that's what I'm here for, to teach you fractions!"--i.e., the purpose of Debbie's program was to teach fractions. If the user typed "Yes [i.e., I do know about fractions]," Debbie's program answered: "That's fine. Now you will understand this work better!"--i.e., the knowledgeable user would understand Debbie's work better. In addition, Rally was inviting everybody in the same way, whereas Debbie was considering two types of users: the ones who did not know about fractions, to whom Debbie was saying: I will teach you about it, the purpose of my work here is to teach and explain fractions. The other type of users Debbie was considering were ones who already knew about fractions, and to them Debbie was saying: that's fine, you know about fractions already, so it will be easy for you to understand my screens; but it will still be interesting for you to use this software, so you can better understand the work and the reasons for designing those representations. She seemed, in this case, to be inviting the more knowledgeable users to enjoy and evaluate her work.

**At the end of the computer time on April 1, Debbie's super-procedure looked like this:**

```
TO FRACTIONS
QUEST         [i.e., the new addition to the program]
WAIT.FOR.USER   CG CT CC
HALF
WAIT.FOR.USER   CG CT CC
CALF
WAIT.FOR.USER  CG CT CC
TWO
WAIT.FOR.USER   CG CT CC
HOUSE
WAIT.FOR.USER   CG CT CC
END
```

### 7.2. "FRONT-Intro": Designing Another Opening Screen

During the computer time on April 2, the day after Debbie finished her Quest Procedure, she worked with a third grader. This first Evaluation Session is described in Section 8 below, and we will side-step it for a moment, since a few days later, on April 6, Debbie began thinking again about her introduction. She was probably not fully satisfied with her introductory screen QUEST, and being strongly influenced by her peers' work, and especially by Rally's opening page (which was described in Section 7.1. above), Debbie added a new design in her Designer's Notebook, which she called the FRONT-INTRODUCTION. (i.e., something that would appear in front of the existing introduction QUEST). Debbie's plans from April 6 are shown below:

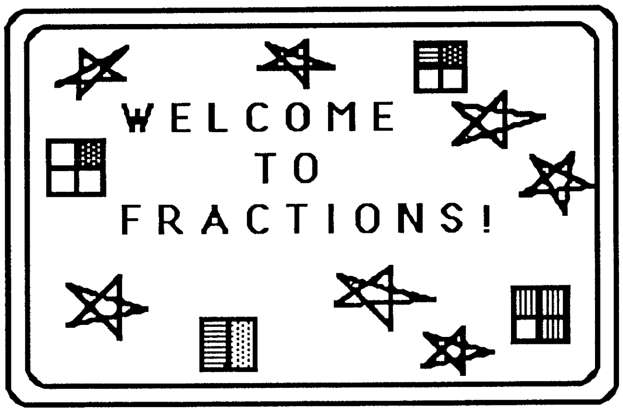


MY PLANS FOR TODAY

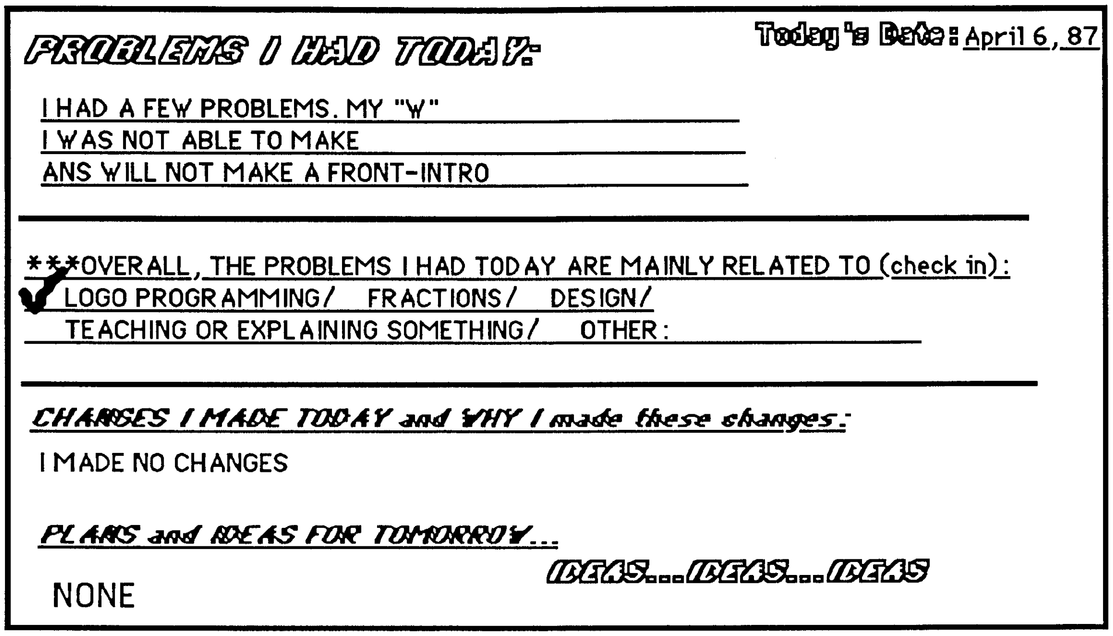
Today's Date: April 6, 86

TODAY I'M GOING TO WORK ON AN, the FRONT-introduction of my fractions project. I'll do it in the box below.:

And presented here again, is Rally's final opening screen...



**The following are Debbie's Reflections of April 6:**



PROBLEMS I HAD TODAY: Today's Date: April 6, 87

I HAD A FEW PROBLEMS. MY "W"
I WAS NOT ABLE TO MAKE
ANS WILL NOT MAKE A FRONT-INTRO

***OVERALL, THE PROBLEMS I HAD TODAY ARE MAINLY RELATED TO (check in):
[✓ LOGO PROGRAMMING]/ FRACTIONS/ DESIGN/
TEACHING OR EXPLAINING SOMETHING/ OTHER:

CHANGES I MADE TODAY and WHY I made these changes:
I MADE NO CHANGES

PLANS and IDEAS FOR TOMORROW...
IDEAS...IDEAS...IDEAS
NONE

Debbie called this design the FRONT-introduction. She meant to put it in front of the introduction she had already created, named QUEST (See Section 7.1 above). One interesting aspect of Debbie's plan above is the fact that she drew representations for the equivalence of 1/2 and 2/4 (i.e., the representations that she circled in her Plan Form for April 6). It seems as though, by that time, she still strongly believed that this particular equivalence between halves and fourths was very basic in one's knowledge about fractions. One recalls that she had designed her first screen (i.e., in Section 1 above) to teach that 1/2=2/4; now she decided to include this fractional equivalence in her introduction as well. She also repeated her representations of halves or two-fourths on the right part of each shape on that plan.

Did she do this thinking that these types of representations (on the right side of each shape) would be more appropriate for beginners in fractions? Was it because she herself had not fully overcome this misconception (i.e., that a half must always be on the right side of the shape)?

These questions require further investigation. I believe that in general, the children's drawings in their Designer's Notebooks did represent their spontaneous and non-reflective thoughts about fractional representations; and that the children's reflections about their drawings usually occurred during the implementation phases or after the computer time was over. (In addition, there is one part of this design that I do not understand at all. I have no idea why Debbie wrote those numbers at the bottom of her design grid. For unknown reasons, Debbie designed three lines at the bottom of her grid that were composed of question marks, messy curved lines, and many numbers.)

During computer time Debbie started working on her plan for the new introduction. In her Logo file she added the following code:

```
TO FRONT     [She left it empty. It represents Debbie's creation of a plan during
             the implementation phase]
END

TO W
pu setpos [-85 55]
rt 135
setc 1+random 14  pd fd 20
lt 100
setc 1+random 14  pd fd 19
rt 100
setc 1+random 14  pd fd 20
rt 100
setc 1+random 14  pd fd 20
rt 100
setc 1+random 14 pd fd 19              [She left it unfinished]
END
```

The analysis of this small piece of code reveals that Debbie thought about her "FRONT" in a modular way. Her plan was to create a sub-procedure for each of the letters in "W-E-L-C-O-M-E T-O T-H-E F-R-A-C-T-I-O-N-S P-R-O-J-E-C-T-!" and to later connect all the sub-procedures together in a super-procedure named "FRONT." However, at the end of that session she wrote in the Reflections Form in her Notebook: "I WILL NOT MAKE A FRONT-INTRO." Although she was quite aware of the reason why she did not intend to make this screen, Debbie checked in "LOGO PROGRAMMING" in the above

Reflection Form as her main problem for that day.

This was the first time in her software design process that Debbie, during the implementation phase itself, decided not to implement a plan because of some technical problems in Logo.

We have seen that Debbie had previously designed screens that she never actually implemented, but we never saw her starting to implement something and stopping it in the middle. Moreover, the problems she had, according to her, were related to Logo programming; but I perceive her problems as mainly related to her impatience, rather than to her lack of knowledge or skills in Logo. I never actually discussed with Debbie why she had decided at first to plan the FRONT-introduction, or why she decided to stop and eliminate it in the middle of the implementation phase. It seems, however, according to what she wrote, that she had problems creating the letter "W." She probably spent a long time trying to figure out how to make the "W" look like a real "W," but with little success, and so she decided not to implement the whole screen at all. It was not a programming problem *per se*, but rather a problem in constructing a letter shape. The "W" was difficult to implement in Logo due to its diagonal lines and internal angles, and Debbie saw it as an obstacle and did not like the way it came out:"When I try it, it doesn't look like a real W," she said, "it looks crooked."

Debbie's peers used the Logo Shapes Page and the Logo Labeling capabilities (see Logo Writer Manual, 1987) to print words or characters in their opening screens; but Debbie, for an unknown reason, did not use the Shapes Page. Her work with the "W" during that session definitely exhausted her patience. However, I assume there were other reasons why Debbie decided not to continue implementing this plan. First, I suspect it was because this particular design was not really her own idea. It was too similar to Rally's opening page and perhaps Debbie did not feel happy about this. Also, it was a rather time-consuming screen to implement. In order to finish it, Debbie needed to work for a long time on constructing each of the characters, one by one. This construction required very precise work and a great deal of attention to details. It seemed to me that Debbie, unlike Rally, did not enjoy this type of work. If one recalls, Rally had worked on her very similar opening screen using the Logo Shapes Page, for 10 days. Rally enjoyed the construction of each character or shape; and the opening screen, in principle, was very important to her (in the same way it was important to Terry, whose screen is described above). Debbie, however, probably had less tolerance for work of this kind, and was not ready to deal with this type of very detailed and precise work. Her struggling with the "W" was enough of an experience to reveal to her that she was not ready to continue or invest many days in this activity.

### 7.3. The New Introduction

During the last 15 minutes of the computer time on April 6, Debbie cancelled her plans for the construction of the FRONT-INTRO, and instead she created and added a new introduction:

```
TO INTRODUCTION
pr [Hello, my name is Debbie. I am going to teach you how to do fractions.
The first thing that I am going to teach you is the half of the screen.
The scene is a half or two fourths.]
END
```

She then entered her super-procedure and replaced the QUEST procedure with the new INTRODUCTION procedure, so that her super-procedure "FRACTIONS" looked like the following:

```
TO FRACTIONS
INTRODUCTION
        [This had been QUEST's position before Debbie replaced it with INTRODUCTION]
WAIT.FOR.USER  CG CT CC
HALF
WAIT.FOR.USER CG CT CC
CALF
WAIT.FOR.USER CG CT CC
TWO
WAIT.FOR.USER CG CT CC
HOUSE
WAIT.FOR.USER CG CT CC
END
```

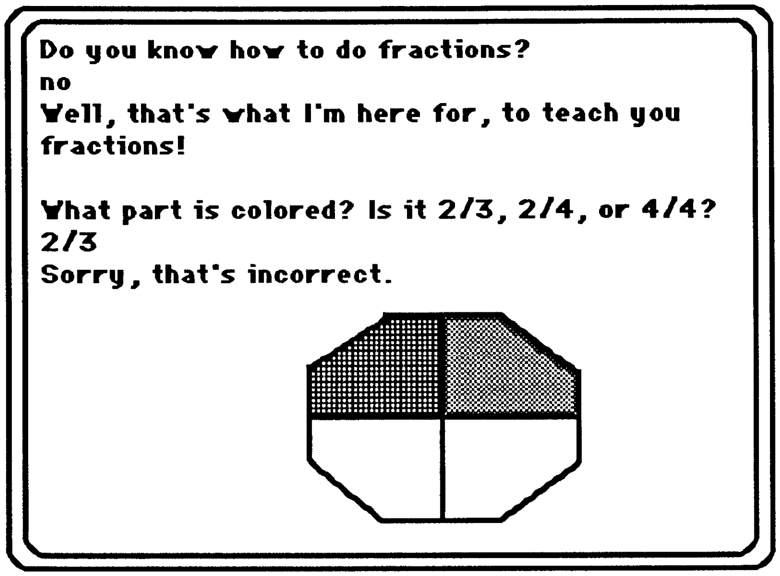
She ran her revised super-procedure twice, and re-entered the Editor. She then added graphics for her procedure QUEST:

```
TO QUEST
pr [Do you know how to do fractions?]
name readlist "answer
ifelse (:answer = [no])
[pr [Well, that's what I'm here for to teach you fractions!]]
[pr [That's fine. Now you will understand this work better!]]
```

**[This was, up to this point, Debbie's original QUEST procedure; from here on the graphics and interactivity are new:]**

```
pu home pd repeat 8 [setc 1 +random 14 rt 45 fd 50]
pu bk 25
rt 90
pd fd 120
bk 60
lt 90
pu bk 60
pd fd 121
bk 70
rt 45
pu fd 20
setc 1+ random 14
pd fill
pu home
rt 90
pu fd 20
setc 1+random 14
pd fill
pr [What part is colored?  Is it 2\/3, 2\/4, or 4\/4 ?]
name readlist "answer
```

```
ifelse (:answer = [2\/4])
[pr [GREAT!]] [pr [Sorry, that's incorrect.] wait 50 cg ct cc QUEST]
END
```



```
Do you know how to do fractions?
no
Well, that's what I'm here for, to teach you
fractions!

What part is colored? Is it 2/3, 2/4, or 4/4?
2/3
Sorry, that's incorrect.
```

**The Procedure QUEST**

**April 6**

As we saw previously, Debbie decided to replace her procedure QUEST with the procedure INTRODUCTION (in her super-procedure FRACTIONS). After this, Debbie created a new super-procedure which she called BIG, and put QUEST within it, at the top of procedure BIG:

```
TO BIG
QUEST
WAIT.FOR.USER   CG CT CC
HALF
WAIT.FOR.USER  CG CT CC
HOUSE
WAIT.FOR.USER  CG CT CC
TWO
END
```

I asked her: "What is this procedure, Debbie?"

**Debbie:** "A different one for my fractions project."

**I:** "How is it different?"

**Debbie:** "I put QUEST, not the INTRODUCTION. I put HOUSE before TWO, I didn't use CALF..."

**I:** "Why?"

**Debbie:** "Because."

**I:** "Aha...." [I started to move away]

**Debbie:** "O.K... O.K... 'Cause now I have two of them. Which one you want to see?"

**I:** "So you created two of them..."

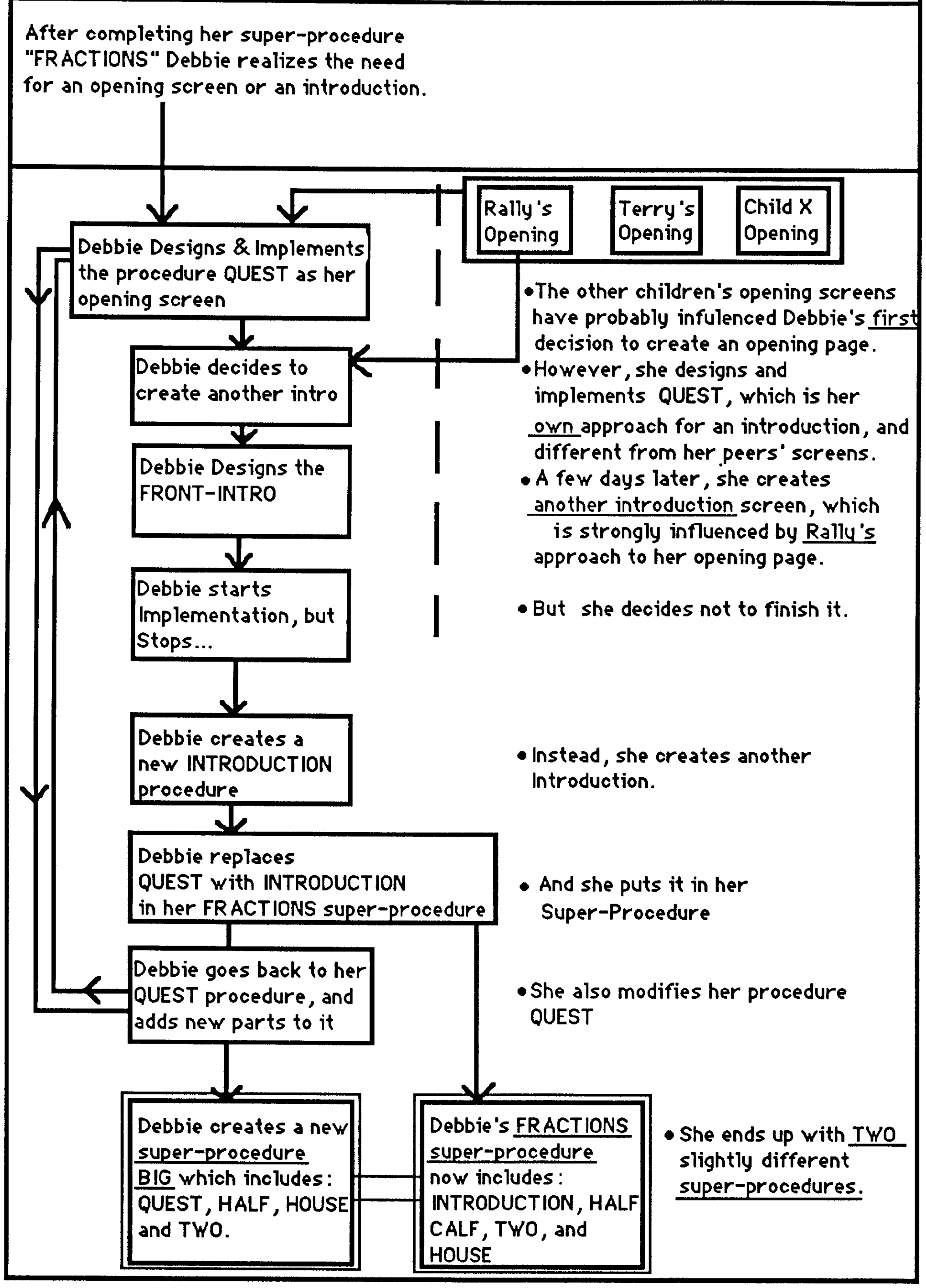
**Debbie:** "Yes. Two different fractions Projects! Which one you want to see?"

<u>I asked to see both.</u> She showed them to me. She seemed to enjoy the fact that she now has <u>two</u> slightly different fractions projects. The two super-procedures (BIG and FRACTIONS) had some parts in common, and some that were different or connected in different sequences. I interpret Debbie's act of creating (and enjoying) her two super-procedures as an experiencing of the purpose and idea of *modularity* -- one of the central ideas in programming (Papert, 1980).

Debbie wrote a modular program in which each piece could be understood and used separately. Each of Debbie's procedures could be read and executed in isolation, on its own. Debbie probably realized during this particular session that she could connect her parts in many different ways. She was very proud when she asked me: "Which one you want to see?" For the first time, she had discovered the trick of using similar elements for making two fractions projects.

To summarize, the structure of Debbie's processes of designing, redesigning, implementing, and modifying her introduction screens on April 1 and 6, and the way her actions related to or were influenced by her peers' actions, are shown in the diagram on the following page. (This is not a final diagram for Debbie's processes of creating an introduction to her software. She would return to the problem of the introduction during April, May, and again in June, and add other parts to it. These additions to the introduction will be described in later sections.)

**THE STRUCTURE OF DEBBIE'S PROCESSES OF DESIGNING, IMPLEMENTING, RE-DESIGNING, AND MODIFYING HER INTRODUCTORY AND OPENING SCREENS ON APRIL 1 AND APRIL 6:**


After completing her super-procedure "FRACTIONS" Debbie realizes the need for an opening screen or an introduction.
Rally's Opening
Terry's Opening
Child X Opening
Debbie Designs & Implements the procedure QUEST as her opening screen
Debbie decides to create another intro
Debbie Designs the FRONT-INTRO
Debbie starts Implementation, but Stops...
Debbie creates a new INTRODUCTION procedure
Debbie replaces QUEST with INTRODUCTION in her FRACTIONS super-procedure
Debbie goes back to her QUEST procedure, and adds new parts to it
Debbie creates a new super-procedure BIG which includes: QUEST, HALF, HOUSE and TWO.
Debbie's FRACTIONS super-procedure now includes: INTRODUCTION, HALF CALF, TWO, and HOUSE

- The other children's opening screens have probably infulenced Debbie's first decision to create an opening page.
- However, she designs and implements QUEST, which is her own approach for an introduction, and different from her peers' screens.
- A few days later, she creates another introduction screen, which is strongly influenced by Rally's approach to her opening page.
- But she decides not to finish it.
- Instead, she creates another Introduction.
- And she puts it in her Super-Procedure
- She also modifies her procedure QUEST
- She ends up with TWO slightly different super-procedures.

## 8. DEBBIE'S FIRST SOFTWARE EVALUATION

On April 2, the teacher of the Experimental class, another third-grade teacher, and I, spontaneously organized a collaborative computer session for one third grade and the Experimental fourth grade. The purpose of this session was to give the Experimental children a first chance to evaluate the effectiveness and appeal of their pieces of software with third graders (their target users). The third-grade teacher was also interested in this collaboration for the purpose of having the fourth graders help her pupils with their programming projects. The teachers assigned one third-grade child to each fourth-grade child according to gender and level, but also based their decisions on which children would most enjoy working together and would work best together. This session was divided into two units: during the first unit, each of the third graders was asked to use his fourth-grade partner's Fractions Software and offer him comments about it. This Evaluation Session lasted 30 minutes. In the second unit, the roles switched, and each fourth grader worked with his third-grade partner on the latter's project; if they preferred, they could work on any other collaborative project. We did not tell the children what to do, what to ask each other, or how to work together. This may be one of the reasons why this spontaneous session was not very successful in terms of the fourth-grade children's understanding of what the evaluation meant. Many of them treated it more like a demo session, and only a few of them used this session to teach about fractions or to evaluate the clarity or appeal of their representations and screens. Decentering was a difficult task for many of the Experimental children. They could think about their users while designing their software, but many of them were not ready to consider their users' comments, or seek for criticism on the quality and contents of their fractions programs. Beyond that, it was still interesting to walk around and see the differences in the ways these children interacted with each other and worked together. The teachers, their pupils, and I felt that this was the beginning of something positive that should happen more often in the school, and that all sides gained something from working together. It was a difficult session to organize logistically, so only three other such sessions were conducted during the months of April and May. It is beyond the scope of this section, or of this chapter to describe what went on during these collaborative sessions, or what the different children, fourth or third graders, gained from it. However, let us briefly look at what Debbie accomplished in that first collaborative session on April 2.

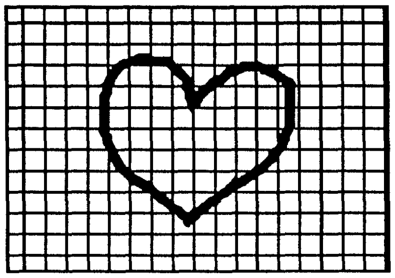
That day, in her Designer's Notebook, Debbie wrote:

*MY PLANS FOR TODAY:* Today's Date: April 2, 87

"Today I will be working, I think, with a third grade. I will show them my fractions project.

I might do a new little procedure from what Harry taught me.

Then I will, maybe, make a heart. This is what the heart looks like:



I can use the:

```
TO I
 REPEAT 35 [setc 1 + random 14 fd 1]
END."
```

During that session, Debbie indeed showed her Fractions Program to Bibby, the third-grade girl who was working with her. They were rather silent during most of the session, probably because it was the first one of its kind for them; they could not yet define the rituals of this specific interaction and were not sure who was supposed to play what role. Debbie's partner, Bibby, was fascinated with Debbie's software, and her face was quite serious while she watched the software, start to finish. She read the text on the screen word by word, and pressed the keys when she wanted to continue. When it was over, she looked at Debbie and said:

**Bibby:** "This is nice. Can you do this again?"

**Debbie:** "You can type the word fractions yourself."

[i.e., fractions is the name of the super-procedure]

**Bibby:** "Oh..." [and she types it slowly in the Logo Command Center]

**Debbie:** "That's it. Press enter"

**Bibby** [Asking after reading the opening screen]: "What will happen if I say, Yes?"

[to the question "Do you know about fractions"]

**Debbie:** "Do you *really* know about fractions already?"

**Bibby:** "Sort of...We started something..."

**Debbie:** "O.K. then..." [Bibby gets an approval to type "Yes," and so she does]

**Bibby** [reads the answer on the computer screen:"That's fine. Now you will understand this work better." She smiles for the first time, and continues watching screen after screen executed before her eyes]: "This is a long program. Really long. How did you do that?"

**Debbie** [probably perceives this comment as a compliment, and answers]: ".. Thanks."

**Bibby:** "Can you show me how you did it?"

**Debbie:** "Sure." [Debbie flips into the Editor page and runs the cursor down the screen, reading the names of each of her procedures] "Quest..Half.....this is Calf.....this is House....this is Two....this is, Question1....and this is Fractions. This is Wait.for.user and Ignore. Fractions is the whole thing together. It's my super-procedure."

**Bibby:** "Aha..."

**Debbie** [sensing that Bibby is puzzled]: "Do you know what's a super-procedure?"

**Bibby:** "No."

**Debbie:** "It's a procedure, that [pause] you can put in it many other procedures, together."

**Bibby:** "Aha..."

**Debbie** [places the cursor within the super-procedure and "walks" Bibby through it]: "This is the opening, Quest. This is the first procedure, Half. Then I put Wait.for.user, to ask you type any key to continue, remember? [doesn't wait for an answer], then CG CC CT."

**Bibby:** "I know...Clear the screen..Clear the words..."

**Debbie:** "Exactly." [She continues] "Then the same thing with Calf. Then the same with Two, the same with House, and this is it. The end. [pause] But I'm going to add many new graphics to it."

**Bibby:** "Can you teach me how to do it?"

**Debbie** [does not really answer that question. Instead, she opens her Designer's Notebook and shows Bibby her design of the heart, which was on her Plans Form of April 2]: "Do you want to do something like this?" [Points at her drawing of the heart in the Designer's Notebook]

In the background the two teachers were announcing that it was time to switch roles, and the fourth graders should now look at the third graders' programs and help them.

Debbie asked her teacher if instead, she and Bibby could program together the heart she had planed. The teacher approved, and they started to work on it.

**Bibby:** "Yeah...let's work on the heart. I like it."

Bibby and Debbie were both sitting beside their own computers, writing the Logo commands simultaneously. Debbie was in fact leading the session, speaking in a louder voice and saying what they should type, while Bibby mumbled and echoed Debbie's words, while typing them into the computer; it was though Debbie were playing the soprano, and Bibby the alto. After every three characters of code, Debbie would look at Bibby's computer to check whether she had mistyped anything. At the end of the computer session both had an identical procedure named "Heart" in their Logo files. They ran it simultaneously, twice. When the third graders went back to their classroom, I approached Debbie and asked her,

**I:** "How did it go?"

**Debbie:** "O.K." [it was a very lengthy O.K., in a somewhat disappointed tone of voice]

**I:** "What did she tell you about your fractions project?"

**Debbie:** "I think she really liked it." [cheering up]

**I:** "Great."

[pause]

**I:** "Did she tell you to change anything?"

**Debbie:** "No."

**I:** "Do you feel you should change anything now after you saw her using it?"

**Debbie:** "No."

**I:** "O.K., I guess that it's perfect then."

**Debbie:** "Yes." [pause] I even showed her the program. I showed her the super-procedure."

**I:** "Aha. And what did she think?"

**Debbie:** "That it was really long."

**I:** "So..."

**Debbie:** "I guess they are not used to such long programs."

**I:** "Does long mean good or bad or...."

**Debbie:** "Good, I guess. 'Cause it has many different things. She also liked my House."

We can see that Debbie was not seriousely inspired by this session, and did not actually use it for a systematic checking or testing of the different parts of her program. This first session showed me that evaluation is a rather sophisticated skill. It seemed, however, that some children intuitively had it while other did not. This skill is definitely not often exercised, if at all, in educational practice. I did notice several children asking the third graders evaluative questions such as: "Is it nice?" "Is it clear?" "Boring!?" "Did you like it?" etc. Also, three of the children actually conducted a mini teaching-session with their third-grade partners, and supplemented their demonstrations with explanations about fractions (i.e., "did you know that 3/6 equals 1/2?" or "Count, one two three. Out of, one two three four five. This is three-fifths"). Some children even explained to their third-grade partners why they had designed a screen in a specific way, or what they had wanted to teach (i.e., "This screen teaches you about equivalence of fractions" or this screen teaches you about fourths. It's tricky..."). But, as far as I know, Debbie did not implement any of these evaluation strategies. She did not seek information about her product, how it worked for Bibby, what Bibby did or did not understand, etc. She just sat there and watched passively, while Bibby looked at Debbie's program being executed. Bibby did give Debbie a feedback: "This is nice. Can you do it again?" In fact, Bibby showed interest in learning "how to do it," but when the time came, Debbie was more interested in programming the "Heart."

On the whole, Debbie gave her third-grade partner a simple demo. And if she taught her anything at all, it was about programming and not about fractions. She "walked" Bibby through her super-procedure and explained to her what a super-procedure was and what its role in her software was. Unlike many of her peers, Debbie also seemed a bit nervous, perhaps because she was socially insecure in general, or perhaps because she feared criticism of some sort. It is reasonable that she was afraid Bibby would not like her screens and representations--or would not find them difficult to program. After all, Debbie had invested a lot of time and thought working on them.

The ideas behind this evaluation session, its role in the children's learning, and the different strategies that were used by the fourth and third-grade children during that session require further investigations. In general, I think that a girl like Debbie needs to undergo more sessions of this sort, in order to learn through experience how to collaborate with younger children, how not to fear their potential criticism, and how to make positive use of such interactions with younger users, thus acquiring new information that might inspire her own designing, programming, and modifications. In addition, the way Debbie handled the programming of the heart shows that she did not really collaborate, but rather told Bibby exactly what to do and how to do it. I am not sure whether Debbie was aware of her own fears of criticism; but she was clever enough to <u>shift the control</u> from Bibby, the evaluator, to herself. Then she, who had originally planned and knew how to do the heart, made Bibby follow her orders. Furthermore, Debbie was probably the kind of girl who needed time to learn to enjoy social interaction. The same phenomenon of Debbie's showing anxiety and lack of interest had occurred in her relationship with me, when the Project first started (i.e., what I described as "Phase I" in the Introduction to Debbie's Case). It took me a long time to understand how to work with her, and what type of interaction or collaboration she would accept. She never "allowed" me to suggest anything to her that was not already an idea of her own. And the few times I did it, she became very reserved, negative, and silent, unlike the other children I had worked with, who grew animated and were inspired by the same exact comment. Finally, it might have been a good idea to conduct a short "Focus Session" before the Evaluation started, in order to clarify, with the children/software-designers, what software evaluation meant, and suggest which kinds of evaluation questions asked by the fourth graders might prove useful and productive.

## 9. CREATING A NEW LOGO PAGE, "FRACT2": DEBBIE MAINTAINS DYNAMIC RELATIONSHIPS BETWEEN NEW KNOWLEDGE AND OLD IN HER SOFTWARE AND IN HER MIND.

The relationship between new knowledge and old is an important aspect of this thesis as a whole. But it is highlighted here because Debbie's plans, processes, and reflections on April 7 and during the following weeks were very rich in information on this particular issue. With regard to these dynamic relationships between Debbie's new and old knowledge, the days from April 7th onwards were very interesting in her processes of software design and Logo programming. We saw that during the month of March (i.e., Sections 1 to 6), almost all of Debbie's plans and designs corresponded, more or less, with her implementations. Even though she designed several screens that never reached implementation, or designed screens in advance and implemented them in later days, we could still follow her thinking in a somewhat organized fashion. During March, when all of her work was included in one Logo Page, it was simpler (for me as well as for Debbie) to analyze what were the new parts she added each day, which parts she changed, eliminated, etc.

However, the day Debbie's program began to grow beyond one Logo Page, and became more complex, was also the day Debbie's thinking started to be more complex and much more dynamic. Moreover, she began to grow relatively more involved in the programming aspect of her software, rather than in principally the learning of fractional concepts or in the designing of fractional representations. She became much more involved in moving procedures from one Logo Page to another, creating new super-procedures, and discovering new combinations of sub-procedures within those super-procedures; she concentrated much more on the user's interaction with her software, and added short introductions and instructional sentences here and there; she was also very involved in finding ways to connect the different Logo Pages (i.e., during April she worked between three Pages: FRACTION, FRACT1, and FRACT2). On the whole, during the month of April Debbie solved many technical problems relating to which procedures should be on which Page, though she occasionally got confused and once gave the same name to two different procedures.

In the previous sections I often analyzed Debbie's learning of fractional concepts and especially her thinking through the concept of halves. However, in the following sections I emphasize much more her learning of Logo. Section **9.1.**, I describe how Debbie planned to connect or integrate her new knowledge with the old, and the ways she wrote about it in her Designer's Notebook; Section **9.2.** describes the actual reality: which connections between pieces of knowledge Debbie actually discovered while working with the computer. In other words, Section 9.2. emphasizes the relationships between Debbie's plans and their reality in implementation.

### 9.1. April 7: Debbie's Plans

This is what Debbie wrote in the Designer's Notebook on April 7:

"TODAY I WILL DO SOMETHING ELSE. I CAN'T FIT ANY MORE WORK ON MY "FRACTIONS PAGE. SO I'M GONNA DO A NEW PAGE CALLED "FRACT2. I'LL DO EQUIVALENT FRACTIONS ON "FRACT2. I MIGHT DO SOME MORE POEMS. IF I HAVE TIME I'LL ADD MUCH MORE STUFF ON OTHER PAGE, THAT I'VE LEARNED."

This particular plan contains many interesting issues that need to be highlighted and interpreted carefully in relation to the dynamic relationships between new and old pieces of knowledge. Here is how I interpret Debbie's plan:

**1.** Today I will do something else. In other words, I have finished everything I wanted to do so far, and all the things I'll do from now on are new and different ("something else").
**2.** I can't fit any more work on my existing Logo Page named FRACTION. Therefore, I'll do a new Page which I am planning to name FRACT2.
**3.** On the new FRACT2 Page I'll create representations of equivalent fractions (for sure).
**4.** On the new FRACT2 Page I'll do some more poems (maybe).
**5.** If I have time, I'll add much more stuff on the old Page (named FRACTION).
**6.** The new things that I'll add to the old Page are things that I've recently learned.

The "doing something else" that Debbie refers to in point 1. above could mean all the points that are listed under it: new Page, called Fract2; new representations for equivalent fractions; new poems; and new procedures in the old Page.

On April 7, Debbie wants to start "something else." She also wanted to create a new

Page, FRACT2. On FRACT2 she planned to do something new. I assume she also meant "something else" in terms of fractional representations, since she writes about her plans to "do equivalent fractions on Fract2." However, Debbie also planned to "add much more stuff on the other page," using the new knowledge that she had recently acquired. This is very interesting, for the fact Debbie was considering starting something new no longer meant that she was completely done with the previous Page she had been working on. On the contrary, her writing reflects her awareness of the need to keep on going back to the existing procedures that were already working, and to maintain them, modify them, and add new parts to them, using new knowledge she had just learned. She was aware of learning new things, and interested in applying them even to existing procedures that already worked well. We can see that the Instructional Software Design Project indeed involved Debbie in complex maintaining processes, and in going back and forth between the old material she had produced in the past and the new. She was willing to update the old, revise it, and connect it with the new material she was now learning.

In fact, Debbie's plan on April 7 is quite complex and represents her broad treatment and interest in her Fractions Project as a whole: the fact that she was planning to start a new Page does not mean that she should put aside the old Page; on the contrary, she was planning to keep on modifying the old Page as well. The fact that she was planning to start a new representation for equivalent fractions does not mean that she was going to leave aside her previous representations; instead, she would continue working on her old representations and add things to them, since she had recently learned new things that could be applied to the old representations or to their procedures. Her desire to write poems is also reflected in this plan, and the fact that she was involved in the Fractions Project did not prevent her from taking short breaks or doing other things when she felt like it.

### 9.2. April 7: Debbie's Implementation

These were Debbie's plans and their interpretation. Let us enter reality, and analyze which plans Debbie actually accomplished, which she wanted to work on, and why.

Towards the end of the computer session on April 6, Debbie had a difficult time trying to add or change things in her FRACTION Logo Page--the Page was full. In general, there is a space for 4,096 bites on each Logo Page (on both of its sides together). In her program, Debbie had an average of 20 characters per line; and by April 6, the computer "complained" that there was no more space on her FRACTION Page, since she had approximately 150 lines of code in her software. Therefore, when she ran her software, the name and format of the Page took 609 bites, the procedures on its Editor side took close to 3,000 bites, and the pictures and text on the executed screen (the flip side) probably took all the remaining 385 bites of the memory space available per Page. Because of this problem, her screens froze, and many times the cursor got stuck--the computer could not work beyond 4,096 bites per Page. Debbie realized that the Page was full, and that she, in her own words, could not "fit any more work on her Fractions Page." She consulted with her friends about what to do about her problem. She learned that many of them would in similar cases start a new Logo Page.

We have seen that on April 7 she wrote about her plan to "do a new page called Fract2." In Fract2, she planned to start new procedures for representing equivalent fractions. From then on we constantly read about her many plans and designs for representing equivalent fractions. However, analyzing her Logo on-line files reveals that she did not actually accomplish much of this during the month of April. Rather, she worked on other parts of her software in April, and only started implementing and modifying the screens on equivalent fractions at the beginning of May. Debbie did create a new Logo Page on April 7, but did not start programming her procedure for equivalent fractions. Instead, she typed in her FRACT2 Page the following:

```
TO EXAM
END

TO INTRODUCTION
END

TO QUEST
END

TO BIG
END

TO TOPO
END
```

```
TO INTRODUCTION
END

TO WAIT.FOR.USER
END

TO IGNORE :KEY
END

TO SUPER
END
```

We can see that Debbie first typed nine procedure-names, which she was planning to include on her new Page FRACT2. Five of those existed already in her old Page (i.e., INTRODUCTION, QUEST, BIG, WAIT.FOR.USER, and IGNORE), while the other four were being mentioned here for the first time as new procedure-names that would probably lead to new parts she was planning to create on that day. In that typing act alone, we can see that Debbie was thinking about, and relating to one another, the old parts with the new. She then went back to her old FRACTION Page, and using the appropriate function keys and arrows, copied her five already existing procedures, one by one, transferring them to her new Logo Page FRACT2. This second step resulted in Debbie's having the following code on her new FRACT2 Page:

```
TO EXAM          [does not exist yet]
END

TO INTRODUCTION          [this procedure exists, it was created on April 6]
pr [Hello, my name is Debbie. I am going to teach you how to do fractions.
The first thing I'm going to teach you is the half of the screen.
The scene is a half or two fourths.]
END

TO QUEST       [one part of this procedure was created on April 1, and the other, on April 6]
```

```
pr [Do you know how to do fractions?]
name readlist "answer
ifelse (:answer =[no]) [pr [Well, that's what I am here for, to teach you fractions!]]
[pr [That's fine. Now you will understand this work better!]]
pu home pu repeat 8 [setc 1+random 14 rt 45 fd 50]
pu bk 25 rt 90 pd fd 120 bk 60 lt 90 pu bk 60  pd fd 121
bk 70 rt 45 pu fd 20 setc 1+random 14 pd fill
pu home rt 90 pu fd 20 setc 1+random 14 pd fill
ct pr [what part is colored? Is it 2\/3, 2\/4, or 4\/4?]
name readlist "answer
ifelse (:answer = [2\/4]) [pr [GREAT!]] [pr [Sorry, that's incorrect] ct cg QUEST]
END

TO BIG                              [this super-procedure was created on April 6]
QUEST
WAIT.FOR.USER  CG CC CT
HALF
WAIT.FOR.USER  CG CT CC
HOUSE
WAIT.FOR.USER. CG CT CC
TWO
END

TO TOPO     [does not exist yet]
END

TO INTRODUCTION                     [probably a new one, does not exist yet]
END

TO  WAIT.FOR.USER                   [this procedure was created on March 31]
type [PRESS ANY KEY TO CONTINUE]
IGNORE readchar
END
```

```
TO IGNORE :key                [was created on March 31]
END

TO SUPER                      [does not exist yet]
END
```

(I know little about Debbie's rationale in choosing these particular procedures from the old FRACTION Page, or in choosing these particular names for her new procedures). The third thing Debbie worked on after accomplishing the above was her new procedure SUPER. She created SUPER because she needed one procedure within Page FRACT2 that would call the old Page FRACTION. So she typed:

```
TO SUPER
GETPAGE "FRACTION
WAIT.FOR.USER
END
```

During her process of creating another Logo Page for her Fractions Project, Debbie, like the other children in her class, had to learn two new Logo instructions: GETPAGE, and GETTOOLS. It is important for us to understand the difference between those two instructions, because Debbie had certain misconceptions regarding them that needed to be overcome.

When Debbie typed GETPAGE "FRACTION while still in the FRACT2 Page, Logo automatically had to move from FRACT2 to the other Page, FRACTION. Debbie forgot to tell Logo what to do after it moved to FRACTION, or what procedures to execute. Instead, she should have typed:

```
TO SUPER
GETPAGE "FRACTION
FRACTIONS  [this is what she should have typed, i.e., the name of the
                super-procedure on her FRACTION Page]
END
```

When Logo moved from FRACT2 it forgot everything about it, including the name of the procedure that had instructed it to move (i.e., SUPER in this case). But Debbie also had the possibility of typing GETTOOLS "FRACTION while on the FRACT2 Page. GETTOOLS "FRACTION would have given Logo access to use any procedure that was on the FRACTION Page. This would have been a better solution for what Debbie meant to do: to use her procedure BIG, which included HALF, CALF, HOUSE, and TWO, four procedures that existed only on her old FRACTION Page.

Debbie struggled with this problem of GETTOOLS and GETPAGE during all the rest of the Project. She got confused about which Page should call which one, and which Page should include which tools or procedures. In a way, both GETTOOLS AND GETPAGE possess qualities of modularity, requiring the construction of relationships between whole Pages as well as between the Pages' individual components. These connections between Pages become a higher-level super-structure of the procedures themselves and of their interrelations with the different super-procedures. A great deal of expertise and experience are needed to fully understand this problem. Debbie's program of April 7 was a turning point in that regard: it was the first time she was dealing with this problem of Pages-relations and interrelations. My observations and a brief analysis of the other children's programs revealed that many children struggled with the same problem, and that the children who had dealt with it earlier in the Project learned to master it, having devoted sufficient time during the Project to constructing and manipulating these interrelations between the various Pages of their Fractions Programs. As we continue to follow Debbie's processes in the following days, we will see that she continued to try to solve this problem and managed to work it out for herself.

The next thing Debbie did on April 7 was to write the new procedure INTRODUCTION. It was the first time I had ever seen Debbie use the same procedure name on the same Page and within the same program. Here is what she typed:

```
TO INTRODUCTION
pr [Hello, again! Haven't you learned enough?]
name readlist "answer
[pr [I guess you want to learn more!]]
wait 50 ct
END
```

Seeing this procedure in her program, I asked her,

**I:** "Debbie, what is this procedure?"

**Debbie:** "It's another one."

**I:** "Another what?"

**Debbie:** "Another introduction for the second time."

**I:** "What do you mean?"

**Debbie:** "I will use it in here..." [she talks as she moves the cursor up to TO TOPO, and types,]

```
TO TOPO
BIG WAIT 90 ct
INTRODUCTION
END
```

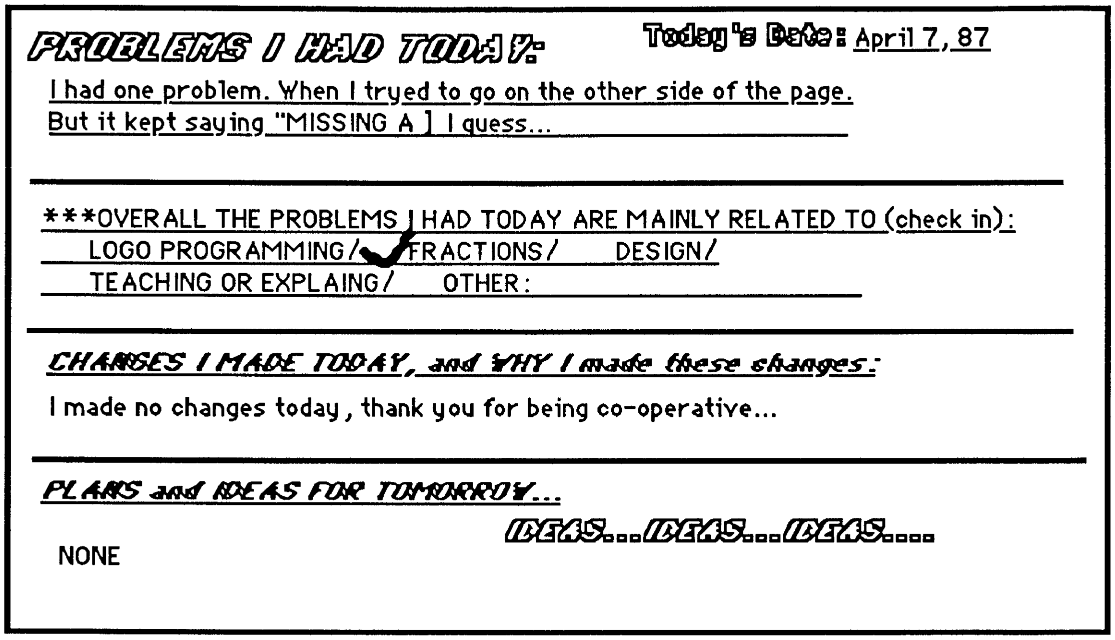
**Debbie:** "That's what it's for."

**I:**"Aha...Are you planning to try it out now?"

**Debbie:** "Yes."

Debbie flipped the Page and tried to run her procedure TOPO, which of course did not work. This was because Debbie was still working within her FRACT2 Page. The procedure BIG was included within it, and its first sub-procedure QUEST was also included, though the rest of the sub-procedures in BIG were not. So Logo, after executing QUEST, gave her a message about not knowing how to do "HALF." Debbie was puzzled for a long time after this. I decided not to help her right away and let her think about it for a while before helping her. I stepped away and watched another child program. It was the end of the computer session, and Debbie had to wait to return to the class before she managed to solve that problem and realized that it was not a good idea to use the same procedure name for two different procedures (i.e., the two INTRODUCTION procedures). She discovered this on the following day.

**In the Reflections Form of April 7 Debbie wrote:**



PROBLEMS I HAD TODAY: Today's Date: April 7, 87

I had one problem. When I tryed to go on the other side of the page. But it kept saying "MISSING A ] I guess...

***OVERALL THE PROBLEMS I HAD TODAY ARE MAINLY RELATED TO (check in):
LOGO PROGRAMMING/✓FRACTIONS/ DESIGN/
TEACHING OR EXPLAING/ OTHER:

*CHANGES I MADE TODAY, and WHY I made these changes:*

I made no changes today, thank you for being co-operative...

*PLANS and IDEAS FOR TOMORROW...*

IDEAS...IDEAS...IDEAS....

NONE

In this Form, Debbie only reported on one problem she had initially had with one of the brackets in her new INTRODUCTION procedure (i.e., MISSING A ] IN INTRODUCTION). I do not know why she did not report on the last problem she encountered while she was running her procedure TOPO, which included BIG and INTRODUCTION.

Strangely enough, although she did make some changes in her program on April 7, under the part that reads: "*CHANGES I MADE TODAY and WHY I made these changes*" she wrote **"I made no changes today."** But she also added: **"Thank you for being so co-operative..."** What did she mean by that sentence? Who was she referring to? Was she talking about the computer's being cooperative, or was she being sarcastic and really commenting on the fact that the computer had not been so very cooperative during that session, especially towards the end? Perhaps she was talking to her Designer's Notebook, which was always cooperative and never gave her any trouble. Or perhaps she was referring to my being cooperative or, on the contrary, if she was being sarcastic, not cooperative at all. Her reasons for writing this sentence remain unknown to me. I invite the reader's interpretation of this "mystery."

In summary, one of the important aspects of the Instructional Software Design Project, as a whole, was its offering Debbie a learning environment that, by its nature, forced her to combine new Logo knowledge and skills with old. (In fact, this is one of the qualities of the processes of programming in general, Papert, 1980, 1986). By analyzing Debbie's software-construction processes, we have seen how she discovered that there were many ways to program the same thing; and also, as she learned a new skill, how she re-programmed an old screen or modified her old procedures using the new skills she had learned (for example, Debbie re-programmed her CALF procedure after learning how to use 1+random 14 for randomming the colors of the shapes); she also learned to connect old procedures with new (for example, Debbie connected HALF with CALF, and later connected both with HOUSE, and then with TWO, and a month later with QUEST); and she used the same parts or procedures in several super-procedures--for example, having so far created two super-procedures BIG and FRACTIONS, she would discover in the following weeks additional new ways to combine the old parts with new parts for different purposes, creating even more super-procedures.

In the more conventional learning situations in her school life, Debbie would usually learn something for a time and later put it aside when the assignments were completed or when the related tests were over: she would start learning something else, put it aside; learn another new thing, put it aside; and so on. Most of the time, Debbie's individual pieces of knowledge would remain unrelated to one another, never to be properly integrated in her mind or her learning experiences. If we believe that Debbie's knowledge and learning abilities were a result of the conditions in which she was being educated, it seems that these routines must have been an obstacle and a negative influence on her style of thinking, her capacity for knowledge, and her attitude towards her own knowledge and thoughts in general. I think that one risk resulting from the learning conditions under which Debbie was studying most of her subjects in school, was that knowledge of these subjects becoming fragile, local, limited, and neither fluent, flexible, connected, nor integrated (e.g, this is what A. diSessa (1986) calls "Knowledge In Pieces," D. Perkins (1985) calls "Fragile Knowledge," or what Gestalt Psychologists in the 40's described as rigidity vs. flexibility in knowing and learning).

However, in the long, integrative, and complex Software Design Project, Debbie could not separate her knowledge about fractions from her knowledge of Logo; nor could she put aside her old knowledge of those subjects. Instead, she constantly had to put this knowledge into use and connect it with the new. In order to make sense of the software as a whole during the four-month period of the Instructional Software Design Project, Debbie needed to learn to connect the old pieces of knowledge with the new, or modify the old parts of her program by using her new knowledge. Several examples of this new ability have already been given, and more will be presented, for in April, May, and June Debbie, having learned many things along the way, went back to the procedures and representations she had programmed in March, revised them, or simply re-used them in new contexts.

## 10. DESIGNING AND REPRESENTING EQUIVALENT FRACTIONS OF HALVES

In her Designer's Notebook, Debbie drew many designs for her new screens about equivalent fractions of which only two were actually implemented by her during the first three weeks of May. Debbie managed to fully complete one screen, and told me that she had created this screen for "Teaching that six-twelfths equal one-half, two-fourths equal one-half, four-eighths equal one-half, and ten-twentieths equal one-half." I shall describe her processes of designing and implementing this screen in Section **10.1.** However, the second screen about equivalent fractions, which she started to work on during the third week of May, was never completed. I shall briefly describe that second incomplete screen in Section **10.2**, and the reasons why she did not complete it. Then, in section **10.3**, I shall analyze in detail one of Debbie's equivalent fractions designs, the one from April 14, comparing it with another girl's, Naomi. This particular design is interesting from the point of view of imitation and cultural learning.

### 10.1. Equivalent Fractions: Screen Number 1

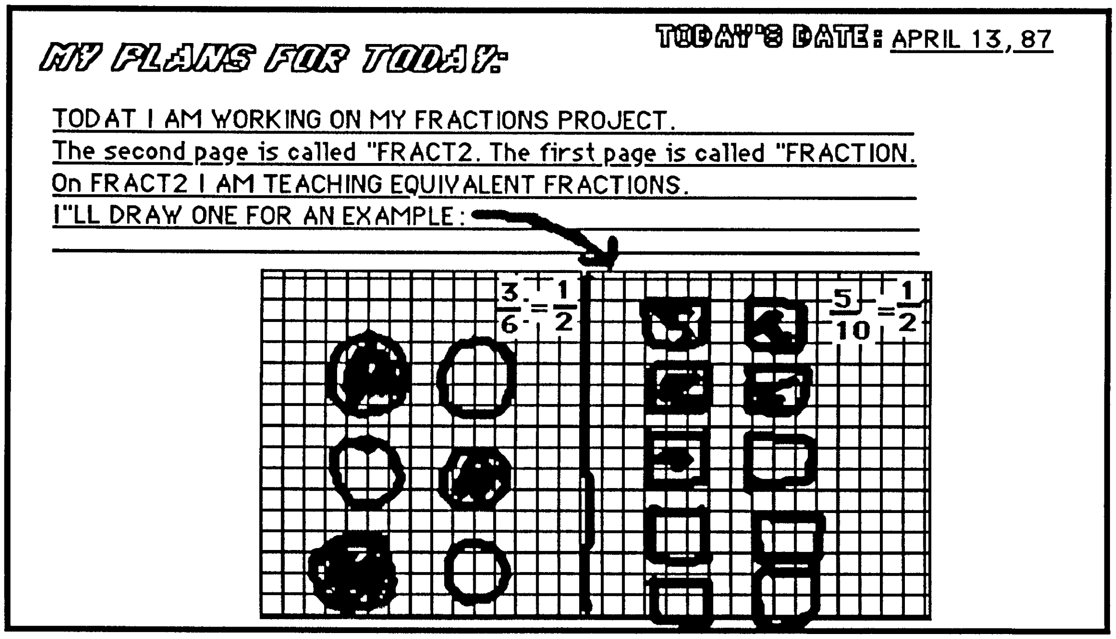
As previously stated, Debbie, as early as April 7, announced for the first time her plan to create a screen for equivalent fractions, which is quoted in the following subsection.

#### 10.1.1. APRIL 7 PLAN:

"TODAY I WILL DO SOMETHING ELSE. I CAN'T FIT ANY MORE WORK ON MY "FRACTIONS PAGE. SO I'M GONNA DO A NEW PAGE CALLED "FRACT2. **I'LL DO EQUIVALENT FRACTIONS ON "FRACT2.** I MIGHT DO SOME MORE POEMS. IF I HAVE TIME I'LL ADD MUCH MORE STUFF ON OTHER PAGE, THAT I'VE LEARNED" [my emphasis].

Throughout the months of April and May Debbie wrote, planned, designed, re-planned, and re-designed in her Notebook a series of screens for equivalent fractions, which she meant to implement in "Frac2," her new Logo Page. I shall now briefly review several of the written plans and hand-drawn designs created by Debbie from April 7 through May, which pertain to her work on equivalent fractions. The reader should realize that during these months Debbie was also working, in parallel, on many other screens and procedures. Some of this parallel work is reflected in her written plans: in almost all of them Debbie wrote about two or three things she was planning to accomplish on the same day. It is very difficult to describe her parallel thinking and programming processes, or her simultaneous accomplishments in this linear medium of writing. As stated previously, the more complex and large Debbie's Project grew, the more complex her actions and thoughts became, and as a result, it became really difficult to describe and analyze in detail her phases and processes during these months.

### 10.1.2. APRIL 13 PLAN:



MY PLANS FOR TODAY:

TODAY'S DATE: APRIL 13, 87

TODAT I AM WORKING ON MY FRACTIONS PROJECT.
The second page is called "FRACT2. The first page is called "FRACTION.
On FRACT2 I AM TEACHING EQUIVALENT FRACTIONS.
I"LL DRAW ONE FOR AN EXAMPLE:

In this plan, we can see that Debbie represented for the first time a representation of the fraction tenths. She used a group of discrete geometrical objects (squares for the tenths and circles for the sixths). I passed by her desk in the classroom, and stopped when I saw her drawing this. She looked at me and said:

**Debbie:** "Equivalent fractions." [she smiled]

**I:** "Yes. I see."

**Debbie:** "I'm going to show all the fractions in the world like this." [she said in a proud voice]

**I:** "All the fractions in the world?!"

**Debbie:** "Well not all of them, but many. All the halves."

**I:** "All the halves? What do you mean?"

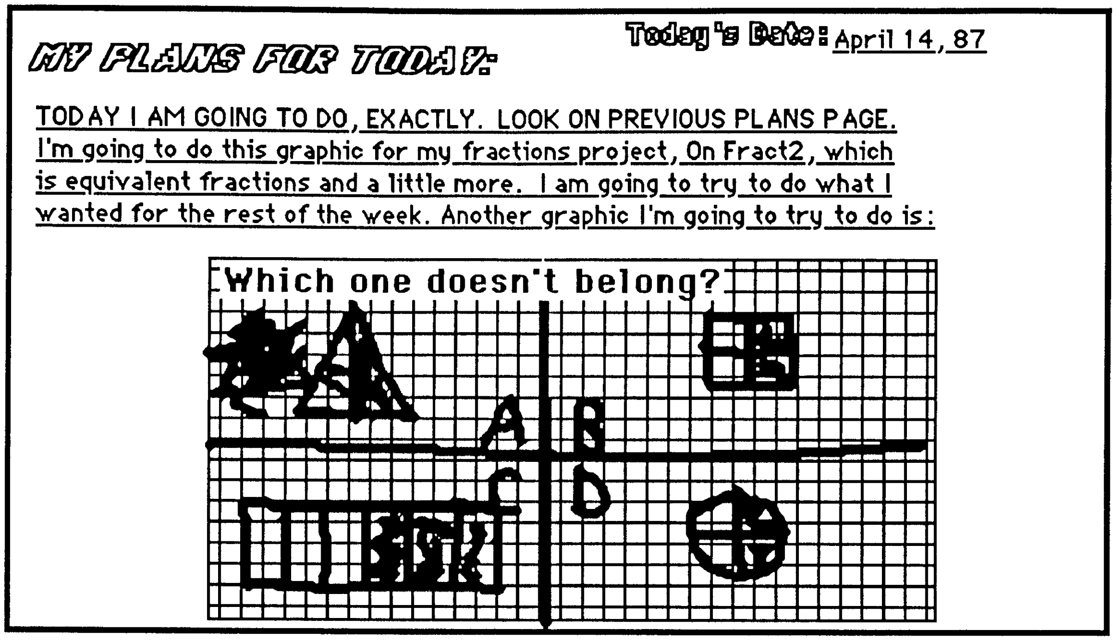
**Debbie:** "You know. Sixths, three-sixths. Fourths, two-fourths. Eighths, four-eighths. Tenths, five-tenths. Sixteenths, eight-sixteenths. Eighteenths, nine-eighteenths. Twentieths, ten-twentieths. All of those at least."

**I:** "It sounds like a lot of work, Debbie! Good luck!"

**Debbie:** "Then, I'm gonna teach each one alone on the screen, and then, all of them together, to show all of them, and that they all equal one-half!"

No doubt Debbie had a very strong concept of equivalences of halves. In her answer to my question, as described above, she demonstrated great ease at verbalizing and moving within the half equivalence class and she included in her statements many fractions of this class (i.e., fourths, sixths, eighths, tenths, twelveths, sixteenths, eighteenths, and twentieths). She was also able to divide these fractions into halves very freely. Furthermore, she had an interesting idea on how to teach it: by showing her users several examples of equivalences of halves, one by one, and then putting them all on the same screen. I think Debbie's initial attempt to represent "all the fractions in the world like this" is very interesting: it shows that she felt more secure about fractions in general. She was no longer intimidated by them. She spoke with greate assurance and gave the impression that she was in control and really meant to do it. In the following plans and drawings we will see that Debbie did indeed make serious attempts to represent as many equivalences of halves as possible.

## 10.1.3. APRIL 14 PLAN:



MY PLANS FOR TODAY: Today's Date: April 14, 87

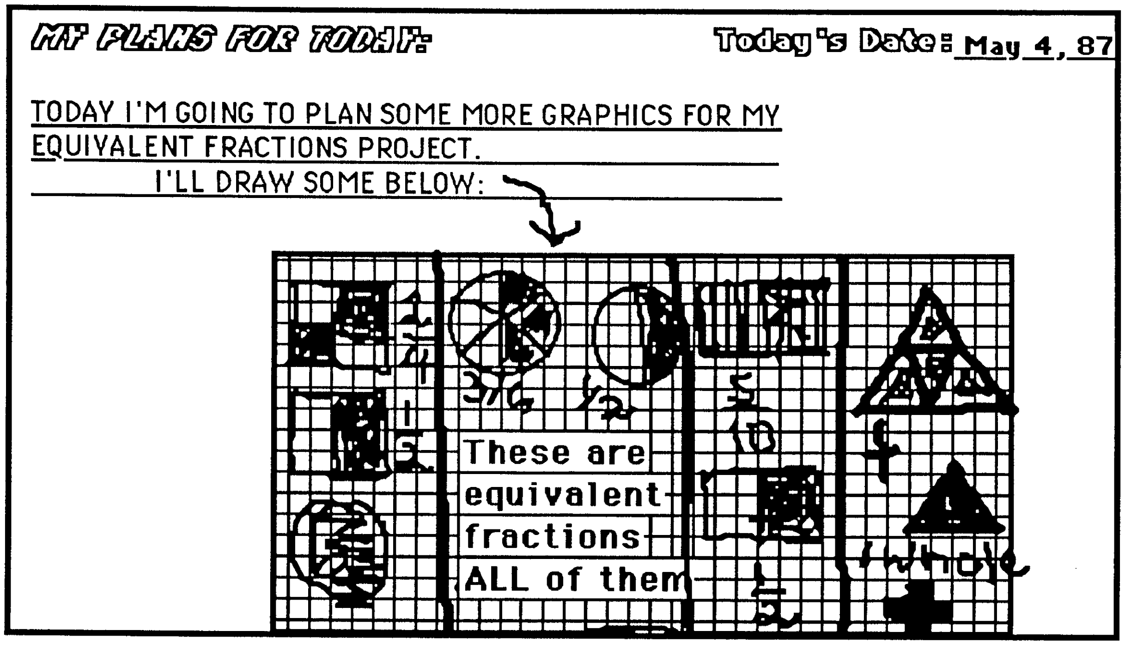
TODAY I AM GOING TO DO, EXACTLY. LOOK ON PREVIOUS PLANS PAGE.
I'm going to do this graphic for my fractions project, On Fract2, which is equivalent fractions and a little more. I am going to try to do what I wanted for the rest of the week. Another graphic I'm going to try to do is:

This particular plan will be discussed in detail later, in Section **10.3.**

We shall now progress with Debbie to the month of **MAY**, during which she continued to work on her screens for equivalent fractions, and in parallel, on several other screens and programming problems.

## (EQUIVALENT FRACTIONS, CONT.)

### 10.1.4. MAY 4 PLAN:



MY PLANS FOR TODAY: Today's Date: May 4, 87

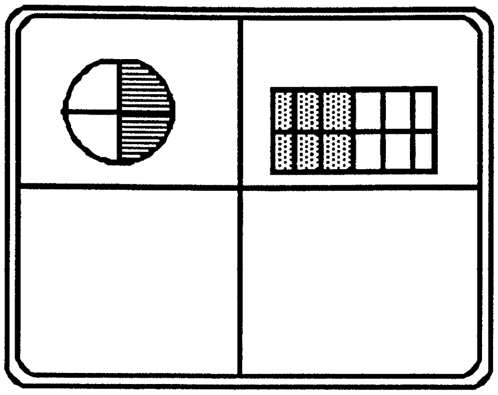
TODAY I'M GOING TO PLAN SOME MORE GRAPHICS FOR MY EQUIVALENT FRACTIONS PROJECT.
I'LL DRAW SOME BELOW:

3/6
1/2
These are equivalent fractions ALL of them
5/10
whole

### 10.1.5. MAY 5 PLAN:

"TODAY, NO MATTER WHAT, I'M WORKING ON EQUIVALENT FRACTIONS. BUT BEFORE THAT I'M GOING TO DO THE TEST FOR WHAT A FRACTION IS. LOOK ON PREVIOUS 'MY PLANS FOR TODAY' SHEETS FOR THE EQUIVALENT FRACTIONS GRAPHICS. BYE!!!"

Debbie did begin to implement her Equivalent Fractions screen on that day. (She also worked on "the test for what a fraction is" but it will be described later.) Here is what she succeeded in programming on the computer for equivalent fractions:

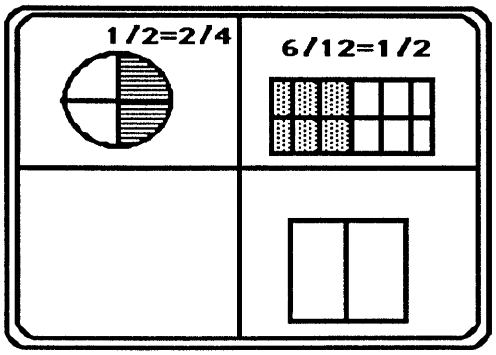


**May 5, Implementation Step 1:**

### 10.1.6. MAY 11 PLAN:

"TODAY I AM GOING TO DO EQUIVALENT FRACTIONS FROM MY DISK. THE NEXT ONE I'LL DO IS ADDING AND SUTRACTING IN ONE OR ON ONE PAGE. I'M GOING TO USE WHAT I LEARNED TODAY IN IT. !!!!!!! BYE !!!!!!!"

On May 11, Debbie continued to implementat the above screen from May 5. But in her plan we can see that in parallel to her thinking about representations for equivalent fractions, she generated a new plan for representing addition and subtraction of fractions. Here is what she succeeded in programming for equivalent fractions on May 11:

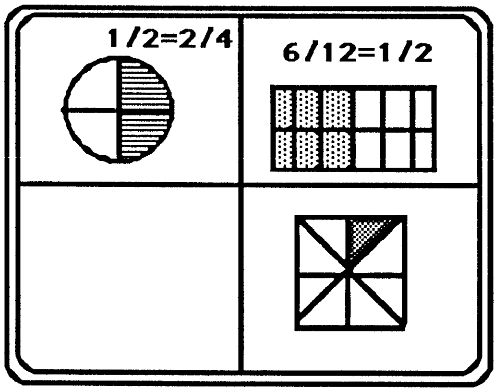


**May 11, Implementation Step 2.**

**10.1.7. MAY 18 PLAN:**

"TODAY I AM GOING TO WORK A WHOLE LOT MORE ON EQUIVALENT FRACTIONS. BUT I'M ALSO GOING TO MAKE THE EXAM. IN OTHER WORDS, THE TEST. I'M FINISHING THE ADDITION ASSIGNMENT FIRST. SEE YOU SOON!" [no design was created that day].

Here is what Debbie accomplished on the computer on May 18:



**May 18, Implementation Step 3:**

**10.1.8. MAY 20 PLAN:**

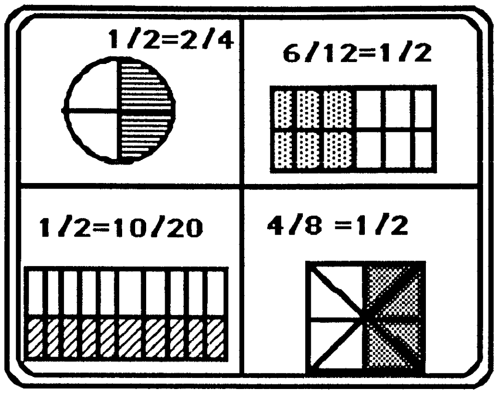
"TODAY I'M GOING TO DO, WHAT I CAN, FOR THE REST OF MY EQUIVALENT FRACTIONS PROJECT. THEN, IF THERE'S TIME LEFT I'M GOING TO EXPLAIN WHAT I DID IN THE ADDITION OF FRACTIONS ASSIGNMENT. !!!! BYE !!!!" [no design was created that day].

(Debbie did not add anything to her Equivalent Fractions procedure. Instead she worked on her Addition scene, which will be described later).

**10.1.9. MAY 22 PLAN:**

"TODAY I'M WORKING ON MY EQUIVALENT FRACTIONS PROJECT. I'LL FIX IT AND FINISH IT, IF I CAN!!! I'M GONNA ALSO FINISH ADDING FRACTIONS. !! BYE !!"

On May 22, Debbie finished her procedure for representing equivalent fractions. Her final screen is represented in the following diagram:



**May 22, Implementation Step 4:**

Debbie's Reflection Form indicates that she was very happy and satisfied with her completion of this screen. On May 22 she wrote in her Designer's Notebook the following:

MAY 22, REFLECTIONS:

"TODAY I HAD NOT ONE LOUSY PROBLEM! I'M GLAD TOO!!! I FINISHED ONE EQUIVALENT FRACTIONS, BUT NOT THE OTHER THAT I PLANNED."

We can see that Debbie has in mind another screen for equivalent fractions (i.e., "But not the other one that I planned"). She will implement it in the next few days.

The following is Debbie's procedure "EQUIV" for the representation of equivalent fractions [with my interpretation of the code in brackets:]

```
TO EQUIV
PU HOME SETC 13                      [changes pen color to purple]
PD FD 190 PU HOME                    [cuts the screen in the middle, vertically]
RT 90 PD FD 320                      [cuts the screen in the middle, horizontally]
PU SETPOS [20 10] SETC 1 PD          [positions the Turtle in the top right of the screen]
REPEAT 2 [FD 60 RT 90 FD 120 RT 90]  [draws a purple rectangle in the top right]
```

| | |
|---|---|
| LT 90 BK 20 RT 90 FD 60 | [starts cutting the rectangle vertically into six equal parts....] |
| RT 90 FD 20 RT 90 FD 60 LT 90 FD 20 | |
| LT 90 FD 60 RT 90 FD 20 RT 90 FD 60 | |
| LT 90 FD 20 LT 90 FD 60 RT 90 FD 20 | |
| RT 90 PU FD 30 RT 90 PD FD 120 | [cuts the rectangle horizontally, in the middle, which creates twelve-twelfths] |
| PU HOME | |
| PU SETPOS [30 30] SETC 9 PD FILL | [shades in blue one-twelfth] |
| REPEAT 2 [PU FD 20 PD FILL | |
| RT 90 PU FD 20 PD FILL] | [shades in four more twelfths; five are shaded now] |
| PU RT 90 FD 20 PD FILL | [shades the sixth twelfth in blue] |
| PU HOME | |
| PU SETPOS [15 80] LABEL [6\/12 = 1\/2] | [prints 6/12=1/2 in yellow over the rectangle] |
| PU HOME | [finishes the representation for **sixth-twelfths equals one-half]** |
| PU SETPOS [-60 50] SETC 3 | [positions the Turtle in the top left of the screen] |
| PD REPEAT 36 [ FD 7 LT 10] | [draws a turquoise circle in the top left of the screen] |
| LT 90 FD 80 | [cuts the circle horizontally in its middle] |
| BK 40 RT 90 FD 43 BK 80 | [goes back the length of the circle's radius, turns right ninety degrees and cuts the circle vertically in its middle. The circle is now divided into fourths] |
| PU HOME | |
| SETC 11 | [changes the pen color to light blue] |
| PU SETPOS [-75 65] | [positions the Turtle at top right fourth of the circle] |
| PU BK 10 PD FILL | [shades one-fourth of the circle in light blue] |
| PU BK 20 PD FILL | [shades the circle's bottom-right fourth in light blue] |
| PU HOME | |
| PU SETPOS [-65 80] | [positions the Turtle over the circle] |
| LABEL [1\/2=2\/4] | [prints the equivalence **one-half equals two-fourths]** |
| PU HOME | |
| SETC 4 PU SETPOS [25 -80] | [changes pen color to red, positions Turtle in the |

```
                                            bottom right of the screen]
PD REPEAT 4 [FD 50 RT 90]                   [draws a red square]
FD 25 RT 90
FD 50 BK 25 LT 90 FD 25 BK 50 FD 25         [cuts the circle horizontally, then vertically]
RT 45 FD 32 BK 64 FD 32 LT 90 FD 32         [cuts the circle diagonally twice, creates eighths]
BK 64
PU HOME
SETC 12 PU SETPOS [60 -35] PD FILL          [changes pen color to orange, fills in one-eighth]
PU BK 5 RT 90 PU FD 10 PD FILL              [shades two more eighths in orange]
RT 90 PU FD 30 PD FILL                      [shades the forth eighth in orange]
PU HOME
PU SETPOS [45 -15] LABEL [4\/8=1\/2]        [prints the equivalence, four-eighths equal
                                            one-half]
PU HOME

SETC 2 PU SETPOS [-140 -85]                 [changes pen color to green, positions Turtle the
                                            bottom left of the screen]
PD REPEAT 2 [FD 60 RT 90 FD 120 RT 90]  [draws a green rectangle]
PD REPEAT 10 [SETH 90 FD 12 LT 90 PD FD 60 BK 60] [cuts it vertically into ten-tenths]
FD 30 LT 90 FD 120                          [cuts the rectangle horizontally, which creates
                                            twenty-twentieths]
PU HOME
SETC 10 PU SETPOS [-135 -40]                [changes pen color to light green, positions Turtle
                                            in the bottom left twentieth of the rectangle]
PD FILL PU RT 90 FD 10 PD FILL              [shades in light green one twentieth after another]
PU FD 15 PD FILL PU FD 10 PD FILL
PU FD 10 PD FILL PU FD 15 PD FILL
PU FD 10 PD FILL PU FD 10 PD FILL
PU FD 12 PD FILL PU FD 15 PD FILL           [finishes shading ten-twentieths in light green]
PU HOME
SETPOS [-105 -15] LABEL [10\/20 =1\/2]      [prints the equivalence, ten-twentieths equals
                                            one-half]

END
```

Debbie's EQUIV procedure is very readable, and is efficiently and beautifully organized. She uses the "PU HOME" to break between each component within a given representation (e.g., separating the drawing the outline of the rectangle and the cutting it into sixths); and she also uses "PU HOME" between each of her representations, separating them from each other. She uses SETPOSitions freely. In fact, this particular screen division (into four parts) was often used by her in her software. Debbie found this method convenient for showing multiple representations on one screen; but it also enhanced her knowledge of the Cartesian coordinate system. I asked her about it:

**I:** "How does it work with the pluses and minuses in the different SETPOS?"

[Even though my question was not phrased very clearly, Debbie understood exactly what I meant]

**Debbie:** [she pointed on the screen, moving her finger on the screen's four fourths counter-clock-wise, and answered very quickly] "plus and plus, plus and minus, minus and minus, minus and plus."

In the EQUIV procedure Debbie also properly used REPEATs for the rectangles, the square, and the circle; furthermore, her solution for cutting the rectangle representing ten-twentieths was very elegant (e.g., REPEAT 10 [seth 90 fd 12 lt 90 pd fd 60 bk 60]).

I asked her why she had not used this set of commands for the top rectangle representing six-twelfths. She told me the following:

**Debbie:** "I did it [the rectangle which represents twelfths] a long time ago and I didn't think about it then, about how to do it like this. I only found out about it today."

**I:** "How did you find it?"

**Debbie:** "I tried and tried and tried until I found out about it."

**I:** "Just like that..."

**Debbie:** "Yea."

**I:** "But how??"

**Debbie:** "I <u>had to</u> do it here, because there are many lines to divide it to [i.e., to divide the rectangle into twenty parts] so it shouldn't take too long to write it this way."

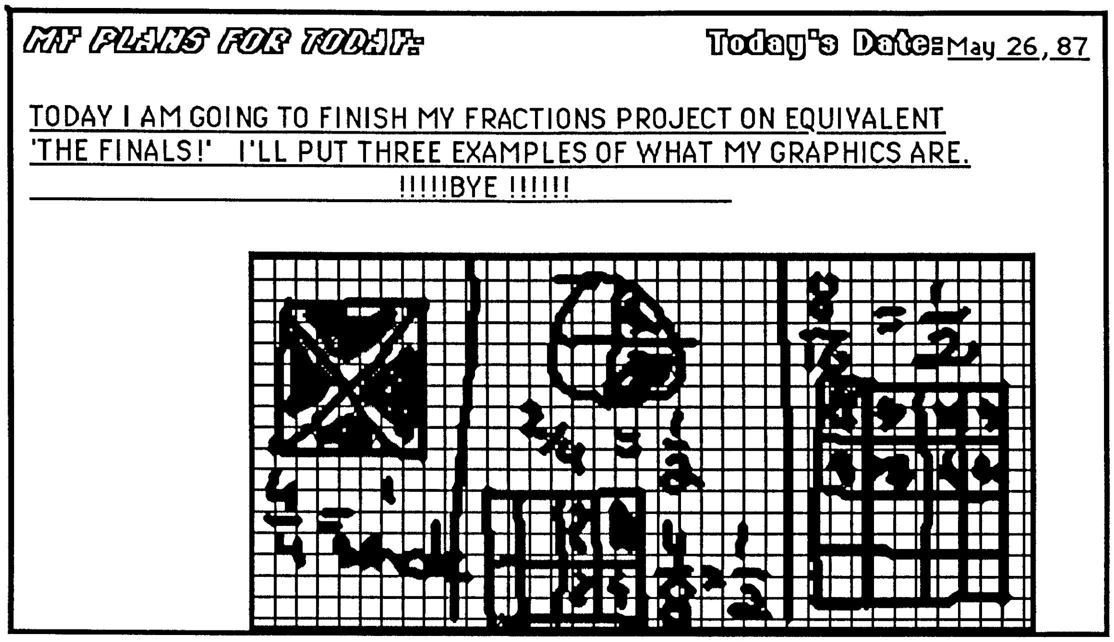
Debbie's answer indicates that she had discovered this elegant code (for dividing the rectangle into twentieths) by trial and error (i.e., "I tried and tried and tried") because she wanted to avoid writing a long and repetitive piece of code. This line of code, as well as her procedure EQUIV as a whole, shows, to my mind, a great jump in Debbie's knowledge of programming.

In addition, Debbie's use of colors is interesting. She matches the colors of the geometrical shapes' outlines with the colors of their shaded-in parts: she paints the outline of the square that represents four-eighths in red, and its shaded-in parts in orange (which on the screen looks like light red); the rectangle that represents six-twelfths she painted in purple, and its shaded-in parts in bluish-purple; the outline of the rectangle representing ten-twentieths is dark green, and its shaded parts light green; finally, she painted the circle representing two-fourths in torquoise, and its shaded-in parts in light blue. In other words she did not pick the colors randomly, but she used her esthetic judgment to decide which colors to use and what sets of colors to use together for one shape.

## 10.2. Equivalent Fractions: Screen Number 2

Apparently Debbie considered her representing of equivalent fractions as a mini project within the whole Project. In what she called the "Equivalent Fractions Project," she wanted to show as many representations as possible for equivalences of halves. Her conversation with me on April 13, her Reflections Form of May 22, and her Plans Form of May 26 indicate this clearly. However, the school year was almost over, and Debbie was not able to accomplish her grandiose plans. During the last three weeks of the Project, she was very concerned with other issues of optimization, re-organization, interactivity, and presentation of her final piece of software. In the following subsections, I shall briefly present the other screen for equivalent fractions that Debbie designed and partly implemented. This screen never became part of her final piece of software because she did not manage to finish it.

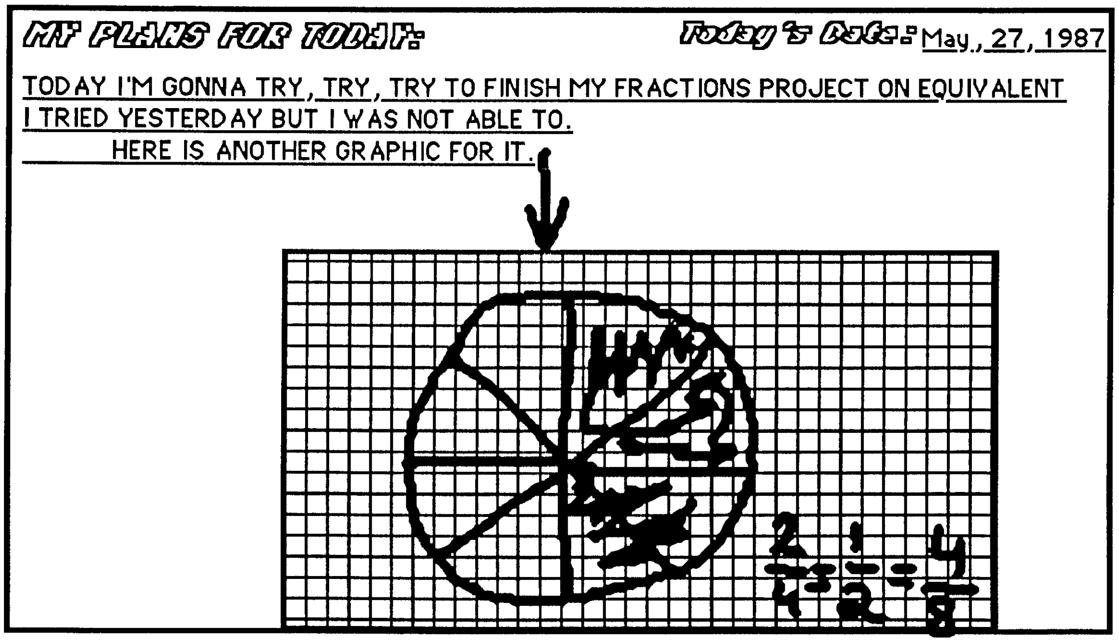
**10.2.1. MAY 26 PLAN:**



MY PLANS FOR TODAY: Today's Date: May 26, 87

TODAY I AM GOING TO FINISH MY FRACTIONS PROJECT ON EQUIVALENT 'THE FINALS!' I'LL PUT THREE EXAMPLES OF WHAT MY GRAPHICS ARE.
!!!!!BYE !!!!!!

**10.2.2. MAY 27 PLAN:**



MY PLANS FOR TODAY: Today's Date: May., 27, 1987

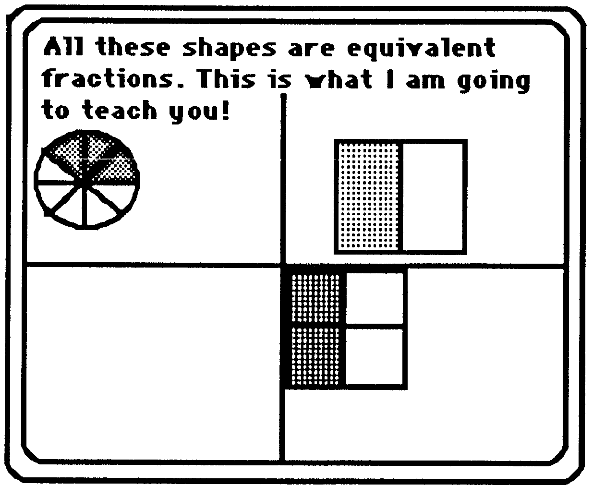
TODAY I'M GONNA TRY, TRY, TRY TO FINISH MY FRACTIONS PROJECT ON EQUIVALENT
I TRIED YESTERDAY BUT I WAS NOT ABLE TO.
HERE IS ANOTHER GRAPHIC FOR IT.

The following is Debbie's incomplete Logo code for her second screen on equivalent fractions with my interpretations in brackets:

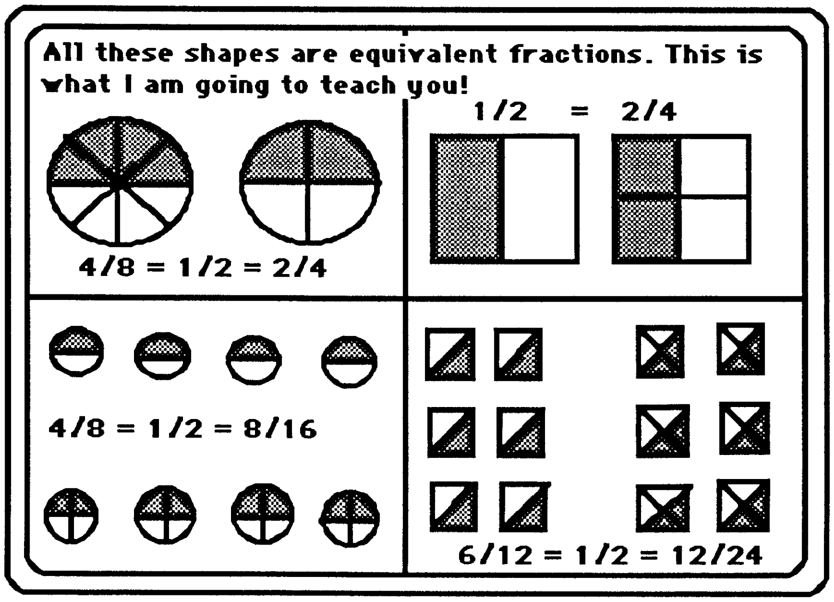
```
TO EQUIV2
SETC 11 PD FD 200 PU HOME               [divides the screen vertically in light blue]
RT 90 SETC 10 PD FD 320 PU HOME         [divides the screen horizontally in light green]
SETC 4 PU SETPOS [15 55]                changes the color to red, positions the Turtle in the
                                        top right of the screen]
PD REPEAT 4 [RT 90 FD 50]               [draws a red square]
RT 90 FD 25 RT 90 FD 50 BK 25           [divides the square into two halves]
PU RT 90 FD 10 SETC 12 PD FILL          [shades the left half in orange]
PU HOME

SETC 12 PD REPEAT 4 [RT 90 FD 50]       [draws an orange square]
PU RT 90 FD 25 RT 90 PD FD 50           [divides it into fourths]
PU BK 25 LT 90 PD FD 25 BK 50
PU SETC 4 LT 90 FD 14.1/2 RT 90
PU FD 5 PD FILL                         [shades in red the top-left fourth of the square]
PU RT 90 FD 15 PD FILL                   [shades in red the bottom-left fourth of the square]
PU HOME

PR [ALL THESE SHAPES ARE EQUIVALENT
FRACTIONS. THIS IS WHAT I AM GOING TO
TEACH YOU!]                             [prints the instructional statement]
PU HOME

SETC 5 PU SETPOS [-155 35]              [changes the pen color into light purple, positions
                                        the Turtle at the top left of the screen]
PD REPEAT 180 [ FD 1 RT 2]              [draws a purple circle]
PU RT 90 PD FD 56                       [divides the circle into halves]
BK 28 LT 90 FD 28 BK 56 FD 28 RT 45     [divides the circle into fourths]
FD 28 BK 56 FD 28 LT 90 FD 28 BK 56     [divides the circle into eighths]
```

```
PU HOME SETC 13                        [changes the pen color to pink]
PU SETPOS [-120 50]                    [positions the Turtle in one of the circle's eighths]
PD FILL RT 90 PU FD 10 PD FILL         [shades two of the eighths in pink]
PU BK 20 PD FILL                       [shades another eighth in pink]
END
```

**Debbie's screen was left unfinished at the end of the Project:**


All these shapes are equivalent fractions. This is what I am going to teach you!

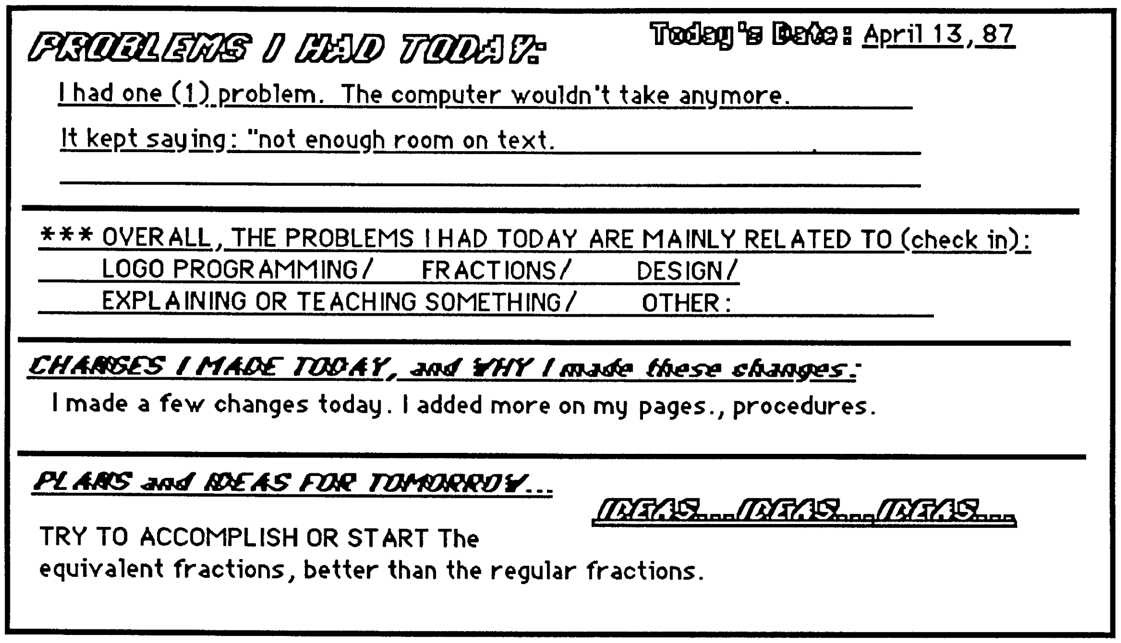
**According to my conversation with Debbie, this is how she wanted this screen to look at the end, when finished:**


All these shapes are equivalent fractions. This is what I am going to teach you!

During April and May, concurrently with her "Equivalent Fractions Project," Debbie continued to work on her two Logo Pages, "Fraction" and "Fract2." She connected and disconnected procedures from one another, created and eliminated procedures from her super-procedures, and so on. Debbie began to grow relatively more involved in the programming aspect of her software, rather than mainly in the learning of fractional concepts or in the designing of fractional representations. She became much more involved in moving procedures from one Logo Page to another, creating new super-procedures, and discovering new combinations of sub-procedures within those super-procedures; she concentrated much more on the user's interaction with her software, and added short introductions and instructional sentences here and there; she was also very involved in finding ways to connect the different Logo Pages. Her processes of doing this, although incredibly interesting and complex, will be not be described in this thesis.

On April 13, for example, she deleted some procedures from her Fract2 Page, and decided to leave only the following: Introduction2, Wait.for.user and Ignore, Equiv, More, and Understanding. On her Fraction Page she included: Half, Calf, House, Wait.for.user and Ignore. She also had her super-procedure, "Fractions," which included: Half, Calf, Two, House, Exam, Question1, and Quest. Later in the month of April Debbie created another super-procedure, which she named "Okay." Okay became a component in the procedure Exam; and Exam asked the user whether or not he remembered what he had learned about fractions so far. If the user answered that he remembered nothing, Okay made it possible for him to **go over** the materials learned (Debbie chose to go over four procedures, in the following order: House, Half, Calf, and Two); if the user answered that he did remember everything he had learned, the procedure Super was called to provide **additional new information**. Debbie connected this to Gettools fract2, Introduction, Equiv, More, and Understanding. While she was working on the various exams, testing questions, and quizzes, Debbie used many of her existing procedures, re-organized them in different orders and sets, and added little pieces of code where she felt, they were needed.

In general, during April and May, there were many discrepancies between Debbie's plans in her Notebook and their implementation (or non-implementation) at the computer; that is, she constantly planned to do equivalent fractions, but every day, after she logged-in and ran her software on her computer, she decided to do other things related to fitting and connecting her pieces of code together. Still, she kept drawing and planning many screens for equivalent fractions, but in her Reflections Forms she constantly and accurately reported: "I didn't accomplish starting my equivalent fractions on fract2," or "I added more procedures," or "I added more things to my pages," or "I changed the order of my screens," or "I made a new super-procedure," etc. Her Reflections Form from April 13 offers a typical example of her reflections during this implementation phase:



PROBLEMS I HAD TODAY: Today's Date: April 13, 87

I had one (1) problem. The computer wouldn't take anymore.

It kept saying: "not enough room on text.

*** OVERALL, THE PROBLEMS I HAD TODAY ARE MAINLY RELATED TO (check in):
LOGO PROGRAMMING/ FRACTIONS/ DESIGN/
EXPLAINING OR TEACHING SOMETHING/ OTHER:

CHANGES I MADE TODAY, and WHY I made these changes:

I made a few changes today. I added more on my pages., procedures.

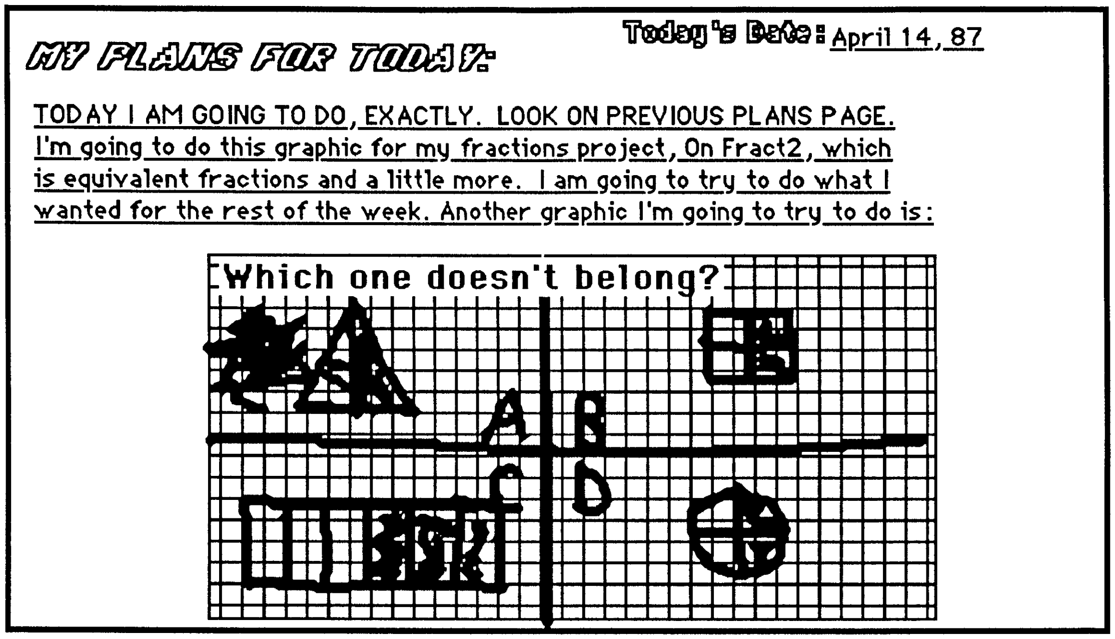
PLANS and IDEAS FOR TOMORROW... IDEAS... IDEAS... IDEAS...

TRY TO ACCOMPLISH OR START The equivalent fractions, better than the regular fractions.

In the following section I shall emphasize one of Debbie's many designs for equivalent fractions: it is the one she designed on April 14, the "Sesame Street Screen."

## 10.3 DEBBIE'S AND NAOMI'S "SESAME STREET SCREENS": AN ANALYSIS OF TWO DIFFERENT KINDS OF CULTURAL LEARNING IN CHILDREN

On April 14 Debbie wrote and drew the following in her Designer's Notebook:


MY PLANS FOR TODAY:

Today's Date: April 14, 87

TODAY I AM GOING TO DO, EXACTLY. LOOK ON PREVIOUS PLANS PAGE.
I'm going to do this graphic for my fractions project, On Fract2, which is equivalent fractions and a little more. I am going to try to do what I wanted for the rest of the week. Another graphic I'm going to try to do is:

Although Debbie never implemented this particular design in Logo, I choose to analyze it here because it provides us with an interesting comparison of the different qualities of imitation or cultural learning in two children. (Obviously, it is not possible to find which one of these four representations that Debbie drew in her Notebook "does not belong," since, in fact, they all represent halves).

In the following subsections I shall clarify the "cultural origins" of Debbie's design. For that purpose, I briefly describe the way Naomi, another girl from the Experimental class, created a similar design by imitating Sesame Street's famous scenario, "Which of these four belong together, and which one of these does not belong"; I also show how Debbie imitated this idea from Naomi's software, and the ways in which she transferred it, though without fully understanding the general structure of this scene from Sesame Street, or the fractional or instructional concepts that were embodied in her friend's screen. From analyzing the two girls' processes and products of imitation, I conclude that <u>two</u>

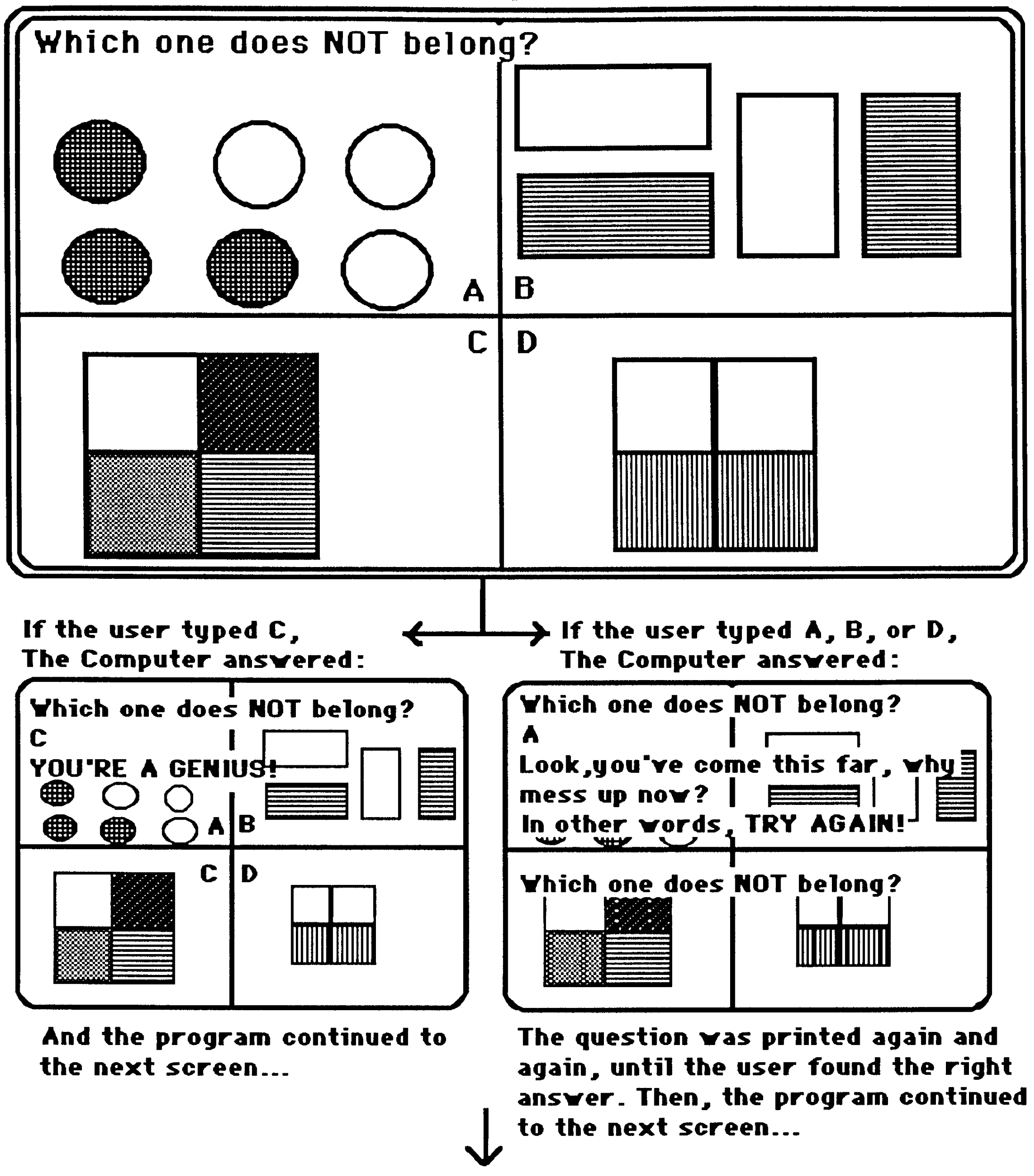
qualitatively different imitations were found. Through this comparison I offer a re-examination of Vygotsky's beliefs about the role of imitation in learning, as well as a re-evaluation of his use of the word "imitation," in terms of these two girls' processes and products.

### 10.3.1. Naomi's Screen

One girl from the Experimental class, Naomi, who was sitting next to Debbie during almost all of the computer sessions, had worked during the second week of April on a very original and creative screen which she had called the "Sesame Street Screen."

One of the instructional goals of the well-known Sesame Street television series (which is produced by Children's Television Workshop) is to teach children about classification, sorting, and grouping (Lesser, 1974). One treatment of this goal is to show children four objects, three of which have an attribute in common; the child is required to sort out the one "inappropriate" object out of the four presented to him, on the basis of: size (i.e., height, length, etc.); form (i.e., circular, square, triangular, etc.); function (i.e., to ride in, to eat, to read, etc.); or class (i.e., vehicles, animals, etc.). These scenes are designed so that children will be able to: observe and examine the properties of the four given objects, verbalize their reasons for grouping three of those together, say why they belong together, and therefore discover "which one does not belong" (e.g., Lesser, 1974, p. 67).

Naomi, on her own, spontaneously came up with the idea of imitating the popular Sesame Street scene that shows four different objects on the TV screen, of which three "go together," and one "does not belong." She successfully managed to appropriate it, use it in her teaching about fractional representations, and implement it in Logo. On April 10, Naomi finished her screen which is presented below:



Which one does NOT belong?

If the user typed C,
The Computer answered:

Which one does NOT belong?
C
YOU'RE A GENIUS!

And the program continued to the next screen...

If the user typed A, B, or D,
The Computer answered:

Which one does NOT belong?
A
Look,you've come this far, why
mess up now?
In other words, TRY AGAIN!

Which one does NOT belong?

The question was printed again and again, until the user found the right answer. Then, the program continued to the next screen...

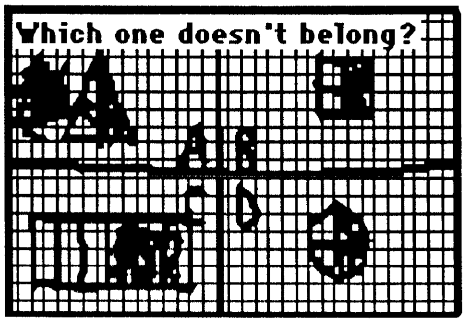
The minute Naomi announced "I am done! It works!" many of her friends, one by one (and Debbie among them), sat in front of her computer and tried it out. Naomi, unlike Debbie, was a very cheerful, talkative, enthusiastic, and successful child, both academically and socially; also, she was always willing to explain anything she did in a very detailed and articulate fashion. On April 10, I approached her, as the other children were doing, and said to her: "What a wonderful idea, Naomi!" and before I even had a chance to ask her anything, Naomi provided me with the following explanation of her screen design, which I find quite unique and delightful:

> "This is my Sesame Street Screen. [Long pause, she watches how one of her friends uses it. Then she looks at me:] It is tricky you know. I designed it so option A will really really trick them. Because, B, C, and D, are showing fourths [pointing on B, D, and C on the screen]: here, there are two-fourths in a group of four rectangles; here, two-fourths in the square; and here, three-fourths in another square. And, these [representations B, C, and D] are all showing pictures of squares and rectangles, so you'll think: 'That's it! I found it! [pointing at A with her left hand and at B, D, and C with three fingers of her right hand] These [B, D, and C] go together!' Confusing, Ha?! But Option A is showing six circles. Three-sixths are shaded in. So that you'll think right away that A doesn't belong [she imitated the voice of a younger child]: 'What are these circles and sixths doing here...' [she laughs] So you'll think, 'they definitely do not belong here!' But, if you look very carefully [she now speaks in a teacher's voice], you can see that A, B, and D are showing halves, and that C is the one that doesn't belong, 'cause it's showing three-fourths! And that's it!"

Naomi's detailed explanations speak for themselves. She was aware of the major strategies the Sesame Street instructional designers used in their scenes on TV; and also aware of their "tricking techniques." She was able to analyze and break apart the underlying instructional concepts of this scene, and to later re-construct it and make it suit her own purposes and instructional goals; she was also seriously considering her target users, what they might think, say, or do, as they solved the puzzle she had given them; and so on. (In fact, even before April 10, while she was still planning that screen and during its implementation phases, Naomi provided me with several quite similar, but not as detailed explanations for it. I shall also note here that throughout the Project, Naomi came up with many creative and unique screen designs, which should be analyzed and will be described in detail in my forthcoming case about her.)

### 10.1.2. Debbie's Design

At first, I was not aware of Debbie's design of April 14, since she had decided not to implement it in Logo. I discovered it in her Notebook three weeks or so after she had designed it; only then did I realize how much Naomi's "Sesame Street Screen" had influenced Debbie. Although Debbie, in her design of April 14, attempted to imitate the Sesame Street scenario, we shall now examine how much less sophisticated she was in her imitation and in her understanding of the Sesame Street instructional design techniques than was Naomi. For that purpose, let us examine Debbie's design once again:



Debbie, to use Naomi's words, did not "trick her users" at all. She created four simple representations, each one of them, in fact, representing halves: Option A represented a half of a triangle; B, of a square; C, of a rectangle; and D, of a circle. One needs to recall here that Naomi also focused on halves as the common attribute among the three representations that "went together," but that in Naomi's screen, the representation that "did not belong" was the one of three-fourths of a square. One also needs to recall that Naomi "tricked her users" by showing them three-sixths in option A, which in fact possessed the common attribute since three-sixths equal one-half. She managed to "trick" Debbie quite successfully. Debbie imitated that "trick of sixths" in her representation of three-sixths (in the rectangle at the bottom-left), but she considered it to be the item that "didn't belong," it being "three-sixths and not a half."

One day in May, I took Debbie out of the classroom. We sat together in the Teachers' Room and went through her Notebook. Among other things, we talked about that particular design. Our short conversation about this design is reported below, with my own comments in brackets:

**I:** "So which one doesn't belong, Debbie?

**Debbie:** "Can't you tell?!"

**I:** "Uhm..."

**Debbie:** "It is like Naomi's. You know..."

*[Note that Debbie did not say, "it's like in Sesame Street," rather, "it's like Naomi's"; in other words, she perceived herself as imitating Naomi's idea. She knew that I "understood" Naomi's screen, because she had seen us talking about it. She therefore referred me to Naomi's screen, and expected me to figure out her own design since it resembled Naomi's]*

**I:** "Oh...Yea...Now I remember, Naomi also made something like this."

**Debbie:** "I did not have enough time to do it on the computer, I made other things instead."

*[She puts her hand on her Designer's Notebook, attempting to turn that page and put an end to our conversation on that particular design].*

**I:** "Yes, I know that you created many other things."

**Debbie:** "I like it though."

**I:** "You like what?"

**Debbie:** "This design"

*[she points to her plan of April 14]*

**I:** "What do like about it?"

**Debbie:** "Do you know Sesame Street? It's like a quiz. They [on TV] always show you four different things and you have to guess which one doesn't belong."

*[Now she relates it to Sesame Street for the first time. She seems to generally understand the Sesame Street scenario; she also quietly mumbles the famous song from that scene on the TV show].*

**I:** "So...Which one [doesn't belong]?"

**Debbie:** "Don't you know about fractions??"

**I:** "Well...What do I have to know in order to solve it?"

**Debbie:** "You know..."

**I:** "Please tell me, help me..."

*[I say this in a pleading voice, and we both smile]*

**Debbie:** "C doesn't belong of course."

**I:** "Oh! I see...But, why?"

**Debbie:** "Because."

**I:** "What do you mean?"

**Debbie:** "C is three-sixths, so it doesn't belong."

**I:** "Oh... What <u>I thought</u> was different. It's really strange. I thought that C shows one-half, and A, B, and D, are also showing one-half. That's why I couldn't solve it!"

**Debbie:** "Yea..that's true...

*[Long pause, she examines her design once again; she puts her fingers on the picture, then looks towards the ceiling, and says]*

**Debbie:** "Yea..that's true too. But C is three-sixths." *[I see this as a sign of her rigidity]*

**I:** "And three-sixths do not belong... I see."

**Debbie:** "Naomi's screen is better, I guess."

*[She begins to feel insecure, but at the same time begins to break away from her rigidity]*.

**I:** What do you mean by 'better'? Why?"

**Debbie:** "I don't know..."

*[A very long pause. She seems a bit puzzled, as though unable to figure out why I am bothering her with this particular design, or wondering whether she has done something "wrong." Perhaps she is trying to recall Sesame Street scenes of this kind, or what Naomi did on her screen, and how she herself could have made her screen "better"]*

**I:** What are you thinking about?

*[No answer. I know that Debbie rarely answers questions of this kind, so I wait a few seconds, see that she is not going to answer, and continue:]*

**I:** "That's it. Now I got it! C doesn't belong because it shows three-sixths!"

*[I am trying to make her feel better and more secure...]*

**Debbie:** "...Or, nothing doesn't belong because they all show one-half !

*[She gives me a shy smile. She says "OR," meaning that she is only partially willing to play my game, and become only a tiny bit more flexible]*

"...Or... *[she pauses, looks towards the ceiling,]*

Or... I should change it...Uhm...I'll change the rectangle to fifths. To three-fifths. Now it really doesn't belong here."

*["Hey researcher, now I know what you were looking for!"]*

**I:** "You'll change it?...Why change!?"

**Debbie:** "Because A, B, and D show halves, and C doesn't. It shows fifths. If I change it, I mean here,"

*[She emphasizes the word "if." She then points to the rectangle. There is a pencil near her hand, but she does not use it. She does not actually change her original design. Apparently, the progress from thought to action is quite difficult for her in this context]*.

**I:** "Are you planning to change it? Are you planning to create it on the computer one day?"

**Debbie:** "NO." *[A very loud "NO"...]*.

"See...I drew it long time ago...Now I am making another scene. I don't think I should use this one at all."

*[pause]*

"Naomi already finished it. I am making another thing instead."

### 10.3.3. "Long Distance" and "Short Distance" Types of Imitation

Several issues need to be discussed in this context. Comparing Naomi's and Debbie's imitation processes, their designs, and the quality of their explanations of the designs, reveals interesting differences in their intellectual levels in general, and between their levels of understanding of the instructional design approach embodied in the Sesame Street scene in particular.

Naomi initiated the idea of imitating Sesame Street, and constructed a comprehensive design for it completely on her own, without the assistance of others, without demonstrations, and without a need for leading questions on my part; Debbie, on the other hand, imitated Sesame Street (via Naomi's demonstration); but in order to fully understand the concepts that were involved in that design, she needed my assistance and required many leading questions on my part in order to grasp the underlying structure of the concepts involved (i.e., of instructional design and fractional representations). According to Vygotsky (1978) imitation is an important factor in children's learning:

> "A full understanding of the concept of the zone of proximal development must result in re-evaluation of the role of imitation in learning. An unshakable tenet of classical psychology is that only independent activity of children, not their imitative activity, indicates their level of mental development....But recently psychologists have shown that a person can imitate only that which is within her developmental level (pp. 87-88)."

To my mind, Vygotsky's belief that "a child can imitate only that which is within his developmental level" does not provide us with a sufficient explanation of the real differences between Debbie and Naomi in this particular context. We have examined how the two girls were able to imitate Sesame Street: can we then say that they 1) imitated it in the same way? 2) were both on the same cognitive-developmental level (i.e., because both could imitate only what was within their developmental level)? I do not think so, for we also saw that one of the girls (Debbie) was indeed imitating an idea in the Vygotskian sense, whereas the other girl (Naomi), strange as it may sound, was imitating an idea in a strong constructivist fashion. The latter's processes and product of imitation were proven to be more systematic and comprehensive, whereas the former's were more superficial, limited, and incomplete.

I find Piaget's famous constructivist argument--that each time one prematurely shows a child something he could have discovered for himself, the child is kept from inventing it, and consequently, from understanding it deeply and completely--to be very relevant here; not in a teacher-pupil framework, but rather, in pupil-pupil framework. To my mind, both girls were equally familiar with the Sesame Street scenario (Debbie explained it very well during our conversation in the Teacher's Room); however, Naomi, in this case, in a way prevented Debbie from "discovering this idea for herself," to use Piaget's argument, and from understanding it completely. Debbie's "cultural" imitation was therefore not as powerful or deep as Naomi's "cultural" imitation of the same ideas.

In other words, Naomi discovered that she could borrow the Sesame Street classification scenario for her own instructional purposes. The Sesame Street scene existed in her "culture" although not directly in her school culture, and she did indeed imitate it, but it was a "long distance" imitation. Naomi's thoughts, actions, and explanations should be perceived and interpreted as processes of re-invention. She was the one who discovered the Sesame Street scene's relevance, its underlying structure and its "tricking" techniques (no one had ever told her about this), and she retro-fitted the concepts involved in these scenes for her own software in a systematic way and with a deep understanding of them. Moreover, she invented her own methods for using this instructional strategy in the context of her software about fractions. (Was it a higher-level imitation? I believe that it was a question of qualitative difference, and not just a difference in level.) In short, although Naomi did indeed imitate Sesame Street, I still believe that she was involved in invention.

Debbie, on the other hand, was involved in "short distance" imitation. Debbie did not discover the idea of borrowing the Sesame Street classification scene for her Project; she was not involved in a high-level reflection or paying attention to the characteristics of the Sesame Street scene and its "tricking techniques," or how they might fit into her software in the way Naomi's had. Instead, she liked Naomi's idea when she saw it on her computer; she was familiar with the idea and was generally able to understand it, since she knew that particular Sesame Street scenario very well; then, with minimal reflection and without a deep understanding of it, she designed an almost identical screen. But, in fact, she was rather confused about how to use it "properly" for the purpose of teaching about fractions and their representations.

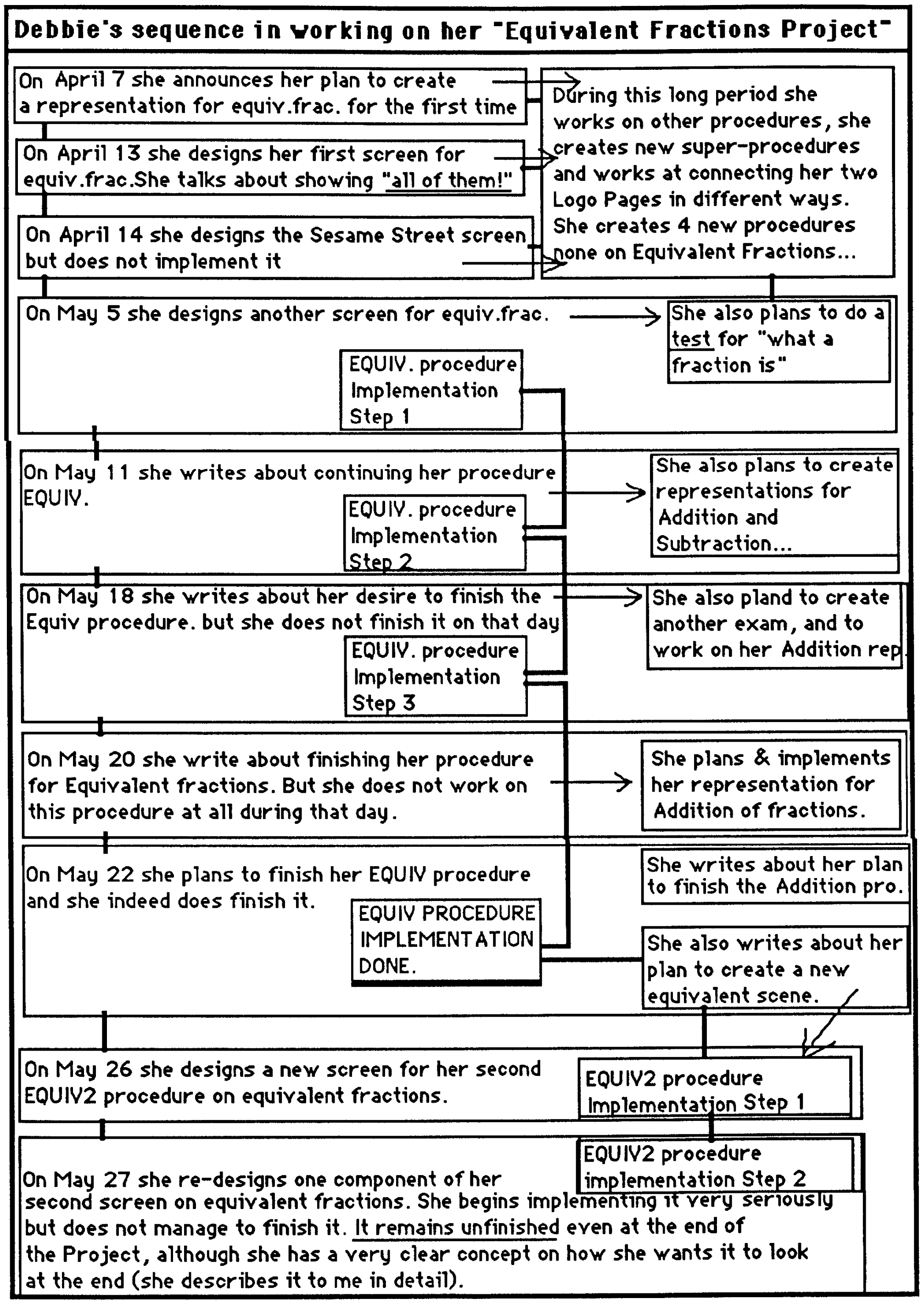
To summarize, I see these two situations of "long distance" and "short distance" imitations as resulting in two different qualities of thinking: one was "mindful" and insightful, the other, "mindless" and superficial (to borrow from Salomon's terminology, 1979, 1986). These became two kinds of cultural learnings: Naomi was strongly influenced by the "Sesame Street culture," but I believe that since she had discovered it on her own and borrowed it from a long distance, she eventually pursued a higher-level and richer route in her cultural learning than Debbie did. Whereas Debbie was only influenced "second hand" by the Sesame Street culture: it was, in fact, Naomi's innovative approach that influenced Debbie's designs on April 14. Perhaps because of its being "second hand" (i.e., not Debbie's own idea) and a "short distance" imitation, Debbie was not as mindful as Naomi when she imitated it, nor motivated to invest time and thought in re-constructing this idea and later in implementing it. However, Debbie gained something from imitating Naomi's imitation, even though she did not expand on it; I also believe that she gained something from her discussion with me. This is why I have interpreted Debbie's imitation activities as "learning," and furthermore, of a "cultural" kind.

## 10.3.4. Two girls--two styles of thinking and expression

Another comparison that should be made here is between the two girls' different styles of thinking and expression. I consider Naomi as someone with very high metacognitive and verbal (or communication) abilities, and Debbie as someone with much lower abilities of this kind (i.e., one needs only to compare the former's holistic and systematic explanations with the latter's short and rigid answers in the two conversations reported above).

All I ever needed to do was to approach Naomi, and with minimal or no promptings at all was provided with very articulate, detailed, and sophisticated explanations about anything I might have wished to know about. Whereas with Debbie, it took many efforts on my part to make her explain and reflect on something. I needed to constantly play games with Debbie, and maneuver around her frequent negative responses such as, "I don't know," "Uhm...Because," or, "You know," etc. Some people might say that Debbie is the type of child who requires a greater experience of adult-child scaffolding, whereas Naomi does not. But through working with Debbie I came to suspect that she did not enjoy scaffolding very much (and one should also differenciate here between scaffolding and paying attention; Debbie very much appreciated the latter). Most of the time I realized that she wanted to be left alone and not be asked to explain too many things; and that she preferred, if at all, to discuss her own ideas and inventions (as opposed to the researcher's ideas and concerns), when she was the one who felt like it (rather than the researcher found it appropriate). Because she was allowed to do so in this context, was able to invest a great deal of time on her own ideas and inventions, and was required to design a piece of instructional software for other children, Debbie came to develop more reflective skills, refined her ability to explain things, opened up, and became relatively more sociable and communicative (and not necessarily and solely because of the researcher's interventions).

The following diagram summarizes Debbie's stage sequences in working on her "Equivalent Fractions Project." The diagram is limited in capturing all of Debbie's activities during this phase; it does, however, attempts to represent the long period of time in which Debbie was involved in this phase of designing, representing, re-designing, and programming Equivalent Fractions of halves, the complexity of her processes, and her parallel thinking on implementing several plans and screens each computer session.



**Debbie's sequence in working on her "Equivalent Fractions Project"**

On April 7 she announces her plan to create a representation for equiv.frac. for the first time

On April 13 she designs her first screen for equiv.frac.She talks about showing "all of them!"

On April 14 she designs the Sesame Street screen but does not implement it

During this long period she works on other procedures, she creates new super-procedures and works at connecting her two Logo Pages in different ways. She creates 4 new procedures none on Equivalent Fractions...

On May 5 she designs another screen for equiv.frac.

EQUIV. procedure Implementation Step 1

She also plans to do a test for "what a fraction is"

On May 11 she writes about continuing her procedure EQUIV.

EQUIV. procedure Implementation Step 2

She also plans to create representations for Addition and Subtraction...

On May 18 she writes about her desire to finish the Equiv procedure. but she does not finish it on that day

EQUIV. procedure Implementation Step 3

She also pland to create another exam, and to work on her Addition rep

On May 20 she write about finishing her procedure for Equivalent fractions. But she does not work on this procedure at all during that day.

She plans & implements her representation for Addition of fractions.

On May 22 she plans to finish her EQUIV procedure and she indeed does finish it.

EQUIV PROCEDURE IMPLEMENTATION DONE.

She writes about her plan to finish the Addition pro.

She also writes about her plan to create a new equivalent scene.

On May 26 she designs a new screen for her second EQUIV2 procedure on equivalent fractions.

EQUIV2 procedure Implementation Step 1

EQUIV2 procedure implementation Step 2

On May 27 she re-designs one component of her second screen on equivalent fractions. She begins implementing it very seriously but does not manage to finish it. It remains unfinished even at the end of the Project, although she has a very clear concept on how she wants it to look at the end (she describes it to me in detail).

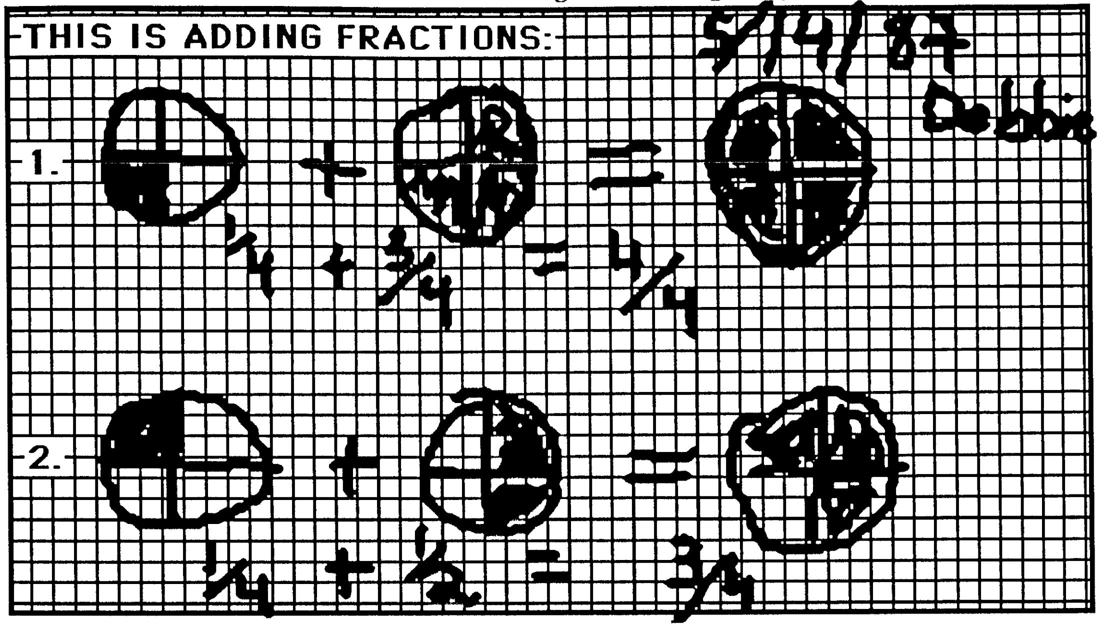
## 11. THE ADDITION ASSIGNMENT

On May 14 the teacher of the Experimental class decided to give the children an assignment about representing addition of fractions. According to her, she was generally "very curious to see what would happen when she gave the children the same assignment as part of their own Project." (Throughout the Project we never gave any assignments or told the children what to represent. We were not sure how the children would react to the idea of an assignment within this open-ended and free Project.) She also initiated it because she "wanted to see how they thought about addition," and also "wanted to create a base-line for talking about representations for fractional operations." In fact, the teacher realized that many of them had not created representations for fractional operations, and felt that if she introduced one fractional operation (addition), they would start thinking more about representing other fractional operations.

It was an interesting experiment. In the Focus Session in the classroom the children generated several ideas on how to represent addition of fractions. What was amazing was that all the children but two clearly stated in their "Plans For Today" (May 14) that the teacher had given them an "Addition assignment." On May 14, Debbie, for example, wrote in her Notebook:

**"TODAY MY TEACHER GAVE US AN ASSIGNMENT.**
**WE DID ADDING FRACTIONS.**
**BYE!!!"**

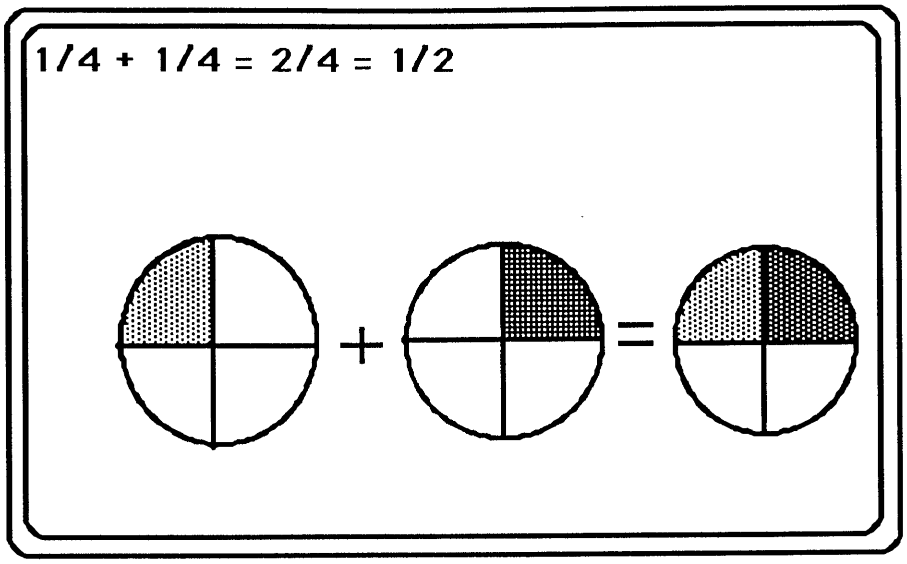
This is an interesting statement on Debbie's part because she always called all her representations "my project,"or "my new scene," etc. But in her writings, she referred to the addition representation as "the addition assignment." Beside this question of ownership, Debbie had no problems understanding the assignment or fullfilling it. On May 14 she designed the two following representations in her Notebook:



The differences between the first and second representations are interesting. The first is the one that the teacher had drawn on the blackboard after her conversation with the children. Most of the children copied it in their Notebooks. The second one was generated by Debbie, after the teacher told the children: "Now, after we did this one together, try to do one representation for addition on your own." In the second representation, Debbie was more sensitive to the instructional purpose of the representation. She created a clear correspondence between the two pictures on the left side of the equation (one-fourth and one-half) and the one on the right (three-fourths). Debbie very clearly stated her reasons for doing it this way:

> **Debbie:** "The first one [number 1] is true and correct but it is not really good for teaching. Because in the second one that I did, they (the users) can really see that the one-fourth goes here [she points to the top-left fourth at the three-fourth representation] and that the two other fourths goes here [she points to the top-right and bottom-right fourths in the three-fourths representation. It's much much much better."

Debbie's mindful instructional approach was implemented by her on the computer in the following screen:



Debbie worked on that screen in parallel with working on her equivalent fractions screen. She never appropriated the addition screen, and never thought of it as "her own thing"; for example, in her Reflections Form on May 18 she writes:

"TODAY I DIDN'T HAVE ANY PROBLEMS. I DIDN'T DO ADDING FRACTIONS ASSIGNMENT. BUT I DID NOT DO MY EQUIVALENT FRACTIONS PROJECT YET, BUT I HAVE TO FINISH IT SOON....I MADE SORT OF LITTLE CHANGES [in her other procedures] BUT THANK YOU AND GOODBYE!!!"

Debbie writes about tWo things she did not do, but describes each of them in a different tone: "I didn't do adding fractions assignment," as opposed to "I did not do **my** equivalent fractions **project,** but I have to finish it soon." However, although she felt uncomfortable about this assignment being imposed on her, she still managed to complete it. The following is Debbie's first version of Logo code for her representation of an addition of two-fourths with my interpretations of it in brackets:

| | |
|---|---|
| **TO ADD** | **[THE SUPER-PROCEDURE]** |
| ADDINGFR | [calls the left side of the equation] |
| HALF | [calls the right side of the equation] |
| PR [1\4 + 1\4 = 2\4 = 1\2] | [prints the instructional statement, ranslating the pictorial representation into symbolic operation.] |
| **END** | |

```
TO ADDINGFR            [FIRST VERSION]
SETC 1+RANDOM 14                     [the computer randomly selects a color for the pen]
PU SETPOS [-100 0]                   [positions the Turtle at the left side of the screen]
PD REPEAT 36 [FD 5 RT 10]            [draws a medium-sized circle]
RT 90 PD FD 57                       [cuts the circle horizontally into halves]
PU BK 28.5 LT 90 PD FD 29 BK 55      [cuts the circle again vertically, so that it shows fourths]
PU HOME
PU SETPOS [-80 10]                   [positions the Turtle at the circle's top-left fourth]
SETC 4 PD FILL                       [shades the top-left fourth in red]
PU HOME PU SETPOS [-25 0]            [positions the Turtle on the right side of the circle]
LABEL "+                             [prints the sign + next to the circle, at its right]
PU HOME                              [finishes drawing one fourth]

SETC 1+RANDOM 13                     [the computer selects one color randomly]
PU SETPOS [-10 0]                    [positions the Turtle on the middle of the screen]
PD REPEAT 36 [FD 5 RT 10]            [draws the same size circle as before]
RT 90 PD FD 57 BK 28.5 LT 90         [cuts the circle horizontally]
FD 29 BK 56 PU HOME                  [cuts the circle vertically to four-fourths]
PU SETPOS [35 10]                    [positions the Turtle on the top-right fourth of the circle]
SETC 12 PD FILL                      [shades in purple the top-right fourth]
END                                  [finishes the pictorial representation of 1/4+1/4]

TO HALF [FIRST VERSION]
PU HOME PU SETPOS [60 0]             [positions the Turtle on the right of the second circle]
LABEL "=                             [prints the sign = for the equation]
PU HOME
PU SETPOS [75 0]                     [positions the Turtle on the right of the screen]
SETC 1+RANDOM 14                     [the computer randomly selects a color for the circle]
PD REPEAT 36 [FD 5 RT 10]            [draws the same size circle as the other two]
RT 90 FD 57 BK 28.5                  [cuts the circle horizontally]
LT 90 FD 30 BK 56 PU HOME            [cuts it vertically]
SETC 4 PU SETPOS [85 10]             [positions the Turtle on the top-left fourth of the third
                                     circle]
```

```
PD FILL                        [shades the top-left fourth in red]
PU HOME
SETC 12 PU SETPOS [110 10]     [positions the Turtle on the top-right fourth of the circle]
PD FILL                        [shades the top-right fourth in purple]
END                            [finishes the representation of a half]
```

**Debbie's modification of her ADDITION PROGRAM during June:**

```
TO ADD                         [THE  SUPER-PROCEDURE]
ADDINGFR                       [calls the left side of the equation]
HALF                           [calls the right dside of theequation]
PR [1\/4 + 1\/4 = 2\/4 = 1\/2]  [prints the instructional statement which is a translation of
                               the pictorial representation into symbolic operation.]
END

TO ADDINGFR             [SECOND VERSION with a sub-procedure for the circle]
SETC 1+RANDOM 14               [the computer randomly selects a color for the pen]
PU SETPOS [-100 0]             [positions the Turtle at the left side of the screen]
CIRCLE.ADD                     [Calls the sub-procedure for drawing the circle]
PU SETPOS [-80 10]             [positions the Turtle on the top-left fourth of the circle]
SETC 4 PD FILL                 [shades the top-left fourth in red]
PU HOME PU SETPOS [-25 0]      [positions the Turtle on the right side of the circle]
LABEL "+                       [prints the sign + next to the circle, at its right]
PU HOME                        [finishes drawing one-fourth]

SETC 1+RANDOM 13               [the computer selects one color randomly]
PU SETPOS [-10 0]              [positions the Turtle in the middle of the screen]
CIRCLE.ADD                     calls the sub-procedure for drawing the circle]
PU SETPOS [35 10]              [positions the Turtle on the top-right fourth of the circle]
SETC 12 PD FILL                [shades the top-right fourth in purple]
END                            [finishes the pictorial representation of 1/4+1/4]

TO HALF           [SECOND VERSION]
PU HOME PU SETPOS [60 0]       [positions the Turtle on the right of the second circle]
```

```
LABEL "=                            [prints the sign = for the equation]
PU HOME
PU SETPOS [75 0]                    [positions the Turtle on the right of the screen]
SETC 1+RANDOM 14                    [the computer randomly selects a color for the circle]
CIRCLE.ADD                          [calls the sub-procedure for drawing the circle]
SETC 4 PU SETPOS [85 10]            [positions the Turtle at the top-left fourth of the third circle]
PD FILL                             [shades the top-left fourth in red]
PU HOME
SETC 12 PU SETPOS [110 10]          [positions the Turtle on the top-right fourth of the circle]
PD FILL                             [shades the top-right fourth in purple]
END                                 [finishes the representation of a half]


TO CIRCLE.ADD                       [THE NEW SUB-PROCEDURE]
PD REPEAT 36 [FD 5 RT 10]           [draws a medium-sized circle]
RT 90 PD FD 57                      [cuts the circle horizontally into halves]
PU BK 28.5 LT 90 PD FD 29 BK 55     [cuts the circle again vertically, so that it shows fourths]
END
```

To summarize my observations during the Addition Assignment, this experiment revealed several things. On one hand, athough Debbie felt that it was "an assignment" and not one of her own projects, she still invested her thought into its instructional designing, and worked at improving the screen so "it would teach better about addition." She worked at combining a pictorial representation of the addition operation with the symbolic representation of the operation. In her instructional statement we saw that she even emphasized the addition algorithm of 1/4+1/4=2/4 and only then reduced it to **1/2**. She was concerned that the algorithm should be emphasized in the pictorial representation as well, by consistently showing a red fourth in the top-left of the circle and a purple fourth in the top-left. She learned several important things from creating this screen.

On the other hand, the major difference between this assignment and the other representations that the children worked on during the Project was the fact that during this week of May all the children worked on the _same_ thing; and since the _teacher_ showed one example on the blackboard, the children felt they had to follow her example (as in all their other school's assignments), and all remained in the same representational framework. They were not strongly motivated to change it or build on it. Although many of them really wanted to work on other screens of their own, they felt that they needed to complete this particular representation because it was "an assignment." Unlike the other screens they had worked on during the Project, this one was imposed on them and interrupted their ongoing work. On the whole, I do not think this assignment was as valuable to the children's learning as the creation of other representations they did on their own. The dominant flavor of this Project was its diversity. One could walk around the class during the computer sessions and observe 17 different children working on 17 different screens and representations. But during the week of the "addition assignment," all the children produced almost the same thing and did not seem very excited or eager about it. It was something that they <u>had</u> to complete, and not something they <u>themselves wanted to</u> complete. This is clear from their writings in the Designer's Notebooks and also from the fact that many of them did not include this screen in their final piece of software.

JUNE...JUNE...JUNE...JUNE...JUNE...JUNE...JUNE

## 12. FINISHING UP: AN OVERVIEW OF DEBBIE'S WHOLE PIECE OF INSTRUCTIONAL SOFTWARE

In the following section I shall describe Debbie's whole software and its Logo programming components. Debbie worked on these long pieces of code throughout the Project, but during the month of June her goal was to work on all the pieces simultaneously, fit them together, and add connections and instructions for the user. We shall see in this section that Debbie's software and her use of Logo grew into a very large and complex undertaking, amazingly so when one considers her age and her initial lack of experience, interest, and apparent capability. And although much could be said about her procedures and thinking processes, we will refain, at this stage, from comment and deep interpretation, and allow Debbie's piece of software, by its very length, to speak eloquently for itself.

From May 29 onward all of Debbie's plans in her Designer's Notebook were about finishing her Project. As examples let us look at two entries from her Designer's Notebook which were very typical of Debbie's writing throughout the month of June:

JUNE 2, PLAN:
"TODAY I'M GOING TO TRY AND FINISH MY FRACTIONS PROJECT. BYE!!!"
JUNE 2, REFLECTIONS:
"I HAD NOT ONE PROBLEM EXCEPT FINISHING. I MADE NO CHANGES EXCEPT THE USUAL!"

JUNE 12, PLAN:
"TODAY I AM GOING TO TRY AND FINISH MY FRACTIONS PROJECT. BYE!"
JUNE 12, REFLECTIONS:
"I HAD PLENTY OF PROBLEMS TODAY AND I DIDN'T GET MY GOAL FOR TODAY DONE. THE PROBLEMS I HAD WERE IN FRACTIONS, LOGO PROGRAMMING, AND DESIGN."

Let us now look at Debbie's Logo Pages one by one:

## A. DEBBIE'S SOFTWARE PART 1: THE LOGO PAGE "FRACTION"

```
TO FRACTIONS                 [THIS IS THE MAIN SUPER-PROCEDURE,
                             THE USER TYPES "FRACTIONS" AT THE
                             COMMANDS CENTER AND THE SOFTWARE
                             BEGINS...]
HT CT CG
INTRODUCTION WAIT 50         ["hello my name is Debbie...I teach you half of the screen"etc.]
HALF   WAIT.FOR.USER         [representing half of the screen]
CG CT CC
CALF   WAIT.FOR.USER         [multiple representations for halves using 4 geometrical shapes]
CG CT CC
TWO    WAIT.FOR.USER         [the representation for two-thirds, and quiz]
CG CT CC
GETTOOLS "FRACT1             [get procedures from the third (new) Logo Page "Fract1]
QUEST                        ["Do you know how to do fractions?..." etc., and quiz]
WAIT.FOR.USER CC CG CT
HOUSE   WAIT.FOR.USER        [the House Scene about halves]
CG CT CC
EXAM                         [The "Do you remember what you learned?..." Screen, where
                             Debbie provides a "going over," or a set of new representations]
SUPER                        [connects this page with "Fract2, includes equivalent fract.]
END
```

```
TO INTRODUCTION
pr [Hello, my name is Debbie. I am going to teach you how to do fractions.
The first thing that I am going to teach you is the half of the screen.
The scene is a half or two-fourths.]
END
```

```
TO HALF
SETC 9 FD 250                    [cuts the screen horizontally]
PU HOME RT 90 SETC 1             [goes back home and changes the pen color]
PD FD 320                        [cuts the screen vertically]
PU SETPOS [40 45]                [goes to the center of the top-right fourth]
LT 90 SETC 4 PD FILL             [fills it in red color]
PU SETPOS [40 -45]               [moves to the center of the bottom-right fourth]
PD SETC 3 PD FILL                [fills it in pink color]
PR [This is my fraction project] [prints instructional sentence at the top]
PR [This is 1/2 or 2/4]
PR []                            [skips a space]
PR [By Debbie]                   [prints her name]
END

TO CALF
HT SETC 1+ RANDOM 14                  [sets the pen color randomly]
PD FD 200                             [draws a vertical line across the screen]
PU HOME
RT 90
SETC 1+RANDOM 14                      [sets the pen color randomly]
PD FD 320                             [draws a horizontal line across the screen]
PU HOME
PU SETPOS [30 50]                     [moves Turtle to position of circle]
SETC 12 PD REPEAT 360 [FD.3 RT 1]     [draws a circle on the top right]
RT 90
PD FD 33                              [divides the circle in half]
PU SETPOS [45 40]
PD FILL                               [shades in the bottom half of the circle]
PD HOME
PR [ALL THESE FRACTIONS ARE 1/2.]     [prints the instructional statement]
SETPOS [-60 50]                       [moves the Turtle to position of square]
SETC 1+RANDOM 14
```

```
PD REPEAT 4 [LT 90 FD 40]                  [draws the square on the top left]
PU LT 90 FD 20 LT 90
PD FD 40                                   [divides the square in half]
RT 90 FD 10 RT 90
PU FD 5 PD FILL                            [shades in the left half of the square]
PU HOME
SETC 1+RANDOM 14
SETPOS [35 -70]                            [moves Turtle to position of rectangle]
PD REPEAT 4 [FD 50 RT 90 FD 100 RT 90]     [draws a rectangle on the bottom right]
RT 90 FD 50 LT 90 PD FD 49                 [divides the rectangle in half]
PU RT 90
FD 25 RT 90 FD 10 PD FILL                  [shades in the right side of rectangle]
PU HOME
SETC 1+RANDOM 14
PU SETPOS [-105 -55]                       [moves the Turtle to position of triangle]
RT 45 PD FD 50                             [creates a triangle]
RT 90 FD 52
RT 135 FD 73                               [moves the Turtle to position of cutting
REPEAT 2 [RT 90]                           the triangle into two equal parts]
FD 36 LT 90 PD FD 35                       [divides the triangle]
PU SETPOS [-55 -40] PD FILL                [shades the half on the right side of the triangle]
END

TO  WAIT.FOR.USER
TYPE [PRESS ANY KEY TO CONTINUE]           [types a message on the bottom of the screen]
IGNORE READCHAR                            ["sends" the character that was typed by the user
 END                                        as the input for the IGNORE procedure]

TO IGNORE :KEY                             [the procedure IGNORE takes one input, a character,
END                                        but does nothing with it, so the program simply goes on]
```

```
TO TWO
PU HOME SETC 1+RANDOM 14
PD REPEAT 360 [FD 1 RT 1]          [creates a big circle]
PU SETPOS [60 5]                   [positions the Turtle in the middle of the circle]
SETC 1+RANDOM 14
PD FD 53                           [draws lines for dividing the circle into thirds]
PU BK 53 RT 135 PD FD 57
PU BK 57 RT 90
PD FD 62 PU BK 62 RT 45 PU FD 20
SETC 1+RANDOM 14
PD FILL                            [fills in one-third]
PU BK 40 PD FILL                   [fills in the second third]
PU HOME
WAIT 100 PR [THIS FRACTION IS 2\/3 ! IT
IS ONE OF THE FRACTIONS TEACHERS USE MOST    [prints her instructional statement]
FOR EXAMPLES TO TEACH THEIR STUDENTS.]
WAIT.FOR.USER CG CT CC
QUESTION1
END


TO  QUESTION1
PR [IS THIS FRACTION, 1\/2, 3\/4, 2\/3, OR 1\/26 ?]
NAME READLIST "ANSWER
IFELSE (ANSWER =[2\/3])
[PR[GREAT!!]] [PR[PLEASE TRY AGAIN] CT WAIT 30 QUESTION1]
END
```

```
TO QUEST                                [one of the procedures that she modified in June)
PR [Do you know how to do fractions?]
name readlist "answer
ifelse :answer = [no] pr [Well, that's what I'm here for, to teach you about fractions!]]
[pr [That's fine. Now you will understand this work better!]]
pu home pd repeat 8 [ setc 1+random 14 rt 45 fd 50]
pu bk 25 rt 90 pd fd 120
bk 60 lt 90 pu bk 60 pd fd 121 bk 70 rt 45 pu fd 20
setc 1+ random 14
pd fill pu home rt 90 pu fd 20
setc 1+random 14 pd fill
pr [WHAT PART IS COLORED? IS IT 2\/3, 2\/4, or 4\/4?]
name readlist "answer
ifelse (:answer = [2/4] [pr [great!!!]]  [pr [sorry, that's incorrect.] wait 35 quest]
END
```

```
TO OKAY                                 [This is the "going over" for the procedure "EXAM"]
HOUSE WAIT.FOR.USER CG CT CC
HALF WAIT.FOR.USER CG CT CC
CALF WAIT.FOR.USER CG CT CC
TWO WAIT.FOR.USER CG CT CC
END
```

**TO SUPER** [This procedure is called by the "Fractions" super-procedure. It connects Debbie's two Logo Pages]

```
GETTOOLS "FRACT2
CT CG CC
PR [PLEASE WAIT A MINUTE OR TWO, THE TURTLE IS SLEEPING.
I'LL GO WAKE HIM UP.  (WAKE UP YOU SILLY JERK.
WE HAVE A BIG AUDIENCE OUT THERE!!!)]
```

[this is an interesting note that Debbie prints on the screen as Logo searches for the other Page "Fract2.]

```
WAIT.FOR.USER
INTRODUCTION
WAIT.FOR.USER
EQUIV
WAIT.FOR.USER
MORE
WAIT.FOR.USER
UNDERSTANDING
END
```

## B. DEBBIE'S SOFTWARE PART 2: THE LOGO PAGE "FRACT1"

```
TO HOUSE
                                        [THE HOUSE, THE DOOR, AND THE ROOF:]
pu home setc 4  pu setpos [-65 25] rt 45 pd fd 100 rt 90 fd 100 rt 135 fd 140  lt 90 setc 1 fd 100 setc 13 lt
90 fd 140 lt 90 setc 10 fd 100  pu setpos [10 0] setc 11 pu bk 75 pd fd 100 pu home pd fill pu setpos [10
25] setc 12 pd fd 68 pu setpos [20 35] pd fill pu setpos [25 -75] setc 5 pd fd 60 setc 9 rt 90 fd 30 rt 90  setc
4  fd 60 pu setpos [25 -40] pu bk 3 rt 90 pd fd 30 pu setpos [36 - 33] pd fill
                                        [THE LEFT WOODEN WAGON:]
pu home pu setpos [-150 -40] pd repeat 2 [fd 30 rt 90 fd 60 rt 90] pu home
pu setpose [-155 -50] setc 1+ random 14  pd repeat 360 [fd .2 rt 1] pu home
pu setpos [-120 -40] setc 4 pd fd 30 pu rt 90 fd 20 rt 90 fd 10 pd fill
pu home pu setpos [-145 -50] pd fill pu home setc 1 +random 14
pu setpos [-90 -50] pd repeat 360 [fd .2 lt 1]  setc 4 lt 90 pu fd 10 pd fill
                                        [THE RIGHT WOODEN WAGON:]
 pu home pu setpos [85  -45] pd repeat 2 [fd 30 rt 90 fd 60 rt 90]
rt 90 fd 30 lt 90 fd 30 pu lt 135 fd 15 pd fill pu home
setc 1+random 14  pu setpos [80 -55] pd repeat 360 [fd.2 rt 1] setc 4 pu rt 90 fd 10
pd fill pu home pu setpos [155 -55] pd repeat 360 [fd .2 rt 1] setc 4 lt 90 pu fd 10 pd fill
                                        [THE SUN:]
pu home  pu setpos [95 65] setc 14
pd repeat 360 [fd .3 rt 1] rt 90 pd fd 35 pu home pu setpos [115 75] pd fill
                                        [THE INSTRUCTIONAL SENTENCE:]
pr [This is a house. Almost every shape is 1/2!  I am trying to say, that you  use fractions, almost every
day of your life!]
END

TO EXAM
PR [DO YOU REMEMBER WHAT YOU LEARNED?]
NAME READLIST "ANSWER
IFELSE (:ANSWER = [YES])  [PR [GREAT!!! I WILL TEACH YOU MORE THEN ON FRACT2]
WAIT 50]  [PR [WELL. LET'S REVIEW IT!] WAIT 100 CT CG CC OKAY]
END
```

## C. DEBBIE'S SOFTWARE PART 3: THE LOGO PAGE "FRACT2"

```
TO BIG
QUEST WAIT.FOR.USER   CG CT CC
HALF WAIT.FOR.USER  CG CT CC
HOUSE WAIT.FOR.USER  CG CT CC
TWO
END
```

**TO FRONT** [Remained empty at the end of the Project]
**END**

**TO W** [Was never completed because Debbie did not like it. See Section 7 in the Case]

```
pu setpos [-85 55]
rt 135
setc 1+random 14  pd fd 20
lt 100
setc 1+random 14  pd fd 19
rt 100
setc 1+random 14  pd fd 20
rt 100
setc 1+random 14  pd fd 20
rt 100
setc 1+random 14 pd fd 19
END
```

**TO INTRODUCTION** [Appears on the other Page as well]]

```
pr [Hello, my name is Debbie. I am going to teach you how to do fractions.
The first thing I'm going to teach you is the half of the screen. The scene is a half or two-fourths.]
END
```

```
TO QUEST                                [Appears on the other Logo Page as well]
pr [Do you know how to do fractions?]
name readlist "answer
ifelse (:answer =[no]) [pr [Well, that's what I am here for, to teach you fractions!]]
[pr [That's fine. Now you will understand this work better!]]
pu home pu repeat 8 [setc 1+random 14 rt 45 fd 50]
pu bk 25 rt 90 pd fd 120 bk 60 lt 90 pu bk 60  pd fd 121
bk 70 rt 45 pu fd 20 setc 1+random 14 pd fill
pu home rt 90 pu fd 20 setc 1+random 14 pd fill
ct pr [what part is colored? Is it 2\/3, 2\/4, or 4\/4?]
name readlist "answer   ifelse (:answer = [2\/4]) [pr [GREAT!]] [pr [Sorry, that's incorrect] ct cg QUEST]
END


TO TOPO                                 [Another super-procedure, was planned during April, and
                                        finally implemented during June]
INTRODUCTION                            [In TOPO, Debbie included her two opening screens. She
QUEST                                   realized that each serves a different instructional purpose]
WAIT.FOR.USER CG CT CC
WARNING@LETTER                          [A very interesting "letter," look below...]
EQUIV
WAIT.FOR.USER CG CT CC
EQUIV2
WAIT.FOR.USER CG CT CC
TOGETHER                                [This procedure was planned in April and implemented
                                        during June, look below]
END
```

```
TO INTRODUCTION                 [Another Introduction, or an opening screen for her second
                                part of the software]
PR [HELLO AGAIN! HAVEN'T YOU LEARNED ENOUGH?]
NAME READLIST "ANSWER
PR [NO MATTER WHAT, I GUES YOU WANT TO LEARN MORE!]
WAIT 50 CT
END

TO WARNING@LETTER
PR [DEAR STUDENT,
        THIS IS VERY SERIOUS. I AM GRADING YOU ON IT ALSO!
        SINCE YOU FINISHED LEARNING 'WHAT A FRACTION IS' SO I THINK
         YOU'LL UNDERSTAND EQUIVALENT FRACTIONS.
         I HAVE GREAT GRAPHICS TO SHOW YOU WHAT IT IS.
                                SINCERELY,
                                YOUR TEACHER DEBBIE D.
WARNING: DANGER!]
END

TO TOGETHER  [CONNECTS THIS PAGE WITH THE FIRST PAGE]
GETTOOLS "FRACTION
WARNING@LETTER
PR [PLEASE WAIT A WHILE. GET IT? LET'S WAIT A WHILE.
I'M SERIOUS I'LL NEED A MINUTE OR TWO.]
HT CT CG
HALF WAIT.FOR.USER CG CT CC
CALF WAIT.FOR.USER CG CT CC
TWO  WAIT.FOR.USER CG CT CC
HOUSE CG CC WAIT 1-- CT
EXAM
INTRODUCTION
END
```

```
TO  WAIT.FOR.USER                 [EXIST ON OTHER PAGES AS WELL]
type [PRESS ANY KEY TO CONTINUE]
IGNORE readchar
END

TO  IGNORE :key                   [was created on March 31]
END

TO EQUIV        [THE REPRESENTATION FOR EQUIVALENT FRACTIONS]
PU HOME SETC 13
PD FD 190 PU HOME
RT 90 PD FD 320
PU SETPOS [20 10] SETC 1 PD
REPEAT 2 [FD 60 RT 90 FD 120 RT 90]
LT 90 BK 20 RT 90 FD 60
RT 90 FD 20 RT 90 FD 60  LT 90 FD 20
LT 90 FD 60 RT 90 FD 20 RT 90 FD 60
LT 90 FD 20 LT 90 FD 60 RT 90 FD 20
RT 90 PU FD 30 RT 90 PD FD 120    [cuts the rectangle horizontaly, in the middle, which creates
                                  twelve-twelfths]
PU HOME
PU SETPOS [30 30] SETC 9 PD FILL
REPEAT 2 [PU FD 20 PD FILL
RT 90 PU FD 20 PD FILL]
PU RT 90 FD 20 PD FILL            [shades six-twelfths in blue]
PU HOME
PU SETPOS [15 80] LABEL [6\/12 = 1\/2]  [prints over the rectangle 6/12=1/2 in yellow]
PU HOME                           [finishes the representation for six-twelfths equals
                                  one-half]
PU SETPOS [-60 50] SETC 3
PD REPEAT 36 [ FD 7 LT 10]
LT 90 FD 80
```

```
PU HOME
SETC 11
PU SETPOS [-75 65]
PU BK 10 PD FILL
PU BK 20 PD FILL
PU HOME
PU SETPOS [-65 80]
LABEL [1\/2=2\/4]                       [prints the equivalence one-half equals
PU HOME                                 two-fourths]

SETC 4 PU SETPOS [25 -80]

PD REPEAT 4 [FD 50 RT 90]
FD 25 RT 90
FD 50 BK 25 LT 90 FD 25 BK 50 FD 25
RT 45 FD 32 BK 64 FD 32 LT 90 FD 32
BK 64
PU HOME
SETC 12 PU SETPOS [60 -35] PD FILL
PU BK 5 RT 90 PU FD 10 PD FILL
RT 90 PU FD 30 PD FILL                  [shades four-eighths in orange]
PU HOME
PU SETPOS [45 -15] LABEL [4\/8=1\/2]    [prints the equivalence, four-eighths equals
                                        one-half]
PU HOME
SETC 2 PU SETPOS [-140 -85]
PD REPEAT 2 [FD 60 RT 90 FD 120 RT 90]
PD REPEAT 10 [SETH 90 FD 12 LT 90 PD FD 60 BK 60]
FD 30 LT 90 FD 120
PU HOME
SETC 10 PU SETPOS [-135 -40]
PD FILL PU RT 90 FD 10 PD FILL
PU FD 15 PD FILL PU FD 10 PD FILL
PU FD 10 PD FILL PU FD 15 PD FILL
PU FD 10 PD FILL PU FD 10 PD FILL
PU FD 12 PD FILL PU FD 15 PD FILL       [finishes shading ten-twentieths in light green]
PU HOME
SETPOS [-105 -15] LABEL [10\/20 =1\/2]   [prints the equivalence, ten-twentieths equals
                                         one-half]
END

TO EQUIV2   [The second representation for Equivalent Fractions. Was never completed]
SETC 11 PD FD 200 PU HOME               [divides the screen vertically in light blue]
RT 90 SETC 10 PD FD 320 PU HOME         [divides the screen horizontally in light green]
SETC 4 PU SETPOS [15 55]
PD REPEAT 4 [RT 90 FD 50]               [draws a red square]
RT 90 FD 25 RT 90 FD 50 BK 25           [divides the square into two halves]
PU RT 90 FD 10 SETC 12 PD FILL          [shades the left half in orange]
PU HOME
SETC 12 PD REPEAT 4 [RT 90 FD 50]       [draws an orange square]
PU RT 90 FD 25 RT 90 PD FD 50           [divides it into fourths]
PU BK 25 LT 90 PD FD 25 BK 50
PU SETC 4 LT 90 FD 14.1/2 RT 90
PU FD 5 PD FILL                         [shades the top-left fourth of the square]
PU RT 90 FD 15 PD FILL                  [shades the bottom-left fourth of the square]
PU HOME

PR [ALL THESE SHAPES ARE EQUIVALENT
FRACTIONS.  THIS IS WHAT I AM GOING TO
TEACH YOU!]                             [prints the instructional statement]
PU HOME
SETC 5 PU SETPOS [-155 35]
PD REPEAT 180 [ FD 1 RT 2]              [draws a purple circle]
PU RT 90 PD FD 56                       [divides the circle into halves]
BK 28 LT 90 FD 28 BK 56 FD 28 RT 45     [divides the circle into fourths]
FD 28 BK 56 FD 28 LT 90 FD 28 BK 56     [divides the circle into eighths]
```

```
PU HOME SETC 13                  [changes the pen color to pink]
PU SETPOS [-120 50]              [position the Turtle in one of the circle's eighths]
PD FILL RT 90 PU FD 10 PD FILL   [shades in pink two of the eighths]
PU BK 20 PD FILL                 [shades in pink another eighth]
END

TO ADD                           [THE SUPER-PROCEDURE FOR ADDITION]
ADDINGFR                         [calls the left side of the equation]
HALF                             [calls the right side of the equation]
PR [1\/4 + 1\/4 = 2\/4 = 1\/2]   [prints the instructional statement which is a translation of
                                 the pictorial representation into symbolic operation.]
END

TO ADDINGFR         [THE SECOND VERSION, with a sub-procedure for the circle]
SETC 1+RANDOM 14
PU SETPOS [-100 0]
CIRCLE.ADD
PU SETPOS [-80 10]
SETC 4 PD FILL
PU HOME PU SETPOS [-25 0]
LABEL "+
PU HOME
SETC 1+RANDOM 13
PU SETPOS [-10 0]
CIRCLE.ADD
PU SETPOS [35 10]
SETC 12 PD FILL
END

TO HALF
PU HOME PU SETPOS [60 0]
LABEL "=
PU HOME
PU SETPOS [75 0]
SETC 1+RANDOM 14
CIRCLE.ADD
SETC 4 PU SETPOS [85 10]
PD FILL PU HOME
SETC 12 PU SETPOS [110 10]
PD FILL
END

TO CIRCLE.ADD
PD REPEAT 36 [FD 5 RT 10]
RT 90 PD FD 57
PU BK 28.5 LT 90 PD FD 29 BK 55
END

TO SQUARE :NUM
REPEAT 4 [FD :NUM RT 90]
END

TO CIRCLE :NUM
REPEAT 36 [FD :NUM RT 10]
END

TO CIRCLE :NUM1
REPEAT 360 [FD :NUM1 RT 1]
END

TO RECT :NUM1 :NUM2
REPEAT 2 [FD :NUM1 RT 90 FD :NUM2 RT 90]
END

TO TALK [She created this procedure during the last week of May]
HT PR [WHAT'S YOUR NAME?]
```

```
MAKE "NAMES READLIST
PR SE [HELLO] :NAMES

PR [HOW OLD ARE YOU?]
MAKE "AGES READLIST
PR SE [THAT'S GOOD FOR YOU!] :NAMES

PR [WHAT'S THE NAME OF YOUR SCHOOL?]
MAKE "SCHOOL READLIST
PR SE [THAT SCHOOL SOUNDS FAMILIAR!] :SCHOOL

PR [WHAT GRADE ARE YOU IN?]
MAKE "GRADE READLIST
IFELSE :GRADE = [4] [PR[THAT'S REALLY GREAT!]]
[PR [I HAVE NOTHING TO SAY ABOUT THAT...]]

PR [HOW MANY SISTERS DO YOU HAVE?]
MAKE "SISTERS READLIST
IF :SISTERS = [0] PR SE [THAT'S GOOD] :NAMES STOP]
PR SE [TOO BAD...] :NAMES
END
```

In her three Logo Pages, "Fraction," "Fract1," and "Fract2," Debbie created so many combinations of procedures and super-procedures, several introductions, quizzes, and instructions for the user; it is almost impossible, on my part, to construct one comprehensive flowchart that would represent the structure of the software as a whole. A further analysis of Debbie's final code and its internal structure is required and will be conducted in my future papers. In fact, Debbie wrote several other procedures in her Project which were not report about in this context, because they require much longer and deeper analyses.

To my mind this--the software's complexity and length, and the fact that many procedures remained unfinished--is the beauty of her Project. Debbie enjoyed the idea of modularity to the utmost; and so, to use her own words, "created many pieces of software [for the same purpose]--all in one." Every day, during the month of June, she figured out a new combination for her "building blocks."

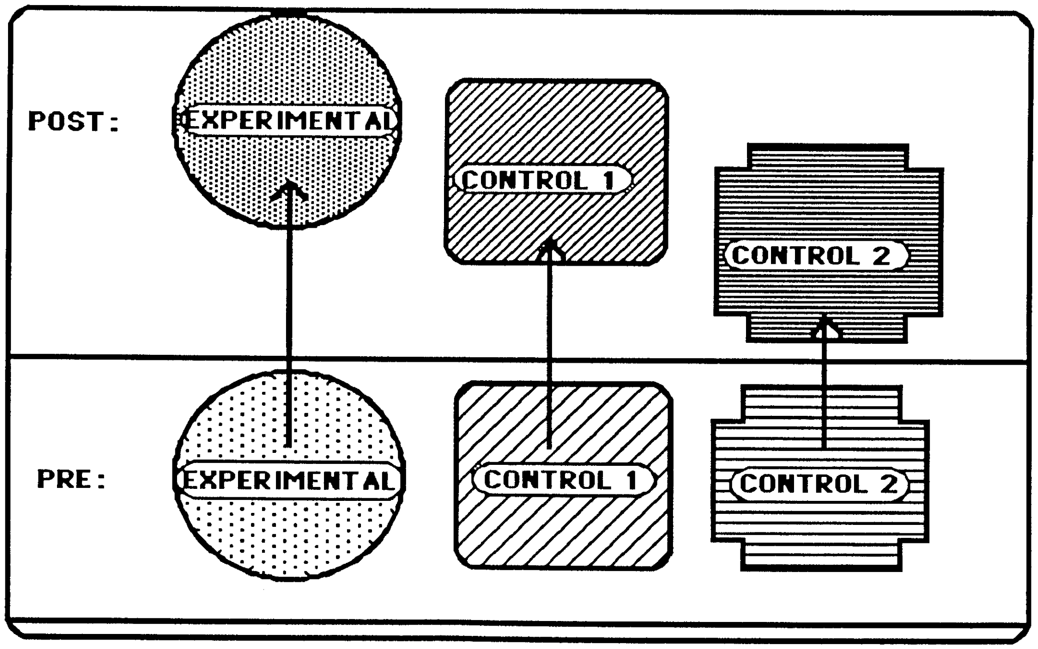
Another beautiful aspect of Debbie's Project lies in the fact that she never really finished it. Debbie was willing to go on and on and continue working on new screens, and on the ones she never completed. But the school year ended on June 23, 1987, and since Debbie did not have a computer at home, she never had a chance to continue working on her Project during the summer. Instead (and unfortunately for Debbie) I took her many procedures and all of her writings and drawings to continue what she had started: to try and make sense of this Project, and to understand what fractions are, what Logo is, and what software design means for fourth-grade children....

**THE RESULTS FROM DEBBIE'S PRE- AND POST-TESTS WILL BE PRESENTED IN THE LAST SECTION, NUMBER 2.3, OF CHAPTER III.**

# SOFTWARE DESIGN FOR LEARNING

# CHAPTER III:

# RESULTS FROM THE EVALUATION

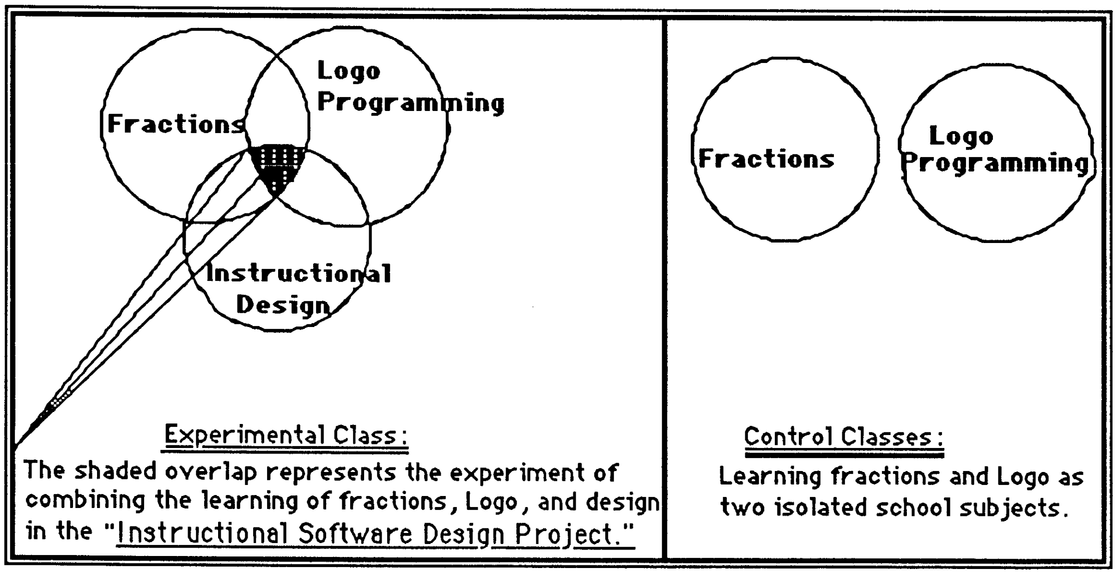


## 1. INTRODUCTION

This chapter presents selected results from the Evaluation of the Treatment. These are the results of the Fractions and Logo Tests that were conducted with the Experimental and two Control classes (N=51) before and after the Instructional Software Design Project was implemented.

### 1.1. The Design of the Evaluation

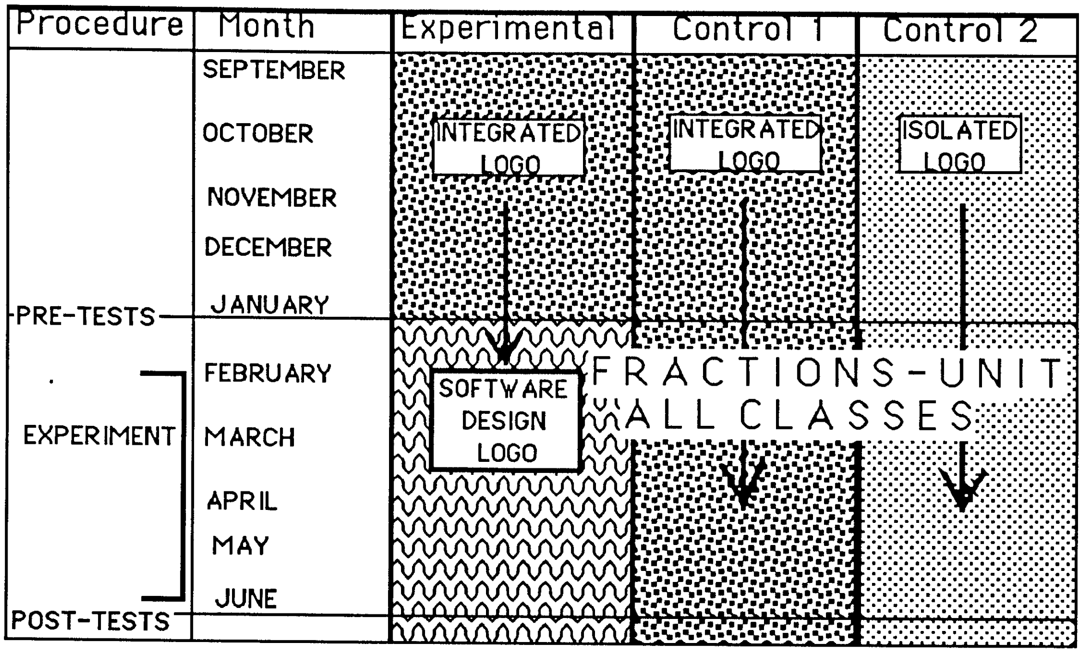
The Evaluation was designed in order to examine in what ways children who learned fractions and Logo by designing a piece of instructional software differed from children who learned fractions and Logo through other pedagogical methods, rather than through software design. The following diagram shows the differences between the experimental and the other pedagogical approaches that were assessed for this Evaluation:



Experimental Class:
The shaded overlap represents the experiment of combining the learning of fractions, Logo, and design in the "Instructional Software Design Project."

Control Classes:
Learning fractions and Logo as two isolated school subjects.

Three fourth-grade classes from the same inner-city public school in Boston were selected for this Evaluation (i.e., N=51: in the Experimental Group N=17, and in the two Control Groups N=18, and N=16). The reasons for their selection, and a detailed description of the characteristics of the pupils, their teachers, and their different fractions and Logo learning approaches were already presented in Chapter I, Section 3. In brief, during January 1987, all three classes were pre-tested on specific skills and concepts in fractions and Logo. After the Pre-Tests were completed, a four-month Experiment in software design was conducted with one of these classes (represented in the shaded overlap on the left side of the above diagram and described in Chapter I). All 51 pupils were tested again in June (when the Software Design Project ended) on their knowledge and specific skills in fractions and Logo. This Evaluation procedure is once again represented in the following diagram, with a brief recapitulation of the differences between the three classes.



"Fractions Unit" means that all teaching of fractions, for all three classes, was conducted for 2 months, through stricly regular math lessons and by following traditional method. **The Experimental class was not provided with any additional instruction on fractions (besides the material they covered in their regular math lessons) during the Project.**

"Isolated Logo" (in Control class 2) means that **a)** the children programmed once a week in the school's computer laboratory; **b)** they followed a conventional approach to Logo programming, using worksheets, exercises, etc., as part of a computer-literacy program; **c)** the programming activities resulted in children's completion of short programming exercises, but not in creative products.

"Integrated Logo" (in Control class 1) means that **a)** the children programmed every day; **b)** they integrated the Logo projects into various topics of the curriculum; **c)** the programming activities resulted in children's completion of various small-scale creative programming projects, each lasting approximately 1-3 weeks.

"Software-Design Logo" (in the Experimental class) means that **a)** the children programmed every day; **b)** they integrated Logo into the curriculum, but in the context of designing and programming instructional software for teaching about fractions; **c)** each child worked on the project for 4 months, and the programming activities resulted in a long and complex product (or in interactive lessons about fractions for third graders).

## 1.2. The Evaluation Objectives and Questions

The Evaluation focused on the children's knowledge of **fractions and Logo.** Using the set of Pre-Tests, the differences between the Experimental and the Control children's knowledge of fractions and Logo before the Experiment began were investigated; a base line was established, and no significant differences were found. Four months later, using the set of Post-Tests, the ways in which these children differred in their knowledge and understanding of fractions and Logo after the Experiment was ended were investigated in detail.

One part of the Post-Evaluation aimed at assessing how much better the children from the Experimental class were at **understanding fractions**. Within this domain, emphasis was placed on one particular aspect of the children's knowledge of fractions: their 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 et al, 1983; Behr et al., 1983).

By giving specific attention to the children's knowledge of fractional representations, the Evaluation aimed at examining how much better the children from the Experimental class were able to translate representations of fractions after the Experiment--to see, for example, whether or not they got better at translating symbols into words, words into pictures, pictures into symbols, etc; whether they showed improved flexibility in recognizing and understanding fractional representations, and an improved ability in identifying and giving examples for representations for fractions in non-school situations (for example, finding representations for fractions in rooms and stores, in money, clocks, calendars, or other manipulative aids, such as play dough, pegs, Pattern-Blocks, etc).

These Fractions Tests were also designed to assess whether the children from the Experimental class were better at constructing their own representations for fractions after being involved in designing the instructional software; and what were the differences between the ways the children from the Experimental and the Control classes "talked about" fractions (i.e., assessing whether the Experimental subjects were more flexible and articulate in their explaining of fractions and in their solutions or comments on fractions problems posed to them by the researcher).

The other objectives of the Evaluation gave specific attention to the children's knowledge of **Logo programming** after the Experiment. For this purpose, the Evaluation was designed to investigate the ways in which the children from the Experimental class differred from those of the Control classes in their knowledge, use, and understanding of Logo programming commands, instructions, and operations. More specifically, it assessed whether the children from the Experimental class knew and understood more programming commands and operations than the Control children once the Experiment was over.

The Evaluation also investigated whether the Experimental children could understand, implement, debug, transform, optimize, and modify someone else's programming code better than the children from the Control classes; and whether they understood and used some of the Logo commands and operations, such as REPEAT, IFELSE, SETPOS, variables, and inputs in their projects, and became better at these skills than the children in the two Control classes. Finally, the Evaluation assessed whether the Experimental children were able to construct Logo routines for someone else's design or picture and were better at this than the children in the two Control classes.

### 1.3. The Evaluation Procedure

A series of tests, computer tasks, and interviews on fractions and Logo programming was administered to the Experimental and the Control classes before and after the experiment. A short description of each of the testing items is given below, and samples of these tests and interviews can be found in Appendix B. The "task analysis" of each item in the tests is presented beside the results themselves, so that the reader can relate the tasks, their requirements, and their underlying conceptual scope, to the actual results. There were two versions for most items, one for the Pre-Tests and one for the Post-Tests. However, the Pre- and Post-Tests were generally identical in terms of their underlying conceptual scope. The Post-Tests were usually longer and more demanding with several difficult questions added on. But in most cases, the Pre-Questions also appeared in the Post-Tests.

#### 1.3.1. The Instruments Used for the Evaluation

The following is a list of all the tests and interviews that were used for the Evaluation of the Instructional Software Design Project.

A. THE RATIONAL-NUMBER PENCIL-&-PAPER TEST:

A set of 43 multiple-choice questions in the Pre-Test, and 65 questions in the Post-Test, including written language, written symbols, and several types of pictorial representations -- was used to assess the childrens' understanding of rational-number concepts before and after the experiment. This paper-and-pencil test was taken from a collection of tests developed for the Rational-Number (RN) Project by Behr, Lesh, & Post (Lesh & Hamilton, 1981; Lesh, Landau, & Hamilton, 1983). The researchers of the RN Project discovered and documented many of the problems young children encounter when translating these representations of rational numbers (as described in Chapter I, Section 2.3.2). The researchers of the RN Project had developed these tests for investigating the nature of children's ideas about rational numbers in grades 2 through 8. The particular questions selected for this Evaluation were geared towards exploring children's ability to translate within and between representational modes (written language, written symbols, pictures) on sets of structurally related tasks. All tasks in the selected test assess children's basic concepts of fractions and ratios; they also assess children's ability to translate representations, ordering, equivalent fractional forms, and simple proportions. The test was administered to each class separately but on the same day. Children were given 60 minutes to complete their tests.

B. WHAT IS A FRACTION? INTERVIEW:

A 30 to 45 minute interview was conducted with each of the 51 pupils before and after the Experiment. This interview was designed by the researcher to investigate the children's concept of what a fraction is, their "favorite" representations of fractions, their way of "talking about" fractions, and their ability to link or translate representations given by themselves or by the researcher, using various materials and manipulative aids.

During the Interview, children answered general questions, such as "What is a fraction?" or "Can you learn about fractions outside of school?" and were also asked to solve more specific problems such as "Please use this set of pegs (play dough, paper, blocks, and other materials in the room) to show an example of the fraction 2/3." While the children constructed the required representations, the researcher probed the solidity of their knowledge by posing problems ("One child told me that this red block and this yellow block together equal one-half, what do you think?"), and investigated their concept of fractions more deeply.

The interview was open-ended. However, while the children were following their personal routes, the researcher, using a Piagetian technique, intervened with a predetermined set of questions. Several more structured tasks were added in the Post-Interview. Most of the added tasks required that the children manipulate several objects such as Pattern-Blocks and answer questions such as: "If this yellow block is the unit, what are these two yellow blocks and three red blocks together (one red block being half the size of the yellow block)?" After the child had "constructed" an answer, the researcher could ask, "But one child told me that all of them together equal five, because there are five pieces here. What do you think?" These interviews were videotaped and transcribed, and a set of categories was developed in order to analyze and describe the results.

C. BOSTON PUBLIC SCHOOLS MATH TEST, LEVEL-4:

The Math Level-4 Curriculum Referenced Test, which is given to all fourth-graders in the City of Boston at the end of each year, was used in the present study as a Post-Test only. The test included 40 multiple-choice questions, of which 15 were about fractions. Children had up to 60 minutes to complete this test.

D. LOGO PENCIL-AND-PAPER TEST:

This test included two open-ended tasks and three well-defined tasks. It was designed by the researcher to assess: a) the children's understanding and use of Logo instructions and commands; b) their ability to follow and execute a given code; and c) their ability to "chunk" a given program using sub-procedures, Repeats, or variables. The Post-Test includes all of the Pre-Test with additional, more difficult sub-tasks. These tasks require that children: a) generate a Logo code for a given picture; b) follow and execute (draw) a given code; and 6. manipulate the inputs for circles, half-circles, rectangles, and squares. Each of these Pencil-&-Paper Logo Pre- and Post-Tests lasted approximately 45-60 minutes.

E. LOGO COMPUTER TEST:

This test was designed by the researcher to assess the children's knowledge and ability to use Logo while working at the computer. This task required that they debug a program, given to them on-line, by analyzing the picture created when the bugged program was executed. In addition, the children had to modify the debugged program according to the researcher's requests ("make it modular..." or "try to use variables now..." etc.). The

Post-Test was the same as the Pre-Test, using a slightly more complex picture. However, the underlying structure and the requirements of the Pre- and Post- Logo Computer Tests were identical. Each of the tests lasted approximately 45-60 minutes.

The following diagram is a summary of the Logo and Fractions tests and interviews that were conducted with the 51 pupils from the Experimental and the Control classes, before and after the Instructional Software Design Project.

**THE PRE- and POST-TESTS and INTERVIEWS**

| | **Name and Purpose of Test** | **Length** | **No.of items** |
|---|---|---|---|
| **Pre-Tests about FRACTIONS** | Rational Number Pencil & Paper | 60 minutes | 43 questions |
| | What is a Fraction? Interview | 30-45 min. | 10 questions |
| **Pre-Tests about LOGO** | Logo Pencil & Paper | 50 minutes | 5 questions |
| | Logo Computer Tasks | 50 minutes | 3 tasks |
| **Post-Tests about FRACTIONS** | Rational Number Pencil & Paper | 60 minutes | 65 questions |
| | What is a Fraction? Interview | 30-45 min. | 15 questions |
| | Boston Public School Level-4 | 60 minutes | 40 questions |
| **Post-Tests about LOGO** | Logo Pencil & Paper | 50 minutes | 10 questions |
| | Logo Computer Tasks | 50 minutes | 4 tasks |

## 1.4. The Structure of Chapter III

No significant differences between the three classes were found in the Pre-Tests; for this reason, this chapter emphasizes and presents in detail only the results of the Post-Tests. This chapter is divided into three major sections: in Section **2.1.** I shall discuss some of the results of the Logo Tests, in Section **2.2.**, some of the results of the Fractions Tests, and in Section **2.3.**, some of Debbie's results from the Logo and Fractions Post-Tests.

Most of the results are presented in simple numbers according to the children's percentage of correct answers to a given question, a subset of questions, or the test as a whole. The quantitative and qualitative analyses focus on all of the 51 children's answers for each question in each of the tests. However, in the presentation of the results, consideration is given to specific questions, according to the children's class (Experimental vs. Control 1 vs. Control 2), and to their math group (High vs. Medium vs. Low).

For establishing reliability, the quantitative and qualitative analyses of all the tests were conducted by the researcher, and in parallel, by an objective research assistant who worked independently at home. Following the researcher's requests, categories, and guidelines, the assistant also created a large data-base for all the results, so that searches and printouts of individual questions, subsets of questions, or the scores of whole tests--for each child, group of children, math-groups, or a whole class--could be performed.

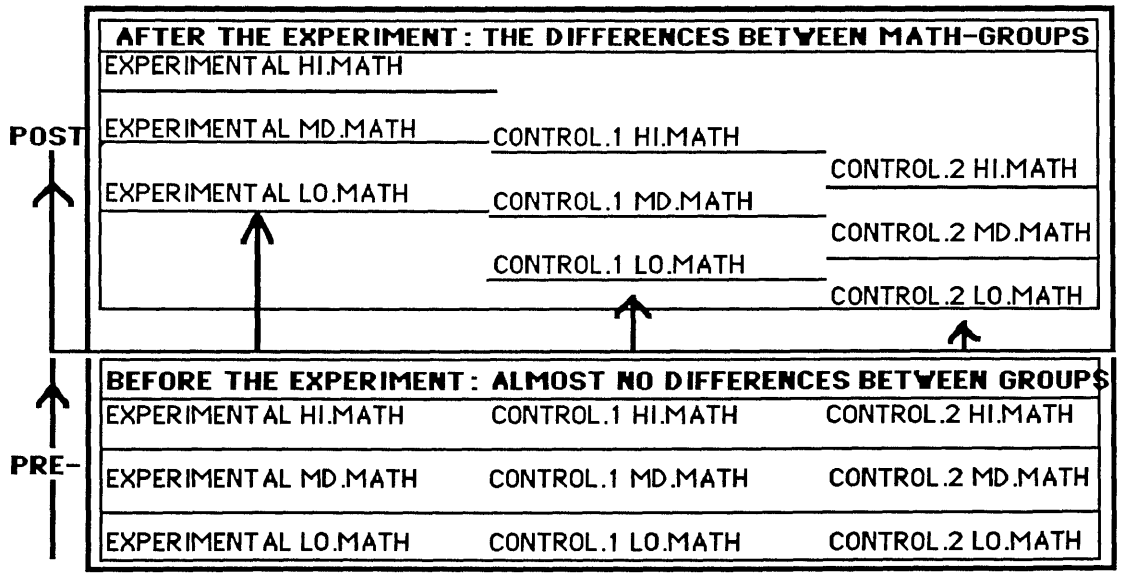
The researcher and assistant were each very careful to follow the guidelines and categories for the analyses, and at the end of their analyses systematically compared their uses of the categories, and the scores and evaluations they had come with. Only rarely (on approximately six occasions) were discrepancies found and discussed; in these cases, final decisions were made in collaboration (usually according to the research assistant's opinions).

A description of how the different results related to the task's requirements and underlying conceptual scope, as well as to the researcher's objectives and questions and to previous findings in the other studies of Logo and fractions reported by the literature (as described in Chapter I, Sections 2.3.1, and 2.3.2), will be presented throughout this chapter with their relevancy to the findings.

## 2. RESULTS FROM THE POST-TESTS

In general, the 17 children of the Experimental class, as a whole, did better than the other 34 children on all Post-Tests (Fractions and Logo). Particular trends were revealed from analyzing the Post-Tests data on fractions and Logo and relating them to the children's classes and their divisions into the three math groups. The Experimental children from the high-math group were consistently better than all the other children. Similarly, the Experimental children from the medium-math group, in many cases, were better, or at other times were the same as the high-math children from the two Control classes--and consistently better than the Control children from the medium or low-math groups. As for the low-math children from the Experimental class, they were usually better than, or the same as the medium-math children of the two Control classes, and consistently better than the Control children in the low-math group.

In other words, the differences between the Pre- and Post-Tests were much greater for the Experimental children (on all math-levels), than for those in the two Control classes. These trends had been originally predicted and were quite consistent in all the results from all the Fractions and Logo Post-Tests. These trends are sketched in the following diagram:



The rest of this chapter is divided as follows. Section **2.1.** describes the results of the <u>Logo Post-Tests</u>. Within this Section there are two subsections. Section 2.1.1., describes the results from the Pencil-&-Paper Logo Test; and Section 2.1.2., describes the results of the Logo Computer Tasks. Thereafter, Section **2.2.** describes the results of the **Fractions Post-Tests.** Within this Section there are two subsections as well. Section 2.2.1., describes the results of the Rational-Number Pencil-&-Paper Test, while Section 2.2.2. describes the results of the Boston Public Schools Math Referenced Test Level-4. Finally, Section 2.3. describes the results from **Debbie's Fractions and Logo Post-Tests**, and compares her results with those of the children from the three classes.

## 2.1. RESULTS FROM THE LOGO POST-TESTS

The Evaluation raised several points regarding fourth-grade children's learning and understanding of basic concepts of **Logo programming**, such as procedures or modularity, variables or inputs, repeat or recursion, conditionals, and other Logo commands and operations. It also investigated these children's abilities at understanding processes in programming, such as generating codes, debugging codes, and manipulating and modifying codes written by themselves or by other people (see the descriptions of the Logo tests in this chapter, Section 1.3.1., A and B, and the samples of the Logo tests in Appendix B).

During January, at the time of the Logo Pre-Tests, the 51 tested children were on approximately the same level in terms of their knowledge of Logo commands and operations, their ability to generate codes, debug codes, and understand given codes. Control class 1 scored slightly better in the Logo Pre-Tests than the other two classes, and the Experimental class scored slightly better than Control class 2. But it appears that the Experimental children (the Software-Design-Logo class) became far better programmers than those of Control class 1 (the "Integrated-Logo" class), and Control class 2 (the "Isolated-Logo" class). The following results will demonstrate in what ways the Experimental children, working under the Instructional Software Design Project conditions, most successfully improved in their programming knowledge and skills.

### 2.1.1. Results from the Logo Pencil-&-Paper Tests

This test included 7 questions which covered many aspects of the children's knowledge of Logo: Question number 1 assessed the children's basic knowledge and understanding of the Logo language, its structure, commands, and uses of these commands. Question number 2 was a classification task of the Logo commands and instructions, and examined what relationships children perceived between different Logo commands and whether they thought about them as general classes or families, or as loose functional and more task-specific. Question number 3 required that children read a linear Logo code, understand its flow of control and what the computer would do when each of the lines in the procedure was executed, and draw the picture accordingly. Question number 4 required that children modularize and optimize the code given to them in Question 3. Question number 5 assessed the children's understanding of inputs. Question number 6 examined the children's ability to transform several given procedures according to specified requests. Finally Question number 7 more deeply investigated the children's ability to read and understand short but quite confusing Logo routines and execute them through drawing. The results from each of these questions are described in the following pages.

## A. Question number 1: The Logo Language Vocabulary, Commands, and Instructions

In Question number 1 of the Logo Pencil-&-Paper Test, the children 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.**

As stated in Chapter I Section 2.3.1., knowledge of Logo programming includes, among other things, a basic knowledge of the vocabulary and the structure of the language itself. We expect children who have learned any other topic to be able to recall its basic method, style, vocabulary, structure, and idioms and explain these clearly in both functional and more abstract ways. For example, in a topic such as "Fairy tales as a form of literature," the structure of the tale, the narrator and narrative, the elements of magic and enchantment, the hero, heroine, and villain, the time and setting, and the morality--are expected to be learned, understood, be remembered by the learners, and used by them when they analyze a given fairy tale or when they themselves write a fairy tale. In the same way, we also expect children who have learned Logo programming to be able to recall many commands and instructions and explain their meaning, give examples of how to use them, not only in the context of using them while working at the computer but also as a reflective activity away from the computer. In other words, this question involved the most basic form of computer programming literacy: the children's knowledge of the structure, functions, and styles of use of the more common commands in the Logo language.

## Results:

The results are divided into two major groups of findings. The first are the 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.

The following table represents the differences between the children in terms of how many Logo commands, operations, function keys, Control keys, etc., they listed in the Post-Test.

| | EXPER. | CTR 1 | CTR 2 |
|---|---|---|---|
| HI | 32 | 16 | 8 |
| MD | 24 | 9 | 8 |
| LO | 21 | 11 | 7 |
| AVE | 25.6 | 12.0 | 8.3 |

**Logo P&P Test, Question No.1:**
**The number in each slot of this table shows the average number of commands and instructions each child's listed and explained in each math-level and class.**

Each child from the high-math Experimental class, for example, listed and explained an average of 32 commands and instructions. The advantages of the Experimental children of all math groups over the children from the two Control classes become clear from examining this table. On the whole, the Experimental children knew more of the common commands in the Logo programming language.

The children were also evaluated on the quality of their definitions and examples for each of the items they had listed. They were evaluated according to the following categories:

1) Did they spell or abbreviate the item correctly? (i.e., "FD means FORWARD"). 2) Did they give a reasonable example? (i.e., "for example FD 20"). 3) Did they describe clearly what it did? (i.e., "FD 30 makes the Turtle go forward 30 steps"). 4) Or did they use some kind of a combination of 1, 2, 3, above (i.e., listing the Logo command "FD," giving an example for how to use it, "FD 45," and explaining that "FD makes the Turtle move forward according to the specified number of steps", or "FD 45 makes the Turtle move forward 45 steps," etc). An example is shown in the table below:

**Results of the evaluation of the high-math children's answers to Question 1:**

**Class / Evaluation of the HIGH-MATH Subjects / Total per Class**

| **Exper.** | sub. 1 | sub. 2 | sub. 3 | sub. 4 | sub. 5 | sub. 6 | sub. 7 | sub. 8 | **VG** | **G** | **M** | **L** | **VL** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| evaluation | G | G | VG | G | VG | VG | G | M | 3 | 4 | 1 | 0 | 0 |
| **Ctr. 1** | sub. 1 | sub. 2 | sub. 3 | sub. 4 | sub. 5 | sub. 6 | sub. 7 | | | | | | |
| evaluat. | M | G | G | M | G | G | L | | 0 | 4 | 2 | 1 | 0 |
| **Ctr. 2** | sub. 1 | sub. 2 | sub. 3 | sub. 4 | sub. 5 | sub. 6 | | | | | | | |
| evaluat. | L | L | VL | L | M | L | | | 0 | 0 | 1 | 4 | 1 |

[KEY: VG=Very Good, G=Good, M=Medium, L=Low, VL=Very Low; sub. 1=Subject No. 1.

The right part of this table shows the evaluation given to each of the subjects in the high-math group of the three classes. For example, subject 4 of the Experimental high-math group was evaluated "G" (i.e., Good). The left side of the table shows how many "GOODs" or "MEDIUMs" etc. were in the high-math group of children of each class. For example, in Control class 1 no one was evaluated as "VG" (Very Good), whereas in the Experimental class, 3 students who wrote over 40 commands and instructions and gave a very good example and definition of each, were therefore evaluated as "VG." No one was evaluated "L" (low) or "VL" (very low) in the Experimental class. However 4 children in Control class 2 were evaluated as "L" since they listed less than 5 commands and instructions and did not provide examples or definitions for all or most of those.

## B. Question no. 2: The Classification Task

In question number 2 the children were asked:

**Look at your list of Logo instructions and commands, and please classify, group, or categorize the list. Give a title, a label, a description to each one of your groups.**

The wording of this question was prepared by the teachers who felt that each child would understand the task through recognizing a different word; they thought that some children would be more familiar with the word "categorize," others with "classify," some with "label," and others with "titles," etc.

The purpose of this classification task was to examine another aspect of children's understanding of the meaning and roles of the Logo commands; more specifically, to get a better idea of the relationships children perceived between the different commands, and see whether or not they thought about them as general classes or families, or in loose, functional, and more task-specific type of categories.

This task, furthermore, was not one of concrete classification. The children had to imagine the possible relationships between the different commands they knew, and group them according to their own experiences, as well to their more general ideas of the properties, functionalities, similarities, and differences between the different commands. This was a rather complex task for children of that age. However, I was interested in examining whether the Experimental children had a more flexible understanding of the Logo language in general; and whether or not, because of their involvement in a Project that by its very nature required "decentration" (in the Piagetian sense), they would be generally better at understanding this hypothetical task, as well as at creating more abstract and mutually exclusive groups than the other Control children.

### Results:

The results revealed three things: the first relates to who was able to perform this task in the first place; the second to how many groups and classes each child created; the third to the content of the groups in terms of their labels and members or components. Similarly to the Piagetian approach to his classification tasks, the children were not evaluated according to how large or small in number were the lists of components in the various classes they made up.

There was a correlation between being in the Experimental class and being able to perform this task. Many children in the Control classes (especially in Control class 2) wrote in their tests "I don't know," or listed several arbitrary commands on the page. During the test I tried to help some of them; I asked: "Is this your group?" or incited them to create a title for their lists of commands. However, on most occasions, the children insisted that there was "no title for that group." Other children asked me, "Do I have to write that whole list again?" (i.e. the list of commands and operations in Question 1.): they saw all of the Logo commands they knew as being members of one large group called "Logo", or in their words, "a group of Logo things." **39%** of the Control children (9 out of the 34 children) were not able to perform this task, while **100%** of the Experimental children were able to do it.

The Experimental children also generated more groups or classes. The following tables show the actual number of groups created by each child, and the average number of groups per child in each of the three classes and math-levels.

**The number of groups created by each low-math child in each class:**

| | EXPERIMEN LOW | CONTROL 1 LOW | CONTROL 2 LOW |
|---|---|---|---|
| NO. OF GRPS PER CHILD | 5, 5, 5, 5 | 3, 2, 1, 0, 0 | 2, 1, 1, 0, 0 |
| AVERAGE | 5 per child | 1.2 perchild | 0.8 per child |

**The number of groups created by each medium-math child in each class:**

| | EXPERIMEN MEDIUM | CONTROL 1 MEDIUM | CONTROL 2 MEDIUM |
|---|---|---|---|
| NO. OF GRPS PER CHILD | 5, 5, 4, 3, 2 | 3, 2, 1, 1, 0, 0 | 4, 1, 1, 0, 0 |
| AVERAGE | 4.8 per child | 1.2 per child | 1.2 per child |

**The number of groups created by each <u>high-math</u> child in each class:**

| | EXPERIMENTAL HIGH | CONTROL 1 HIGH | CONTROL 2 HIGH |
|---|---|---|---|
| NO. OF GRPS PER CHILD | 6, 6, 5, 5, 5, 4, 4, 3 | 3, 3, 3, 3, 2, 2, 2 | 3, 2, 2, 2, 1, 0 |
| AVERAGE | 4.75 per child | 2.6 per child | 1.6 per child |

The numbers in each slot of this table show the average number of groups created by each child in each math-level and class (the number 0 means that the child did not generate any groups at all). For example, the low-math experimental children created an average of 5 groups per child, while the low-math children of the two control classes created 1.2 groups per low-math child in Control class 1, and 0.8 groups per low-math child in Control class 2. This resulted in an average of **4.85** groups per child in the whole Experimental class, **2.6** per child in Control 1, and an average of **1.2** per child in Control class 2.

As for the classification categories, all children from the three classes came up with very interesting ideas. On the whole, two major schemas were found:

**1)** The <u>"abstract" classification:</u> the child constructed general or abstract relationships between the Logo commands, gave abstract reasons for including certain elements and giving certain labels or titles, classified the items according to their properties, and created mutually exclusive groups (in the Piagetian sense).

Here are some examples of the "abstract groups" taken from the children's tests:

<u>a) "things that change the color"</u>

setc, setbg, setc 1+random 14, repeat 10 [setbg :num +1 wait 15]

<u>b) "Opposites:"</u>

fd-bk, lt-rt, pu-pd

<u>c) "things that turn the turtle to different sides"</u>

rt, lt, seth (setheading)

<u>d) Things that erase:</u>

PE (Pen Erase),

DEL, CG, CT, CC,

BACKSPACE

<u>e) "Things that go to another page:"</u>

Gettools

Getpage

Escape goes to Content

f) things that write on the screen in a Logo program:

Print, Type, Label, stamp (the shapes of my letters)

g) "function keys:"

fn.1, fn.2, fn.3, fn.4, fn.5.

(this is a problematic case, since all of these are indeed function keys, but the problem is that each key is very different in its function from the other).

In this classification schema we can look at the title and predict the content of the list of Logo items in the group; or we can look at the list of items, and predict the title. "Things that erase" is the common factor of group d. above; and "changing the color" is that of group a. All members of each group share some conceptual similarity in terms of the Logo language characteristics; and defining the properties of each element of a given class determines what other items could also be placed within it.

2) The "loose" or "functional" classification: certain children constructed loose, confused, or functional groups, where the properties of the group, such as those expressed by its title and elements, did not determine the elements of that group . For example:

a) "Things to make a boat:"

fd, lt, rt, repeat, setc, pu, pd, bk, fill.

(other examples of this type: "things to make a house," "a heart," "a circle," etc.)

b) A procedure:

```
To Square
REPEAT 4 [fd 15 rt 90]
End
```

c) "A lot of things to do in Logo:"

fd, bk, lt, rt, pu, pd, repeat, setc, setbg, seth.

d) "Things used in REPEAT:"

repeat, fd, rt, lt, bk, pu, pd, fill.

e) "Things that make the Turtle move on the screen:"

fd, bk, rt, lt, setpos, home, repeat

On the other hand, in this classification schema we cannot look at the title and predict the content of the list of items in the class, for there are many other options; and we definitely cannot look at the list of items and predict the title. The similarities between the individual members of each group are vague, although it is understandable that a nine-year-old child should have put certain items together under some of these titles.

The classification task in the Logo Test was a hypothetical one by nature (i.e., it was a conceptual, imaginary, or a non-concrete classification situation for the children). There was a correlation between being in the Experimental class and creating more of the abstract, general mutually-exclusive groups and labels, and less of the "loose" and "functional" groups or classes (in fact, the abstract examples above are taken from the Experimental children's tests).

Piaget's early and late studies of young children's reasoning (2-11 year-olds) show that the pre-operational child has a tendency to group together various different events into a "loose and confused whole" (i.e., "syncretism"); that he sometimes fails to see the relationships between separate events ("juxtaposition"); and that he cannot deal with the relations between a part its a whole. All of these types of reasoning reveal common deficiencies: an inability to reverse a situation, to compensate one aspect of the situation for another, or to think about or perceive several aspects of a situation simultaneously.

According to Piaget, as the child grows older and comes into contact with opposing points of view, his thinking goes through the process of "decentration." In speech, he considers both what he wants to express and the listener's needs; in games, he considers the other child's interests and rules as well as his own; in moral judgment, he considers both the outcomes of a person's behavior and his intents, and in reasoning (or in classification tasks), he tries to consider the complexities of the problems--both the differences and similarities between the same event, the part-whole relationship and the reasons for the part's being connected with the whole and vice versa.

In his later work on classification, Piaget found that children from about 7-11 years of age (the children in my experiment were 9 -10 years old) are both capable of creating hierarchical classifications and of comprehending inclusion (i.e., stage 3 in the developmental stages of classification) of **concrete** objects (flowers, beads, etc); but when a child of the same age is asked similar questions about **hypothetical** objects, he often fails to give an adequate answer. Apparently, at that age, the child does not understand that the same relationships can exist between imaginary items in classification, as in cases where concrete items are involved. This gap between hypothetical and concrete reasoning is one example of the "vertical decalage."

The analysis of this task reveal that Logo programming is not strictly a "concrete" material, or a "formal" one in the traditional sense, but rather that it combines aspects of the formal and of the concrete; and that many Piagetian ideas and experiments, such as the one of classification, could be extended to this Logo programming domain. Furthermore, these results suggest that Logo can be further used (as both concrete and formal material) to conduct studies about children's cognitive development within the Piagetian framework; and that children who were working with the Logo conceptual objects were similar in their thinking and stages of development to children involved with concrete materials (i.e., beads, flowers) or formal concepts (imagining flowers, or imagining animals). In fact, the Experimental children probably developed a cognitive mechanism that allowed them to perceive the more abstract relations between the Logo commands and operations, and to generate more mutually exclusive groupings and titles. This finding requires further investigation.

## C. Question Number 3: Drawing Pictures According to Someone Else's Programming Code

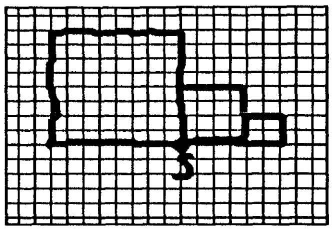
A long, linear Logo code composed of short strips of Logo primitives was given to the children. The children were asked:

3. CAN YOU DRAW THE FOLLOWING?

Logo Code: Draw It Here:

```
TO DESIGN
lt 90
fd 20
fd 20
rt 90
fd 40
rt 90
fd 40
rt 90
fd 40
rt 180
fd 20
rt 90
fd 10
fd 10
rt 90
fd 20
rt 90
fd 20
bk 20
rt 45
rt 45
fd 10
rt 90
fd 10
rt 90
fd 10
rt 40
rt 50
fd 10
END
```

This task required that children 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. The children had to read this simple procedure and understand what each line accomplished. They basically could try to execute line by line, keeping track at each step of what they had executed already, in a rather mechanistic fashion.



**This is a correct, hand-drawn picture for the given Logo code.**

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 learning how to generate effects without any consideration and understanding of how these effects were originally generated, and how they relate to one another (e.g., Kurland et al. 1987; Carver, 1987; Soloway, 1984; and others); on the other hand, a modular program emphasizes learning to program elegantly and efficiently, and with a higher-level of understanding of programming in general, and of the programming language characteristics in particular (ibid.).

These researchers also claim that children usually find it quite difficult to write modular programs, and that children should learn very early on in their process of programming how to write them, since, "programs that consist of long strips of Logo primitives [i.e. linear programs]...are nearly impossible to read, modify, or debug, even for students who have written them" (Kurland et al., 1987).

## Results:

The results from this task show that the more knowledgeable children and who, during their process of learning to program, had written linear as well as modular programs, were also better at understanding and correctly executing this confusing linear program; whereas the children who only knew how to write linear programs were not able to solve this problem accurately.

The more knowledgeable the children were, the less problematic this task was for them: they did not find this linear program impossible to read, follow, or execute. Each child was evaluated according to three categories: 1) the number of squares in the final picture (there were supposed to be three); 2) the number of times they tried to draw the picture (assessing the child's stamina and determination in solving the problem); 3) the correctness of the picture in terms of the child's use of correct proportions (i.e., 4 : 2 : 1), and the child's use of a consistent unit throughout the picture (i.e., consistently using one small square on the grid to represent 10 Turtle Steps, or 2 squares for 10 Turtle Steps, etc.). We can see these results in the following two diagrams:

**<u>The results of the Medium-Math children's drawings of the picture according to the given code</u>**

| | Number of Squares in final picture | | | | | | Number of Trials of drawing the picture | | | | | | Correct Scale & Consistent Unit | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Subject | S1 | S2 | S3 | S4 | S5 | S6 | S1 | S2 | S3 | S4 | S5 | S6 | S1 | S2 | S3 | S4 | S5 | S6 |
| Exp. Med. | 3 | 3 | 3 | 3 | 3 | x | 2 | 1 | 1 | 3 | 3 | x | y | y | y | y | y | x |
| Cot1 Med | 0 | 3 | 2 | 1 | 3 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | n | n | n | n | y | n |
| Cot2 Med | 0 | 1 | 4 | 0 | 0 | x | 1 | 1 | 1 | 1 | 1 | x | n | n | n | n | n | x |

**<u>The results of the High-Math children's drawings of the picture according to the given code</u>**

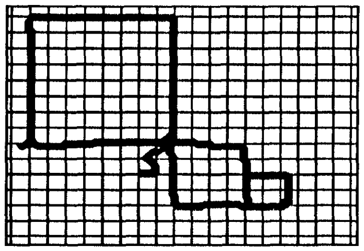
| | Number of Squares in final picture | | | | | | | | Number of Trials of drawing the picture | | | | | | | | Correct Scale & Consistent Unit | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Subject | S1 | S2 | S3 | S4 | S5 | S6 | S7 | S8 | S1 | S2 | S3 | S4 | S5 | S6 | S7 | S8 | S1 | S2 | S3 | S4 | S5 | S6 | S7 | S8 |
| Exp. high | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 5 | 1 | 2 | 1 | 1 | 1 | 2 | 2 | 1 | y | y | y | y | y | y | y | n |
| Cot1 high | 3 | 2 | 3 | 3 | 3 | 3 | 3 | x | 1 | 1 | 1 | 1 | 2 | 1 | 1 | x | y | n | y | y | n | n | y | x |
| Cot2 high | 2 | 1 | 1 | 1 | 3 | 3 | x | x | 1 | 1 | 1 | 1 | 1 | 1 | x | x | n | y | n | n | y | y | x | x |

All the Experimental children but one, from both the high-math and medium-math groups, understood the given code and executed it correctly. And when they actually drew the three squares, they always used the appropriate scaling and propotions and were consistent in their selection of the unit. On the other hand, the Control children had a more difficult time finding how many squares were in the code: most of them drew one or two, and even when they "discovered" the three squares and drew them, their scaling and units were not consistent or accurate.

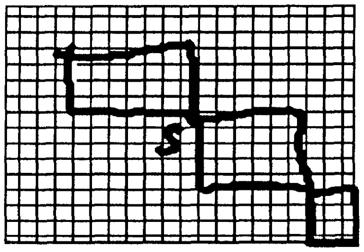
Another interesting aspect of these results came to view in the "number of trials" category. Many of the Experimental children tried more than once to draw the picture before they gave up, therefore achieved better results and finally found the right solution; but the children in the Control classes who had got it wrong in their first trial were not self-motivated to try again and check and re-check their pictures against the code, or determined 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.

The following diagrams show the most typical incorrect pictures that the children in the Control classes produced as they tried to solve this problem. It is not in the scope of this chapter to analyze these problematic versions in detail, but I shall describe some of the problems involved in these incorrect versions, so that the reader can infer what problems the Experimental children solved, or did not encounter at all, while working on this complex task.

**Incorrect Version 1.1**



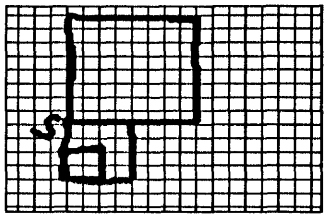
**Incorrect Version 1.2**



These two versions are very interesting because they clearly demonstrate several problems encountered by the children who produced the pictures. The children who produced these two versions had problems understanding the relationship between the squares. These were the closest versions to the correct one, since the children identified the three squares but were not able to understand the relationship between them through the use of the rt 180 degrees between the first two squares (line 10 in the code), or the bk 20 rt 40 rt 50 between the second and third (lines 19, 20, & 21 in the code).

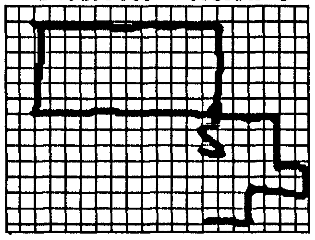
Version 1.1 shows that the child at least used appropriate scaling and proportions and was consistent in the use of the unit, and his only mistake was in the rt 180 between the first and the second square; Version 1.2. shows that the child was not accurate in the proportions of the squares, in his use of unit, or in understanding the connections between the first and second squares, or between the second and third.

**Incorrect Version 2**



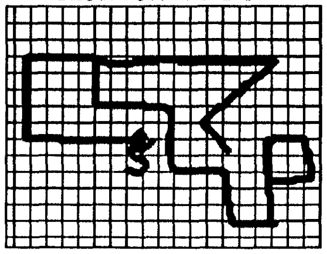
The child who produced Version 2 used the correct proportions and was consistent in his use of the unit for all three squares. However, he did not at all understand the relationships between the squares, and did not pay attention to line 10, the rt 180, or to lines 19, 20, and 21, the bk 20 rt 40 rt 50. Instead he got confused and invented a new set of relationships between the squares.

**Incorrect Version 3**



The child who produced Version 3 discovered that there was one square in the code, but was not able to figure out the other two. Although he did not use one unit consistently and followed the first 9 lines of code correctly, the rest of the code was executed erratically.

**Incorrect Version 4**



The child who produced Version 4 followed the first 6 lines of code correctly, then lost control for a while, came back to life in the last 7 lines of code, producing the small square correctly, although not in the right spaces of the grid.

What we see from these results is that the "brute-force" programmers (in all the classes) did not find it very easy to follow and execute someone else's code; although in their regular on-going programming experiences they had been able to produce a linear code quite well, they were not able to read one given by someone else; they were not determined to try and solve the problem more than once; and, as has been shown in previous findings in the field, they did not fully understand the flow of the program, the relationships between the parts, or such simple things as executing angles, fd's or bk's correctly and accurately. In general, however, the Experimental children, though some of them in their ongoing experiences could not write very elegant or efficient modular programs (in the hard, computer-science sense of the word modular), were still better able to follow the code given by someone else. They used appropriate scaling and kept the correct proportions between the squares; they were careful about being consistent in the use of their unit; and, above all, they demonstrated their stamina in trying to solve the problem more than once. Many of them tried 2 or 3 times, checked and re-checked their picture by re-reading the code line by line, until they had found the correct answer. They believed that they would eventually find the solution if they worked hard on it.

## D. Question no. 4: Simplify and Modularize a Given Linear Code

In this question the children were asked:

**AFTER YOU FINISH YOUR DRAWING (from Question 3),
TRY TO SIMPLIFY THIS PROGRAM OR "CLEAN IT."
IN OTHER WORDS, THINK ABOUT OTHER WAYS TO WRITE A
LOGO PROGRAM THAT WILL PRODUCE OR MAKE THE SAME DRAWING.
ARE THERE LOGO COMMANDS THAT WILL MAKE THIS PROGRAM
SHORTER?
(ALSO, THINK ABOUT CREATING SUB-PROCEDURES.)**

The children were asked to optimize the code given to them in Question number 3. It should here be noted that the wording of this question was prepared with the assistance of the teachers who thought that, for fourth-grade pupils, these would be better words to use, rather than "modular," "optimization," etc.

In order to answer this question, the children had to make the program clearer and shorter, stop operating on the individual command level, and start thinking in a procedural mode, using repeats, sub-routines, or procedures. They had to find a way to make the procedure called DESIGN more modular, useful, and flexible, think of what functions the different parts of the program served, and how the different parts lined up and were connected together. This task was administered in order to examine how the different children would approach a program that already worked well, and produce a different program for the same purpose by using higher-level units. In other words, when asked to read this program again, the child had to re-scan the code and mentally simulate what the program was doing overall, rather than line by line, see how the goals of the original program were achieved, and how the program could be written in another way. This is a very important aspect in analyzing children's knowledge of programming. Pea and Kurland (1984, 1987), for example, believe that students who can barely decode or comprehend text are not expected to be proficient writers (1987, pp.167); similarly, students with a low-level ability in reading, decoding, and understanding programs, programming structures, components, and their inter-relations, will not be able to write functional higher-level programs or gain insights into other domains.

## Results:

Three levels of modularization or optimization of the given code were found among the tested children. Many versions were found among the children for each Level of simplification. Many children (especially from the Control classes) were in transitional stages between the first two levels. However, all the children were always evaluated according to the higher level of their transition. The three levels are described below.

**Simplify Level 1.** On this level, the child combined all the fd 20 and fd 20 into fd 40, or rt 50 and rt 40 into rt 90, etc., which indeed made the program shorter and clearer, and was a first step towards optimization. **100%** of the Experimental children produced

Level 1 modifications, while only **78%** of Control class 1 (i.e., 4 children were not able to do it at all), and **69%** of Control class 2 (i.e., 5 children were not able to do it) produced Level 1 modification to the procedure DESIGN.

**Simplify Level 2.** On this level, the child simplified the code according to Level 1, but also tried to use REPEAT for the squares. A correct and consistent simplification in Level 2 resulted in the following procedures.

```
TO DESIGN (Version 1)
REPEAT 4 [fd 40 rt 90]
rt 90 fd 40 lt 90
REPEAT 4 [FD 20 rt 90]
rt 90 fd 20 lt 90
REPEAT 4 [fd 10 rt 90]
END
```

```
TO DESIGN (Version 2)
lt 90
REPEAT 4 [fd 40 rt 90]
rt 90
REPEAT 4 [fd 20 rt 90]
lt 90 bk 20 rt 90
REPEAT 4 [fd 10 rt 90]
END
```

A common trend on this level was to use the REPEAT correctly, and keep the right propotions between the squares (i.e., the proportion of **4:2:1** was kept by using three Repeat statements as follows: REPEAT 4 [fd **40** rt 90], REPEAT 4 [fd **20** rt 90], REPEAT 4 [fd **10** rt 90]). However, they connected the REPEAT parts incorrectly by copying the "connection commands" from the original program, which resulted, for example, in the following incorrect procedure:

```
TO DESIGN
lt 90    REPEAT 4 [fd 40 rt 90]
```

```
rt 180  REPEAT 4 [fd 20 rt 90]
bk 20   REPEAT 4 [fd 10 rt 90]
END
```

Some children on this level who used the REPEATs for creating each of the squares did not even attempt to connect the squares, and simply answered by writing three REPEAT statements, one for each square. **100%** of the high-math and medium-math children, and **75%** of the low-math children of the Experimental class simplified the given procedure on Level 2. Only **20%** of Control class 1 (i.e., only 4 children), and **25%** of Control class 2 (again, only 4 children) worked on Level 2 in these two Control class.

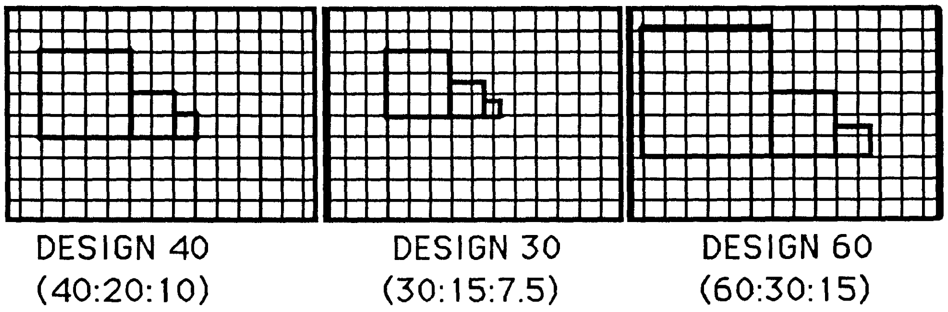
**Simplify Level 3.** On this level the child simplified the code according to Level 1 and 2 above, but also used a sub-procedure for the square with a variable for the size of the square's side. A correct and consistent simplification on Level 3 resulted in the following procedures:

```
TO SQUARE :num
REPEAT 4 [FD :num rt 90]
END

TO DESIGN
LT 90
SQUARE 40
rt 90
SQUARE 20
lt 90 BK 20 RT 90
SQUARE 10
END
```

It was also possible to make the procedure DESIGN even more generic, by keeping the proportions constant, but creating an option for drawing the whole design in different sizes:

```
TO SQUARE :num
REPEAT 4 [FD :num rt 90]
END

TO DESIGN :num
LT 90
SQUARE :num
rt 90
SQUARE :num/2
lt 90 BK 20 RT 90
SQUARE (:num/2)/2
END
```

This program will result in the following:



13 children from the Experimental class tried to write the program on Level 3; out of those, 7 succeeded in writing it correctly, and the other 6 children were almost correct. 4 children got some of the variable calls confused, and were not very consistent in their use. For example, one child wrote the following program:

```
TO SQUARE
REPEAT 4 [FD :num rt 90]
END

TO DESIGN
LT 90
SQUARE :num
```

```
rt 90
SQUARE :num
lt 90 BK 20 RT 90
SQUARE :num
END
```

However, none of the children from Control class 2 even attempted to approach this problem on Level 3, and only one child in Control class 1 worked on Level 3, but he did not write a fully accurate program. The following diagram summarizes these results from Question 4.

**The number of children whithin each math-level and class who simplified the given code in Levels 1, 2, or 3:**

| | Simplify Level 1 | | | Simplify Level 2 | | | Simplify Level 3 | | |
|---|---|---|---|---|---|---|---|---|---|
| | Exp. | Cot.1 | Cot.2 | Exp. | Cot.1 | Cot.2 | Exp. | Cot.1 | Cot.2 |
| Hi | 8/8 | 7/7 | 6/6 | 8/8 | 1/7 | 1/6 | 7/8 | 1/7 | 0/6 |
| Med | 5/5 | 3/6 | 3/5 | 5/5 | 2/6 | 1/5 | 5/5 | 0/6 | 0/5 |
| Lo | 4/4 | 4/5 | 3/5 | 3/4 | 1/5 | 2/5 | 1/4 | 0/5 | 0/5 |

The numbers in the slots of this table show how many children in each group simplified on a specific Level (1, 2, or 3): the number 4/5, for example, means that 4 out of the 5 children in the low-math group simplified the given program on Level 1. On the whole, the Experimental children were more flexible than the others and attempted to explore several different ways for producing the same drawing. They reached a more modular level of code, and many of them tried to use repeats, sub-procedures and variables.

This particular data requires further investigation for at least one reason. It was assumed that many of the children would approach this problem in a more flexible way while working at the computer than they would on the written test. A similar problem was given to the children to be solved while working at the computer; still, it was important to examine which children could also reach a high level of sophistication away from the computer by using their mental models of Logo, and by reflection. We shall see later in this chapter that the Experimental children performed even better on a similar task while working at the computer, whereas the Control children, especially those from Control class 2, performed much lower than they did here, on a similar task given to them while at the computer.

## E. Question no. 5: Inputs

In this question the children were asked a set of 8 sub-questions; each read, for example:

**HOW MANY INPUTS DOES REPEAT TAKE?____**

**GIVE AN EXAMPLE:______________________**

They were asked this same question about the following **8** Logo commands, and in the following order: **FD** (i.e., Forward, which takes one input, for example, FD 56); **BK** (i.e., Back, one input, for example, BK 97); **SETC** (i.e., Set Color, one input, for example, SETC 12); **HOME** (no inputs, one just typeS "home"); **REPEAT** (2 inputs, one being the number of times to repeat, the other the list of things to repeat, for example, REPEAT 4 [fd 50 rt 90 fd 15 rt 90]); **SETPOS** (i.e., Set Position, one input, a list of two numbers, the X,Y coordinate points on the screen. For example SETPOS [1 4]); **LT** (i.e., Left, one input, for example LT 45); and **CG** (i.e., Clear Graphics, no inputs, one just types CG).

This question was asked in order to assess the children's understanding of the structure of the commands they had most often used in their programming; whether they understood the meaning of the numbers attached to these commands, understood what inputs meant in general or in these different situations, and could give appropriate examples of their use in these contexts.

### Results:

The following tables show the percentages of correct answers for all the children in each math level and class to the 8 sub-questions about inputs.

**The average of percentages of correct answers to the "How Many Inputs" question (among the Low-Math children from the three classes):**

| | FD | BK | SETC | HOME | REPEAT | SETPOS | LT | CG | AVERAGE |
|---|---|---|---|---|---|---|---|---|---|
| EXP LO | 75% | 75% | 75% | 75% | 50% | 50% | 75% | 75% | 68.7% |
| CT1 LO | 40% | 40% | 40% | 60% | 0% | 40% | 40% | 60% | 40.0% |
| CT2 LO | 20% | 20% | 0% | 0% | 20% | 0% | 20% | 20% | 10.2% |

**The average of percentages of correct answers to the "How Many Inputs" question (among the Medium-Math children from the three classes):**

| | FD | BK | SETC | HOME | REPEAT | SETPOS | LT | CG | AVERAGE |
|---|---|---|---|---|---|---|---|---|---|
| EXP MD | 100% | 100% | 100% | 100% | 80% | 60% | 100% | 100% | 67.5% |
| CT1 MD | 50% | 50% | 50% | 34% | 17% | 17% | 34% | 50% | 37.7% |
| CT2 MD | 40% | 0% | 0% | 0% | 0% | 0% | 20% | 0% | 7.5% |

**The average of percentages of correct answers to the "How Many Inputs" question (among the High-Math children from the three classes):**

| | FD | BK | SETC | HOME | REPEAT | SETPOS | LT | CG | AVERAGE |
|---|---|---|---|---|---|---|---|---|---|
| EXP HI | 100% | 100% | 100% | 100% | 100% | 56% | 100% | 100% | 94.5% |
| CT1 HI | 100% | 100% | 100% | 100% | 27% | 13% | 87% | 100% | 78.3% |
| CT2 HI | 34% | 34% | 34% | 66% | 0% | 0% | 28% | 50% | 30.7% |

These three tables clearly demonstrate the superiority of the Experimental children over those from the two Control classes in their understanding of inputs and how they were being used in these 8 Logo commands.

Four of these sub-questions were especially difficult for many of the Control children: those on HOME, CG, REPEAT, and SETPOS. These four questions required a deep understanding of what inputs are; and also required from the children a higher-level generalization from their programming experiences.

The first two (HOME and CG) were difficult for the children who did not understand the concept of inputs in the first place. These children did not know that HOME and CG took no inputs. Only **one** child in the Experimental class did not answer these two questions (on HOME and CG) correctly; **7** children in Control class 1 and **10** in Control class 2, gave the wrong answers.

The children who connected the concept of inputs with the idea that inputs are numbers, therefore inferring that the amount of numbers in a Logo statement is the amount of inputs, had difficulty answering the questions about REPEAT and SETPOS correctly.

The number of inputs used in REPEAT is rather difficult for children to understand. They come to understand it through a complex process of using REPEATs in various situations and for various purposes in their course of programming. In fact, the children know from their actual programming experiences that, most of the time, there are "many inputs" involved in a REPEAT statement. As one child from Control class 1, who answered that REPEAT TAKES "6 or more inputs," told me during the exam, "it's because there are so many FD's and RT's and many other numbers in REPEAT." The most common answer for many of the Control children was "REPEAT takes 3 inputs"; as an example, they gave: REPEAT 4 [FD 50 RT 90].

Children with minimal experience and understanding of Logo programming understood the REPEAT statement in its local and limited use for the square. However, the children who accumulated a variety of experiences with the REPEAT statement were better able to generalize the number of inputs and on a higher level of understanding. They came to understand that, as one girl in the Experimental class told me during the exam, "everything that is in the brackets is being repeated the amount of times you tell it to. And it doesn't matter what is in the brackets or how many things are in the brackets, the whole thing [that is in the brackets] will repeat itself." **75%** of the Experimental children answered this question correctly; only **14%** in Control class 1 and **6%** in Control class 2 gave the right answer.

SETPOS posed another difficulty to the children who did not understand inputs very well, or did not use SETPOS often. When a child uses SETPOS, he always has to type in brackets one number for the x point in the coordinate system and another for the y point. Together, these two points (or numbers), x and y, specify one location on the computer screen. The children from all classes found it difficult to understand that the two numbers in brackets next to SETPOS represented only one input (one list of two numbers), rather than two. **55%** of the Experimental children answered this question correctly; **20%** in Control class 1 and **no one** in Control class 2 gave the right answer.

## F. Question no 6. Transformations of Given Procedures

In this question the children were asked to create three transformations: 1) to transform a given procedure of a circle into a procedure of a bigger circle; 2) to transform the same given procedure of a circle into a procedure for half a circle; 3) to transform a given procedure of a square into a procedure for a rectangle. It had to be done in the following way:

```
1) TO CIRCLE
     REPEAT 360 [FD 1 RT 1]
    END
 1.1) CAN YOU MAKE A BIGGER CIRCLE? (fill in the blanks)
      TO BIGGER.CIRCLE
       REPEAT___ [FD___ RT___]
      END

1.2) CAN YOU MAKE A HALF CIRCLE? (Fill in the blanks)
       TO HALF.CIRCLE
        REPEAT___ [FD___ RT___]
       END

2) TO SQUARE
     REPEAT 4 [FD 50 RT 90]
    END

2.1) CAN YOU MAKE A RECTANGLE? (Fill in the Blanks)
    TO RECTANGLE
      REPEAT ___ [___________________________]
    END
```

This question assessed several things: the children's understanding of the role of each input in the given procedures; their ability to understand the properties of these basic geometrical shapes, which are often used in Logo Graphics, and how these properties are expressed in the Logo language; their flexibility in understanding the structure and properties of these procedures as general templates, rather than specific or local ones; and their ability to manipulate the inputs and transform them in order to create new graphical effects required by another person.

## Results:

The following table shows how many children of each math level and class were able to transform the procedures correctly (i.e., 3/5 means that 3 out of 5 children of a particular math level answered the question correctly; 0/6 means that zero out of 6 answered the question correctly):

**Results from the children's transformations of given simple Logo procedures:**

| | Circ to bigger circ | | | Circ to half-circ | | | Square to rectangle | | |
|---|---|---|---|---|---|---|---|---|---|
| | Exp. | Cot.1 | Cot.2 | Exp. | Cot.1 | Cot.2 | Exp. | Cot.1 | Cot.2 |
| Hi | 8/8 | 5/7 | 0/6 | 7/8 | 5/7 | 0/6 | 7/8 | 2/7 | 0/6 |
| Med | 3/5 | 2/6 | 0/5 | 4/5 | 0/6 | 0/5 | 4/5 | 0/6 | 0/5 |
| Lo | 3/4 | 3/5 | 1/5 | 2/4 | 0/5 | 0/5 | 2/4 | 0/5 | 0/5 |

Only one low-math child in Control class 2 was able to transform the circle into a bigger circle. The rest of the children in the class were not able to perform any of the three transformations. Out of the 7 high-math children in Control class 1, only 5 knew how to transform the circle into a bigger circle or a half-circle, and only 2 of these were able to transform the square into a rectangle. Only 2 medium-math and 3 low-math children in this class were able to transform the circle into a larger circle; the rest of the children in this class were not able to perform any of these tasks.

Although several of the Experimental children were not able to perform these transformation, on the whole, most the Experimental children on all math levels answered these questions correctly, and were able to perform all the three transformations.

Two typical inaccurate answers in the transformation of the square into a rectangle were found: 1) REPEAT 2 [FD 50 RT 90]; or 2) REPEAT 4 [FD 50 RT 90 FD 20 RT 90].

In the first version, the child remembered the template of the rectangle's "REPEAT 2," but was not able to fill in the brackets correctly; in the second version, the child knew the elements in the brackets, but was not aware that these should be repeated only twice (instead of four times). Both versions demonstrate the child's inability to free himself from the rigid and local "square template" of REPEAT 4 [FD 50 RT 90], and his misunderstanding of the properties of a rectangle and how they are expressed in Logo.

### Summary of the Results from the Logo Pencil-&-Paper Test:

The Experimental children performed significantly better than the Control children on all the items of this test. If knowing Logo programming means understanding of the language's basic vocabulary and of its basic concepts and processes such as procedures, variables, and transformations of programs; and if it also means knowing how to generate code as well as to read, understand, interpret, and optimize a given code--then the results of this test demonstrate the Experimental children's superiority over the Control children in their knowledge of programming. The children's learning of Logo under the Software-Design-Logo conditions was an unusually effective way for: 1) their learning of the language (its vocabulary, its structure, and its complex concepts and processes), and 2) their acquiring reflective and sophisticated knowledge and skills that were essential for solving this Logo Test while <u>away</u> from the computer. In the following section we shall contrast the Experimental children's abilities to debug and optimize a program given to them on the computer (i.e., in a hands-on task) with those of the Control children.

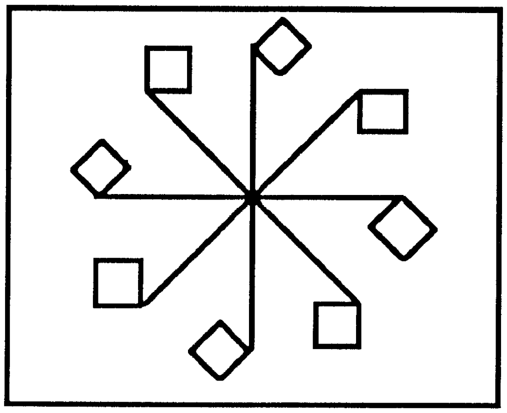
## 2.1.2. Results from the Logo Computer Tests

This test included five related sub-tasks: sub-tasks 1 and 2 were about debugging; 3 and 4 were about modularity and optimization of the given procedure; and 5 was on an even higher level of optimization. The purposes of this Logo Computer Test were very similar to those of the Logo Pencil-&-Paper Test, but assessed similar skills and knowledge of Logo while the child was actually using the computer. In the following pages the results from the different sub-tasks are described.

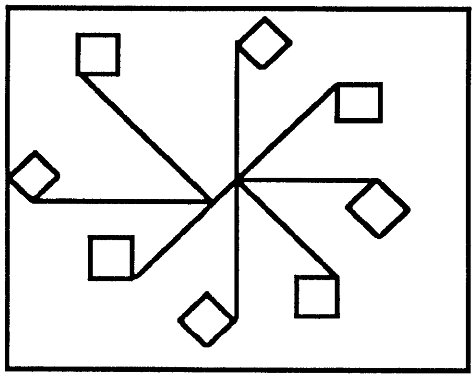
### A. Tasks 1 and 2:

In Tasks 1 and 2 of the Logo Computer Test the children were asked to solve problems related to debugging.

**1) THIS IS WHAT I WANTED TO DRAW ON THE COMPUTER;**
**THE NAME OF IT IS "FLAGS."**



**THIS WAS MY GOAL. I THINK I HAVE 2 BUGS, AND THIS IS WHAT APPEARED ON MY COMPUTER SCREEN:**



**RUN THE PROGRAM "FLAGS" (WHICH IS ON YOUR DISKETTE)**
**FIND THE 2 BUGS**
**FIX THE PROGRAM**
**DO IT UNTIL YOU GET IT TO LOOK LIKE THE TOP PICTURE.**

**2) THIS IS THE PROGRAM WITH THE TWO BUGS. PLEASE MARK YOUR CORRECTIONS HERE:**

[The comments in the brackets, and the location of the bugs were not included in the test-form that the children received.]

```
TO FLAGS
ht home
fd 45                 [draws the first stick of the first flag. It is the top vertical stick]
rt 45
fd 10                 [the next 9 lines draw the first flag]
rt 90
fd 10
rt 90
fd 10
rt 90
fd 10
rt 90                 [finishes drawing the first flag]
lt 45                 [turns left 45]
bk 45                 [goes back to home position]
rt 45                 [turns right 45 degrees, preparing for the second stick...]

fd 30 fd 15 rt 45     [draws the second stick]
fd 10 rt 90           [starts drawing the second flag]
fd 10 rt 90
fd 10 rt 90
fd 10 rt 90           [finishes drawing the second flag]
lt 45 bk 30 bk 15     [goes back to home position]
rt 45
```

| | |
|---|---|
| fd 12 fd 12 fd 21 | [draws the third stick] |
| rt 40 rt 5 | |
| fd 10 rt 90 fd 10 rt 90 | [starts drawing the third flag] |
| fd 10 rt 90 fd 10 rt 90 | [finishes drawing the third flag] |
| lt 45 bk 45 rt 45 | [goes back to home position and turns 45 degrees] |
| fd 40 fd 5 rt 45 | [draws the fourth stick] |
| repeat 1 [fd 10 rt 90 fd 10 rt 90 | [starts drawing the fourth flag] |
| fd 10 rt 90 fd 10 rt 90] | [finishes drawing the fourth flag] |
| lt 45 rt 45 bk 45 | [turns and goes back to home position] |
| | |
| fd 45 rt 45 | [draws the fifth stick] |
| fd 10 rt 90 | [starts drawing the fifth flag] |
| fd 10 rt 90 | |
| fd 10 rt 90 | |
| fd 10 rt 90 | [finishes drawing the fifth flag] |
| lt 45 bk 45 rt 30 rt 15 | [goes back to home position and turns 45 degrees preparing for next flag] |
| | |
| fd 45 rt 45 | [draws the sixth stick] |
| repeat 2 [ fd 10 rt 90 | [starts drawing the sixth flag] |
| fd 10 rt 90] | [finishes drawing the sixth flag] |
| lt 45 bk 30 rt 30 rt 10 rt 5 | [goes back only 30 steps instead of 45 to home position. This is the **first bug**. Then it turns right 45 degrees preparing as usual for the next flag] |
| fd 65 rt 45 | [draws the seventh stick. This is the **second bug,** it goes fd 65 instead of 45 steps] |
| fd 10 rt 90 | [starts drawing the seventh flag] |
| fd 10 rt 90 | |
| fd 10 rt 90 | |
| fd 10 rt 90 | [finishes drawing the seventh flag] |
| lt 45 bk 65 rt 45 | [goes back to beginning of stick. This is a **sub-bug**, related to the second bug. Once one has discovered the second bug, this one is easy to find.] |

| | |
|---|---|
| fd 65 rt 45 | [draws the eighth stick. But goes fd 65 instead of 45, a **sub-bug** to the second bug] |
| fd 10 rt 90 | [starts drawing the eighth flag] |
| fd 10 rt 90 | |
| fd 10 rt 90 | |
| fd 10 rt 90 | [finishes drawing the eighth flag] |
| lt 45 bk 65 rt 45 | [goes back 65 steps. This **sub-bug** is also related to the second bug] |
| **END** | |

Tasks 1 and 2 of the Computer Test were designed to assess: 1) the children's method and ability to debug a program given to them by someone else; 2) their understanding of the Logo-code structure on the computer; and 3) their understanding of the same Logo-code structure given on paper.

Debugging is a very complex skill. One central aspect of learning to program "is learning how to debug faulty programs...this aspect has been identified as one of the "powerful ideas" that can generalize far beyond the programming context in which it is acquired" (Carver & Klahr, 1986, p.2).

In their systematic analysis of the debugging processes, Carver and Klahr distinguished between a discrepancy, that is, the difference between the original goal ("what I wanted to draw") and the program's output ("what came out on the computer screen"), and the bug itself (the error in the program that caused the discrepancy). They found four phases in the debugging process, which I also used for the analysis of this computer test. The four phases are described in detail in Carver & Klahr (1986, pp. 3-6), and I briefly summarize them here:

Phase 1) "Program Evaluation." In this phase, a knowledgeable child (in terms of his debugging skills) is expected to run the program and compare the goal drawing with the actual output. When these two (the goal and actual output) do not match perfectly, then the child needs to identify the bug, locate the bug, and correct it.

Phase 2) "Bug Identification." In this phase, the child must pay attention to the pictorial features of the output, generate a description of the discrepancies between the goal and the output, and think about several possible types of bugs that might be responsible for the discrepancy. After considering all possible options, the child hypothesizes on one specific bug (the reason for each discrepancy).

Phase 3) "Bug Location." In this phase, the child enters the Logo code, and through identifying the structure of the code, investigates the probable location of the bugged command in the code.

Phase 4) "Bug Correction." In this phase, the child replaces the bug with a correction, and re-evaluates the program (retracing the steps described in the "Program Evaluation," phase number 1 above).

Carver and Klahr created a related production-system model called GRAPES. They also specified in detail two sets of debugging heuristics, one for identifying bugs, and one for locating them in a given code, and identified several operators that used in the process of debugging, whether done by humans or by the computer-program they had developed (i.e., the MATCH, CONTRAST, EXAMINE, INTERPRET, GENERATE, RUN, ENTER, SKIP, READ, DELETE, and INSERT operators, all of which are sub-skills necessary for processing information and performing actions in the process of debugging). They used this very detailed and formal task analysis of Logo debugging skills (based on an expert's skills of debugging) to assess children's performances of debugging, and found that children were not able to develop these sophisticated skills, follow the four phases, or use the needed operators and heuristics in what they called, "a normal Logo course" (ibid. p.12). As a result, Carver (1986-87) developed an intensive Logo course that explicitly and directly teaches children the four phases, the operators, and the heuristics used in the process of debugging. Through the use of several evaluative tests with experimental and control children, she found that her debugging course was very effective and resulted in children's developing very sophisticated and systematic debugging skills that were even transferred to other more or less related debugging tasks in situations other than Logo programming. In the context of my research, non of the tested children (from the three classes) were formally instructed as were Carver's pupils. Also, unlike Carver in her research goals, I was not interested in the transfer of the debugging skills; rather, I was interested in investigating children's ability to learn these skills through their unique, open-ended, and intensive programming experience in the Instructional Software Design Project. However, I was strongly inspired by Carver's debugging tasks (1987) when I created these particular computer tasks for my research (all the ones described above and below), and I was also interested in using her debugging-process model in analyzing the children's knowledge of debugging.

## Results (for Tasks 1 and 2):

On the whole, it appears that the Experimental children developed the debugging skills required for solving Parts 1 and 2 during their processes of designing and programming instructional software, because they performed much better on these two sub-tasks, were faster at identifying the bugs, were able to locate the bugs in the given program on the computer and on paper, then re-evaluated the program and created an output that corresponded perfectly with the original goal given to them in that task.

The following data-table shows the results for Parts 1 and 2, which required that the children run the given program, analyze the features of the picture (or defaulted output), identify the discrepancies, enter the Logo code on the computer, locate, one by one, the different bugs that were causing the discrepancies, fix the program on the computer, and finally, add the corrections on the program that was written on the paper.

**Results from the debugging tasks for all the 51 children:**

| | 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<br>1- one bug | 17 children- yes<br>100% succeeded | 17 children- yes<br>100% succeeded | 15 minutes per child |
| Control 1 N=18 | 9 - all bugs<br>4 - one bug<br>5 - none | 13 children -yes<br>5 children - no<br>70% succeeded | 8 children - yes<br>10 children -no<br>44% succeeded | 35 minutes per child |
| Control 2 N=16 | 2 - all bugs<br>4 - one bug<br>10 - none | 6 children - yes<br>10 children - no<br>37% succeeded | 2 children - yes<br>14 children - no<br>12% succeeded | 55 minutes per child |

This data-table speaks for itself. The superiority of the Experimental children (on all math-levels) over the other pupils is clear, as is that of Control class 1 over Control class 2 (the former having programmed more intensively and integratively than the latter).

The first thing all the Experimental children did was to change the HT (Hide Turtle), which was at the very beginning of the procedure, to ST (Show Turtle), so that they could follow the Turtle as it executed the code. This was one of the strategies that helped them to analyze the program's features, identify the repeated pattern of the picture, its units, and the way these units related to each other, which resulted in their finding the bugs rather quickly.

On the other hand, the first strategy that most of the children in Control class 2 and many of those in Control class 1 used, rather than running the program several times, analyzing it, and trying to identify its structure and bugs, 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 children got confused because it did not work, and they were sure that they had located a bug; they "fixed the bug" but, of course, with little success. The procedure then became very different from the original one. Instead of trying a new strategy, these children erased everything and started to copy the procedure into the direct mode again, which resulted in the same thing happening again, and so on.

Carver and Klahr (1986) list three reasons why children usually fail to acquire debugging skills: "a) Debugging is a complex skill; b) it requires extra capacity; and c) it is rarely taught directly" (p. 27-30). I definitely agree with the first two reasons, but believe the third should be revised.

To my mind, children who study programming under exceptionally rich learning conditions should be able to acquire basic debugging skills. First and foremost, they should acquire these skills by frequently exploring and experiencing debugging situations in the course of their learning how to program, rather than just being taught these skills directly and beforehand. One should not necessarily assume that a deficiency in children's debugging skills is a direct result of their not having received instruction in the subject. It also seems that though explicit teaching of debugging might be needed in some cases, it does not always needed to take place before a child has experienced and acquired some of these skills on his own.

The Instructional Software Design Project offered another method for helping children to acquire these skills, a method that is radically different from Carver's (1987) instructional method. During their course of designing and programming instructional software, the Experimental children went through many phases of debugging. There were many situations when a child designed something in his Designer's Notebook in one way, but when he actually programmed it there were discrepancies between his original drawing and the program's actual output. In some cases the child took advantage of the unexpected outcome, re-evaluated his original plan or design, and decided to go on with whatever had appeared on the screen (in other words, the child modified his plan instead of treating the output as a result of discrepancies or bugs). However, in other cases, the child felt a strong need, for various reasons, to find the discrepancies between the program and its plan, and re-evaluate the program's output, rather than the plan; in other words, he located the bugs, and fixed them so that the program's output would match the original design drawn in his Notebook.

I find these two aspects equally important in children's programming processes. For various reasons related to children's developing creative skills or their breaking away from rigidity into developing flexible strategies in their thinking, designing, and programming, it would seem desirable to put them in situations where they might sometimes break away from their original plans and designs and learn how to benefit from unexpected outcomes or from programming constraints. However, children should also learn how their plans can be followed and implemented precisely, and how to locate and correct the discrepancies between their hand-drawn picture and the program that has been created for drawing that picture on the computer.

It appears that through this combination of thinking and programming strategies, the Experimental children learned to become good debuggers, as well as creative and flexible problem-solvers (we will see evidence for this in the children's solutions to sub-tasks 3, 4, and 5). I therefore suggest that children who learn under the Carver conditions will probably develop excellent debugging skills and will be able to transfer them; but I am not quite sure how creative these children will be in their ongoing problem-solving or programming processes, or how flexible they will become in terms of their cognitive development.

To summarize, in contrast to Carver's and Klahr's (1986) and Carver's (1987) assumptions and findings, the children of the Experimental class, who were not directly taught debugging, but instead experienced a variety of debugging situations during their course of learning how to program, were able to learn basic debugging skills.

What still remains unknown and requires further investigation is the Experimental children's ability to transfer these skills. It was not in the scope of my research to gather information about transfer in the same way that Carver did in her very systematic studies; however, I feel safe in saying that the Experimental children would probably be able to transfer these skills better than those from Control classes 1 and 2.

## B. Tasks 3 and 4:

Tasks 3 and 4 of the Logo Computer Test were designed to assess, through the use of the same procedure FLAGS, the children's understanding of the Logo-code structure, and their ability to modularize and optimize someone else's procedure, which was given to them on the computer.

**3) NOW, TRY TO SIMPLIFY THIS PROGRAM, OR "CLEAN IT."**
**IN OTHER WORDS, THINK ABOUT OTHER WAYS TO WRITE A LOGO PROGRAM THAT WILL PRODUCE THE SAME PICTURE.**
**CAN YOU MAKE IT SHORTER?**
**(THINK ABOUT CREATING SUB-PROCEDURES...)**

**WRITE YOUR SHORTER VERSION HERE:**
**AND ON THE COMPUTER, CALL IT "FLAGS1"**
**TO FLAGS1**

**4) NOW, TRY TO WRITE THE SAME PROGRAM WITH INPUTS (IF YOU DID NOT DO IT ALREADY), SO YOU CAN TYPE: FLAGS 45 10 AND IT WILL DRAW THE PICTURE OF 8 FLAGS AS BEFORE. OR IF YOU TYPE: FLAGS 65 20 IT WILL MAKE THE SAME PICTURE BUT BIGGER.**

**WRITE YOUR PROGRAM HERE:**

**AND ON THE COMPUTER CALL IT "FLAGS2"**

(Note: For these two sub-tasks, the wording was prepared with the assistance of the three teachers who thought that, for fourth-grade pupils, these would be the best words to use for this purpose, instead of "modularity" or "optimization," etc).

In Tasks 3 and 4, the children were asked to optimize the code given to them in Tasks 1 and 2 of this test. In order to solve these sub-tasks, the children had to make the procedure clearer and shorter by ceasing to operate on the individual command level, and starting to think in a procedural mode, using repeats, sub-routines, procedures, and later also thinking about inputs. They had to think about how to make the procedure FLAGS more modular, useful, and flexible. They had to think about what functions the different parts of the program served, and how the different parts were lined up and connected together (some of these issues they had already had to think about while working on Parts 1 and 2 above).

These tasks were also administered in order to examine how the different children would approach a procedure that already worked well, and produce a different program for the same purpose by using higher level units. As in Question number 4 of the Logo Pencil-&-Paper Test, the children, when asked to read the FLAGS program again, after having debugged it, had to re-scan the code and mentally simulate what the procedure was doing overall, what its whole structure was (rather than each separate line), see how the goals of the original procedure were achieved, and how it could be written in more sophisticated way.

## Results (for Tasks 3 and 4):

As we saw from the findings of Question number 4 in the Logo Pencil-&-Paper Test, three levels of modularization or optimalization of the given code were found among the tested children.

**Simplify Level 1.** On this level, the child combined all the fd 20 and fd 25 into fd 45, or rt 30 and rt 15 into rt 45, etc. which indeed made the program shorter, and clearer, and was a first step towards optimization. **100%** of the Experimental children produced Level 1 modifications; but only **66%** of Control class 1 and **25%** of Control class 2 produced Level 1 modification for the procedure FLAGS (i.e., complete failure of 6 children in Control class 1 and 12 children in Control class 2).

This finding is very interesting since in the Logo Pencil-&-Paper Test many more children from the Control classes were able to simplify the given procedure DESIGN (Question number 3 in that test) on Level 1. This might be a result of two things. First, DESIGN was an easier procedure to simplify than FLAGS; secondly, many of the children, especially from Control class 2, had already become confused during their process of debugging, and changed the original procedure given to them in Tasks 1 and 2 in such a way that it was later difficult to identify the procedure's structure and units for Tasks 3 and

4. In fact, many of these children who worked in direct mode and tried to copy, line by line, the procedure given to them on paper, made many mistakes while typing it into the computer and got rather lost in the process.

**Simplify Level 2.** On this level there were two versions, which I call Level 2.1. and 2.2. On Level 2.1., the child simplified the code according to Level 1, but also tried to use REPEAT for the flags at the ends of the sticks. A correct and consistent simplification in Level 2.1 resulted in the following procedures.

```
TO FLAGS1 (Version 1)
home
fd 45 rt 45
REPEAT 4 [fd 10 rt 90] lt 45 bk 45 rt 45
fd 45 rt 45
REPEAT 4 [fd 10 rt 90] lt 45 bk 45 rt 45
fd 45 rt 45
REPEAT 4 [fd 10 rt 90] lt 45 bk 45 rt 45
fd 45 rt 45
REPEAT 4 [fd 10 rt 90] lt 45 bk 45 rt 45
fd 45 rt 45
REPEAT 4 [fd 10 rt 90] lt 45 bk 45 rt 45
fd 45 rt 45
REPEAT 4 [fd 10 rt 90] lt 45 bk 45 rt 45
fd 45 rt 45
REPEAT 4 [fd 10 rt 90] lt 45 bk 45 rt 45
fd 45 rt 45
REPEAT 4 [fd 10 rt 90] lt 45 bk 45 rt 45
END
```

No one in Control class 2 was able to systematically revise the FLAGS procedure on this Level. Most of these children used repeats in some places but not in others; in other words, they did not understand the structure of the procedure or the units that needed to be repeated in full. Many children in Control class 1, who managed to reach Level 2.1 stopped there, not realizing that a higher level of repeats could be used (i.e., nesting a repeat within a repeat). 90% of the Experimental children succeeded in simplifying the procedure on this level.

```
TO FLAGS1 (Version 2)
REPEAT 8 [fd 45 rt 45 REPEAT 4 [fd 10 rt 90] lt 45 bk 45 rt 45]
END
```

**Simplify Level 3.** On this level there were also two possible versions, which I call Level 3.1., and 3.2. On Level 3.1. children simplified the code according to Level 1, 2.1, and 2.2. as described above, but also used a sub-procedure for making a FLAG.

```
TO FLAG
fd 45 rt 45 REPEAT 4 [fd 10 rt 90]
END

TO FLAGS3
REPEAT 8 [FLAGG lt 45 bk 45 rt 45]
END
```

<u>Or, still on the same Level, other children wrote:</u>

```
TO FLAG
fd 45 rt 45 REPEAT 4 [fd 10 rt 90] lt 45 bk 45 rt 45
END

TO FLAGS
REPEAT 8 [FLAG]
END
```

15 out of the 17 Experimental children were able to write the procedure FLAGS on this Level; only 5 out of 18 children in Control class 1 succeeded, but no one in Control class 2.

On Level 3.2. children used variables, one for the stick and another for its flag. 12

Experimental children were able to do it:

```
TO FLAG :num1 :num2                [num1=stick's length, num2=flag's size]
fd :num1 rt 45 REPEAT 4 [fd :num2 rt 90] lt 45 bk :num1 rt 45
END
TO FLAGS4 :num1 :num2
REPEAT 8 [FLAG :num1 :num2]
END
```

## C. Task number 5:

In this part the children were asked:

**CAN YOU MAKE A PROGRAM THAT DRAWS THE SAME PICTURE, BUT EACH FLAG WILL BE IN A DIFFERENT COLOR?**

10 Experimental children, and 2 children from Control class 1 managed to solve this task. They did it as follows:

```
TO FLAGS5 :num1 :num 2
REPEAT 8 [FLAG :num1 :num2  SETC 1 + random 14]
END
```

5 Experimental children used a variable even for the color (instead of using random). Their reason was: "When I used the random I sometimes got the same color twice, and you wanted each flag to be in a different color, right?" Here is what these 5 children did:

```
TO FLAGS6 :num1 :num2 :num3        [num1=stick's length, num2=flag's size, num3=pen color]
REPEAT 8 [FLAG :num1 :num2 setc :num3+1]
END
```

## Summary of the Results from the Logo Computer Test

On the whole, the Experimental children were more flexible and attempted to explore a greater variety of ways for producing the same drawing. They understood and reached a more modular level of code, and many of them tried to use repeats, sub-procedures, and variables.

Interestingly enough, all the Experimental children, who had already performed much better than the Control children in the similar task (Question number 4) of the Logo Pencil-&-Paper Test, performed even better here, using the computer. But the children from Control class 2 got more confused at the computer, and performed less well than they had on the Pencil-&-Paper task. Control class 1 was somewhere in between: the high-math children, like those from the Experimental class, performed much better at the computer, and the medium and low-math children performed similarly to those from Control class 2, that is, far less successfully than they had in the Pencil-&-Paper task.

## 2.2. RESULTS FROM THE FRACTIONS POST-TESTS

All the teaching of fractions, for all the three classes, was conducted for two months, during regular math lessons only and following the City of Boston's requirements and traditional teaching methods. While the Instructional Software Design Project was being conducted, the Experimental class was <u>not</u> provided with any additional instruction on fractions besides the material they had covered in their regular math lessons with the other Control children.

In the following subsection, selected results from the two fractions Post-Tests, the Rational-Number Pencil-&-Paper Test, and the Boston Public Schools Curriculum Referenced Math Test are presented. The results of the What Is A Fraction? Interview are not available for this Thesis. They will be presented in detail in a forthcoming paper entitled: "What Is A Fraction? Children Constructing Meaning for Fractions Through Instructional Software Design" (Harel, in progress).

### 2.2.1. Results from the Rational-Number Pencil-&-Paper Test

The Experimental class as a whole did better on the Rational-Number Concepts Post-Test than the other two Control classes, as is shown in the following diagram:

**RATIONAL NUMBER PENCIL & PAPER TEST**

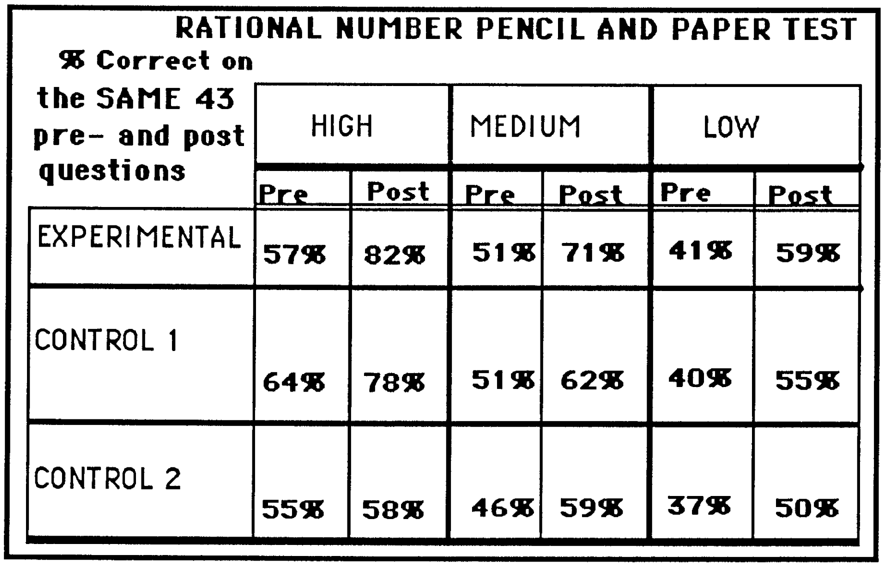
**% CORRECT POST-TEST** **(For All 65 Questions)**

| | HIGH | MEDIUM | LOW |
|---|---|---|---|
| EXPERIMENTAL | 83% | 70% | 57% |
| CONTROL 1 | 76% | 55% | 50% |
| CONTROL 2 | 57% | 53% | 47% |

The diagram shows, for example, that the high-math children from the Experimental class scored an average of 83% correct answers in this entire test. The Experimental low-math children scored an average of 57% correct on this test, exactly the same as the high-math children from Control class 2 and higher than all the medium-math children from the two Control classes.

The Post-Test included 65 questions. Out of these, 60 were taken from the Rational-Number Project (RN Project, Lesh et al., 1983 pp. 309-336). The remaining 5 were designed by the researcher and included word problems and representation-constructions (see Appendix B).

Of the 60 RN Project questions, 43 were given to the children in the Pre-Test, then again in the Post-Test. The following diagram shows the average percentages of correct answers in the Pre-Tests and the Post-Tests of each math group, and within each class:



**RATIONAL NUMBER PENCIL AND PAPER TEST**

| **% Correct on the SAME 43 pre- and post questions** | HIGH | | MEDIUM | | LOW | |
|---|---|---|---|---|---|---|
| | Pre | Post | Pre | Post | Pre | Post |
| EXPERIMENTAL | **57%** | **82%** | **51%** | **71%** | **41%** | **59%** |
| CONTROL 1 | **64%** | **78%** | **51%** | **62%** | **40%** | **55%** |
| CONTROL 2 | **55%** | **58%** | **46%** | **59%** | **37%** | **50%** |

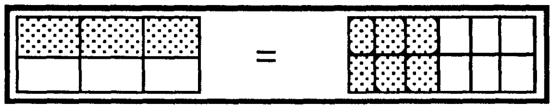
In the case of the test-scores of the children from the Experimental group, the difference (in the percentage of correct answers) between the Pre-Test and the Post-Tests scores was almost <u>twice</u> as great as in that of the children from Control class 1, and <u>two-and-a-half</u> times as great as in that of Control class 2. This is shown in the following diagram:

**RATIONAL NUMBER PENCIL AND PAPER TEST**

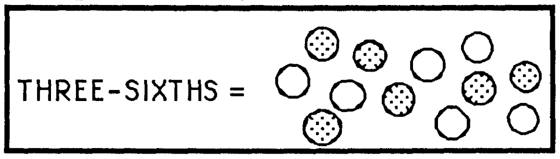
| Post minus Pre for SAME 43 questions | HIGH | MEDIUM | LOW | Class's Average |
|---|---|---|---|---|
| EXPERIMENTAL | 25 % | 20 % | 18 % | 21 % |
| CONTROL 1 | 14 % | 11 % | 11 % | 12 % |
| CONTROL 2 | 3 % | 13 % | 12 % | 9 % |

As described previously in Chapter I, Lesh et al. (1983) tested their subjects on different "translation modes" between representations of fractions. In their large-scale testing program they found that some translations were more difficult to process than others (from the easiest type number1, to the hardest type number 9):

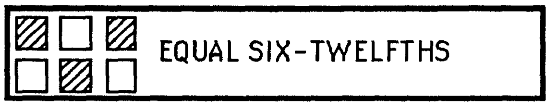
1. Translating word representation into word representation (this was the easiest for children, for example, three-sixths equal six-twelfths).
2. Translating symbols into symbols (for example, 3/6 = 6/12).
3. Translating symbols into words (for example, 3/6 equal six-twelfths).
4. Translating words into symbols (for example, three-sixths = 6/12).
5. Translating a picture into a picture, for example,



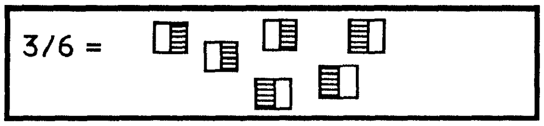
6. Translating words into a picture, for example,



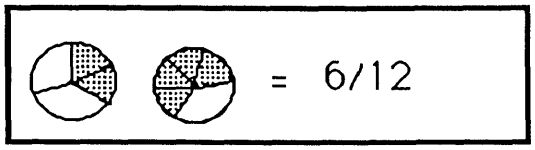
7. Translating pictures into words, for example,



8. Translating symbols into pictures, for example,



**9.** Translating pictures into symbols (the hardest for children of all ages), for example ,



In the following sections, the results from the last four modes of translations will be presented (i.e., Level 6, 7, 8, and 9 above), since they were the most difficult for children of all ages in the Lesh et al., and Behr et al. studies (1983), and were equally difficult for all the children in this study's Pre-Tests. In the Post-Tests, however, these four translation modes were still relatively difficult for the children from the two Control classes; but the Experimental children dramatically improved on these four modes in their Post-Tests.

From this point forward, the scores of the children in the RN Project will be considered as standard scores (in their percentage of correct answers). In this way, the children from the Experimental group (N=17) will be compared not only with those from the two Control classes (N=18, N=16), but also with the fourth-grade children from the RN Project (N= 43). In other words, the 17 Experimental children will be compared from here on with 77 other children on all the selected questions from this Post-Test.

## A. Results from the translations of rational-number **written** representations into **pictorial** representations (sixth level of difficulty, translation number **6** above):

In order to answer questions **24, 5, 4, 13, 11,** and **7** in the Post-Test (see Appendix B), the children were required to translate a written representation of a rational number into pictorial representations of a rational number. The questions are listed here according to their level of difficulty for the children in the RN Project (Lesh et al., 1983, pp. 309-336). The following diagram shows the scores of all 51 children on these questions. The children are grouped according to their class (Experimental, Control 1, and Control 2) and math level (high, medium, or low).

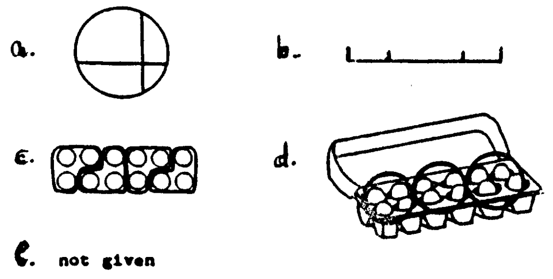
| | EXPER. | CTR 1 | CTR 2 | STAND. |
|---|---|---|---|---|
| HI | 88% | 80% | 75% | |
| MD | 84% | 66% | 80% | |
| LO | 75% | 66% | 63% | |
| AVE | 82% | 70% | 72% | 67% |

WORDS TO PICTURES TRANSLATIONS
QUEST. 24, 5, 4, 13, 11, AND 7.

This table demonstrates the advantages gained by the children from the Experimental group over those from the two Control classes and the RN Project, once the Experiment was completed.

Question 7 was the most difficult in this subset, and the 51st most difficult in the whole set of 60 questions (see Appendix B). The children were asked:

7. WHICH PICTURE SHOWS FOURTHS?



In order to answer this question, the children had to translate the word "fourths" into the four options of discrete pictorial representations. Option b. for example, was confusing since a measurement was imposed on the picture but was not applicable. Option c., which was the correct answer, included a perceptual distraction. The children first had to identify the number of parts in each option, then compare the sizes of the parts in each object and see if the parts within each object were equal. This process of finding the right answer for question 7 was rather complex. The options given as answers were definitely not "standard" ones (such as one circle divided into four equal parts). Option a., considered to be the most basic and familiar representation of a circular region, assessed the children's knowledge of "the parts must be equal" schema. The following table shows the Post-Test scores for question number 7:

| | EXPER. | CTR 1 | CTR 2 | STAND. |
|---|---|---|---|---|
| HI | 78% | 70% | 32% | |
| MD | 40% | 32% | 25% | |
| LO | 50% | 0% | 20% | |
| AVE | 56% | 34% | 25% | 30% |

% CORRECT ON QUESTION # 7
TRANSLATING THE WORD "FOURTHS"
INTO A PICTURE REPRESENTATION.

The Experimental children scored significantly higher than those in the two Control classes and in the RN Project. The medium and low-math children from the Experimental group scored higher than the medium and low-math children from the two Control classes, higher than the Control 2 high-math children, and also higher than the RN Project children.

In the Pre-Tests, the average of percentages of correct answers for this question was 12% for the Experimental children from the medium and low-math groups taken together. The other children had similar results in the Pre-Test. However, the Experimental children made far more progress than the other children in the period between January to June.

## B. Results from the translations of rational-number **pictorial** representations into **written** representations (seventh level of difficulty, translation number 7 above):

In order to answer questions **8, 32, 2, 49, 30, 35, 23, 25, 36, 22, 37, 38, 50**, and **60** in the Post-Test (see Appendix B), the children were required to translate a pictorial representation of a rational number into written representations of a rational number. The questions are listed here according to their level of difficulty for the children in the RN Project (Lesh et al., 1983, pp. 309-336). The diagram below shows the score of all 51 children on these questions, and the score of the fourth-graders in the RN Project.

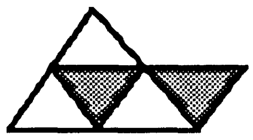
| | EXPER. | CTR 1 | CTR 2 | STAND. |
|---|---|---|---|---|
| HI | 85% | 80% | 57% | |
| MD | 60% | 60% | 46% | |
| LO | 59% | 56% | 50% | |
| AVE | 68% | 65% | 51% | 56% |

PICTURES TO WORDS TRANSLATIONS
QUEST. 8, 32, 2, 49, 30, 35, 23, 25
36, 22, 37, 38, 50, AND 60.

This subset (Level 7) was much more difficult than the previous one (no. 6 above). Questions 38, 50, and 60 were particularly complex (see Appendix B). Therefore, the differences between the groups in this subset, as a whole, were not as significant as those in the previous subset. However, the differences were there, and the Experimental children did have an edge over the rest of the children. The differences between the Experimental children of all three math groups and those of Control class 2 are very high: between the high-math children from the Experimental and Control 2, the difference is 28%, between the medium-math children,14%, and between the low-math children, 9%. In addition, the low-math children of the Experimental group did better than all the children in Control Group 2. When analyzing specific questions of this subset (pictures into words translations), we can see more dramatic differences.

**<u>Question 50</u>** is considered 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 children (Lesh et al., p. 326). The children were asked:

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

This complex question was one of the two most difficult in this subset, and the 55th most difficult in the whole set of 60 questions. It presented children 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 children 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 do not mention "fifths," but rather "five" (the denominator is "five"). The following diagram shows the scores in their percentage of correct answers for question 50.

| | EXPER. | CTR 1 | CTR 2 | STAND |
|---|---|---|---|---|
| HI | 88% | 78% | 16% | Not |
| MD | 60% | 16% | 25% | given to 4th. |
| LO | 50% | 0% | 40% | 7th gr. got |
| AVE | 66% | 31% | 27% | 26% |

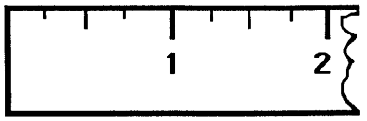
% CORRECT ON QUESTION # 50 TRANSLATING THE DENOMINATOR IN A PICTURE SHOWING TWO FIFTHS, INTO THE WORD "FIVE."

The Experimental children's scores were twice as good as those of the Control children, and even twice as good as those of the <u>seventh</u> graders from the RN Project. The low and medium-levels of the Experimental group did much better in this question than the low and medium-level of Control 1, and all the levels of Control 2. The average percentage of correct answers for the Experimental low and medium-levels taken together was <u>55%.</u> The average of Control 1's low and medium-math levels, taken together with all three math groups in Control 2, was only <u>32%.</u> This finding is significant to this study because question 50 was a rather complex question according to Lesh et al., and Behr et al. (1983).

In addition, my assumption here regarding this particular question is that the children who were better at Logo programming were probably better at answering this question correctly. What Lesh et al. (1983) and Behr et all. (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 children who looked at the picture with "Logo eyes" and decomposed it into its five geometrical components. Decomposing a given picture into its geometrical components is a common process in Logo programming, and a skill children usually acquire in their ongoing programming experiences.

<u>**Question 22**</u> was another rather difficult question in this subset. This question was based on the number-line sub-construct of a rational-number. It was the 10th most difficult question in this subset, and the 49th most difficult in the whole test of 60 questions (see Appendix B). The children were asked:

22) THIS RULER MEASURES INCHES BY:



a. WHOLES, HALVES, AND FOURTHS.

b. WHOLES AND HALVES ONLY.

c. WHOLES, HALVES, AND THIRDS.

d. WHOLES AND FOURTHS ONLY.

e. NOT GIVEN.

Only 17 out of the 60 questions in the Post-Test required translations of number-line representations. Question 22 was the 14th most difficult among them. In order to answer this question correctly, the children had to identify all the lines between the heavier lines (of number 1 and 2), see if the distances between these lines were equal, and carefully read the options given to them. The diagram below shows how many of the children selected option a. as the correct answer.

| | EXPER | CTR 1 | CTR 2 | STAND. |
|---|---|---|---|---|
| HI | 100% | 72% | 16% | |
| MD | 50% | 32% | 25% | |
| LO | 50% | 20% | 20% | |
| AVE | 66% | 41% | 20% | 23% |

% CORRECT ON QUESTION # 22
TRANSLATING A PICTORIAL NUMBER-LINE REPRESENTATION, INTO WRITTEN WORDS.

Again, we can see how much better the Experimental class as a whole performed than the Control classes. The Experimental children's scores are almost three times as great, in fact, than those of the fourth graders from the RN Project. It is important to note, however, that <u>none</u> of the Experimental children had previously used a number-line representation in their instructional software--in other words, <u>they had not practiced</u> using this particular rational-number sub-construct while involved in the Experimental project. Moreover, <u>all</u> the children in the three math groups solved several number-line problems in their worksheets in the course of their regular math curriculum. These results show that the children's learning from worksheets alone was unsatisfactory.

As in question 50, the low and medium-math levels of the Experimental class did much better on this question than the low and medium-math levels of Control 1, and all the levels of Control 2. The average of their percentage of correct answers for the Experimental low and medium-math levels, taken together, was <u>50%</u>. The average of Control 1's low and medium-math levels, taken together with all the three math groups in Control 2, was only <u>22%</u>.

<u>C. Results from the translations of rational-number **symbolic** representations into **pictorial** representations (eighth level of difficulty, translation number **8** above):</u>

In order to answer question numbers **6, 27, 21, 14, 51,** and **12** in the Post-Test (see Appendix B), the children were required to translate a symbolic representation of a rational number into pictorial representations of rational numbers. The questions were listed above according to their level of difficulty for the children in the RN Project (Lesh et al., 1983, pp. 309-336). The following diagram shows the scores of all 51 children on these questions, as well as those of the fourth-graders in the RN Project.

| | EXPER. | CTR 1 | CTR 2 | STAND |
|---|---|---|---|---|
| HI | 78% | 72% | 55% | |
| MD | 66% | 64% | 54% | |
| LO | 51% | 50% | 45% | |
| AVE | 65% | 58% | 31% | 52% |

% CORRECT FOR QUESTIONS. 6, 27, 21, 14, 51 AND 12. TRANSLATING FROM SYMBOLS INTO A PICTORIAL REPRESENTATION.

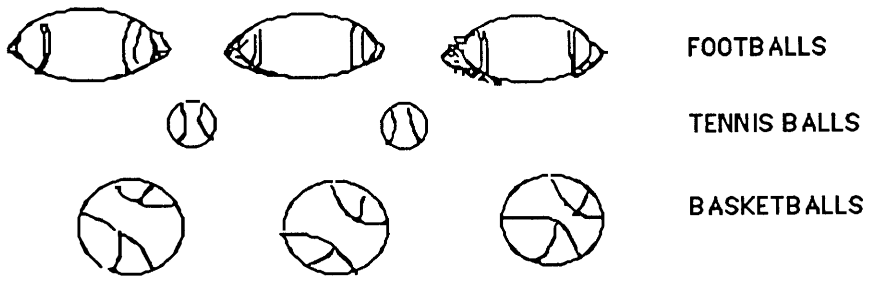
The advantage of the Experimental children over the RN Project fourth-graders and the children from Control class 2 is clear. There are differences between Experimental and Control class 1, but they are not very dramatic in this subset.

## D. Results from the translations of rational-number **pictorial** representations into **symbolic** representations (eighth level of difficulty, translation number **9** above):

In order to answer question numbers **1, 16, 3, 19, 33, 31, 47, 29, 44, 45, 34, 41, 42, 48, 46, 53, 39,** and **43** in the Post-Test (see Appendix B), the children were required to translate a pictorial representation of a rational number into symbolic representations of a rational number. Again, the questions are listed here according to their level of difficulty for the children in the RN Project (Lesh et al., 1983, pp. 309-336).

**Question 42** of this subset required translating a ratio represented by a picture into its symbolic representation. This question reads as follows.

42) WHAT FRACTION OF THE BALLS ARE TENNIS BALLS?



a. 2/8  b. 3/2  c. 2/6  d. 6/2  e. not given

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 children from 4th through 8th grades (Lesh et al., 1983, p.323). It included a discrete object representation in which the represented rational number was less than 1; moreover, parts of this object were not congruent and were visually distracting. The following diagram shows the scores (in their % of correct answers) on this question:

| | EXPER. | CTR 1 | CTR 2 | STAND |
|---|---|---|---|---|
| HI | 100% | 72% | 72% | |
| MD | 72% | 68% | 50% | |
| LO | 50% | 40% | 20% | |
| AVE | 74% | 60% | 47% | 35% |

% CORRECT FOR QUESTION # 42.
TRANSLATING A RATIO REPRESENTED IN A PICTURE, INTO ITS SYMBOLIC REPRESENTATION.

In this complex question none of the high-math Experimental children made any mistakes. The medium-math Experimental children scored like the high-math children in the two Control classes. The Experimental class as a whole did twice as well on this subset than the children in the RN Project, and 14% better or 27% better than Control class 1 and 2 respectively.

## <u>Summary of the Results drawn from the Rational-Number Concepts and their Representations Pencil-&-Paper Test:</u>

Specific attention has been given to the most difficult translation modes between rational-number representations. These were: translating words into pictures (i.e., Level 6 in Lesh et al. 1983); translating pictures into words (i.e., Level 7); translating symbols into pictures (i.e., Level 8); and the most difficult, translating pictures into symbols (I.e., Level 9). These four levels of translations were the most difficult for children of all ages in previous strudies, and were equally difficult for all children in the present study's Pre-Tests. In the Post-Tests however, we have seen how these four translation modes were still relatively difficult for the Control children, but the Experimental children dramatically improved in these particular translations (and even more significantly in other translations that were not reported here). We have examined more deeply the most difficult questions in these subsets; they were: Question number 7 in Level 6; Questions number 50 and 22 in Level 7; and Question number 42 in Level 9. In these difficult questions, the Experimental children consistently scored much higher than the Control children, as well as dramatically better than those from the RN Project.

On the whole, the trends that were found in the results of the Logo Post-Tests were also found in the results of this Post-Test: the Experimental children constantly scored higher than the other two classes; and Control class 1 usually scored higher than Control class 2.

The high-math children from Control class 1 were an exception to the rest of the Control children and to those from the RN Project (Behr et al., Lesh et al, 1983). They were never as good as the high-math Experimental children, and most of the time they were as good as the Experimental medium-math children. Yet their scores are often higher than those of the children from the RN Project, and of the rest of the Control children.

These findings are interesting if, and only if, there exists some kind of correlation between being a part of Project Headlight and having the ability to understand the relationships between different rational-number representations. This is all the more interesting because if the correlation exists, it only does so in the case of the high-math children from Control class 1. It seems as though only the high-math Control 1 children strongly benefited from Project Headlight (see Chapter I, Section 1.2, and 3.2). This is probably due to the fact that their on-going programming projects, which consisted in manipulating graphics and text, contributed to their ability to translate picture representations into written ones, and vice versa. The other children from the same class, however, who were also part of Project Headlight did not score significantly higher than those of Control class 2. This phenomenon requires further investigation. It is an interesting one, since it suggests a correlation between the child's level of understanding and involvement in Logo programming, and his ability to understand and use different representational systems. This could also give an indication of the child's level of readiness. Perhaps the high-math children of Control class 1 were cognitively more developed and therefore had a better understanding of these complex translations between fractional representations--or were better at automatically transferring their knowledge from Logo programming into fractional representation tasks. It is important to note that by the time these Post-Tests were given, all the children from the Experimental class had both a higher level of programming and of experience in creating representations for their instructional software. These two factors combined resulted in the consistently and significantly higher scores across the board. Still, it seems as though the combination of being relatively bright and a good programmer (like the high-math children in Control class 1) might also result in being able to understand representational systems quite well.

### 2.2.2. Results from the Boston's Public Schools Curriculum Referenced Test, Math Level-4

On June 15 and 16 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 of incorrect answers per child on the whole test (i.e., for all 40 questions) was **5.06** incorrect answers per child in the Experimental class; an average of **6.27** incorrect answers per child in Control class 1; and an average of **9.45** incorrect answers per child in Control class 2.

Out of those 40 questions, 6 were specifically on fractions ordering and equivalence; 4 on decimals; 4 on measurement of distance and time that included fractions; and 1 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 of incorrect answers per child on this subset of 15 rational-number questions was: only **1.60** incorrect answers per child in the Experimental class; an average of **3.16** incorrect answers per child in Control class 1; and an average of **4.62** incorrect answers per child in Control class 2.

Several conclusions can be drawn from analyzing these results. The first is that the Experimental children, in general, did much better on the entire conventional school test than the other two Control classes (i.e., the Experimental's average of incorrect answers per child was **5.06, as compared with 6.27, and 9.45** in Control class 1 and 2 respectively).

The second conclusion is related to the children's incorrect answers in the rational-number concepts subset of this test (i.e., the 15 questions, see Appendix B). In the Experimental class, only **29%** of the average incorrect answers per child in the whole test (40 questions) were incorrect answers about rational-number concepts (i.e., an average proportion of 1.60 incorrect per child in fractions, to 5.06 incorrect per child on the whole test). But in both Control classes 1 and 2, approximately **50%** of the incorrect answers per child were, in fact, about rational-number concepts (i.e., in Control class 1 the average proportion was 3.16 to 6.27; in Control class 2 it was 4.62 to 9.45).

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 Control class 1 6.27 - 3.16 = **3.11;** and for Control class 2, 9.45 - 4.62 = **4.83.** It seems however that the differences between the Experimental class and Control class 1 are not significant, but that the differences between these two classes and Control class 2 are. This finding is interesting because it might be that the Project Headlight children's (Experimental and Control class 1) experience with Logo programming contributed to their general mathematical ability.

The results from the school's math-test are shown in the following diagram:

| class: | Average of incorrect ans. per child on all 40 quests. | Average of incorrect ans per child on the subset of 15 rational-number quest. | proportion of inc. ans. in subset/whole |
|---|---|---|---|
| Experimental class | an average of 5.06 incorrect ans. per child | an average of 1.60 incorrect ans. per child | 1.60 to 5.06 = 29% |
| Control class 1 | an average of 6.27 incorrect ans. per child | an average of 3.16 incorrect ans. per child | 3.16 to 6.27 = 51% |
| Control class 2 | an average of 9.45 incorrect ans. per child | an average of 4.62 incorrect ans. per child | 4.62 to 9.45 = 48% |

**Results from the Boston Public Schools Curriculum Referenced Test Math Level-4.**

Let us focus on some specific questions in this test. Questions, **19, 20, 21, 25,** and **31** were about fractions or mixed fractions equivalence and ordering, and they were also the most difficult for many of the children. Many children in all three classes answered these questions incorrectly, but <u>fewer</u> children answered them incorrectly in the Experimental class.

<u>In **Question 19** the children were asked:</u>

19. Which sign makes this number sentence true?

$$\frac{1}{5} ? \frac{1}{3}$$

a) $>$ b) $=$ c) $\ngtr$ d) $<$

In order to answer this question, the children had to have a conceptual understanding of fractions ordering. Behr, Wachsmuth, Post, and Lesh (1984) found that fourth-grade children do not have a strong quantitative notion of fractions for dealing with problems of ordering fractions such as this one, for their whole number concepts get in the way. In this question, for example, most children probably looked at the denominators 5 and 3 and thought that because 5 was bigger than 3, therefore option "a" was the correct answer. The idea that when the two numerators are "1" (as in the case of 1/5 and 1/3), the larger the denominator is the smaller the fraction, is a very very difficult concept for children to understand in ordering fractions, because it conflicts with their concept of ordering whole numbers.

Also, in order to compare two fractions that have nothing in common, such as one third and one fifth, children must reason about two groups at once: the internal relationship between the numerator and the denominator within each fraction, and the external relationship between the two fractions and their denominators and numerators (see also Tierney, 1988). Piaget and Inhelder found that this comparison between two groups or "families" of numbers required performing formal operations, an ability which, according to them, does not usually develop before age twelve or thirteen (the children in my study were 9-10 years old). In the Experimental class, only **3** children answered this question incorrectly; in Control class 1, **9** children, and in Control class 2, **10** children.

<u>In **Question 20** the children were asked:</u>

20. Write as a mixed number: $\frac{13}{7}$

a) 2 b) $\frac{7}{13}$ c) $1\frac{6}{7}$ d) $1\frac{6}{13}$

In order to answer this complex question, the children had to understand both the concept of ordering and equivalence of fractions, and of mixed fractions. A child's ability to link the written symbols with the words representing them could help in this situation as well. In this question, in fact, the children could choose the right answer without actually calculating it, by paying attention to the word "sevenths" as the denominator, then eliminating the wrong options. If a child understood that he had to look for a number that had "sevenths" in it, he would choose option "c" right away. He might try to consider option a, but by using his equivalence concept he would realize that 2 wholes would need

14 sevenths, not 13. The other two options were confusing for the children who did not have a clear concept of ordering, equivalence, or mixed fractions.

In the Experimental class, only **5** children answered this question incorrectly; in Control class 1, **8** children, and in Control class 2, **11** children.

In **Question 21** the children were asked:

21. Which fraction equals $8\frac{2}{3}$

a) $\frac{19}{3}$ b) $\frac{13}{3}$ c) $\frac{48}{3}$ d) $\frac{26}{3}$

This question requires the same conceptual understanding as question 20, but in reverse. However, unlike question 20, the correct answer requires a precise computation. In order to answer this question, the children needed to understand the algorithm of how to translate a number composed of wholes and fractions into a mixed fraction. Option "b" was selected by several children who added 8+3+2 = 13, and therefore selected 13/3 as the correct answer. Option "c" was selected by several others who multiplied 3 * 8 = 24, and then instead of adding 2, they multiplied 24 * 2 = 48, and selected 48/3 as the correct answer. All together, only **6** children answered this question incorrectly in the Experimental class; **9** in Control class 1, and **12** in Control class 2.

In **Question 25** the children were asked:

25. Which of these fractions equals $\frac{1}{2}$ ?

a) $\frac{5}{10}$ b) $\frac{2}{5}$ c) $\frac{2}{3}$ d) $\frac{1}{4}$

This question required a very basic concept of equivalency of halves. Although Smith (1987) found evidence of children's great "ease of movement" within the half-equivalence class, by use of the Halving Operator (ibid, p. 9). **4** children answered this question incorrectly in Control class 1, and **10** in Control class 2. Only one child in the Experimental class did not answer this question correctly; in fact, for an unknown reason, he did not answer it at all.

In **Question 31** the children were asked:

31. Aaron starts his gymnastics class at 3:15. The class lasts $1\frac{1}{2}$ hours. What time is class over?

a) 4:45 b) 4:30 c) 4:00 d) 3:45

This question required that the children first translate the symbol 1/2 into minutes, then add 1 whole hour, and 1/2 of an hour in minutes, to the number 3:15 (i.e., 3:15 + 1 = 4:15, and 4:15 + 0.30 = 4:45). The children's concept of a whole number got in their way when they had to figure out that a whole hour equaled 60 minutes (not 100), and that half an hour therefore equaled 30 minutes. The correct answer, option "a", was not selected only by only **3** children in the Experimental class, **6** children in Control class 1, and **12** in Control class 2.

**The following diagram summarizes the total incorrect answers in each class for these 5 difficult questions:**

| class: | Ques. 19 incorrect per class | Ques. 20 incorrect per class | Ques. 21 incorrect per class | Ques. 25 incorrect per class | Ques. 31 incorrect per class | Total of incorrect answers per class |
|---|---|---|---|---|---|---|
| Experimental class | 3 children | 5 children | 6 children | 1 child | 3 children | 18 incorrect ans. an average of 1.05 per child |
| Control class 1 | 9 children | 8 children | 9 children | 4 children | 6 children | 36 incorrect ans. an average of 2.01 per child |
| Control class 2 | 10 children | 11 children | 12 children | 10 children | 12 children | 56 incorrect ans. an average of 3.50 per child |

## Summary of the Results from the Curriculum Referenced Math Test:

In this section we have seen how much better the Experimental children did on this whole conventional school test than those from the other two Control classes (i.e., the Experimental's average of incorrect answers per child was **5.06,** as compared with **6.27,** and **9.45** in Control class 1 and 2 respectively). Moreover, in the Experimental class, only

**29%** of the average incorrect answers per child in the whole test (40 questions) were incorrect answers about rational-number concepts (i.e., an average proportion of 1.60 incorrect per child in fractions, to 5.06 incorrect per child on the whole test). But in both Control classes 1 and 2, approximately **50%** of the incorrect answers per child were in fact about rational-number concepts (i.e.,in Control class 1 the average proportion was 3.16 to 6.27; in Control class 2 it was 4.62 to 9.45). Furthermore, in giving specific attention to the subset of the five most difficult questions (questions number 19, 20, 21, 25 and 31) we have seen even more dramatic differences between the Experimental and the Control children: a total of only **18** incorrect answers in the Experimental class; but a total of **36** in Control class 1; and **56** in Control class 2.

On the whole, these results are very important to this study, because the Experimental children did not spend more time than the others in being directly instructed on how to perform such algorithms as those needed for solving Questions 20 and 21. They had not been given more formal instruction than the other children in the Control classes on fractions or mixed fractions ordering and equivalence, or the algorithms involved. But they did do better on this test, and especially on the subset of the most difficult questions on fractions. My reasoning for these results, as well as my discussion of the results of the other Post-Tests, will be presented in Chapter IV of this Thesis.

## 2.3. DEBBIE'S RESULTS FROM THE EVALUATION

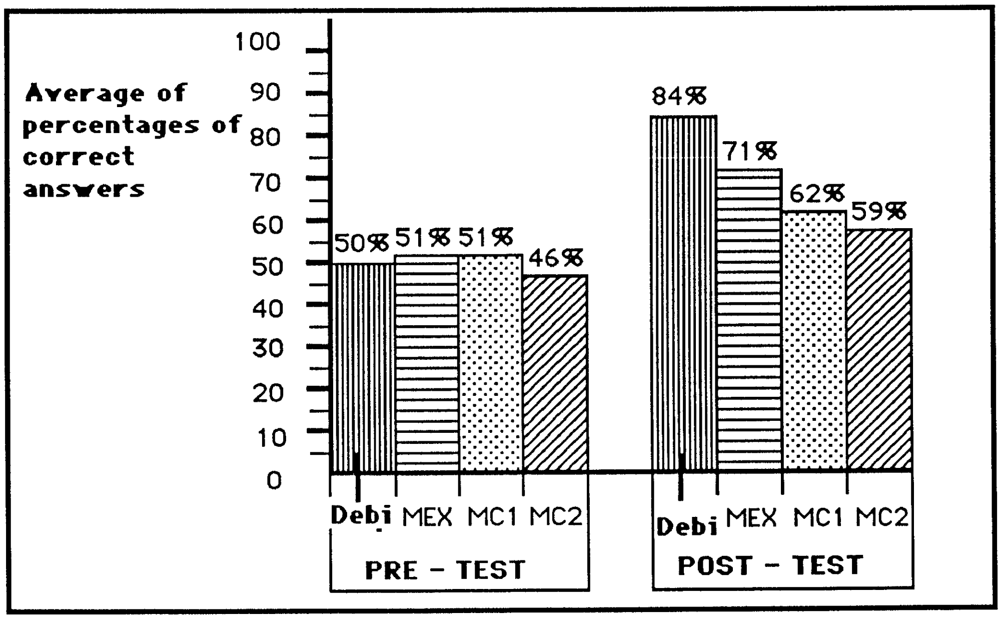
In this section I shall briefly summarize some of Debbie's scores from the Post-Tests. A detailed description of these tests, their objectives, purposes, and content was provided to the reader in the previous sections of Chapter III as well as in Chapter I of this thesis. For this reason, the results will be presented without task analyses or deep interpretations. The purpose of this section is to highlight Debbie's exceptional progress between January (Pre-Tests) and June (Post-Tests), and to compare her progress with that of the other children. In subsection 2.3.1., I compare Debbie's scores on the Rational-Number Pencil-&-Paper Test with those of her peers; in subsection 2.3.2., I do the same with her scores on the School's Math Test; in subsection 2.3.3., with her answers in the written Logo Test; and finally, in subsection 2.3.4., with her accomplishment on the Logo Computer Tasks. The results from Debbie's Pre- and Post- "What Is A Fraction?" Interview are not provided here; many of them were described in Debbie's Case (Chapter II). A further analysis of Debbie's Interviews, and the comparison between her Interviews and those of others will be provided in a forthcoming paper.

### 2.3.1. The Results from Debbie's Rational-Number Test

Debbie's score on the Fractions **Pre-Test** on translations among representations was **50%** correct. This was a typical score for this Pre-Test among the tested fourth-graders from the medium-math group (i.e., Debbie's math-group); the Experimental medium-math group's average percentage of correct answers on this Pre-Test was 51%, 51% for the medium-math children from Control class 1, and 46% for the medium-math children from Control class 2. Most of the children in this study, as well as the children from the RN Project (Behr et al. 1983; Lesh et al. 1983) had many difficulties with the questions that required translations from pictures to symbols, symbols to pictures, and pictures to words. Debbie's Post-Test scores show that she overcame many of these difficulties. Debbie's score on this Rational-Number Post-Test was **84%** correct. This was a very high score compared with those of the other children in her math group, and even compared with most children of the high-math group. The average **Post-Test** score of the Experimental's medium-math group was 71% correct; the average score of Control group 1was 62%, with 59% correct for Control class 2. Debbie's Post-Test score was not only much higher than those of her medium-math peers, but also higher than the average score of the children in the high-math group. In the Post-Test, the Experimental high-math children's average score was 82%, the high-math of Control class 1 scored an average of 78% correct, and the high-math children from Control class 2, an average of 55% correct.

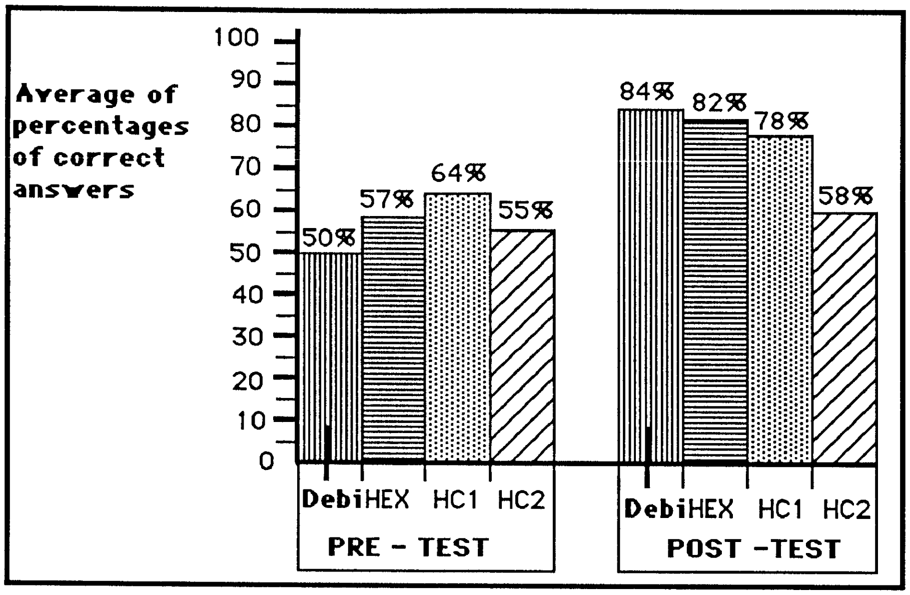
The diagrams shown below compares Debbie's Pre- and Post-Test scores with those of her fifteen peers from her medium-math group (i.e., **MEX**=Medium-math EXperimental children; **MC1** = Medium-math children from Control class 1; and **MC2** = Medium-math children from Control class 2).

**Debbie's Pre- and Post-Scores on the Rational-Number Pencil-&Paper Test, and how they compare, Pre- and Post-, with those of her peers from the MEDIUM-MATH GROUP:**



The following diagram compares Debbie's Pre- and Post-Test scores with those of the 21 children from the high-math group (i.e., **HEX** = High-math EXperimental children; **HC1** = High-math children from Control class 1; and **HC2** = High-math children from Control class 2).

**Debbie's Pre- and Post-Scores from the Rational-Number Pencil-&-Paper Test, and how they compare with those of the children from the HIGH-MATH GROUP:**



These two diagrams demonstrate Debbie's exceptional progress in the period of time between the Pre- and the Post-Tests. Before the Project started, Debbie was on the average level of the medium-math children (50%); however, after the Project ended, Debbie's answers were far better than those of the children in the medium-math group. The average of percentages of correct answers for the <u>whole medium-math group taken together</u> (i.e., the children from experimental, Control 1, and Control 2 classes taken together), was <u>64%</u>, while Debbie's was <u>84%</u> correct.

Moreover, the comparison between Debbie's Pre-Test score and those of the <u>high-math</u> children, shows that before the Project started, Debbie scored quite a bit lower than the high-math children from the three classes (50% for Debbie as opposed to 60% for all the high-math children taken together); in the Post-Test, however, we saw that she progressed much more than many other children of the high-math group. She scored **84%** correct, whereas the high-math <u>Experimental</u> children scored an average of **82%**, and the <u>three class's high-math children, taken together</u>, an average of **71%** correct.

### 2.3.2. The Results from Debbie's Curriculum Referenced Test, Math Level-4

Debbie answered only 2 questions incorrectly out of the 40 given to the children in this test. One of her wrong answers was on question 21 (see Section 2.2.2 in this Chapter).

The following diagram compares Debbie's answers on the most difficult subset of questions (about fractions) in this test.

| class: | Ques. 19 incorrect per class | Ques. 20 incorrect per class | Ques. 21 incorrect per class | Ques. 25 incorrect per class | Ques. 31 incorrect per class |
|---|---|---|---|---|---|
| DEBBIE | (+) | (+) | (-) | (+) | (+) |
| Experimental class | 3 children | 5 children | 6 ● children | 1 child | 3 children |
| Control class 1 | 9 children | 8 children | 9 children | 4 children | 6 children |
| Control class 2 | 10 children | 11 children | 12 children | 10 children | 12 children |

Debbie was one of the six children in the Experimental class who answered question 21 incorrectly. However, the other questions that posed great difficulty to many children from the medium-math group, were not difficult for Debbie, and she answered them correctly.

### 2.3.3. The Results from Debbie's Logo Pencil-&-Paper Test

In **Question 1** of this test, the children were asked to list all the Logo instructions and commands they knew, to define them, and to give examples of how they were used in Logo programming. In the **Pre-Test,** Debbie listed only **9** Logo commands and did not define exactly what they did, whereas in the **Post-Test** she listed **29**. In the Post-Test, Debbie correctly abbreviated all the 29 commands she listed. Also, out of these 29 commands, she gave good examples for 27 of them, and for 26 of them she gave excellent and accurate descriptions for what they did and how they should be used in Logo programming. In short, for her math-level, she performed exteremely well on this task.

The diagram shown below compares Debbie with the high-, low-, and medium-math children from the three classes, on the average number of commands listed by each child in the Post-Test.

**The number of Logo commands and instructions that were listed by the children in the Post-Test:**

| | EXPER. | CTR 1 | CTR 2 | **DEBBIE** |
|---|---|---|---|---|
| HI | 32 | 16 | 8 | |
| MD | 24 | 9 | 8 | **29** |
| LO | 21 | 11 | 7 | |

In **Question 2,** the children were asked to classify their lists of Logo commands and instructions. In the **Pre-Test**, Debbie <u>did not list any group</u> (her page in this Pre-Test shows that she listed one arbitrary group of "Logo things" but erased it and wrote "I don't know" next to it). In the **Post-Test**, Debbie created only 2 groups; however, both of her groups were mutually exclusive, and of the **abstract** kind; whereas no one in Control class 2 (among the medium-math children) created abstract groups, and only two children in Control class 1 did so (each of these two children created one abstract and one loose group). The two groups that Debbie created in the Post-Test were the following:

<u>Groups number 1</u>. Title: "Things that Erase."
Members: " PE, DEL, BACKSPACE."

<u>Group number 2.</u> Title: "Things that go to another page (or connect pages together)."
Members: "GETTOOLS, GETPAGE, ESC (goes to content)."

**The following diagram summarizes the results on Debbie's ability in this non-concrete classification task, and how she compares with her peers from the medium-math group.**

| | EXPERIMEN MEDIUM | CONTROL 1 MEDIUM | CONTROL 2 MEDIUM |
|---|---|---|---|
| NO. OF GRPS PER CHILD | 5, 5, 4, 3, 2 Debbie 2 grp. | 3, 2, 1, 1, 0, 0 | 4, 1, 1, 0, 0 |
| Abstract vs "loose" | Debbie 2 Abs. groups. | 2 children each created 1 Abst. | No abst. grps in all childrn. |

In **Question 3,** the children were asked to execute a given linear Logo code given to them on paper. The following diagram shows that Debbie (Subject number 5 in the Exp. Med.) executed this code accurately, found the three squares, and used a consistent unit and correct scaling (4 : 2 : 1). Her page from this written page indicates that she tried it three times and did not give up after she had made mistakes the first time; she checked and re-checked her drawing until she was sure she had found the right answer, which she finally did.

| | Number of Squares in final picture | | | | | | Number of Trials of drawing the picture | | | | | | Correct Scale & Consistent Unit | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Subject | S1 | S2 | S3 | S4 | S5 | S6 | S1 | S2 | S3 | S4 | S5 | S6 | S1 | S2 | S3 | S4 | S5 | S6 |
| Exp. Med. | 3 | 3 | 3 | 3 | 3 | x | 2 | 1 | 1 | 3 | 3 | x | y | y | y | y | y | x |
| Cot1 Med | 0 | 3 | 2 | 1 | 3 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | n | n | n | n | y | n |

In **Question 4,** the children were asked to simplify (or optimize) the code given to them in Question number 3. All the medium-math children from the Experimental class, and Debbie among them, managed to reach Level 3, which was a high level of modularization (i.e., using procedures, sub-procedures, a variable for the square's size, and REPEATs). The following diagram shows that not one of the medium-math children from Control classes 1 or 2 reached that level of sophistication.

**The number of children, in each class and math group, that reached Level 3 of optimization in Question number 4:**

| | Simplify Level 3 | | |
|---|---|---|---|
| | Exp. | Cot.1 | Cot.2 |
| Hi | 7/8 | 1/7 | 0/6 |
| Med | 5/5 | 0/6 | 0/5 |
| Lo | 1/4 | 0/5 | 0/5 |

In **Question 5,** the children were asked how many inputs certain basic Logo commands took (i.e., FD, BK, LT, SETPOS, REPEAT, HOME, CG, and SETC). In the

**Pre-Test**, Debbie did not have a clear concept of inputs, what they meant, or how many each of the above Logo commands took. She answered **5 sub-questions incorrectly** in the Pre-Test: she wrote that FD took "A lot of inputs," did the same for BK, wrote that REPEAT took "1 input," and gave as an example "Repeat 10[etc.]"; she wrote that LT took "2 inputs," and gave an example "LT 45," than answered "I don't know" beside the sub-question about SETPOSition. At the bottom of the page in the Pre-Test (under Question 5) she wrote: "I can't answer it. I'm not so sure!" Her answers were typical; most children, from all the three classes and math groups, were very confused about this question in the Pre-Test, and many of them wrote "I don't know."

In the **Post-Test**, however, Debbie answered **all** of these questions correctly, and scored **100%**. The following diagram compares Debbie's success in answering this question with that of the medium- and high-math children from the three classes. Debbie scored much higher than her peers from the medium-math group, was at the level of most of the Experimental high-math children, and definitely superior to all the Control children from the medium- and high-math groups.

**Results from the Post-Test: Question number 5 about Inputs**

| | Med-math | Hi-math | **DEBBIE** |
|---|---|---|---|
| Experimen. | 67.5% | 94.5% | **100%** |
| Control 1 | 37.7% | 78.3% | |
| Control 2 | 7.5% | 30.7% | |

* In **Question 6,** the children were asked to transform three simple Logo procedures: to transform a given procedure for a circle into one for a larger circle; to change a given circle to half-circle, and a square to a rectangle. This question was not given to the children in the Pre-Test. In the Post-Test Debbie was able to create all the 3 required transformations. None of the Control children from the medium-math group were able to transform a circle into half-circle or a square into a rectangle. The diagram below summarizes these results

**The three black dots represents Debbie's success in the three transformations:**

| | Circ to bigger circ | | | Circ to half-circ | | | Square to rectangle | | |
|---|---|---|---|---|---|---|---|---|---|
| | Exp. | Cot.1 | Cot.2 | Exp. | Cot.1 | Cot.2 | Exp. | Cot.1 | Cot.2 |
| Hi | 8/8 | 5/7 | 0/6 | 7/8 | 5/7 | 0/6 | 7/8 | 2/7 | 0/6 |
| Med | ● 3/5 | 2/6 | 0/5 | ● 4/5 | 0/6 | 0/5 | ● 4/5 | 0/6 | 0/5 |

### 2.3.4. Results from Debbie's Logo Computer Test

Debbie accomplished the Logo computer task in both the Pre- and the Post-Tests. In both cases she scored higher than many of her peers from the medium-math group. However, the Post-Test was more difficult than the Pre-Test, and many children from the medium-math were not able to perform all the debugging task or the optimization requirements. In the Post -Test Debbie, like all the members of her class, found all the bugs rather quickly and optimized the procedure according to all the requirements. No one from Control class 2, and only 2 children from the medium-math group of Control class 1 were able to locate all the bugs on paper and in the computer program, and to fix the program. No medium-math child in Control classes 1 or 2 reached Level 3 of optimization of the given code on the computer, and no medium-math Control children knew how to change the colors (sub-task number 5) or how to use variables in this context.

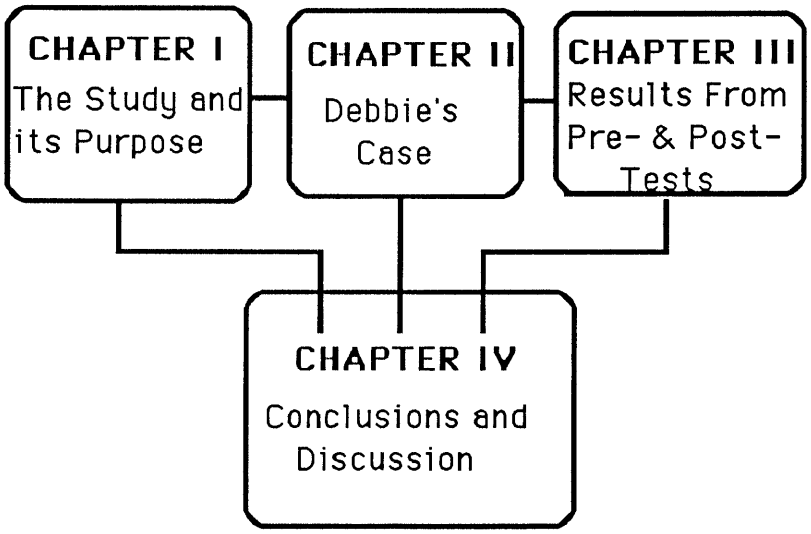
### 2.3.5. Summary of Debbie's Results from the Evaluation

The preceding results demonstrate Debbie's exceptional progress in learning fractions and Logo through the Instructional Software Design Project. The Pre-Test scores indicate that Debbie's abilities in Logo and Fractions before the Project began were average or low (before the Project started she was one of the lowest pupils in her medium-math group); however, her scores at the end of the year after the Project ended (in the Post-Tests), were of a high-math child, often at the top 10 percentile of all the 51 tested children. In the Post-Tests, as we have just seen, she constently scored higher than all the medium or low-level children in the Experimental and Control classes. These finding are particularly interesting because, before the Project started, Debbie had been far less proficient in her math than in her writing skills. After the Project ended, Debbie continued to progress in her acquisition of high math skills as well. Such findings must be considered in relation to the many aspects of Debbie's learning, thinking, and development during the Project, and which have been described in detail in chapter II (see in particular the Introduction to the Case, end of Section 1).

# SOFTWARE DESIGN FOR LEARNING

# CHAPTER IV:

# CONCLUSIONS AND DISCUSSION

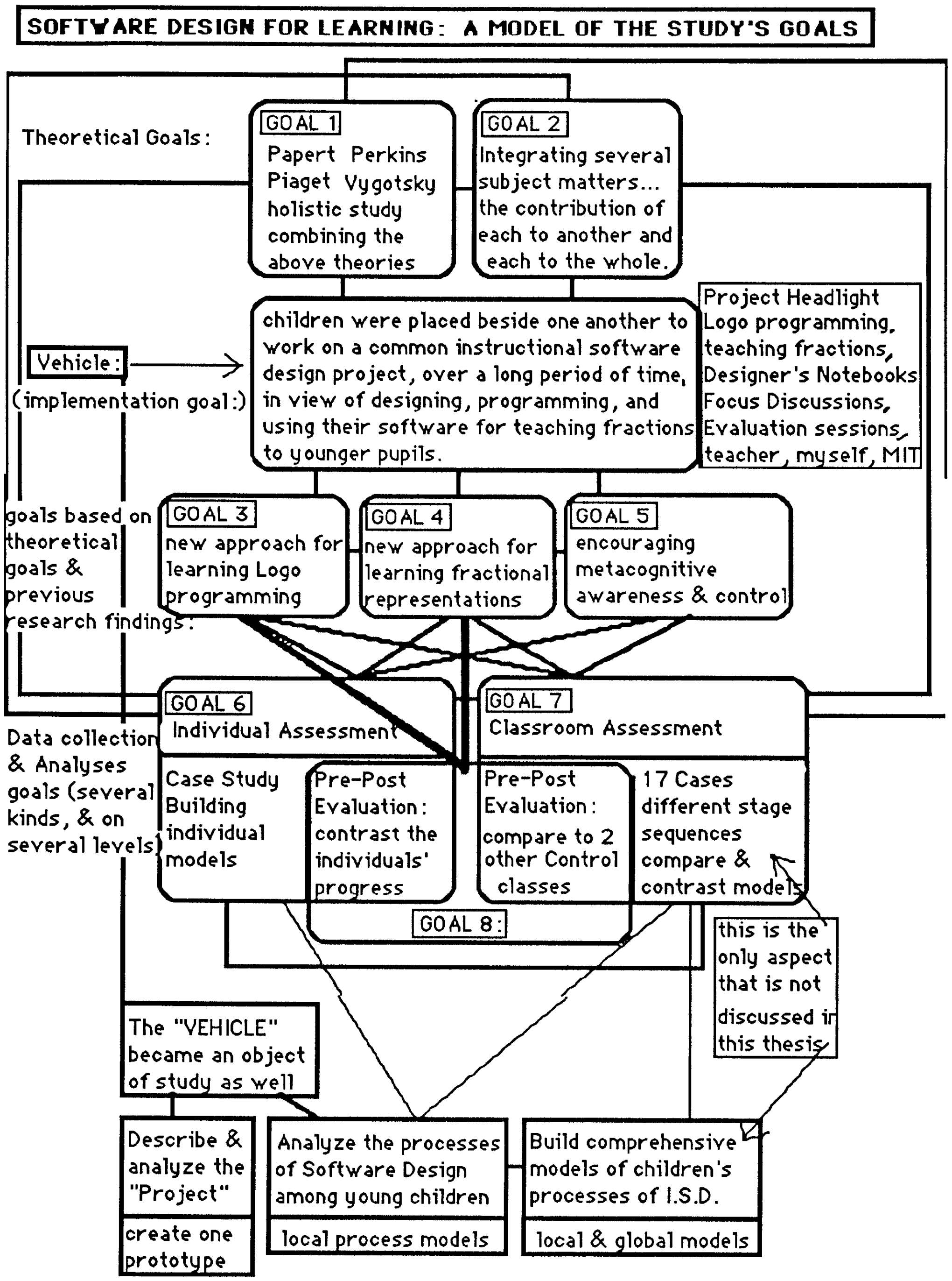

CHAPTER I
The Study and its Purpose
CHAPTER II
Debbie's Case
CHAPTER III
Results From Pre- & Post-Tests
CHAPTER IV
Conclusions and Discussion

## Introduction: Review of the study's goals and how these goals were accomplished in this thesis, and the structure of this chapter.

The overall goal of this study was to implement, assess, and describe an experiment called the Instructional Software Design Project. In the first chapter of this thesis, I analyzed the new conditions for learning fractions, Logo programming, metacognitive skills, and software design which were explored in this Project, and indicated that the in-depth studying of children's minds in this context had several goals. The **two general theoretical goals** were: **1)** To construct a holistic picture--using Papert's theories (e.g., *Mindstorms*, 1980; *Constructionism*, 1986) and Perkins' theories (e.g., *Knowledge As Design,* 1986) as well as Vygotskian and Piagetian perspectives--of what it meant for a child to learn and be involved in an open-ended, complex, and unusually long problem-solving process related to the learning of mathematics, designing, and computer programming. To use instructional software design as a vehicle for analyzing a child's total learning in an "extreme" environment and over a long period of time; analyzing an activity that involved several theoretical perspectives and many variables (many of which could be considered ill-defined variables); and analyzing how and what was learned by the children through their complex solution processes. **2)** To experiment with this one prototypical activity ("The Project") of children learning fractions and Logo through instructional software design, to understand how children can learn several topics or subject matters at the same time, and how the learning of one can contribute to the other; furthermore, to assess what major changes this activity can offer to educational practice, and to judge whether or not it reveals any information that might lead to changes in the teaching and learning approaches of fractions and Logo in elementary schools.

In addition to these two general goals my study had **three specific objectives based on the above theoretical goals and on previous research in the relevant fields.** These were: **3)** To investigate the ways in which this Project enhanced children's learning and understanding of basic concepts of Logo programming, such as procedures or modularity, variables or inputs, repeat or recursion, conditionals, and other Logo commands and operations. I aimed at investigating the children's ability to learn Logo through the Project, their ability to generate, debug, manipulate, and modify codes, written by themselves or by other people. But, as goals 1 and 2 clearly show, I was not interested in investigating the learning of Logo for the sake of learning Logo, as an isolated piece of knowledge; neither was I interested in a microcausal type of investigation of what method of explicit instruction would result in the transfer of specific Logo skills into other specific domains. Instead, I used Logo as a tool for reformulating and manipulating other kinds of knowledge (e.g., instructional design, software production, or rational-number representations); I assessed the learning of Logo as one aspect in the child's whole learning situation; and investigated whether or not children who learned programming by designing and programming instructional software for others on fractions would become better programmers than those who, meanwhile, were learning programming in more limited situations (such as the ones reported by the literature). **4)** To assess the children's knowledge of fractions. To focus on children's understanding of the rational-number representational system, and their ability to translate and combine representations. By giving specific attention to rational-number representations, I investigated whether or not the Experimental children's designing and programming representations for fractions, and their creating graphical displays for equivalence and operations of rational numbers, influenced their thinking and learning about representations of fractions and the relationships between them; and in what ways the children's representing and explaining fractions to another pupil helped them to overcome their own difficulties in understanding fractional representations, which had been reported in other studies. **5)** To explore the ways the Project enhanced children's cognitive awareness (i.e., children's thinking about their own thinking), their cognitive control (i.e., planning, reflection, self-management, and thinking about these cognitive processes), and their meta-conceptual thinking (i.e., children's thinking about their own knowledge and understanding of concepts).

With these 5 goals in mind, I had **three more goals relating to the collecting and analyzing of data in this context,** which consisted in conducting several kinds of investigations. These were: **6)** To gather data on the day-by-day microgenetic process of an individual creating an interactive teaching device, more specifically, investigating goals 1, 2, 3, 4, and 5 on an individual level. **7)** To gather data on the process of the whole class participating in this Project, more specifically, investigating goals 1, 2, 3, 4, and 5 on a classroom or "cultural" level. **8)** To conduct an Evaluation with the Experimental class and two other Control classes, and to contrast the achievements and knowledge of the children who participated in the Project with those of other children who did not. The results of the Evaluation were needed for investigating the progress of each individual child and the Experimental class as a whole. The diagram on the following page summarizes these goals and their inter-relationships.


SOFTWARE DESIGN FOR LEARNING: A MODEL OF THE STUDY'S GOALS
Theoretical Goals:
GOAL 1
Papert Perkins Piaget Vygotsky holistic study combining the above theories
GOAL 2
Integrating several subject matters... the contribution of each to another and each to the whole.
Vehicle:
(implementation goal:)
children were placed beside one another to work on a common instructional software design project, over a long period of time, in view of designing, programming, and using their software for teaching fractions to younger pupils.
Project Headlight Logo programming, teaching fractions, Designer's Notebooks Focus Discussions, Evaluation sessions, teacher, myself, MIT
goals based on theoretical goals & previous research findings:
GOAL 3
new approach for learning Logo programming
GOAL 4
new approach for learning fractional representations
GOAL 5
encouraging metacognitive awareness & control
Data collection & Analyses goals (several kinds, & on several levels)
GOAL 6
Individual Assessment
Case Study Building individual models
Pre-Post Evaluation: contrast the individuals' progress
GOAL 7
Classroom Assessment
Pre-Post Evaluation: compare to 2 other Control classes
17 Cases different stage sequences compare & contrast models
GOAL 8:
this is the only aspect that is not discussed in this thesis
The "VEHICLE" became an object of study as well
Describe & analyze the "Project"
create one prototype
Analyze the processes of Software Design among young children
local process models
Build comprehensive models of children's processes of I.S.D.
local & global models

In the second chapter of this thesis (Debbie's Case), I accomplished many of these goals by analyzing Debbie's seventy-hour-long problem-solving enterprise, and describing the day-by-day learning and cognitive processes involved as Debbie and her peers interlaced their learning of fractions and Logo programming through planning, designing, programming, and evaluating their own pieces of instructional software in view of using it to teach pupils from a lower grade. In the third chapter, I showed the ways in which the children involved in the Project became superior in their mathematical and programming knowledge and expertise to two classes who learned fractions and Logo by methods other than instructional software design. The Case Study and the results from the Evaluation, taken together, demonstrated that fourth-grade children, placed beside one another to work on a common software-design project over a long period of time, were strongly motivated and fully in charge of their own learning, established sequences of learning and development that were both an interesting mixture of Piagetian and Vygotskian processes and a combination of Papert's and Perkins' theories of knowledge construction and learning. The Project resulted in different children's establishing different stage sequences in organizing and implementing their long-term projects. The use of Logo to design instructional software was an unusually effective way for learning Logo programming. The learning of fractions through the use of Logo, through teaching and explaining, and through designing instructional software on fractions, involved children in creating and translating multiple representations for fractions, and enhanced their mathematical understanding of what fractions are. The children's ongoing work during the Project also seemed to enhance their metacognitive awareness and cognitive control strategies.

In the following sections, 1 through 6, I shall discuss and highlight the ways in which the thesis succeeded or did not succeed with respect to its major purposes and goals. Several questions for further research are discussed in this context--for example, what microgenetic phenomena could be further investigated in such an extreme learning environment, what are the necessary conditions under which this Project could be implemented, or what other subjects could be learned by elementary-school children through the method of instructional software design. In the seventh section of this chapter, basing myself on my research findings, I discuss what design tools or computer technologies could be further developed and used in this learning process. I shall outline my preliminary ideas for an integrative Software-Design-and-Production Tool for children's learning. Finally I summarize the conclusions of this thesis.

**I. A group of fourth-grade children, working side by side in front of computers, over a four-month period, were involved in a software design project in view of designing and using their individual software for teaching fractions to younger children. They were found to be strongly motivated and fully in charge of their own learning; seemed to establish sequences of learning and development that were an interesting mixture of Piagetian and Vygotskian processes; and also bore out Papert's and Perkins' theories of knowledge construction and learning. Furthermore, the children treated their knowledge in a functional manner, designed and built meaningful products, and established deep relationships with the objects involved.**

The first purpose of this thesis was to construct a picture of what it meant to a fourth-grade child to learn and be involved in an open-ended, complex, and unusually long problem-solving process related to designing, the learning of mathematics, and computer programming. The Instructional Software Design Project proved to be a useful and informative vehicle for constructing this picture. Chapter III provided us with several straightforward results on what the Project meant for children: becoming better programmers, growing better at understanding rational-number concepts and their representations, and being successful in the school-tests. Chapter II provided us with a very detailed analysis of one Experimental girl, Debbie, and what were the results of her being involved in this Project. Through the analysis of Debbie's Case we examined many of the reasons that led to the results presented in Chapter III. We saw that, in fact, the Project meant much more to the children than being successful in tests, acquiring more information of various kinds, or understanding the related topics better. To them it meant being in charge of their own learning and enjoying their learning; it also meant being productive, creative, and self-motivated. No one instructed them on what to do or how to do it. For four months, almost every single day, they worked very hard on their products and invested a great deal of time and thought in their planning, re-planning, designing, implementing, programming, and revising of an entity that was never even graded, or evaluated in a conventional way. The major feedback they received all along was their own enjoyment and satisfaction during the various phases of production, seeing their product grow and grow, getting to program more complex and beautiful screens, putting their products (and therefore their knowledge) into use, and seeing the pieces of software appreciated and enjoyed by their best friends, by third graders, and by other people in their "culture." On the whole, this Project meant a great deal to these children, and the best evidence for this was their not wanting to cease working on it.

One aspect that was highlighted in this context is what the Project meant for the children's self-esteem and their attitude towards school learning in general. Debbie's personality, for example, and her particular attitude towards school learning and socializing, have been discussed in detail in Chapter II. Debbie was a model case for an average inner-city public-school fourth-grader: a young black girl, very reserved and somewhat unhappy, who seemed wary about trying anything new. She was not academically or socially successful, but through the Project she learned to be independent and was "discovered" by her peers and teachers and quite often received positive responses about her work from them. She grew to feel better about herself, opened up, and interacted more freely with the people around her.

For these children, the Project also meant breaking away from simplistic thinking and rigidity, becoming aware of several aspects of a situation simultaneously, and being able to perceive and generate complexities and to think more abstractly about fractions, Logo, and design. Debbie was an interesting case in this context, especially in the light of Piaget's theories on constructivism and on the underlying patterns of thought during the different stages of cognitive development. In her processes of thinking, designing, programming, and learning throughout the Project, Debbie moved according to Piaget's progression scheme overall (although only local aspects of the Piagetian progression were studied here); furthermore, within each step or stage of her software design we saw a recapitulation of the Piagetian developmental stages. In her thinking and actions during the Project, Debbie moved back and forth between being attentive to limited and static amounts of information and considering several aspects of a situation simultaneously; she moved from concrete pre-operational thought to more formal thought; from rigid thinking that focused, for example, on one dimension of a programming problem, to more fluent and dynamic thinking related to several dimensions of her computer programs; and though she began by constructing very simple plans, designs, and screens, she was later able to create more complex ones, moving back and forth between simplicity and complexity. In general, Debbie, like her other classmates, learned to free herself from rigidity and acquired more flexibility within the Project's length and breadth (across the board)--in her translating and combining representations of fractions, in her Logo programming techniques, in her planning and reflecting in the Designer's Notebook, and in her attitude towards the Project as a whole.

Both Vygotsky's and Piaget's views on learning and intelligence were emphasized and combined in this study. Debbie's learning, for example, was interpreted as a clear combination of both perspectives. In several situations she was shown to be a strong case for constructivist learning (e.g., the House Scene, or her writing of several super-procedures for the same purpose), and at the same time for social-cultural learning (e.g., the design of her Introduction Screen, many of her instructional statements, the Sesame Street screen, or her use of the Wait.For.User Logo procedure in her program). By studying Debbie's learning and cognitive development as a total activity, we realized that some aspects of her learning were influenced by the "scaffolding" of a guiding adult (e.g., imitating the teacher's approach in her screen about Thirds), a helpful peer (e.g., learning how to create a triangle in Logo), or a probing researcher (e.g., understanding that the "empty" unshaded parts of divided shapes were also fractions). Those situations were interpreted according to Vygotsky's perspective:

> "Every function in the child's cultural development appears twice, on two levels. First on the social, and later on the psychological level; first *between* people [as an] *intrapsychological* category, and then *inside* the child, [as an] *interpsychological* category" (Vygotsky, *Mind In Society,* 1978, p. 57).

Other aspects of her learning were interpreted as a result of her own interaction with the specific learning tools and of her own thoughts, and resulted in various spontaneous inventions (e.g., rhyming procedure names, or attaching symbols to a real-life fractional representation) and spontaneous constructions (e.g., discovering that she could connect procedures in a super-procedure only after she had spent a whole month constructing five separate ones). These situations were interpreted through Piaget's perspective of how a child must invent and re-invent the basic concepts and logical-thought forms that constitute his intelligence, then develops cognitively through being involved in spontaneous constructions and inventions of ideas for himself:

> "The development of intelligence itself--what the child learns by himself, what none can teach him and he must discover alone..it is precisely this spontaneous development which forms the

obvious and necessary conditions for the school development" (Piaget, *The Child and Reality*, 1973, pp. 2-4).

These two different aspects of learning (i.e., imitating as opposed to inventing, or being guided by an adult as opposed to being guided by her own ideas), created an interesting learning pattern that repeated itself in Debbie's work, as well as in those of the other Experimental children, throughout the Project.

Papert's perspective on the important role of self-constructed, interactive objects in children's learning, thinking, and socializing was emphasized throughout. In Debbie's case, for example, the role of her software construction in Logo was crucial to her change of attitude towards learning, her growing to feel good about herself, her developing expertise in Logo, and her better understanding of fractional representations.

Perkins' ideas of treating knowledge as design, or knowledge as an object whose meaning children construct through design, were emphasized throughout, and were integral to this thesis. The processes of children's producing complex products, learning about large meaningful chunks of information through building instructional systems, identifying the purposes and structure of each component in that system, and creating inter-relationships between components--processes that rarely exist in conventional education--had a crucial role in children's establishing strong relational networks between concepts, and making their pieces of knowledge highly connected to each other. We examined how Debbie, for example, connected and related her knowledge of Logo to her knowledge of fractions, and how, during May and June, she also updated her "old" Logo knowledge with the "new" (updating and revising the existing parts of her program and connecting these to her new parts). Knowledge was manipulated and treated as functional, and as Perkins believes:

"Knowledge should be seen and treated as functional, as something that gets put to use much as a screwdriver or a hammer. To do justice to knowledge and to the learner, we have to keep the learner doing something with the knowledge gained...Knowledge as design holds that all knowledge has a tool-like character...some knowledge is explicitly tool-like, designed to manipulate other knowledge and facilitate thinking and problem solving activities of various sorts" (Perkins, *Knowledge As Design*, 1986, pp. 215-216).

We examined the important role of language (natural language as well as the Logo language) in the children's processes of learning. We saw this in Debbie's writing and planning, in the way she sometimes "talked to" the Designer's Notebook (e.g., after she wrote her plans and drew her designs, she often seemed to be addressing her Notebook directly: "I hope this is clear. Bye now, and see you tomorrow!"); or in her conversations with herself as she typed on the computer keyboard while talking to the computer (e.g., "Why did you do it to me?!, Ah...I know there's a bracket missing there..."), or in her explanations and comments to others (e.g., "All these shapes are one half!" or "I am trying to say that you can use fractions almost every day of your life!" or, "Naomi, do you think these colors are o.k.?"). She developed cognitively during the Project also because she was involved in thinking about teaching and explaining, and because she was thinking about communicating knowledge to other children through the use of language and via her software. The path from the objects involved (i.e., the computer and the software or the Notebook) to Debbie, and from Debbie to these objects, was one of language, either natural speech or Logo. Language was used intensively in this context: in children's teaching and explaining about fractions, in their planning and reflecting in writing, in their discussion of designs and screens with their peers, in their talking about Logo, and so on. This intensive use of language facilitated in many ways their gaining control over their actions and processes. As Vygotsky believes, speech acts as an organizer, unifier, and integrator of many disparate aspects of children's behavior, such as perception, memory, and problem solving. Words provide learners with ways to become more efficient in their adaptive and problem-solving efforts (*Mind In Society*, 1978, Chapter 4).

In Debbie's Case we also examined some of the complex relations between the 17 minds involved in the same job: how the children worked together, influenced each other or sometimes resisted influence, the relationships between the internal and external knowledge of each child, but also between all the Experimental children and the other people, objects, products, and knowledge forming their "culture." The culture created within the Experimental class as a whole (i.e., the environment of each child at the time he was working on his Instructional Software Design Project) enhanced interactions, connections, and reciprocal relations involving many of the following: objects such as computers, computer tools, Designer's Notebooks; people such as other children, the teacher, the researcher (myself), and members of the MIT staff--all of whom walked around the computer-area, talked together, helped each other, expressed their feelings on various subjects and issues, or worked on different programming projects; and finally, knowledge of fractions, Logo, and design as communicated by those involved, or knowledge inherent in other people's projects, which the children could observe simply by walking around and looking at the various computer screens or the different plans and designs in the Designer's Notebooks.

Finally, the Experimental children's exceptional progress in learning fractions and Logo through the Project was emphasized in Chapter III. Debbie's Pre-Test scores, for example, indicated that her abilities in Logo and Fractions before the Project began were average or low (she was a member of the medium-math group, and was one of the low pupils in that group); however, her scores (in the Post-Tests) at the end of the year, after the Project ended, were of a high-math child, often at the top 10 percentile of all the 51 tested children. In the Post-Tests she consistently scored higher than all the medium or low-level children in the Experimental and Control classes. These findings are particularly interesting because, before the Project started, the teacher had described Debbie as someone who was low in math skills but quite good in writing skills. After the Project ended, Debbie progressed dramatically in her acquisition of high math skills as well, and in the following year (1987-1988), became a member of the high-math group of her fifth grade. It seems that the combination of: changes in the Experimental children's attitudes towards learning and towards new and open-ended projects; positive responses they received from their teachers and peers about their pieces of software and ideas; their overall joy and satisfaction during the Project and feeling positive about themselves afterwards; their particular writing, designing, programming, and learning processes throughout the Project; their being able to work on their own as well as with other people during the Project; and their particular cognitive progress from rigidity to generating and understanding complexities--should be all equally considered as important factors when we examine the results from these children's Post-Tests, or assess what the Project meant for Debbie, or for all the children in the Experimental class.

## II. The children's intensive and meaningful use of the Logo programming language to design instructional software on fractions, over a long period of time, was an unusually effective way to learn Logo programming concepts and skills.

The Software-Design-Logo learning approach that was implemented in this study was strongly inspired by and very similar in nature to Papert's approach to programming (e.g., 1980, 1986), and radically different from the typical "Logo Course" or "Logo Curriculum" approaches, the most predominant ones used in schools today or in Logo research over the last decade (e.g., Pea & Sheingold, 1987; Soloway, 1984; Carver, 1987; Perkins et al., 1985; or Heller, 1986). Software-Design-Logo took Papert's ideas one step further. It consisted in children's using Logo for the purpose of designing and programming instructional software. It involved the children in programming something for others, rather than just for themselves. It seemed to enhance children thinking about a target audience and constructing a program that would work for a younger pupil in a lower grade. The Integrated-Logo projects (i.e., in Project Headlight, see Chapter I, Section 1.2 and 3) usually lasted from a few days to three weeks, whereas the Software-Design-Logo Project lasted four months. The <u>length, structure, and complexity</u> of the Project enhanced the children's learning of Logo, as well as their reflections, revisions, debugging, and maintenance of their programs. In addition, the children in Software-Design-Logo had their minds on other issues besides "a program that works." They worked on their Project while thinking about teaching other pupils, about screen designs, about interactivity and feedbacks--much like professional software designers.

If knowing programming means understanding of basic concepts such as procedures, variables, functions, conditionals, and transformations (all of which are used extensively in programming), then the children's software products and the results presented in Chapter III prove that the Experimental children learned these skills. They demonstrated expertise in generating and interpreting both their own or someone else's code. Debbie's software provided us with an example of how she was able to generate a great amount of code in an open-ended free way during her software production, but we also saw how she was able to do so very efficiently and precisely, following the pre-defined constraints and requirements of the Logo Post-Tests. Through the Software-Design-Logo approach they also experienced a great variety of debugging situations; it seemed that they learned several general ideas about debugging, as was shown in the Post-Tests by their superiority over the Control children in their debugging strategies and achievements.

To summarize, unlike many Logo studies over the last decade, it has been shown here that the Experimental children did not find it difficult to learn Logo--on the contrary, they learned a great deal of Logo, and enjoyed it tremendously. They were not offered explicit and direct instruction, and they did not complete well-defined exercises or worksheets that highlighted or simplified complex Logo concepts; rather, banal as it may sound, they learned Logo by working intensively and productively with Logo. Their programming activity was integrated into a larger context of software design and development (in the domain of fractions), and when extensive practice, meaningful objectives, and a greater amount of time "on task" were available in the process of learning how to program, the children were able to learn and understand Logo on an even higher level. In short, the children's involvement in a rich, meaningful, and complex task, working towards designing and programming a "real" product for "real" people, enhanced their understanding of what Logo is and their knowledge of how to use it.

The <u>great differences between the Experimental class and Control class 2</u> had been predicted from the beginning. The children from Control class 2 learned programming once a week in the school's computer lab and were <u>not</u> involved in large-scale, meaningful, or complex programming projects; in the five-month period between the Pre- and Post-Tests, these children programmed a total of approximately 23 hours. It was therefore assumed that they had not developed understanding or expertise in programming as well as the Experimental children, who had spent a total of 70 hours programming. Still, it was important to compare these two classes, since many studies in the Logo field are conducted with children of the "Control class 2 type." Such studies have influenced many of the research and development projects used today in the field, and have determined the instructional strategies to be used in classrooms. These studies often link the children's low performances in Logo with the fact that they have not been taught it properly, if at all; they also recommend explicitly teaching children what they should understand about programming concepts, and instructing them how, by using various models and techniques, they can best approach different programming problems. Many "Logo-programming courses" have been developed, and many Logo exercises and worksheets created, in view of an isolated "Logo curriculum" to be used in a class (such as Control class 2). But many of these, in fact, miss the purposes and the major ideas rooted in Logo (or programming in general), reduce the creative learning possibilities offered to young children though Logo programming, and minimize the use of Logo as a thinking and learning tool, or a "knowledge re-formulation tool" (Papert, 1980) to be integrated into the learning of other subject matters.

Although the differences between Control class 2 and the Experimental class had been expected from the start, this was not the case with Control class 1 and the Experimental class. Since both of these classes were part of Project Headlight (i.e., Control class 1 was programming as intensively, and for the same number of hours as the Experimental class, and worked on meaningful projects that were integrated into the curriculum), no one could really know whether they would show any difference in their knowledge and understanding of programming by the end of the experiment. However, there were many differences, which proved to be quite interesting. It seems that the Experimental conditions (Software-Design-Logo) helped develop a higher-level knowledge and expertise in Logo programming for all the Experimental children on all math levels. On the other hand, Project-Headlight conditions also developed some kinds of expertise among the children of Control class 1, since after all they had consistently scored higher than those of Control class 2. But when compared to the Experimental class, only the high-math children of Control class 1--and only occasionally--achieved the average level of Logo understanding of the Experimental children; whereas the medium and low-math children from Control class 1 did not seem to benefit as much from the Project Headlight conditions. The same math-level children in the Experimental class of the same main level (i.e., medium and low-math) did much better on the Logo Post-Tests than those in Control class 1. The results from the Logo Post-Tests therefore demonstrate that the Experimental children, working under the Instructional Software Design Project conditions, were those who improved most dramatically in their Logo programming knowledge and skills.

**III. The children's designing of software to teach third graders, using Logo, was an effective way for learning, creating, and translating rational-number representations, and for enhancing their mathematical understanding of what fractions and basic fractional representations are. This method also affected their success in the final Post-Tests.**

The Instructional Software Design Project consisted in children's working <u>between</u> several representational systems, <u>combining</u> the different islands of rational-number knowledge (e.g., connecting or translating between two different rational-number sub-constructs such as 50% and1/2), and <u>creating</u> and <u>translating</u> several representational modes (e.g., designing a screen that combined both graphical and written representations for 1/3).

Contrary to previous projects, the present study did not attempt to identify the "best" manipulative aids for the fractions learning process. It emphasized, however, that while working with Logo, children could decide by themselves which representations they wanted to create and manipulate. Since many representation for fractions could be programmed in Logo, children could choose which ones they felt comfortable with, and which ones they wanted to learn, work with, or combine. The computerized Logo environment proved to be rich in its potentially available representations (i.e., the representations were not there, in the computer, but the children created procedures for many representations) and in its allowing children's combinations of several representations at the same time and on the same screen. The Logo programming <u>code</u> itself became a new kind of rational-number representation, which the children generated and manipulated; we saw how the writing and the execution of a piece of Logo code could be, for example, a procedural representation of a pictorial representation of a fraction.

It seemed that the Experimental children did well on the Fractions Post-Tests not because they were explicitly and formally instructed on fractions and their representations, but because--through designing and programming instructional software on fractions, creating representations of fractions through Logo, and explaining and teaching about fractions and their representations--they had the opportunity to think about, reflect upon, revise and experience (although on a very small scale) the principles involved in the relationship between the different rational-number sub-constructs and their representational systems. The designing and programming of representations for fractions, and the creating of graphical displays for equivalence and operations of rational numbers, influenced children's thinking and learning about representations of fractions and the relationships between them. Furthermore, the children's goal of representing and explaining fractions to another person helped them to overcome their own difficulties in understanding fractional representations (which had been identified in the Pre-Tests and reported in other studies about fractions; e.g., Behr et al. 1983; Lesh et al. 1983; or Tierney, 1988).

But in addition to getting higher scores on the Fractions Post-Tests, the Experimental children also talked, thought about, and related to fractions in a special way, both during their involvement in the Project and in the interviews and tests that took place afterwards. From their point of view, they learned "a lot" about fractions through this project because, using their words, "how can you teach it if you don't know it yourself?" Much like the experience of the Israeli fighter pilots, who gain new perspectives and understanding about flying by flying with and teaching less experienced cadets how to fly; and much like the experiences of 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 an example, one girl told me in one of our interviews: "I think I now know about equivalent fractions and addition and subtraction. I explained it in my project. I didn't teach about division in my program 'cause I don't understand it myself." Another child said: "I learned how to do fractions [represent fractions] in many ways...you can show fractions using almost anything you want, shapes, groups of shapes, words, buildings, money, or food. But now I know that you have to be careful when you divide them into parts. They [the parts] must be equal!" I found it exciting that 9 and 10 year olds should discuss what was most difficult about fractions, how the same fraction could be represented in different ways, and how the equivalence or addition of fractions could be explained both verbally and graphically.

Logo, the computer, and the Designer's Notebooks played very important roles in the children's thinking about what fractions were. It appears that the Experimental children developed a clear concept of the role of the computer in their own learning of fractions and in that of their "users." One girl said: "I love drawing the pictures in my Designer's Notebook. This way I know what I want to teach. But with the computer you can see it or do it yourself. It's much more fun." One boy said: " With the computer, I got answers to what I did wrong [in programming or in fractions], and they [the users] get answers to what they do wrong [in the tests]. I give them the answers, show them things, like that 2/6 equals 1/3, and I tell them how to do things like that." Another girl stated: "On the computer you learn better about fractions. They're [fractions] not boring anymore, 'cause I make them with colors and I use different shapes and pictures and I make stories about them [about fractions]. It's fun." The Instructional Software Design experiment invited children to think about their own knowledge of fractions, and how this knowledge might become their own, or someone else's. They were also able to distinguish between what they learned from the process of creating instructional software, and what they saw other children learn from their interactive lessons on the computer. As one girl said, "Some of my things [fraction representations] were very confusing at first. At least they were confusing to me. Now I know these things. But this [my piece of software] is just a little demonstration with simple testing. It should be used with my good explanations before, during and after you use it."

Finally, hardly anyone in the Experimental group was able to design or program more than a maximum of fifteen representations during the course of the Instructional Software Design Project. In the process of learning fractions in the Project, the children did not use any manipulative aids, cover a very wide range of rational-number sub-constructs, or translate or manipulate a large number of representations. However, the Post-Tests did include a large number of rational-number sub-constructs, their multiple representations (in the pencil-and-paper test), and a variety of manipulative aids (in the Interview). Therefore, these tests were assessing some kind of transfer, in the sense of "learning and understanding." The Experimental children did well on these tests and transferred the knowledge they had gained from producing software about fractions to the tests quite successfully. It was not because they had actually learned with these materials that they succeeded, but because they--through designing and programming instructional software about fractions--thought about and experienced (although on a very small scale) the principles involved in the relationships between the different rational-number concepts and their representations.

**IV. The children's working at representing, teaching, designing, and instructional designing enhanced their metaconceptual and metacognitive awareness. The Designer's Notebooks, among the other production tools that were used in this Project, were very useful in the children's design processes, and seemed to enhance their planning, reflections, and other metacognitive and cognitive control skills. The children also acquired cognitive flexibility, control over their solution processes, and confidence in their thinking.**

Another goal of this thesis was to find out whether the Project encouraged children's <u>cognitive awareness</u> (i.e., thinking about their own thinking and learning), <u>cognitive control</u> (i.e., managing and controlling their own learning, constructing their own plans and revisions, allocating their time throughout the Project, accomplishing the task given to them, and being able to reflect on all the above) and <u>meta-conceptual understanding</u> of the topics involved (i.e., thinking about the structure of fractions and their representations, how they had come to perceive it, how other children might learn to perceive it, understanding the meaning and purpose of Logo commands in various situations, thinking about the purpose of their screens and the reasons for designing them in particular ways, etc).

The Project did not involve any direct or explicit teaching of these skills; rather, it emphasized them as being completely integrated and interlaced with the learning of the subject matters involved. I therefore assumed that treating and discussing the learning of these skills as isolated units would harm the children's acquiring of them. However, no <u>formal</u> instruments were used for measuring the success of this approach, or for comparing it with other approaches. In other words, no systematic measurements were made of the children's metacognition and cognitive levels before the Project started, or of their improvement in their metacognitive and meta-conceptual knowledge of the topics involved after the Project ended; comparisons with Control classes regarding these skills were not conducted in this context either. But several attempts were made throughout the Project to encourage these higher-level cognitive processes, as well as to document them among the Experimental children.

All children had to develop some kind of expertise in terms of their <u>cognitive control strategies</u> because they were in charge of their own learning and progress throughout the Project. <u>They</u> were the ones who decided what to do each day, what to teach, how to allocate the time they spent working at the computer, what to design, what to implement, how and when to revise their plans or programs, etc. Indeed, they managed to do these things quite well, since, after all, each one of them succeeded in understanding and solving the given task of software design, achieved a great deal through the Project, and programmed a complex piece of software.

Evidence was also gathered on the children's metacognitive skills during the Project. I followed and observed the software designers every day while they were working on their designs and programs, and often asked them questions which required that they reflect upon their own thinking: why, for example, they were creating a particular thing on the computer or in their Designer's Notebooks, and what they thought about the issues they discussed with me or with their teacher and friends. During the course of the Project they became quite good at answering these questions (i.e., they were able to think better about their own thinking and learning).

This Project seemed to enhance the Experimental children's "metacognitive experiences," as well as to promote the growth of several cognitive "executive processes" and skills. According to Flavell (in one of the first papers on metacognition, 1979, *Metacognition and Cognitive Monitoring)*, the term "metacognition" generally refers to "metacognitive knowledge," that is, a cluster of competencies that have to do with people's understanding of their own thinking and other cognitive processes; and "metacognitive experience," that is, a cluster of experiences that occur before, during, or after a cognitive enterprise, and which stimulate highly conscious thinking and encourage the establishing of goals and the activating of metacognitive knowledge and strategies (i.e., executive or cognitive control processes). In a slightly different way, Schoenfeld (1985, *What's The Fuss About Metacognition?)* specified three related but distinct categories of metacognitive intellectual behavior: knowledge about one's own thought processes--one's control or self-regulation and one's beliefs and intuitions about bodies of knowledge and ways of knowing them. In the light of Flavell's and Schoenfeld's perspectives on what metacognition is, it is clear that this Project encouraged the Experimental children to undergo a great many "metacognitive experiences"; they learned and very often activated metacognitive knowledge and control strategies of various kinds; they also came to change many of their beliefs and intuitions about their abilities in math, programming, or problem-solving in general. I shall now specify and discuss some of these strategies and skills, and show how they were acquired by the children during the Project.

1) Through the Project the Children Developed Problem-Finding Skills. For four months, the children were involved in day-by-day processes of self-generating or self-discovering problems 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 in order to explain fractions, what Logo procedures to create and how, etc. They were also involved in constantly specifying their ideas for possible solutions, and in re-defining and revising the problems and the solutions they had thought of, or found.

2) Children Developed an Awareness of the Skills and Processes Necessary for Solveing Various Problems. The Designer's Notebooks required that children design many of their screens in advance. The children's drawings and planning in their Designer's Notebooks demonstrated that they were aware of the Logo-programming and fractions knowledge and skills needed for accomplishing their designs, in that they rarely came up with a design that they could not manage to solve in Logo. In addition, they were aware of their knowledge of fractions and worked along the edge of their knowledge. They also had to be aware of their users' knowledge of fractions and to clearly explain the representations they had created on the computer.

3) Children Learned how to Activate Various Cognitive Rules and Strategies. The children's being aware of the needed strategies to solve a problem at hand was one important aspect of the Project, but activating these strategies was another. We have seen in the Logo Post-Tests, for example, that the Experimental children were aware of the skills needed to optimize or modularize a given Logo procedure, and also of the time constraints; moreover, they were able to activate the proper knowledge, rules, and strategies in solving the problems given to them. We have also seen how at the beginning of the Project, Debbie was not fully aware that there might not be enough time to accomplish all of her plans; later in the Project she wrote, "I'll do this if I'll have time," or learned to plan less and to allocate her time better.

4) The Children Adjusted their Cognitive Efforts to Match the Difficulty of the Problem at Hand. Often children would begin to implement their designs in Logo; but when they realized that too many efforts were needed to accomplish a rather simple or "unimportant" design, they ceased working on it and moved on to a more important screen for their software, or decided to re-design the problematic screen. We saw, for example, how Debbie designed one opening screen for her software that read: "WELCOME TO FRACTIONS!" She began to implement it, and when she realized that too much time and

too many efforts were needed to accomplish the programming of each letter, she decided that the efforts and time needed were too great in relation to the general importance of that screen in her software; she then designed another opening screen instead, and concentrated her efforts on a problem that was more important to her, or on an instructional screen that had a more significant role in her software.

5) Children Developed Cognitive Flexibility. During the Project, the children developed the ability to discard inefficient designs, plans, and solutions that were not working, and to search for better alternatives. We have seen, for example, the Experimental children's number of trials in the Logo Post-Tests. They rarely gave up, were not rigid in their solution processes, and did not stop working on a difficult problem (whereas many of the Control children simply answered "I don't know"); rather, they kept on trying until they had found the right solution.

6) Children Learned to Control Distractions and Anxiety. In this open-ended Project (and in Project Headlight in general), the children were working in an open area next to their classroom. Different children were working on different problems, with children, teachers, and visitors often walking around; still, the children learned to keep their attention focused on the problems they were working on, and to resist being distracted by external stimulation (such as noises, other people's behavior, etc.). They also learned to control their anxiety when a problem was difficult. During the Post-Tests I realized that the Project Headlight children seemed to be better at avoiding "test fears," focusing very efficiently on the problems given to them and not letting external interference distract them in their productive thoughts and actions.

7) Children Learned to Monitor their Solution Processes. The Experimental children were constantly relating their ongoing performances and implementation phases to the general goals of the task (i.e., designing instructional software); they also made appropriate changes if they decided that their performances were too slow or unclear, or that they were unlikely to reach a successful solution to the task given them. They also evaluated their performances every day when they logged in and ran their software, or when other people in the "culture" used parts of their software.

8) The Children Developed Faith in their Thinking. Because the children themselves were in charge of their own learning and production, they knew that when they had a problem or difficulty, their own thoughts and actions might help them to solve the difficulty and generate the needed solution. We saw by the Post-Tests that whenever an initial solution

did not work for the Control children, they stopped trying; however, most of the Experimental children learned to appreciate that their own mental work could be useful and result in their being successful in the test.

In this extreme learning environment, design, or instructional design, and mathematical thinking and programming processes became integrated and overt; metacognitive and meta-conceptual thinking were facilitated and explicit in the "culture." The records of the design and cognitive processes that were created by the children themselves benefited the child, the teacher, and the researcher at the same time.

My conversations with the children, as well as the short classroom discussions that were conducted during the period of the Instructional Software Design Project, were very useful vehicles for encouraging and gathering data about the children's meta-conceptual thinking. These discussions and conversations focused, for example, on the difficulties of specific fraction concepts or of Logo programming techniques. They required that the children generate ideas on why some fractions concepts were difficult, and how they might be explained, represented, or taught. In two of the classroom 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 OR REPRESENTATIONS COULD BE DESIGNED TO EXPLAIN THESE DIFFICULT CONCEPTS?" We asked the children to generate ideas for both posters simultaneously, by asking them: "Think about what was difficult for you, which concept you did not understand at first, and what it means to know this concept." Long lists of ideas were generated by the children, meaning that under these conditions they were able to think about these issues and express them verbally in the classroom--and later in Logo, through their pieces of software. The children, as instructional software designers, built their knowledge of fractions, Logo, and instructional design by going back and forth between at least three different modes: **1)** the child's awareness of his or her own knowledge (i.e., meta-conceptual thinking, for example, "What do I know about fractions?" or "Can I program this thing in Logo?"); **2)** his thinking about how another child might come to know it (for example, "What screen should I design to clearly explain how 3/6 equals 1/2?" or "Should I use the same or different colors to show that all the objects on my screen are one half?"), and **3)** his thinking about what other people, such as the other children in his class or his teacher, thought or did about these same concerns regarding fractions, Logo, or design (for example, "Let's see what Debbie did today," or "Gee, Naomi's game about halves and fourths is really cool, and really difficult to program" or "My teacher's explanation of this fractional algorithm was quite confusing" or "Why did Idit (the researcher) asked that question about my screen?").

During this highly interactive and recursive process--the instructional software design learning activity--the children, who were at the same time learners, designers, and computer-program evaluators, came full circle and became aware of their own process of learning of a given concept, of the structure of a specific piece of knowledge, and how it related to other forms of knowing, such as a teacher's way of teaching, a friend's knowledge, another child's piece of software, or the conventional final fractions test. And as a result of the above, in terms of their cognitive and metacognitive development, we saw how the children learned to break away from rigidity and literal thinking and become more aware of their problem and processes, more fluent and flexible, and better able to grasp and create complexities.

The writing in the Designer's Notebook was very important for the production process and for children's metacognitive development in general. We did not tell the children what to write, how much to write, what or how to plan or draw, or how to reflect or make changes. However, like professional software designers, they kept track of their ideas and changes, which also helped their implementation, their concentration, their not losing good ideas from one day to the next, and so on. The writing in the Designer's Notebook was a problem at first, since the children were not accustomed to handling in writing such a routine of planning, note-taking, and reflecting. But after two weeks into the Project, the Designer's Notebook became a personal and important element for the children and made them aware of the benefits of reflecting, keeping track of their own planning, note-taking, and changes. Moreover, they realized that they did not need to implement what they wrote unless they wished to; they realized that the Notebook facilitated their thinking of "new" ideas while still implementing "old" ones; that going back and forth in their Notebooks to "old" drawings and notes was beneficial to them and very useful in their programming processes. We saw several examples in Debbie's Case of her "personal relationship" with the Notebook, and of her use of it for planning and reflecting on new ideas as well as on changes she made.

The Designer's Notebooks were also found to be useful and informative research instruments for assessing metacognitive growth among the children. In Debbie's Case we saw many examples of this. Thanks to the Notebook I was able to accumulate records on her design and reflection processes, her plans, drawings, changes, problems, etc. At this stage I can only offer a brief analysis of the Designer's Notebooks and Logo files of the 17 Experimental children, but it is clear that the Notebooks themselves reveal precious information on the differences and similarities in the children's higher-level cognitive and metacognitive skills. Through them I realized that some children liked to plan in writing in detail, while others chose to plan less in writing, preferring to draw more screens and create more story-board scripts. Some reflected in detail on the problems they had encountered during the sessions, while others did not consider the problems they had managed to solve as problems, and therefore did not report on them. Some children, as they progressed in the Project, created more complex and detailed plans, while others came to plan less and less, since they, using their own words, "already knew what they wanted to do." A deeper investigation and analysis of these Notebooks would probably reveal very important information about the children's processes of planning, designing, and reflection, how they became more adept at these during the Project, and what they gained by them. Another investigation should be aimed at relating the 17 children's plans and reflections to their actual programs in Logo (their implementation). And while Debbie's Case revealed to us very important information in that regard, 17 cases, constructed and interpreted like Debbie's, would allow us to generalize and build more detailed models for these processes.

In order to find more information about the children's metacognitions, a more systematic questioning method could be developed for use in this context: the researcher, for example, could ask all the children a pre-determined set of "metacognitive" questions every day, compare the children's answers, and measure more precisely their metacognitive development. However, we should not forget that these questions might also lead the children to follow different routes from those they would spontaneously choose to take; and if we are interested in children's own inventions and in the meanings they construct in their processes and give to their products, it is clear that in many cases a researcher's intervention might interrupt instead of support the children's own processes.

My philosophy during this Project was not to ask too many questions, to let the children work as they wished, and to ask them meta-questions only after I saw something that captured my interest. When asked, many of these children spoke of the importance of their process of thinking about the future users of their programs. They often commented on, and were aware of, what they had learned about fractions and Logo from their creation of a particular screen, and what was or was not interesting for them or for their users. They were ready to describe the rationale behind their screens' designs, why they had designed a screen in a particular way, which problems they had encountered, why they had selected certain colors or shapes rather than others or provided particular instructions, explanations, or introductions, which feedback they had given and why, when and why they had combined graphics and text, etc. These observations, as well as others that have been made in my pilot study (Harel, 1986), demonstrate that the children were involved in a higher-level and reflective form of thinking across the board--in the ways they thought about their own knowledge of fractions and Logo, their own design strategies, and their own learning throughout the Project. To my mind, placing children in a learning situation where these issues were actually raised in the first place, as well as discussed and shared, is a powerful and appropriate measure of metacognition in its own right.

Often, during my Project, I would wonder how much "scaffolding" I needed to provide, whether I was asking a child too many questions, or interrupting him rather than helping him along, or how much problem-solving modelling should I do. These are great dilemmas for people who believe in constructivist-discovery types of learning and are interested in what the child can construct and invent on his own--but are also interested in "what is going on inside the child's mind," in the development of the child's metacognitive awareness and control, and in how the child has come to develop his abilities to think about or answer the "meta-questions" posed to him by the researcher.

**V. Instructional software design proved to be manageable by fourth-grade children, and a vehicle for their learning of problem-solving in general. Through the intensive and constructive experience of software design, the children experienced complex and productive problem-solving processes and also came to learn and understand different subject matters, concept and skills at the same time. The analyses reached through the study of one individual pupil suggest new methods and models for analyzing children's different styles of instructional software design, as well as the different stage sequences in their organization and implementation of long-term projects. More comprehensive and global models of the processes involved in children's software design could be developed by analyzing several other cases as was Debbie's Case in this thesis.**

In Chapter I of this thesis, I described several of the studies whose aim was to examine the processes involved in software design (e.g., Adelson and Soloway, 1984; Guindon, 1987; Guindon, Krasner, & Curtis, 1987; or Jefferies, Turner, Polson, & Atwood, 1981) and I concluded that these studies provided a detailed description of very specific cognitive processes in software designers with various levels of expertise, performing very logical and well-defined tasks over a short period of time. Only a few generalizations could be drawn from these researchers' work that were strongly relevant to my study. Furthermore, besides the collection of verbal protocols and the use of videotaping, only a few models existed that could be directly adopted for the Instructional Software Design Project, since the problem given to my subjects was open-ended and presented in the following manner: "Design a program that teaches a younger child about fractions." The children/designers' processes toward solving the problem took approximately 70 hours of programming, and within this long and complex context each child established his own route, goals, and sub-goals, which could not be predicted in advance in a production-system model, or later analyzed by the researcher--in the sense of "good" or "bad" designs, or "expert" as compared to "novice" approaches.

In the present study I placed great emphasis on what was learned through the long solution process, and how it was learned. In Debbie's Case, I emphasized these aspects throughout. I paid careful attention to the skills the problem-solver gained in the solution process rather than limit myself to analyzing her solution processes *per se* or the qualities of her final product.

Unlike the participants studied in the literature mentioned above, the children in my Project learned about the a computer language (i.e., their major software design tool) during their actual processes of software design. And whereas the existing studies emphasized the role of prior domain experience and prior knowledge of designing software, the children I studied here learned the domain while producing the software for it. These processes of learning were a main object of my study, rather than the relationships between knowing more fractions or Logo and producing better software. In fact the relationships I stressed were quite different: it did not matter what particular product each Experimental child produced, or whether particular concepts were included in one child's software and resulted in his being more successful in the Post-Tests than another who had not included it; rather, emphasis was placed not only on how the processes for producing these different products resulted in the Experimental children's learning and understanding the topics, concepts, and skills involved, but also on their consistent superiority over the Control children in the Post-Tests.

Software design proved to be a great vehicle for children's learning and problem-solving. This study indicates that through the intensive and constructive experience of Instructional Software Design, the children learned many Logo operations and commands and wrote long and complicated programs which pupils of that age are not usually considered able to manage. The children also demonstrated expertise in the understanding, interpreting, and modifying of their own Logo programs, as well as those of others (in the Post-Tests). At the same time, the integrated approach manifested in the software design activity (as shown in the diagram on p. 1) was a challenging way for the children to learn basic rational-number concepts. Their experience in this complex task, which is rarely mastered in school mathematics, encouraged them to engage in difficult academic constructs such as relationships between fractional representations. Throughout their Instructional Software Design activity, the children were constantly involved in metacognitive activities: creating and discussing knowledge representations, finding instructional design strategies, learning through explaining, and reflecting on all the above.

In my project, new research methods were used for studying children's open-ended and prolonged problem-solving processes, and proved to be successful for the children; for the researcher, they were very informative on many levels. This experiment also aimed at shifting away from what has been most predominantly emphasized in problem-solving research over the last decade, concentrating instead on children's problem-solving and cognition in a richer and much more complex learning environment, and over a longer period of time. On the whole, the Project suggested two major methodological changes for studying children (or even software design): the first was to study an open-ended problem-solving enterprise, the second to study children solving the same problem over a long period of time (70 hours), rather than in various short and well-defined problem-solving sessions (the most common approach in the field). All the Experimental children succeeded at the same "job," producing a piece of instructional software, each one in his or her own way. As shown in previous findings on Logo (e.g., Turkle, 1984; Weir, 1987; Lowler, 1985; Papert, 1980, 1984a, 1984b), software design also proved to be a learning activity that allowed for individual differences in learning, mastery, and expression. New microgenetic phenomena were seen in this extreme environment--this was shown, for example, in the way the different children went through different stage sequences in organizing and implementing their long-term projects. New methods would have to be developed for identifying, analyzing, and describing the individual differences between all the participating children, their various styles of expression, and the different ways they created their different products. Furthermore, these phenomena are worth further exploration also because the complicated learning environment that was implemented in this study touches upon the complexities of the everyday world in which children and adults learn and live.

A comprehensive analysis of more children/software-designers cases would probably reveal an interesting model, or several models, of the processes involved in children's software design; these models could be used later and more systematically in future studies of this kind. Better strategies for studying the processes could also be revealed by interpreting all 17 cases as I interpreted Debbie's. The data exists, and more cases like Debbie's should be, and will be constructed in the near future.

**VI. The Project offered major changes for educational practice in general, and for the teaching of fractions and Logo in particular. The techniques developed in this study could be used for teaching fractions and Logo programming, as well as other subject matters, in other schools. However, these techniques must be implemented within an umbrella or educational philosophy that is child-centered, open-ended, dynamic, and oriented towards working on common projects and on children's construction of meaningful products; connecting various aspects of their knowledge, and integrating them into an environment that is rich in computers, and in which the computers are fully integrated into the school's curriculum and culture--all of which were established in Project Headlight's model school.**

The experimentation in this prototypical Project aimed at exploring what major changes the Project offered in educational practice in general, and in the teaching of fractions and Logo in particular. In Debbie's case I investigated many of these issues. In Chapter III, I described in what ways the Experimental children succeeded in the Post-Tests. Their consistent superiority over the Control children revealed that the changes offered to the existing curriculum by this Project were very effective for the children's learning and understanding of fractions and Logo, among other things.

The first change this Project offers to educational practice relates to the fact that different subject matters and skills were found to be mutually supportive of one another, were learned by the children at the same time, and contributed to each other in many ways. This idea was highlighted throughout this thesis. The Project was a rare pedagogical approach, difficult to implement in educational practice. Still, we saw that the Experimental children did not find it difficult or confusing in any way to learn different skills or subject matters at the same time; in fact, they benefited from this pedagogy, which resulted in the learning of one skill's contributing to another. Chapter II provided us with Debbie's processes of parallel learning: she learned fractions and Logo at the same time, and often the learning of Logo contributed to her understanding of fractional representations, or the learning of fractions contributed to her acquiring new Logo skills. Chapter III provided us with the evidence that the learning and understanding of the topics and skills involved did occur in Debbie's case, as well as in the Experimental class in general. On the whole, this was a Logo-based and fractions-based Project that promoted and integrated thinking and learning among fourth-grade children on many levels and in several domains at the same time. As Perkins wrote,

> "What does one school subject have to do with another? Whatever answers there might be, it is clear that conventional schooling pays little heed to them. The several subjects run their courses as separately as rivers on different seaboards. Yet building connections is not hard....the usual subject matters need not stand so distant from one another. In fact, it might be said that they only stay so because they are left there. Instruction based on knowledge as design need not, indeed should not, accept this status quo. Knowledge as design is a natural bridge builder, pointing up commonalities and inviting contrasts between the various disciplines" (my emphasis, Perkins, *Knowledge As Design*, 1986, pp. 221-222).

Another change that the Project offered to educational practice lay in the learning of thinking skills and cognitive control strategies' being completely integrated into the subject matters involved. They were integrated to such an extent that it was even difficult for the researcher to study and document these skills as separate from the content knowledge involved. However, in conventional schooling, thinking skills are usually taught in separate courses, outside the normal curriculum (in the same way that Logo is taught separately). The Instructional Software Design Project argues for integrating the learning of these skills into the subject matters, because, "without integration into subject matters, thinking skills are not as likely to empower the active use of knowledge for critical and creative thinking" (Perkins, 1986, p.217).

Other points this Project stressed were the children's learning through producing products, and the teacher's focusing on her pupils' processes of learning and her favoring a variety of complex, large, non-unified, and creative products, rather than unified short "right" answers as the outcomes of her pupils' learning activities. We must remember that

> "...outside of school almost all worthwhile activities involve products rather than short answers, whether the product is a plan for an advertising campaign, a poem, or a well-crafted chair...[learning through the processes of design and construction of products] reveals designing to be the paradigmatic human activity, and designing anything invariably involves producing some sort of extended product rather than a brief answer. Although some might think that students do not know enough to deal with products, [a Project of this kind, and many other tasks

which are mentioned in Perkins' book shows] how appropriate tasks can be found and [offers] numerous examples for a range of subject areas" (Perkins, *Knowledge As Design*, 1986, p. 217).

In summary, this Project, with its approach of learning through designing and programming instructional software, offered new conditions for learning. This meant that young children learned fractions, Logo programming, designing, instructional designing, planning, story-boarding, reflection, explaining, self-management, etc.--all at the same time and in a synergistic fashion. In the context of the Project, and through the use of the computer, these different kinds of knowledge and disciplines were integrated and proved to be mutually supportive of one another and contributed to each other while the child was in the process of learning them.

This particular Project offers changes in the approach to the learning and teaching of Logo and fractions concepts and skills, and of several important and more general thinking and problem-solving skills. But it also proposes a more general approach that could be implemented with other topics in the elementary-school curriculum. Children should be able to learn other topics, such as science, geography, literature, art, and social studies through this method of instructional software design. Further experiments should be conducted for investigating which topics lend themselves best to this approach, and how this method contributes and changes children's understanding of these topics and their ways of thinking about them.

Several teachers have already approached me, requesting curriculum materials that would guide them in implementing this Project and in adopting a specific pedagogical approach in classrooms. Researchers have also requested these materials, so that they might implement similar studies on children's cognition and problem-solving processes. The latter are accustomed to reading long and complex papers such as the one presented here, and might perhaps find many answers to their questions, concerns, and interests in my thesis. Educators and teachers, on the other hand, are usually less accustomed and have less time for this form of presentation and tend to request clear and precise guidelines and directions on "exactly how to *do it*, step by step." In living with teachers day-by-day within Project Headlight, I grew to be very sensitive and respectful regarding this issue. I learned to admire teachers for their very difficult task of teaching and educating children, and for "covering" all the subjects and concepts they are required to teach throughout the year. It would be an easy thing for me to say to a teacher: "Try it and see," or "experience it and learn." But I learned that these are not appropriate approaches for involving teachers in innovative educational methods. Teachers are part of a very complex and rigid system that does not very often allow them to try new things, or explore new pedagogies even when they strongly wish to learn innovative approaches, or seek changes in their curricula. We saw how interesting, informative, and successful this Project was in the study here described, and how effective were the changes this Project suggested for educational practice and for research in the fields involved. Clearly, curriculum materials are very important for the dissemination and distribution of ideas and instructional methods, for making teachers and educators rethink their curricula, and especially for making educational policy-makers understand the ideas behind such projects as mine and see them as models for offering changes in educational practice. But developing these materials poses a difficulty for me. I have given a great deal of thought to what strategy might be used in writing these materials, what their contents should be, etc.; still, I find it quite difficult to develop concrete, written materials for an open-ended, integrative, and complex Project of this kind. It is difficult to develop a curriculum that would provide teachers with a clear model they could follow, but without step-by-step well-defined guidelines, worksheets, or exercises. The holistic characters of the Project, of the children's learning and thinking in this Project, of the adults involved in this Project, as well as the complex structure of the culture that was created during this Project, are all very difficult to transmit in a written curriculum package.

Could the Project be implemented in this way without my being there? I do not know the answer to this question. I am aware that the individual attention I gave to each pupil was crucial. However, the teacher I worked with played a very important role throughout this Project, and I am sure that she will be able to carry on many projects of this kind in the future, and in the most wonderful way. But, in order for her to learn this, she had to experience it and work very intensively with me and with her pupils. We collaborated and constantly discussed various issues of implementation, as well as the various steps in the children's progress, and their problems; this, to my mind, was the best approach for her (and for me) to learn how to implement and manage a Project of this kind. It should also be emphasized that this teacher had participated in Project Headlight for approximately 20 months before we started to work on my Project. She was very familiar with our philosophy of integrative, child-centered projects, and had the opportunity to discuss with other MIT members, including Papert, and to work on various other projects during that period. This was an important factor in her development as a teacher, and in her ability to implement and manage projects of this kind.

The Experimental children also played a very important role in this Project. We had to collaborate, and we constantly shared our problems and ideas with one another. As I stated previously, I did not have a well-defined plan for the Project as a whole, or for the classroom discussions (the "Focus Sessions"); these were generated on the run, according to the children's requests, or to our realizing that a short discussion (on Logo's IFELSE, for example) was needed at a specific time and point in their processes. Many times it was even difficult for the teacher and me to decide what to discuss with the children and when, since each child took his own personal route in the Project; moreover, at any given time in the Project, different children were involved with different fractional concepts, instructional design concerns, and programming problems.

Could this Project be implemented outside of Project Headlight? I doubt it. It is my belief that computers must first be generally integrated into a school, in the way that is done with Project Headlight, in order for a Project of this kind to become a natural element in the environment. It could probably not stand alone in a conventional school curriculum, for it needs to be rooted in a culture that uses computers very intensively and extensively for other projects of various kinds, which are integrated into other domains through various other methods.

Could this Project be implemented with children who have never used the computer or Logo before? My answer to this is YES. In fact, we saw how this Project became a great tool for both the teacher and her pupils to learn about Logo and experience its "original" philosophy (see Papert, 1980). This Project provided us with evidence and support for the many benefits of integrating the learning of Logo into the curriculum from day one and never as an isolated subject, and of allowing children to explore and construct meaningful products with Logo over a long period of time. This study has shown us that such an approach resulted in children's learning Logo on a very high level. Furthermore, I conducted my first Pilot Study with fifth graders (Harel, 1986), at the end of their school year. They were quite knowledgeable in Logo and fractions at the time they worked on their projects. But by examining the effects of this pedagogy on these fifth graders, I decided to conduct this study with younger children, from the fourth grade, who knew much less about Logo and fractions at the time the Project began. In fact, many of my colleagues had warned me that because of the children's young age and particular stage of cognitive readiness, they might not be able to handle this complex Project, be capable of writing in the Designer's

Notebooks, or manage to accomplish the sophisticated tasks that were given to them. But we have seen how successful these children were, and how much this Project meant to them in terms of their learning, their feelings, and their attitudes towards school learning, cognitive development, etc. The data given in this thesis speaks for itself in that regard.

Several other dilemmas relating to educational practice and curriculum design were raised during the Project and through its analysis. The fact that Debbie, for example, spent a great deal of time on halves (one month) poses an interesting dilemma: How can a teacher detect who "needs" to spend more time on something, as opposed to who "needs" a broader learning experience? This is a very difficult question to answer. In fact, in the context of the Instructional Software Design Project, it was the children themselves who were in charge of "solving" this dilemma. We could not foresee that Debbie would actually spend so much time on the concept of halves; nor could we predict in what ways her particular activities during the Project would affect her knowledge of fractions or Logo, or how well she would finally do on the school math tests, or the Post-Tests. However, Debbie's Case, as well as other Experimental children's processes of learning, demonstrate that the long periods of time they were able to spend on one concept affected their total learning in a positive way. The point I therefore wish to make here is that we should think about creating learning contexts that are rich and complex enough so that children will be allowed to make relevant decisions and experience "conceptual principles" even when dealing with one concept alone. The children's creation of a "home base" (White, 1988) as they worked on this new task (the "home base" was one aspect that I highlighted in Debbie's Case), should probably be considered an important aspect in their learning, or in our designing learning environments for them. Also, the Project lasted long enough so that Debbie was eventually able to "free" herself from her obsession with halves, and go on to explore other concepts and algorithms. In addition, the children in this Project were quite independent, so that the teacher was able to walk around, help them, consult and advise them according to their needs. In this way, the teacher found the time to give individual attention to everyone, so that each child's talent was cultivated in a particular way (see Weir, *Cultivating Minds,* 1986). This, of course, contributed greatly to the success of the pupils. I assume that any teacher (or researcher), after experiencing several Projects of this kind, would become sensitive to this issue, and capable of making decisions about children's needs of this kind along the way.

Another challenge this Project poses for educational practice is how to create constructively oriented learning settings in which a great deal of learning would occur for both high and low-level pupils. Several questions could be raised regarding this issue: What are the crucial elements in a learning environment that would allow low and high-level pupils to "blossom" by it? (Project Headlight conditions were proved to be effective mostly for the high-level pupils, whereas my Software-Design-Logo conditions were effective for the whole Experimental class. Why was this the case?) Was it important that the Instructional Software Design Project occur near the beginning stage of these pupils' learning of Logo or fractions? What will happen in these children's next "Fractions Project"? How can we take a pupil who is at the "top" of his learning curve, and put him at a beginning of another learning curve? How can we "renew" the subject matter for students so that when they become sophisticated on "level 1" they still have a great deal to learn in order to be sophisticated on "level 2"?

Another related issue for educational practice that was raised in this context is: What are some of the characteristics of learning activities that, by their nature, involve children in connecting new knowledge with old? How can we encourage children to not "put aside" their finished work and old knowledge, but constantly integrate these with new ideas and pieces of knowledge? It seems to me that we must differentiate between learning activities that foster among children an attitude towards treating their work as a static entity that can be finished up and put aside, and those that foster an attitude towards treating their work or knowledge as a dynamic entity that can be continually revised, manipulated, and integrated with new knowledge and new ideas. But do we always want children to integrate old knowledge with new? Or do we sometimes want them to learn how to "finish" something, leave it aside and go on to another, in order to gain new perspectives or skills? All these questions require further investigation.

Finally, in *Mindstorms* (1980), Papert discussed computers, Logo, and computer cultures in a way that was both very influential and controversial and often resulted in people's rethinking education, reflecting on their own learning, reconsidering the possibilities of using technologies for learning, and reexamining children's cognition and development through using Logo. In 1980, Papert wrote about the computer (or Logo) as being

> "...not a culture unto itself but [as a thing which] can serve to advance very different cultural and philosophical outlooks" (Papert, *Mindstorms*, 1980, p. 31).

And to the teachers and researchers who consulted (or criticized) him about "Logo curriculum," classroom organization, scheduling problems, pedagogical issues, and how Logo "really" conceptually related to the rest of the curriculum, Papert answered:

> "I have thought about it [Logo] as a vehicle for Piagetian learning, which to me is learning***without*** a curriculum" (ibid., p. 31).

This statement raised a great many debates in the field, especially among people who had not carefully read the paragraph immediately following the sentence just quoted:

> "But 'teaching without curriculum' does not mean spontaneous, free-form classrooms or simply 'leaving the child alone.' It means ***supporting*** children as they build their own intellectual structures with materials drawn from the surrounding culture. In this model, educational intervention means changing the culture, planting new constructive elements in it and eliminating noxious ones. This is a ***more ambitious*** undertaking than introducing a curriculum change [or a new curriculum package], but one which is feasible under conditions now emerging...The educator as an anthropologist must work to understand which cultural materials are relevant to intellectual development. Then, he or she needs to understand which trends are taking place in the culture. Meaningful interventions must take the form of working with these trends" (my emphases, ibid., p. 31-54).

The Instructional Software Design Project was a <u>project</u>, not a curriculum. It could be integrated into, and change many of the existing curricula. It implied revision in the learning of "old" information (fractions), but also offered new subjects and matters for the children to learn and think about (computation, instructional software design in Logo, Designer's Notebooks, etc); it certainly did not imply learning each of the subject matters involved as isolated pieces of information. It would be a poor idea, in my opinion, for a school to decide to teach a new "instructional software design curriculum."

This Project meant "learning without a curriculum" in the Papertian sense described above; moreover, it meant learning fractions without a curriculum, learning Logo without a curriculum, and learning planning, reflection, design, or software design--all at the same time--without a step-by-step, well-defined curriculum, but with a great deal of integration and structure, and many rituals and activities that <u>supported</u> the adults' and the children's learning and functioning in that culture. What remains to be explored is how to describe and transmit the ideas in this thesis to teachers, educators, and policy makers, in a way that would not destroy the holistic character of the Project, or the manner in which the teacher, the children, and the researcher learned and thought about it, or interacted and worked together because of it.

**VII. The findings of this thesis revealed the need to create an integrated environment that would allow children to program, design, reflect, plan, take notes, make changes, and in which all the information produced by one child or by a group of children would be presented, simply and efficiently, within the same computer system. More sophisticated and dynamic tools could be developed and used for this purpose. Computerized Designer's Notebooks, for example, would encourage children's reflections <u>during</u> the design and implementation phases, rather than after; a specially designed computer interface, including on-line Designer's cards and links, could assist the children during the different phases of their software presentation or execution; videodisc technology could also enhance their thinking of many more real-life representations of rational-number. My preliminary Ideas for a Logo-Based, Integrative, Software Development Environment for Children are briefly presented below.**

The unique aspect of the Experimental children's day-by-day copying of their Logo files, their saving of them on a special diskette, and their ongoing writing in the Designer's Notebooks lay in the fact that these tools not only represented the children's processes, but also kept track of their ideas, actions, and thoughts while solving the software design task. In this way, a child's software design process and its history became objects of study for both the children, their teacher, and the researcher. Although a great deal of rich data was gathered by the use of these tools (the Logo files and the writing in the Notebooks, supplemented by videotaping and the researcher's notes), other more sophisticated and dynamic tools could be developed and used for this purpose. While working on this thesis, I realized that more sophisticated methods for organizing and presenting the software were needed for encouraging higher levels of metacognition in children, and of their meta-conceptual understanding of the topics involved (Logo, design, and fractions), than were described in my study. In this section I outline my preliminary ideas for an integrative computer environment for software design, software production, and its presentation.

The system I shall outline here is aimed at facilitating <u>young children's learning through software design and programming</u>, rather than the developing of software by experts. It is based on many of the findings and observations of this thesis, and seriously takes into consideration the <u>learning and cognitive</u> (or metacognitive) processes that were (or could be) involved in children's software design. It is based on children's constructivism, rather than on the using of the system for better mediation processes, such as tutoring and intelligent promptings. It is based on the philosophy of Logo programming, rather than on the philosophy of intelligent tutoring systems (e,g., Anderson et al., 1986; Collins and Brown, 1985). In fact, when a sophisticated system is developed, the children themselves could construct intelligent tutoring systems through the use of this system. In other words, it would not provide the child with "better" or more varied representations of fractions, with sophisticated instruction and modelling for solving fractional algorithms, or with sophisticated links between representations and processes; rather, through using this proposed system, the child himself would be able to construct these representations and links in the system. The system would not produce process records for the child, but instead, would allow him to produce his own records and written commentson his processes of software design and production. Finally, the system outlined here is Logo-based, but it is aimed at making the processes of software design in Logo more flexible, and integrates Logo with other on-line design and production tools.

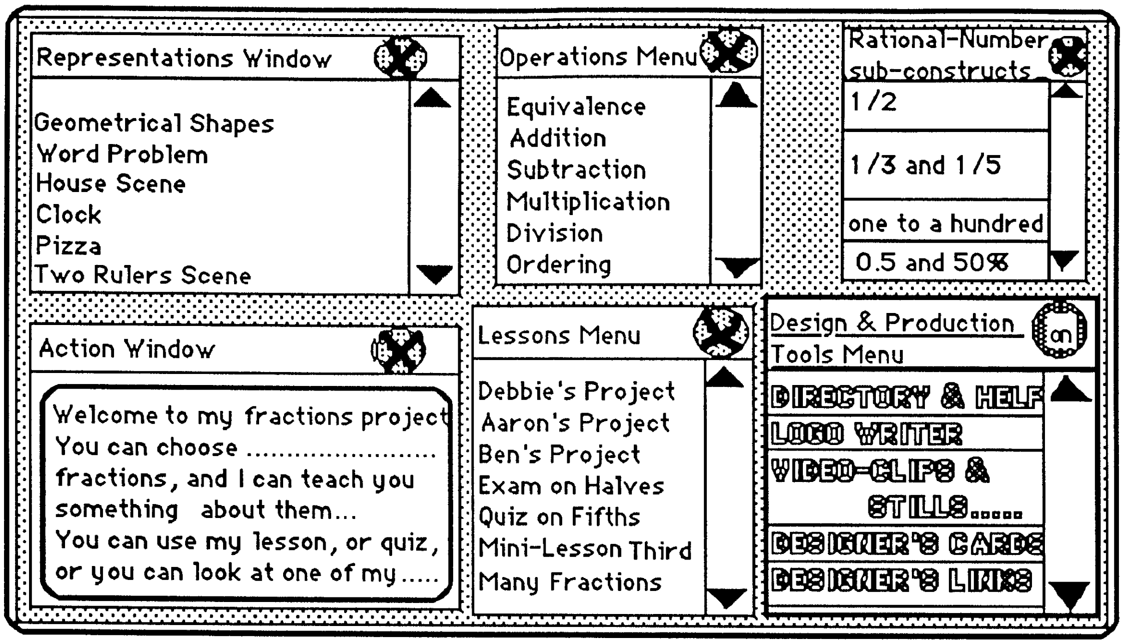
The computer is a very dynamic medium, much more so than the Designer's Notebook, which was, after all a linear, non-integrative, and difficult medium for the child's establishing inter-relations between the Notebook's parts and efficiently keeping track of the history of his ideas, the dynamic relations between these ideas, and his conceptual development during the Project. In the area of cognitive sciences, "computational techniques have proved to be powerful tools for both theoretical and experimental investigations of mind" (e.g., A. Collins and J. S. Brown, 1985, p.1). The computer has an ability to record and represent processes as well as products of thought, and also allows the organization of topics, materials, concepts, in different ways; therefore it makes sense to create an integrated environment that would allow children to program, design, reflect, plan, take notes, make changes, and present the information within the same computer system.

A computerized Designer's Notebook, for example, could become a very useful tool in children's software production; and once it was computerized and strongly integrated with Logo would it really encourage children's reflections during the design and implementation processes. Moreover, it should be linked to the actual programming and production environment for various other reasons that are listed below.

The Designer's Notebook was found to be a very important medium in the children's planning, designing, and reflections. However, one problem lay in the Notebook's being rather large in size, so that, because of space constraints at the computer table (i.e., the computer-table was too small to hold both the computer and the Notebook), the children usually put their Notebooks on their laps when they used them, or on top of the computer monitor or under their chairs when they did not. This was not a very practical solution for the children, and it resulted in their not using the Notebooks as much as they, or we wanted them to, while actually working at their computers. Before and after the computer sessions there were no such problems, since the children wrote in the classroom, at their regular tables.

Another related problem lay in the fact that when the children were typing at the computer, they were obviously not able to freely shift from using the computer keyboard to using a pencil, or vice versa. Many children took notes on fractions, Logo concepts, and algorithms during their processes of programming; but in order to do this, they had to look for their pencil, which in the meantime had fallen to the floor, then find it, move the computer keyboard aside, and in this peculiar way, somehow manage to write in their Notebooks during their production phases. This may explain why the children often did not write in detail about the problems they encountered during their production phases, or forgot new ideas that occurred to them during their implementation processes.

Furthermore, during the month of June, towards the end of the Project, when the third graders were trying out many of the fourth graders' products, or when we attempted to connect some children's pieces of software together into one larger piece (which we called a "package"), we encountered several other difficulties. These difficulties were mainly related to questions of organization and presentation, the Logo interface, its memory space, and its slow program-execution pace when it had to move from one Logo Page to another and search for procedures in other Logo Pages. Other difficulties were related, for example, to two children's giving similar names for their two different procedures, to all the children's wanting their piece of software to appear first in the "package," or to their wanting the "testing procedures" of all the children in their group to appear at the end of the "package," etc. A more sophisticated system could be developed for solving some of these problems. This system should be developed for the purpose of assisting the children during different phases of software design and programming, as well as for improving software presentation or execution. A tentative computer interface for this purpose is sketched below:



Welcome to my fractions project
You can choose ........................
fractions, and I can teach you
something about them...
You can use my lesson, or quiz,
or you can look at one of my.....

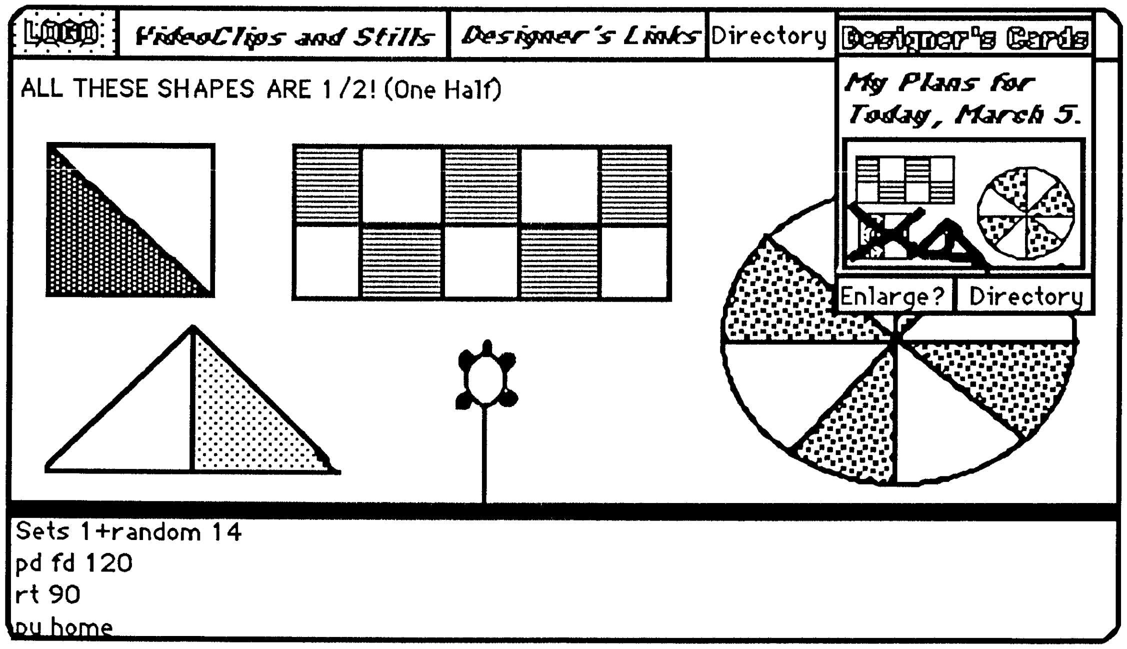
**A model interface for an integrative software-design-and-production tool for young children**

The appearance of this interface might look familiar; however, a major difference between this proposed system and others is that the content of each window would be written by the children, rather than by the system's developer. This interface would be suitable for children's designing, programming, executing, and evaluating processes. It would also be useful when several children were connecting their pieces of software together. This particular model screen demonstrates the system's capabilities and features in the context of fractions (this particular subject being relevant to my thesis), but the system could be also used for any other topic.

A child can activate a window by clicking on the "ON/OFF" button. Then he can select any option offered at any given window. It is important to note that the content of all the windows (excluding the Design & Production Tools Window) are defined and created by the child or by a group of children during or after their software designing.

The only window of which the content is pre-defined by this system's developer is the Design & Production Tools Window. In this window, the child/software-developer can choose whether he wishes to work in the Logo Writer environment, use Video Clips or Stills that are stored in an integrated videodisc environment, use the Designer's Cards environment, or the Designer's Links (the latter two environments I am proposing here are for assisting children's ongoing planning, designing, and reflections, and are based on the role of the Designer's Notebook in the Project). Furthermore, when a child is in one of these environments (Logo, Videodisc, Designer's Cards, or Designer's Links), the other three are always available for interaction and use.

The following screen is an example of how the interface might connect Logo with the plans and designs a child has created on Designer's Cards. The child would be able to call his Designs Cards, Story-Board Scripts, Plans Cards, Reflections Cards, Notes Cards, Grid Cards, etc., while working in LogoWriter:



ALL THESE SHAPES ARE 1/2! (One Half)

```
Sets 1+random 14
pd fd 120
rt 90
pu home
```

**A model interface for connecting LogoWriter with Designer's Cards on-line**

This screen shows the LogoWriter environment being activated and enlarged over the whole screen. Designer's Cards is also activated on the top right of the screen. In this example, a child has chosen to work on one particular plan he has created, and while working in Logo has decided to cross out the bottom part of his original plan (i.e., not implementing it) and briefly sketch a new design instead. Designer's Cards gives the child an option to ***Enlarge*** the Card he is working with to any desired and comfortable size (or "zoom into" it), and to look at his ***Cards Directory*** for other plans and designs. Video Clips and Stills and Designer's Links are not activated in this particular example, but they are available for the child's use while he is in the Logo environment or in any other of these four.

As a further example, while programming in the Logo Writer environment, children might select "Video Clips and Stills." A library of discs would be available in the classroom so that children could select whatever appropriate video clips or stills they needed for the particular fractional representation they were working on, or for any other purposes such as an opening scene or a feedback (a reward for a correct answer in their testing procedure, for example). During their processes of programming or of viewing visuals from the videodisc, they could also click on Designer's Cards, examine their pre-written plans for that day and their designs from the day before, or annotate the reasons for their having made certain changes and choices while developing their product.

Designer's Cards would allow children's on-line, day-by-day planning, screen designing, story boarding, note taking, or other written reflections, in the same way that the Designer's Notebook was used by the children during the Instructional Software Design Project. The advantage of having Designer's Cards on-line is that children would be able to take notes and reflect more freely **during** their process of designing, programming, and production. They would be able to more easily compare their original designs and plans (goals) with their outcomes (the screens they had implemented) and decide whether to revise the plan or the program's output.

Designer's Links would allow children various options. **1)** To create links in their computerized Designer's Notebook (i.e., their personal Designer's Cards). As we remember, Debbie would sometimes design a screen and not implement it for several days or weeks: Designer's Links would allow Debbie to keep better track, on-line, of her ideas, and to establish inter-relations between her various designs and plans. **2)** To create links between the various Logo or videodisc files and screens that had already been programmed. We saw how Debbie got confused when her program became long and complex: Designer's Links would allow her to keep better track of all the super-procedures or routines in which a particular procedure was used, or to create inter-relations between the various parts of her software and Logo Pages. **3)** To plan links between procedures even before they fully existed. We saw how Debbie sometimes created lists of "empty" procedures (i.e., To Show...To Test...To Again), which were her plans for a teaching unit, or parts of her software. On several occasions, a day or two after she had created these "empty" procedures, she would lose track of where these should fit within her whole project, and sometimes decide not to implement them: Designer's Links would allow Debbie to sketch connections and relations, and keep track of where these procedures should or could fit into, and how they might relate to her existing product.

During and after the programming of their software in Logo, a child, or a group of children, could place in the Rational-Number Sub-Constructs Menu all the rational numbers they had created representations for; in the Operations Window they could place the operations they had made representations for; in the Lessons Menu, the titles of the lessons or quizzes they had created; and in the Representations Window, the titles of the representations they had made for specific rational-number concepts, and/or operations, etc.

As a concrete example let us examine what Debbie could do by using a system of this kind. Debbie could list "1/2" in the Fractions Menu Window, list "Half-the-Screen," "Geometrical-Shapes," and "House-Scene," in the Representations Window; she could also list "Equivalence" in the Operations Menu, and her "Equivalence-Scene" in the Representation Window. In the Lessons Menu, she could list "Debbie's Fractions Project," that is, her whole piece of software, or choose to list each meaningful part of it separately, such as "On Halves," "The Addition Lesson," "The Exam," etc. She could also pre-select to put her "Introduction" in the Action Window, and use this space for giving instructions to the user on what to do.

This organization would probably help Debbie develop a better mental model of her product and its parts (i.e., its part-whole relationship). It could also encourage her thinking about the differences between various rational-number concepts, operations, representations, etc., and organize those in mini-lessons, quizzes, projects, etc. The programming process itself would remain similar to the one she went through during the Project. However, the organization of her finished or partially finished procedures and super-procedures would be much facilitated by the use of this system.

If we can imagine Debbie's software and its parts being organized in this way, we can also imagine in what ways Bibby, a third grader, might later use it. Bibby could choose to click on the "1/2" in the Fractions Menu; then click on the "House Scene" in the Representations Menu; then switch the "ON" button in the Action Window, causing Debbie's "House-Scene" to appear. Bibby could then switch this window to "OFF" and choose the title "Equivalence" in the Operation Menu. The fractions Debbie had designed Equivalencies for would now be highlighted in the Fractions Menu, and Bibby could choose to click on one of them; then she could go back to the Action Menu and study Debbie's representation for the "Equivalencies of Halves". Bibby could then choose to stop viewing Debbie's representations one by one, and instead, look at everything Debbie had created. She could click the Lessons Menu "ON", then "Debbie's Project," which would result in the Action Window's being blown up and filling the entire screen, and Debbie's whole piece of software being executed.

Debbie would be the one in charge of creating the items that appeared in the windows; she would also be in charge of establishing (programming) the links between items and windows. This way, if Debbie had not already created a representation for the addition of 1/5 and 1/3, for instance, when Bibby clicked on it, a message previously programmed by Debbie would appear in the Action Window, saying: "Sorry, I did not create a representation for this one. Please try Equivalence," or "Please try Aaron's Project, he created a representation for adding 1/5 and 1/3."

I have already given some of my reasons for integrating the Designer's Notebook on-line in the form of Designer's Cards, and Designer's Links. I shall now briefly explain why I recommend a Logo-based system, and why I wish to integrate it with a videodisc system.

<u>Why is it a Logo-based system</u>? The various reasons for this are related to the cognitive processes involved and to children's experiences in learning this programming language (e.g., see Chapter I, Section 2.3.1., and Chapter III, Section 2.1). What I wish to stress here again is the fact that my study revealed that children felt very comfortable programming in Logo, enjoyed it a great deal, and learned how to program quite rapidly. In my project, which was based on the learning of fractions, Logo was ideal for creating fractional representations and links or translations between them.

<u>Why integrate a videodisc into the system</u>? The reason for this, to my mind, is that many interesting representations could be created by children with <u>real-life visuals</u>, and by relating Logo graphics and text to them. In the case of fractions learning, this might involve children in thinking more about real-life rational-number situations. It could make Logo a more flexible tool, since it is quite difficult, in Logo, to program real-life images of people, animals, food, nature, shops, movement, etc.--all of which are easy to store on the visual data-base of a videodisc. Logo could also be used for highlighting various features on these pictures, creating interesting patterns, or superimposing text and graphics on the visual (or

with two separate monitors, in parallel with the visuals). The use of a videodisc could also bring into play the use of narrations and music in the children's products; with sound and narrations, children could explore many more rational-number representations related to words, poems, and musical rhythms.

Another reason for integrating videodisks is that videodisc developers and researchers have proved that these have great potential and are very powerful in the way they help children to learn through exploration and manipulation of visual material. It is not in the scope of this section to review these interesting and valuable findings from videodisc research. However, one problem I shall mention in that with this type of technology very expensive hardware and very complex, time-consuming, and expensive software development are required. Education, so far, has not been able to afford to spend much money on research and development of videodisc systems and software of this kind. To my mind, one way we could incorporate this sophisticated and valuable technology into education is by making it as generic as possible (like word processors or the Logo programming language, which can be incorporated into many domains). This would mean providing schools with a library of discs on many topics and concepts. Such discs are already widely available (e.g., from the National Geographic Society, NASA, Nova, etc), are rather inexpensive, and could be used just as they are as data-base for visuals. Children would be able to use these discs in their processes of programming (and computer programming already exist in many schools) once an appropriate interface between Logo and a videodisc player had been developed (several are under development already).

In this way, the many problems caused by videodisc-systems compatibility, the high expense and efforts of large teams of people required for development of videodisc-based educational material, the need for long and complex videodisc software development, etc.--would be solved. Children could benefit from this technology right now, and would find it quite simple to use it as raw material for their software construction and programming processes.

This brings us back to one of the points I made in the first chapter of this thesis: that children learn much more from designing interactive software than from using many of the pieces of instructional software that have been pre-developed for them. This point becomes even stronger when we analyze the existing educational videodiscs that are available for use in schools today. Most of them are "CAI with movies," very rich in terms of their visuals, but very limited for creative and productive learning because of their "drill-and-practice" modes of presentation. They were found to be rather useful for industrial training, museums, or even teacher's demonstrations and instruction, but they are not suitable for children's constructivist and individual explorations and inventions.

It should be noted here that several current research and development projects, using HyperCard, for Apple Macintosh, are beginning to make creative use of videodisc technology for constructive, productive, and child-centered education. But my emphasis on "letting children do it too" (i.e., letting them experience and learn from design, production, and programming of instructional systems that incorporate videodisc materials) remains unchanged.

These are only a few preliminary ideas, on a very superficial level, for creating a system for children's software development and production. It is based on my finding that the Experimental children learned a great deal from their processes of software development even through Logo only. I invite the reader to begin to imagine what these same children could learn from using a system integrating Logo with Designer's Cards, Designer's Links, and Video Clips and Stills (offered for their use through videodisc technology). It goes without saying that children's intensive use of a system of this kind could open up many new areas to researchers for investigating children's minds, cognitive development, problem-solving processes, and their processes and ability to learn several mutually supportive topics and skills at the same time.

# BIBLIOGRAPHY

Adelson, B., & Soloway, E. (1984). *A Cognitive Model of Software Design.* Cognition and Programming Project, Research Report no. 342. New Haven, CT: Yale University.

Adelson, B. & Soloway, E. (1984). *The Role of Domain Experience in Software Design.* Cognition and Programming Project, Research Report no. 25. New Haven, CT: Yale University. Also in IEEE Transactions on Software Engineering 11, 11 (1985)

Atwood, M. E., Jefferies, R., & Polson, P. G. (1980, March). *Studies in Plan Construction I and II.* Technical Report no SAI-80-028-DEN. Englewood, Colorado: Science Applications Inc.

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

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.

Bennett, A. B. (1981). *Fraction Bars: A Step By Step Teacher's Guide.* University of New Hampshire & Scott Resources Publishers.

Bransford, J. D. (1985). *Computer, Videodiscs, and the Teaching of Thinking.* Unpublished paper from the Vanderbuilt's Learning and Technology Center. Also presented at the American Educational Research Association, Washington, DC.

Brown, A. L. (1978). Knowing When, where, and how to remember: A Problem of Metacognition. In R. Glaser (Ed.), *Advances in Instructional Psychology.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Brown, A. L., Bransford, J. D., Ferrara, R. A., & Campione, J. C. (1983). Learning, Remembering, and Understanding. In W. Kessen (Ed.) *Handbook of Child Psychology: Cognitive Development, Vol. 3.* New York: Wiley.

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

Bruner, J. S. (1985). Vygotsky: A Historical and Conceptual Perspective. In Wertsch, J. V. (Ed.), *Culture, Communication, and Cognition.* Cambridge: Cambridge University Press.

Bruner, J. S. (1966). *Towards a Theory of Instruction.* New York: W. W. Norton.

Burton, R. R., Brown, J. S., & Fischer, G. (1984). Skiing as a Model of Instruction. In B. Rogoff, & J. Lave, *Everyday Cognition.* Cambridge, MA: Harvard University Press.

Carpenter, T. P., Coburn, T. G., Reys, R. E., & Wilson, J. W. (1976). Notes from the National Assessment: Addition and Multiplication with Fractions. *Arithmetic Teacher, 18(4).* 245-249.

Carver, S. M. (1986, December). *Transfer of LOGO Debugging Skill: Analysis, Instruction, and Assessment.* Ph.D. Thesis. Pittsburgh, PA: Carnegie-Mellon University, Department of Psychology.

Carver, S. M., & Klahr, D. (1986, February). Assessing Children's LOGO Debugging Skills With a Formal Model. Pitsburgh, PA: Carnegie -Mellon University, Department of Psychology. (In Press: *The Journal of Educational Computing Research).*

Case, R. (1985). *Intellectual Development: From Birth to Adulthood.* New York: Academic Press.

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

Clements, D. H. (1985, April). *Effects of Logo Programming on Cognition, Metacognitive Skills, and Acheivement.* Paper presented at the American Educational Research Association, Chicago, Illinois. (In Press).

Clements, D. H., & Gullo, D. F. (1984). Effects of Computer Programming on Young Children's Cognition. *Jurnal of Educational Psychology, 76(6).* 1051-1058.

Collins, A., & Brown, J. S. (1985, December). *The Computer as a Tool for Learning Through Reflection.* Paper presented at the American Educational Research Association, Washington, DC.

Collins, A. & Brown, J. S. (1985, December). *The New Apprenticeship.* Paper presented at the American Educational Research Association, Washington, DC.

Daiute, C. (1985). *Writing with Computers.* Addison-Wesley Publishing Company.

Davidson, P. S.(1977). *Idea Book for Quisenaire Rods at the Primary Level.* New Rochelle NY: Cuisenaire Company of America.

Davidson, J. (1969). *Using The Quisenaire Rods--A Photo-Text Guide for Teachers.* New Rochelle NY: Cuisenaire Company of America.

Dewey (1902). *The Child and the Curriculum.* In R. D. Archambault (Ed.), John Dewey on Education. Chicago: University of Chicago Press (1974).

DiSessa, A. (1985). Learning About Knowing. In *Children and Computers,* E, Klein (ed.), New Directions in Child Development 28, San Francisco: Jossey-Bass.

ETC (1985, November). Technical Report no. 85-21, written by Davidson, P. S., Dickenson, A. M., & Tierney, C. C. *"Pies are hard to find out about..." An Inquiry into children's understanding and nature of fractions.* Harvard Graduate School of Education, Cambridge: The Educational Technology Center.

ETC (1986, July). *Seeing The Unseen: The Science Videodisc Project.* Cambridge, MA: The New Technology Group, Educational Technology Center, Harvard Graduate School of Education.

Feurzeig, W., Horwitz, P., & Nickerson, R. S. (1981, October).*Microcomputers in Education.* Report no. 4798. Cambridge Ma: Bolt Beranek & Newman.

Feurzeig, W., Papert, S., Bloom, M., Grant, R., & Solomon, C. (1969). *Programming Languages as a Conceptual Framework for Teaching Mathematics.* Technical Report no. 1899. Cambridge MA: Massachusetts Institute of Technology and Bolt, Beranek, & Newman.

Flavell, J., & Draguns, J. (1957). A Microgenetic Approach to Perception and Thought. *Psychological Bulletin, 54* , pp. 197-217.

Flavell, J. (1979, October). Metacognition and Cognitive Monitoring: A New Area of Cognitive-Developmental Inquiry. *American Psychologist*, Vol 34 (10), pp. 906-911.

Flavell, J. H., & Wellman, H. M. (1977). Metamemory. In R. V. Keil, & J. W. Hagen (Eds.), *Perspectives on the Development of Memory and Cognition.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Ginsburg, H., & Opper, S. (1978). *Piaget's Theory of Intellectual Development.* Englewoog Cliffs, NJ: Prentice-Hall Inc.

Ginsburg, H., & Allardice, B. (1984). Children's Difficulties with School's Mathematics. In B. Rogoff, & J. Lave, *Everyday Cognition.* Cambridge, MA: Harvard University Press.

Gruber, H., & Voneche, J. J. (1977). *The Essential Piaget.* New York: Basic Books.

Guindon, R., & Curtis, B. (1988). *Control of Cognitive Processes During Software Design: What Tools Are Needed?* Autin, Texax: Microelectronics and Computer Technology Corporation.

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.

Harrison B. (1972). *Pattern Blocks Activities for Children.* Weston, MA.

Harvey, B. (1985). *Computer Science Logo Style. Volume 1: Intermediate Programming.* Cambridge MA: MIT Press.

Heller, R. S. (1986). *Different Logo Teaching Styles: Do They Really Matter?* Paper presented at the First Workshop of Empirical Studies of Programmers. Washington, DC.

Inhelder, B., Sinclair, H., & Bovet, M. (1974). *Learning and the Development of Cognition.* Cambridge, MA: Harvard University Press.

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: Lawrence Erlbaum Associates.

Kurland, D. M., & Pea, R. D. (1983). Children's Mental Models of Recursive Logo Programs. In *Proceedings of the Fifth Anual Cognitive Science Society.* Rochester, NY: Cognitive Science Society.

Lesh, R.,& Landau, M. (1983). *Acquisition of Mathematics Concepts and Processes.* Academic Press.

Lesser, J. S. (1974). *Children and Television: Lessons from Sesame Street.* New York: Vintage Books.

Linn, M. C. (1985). The Cognitive Consequences of Programming Instruction in the Classrooms. *Educational Researcher, 14.* 14-29.

Lowler, R. W. (1985). *Computer Experience and Cognitive Development: A Child Learning in a Computer Culture.* West Sussex, England: Ellis Horwood Limited.

Mandinach, E. B. (1984). *Classifying the A in CAI for Learners of Different Ablities.* Paper presented at the American Educational Research Association. New Orleans, Louisiana.

Markman, E.L. (1977). Realizing that You Don't Understand: A Preliminary Investigation. In *Child Development*, 48, pp. 986-992.

Meichenbaum, D., Burland, S., Gruson, L., & Cameron, R. (1985). Metacognitive Assessment. In *The Growth of Reflection in Children.*

Milojkovic, J. (1985). *Children Learning Computer Programming: Cognitive and Motivational Consequences.* Ph.D. Thesis. Palo Alto, CA: Stanford University.

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

Nachmias, R., Mioduser, D., Chen, D. (1985). *Acquisition of Basic Computer Programming Concepts by Children.* Technical Report no. 14. Tel Aviv, Israel: Center for Curriculum Research and Development, School of Education, Tel Aviv University.

Nickerson, R. S. (1986). *Using Computers: Human Factors in Information Systems.* Cambridge MA: MIT Press.

Nickerson, R. S., Perkins, D. N., & an Smith, E. E. (1985). *The Teaching of Thinking.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Newell, A. & Simon, H. A. (1972). *Human Problem Solving.* Englewood Cliffs, NJ: Prentice-Hall.

Papert, S. (1987). *Using Computers to Combat Illiteracy: Towards a Constructionist Theory of Creative Learning.* A Proposal to the MacArthur Foundation. Cambridge, MA: the Media Tech. Lab.

Papert, S. (1986). *Constructionism: A New Opportumity For Elementary Science Education.* A Proposal to the National Science Foundation. Cambridge, MA: the Media Technology Laboratory, MIT.

Papert, S. (1985, July). *Computer Criticism vs Technoratic Thinking.* Cambridge, MA: the Media Technology Laboratory, MIT. Also appeared in LOGO 85 Theoretical Papers. 53-67.

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

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

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

Papert, S., Watt, D., diSessa, A., & Weir, S. (1979). *Final Report of the Brookline Logo Project. Parts I and II.* LOGO MEMO no 53. Cambridge, MA: MIT.

Papert, S. (1971a). *Teaching Children Thinking.* AI Memo no 247, and Logo memo no 2. Cambridge, MA: MIT.

Papert, S. (1971b). *Teaching Children to be Matematicians vs. Teaching about Mathematics.* AI Memo no 249, and Logo Memo 4. Cambridge, MA: MIT.

Pea, R. D. (1988, In Press). Putting Knowledge To Use. In R. Nickerson, & P. Zodhiates (Eds.), *Technology in Education in 2020.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Pea, R. D. (1987). Cognitive Technologies for Mathematics Education. In A. Schoenfeld (Ed.), *Cognitive Science and Mathematics Education.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Pea, R. D. (1985, October). *Transfer of Thinking Skills: Issues for Software use and Design.* Paper presented at the National Conference on Computers and Complex Thinking. National Academy of Sciences, Washington, DC.

Pea, R. D. (1984). *Symbol Systems and Thinking Skills: Logo in Context.* Paper presented at the LOGO 84 Conference, Cambridge MA: MIT. Appeared in the LOGO 84 Pre-Proceedings Book. 55-61.

Pea, R. D., & Kurland, D. M.. (1983, May). *On the Cognitive Effects of Learning Computer Programming.* New York, Bank Street College: Center for Children and Technology.

Pea, R. D., & Kurland, D. M.. (1984). On the Cognitive Effects of Learning Computer Programming. *New Ideas In Psychology, 2(2).* 137-168.

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

Peck, D. M., & Jencks, S. M. (1981). Conceptual Issues in the Teaching and Learning of Fractions. *Jurnal for Research in Mathematics Education. 12(5).* 339-348.

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

Perkins, D. N. (1985). The Fingertip Effect: How Information-Processing Technology Changes Thinking. *Educational Researcher, 14(7).* 11-17.

Perkins, D. N. & Martin, F. (1985, October). *Fragile Knowledge and Neglected Strategies in Novice Programmers.* Technical Report no. 85-22. Cambridge MA: Educational Technology Center, Harvard Graduate School of Education.

Piaget, J. (1976). *The Grasp of Consciousness: Action and Concept in the Young Child.* Cambridge, MA: Harvard University Press.

Piaget, J. (1973). *The Child and Reality: Problems of Genetic Psychology.* New York: Grossman.

Piaget, J. (1972). Intellectual Evolution from adolescence to adulthood. *Human Development,* 15, 1-12.

Piaget, J., (1968). *Six Psychological Studies.* New York: Vintage Books.

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

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

Post, T. R., Wachsmuth, I., Lesh, R., & Behr, M. J. (1985). Order and Equivalence of Rational Numbers: A Cognitive Analysis. *Jornal on Research in Mathematics Education, 16(1).* 18-36.

Post, T. R. (1981). Results and Implications from the National Assessment. *Arithmetic Teacher, 28(9).* 26-31.

Resnik, M., & Ocko, S. (1986, December). *Lego/Logo and Science Education.* Cambridge, MA: MIT.

Resnick, M. (1987, December)*Lego/Logo: A learning Environment for Design.* Cambridge, MA: MIT.

Rogoff, B., & Lave, J. (1984). *Everyday Cognition : Its Development in Social Context.* Cambridge, MA: Harvard University Press.

Rogoff, B., & Wertsch, J. V. (Eds.)(1984).*Children's Learning in the "Zone of Proximal Development."* San Francisco: Joseey-Bass.

Sachter, J. (1987). *Children's Development of Spatial Understanding through the use of a 3-D Computer Graphics System: An Exploratory in Project Headlight.* Unbublished Paper. Cambridge, MA: MIT.

Salomon, G. (1979). *Interaction of Media, Cognition, and Learning.* San Francisco: Jossey Bass.

Salomon, G. (1986). *Information Technologies: What You See is Not Always What You Get.* Report no. 3. Tel Aviv University, Israel: Unit for Communication and Computer Research in Education.

Salomon, G., & Perkins, D. N. (1986, February). *Transfer of Cognitive Skills from Programming: When and How?* Technical Report no. 2. Tel Aviv, Israel: Unit for Communication and Computer Research in Education. Tel Aviv University.

Schneiderman, B., Mayer, R., McKay, D., & Heller, P. (1977). Experimental Investigations of the Utility of Detailed Flowcharts in Programming. *Communications of the Association for Computer Machinery (ACM), 20.* 373-381.

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

Schoenfeld, A. H. (1985). *Mathematical Problem Solving.* Academic Press.

Schoenfeld, A. H. (1985). *What's The Fuss About Metacognition?* (To appear in Cognitive Science and Mathematics Education. Hillsdale, NJ: Lawrence Erlbaum, Inc. 1987)

Siegler, R. S. (1983a). Five Generalizations about Cognitive Development. *American Psychologists* 38. 263-277.

Siegler, R. S. (1983b). Information Processing Approaches to Cognitive Development. In W. Kessen (Ed.) *Handbook of Child Psychology: History, Theory and Methods. Vol. 1.* New York: Wiley.

Siegler, R. S., & Klahr, D. (1982). When Do Children Learn: The Relationship Between Existing Knowledge and the Ability to Acquire New Knowledge. In R. Glaser, (Ed.), *Advances in Instructional Psychology*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Smith, J. (1987). *What is Fraction Conceptual Knowledge?* Paper presented at the annual meeting of the American Educational Research Association. Washington, DC.

Solomon, C. (1986). *Computer Environments for Children: A Reflection on Theories of Learning and Education.* Cambridge, MA: MIT Press.

Soloway, E., & Enhlich, K. (1984). *Empirical Studies of Programming Knowledge.* Cognition and Programming Knowledge Research Report no 16. New Haven, CT: Yale University.

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

Steinberg, R. E. (1984). *Teaching Computers to Teach.* (On creating CAI). Hillsdale NJ: Lawrence Erlbaum Associates.

Sternberg, R. J. (Ed), (1984). *Mechanisms of Cognitive Development.* New York: W. H. Freeman and Company.

Simon, H. A., & Chase, W. (1973). Skill in Chess. *American Scientist, 61.* 394-403.

Tierney, C. C. (1988). *Construction of Fraction Knowledge: Two Case Studies.* Unpublished Doctoral Thesis. Cambridge, MA: Harvard Graduate School of Education.

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

Vygotsky, L. S. (1962). *Thought and Language.* Cambridge, MA: MIT Press.

Vygotsky, L. S. (1978). *Mind In Society: The Development of Higher Psychological Processes.* Cambridge, MA: Harvard University Press.

Watt, D. (1979). *Final Report of the Brookline Logo Project. Part III: Profiles of Student's Work.* Logo memo no 54, A.I. Memo no 546. Cambridge MA: Artificial Intelligence LAB, MIT.

Wearn-Hilbert, D. C., & Hilbert, J. (1983). Junior High Students Understanding of Fractions. *School Science and Mathematics, 83(2).* 96-106.

Webb, N. M., Ender, P., & Lewis, S. (1986, June). Problem-Solving Strategies and Group Processes in Small Groups Learning Computer Programming. *American Educational Research Jurnal 23(2).* 243-261.

Weir, S. (1986). *Cultivating Minds: A Logo Case Book.* New York: Harper & Row.

Wertsch, J. V. (1985). Culture, Communication, and Cognition: Vygotskian Perspectives. Cambridge University Press.

White, S. H. (1978). Psychology In All Sorts of Places. In R. A. Kasschau & F. S. Kessel (Eds.), *Houston Symposium, Vol. 1. Psychology and Society: In Search of Symbiosis.* New York: Holt, Rinehart & Winston.

White, S. H. (1980). Cognitive Competence and Performance in Everyday Environments. *Bulletin of the Orton Society 30*. 29-45.

White, S. H., & Siegel, A. W. (1984). Cognitive Development in Time and Space. In B. Rogoff, & J. Lave (Eds.) *Everyday Cognition: Its Development in Social Context.* Cambridge, MA: Harvard University Press.

# APPENDICES

## APPENDIX A:

## PROJECT HEADLIGHT A MODEL SCHOOL OF THE FUTURE

An inner-city elementary public-school in one of Boston's low SES communities was the site of my study. One third of this public school, with children from first through fifth grade, of which approximately 40 percent were black, 40 percent Hispanic and 18 percent white or Asian, had been participating in Project Headlight (Papert, 1985,1986, 1987). As part of my work at Project Headlight, I implemented this Instructional Software Design Project, conducting my investigation in one fourth-grade classroom in this school, in the Project Headlight Area, during a four-month period. Earlier, during the Spring of 1986, a pilot study had been conducted with fifth graders of Project Headlight, the findings of which revealed the need for further investigations and a more systematic study of the Software Design Project. (See Harel, Children As Software Designers: An Exploratory Study in Project Headlight.1986, MIT).

The Instructional Software Design Project was <u>the project I created within MIT's Project Headlight</u>, and was strongly influenced by its educational philosophy, as well as an integral part of it. In the following paragraphs, I shall briefly describe Project Headlight and its educational goals.

During the 1985-86 academic year, a collaborative project involving Boston Public Schools, the MIT Media Technology Laboratory, and the IBM corporation laid the foundations for a <u>model school of the future</u>. Today (1988) we are in the third academic year of the running of Project Headlight, and it is now being funded by other organizations such as the National Science Foundation (NSF), the Apple Computer Inc., the Lego Company, the McArthur Foundation, and others. Project Headlight was originated and mainly inspired by Seymour Papert, Professor of Media Technology at the Massachusetts Institute of Technology in Cambridge. It was designed in anticipation of a near future in which technology would be used far more extensively than anything seen today in schools. It anticipated, for example, a system of free access to computers that might eventually involve two computers per student--one at school and one at home. In the model school, the number of computers introduced at the outset was one per three students; and it is to be hoped that this number will gradually increase as more money comes in, and as new ways are found to integrate the computers into the educational life of the school. A committee of

MIT scholars and school department officials selected this specific inner-city public school as the site for Project Headlight, after inviting proposals from all of Boston's Public Elementary Schools.

Although the project uses technology extensively, it is not defined as a "technology project," but rather as an <u>education project</u>. It explores new approaches toward learning and teaching in the context of a school that is rich in technology. Its educational goal is open classrooms, centered around students, integrating the learning of several subjects, and using computers as tools for learning these subjects.

Project Headlight currently operates as a "school-within-a-school," comprising about <u>one-third</u> of the School's students and teachers. The two hundred and fifty participating students are in grades one through five, and are divided into Advanced Work Classes, Regular Classes, Bilingual Classes, and Special Education Classes. The participating teachers learn about computers and the Project's educational methodology during several three-week-long summer workshops, which, since the summer of 1985, have been repeated each year at progressively more advanced levels and according to the teachers' needs and interests. In addition, there are regular in-service meetings once every week or two weeks. Operational decisions are also made in these regular meetings, which are attended by the school's teachers and by the principal of the school, as well as by researchers from the MIT Media Lab. Among the sixteen participating teachers, five teach Bilingual Classes in Spanish, three (including two of the bilingual teachers) are Special Education teachers, four teach Advanced Work Classes in English, and five teach Regular Classes in English. The Project uses approximately <u>120 IBM PCjr computers</u> in classrooms, in teachers' homes, and in the open areas next to the classrooms (see next page for Project Headlight's floor plan). Most of the computers are connected to each other in a local-area network.

The MIT staff participates in several ways. They support and train the teachers, are responsible for the instruction of teachers and students in computer skills such as Logo, word processing, using the network, etc. They teach some special courses, conduct a number of projects within several classes (the Instructional Software Design Project was one of these), and work towards integrating the computer into the basic curriculum. Finally, MIT researchers conduct an extensive program of observations and documentation of the many projects that have been implemented at the school, and of the teachers' and students' progress.

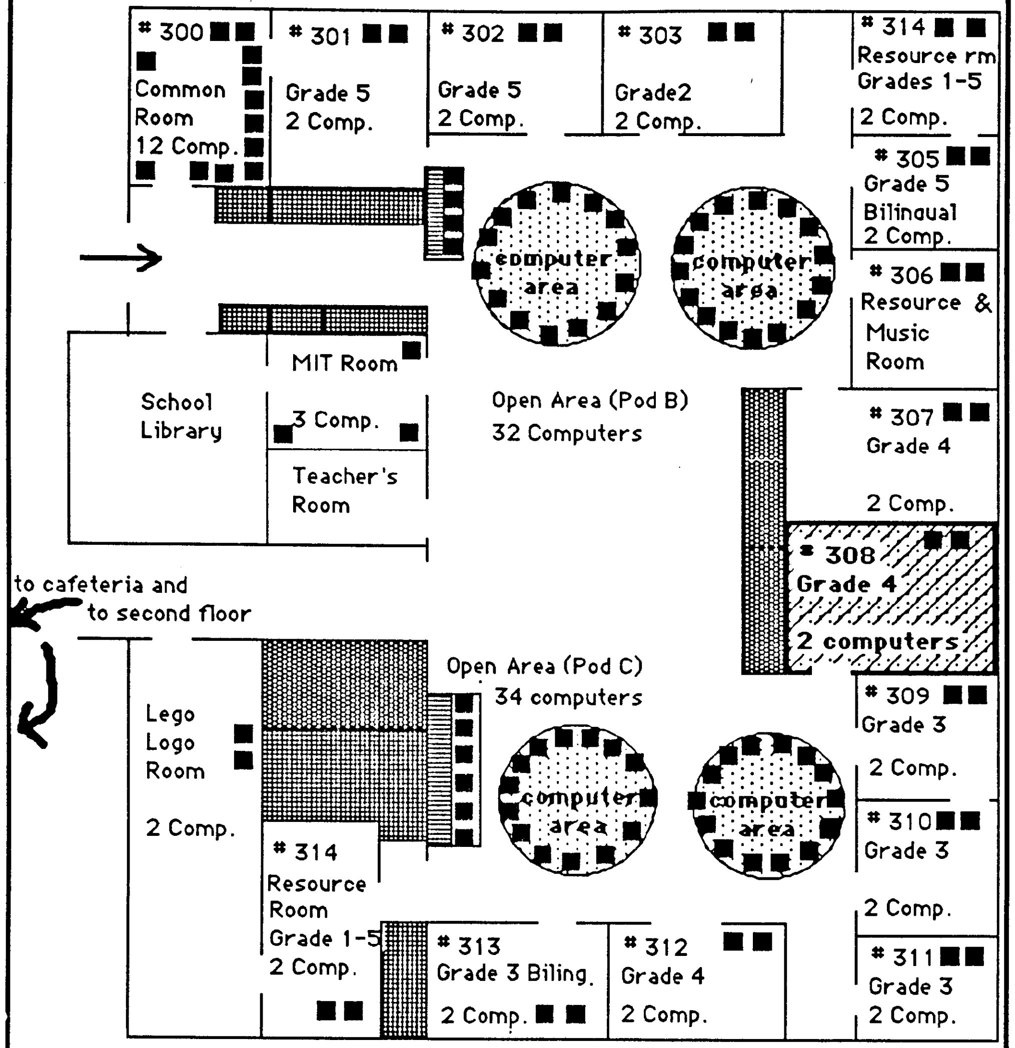
On the whole, the goals for Project Headlight's first year were to create its technological and conceptual infra-structure, build a team relationship among the various participants, and develop projects and systematic methods of observations. The goals of the second and third years were similar, and many more projects integrating the computer into the basic curriculum were implemented by the teachers and MIT staff. However, the MIT staff concentrated far more than previously on creating among teachers and students a degree of mastery over computers and their use that might support the growth, over a certain period of time, of an educational environment rich in computers.

The diagram on the following page shows Project Headlight's floor plan and the location and number of the computers. Other first and second-grade classrooms (besides the ones indicated here) participate in the project even though they are located on other floors of the school, and are not shown in the floor plan, which represents the center of Project Headlight.

**For more information and publications about Project Headlight, please contact The Epistemology and Learning Group, The Media Technology Laboratory. MIT. 20 Ames St. Cambridge, MA. 02139.**

## Project Headlight: Floor Plan & no. of Computers

(As of 1986-1987 School-Year)



▨ The Experimental Class: Instructional Software Design Project

▤ ◌ Tables for the computers in circles or along the walls.

■ ■ ■ Each black square represents one IBM PCjr computer.

▦ Bathrooms, or other non-used spaces

Total Number of Computers in Project Headlight = 110 Computers
(15 other computers are in teachers' homes).

## APPENDIX A.1:

## A DESCRIPTION OF THE INSTRUCTIONAL SOFTWARE DESIGN PROJECT

**Seventeen fourth-grade children placed beside one another worked on a common Project, one hour a day for four months:**

In the context of the Instructional Software Design Project, 17 fourth-grade children (all in one class) each designed, programmed (using the new LogoWriter programming language), and evaluated a collection of interactive screens (an interactive lesson) to teach third graders about basic concepts of fractions. Each child designed and programmed his piece of software for one hour each day, for approximately four months. On the whole, the children spent close to 70 hours each in this problem-solving enterprise, designing and working on implementing their software.

**The Instructional Software Design Project was open-ended but included a series of rituals and activities:**

The Project was <u>open-ended</u> but included a series of activities or routines that all the Experimental children followed. Each working day, for 5-7 minutes, the children wrote their plans and drew their designs in their personal Designer's Notebooks before going to the computer. (The Designer's Notebook is a notebook I created especially for this Project and will be described in detail later; see Appendix C for a sample.) Then, after completing their writing and drawing of plans and designs, the children worked at their own computers for approximately 45 to 55 minutes. At the computer, the children implemented their plans and designs, created new ones, and revised old ones. When they wished, they were allowed to work with their friends, help each other, or walk around to see what other children were doing. When the computer time was over, each child saved his (or her) daily files on a special diskette and went back to the classroom. In his Designer's Notebook, he then wrote and reflected on the problems and changes he had made that day, and sometimes added plans and designs for the next. The children had full freedom in choosing which concepts they wanted to teach, how to design their screens, what the sequence of their lesson should be, and what testing to include, if at all. In short, the Project was open-ended in terms of what the children 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 the time limitations in using the computer, was to allow the Project to fit into the schedule of the class and of the school. This requirement made it possible to estimate and draw generalizations about what children could accomplish in a project of this kind, designed to fit easily into the regular scheduling of any class or school in the future.

**Each child used his personal Designer's Notebook for designing and drawing screens, story-boarding, planning, and reflecting on changes and problems:**

The first requirement, the writing in the Designer's Notebook, was very important for the Project and for children's cognitive development in general. We did not tell the children what to write, how much to write, what or how to plan or draw, or how to reflect or make changes. However, we explained to them that, like professional software designers, they would enjoy keeping track of their ideas and changes, since this could help their implementation, their concentration, their not losing good ideas from one day to the next, and so on. If either of these two limitations became a problem for these children, it was the time requirement at the computer. The children always seemed to be frustrated about having to stop work and leave their computers, and often requested more time. In addition, the writing in the Designer's Notebook was a problem at first, since the children were not accustomed to handling in writing such a routine of planning, note-taking, and reflecting.

Let us consider for a moment the rituals and skills involved in children's using the Designer's Notebook in the process of designing and programming their software. This routine was essential to the Project, and its importance was clear to the teacher as well as to me. The rationale behind teaching children to plan, reflect, and take notes had been documented in several theories and rigorous experiments reported in educational psychology literature; more specifically, in the literature on learning metacognitive, cognitive control, and other related thinking skills (Nickerson et al., 1985; Chipman et al., vol 1 & 2, 1985; Brown A. L. et al., 1983). In none of these experiments, however, were the children required to plan and reflect in writing every day for as long as a period of time as they were in this Project (which went on every day for approximately four months). It was therefore expected that, in this context, new insights would be revealed about children's development and abilities in acquiring these executive processes and cognitive control skills.

Writing in their Designer Notebooks during the course of the Instructional Software Design Project was clearly a process that the children had to "get used to." After approximately ten days into the Project, the children realized its importance and made it their own. The length and structure of this Project, the fact that the children were in charge of their own learning throughout the Project, and the complete integration of these cognitive skills into other kinds of learning, were the crucial factors in encouraging these skills among children (rather than implementing a direct or explicit instruction of them). The Designer's Notebook became a personal and important tool for the children, and made them use these skills extensively and grow aware of the benefits of keeping track of their own planning, note-taking, and changes. Moreover, they realized that they did not need to implement what they wrote unless they wished to; they realized that the Notebook facilitated their thinking of "new" ideas while still implementing "old" ones; that going back and forth in their Notebooks to "old" drawings and notes was beneficial to them and very useful in their programming processes.

**Several "Focus Sessions" about software design, Logo programming, and fractions representing were conducted in the classroom during the Project:**

Several short classroom discussions (10 to 15 minutes each) were conducted during the period of the Instructional Software Design Project. In the first one, I briefly introduced, and discussed with the children, the concept of instructional design and educational software. Together, we defined the meaning and purpose of educational or instructional software, and briefly discussed a few pieces of software that the children were familiar with. I showed the children my own designs, plans, flowcharts, and screens from various projects that I had worked on in the past. I also passed among the children the book *Programmers At Work* (1987) and asked them to look at the notes, pieces of programs, and the designs of "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 children also received their personal diskettes and their Designer's Notebooks, and we discussed the ways in which they should and could be used during the Project.

During the other Focus Sessions we concentrated on issues such as the difficulties of specific fraction concepts and the children's ideas 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 OR REPRESENTATIONS COULD BE DESIGNED TO EXPLAIN THESE DIFFICULT CONCEPTS?" We asked the children to generate ideas for both posters simultaneously. After long lists of ideas were created, the children could copy all of the listed ideas, or parts of them, into their Designer's Notebook; or, if they wished, they could use these ideas as inspiration for generating new ones. Other discussions focused on specific Logo programming skills. For example, in some of these short "focused sessions" about programming, the teacher, the researcher, or one of the children, could stand next to one of the computers that were in the classroom, in front of the whole class or a group of children, and explain about how to use REPEAT, IFELSE, variables, etc. Again, the children could take notes on such concepts and programming routines in their Designer's Notebooks, or go directly to their computers and write a procedure that included that new programming skill or concept.

**The teacher and the researcher collaborated and actively participated in all the children's software design and programming sessions during the Project:**

The teacher and I collaborated, and were present during <u>all</u> the Software Design Project working hours. We walked around the children, sat next to them, looked at their programs, helped them when needed, and discussed with them their designs, programming, and problems in a friendly and informal way. In general, we had no specific plans for the Project's sequence, or for our presentations and focus discussions; rather, they were initiated by the teacher or by me at times when they were relevant to the children's work or problems, or at the children's request. (More information about the Project's learning atmosphere, objectives, and procedure, will be described later in this thesis. See also Harel's Pilot Study, "Children As Software Designers: An Exploratory Study in Project Headlight," MIT, July, 1986).

**Seventeen personal software-design-portfolios were created:**

Finally, these daily activities resulted in seventeen products (i.e., seventeen different pieces of instrictional software about fractions), and seventeen personal portfolios--one for each Experimental child--consisting of their daily sets of written plans, designs, the pieces of Logo code they programmed, and their written reflections on problems and changes for each working day.

# APPENDIX B:

# SAMPLES OF PRE-TESTS AND POST-TESTS

## LOGO PENCIL-&-PAPER TESTS

NAME:____________________ROOM:__________DATE:______________

1. Please list all the Logo instructions and commands that <u>you</u> know and use--in column A;
then, write an explanation and give an example for each one--in column B."

| Column A: Logo Commands | Column B: Definition, and example for each command |
|---|---|
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |

Frst Name: ——— (Page 12)

| Column A: | Column B |
|---|---|
| Logo commands | definition, and example for each command |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |

**NAME:**________________

**2. Look at your list of Logo instructions and commands, and please <u>classify, group, or categorize</u> the list. Give a <u>title, a label, a description to each one</u> of your groups.**

**NAME:**__________________

**3. CAN YOU DRAW THE FOLLOWING?**

**Logo Code:**

```
TO DESIGN
lt 90
fd 20
fd 20
rt 90
fd 40
rt 90
fd 40
rt 90
fd 40
rt 180
fd 20
rt 90
fd 10
fd 10
rt 90
fd 20
rt 90
fd 20
bk 20
rt 45
rt 45
fd 10
rt 90
fd 10
rt 90
fd 10
rt 40
rt 50
fd 10
END
```

**Draw It Here:**

Logo Activity.    Page 4

**NAME:**__________________

**4**. AFTER YOU FINISHED YOUR DRAWING (Question 3),
TRY TO **SIMPLIFY** THIS PROGRAM OR "CLEAN IT."
IN OTHER WORDS, THINK ABOUT OTHER WAYS TO WRITE A
LOGO PROGRAM THAT WILL PRODUCE OR MAKE THE **SAME** **DRAWING**.
ARE THERE LOGO COMMANDS THAT WILL MAKE THIS PROGRAM **SHORTER**?
(ALSO, THINK ABOUT CREATING SUB-PROCEDURES.)"

**NAME:**__________________

5. PLEASE ANSWER THE FOLLOWING QUESTIONS:
(IF YOU DO NOT KNOW HOW TO ANSWER THE FOLLOWING QUESTIONS, PLEASE WRITE "I DON'T KNOW."

- HOW MANY INPUTS DOES FD TAKE?______
  GIVE AN EXAMPLE:________________

- HOW MANY INPUTS DOES BK TAKE?______
  GIVE AN EXAMPLE:________________

- HOW MANY INPUTS DOES SETC TAKE?______
  GIVE AN EXAMPLE:________________

- HOW MANY INPUTS DOES HOME TAKE?______
  GIVE AN EXAMPLE:________________

- HOW MANY INPUTS DOES REPEAT TAKE?______
  GIVE AN EXAMPLE:________________

- HOW MANY INPUTS DOES SETPOS TAKE?______
  GIVE AN EXAMPLE:________________

- HOW MANY INPUTS DOES LT TAKE?______
  GIVE AN EXAMPLE:________________

- HOW MANY INPUTS DOES **CG** TAKE?______
  GIVE AN EXAMPLE:________________

NAME:________________

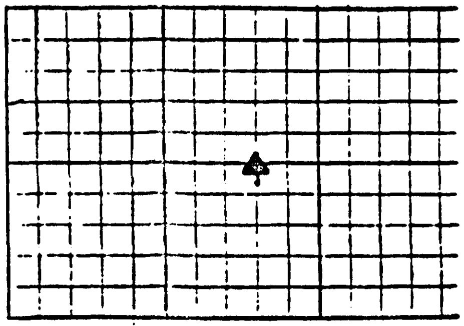
6. FILL IN THE BLANKS:

```
1) TO CIRCLE
     REPEAT 360 [FD 1 rt 1]
     END

   - CAN YOU MAKE A BIGGER CIRCLE? (fill in the blanks)
      TO BIGGER.CIRCLE
      REPEAT___ [FD___ RT___]
      END

1.1) CAN YOU MAKE A HALF CIRCLE? (Fill in the blanks)
       TO HALF.CIRCLE
        REPEAT___ [FD___ RT___]
        END

2) TO SQUARE
     REPEAT 4 [FD 50 RT 90]
    END

-CAN YOU MAKE A RECTANGLE? (Fill in the Blanks)
    TO RECTANGLE
     REPEAT ___ [______________________________]
     END
```

name ____________ Page 4.

7. Directions: Use the marked areas to the right of each problem to draw the programs output. (Please indicate which direction the turtle is facing everytime there is a change in direct

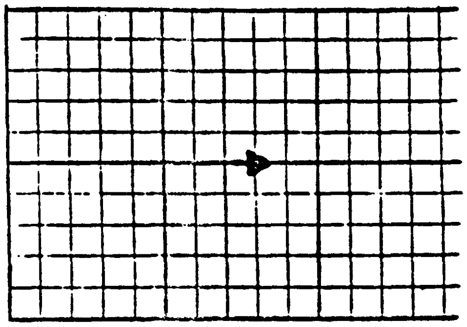
1) RT 90 FD 5



How would you rate this problem?

| Easy | | | | Hard |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |

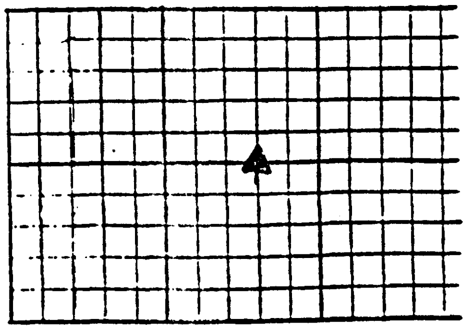
2) FD 3 LT 90 FD 2



How would you rate this problem?

| Easy | | | | Hard |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |

3) BK 2 RT 90 FD 4

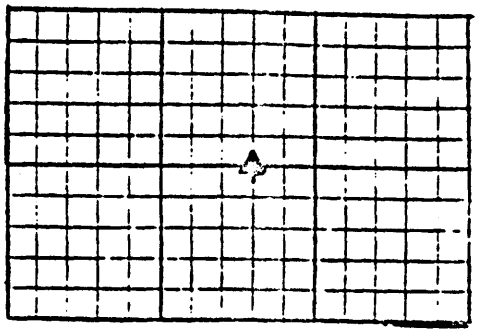


How would you rate this problem?

| Easy | | | | Hard |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |

name ———— page 8

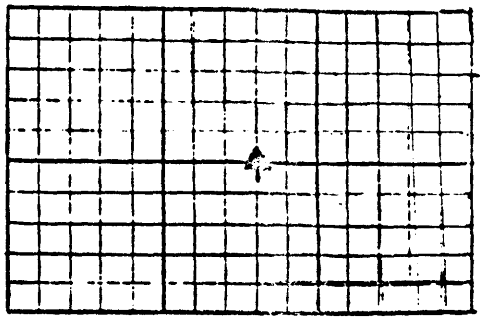
4) RT 180 FD 2



How would you rate this problem?

Easy 1 2 3 4 5 Hard

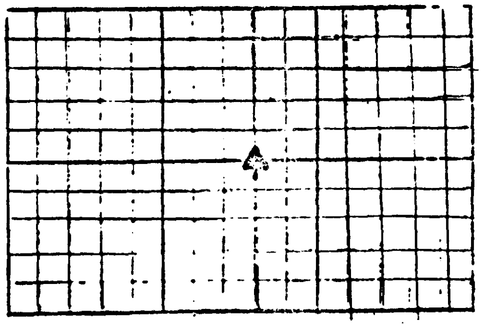
5) REPEAT 2 [FD 3 RT 90] FD 2



How would you rate this problem?

Easy 1 2 3 4 5 Hard

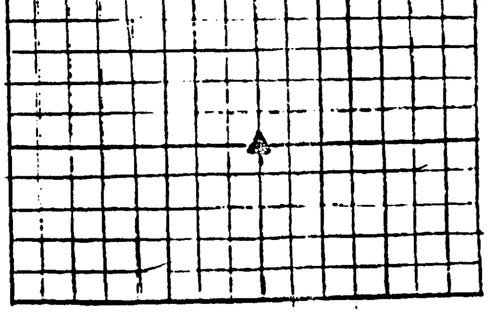
6) TO TRIANGLE
FD 2 RT 90 FD 3
END
REPEAT 2 [TRIANGLE]



How would you rate this problem?

Easy 1 2 3 4 5 Hard

7) TO BOX :SIZE
REPEAT 2 [FD :SIZE RT 90]
END
BOX 3



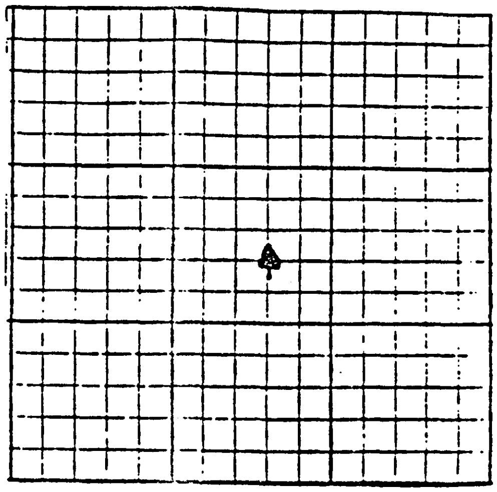
How would you rate this problem?

Easy 1 2 3 4 5 Hard

Page 9

Name: ____________

```
8) To Box
   RT 90 FD 1 RT 90 FD 1 RT 90 FD 1
   END

   TO SIDES
   FD 1 RT 90 FD 3
   END

   TO SHAPE
   SIDES
   BOX
   SIDES
   END
   ______________________
   SHAPE
```

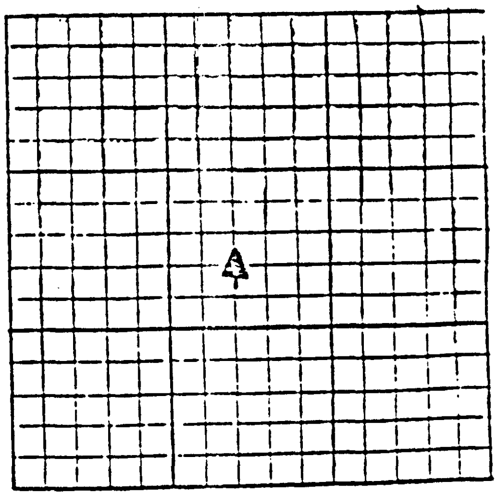


How would you rate this problem?

```
Easy            Hard
 1    2    3    4    5
```

---

```
12) TO BOX
    FD 2 RT 90 FD 2
    END

    TO CIRCLE :SHAPE
    IF :SHAPE > 0 [BOX]
    IF :SHAPE = 0 [STOP]
    CIRCLE :SHAPE-1
    END
    ______________________
    CIRCLE 2
```



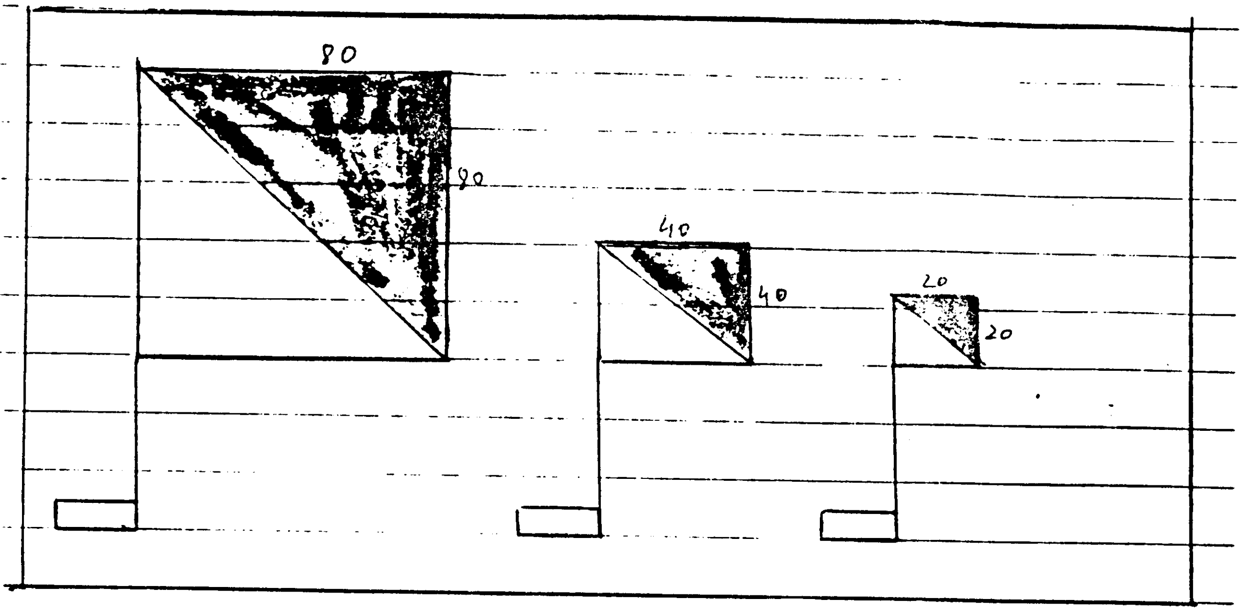
How would you rate this problem?

```
Easy            Hard
 1    2    3    4    5
```

Name: ________ p. 10

8. optional: (if you have time...)

Write a program to draw this:



(Page 1

Name John Batuyios Room 307 Date 21/1/87

1) Please list all the LOGO instructions and commands that you know and use in column A. Then, write an example of how you use each one on column B.

| Column A: Logo instructions, command | column B: examples for each instruction or comma |
|---|---|
| fd (certain amout) | moves turtle forward |
| bk (certain amout) | moves turtle backward |
| rt (certain amout) | turns turtle right |
| lt (certain amout) | turns turtle left |
| setpos (no. of pos ) | brings turtle somewher |
| home | brings turtle to normal po |
| ESC | bring you to contents |
| getpage "(name) | brings you to page |
| F 9 | makes turtle move with |
| | arrow keys |
| rg | clears screen |
| cg | clears graphics |
| cc | clears command centre |
| ct | clears text |
| ESC in F 9 | puts you to cursor |

[illegible] John (Page

| Column A: Logo instructions | Column B: Example for each Logo instruction |
|---|---|
| fill | fills object or scre |
| pu | puts pen up |
| pd | puts pen down |
| repeat (no)(commands) | repeat commands |
| wait (no) | waits in procedure |

First Name: (Page 3

2) Now, look at your list of Logo instructions and Commands, and please classify/group/categorize the list.
give a title, label, or description to each one of your categories or groups.

| Turtle Commands | Turn Turtle | Moves Turtle |
|---|---|---|
| fd | rt | fd |
| bk | lt | bk |
| pd | seth | setpos |
| pu | | |
| fill | | |
| wait | | |
| setpos | | |
| lt | | |
| rt | | |
| seth | | |

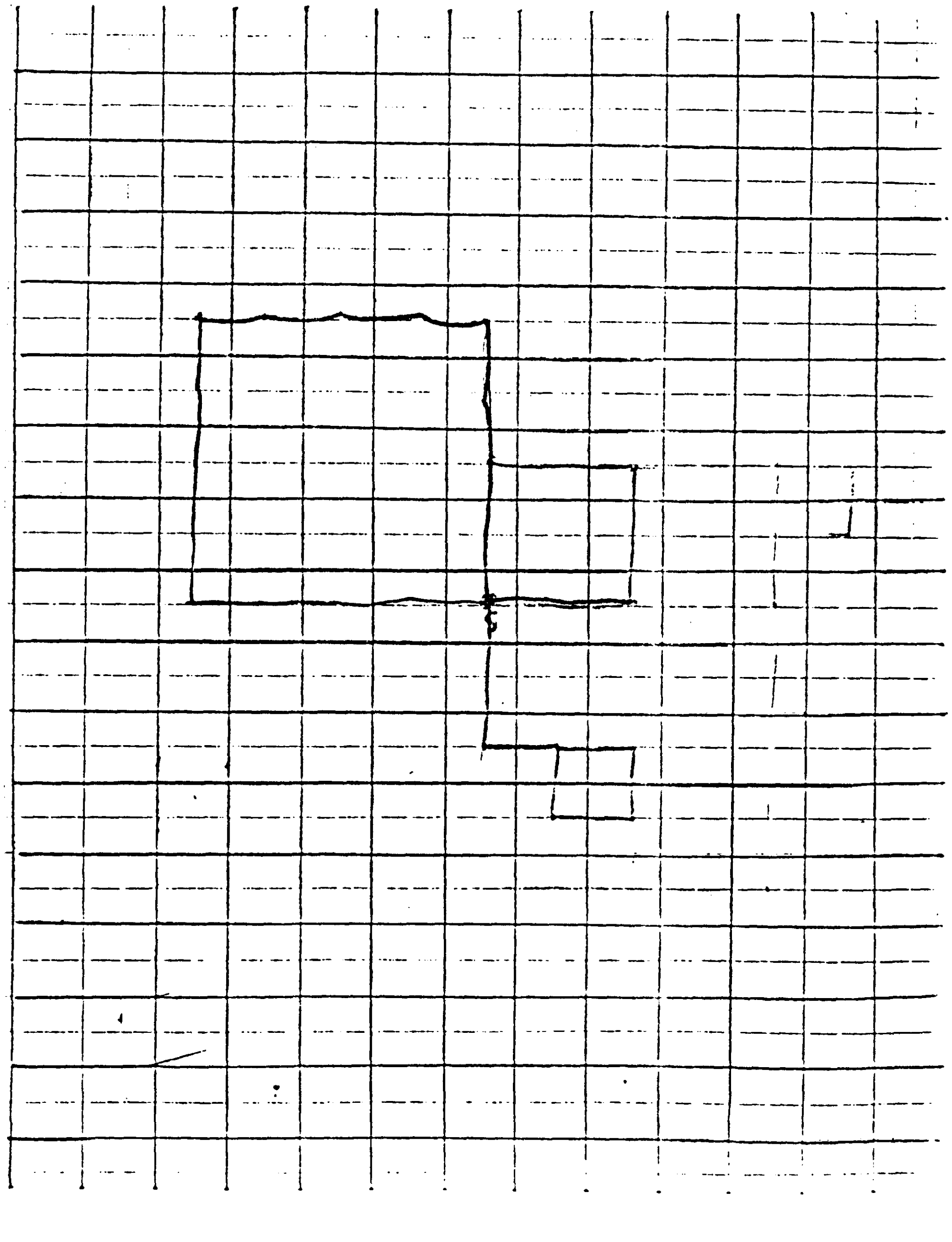
First Name: ______ Page 4

3) Can you draw the Following:

TO DESIGN

```
LT 90
FD 20
FD 20
RT 90
FD 40
RT 90
FD 40
RT 90
FD 40
RT 180
FD 20
RT 90
FD 10
FD 10
RT 90
FD 20
RT 90
FD 20
BK 20
RT 45
RT 45
FD 10
RT 90
FD 10
RT 90
FD 10
RT 40
RT 50
FD 10
END
```

DRAW IT HERE:



First Name: ______

(Page

4) After you finish the drawing, try to Simplify this program, or "clean it."

In other words, think about other ways to write a Logo program that will produce or make the same drawing.

are there Logo commands that will make this program shorter?
think about creating subprocedures.

```
TO DESIGN
LT 90 FD 40 repeat 3 [RT 90 FD 40]
rt 180 FD 20 repeat 3 [RT 90 FD 20]
bk 20 repeat 4 [RT 90 FD 10]
```

[illegible] Name: [illegible]

ote: If you don't know how to answer the following questions, write: "don't know"

5). How many inputs FD takes? 1
give an example: FD 50

- How many inputs BK takes? 1
example: BK 50
- How many inputs CLEARSCREEN takes? 0
example: CS
- How many inputs SETCOLOR takes? 1
example: SETC 1
- How many inputs HOME takes? 0
example: HOME
- How many inputs REPEAT takes? 2
example: REPEAT 3 [ ]
- How many inputs SETPOSITION takes? 1
example: SETPOS [ ]
- How many inputs LT takes? 1
example: LT 90

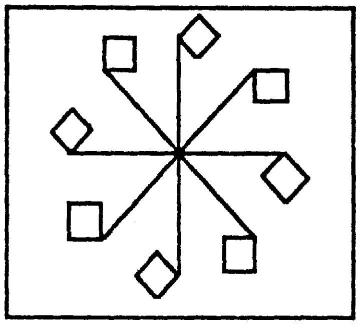
## **LOGO COMPUTER TESTS**

LOGO COMPUTER ACTIVITY. PAGE 1

NAME:____________________

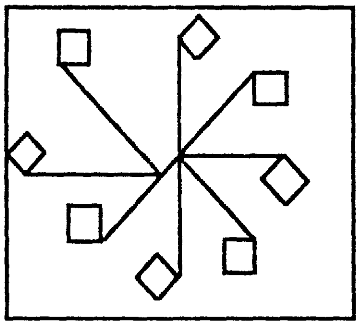
ROOM:__________DATE:________

1) THIS IS WHAT I WANTED TO DRAW ON THE COMPUTER;
THE NAME OF IT IS "FLAGS."



THIS WAS MY GOAL...

I THINK I HAVE 2 BUGS, AND THIS IS WHAT APPEARED ON THE COMPUTER SCREEN:



RUN THE PROGRAM "FLAGS (WHICH IS ON YOUR DISKETTE)

FIND THE 2 BUGS

FIX THE PROGRAM

DO IT UNTIL YOU GET IT TO LOOK LIKE THE TOP PICTURE.

LOGO COMPUTER ACTIVITY. PAGE 2

**NAME:____________________**

**2)THIS IS THE PROGRAM WITH THE TWO BUGS.**

**PLEASE MARK YOUR CORRECTIONS HERE:**

```
TO FLAGS
ht home
fd 45
rt 45
fd 10
fd 10
rt 90
fd 10
rt 90
fd 10
rt 90
lt 45
bk 45
rt 45

fd 30 fd 15 rt 45
fd 10 rt 90
fd 10 rt 90
fd 10 rt 90
fd 10 rt 90
lt 45 bk 30 bk 15
rt 45
fd 12 fd 12 fd 21
rt 40 rt 5
fd 10 rt 90 fd 10 rt 90
fd 10 rt 90 fd 10 rt 90
lt 45 bk 45 rt 45
fd 40 fd 5 rt 45
repeat 1 [fd 10 rt 90 fd 10 rt 90
fd 10 rt 90 fd 10 rt 90]
lt 45 rt 45 bk 45

fd 45 rt 45
fd 10 rt 90
fd 10 rt 90
```

LOGO COMPUTER ACTIVITY. PAGE 3

**NAME:**____________________

```
fd 10 rt 90
fd 10 rt 90
lt 45 bk 45 rt 30 rt 15

fd 45 rt 45
repeat 2 [ fd 10 rt 90
fd 10 rt 90]
lt 45 bk 30 rt 30 rt 10 rt 5

fd 65 rt 45
fd 10 rt 90
fd 10 rt 90
fd 10 rt 90
fd 10 rt 90
lt 45 bk 65 rt 45
fd 65 rt 45
fd 10 rt 90
fd 10 rt 90
fd 10 rt 90
fd 10 rt 90
lt 45 bk 65 rt 45
END
```

**NAME:**____________________

**3)** NOW, TRY TO **SIMPLIFY** THIS PROGRAM, OR "CLEAN IT."

IN OTHER WORDS, THINK ABOUT **OTHER WAYS** TO WRITE A LOGO PROGRAM

THAT WILL PRODUCE THE SAME PICTURE. CAN YOU MAKE IT **SHORTER?**

(THINK ABOUT CREATING SUB-PROCEDURES...)

WRITE YOUR SHORTER VERSION HERE:

AND ON THE COMPUTER, CALL IT **"FLAGS1"**

TO FLAGS1

**NAME:**____________________

**4)** NOW, TRY TO WRITE THE SAME PROGRAM WITH **INPUTS** (IF YOU DID NOT DO IT ALREADY),

SO YOU CAN TYPE: **FLAGS 45 10** AND IT WILL DRAW THE PICTURE OF 8 FLAGS AS BEFORE.

OR IF YOU TYPE: **FLAGS 65 20** IT WILL MAKE THE SAME PICTURE BUT BIGGER.

WRITE YOUR PROGRAM HERE:

AND ON THE COMPUTER CALL IT "**FLAGS2**"

LOGO COMPUTER ACTIVITY. PAGE 6

**NAME:**__________________

**5) CAN YOU MAKE A PROGRAM THAT DRAWS THE SAME PICTURE, BUT EACH FLAG WILL BE IN A DIFFERENT COLOR?**

WRITE YOUR PROGRAM HERE:

AND ON THE COMPUTER CALL IT "**FLAGS3**"

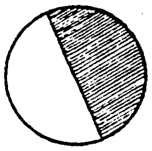
## <u>RATIONAL-NUMBER PENCIL-&-PAPER TESTS</u>

Pre - 75 question (I)
Post
65 questions

• name: ________________
• ROOM: ______

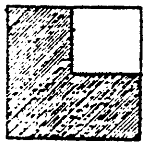
Directions: Read each question and set of answers carefully. Select the choice that you think answers the question. Mark the appropriate space on your answer sheet. If a question does not have the correct answer given, mark space "e" on your answer sheet. If you do not know how to do a problem, leave the answer space blank.

1. What fraction of this circle is shaded?



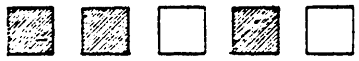
a. 2    b. $\frac{1}{2}$    c. 1    d. $\frac{1}{4}$    e. not given

2. What fraction of this picture is shaded?



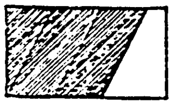
a. four-thirds    b. three-fourths    c. one-third
d. one-half    e. not given

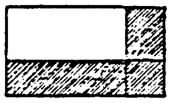
3. What fraction of this picture is shaded?

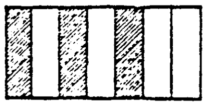


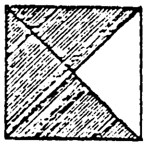
a. $\frac{3}{2}$    b. $\frac{5}{2}$    c. $\frac{2}{5}$    d. $\frac{3}{5}$    e. not given

4. Which picture shows three-fourths shaded?

a. 

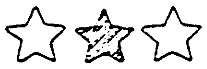
b. 

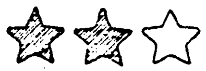
c. 

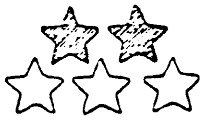
d. 

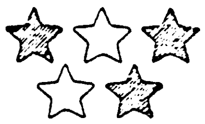
e. not given

5. Which picture shows two-thirds shaded?

a. 

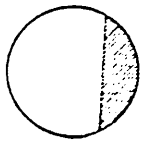
b. 

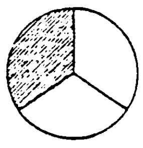
c. 

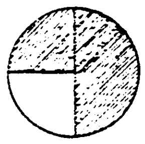
d. 

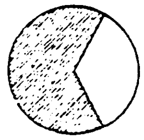
e. not given

6. Which picture shows $\frac{1}{2}$ shaded?

a. 

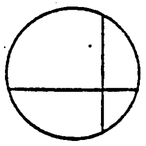
b. 

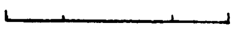
c. 

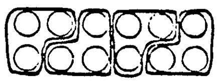
d. 

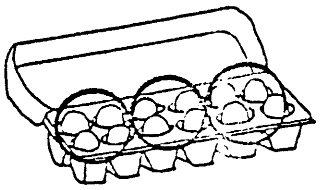
e. not given

7. Which picture shows fourths?

a. 

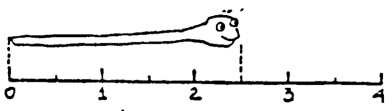
b. 

c. 

d. 

e. not given

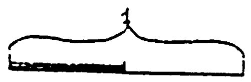
8. How long is the snake?

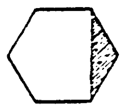


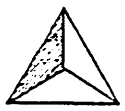
a. two  b. two and one-half  c. three

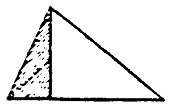
d. three and a half  e. not given

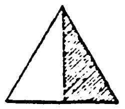
9. Which picture shows the same fraction as the shaded part of this line segment?



a. 

b. 

c. 

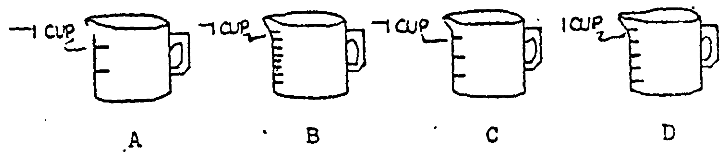
d. 

e. not given

10. Which fraction says "three-fourths"?

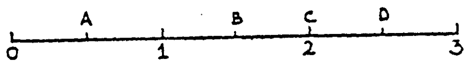
a. 34 b. $\frac{3}{4}$ c. $3\frac{1}{4}$ d. $\frac{4}{3}$ e. not given

11. Which container measures cups in thirds?



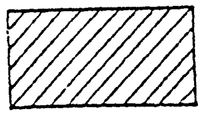
a. A b. B c. C d. D e. not given

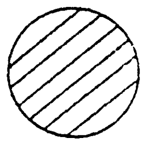
12. Which letter is above the point $\frac{1}{2}$?

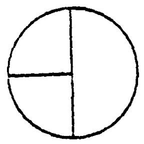


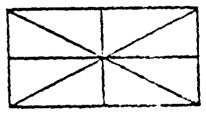
a. A b. B c. C d. D e. not given

13. Which picture is divided into equal size parts?

a. 

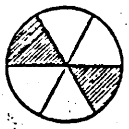
b. 

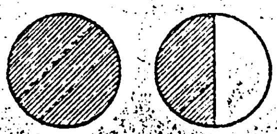
c. 

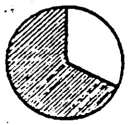
d. 

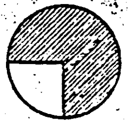
e. not given

14. Which picture shows $\frac{2}{3}$ shaded?

a. 

b. 

c. 

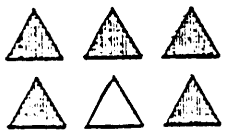
d. 

e. not given

15. How many halves equal one whole?

a. $\frac{2}{2}$ b. 2 c. $1\frac{1}{2}$ d. 1 e. not given

16. What fraction of the set of triangles is shaded?



a. $\frac{3}{2}$ b. $\frac{5}{2}$ c. $\frac{2}{5}$ d. $\frac{3}{5}$ e. not given

17. Which fraction says "two-fourths"?

a. 6 b. 24% c. 8 d. $\frac{2}{4}$ e. not given

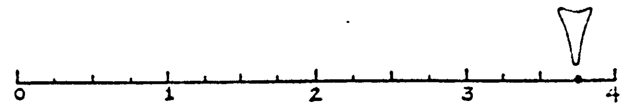
18. Which fraction says $\frac{2}{3}$?

a. five b. six c. two-thirds

d. twenty-three percent e. not given

---

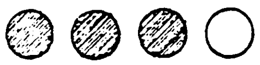
19. Which number goes with the point?

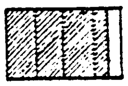
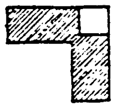


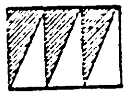
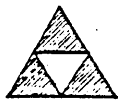
a. $\frac{14}{4}$ b. $3\frac{4}{5}$ c. $\frac{15}{5}$ d. $3\frac{3}{4}$ e. not given

---

20. Which picture below shows the same fraction shaded as this set of circles?

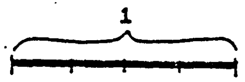


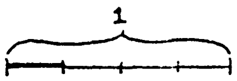
a.  b. 

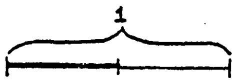
c.  d. 

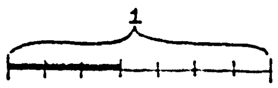
e. not given

21. Which picture below shows the fraction $\frac{3}{4}$?

a. 

b. 

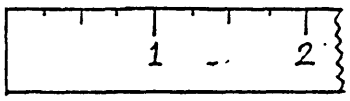
c. 

d. 

e. not given

---

22. This ruler measures inches by



a. wholes, halves, and fourths

b. wholes and halves only

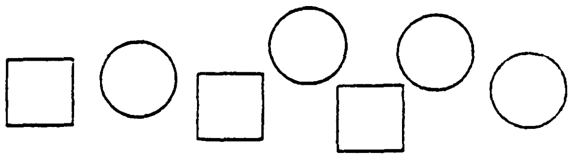
c. wholes, halves, and thirds

d. wholes and fourths only

e. not given

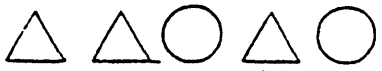
---

Many questions that follow use the word "ratio." In the picture below, the ratio of circles to squares is 4 to 3.



C1 4

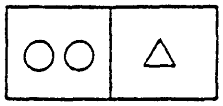
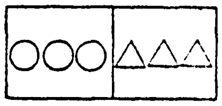
23. What is the ratio of circles to triangles?

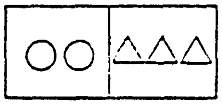
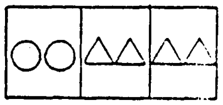


a. 3 to 2 b. 3 to 5 c. 2 to 3

d. 2 to 5 e. not given

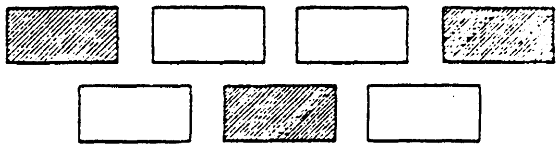
24. Which picture shows the ratio of two circles to three triangles?

a.  b. 

c.  d. 

e. not given

25. What is the ratio of shaded to unshaded rectangles?



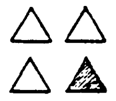
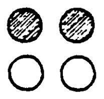
a. three to seven b. three to four

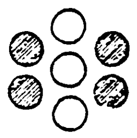
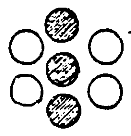
c. four to three d. seven to three e. not given

26. How many thirds equal one whole?

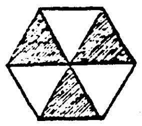
a. 1 b. 2 c. 3 d. 4 e. not given

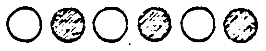
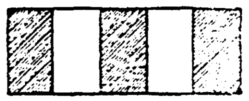
27. Which picture shows $\frac{3}{4}$ shaded?

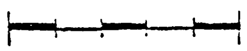
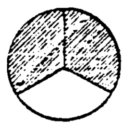
a.  b. 

c.  d.  e. not given

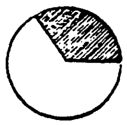
28. What picture shows the same ratio as the ratio of shaded to unshaded triangles in this picture?



a.  b. 

c.  d.  e. not given

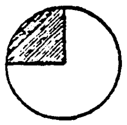
29. What fraction of this circle is shaded?



a. $\frac{1}{2}$   b. $\frac{1}{5}$   c. $\frac{1}{3}$   d. $\frac{3}{1}$   e. not given

---

30. What fraction of this circle is shaded?

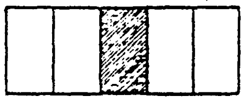


a. one-third   b. one-fifth

c. one-fourth   d. three   e. not given

---

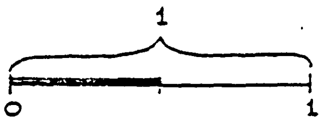
31. What fraction of this rectangle is shaded?



a. $\frac{1}{5}$   b. 4   c. $\frac{1}{4}$   d. $\frac{3}{2}$   e. not given

---

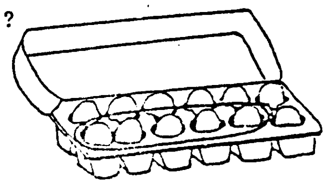
32. What fraction of this line segment is shaded?



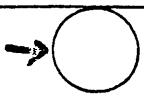
a. one-half   b. one

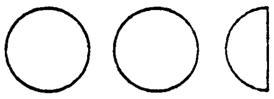
c. two   d. one-third   e. not given

33. What fraction of the eggs are circled?



a. $\frac{5}{12}$ b. $\frac{5}{7}$ c. $\frac{5}{8}$ d. $\frac{7}{12}$ e. not given

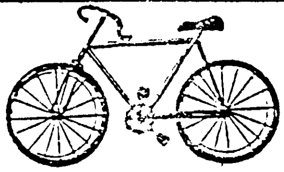
34. If this is the unit,  , then what fraction

is shown by this picture? 

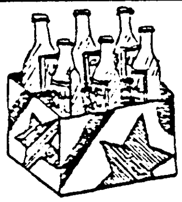
a. 3 b. $\frac{1}{2}$ c. $2\frac{1}{2}$ d. $\frac{3}{2}$ e. not given

In 35-37, tell what ratio is suggested by each picture. Here is a sample problem:

Wheels on one bicycle is 2 to 1.



35. Bottles in one carton is

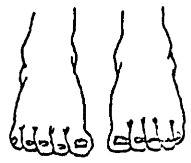


a. 5 to 1 b. 6 to 1 c. 1 to 6

d. 3 to 3 e. not given

6

36. Toes on two feet is

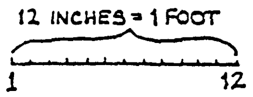


a. ten to two  b. two to ten  c. five to two

d. two to five  e. not given

---

37. Feet to inches is .

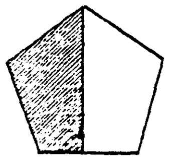


a. 1 to 12  b. 6 to 6  c. 12 to 1

d. 3 to 4  e. not given

---

38. What is the ratio of shaded to unshaded parts in this picture?

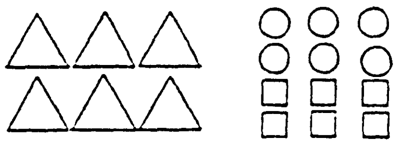


a. 1 to 2  b. 1 to 5  c. 1 to 8

d. 3 to 2  e. not given

---

39. What fraction of the set of objects are triangles?

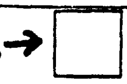
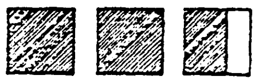


a. $\frac{6}{6}$  b. $\frac{1}{2}$  c. $\frac{1}{3}$  d. $\frac{2}{1}$  e. not given

40. How many thirteenths equal one whole?

a. $\frac{1}{13}$  b. 1  c. 12  d. 13  e. not given

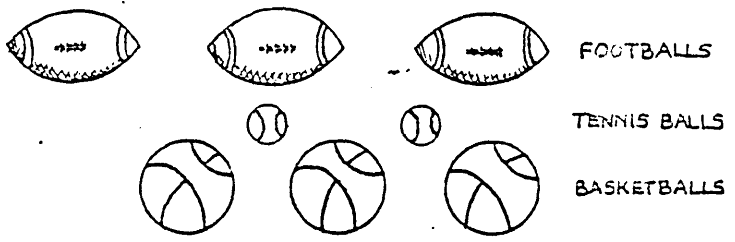
---

41. If this is the unit,  , then what fraction is shaded in this picture? 

a. $2\frac{1}{3}$  b. $2\frac{2}{3}$  c. 3  d. $\frac{3}{8}$  e. not given

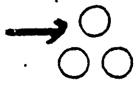
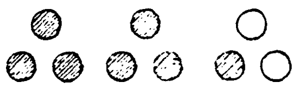
---

42. What fraction of the balls are tennis balls?



a. $\frac{2}{8}$  b. $\frac{3}{2}$  c. $\frac{2}{6}$  d. $\frac{6}{2}$  e. not given

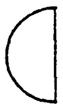
---

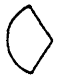
43. If this is the unit,  , then what fraction is shaded in the picture below? 

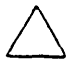
a. $\frac{7}{9}$  b. $\frac{7}{3}$  c. $\frac{7}{2}$  d. $\frac{2}{9}$  e. not given

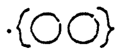
C1

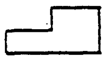
In 44-48, each picture is a fraction of a whole. Tell which answer says how many more parts like the one shown below must be added to make the whole. A sample problem is given below.

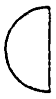
 $\frac{1}{2}$ 1 more part must be added to make a whole.

44.  $\frac{1}{3}$ a. 1 b. 2 c. 3 d. 4 e. not given

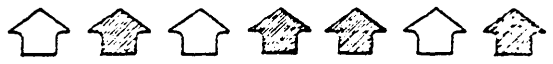
45.  $\frac{1}{4}$ a. 1 b. 2 c. 3 d. 4 e. not given

46.  $\frac{1}{3}$ a. 1 b. 2 c. 3 d. 4 e. not given

47.  $\frac{1}{2}$ a. 1 b. 2 c. 3 d. 4 e. not given

48.  $\frac{1}{4}$ a. 1 b. 2 c. 3 d. 4 e. not given

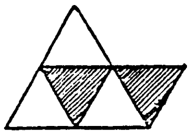
49. What is the numerator of the fraction that tells what part of the picture below is shaded?



a. three b. four c. seven d. three-sevenths

e. not given

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

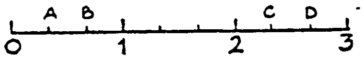


a. five-thirds　　b. five　　c. three

d. two　　e. not given

---

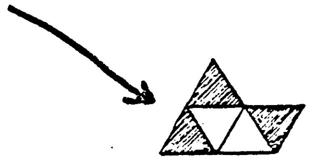
51. Which letter shows the point $\frac{8}{3}$?

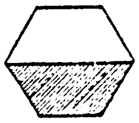
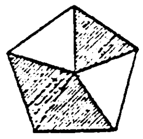


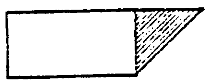
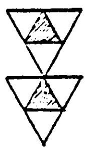
a. A　　b. B　　c. C　　d. D　　e. not given

---

52. Which picture shows the same fraction shaded as this picture?



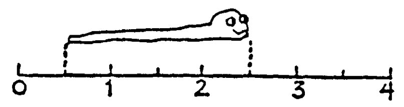
a. 　　b. 

c. 　　d. 

e. not given

C1 7

53. How long is the snake?



a. $\frac{4}{2}$ b. $\frac{5}{2}$ c. $\frac{6}{2}$ d. $\frac{3}{2}$ e. not given

---

54. $2\frac{1}{3} = ?$ a. $\frac{4}{3}$ b. $\frac{8}{3}$ c. $\frac{7}{3}$ d. $\frac{3}{3}$ e. not given

---

55. $3\frac{4}{3} = ?$ a. $\frac{7}{3}$ b. $4\frac{1}{3}$ c. $\frac{12}{3}$ d. $\frac{16}{3}$ e. not given

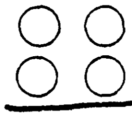
---

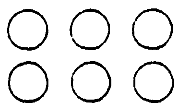
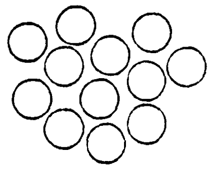
56. $\frac{11}{3} = ?$ a. $3\frac{3}{2}$ b. $2\frac{3}{2}$ c. $3\frac{2}{3}$ d. $10\frac{1}{3}$ e. not given

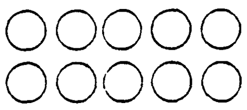
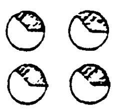
---

The first picture of 57 and 58 is a fraction of the whole.
For each of these problems, tell which answer shows the whole.

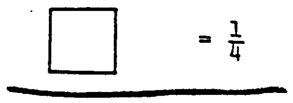
---

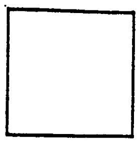
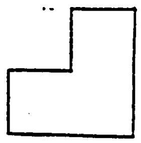
57.  $= \frac{1}{3}$ $\longrightarrow$ which answer shows the whole?

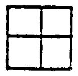
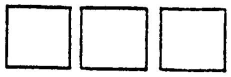
a.  b. 

c.  d. 

e. not given

58.  $\longrightarrow$ Which answer shows the Whole?

a. 
b. 

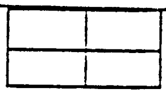
c. 
d. 

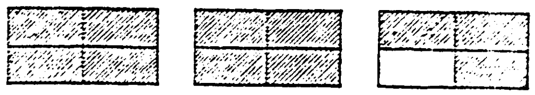
e. not given

59. What fraction of the whole numbers 1,2,3,4,5,6,7 are odd numbers?

a. three-sevenths b. four-sevenths

c. four-ninths d. one-third e. not given

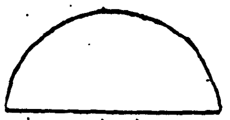
60. If  $= \frac{1}{3}$, then what fraction of the picture below is shaded?



a. three b. two and one half

c. two and three fourths d. eleven-twelfths

e. not given

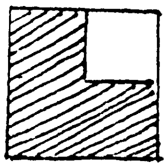
61) ~~m.~~ If  is made up of three equal pieces, draw one piece below.

62) ~~n.~~ Jeff cut a cake into 5 equal-sized pieces and ate two of these pieces. What fraction of the cake did he eat?

WRITE IT HERE: 62 a

DRAW IT HERE: 62 B

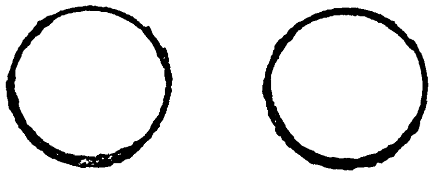
63) ~~o. What fraction of this picture is shaded?~~



64) ~~p.~~ Write a fraction greater than 2/4 and less than 3/4.

65)

3 children are about to share these 2 cakes.
Can you help them cut it, so each child gets an EQUAL share?
Shade one child's share.



65 a

How much is one child's share in numbers? 65 b ____________
How much is one child's share in words? ____________

## APPENDIX B.1.1.

**Classification of the questions according to their modes of translations:**

EASIEST

1. words to words questions: 59,
2. sym to sym questions: 54, 55, 56
3. sym to words questions: 18,
4. words to sym questions: 17, 10, 15, 26, 40,
5. pic to pic questions: 52, 20, 28, 9, 58, 57,
6. words to pic questions: 24, 5, 4, 13, 11, 7,
7. pic to words questions: 8, 32, 2, 49, 30, 35, 23, 25, 36, 22, 37, 38, 50, 60,
8. sym to pic questions: 6, 27, 21, 14, 51, 12,
9. pic to sym questions: 1, 16, 3, 19, 33, 31, 47, 29, 44, 45, 34, 41, 42, 48, 46, 53, 39, 43,

MOST DIFFICULT

Total of 60 Questions.

[constructing fractions: 61, 62a, 62b, 63, 64, 65a, 65b, 65c

(not multiple choice questions) ]

LIST OF PRE-TEST QUESTIONS

2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 18, 19, 20, 21, 26, 27,

29, 31, 32, 33, 34, 40, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51,

52, 53, 54, 55, 56, 57, 58, 60.

TOTAL: 43 QUESTIONS

## BOSTON PUBLIC SCHOOLS

## CURRICULUM REFERENCED TEST

## MATH-LEVEL 4

1985 - 1986

# BOSTON PUBLIC SCHOOLS

CURRICULUM REFERENCED TEST

## MATH - LEVEL 4

**DIRECTIONS:**
- Choose the correct answer.
- Mark only one answer for each problem.
- Record your answer on the separate sheet.

1. What number comes next in this sequence?

   382, 386, 390, ____

   a) 392 b) 400 c) 395 d) 394

2. Amy is sitting in an even-numbered row. What is her row number?

   e) 18 f) 13 g) 11 h) 9

3. Which numeral is the same as <u>six hundred twenty-five thousand eighty-three</u>?

   a) 652,803 b) 683,025 c) 625,083 d) 6,283

4. Which number has a 9 in the <u>hundreds</u> place?

   e) 9,580 f) 5,809 g) 5,980 h) 598

5. Add:

$$\begin{array}{r} 4712 \\ 5153 \\ +\ 1775 \\ \hline \end{array}$$

   a) 11,630 b) 11,640 c) 10,150 d) 91,310

6. Subtract:

$$\begin{array}{r} 1937 \\ -\ 1829 \\ \hline \end{array}$$

   e) 112 f) 108 g) 118 h) 8

**BOSTON PUBLIC SCHOOLS**
CURRICULUM REFERENCED TEST
**MATH - LEVEL 4**

7. Multiply: 8 × 7 = _____ .

   a) 48 b) 54 c) 56 d) 42

8. Divide: $6\,\overline{)\,54}$

   e) 8 f) 6 g) 7 h) 9

9. Subtract: 747 - 38 = _____ .

   a) 719 b) 711 c) 709 d) 701

10. Add: 5563 + 452 = _____ .

    e) 6,015 f) 6,005 g) 9,093 h) 6,915

11. Multiply: $\begin{array}{r} 846 \\ \times\ 37 \\ \hline \end{array}$

    a) 8,460 b) 31,302 c) 24,128 d) 32,302

12. Multiply: 57 × 4 = _____ .

    e) 208 f) 221 g) 228 h) 225

13. Karen played 8 games. In each game she scored 42 points. She needs 600 points to win. How many points has she scored already?

    a) 326 b) 264 c) 550 d) 336

**BOSTON PUBLIC SCHOOLS**
CURRICULUM REFERENCED TEST
**MATH - LEVEL 4**

14. Divide: $7 \overline{)3247}$

e) 464 r. 6 f) 463 g) 463 r. 6 h) 460 r. 6

15. Sandy had 41 pieces of candy and kept 5 for herself. She then divided the remaining pieces evenly among 9 friends. How many pieces did each friend get?

a) 9 b) 14 c) 5 d) 4

16. Which fraction equals one whole?

e) $\frac{6}{6}$ f) $\frac{3}{4}$ g) $\frac{7}{8}$ h) $\frac{5}{6}$

17. Solve: $\frac{2}{3} = \frac{?}{9}$

a) 6 b) 4 c) 8 d) 5

18. Add:

$$\begin{array}{r} 4\frac{3}{8} \\ +\ 5\frac{2}{8} \\ \hline \end{array}$$

e) $9\frac{5}{8}$ f) $9\frac{5}{16}$ g) $9\frac{6}{8}$ h) $9\frac{11}{10}$

**BOSTON PUBLIC SCHOOLS**
CURRICULUM REFERENCED TEST
**MATH - LEVEL 4**

19. Which sign makes this number sentence true?

$$\frac{1}{5} ? \frac{1}{3}$$

a) $>$ b) $=$ c) $\geq$ d) $<$

20. Write as a mixed number: $\frac{13}{7}$

e) 2 f) $\frac{7}{13}$ g) $1\frac{6}{7}$ h) $1\frac{6}{13}$

21. Which fraction equals $8\frac{2}{3}$ ?

a) $\frac{19}{3}$ b) $\frac{13}{3}$ c) $\frac{48}{3}$ d) $\frac{26}{3}$

22. Forty-seven dollars and thirty-two cents equals:

e) \$40.73 f) \$47.32 g) \$4.732 h) \$473.20

23. One half-dollar equals:

a) \$50.00 b) \$0.50 c) \$5.00 d) \$0.25

24. Which is the most money?

e) 4 dimes, 2 nickels, 6 pennies f) 2 quarters, 2 pennies
g) 3 dimes, 8 pennies h) 8 nickels, 1 penny

25. Which of these fractions equals $\frac{1}{2}$ ?

a) $\frac{5}{10}$ b) $\frac{2}{5}$ c) $\frac{2}{3}$ d) $\frac{1}{4}$

**BOSTON PUBLIC SCHOOLS**
CURRICULUM REFERENCED TEST
**MATH - LEVEL 4**

26. How many minutes are there in 3 hours?

e) 72  f) 120  g) 180  h) 30

27. How many years are there in one century?

a) 10  b) 100  c) 24  d) 1000

28. What would be the best way to describe how much water a bathtub holds?

e) cups  f) ounces  g) pints  h) gallons

29. What should you use to time a race?

a) ruler  b) scale  c) measuring cup  d) stopwatch

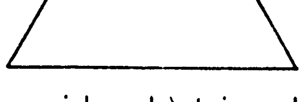
30. How many inches are there in 5 feet?

e) 50  f) 30  g) 60  h) 120

31. Aaron starts his gymnastics class at 3:15. The class lasts $1\frac{1}{2}$ hours. What time is class over?

a) 4:45  b) 4:30  c) 4:00  d) 3:45

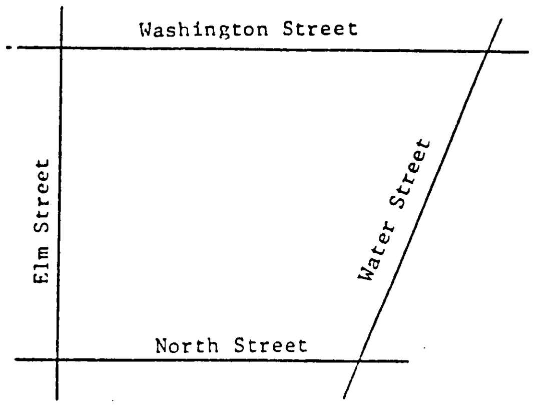
32. Identify this shape:



e) circle  f) square  g) trapezoid  h) triangle

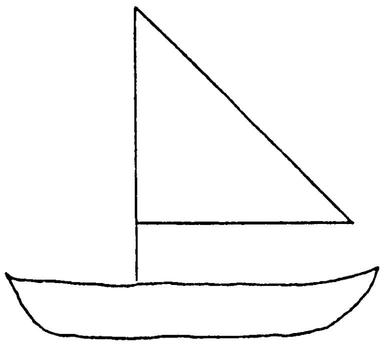
**BOSTON PUBLIC SCHOOLS**
CURRICULUM REFERENCED TEST
**MATH - LEVEL 4**

33. What street is perpendicular to Washington Street?



a) Water Street b) Elm Street c) North Street d) None

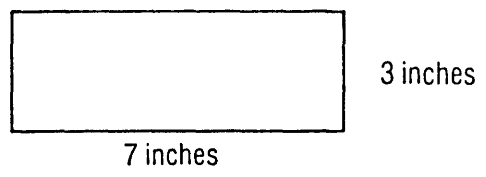
34. What shape is the sail on this boat?



e) triangle f) square g) hexagon h) rectangle

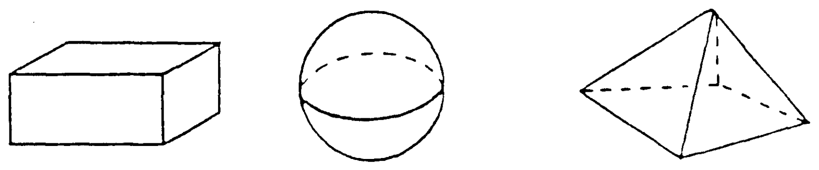
**BOSTON PUBLIC SCHOOLS**
CURRICULUM REFERENCED TEST
**MATH - LEVEL 4**

35. What is the perimeter (the distance around the outside) of this rectangle?



a) 14 inches b) 20 inches c) 21 inches d) 10 inches

36. Name these shapes in the correct order from left to right.



e) sphere, pyramid, rectangular solid
f) rectangular solid, sphere, pyramid
g) pyramid, rectangular solid, sphere
h) rectangular solid, pyramid, sphere

37. Round 4480 to the nearest thousand.

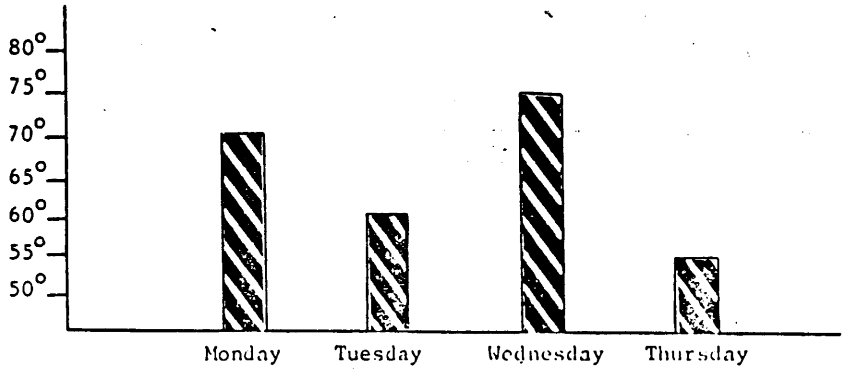
a) 5000 b) 4400 c) 4500 d) 4000

38. A can of Coke costs 55¢. <u>About</u> how much do 6 cans cost?

e) $2.00 f) $3.00 g) $4.00 h) $2.50

**BOSTON PUBLIC SCHOOLS**
CURRICULUM REFERENCED TEST
**MATH - LEVEL 4**

Use this chart to answer Questions 39 and 40.



39. Which was the hottest day?
    a) Monday b) Tuesday c) Wednesday d) Thursday

40. What was the highest temperature on Monday?
    e) 70° f) 75° g) 60° h) 80°

[illegible]

## APPENDIX C:

## SAMPLES FROM THE DESIGNER'S NOTEBOOK

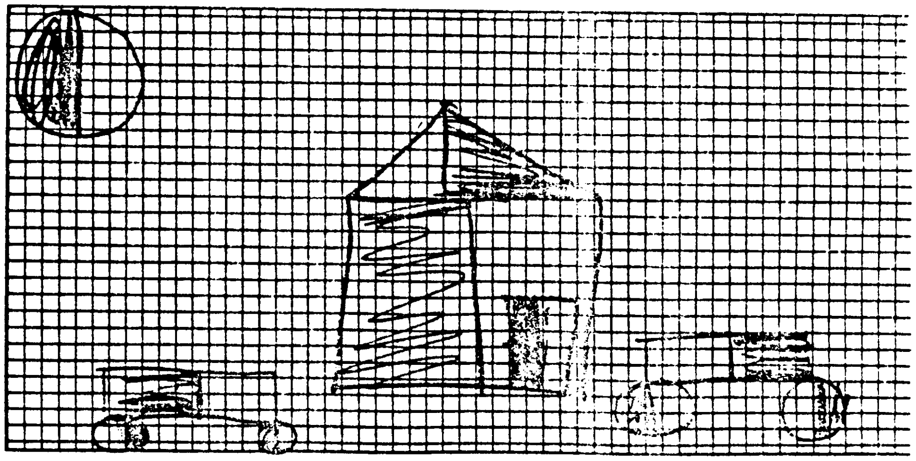
Ideas.......Ideas.......Ideas......Ideas......

Name____________

Today's Date March 23, 1987

## MY PLANS FOR TODAY

I think I have to finish my old fractions project. I'm going to make a scene, where every shape is shaded half. When i'm finish i'm going to do 2/3 for my project & so on. The seeno is a sun, a house, made of triangles, regtangles, & squares. I'm going to have two little wooden wagons. The wagons are made of one rectangle & two circles, each. It looks like this:



Ideas...Ideas.....Ideas.....Ideas......Ideas.......Ideas......Ideas......Ideas....

Ideas.......Ideas.......Ideas......Ideas....

Name

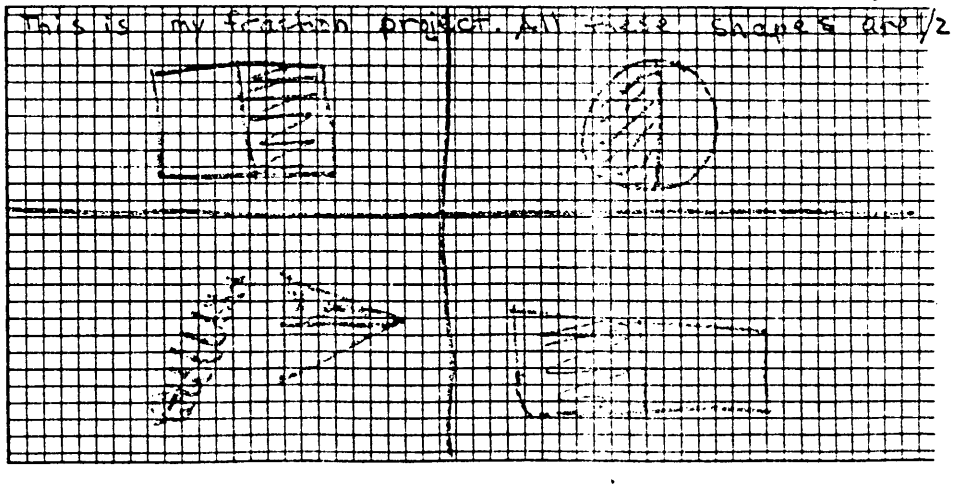
Today's Date March 23, 1987

MY PLANS FOR TODAY

TODAY I'm working on my new fraction project. I have to finish the old one, first.

Bye - Bye - Bye,
Good - Bye,

this is what my other fraction project looks like: ↓



Ideas...Ideas.....Ideas.....Ideas......Ideas.......Ideas......Ideas......Ideas....

NAME [redacted] TODAY'S DATE March 23, 1987

**PROBLEMS I HAD TODAY:**

I had one problem it was making a perfect triangle. Other wise I had no problems. I ~~did~~ was not able to start the fractio scene. I want to start tomorrow! I finishe the other project though.

*** OVERALL, THE PROBLEMS I HAD TODAY ARE MAINLY RELATED TO (check in):
__ LOGO PROGRAMMING; ✓ FRACTIONS; __ DESIGN; __ EXPLAINING or TEACHING SOMETHING;

__ OTHER? ____

*CHANGES I MADE TODAY, and WHY I made these changes:*

I made one change because it messed up!, So I did it over & over finally I asked Cassandra and she helped me. She showed me how to make a perfect triangle.

*PLANS and IDEAS FOR TOMORROW:* None

***...IDEAS.....IDEAS.....IDEAS.....IDEAS***

~~March [illegible]~~

~~[illegible]~~

ON NEXT PAGE

**IDEAS....IDEAS......IDEAS....IDEAS......IDEAS.....IDEAS......IDEAS.......**

NAME___ , ___ TODAY'S DATE March 24, 1987

**PROBLEMS I HAD TODAY:**

I had no problems today.
I had not accomplished to start my graphic called "house, The ~~fract~~ion scene.

*** OVERALL, THE PROBLEMS I HAD TODAY ARE MAINLY RELATED TO (check in):
__LOGO PROGRAMMING: ✓ FRACTIONS: __DESIGN: __EXPLAINING or TEACHING SOMETHING:

__OTHER?____

*CHANGES I MADE TODAY, and WHY I made these changes:*

I made No changes, either for the same reason.

*PLANS and IDEAS FOR TOMORROW...*

***..IDEAS.....IDEAS.....IDEAS....IDEAS***

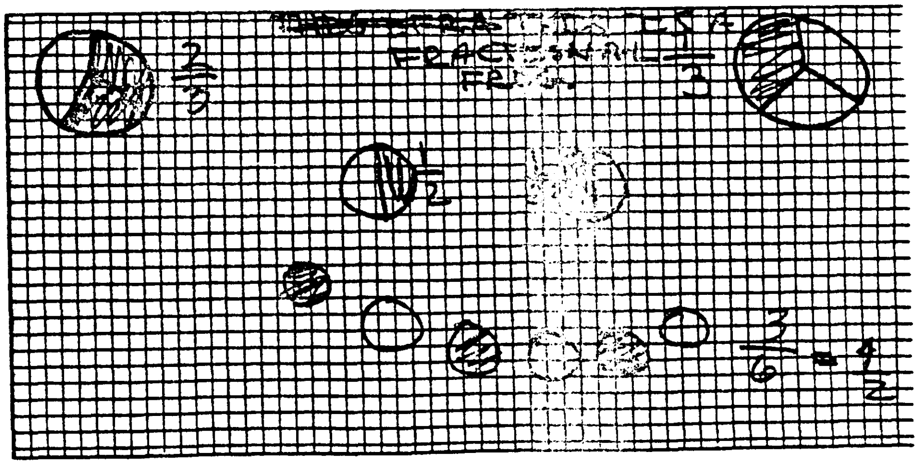
None............... for now!

Ideas.......Ideas.......Ideas......Ideas...... Name

Today's Date March 31, 1987

## MY PLANS FOR TODAY

TODAY I AM GOING TO MAKE UP TWO PEOMS AND THEN I WILL DO, A DIFFERENT SCENE FOR MY FRACTIONS PROJECT. I WILL DRAW THE SCENE BELOW: I'll TRY TO DO THIS, OR I'll DO SOMETHING DIFFERENT.

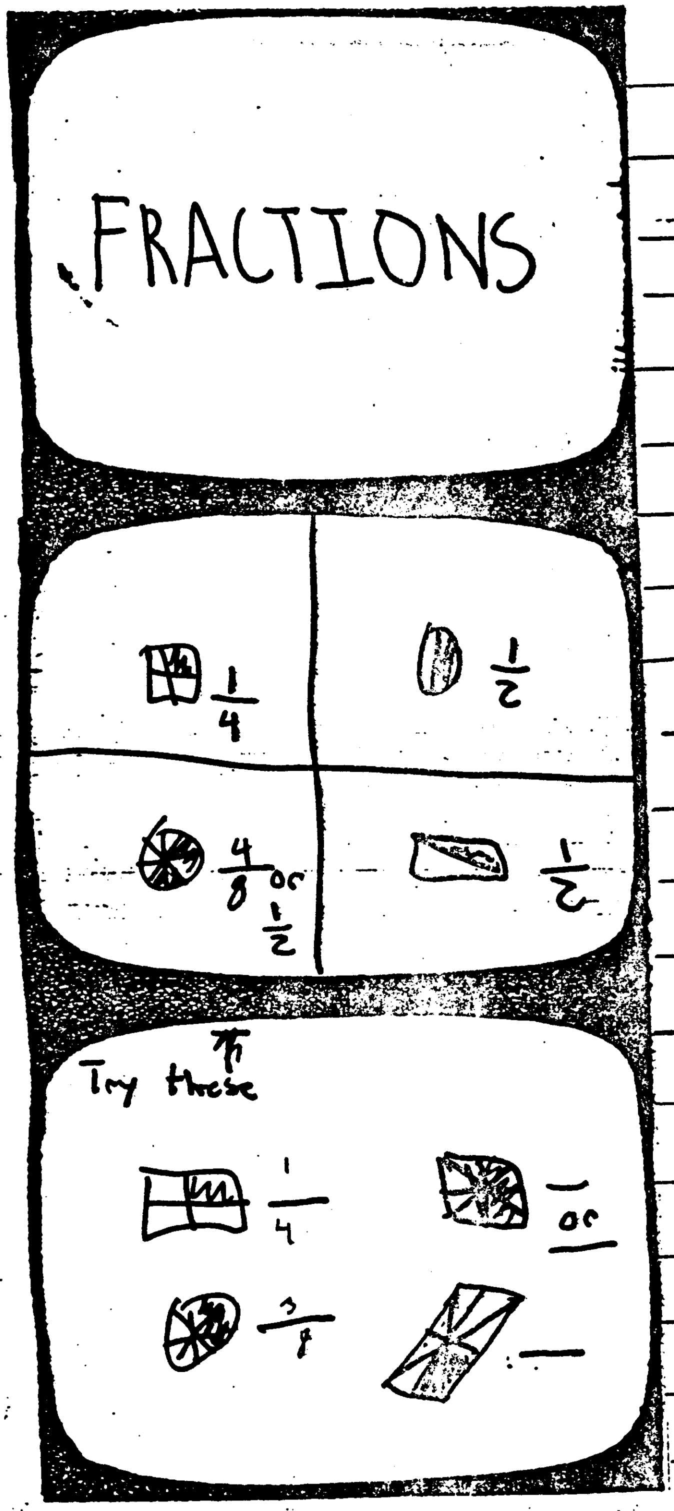


Ideas...Ideas.....Ideas.....Ideas......Ideas.......Ideas.......Ideas......Ideas....

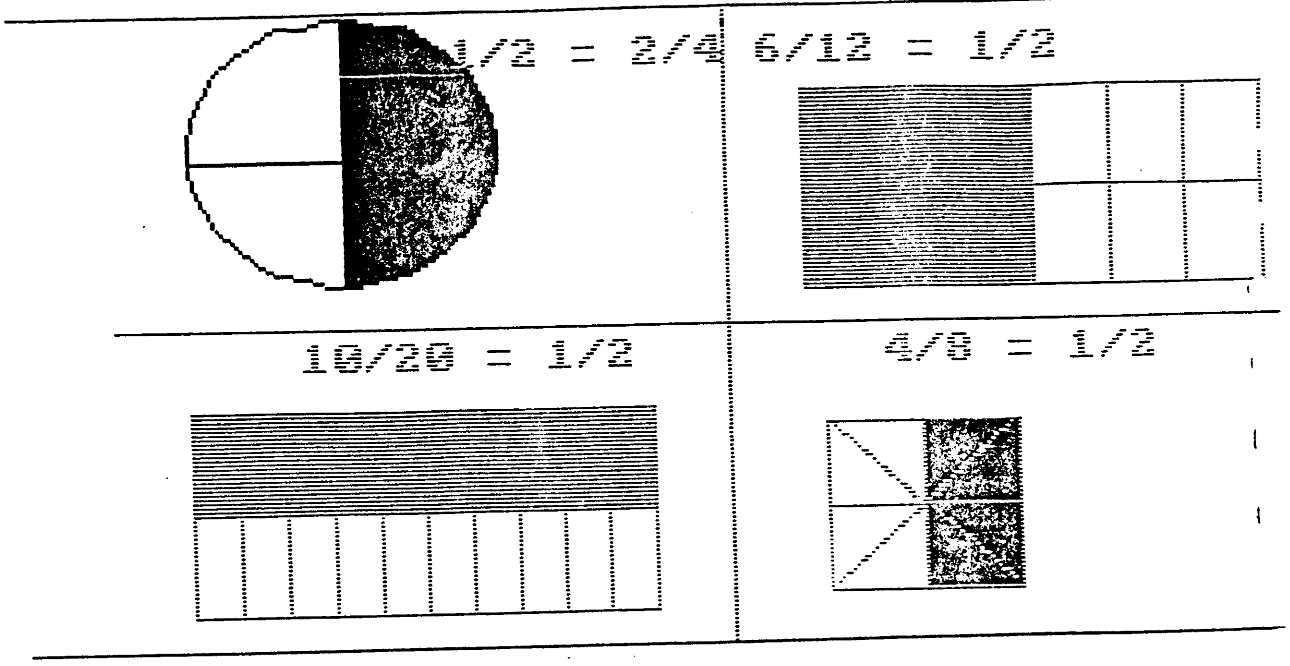
SCRIPT SHEET

NAME ______

DATE March 10, 1987



Logo File (computer printout)
May 18: Equivalence



Ideas.......Ideas.......Ideas......Ideas......

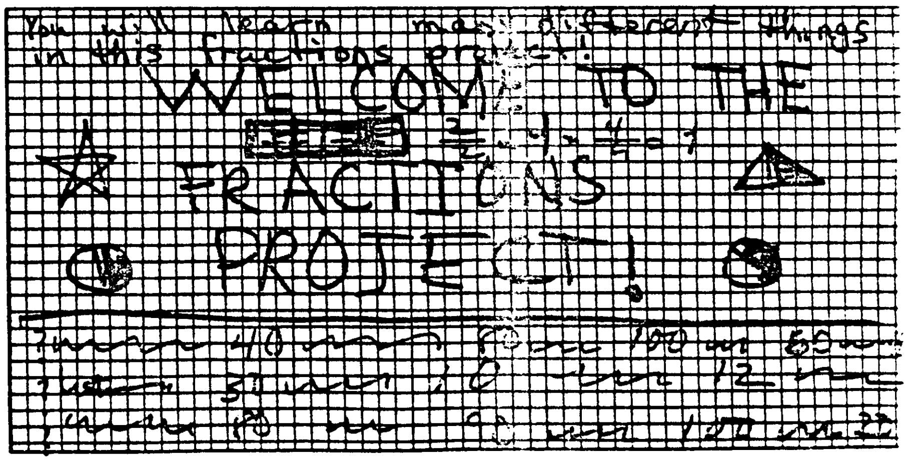
Name

Today's Date April 6, 1987

## MY PLANS FOR TODAY

TODAY I'M GOING TO WORK ON AN the FRONT = introduction of my fractions project. I'll do it in the box below.

◐ = $\frac{1}{2}$ or $\frac{2}{4}$ ◐


You will learn many different things in this fractions project!
WELCOM TO THE
FRACTIONS
PROJECT!

Ideas...Ideas.....Ideas......Ideas......Ideas.......Ideas......Ideas......Ideas...