Massachusetts Institute of Technology

Artificial Intelligence Laboratory

A.I. Memo No. 545

September 1979
LOGO Memo No. 53

FINAL REPORT OF THE BROOKLINE LOGO PROJECT
PART II: Project Summary and Data Analysis

Seymour Papert, Daniel Watt,
Andrea diSessa and Sylvia Weir

A b s t r a c t

During the school year 1977/78 four computers equipped with LOGO and Turtle Graphics were installed in an elementary school in Brookline, Mass. All sixth grade students in the school had between 20 and 40 hours of hands-on experience with the computers. The work of 16 students was documented in detail.

This volume includes: (1) an overview of the Brookline LOGO project, (2) a description of the learning styles of different students who took part in the project, (3) the experiences of students at both extremes of the range of abilities present in a typical public school, (4) a breakdown of the computer programming skills and concepts learned by the students during the course of the project, (5) a breakdown of the mathematical and geometrical skills and concepts learned by the students during the course of the project, and (6) a description of the results of a brief exposure of students to a dynamic turtle which simulates Newtonian motion. (See LOGO Memo 54 for Part III of this report.)

The work reported in this paper was supported in part by the National Science Foundation under grant number 77-19083SEI) and conducted at the Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts. The views and conclusions contained in this paper are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the National Science Foundation or the United States Government.

TABLE OF CONTENTS PART II

General Preface iii

I. Overview: Brookline LOGO Project 1.1

1. Participants 1.1
2. Learning/Teaching Goals and Strategies 1.3
3. Data Collection 1.9
4. Themes of Our Research 1.10
5. Summary of Findings 1.14
6. Acknowledgments 1.25

2. Learning Styles in the LOGO Environment 2.1

1. Kathy: A Modular, Bottom-Up Approach to LOGO Activities 2.1
2. Deborah: A Contrasting Bottom-Up Approach 2.5
3. Donald: A Structured Top-Down Approach to LOGO Activities 2.8
4. Kevin: An Expert "TURTLE Driver;" An Intermediate Example 2.13
5. Working Styles of Other Students 2.17

3. The Experience of Exceptional Children in the LOGO Learning Environment 3.1

1. The Work of Gary: An Intellectually Gifted Student 3.1
2. The Work of Karl: A Severely Learning Disabled Student 3.8
3. The Work of Tina: A Learning Disabled Student 3.15

4. Computer Programming: What the Students Learned	4.1
1. Acquiring the Sense of Command	4.2
2. Developing the Notion of a Procedure as an Entity	4.3
3. Separating the Process from the Product of a Procedure	4.5
4. Acquiring Flexibility in Establishing Hierarchies of Procedures	4.9
5. Fitting a Procedure into a Hierarchy: Top Down vs. Bottom Up	4.22
6. Developing Patterned Procedures Using REPEAT, Recursion and Iteration	4.30
7. Using Variables in LOGO Procedures	4.40
5. Turtle Geometry: What the Students Learned	5.1
1. Qualitative Structuring of the Number World	5.1
2. Quantitative Structuring of the Number World	5.13
3. The Group Properties of the Number Worlds: The Structuring of Mathematical Operations	5.19
4. The Use of Turtle Coordinates: The Beginning of Differential Geometry	5.31
5. The Use of Coordinate Systems: The Global Structuring of the Geometric World	5.37
6. Theorems and Heuristics: Towards Formal Mathematics	5.54
6. Dynamics: Learning Physics with a Dynaturtle	5.1
1. Description of Dynaturtle	6.1
2. Overview of Results	6.3
3. Detailed Observations of Two Students	6.5
4. A Learning Paths Chart for TARGET	6.11
5. Linking to Physics	6.16

LOGO AT THE LINCOLN SCHOOL

GENERAL PREFACE

September 1979 will begin the fourth year of a highly productive relationship between the M.I.T. LOGO group and the Lincoln School in Brookline, Mass. This report presents a particular slice of what has been learned from the experience - the slice that we think is likely to be most relevant to teachers and administrators with enough foresight to be interested in what schools will be able to do with the next generation of personal computers -- those that can be expected to become the dominant species of school computer in the first few years of the 1980's. This is not very far away. By 1982 a "BASIC speaking" TRS-80 with virtually no graphics will seem as obsolete as the time shared computers which are now so rapidly being displaced by the little micros. And 1980 is hardly too soon for schools to be thinking about what they will be doing in 1982.

An important part of the work at the Lincoln School was carried out under a grant from the NSF program for Research in Science Education. Parts II and III of this report constitute the final technical report to the NSF on what was done with that grant. They describe the learning experience of sixteen sixth grade students who learned LOGO during the school year 1977/8 under conditions designed to allow us to collect and analyze an unusually large body of data about their progress.

Part II presents systematic analyses of this data from a number of points of view that will be of interest to educators. It defines a set of programming concepts and skills that it then demonstrates to be within the reach of most sixth grade students. By careful examination of data about the learning paths of the marginal students it develops conjectures about how the proportion of students "proven to be capable of learning to program" could be further increased in the environment defined by systems as flexible as LOGO. Moreover it shows how even with the existing techniques LOGO's flexibility allowed students with the most severe learning problems to make significant gains in directions other than learning to program.

The report presents a detailed analysis of the mathematical content of an experience with a LOGO turtle learning unit. It describes some tests of transfer of learning gains into non-computational contexts.

The report also focuses special attention on "exceptional students" at both ends of the spectrum of school performance. Finally, it shows how programming in LOGO bears on the important problem of bringing out and enhancing individual cognitive styles.

Part III gives the report a dimension of concreteness and specificity that is often missing when the identity of the individual students in an educational experiment disappears into summative statistics. To prevent this we asked the teacher to write reports on each student. These reports are personal in two senses: they discuss each student individually, presenting each student's intellectual personality and his or her learning experience as seen from the personal perspective of a teacher who had the opportunity to get to know the students particularly well.

These two parts of the report, promised to NSF at this time, are now being made available.For general diffusion we shall add an introductory Part I, which will be available in the spring of 1980, to provide readers, unfamiliar with our work, with a context for understanding the research reported in Parts II and III. This will include a discussion of LOGO's educational and computation-theoretical perspectives and of other related work in Europe and in Quebec as well as in the United States.

The future of an intended Part IV is less certain. The work with the sixteen sixth grade students was the beginning of something bigger, the spreading of LOGO culture in the schools. The most satisfying mark of success is the fact that the Brookline School system used its own funds to allow LOGO's presence at Lincoln to continue after the NSF grant ended. If we have the resources to do so we shall be adding a Part IV describing these further developments. This document will, if it comes into being, act as a more concrete guide to schools who wish to emulate the kind of work we report here.

1. Overview: Brookline LOGO Project

This document describes what sixteen sixth grade students made of a LOGO/turtle learning environment and how we interpreted their reactions to it. We begin in this first chapter by presenting in a general way the conclusions we have drawn from the study. Each of the later chapters deals in greater detail with some aspect of the work. It should be read in conjunction with the document called "PART III" which contains the teacher's report on each student.

1. Participants

The sixteen students were selected according to two criteria each of which was intended to insure a variety of students including "average" and "exceptional" students at both ends of the spectrum of academic achievement. The first criterion was distribution on national achievement scores. Table 1.1 shows these scores for fifteen of the students. The sixteenth student, Karl, had not been tested with his classmates due to "severe learning disabilities". The second criterion was the judgment of the classroom teachers who knew the students personally. Teachers were asked to rank them as below average, average, and above average. Our sample includes six students from each of the "non-average" groups.

NAME	TOTAL READING SCORE	TOTAL LANGUAGE SCORE	TOTAL MATH SCORE	TOTAL BATTERY SCORE
Harriet	98	99	92	99
Dennis	77	90	72	83
Gary	79	78	74	80
Donald	82	64	66	73
Laura	71	76	64	73
Kathy	52	71	56	60
Jimmy	49	53	69	57
Monica	69	64	40	56
Albert	48	57	51	51
Darlene	49	66	40	49
Kevin	45	26	74	45
Betsy	38	37	32	32
Deborah	35	44	19	28
Ray	11	28	40	24
Tina	15	2	2	3

The Students' CTBS Scores (Percentiles), April 1978
Table 1.1

Adults in the Classroom

The teacher involved in the study had taught for seven years in the school and had been trained in LOGO at M.I.T. during the year preceding the work with the students. Before joining the school staff he had participated in the E.D.C. Elementary Science Study (ESS) project. He was selected as someone who had had experience as an elementary school teacher and who had been involved in the development of new educational materials.

Besides the teacher three kinds of observers were present from time to time:

(1) regular professional observers who were not members of the M.I.T. staff;
(2) occasional observers who were not members of the M.I.T. project staff;
(3) members of the M.I.T. staff.

2. Learning/Teaching Goals and Strategies

2.1 The Classroom

The classroom included four independent computers, each with its own keyboard, display screen and disc drive. A printer was available for use with one of the computers when necessary. The students were supplied with notebooks, graph paper, drawing paper, different kinds of pens, pencils and markers, and a full set of stationery supplies. A small round table near a blackboard provided a setting for group lessons or discussions and for informal conversation among the students. Samples of the children's work were displayed on bulletin boards around the room.

The students worked in classes of four, so that there was always a one-to-one ratio of students to computers. The length of the class periods ranged from 40 to 90 minutes as a function of the overall school schedule. Table 1.2 shows the distribution of meetings. We note in passing that we believe that the ratio of students to computers is essential to the results we obtained and will be typical of the computer-rich world of the near future. On the other hand, we believe that the student/teacher ratio of 4/1 would not have been necessary under more operational conditions.

CLASS	STARTING AND ENDING DATES	NUMBER OF CLASSES PER WEEK	TOTAL NUMBER OF CLASSES	TOTAL HOURS OF EXPOSURE
I	11/4/77-12/21/77	4	25	25 1/6
II	11/4/77-12/23/77	4	24	25 1/6
III	3/13/78-6/1/78	3	28	37 1/3
IV	3/21/78-6/2/78	3	25	36

Table 1.2

Fall classes met four times a week, for periods ranging from 40 minutes to 90 minutes. In the Spring it was possible to arrange to hold the classes three times a week, and to standardize the times at approximately 80 minutes for each class. Each of the four groups had approximately 4 hours of exposure to LOGO per week. Late notification of the award by the National Science Foundation resulted in a late start for the fall classes, and a total duration of approximately three weeks less than that of the Spring classes.

2.2 Our Goals for the Students

The LOGO system is designed to be flexible enough to serve as a vehicle for many different patterns of learning. Thus an important part of the design of the project was making a set of decisions about what we hoped the students would learn and what strategies the teacher would adopt to bring this about. It was part of the strategy that while the teacher would exert some pressure for the students to achieve the goals we had set for them he would also allow deviations if he felt that a particular student would not respond to the pre-determined goals. This policy proved to be immensely valuable. Significant deviations took place in two cases. In each of these the student taught us something very profound about how a computer can be appropriated to the service of an individual's learning needs. In one of the cases the student eventually returned to the "standard goal"; in the other the deviant student (Tina)had a learning experience of a totally different but extremely rich sort. We shall return to the deviant students after discussing the goals we set up in advance as the educational objectives of the classes. In presenting them here we make an artificial separation of five "kinds" of learning. The reader familiar with LOGO methods will understand that in fact these happen simultaneously as aspects of an organic whole . The students do not perceive them as separate in the early stages of learning; indeed, only those students who achieved a relatively high level of sophistication did so at all in the course of the study.

The five general teaching objectives were as follows:

Objective 1. Learning to feel comfortable with and in control of the computer. Students learn that they can decide what the computer will do: they teach the computer instructions for each of their individually conceived projects.

Objective 2. Learning the elements of the LOGO computer language. This includes

(A1) The idea of computer instruction in a formal language: its syntax, effect, and associated error messages; the LOGO commands FORWARD, RIGHT etc. and the arithmetic operations.

(A2) The idea of sequential procedure and the ability to translate an informally defined plan into a working program;the LOGO commands TO, END.

(A3) The use of sub-procedures and superprocedures.

(A4) Editing and debugging; LOGO commands EDIT, PO, POTS, etc.

(B1) Control of continuing processes with loops and/or recursion;

(B2) Use of variables;

(B3) Conditionals and stop rules;

(B4) Writing interactive programs;

The subgoals listed above are divided into two groups by the labels A and B. This is a post-hoc classification on the basis of observation of these classes. It has become apparent that group A defines a coherent and accessible minimum core knowledge of programming. What this means and what it implies will be developed below in Section 5, Summary of Findings, and in later chapters.

Objective 3. Learning the "subject matter" of Turtle Geometry. Major subgoals include

(A1) The use of numbers to measure lengths and angles; an intuitive as well as a formal mastery of the turtle commands FORWARD, RIGHT etc. Formal and intuitive understanding of special angles: 90, 360, 180, 10.

(A2) The group properties of numbers; for example, seeing FORWARD 10 FORWARD 10 as equivalent to FORWARD 20, seeing BACK 10 and FORWARD -10 as inverses of FORWARD 10; the modularity of the rotational group with respect to 360 degrees.

(A3) Internal relations of angles defining polygons and other regular figures; POLY, SPIRALS.

(A4) Similarity and symmetry; the similarity theorem in Turtle Geometry (if you leave the angles alone and double the lengths you get the same shape twice as big.); the symmetry theorem (if you leave the lengths the same, and reverse the direction of the angles, you get a "mirror image".

(A5) Cartesian coordinate systems.

(B1) Non-Cartesian coordinate systems; inventing ad hoc coordinate systems, polar coordinates, etc.

(B2) The concept of state; state transparent procedures.

(B3) Curves as made up of "infinitesimal" line segments; algorithm for a circle as
REPEAT [FORWARD 1 RIGHT 1].

(B4) Combining movements; eg.
SPIN 50 MOVET 50
produces a circle.

(B5) The Total Turtle Trip theorem.

Items in group A are computation-theoretic forms of items of knowledge found in standard school curricula. Most of the students had encountered related ideas in their previous school work. However, are tests show that their understanding and use of the ideas were, to say the least, shaby. Items in Group B, on the other hand, are computational forms of material usually found in college courses on calculus, topology etc. We could not compare this knowledge acquired in the LOGO environment with what had been learned in school.

Objective 4. Understanding the Relation between Force and Motion

It is now widely recognized that most college students who have not had several courses in physics have incorrect ("Aristotelian") ideas about physical dynamics and that this is very hard to remedy by traditional teaching. With the small time available in these LOGO classes it would have been absurd to set a goal of having the students "understand dynamics". But it was realistic to set a more limited goal: to find situations in which the elementary school student could meaningfully come to grips with laws of motion. The most direct form of achieving this was the creation of computer "games" in which the students' knowledge of turtles could be used to manipulate dynamic turtles which in fact behave like Newtonian particles. The educational objectives were for the student to become sufficiently involved in such activities to feel the contradiction between the turtle and his intuition and to give him/her the intellectual tools that could in principle provide a way out of the dilemma. Our research goal was to demonstrate the possibility of doing so, to identify more precisely the content of the students' "Aristotelian" physics and to probe its resistance to change. This component of the work, reported in chapter 6, is of a much more exploratory nature than the work on geometry with which we have had very much more systematic experience in the past.

Objective 5. Developing Problem-Solving Skills.

Those skills stressed in the present project and discussed in this report include

(a) "playing turtle" and "playing computer";
(b) the concept of a "bug" in a computer program and strategies for debugging and planning;
(c) procedural thinking;
(d) the usefulness of generalizations and "big ideas"; and
(e) the development of a language with which the student and adult participants in the project were able to discuss these skills.

2.3 Teaching Strategies

Although the teacher had certain goals in mind for the students, he allowed them a diversity of paths to get there. This allowed for the fact, clearly documented in the following chapters, that since student vary in which aspects of the LOGO work they initially find easiest, they get to the same end result most effectively by following individually chosen paths. Thus, for example, some of the 16 students quickly became adept at using elements of Turtle Geometry but had more difficulty with the syntax of the computer language. For others, the reverse was true. As the students began to develop their own working styles and sets of priorities, they developed a sense of confidence about what they had done and about their individualized paths to learning LOGO.

The initial contact with the computer was concentrated on (1) learning the elementary TURTLE commands (FORWARD, BACK, RIGHT, LEFT and CLEARSCREEN), (2) mastering elements of syntax such as spacing and the use of numerical inputs and (3) reading and acting on error messages. The students would then be encouraged to define their own tasks, typically involving drawing a specific, "simple" figure such as a square, a house, a flower, or their initials. Their first drawings were done by direct command. They are encouraged to keep a written record of the steps as they go along, and gradually learned to translate these lists into their first computer procedures.

Having procedures that could be saved, repeated, shown off to friends and integrated into a larger design built the student's sense of pride and confidence. The fact that the student's first procedure was a personal invention (even if developed in collaboration with the teacher) was critical in setting the tone of the relationship with the computer through the whole period.

After the introductory phase the students' ways of working rapidly diverged. Some were most interested in repeating simple figures, introducing variations and repeating again. For these students the natural next step was recursion and the use of variables and some took the step in a fairly short time. Others had elaborate ideas for computer drawings, and for these the natural next step was the use of superprocedures and subprocedures. An interchange of approaches occurred as students began to show each other their work and to swap ideas. Students were encouraged to borrow each other's procedures, even to copy them line-by-line at times. (A lot of very useful debugging can occur when a "copied" procedure leads to an unexpected result.)

As the classes continued, the teacher helped the students choose projects or suggested projects based on their interests and abilities. His role in general was to introduce new material when appropriate, to help the students improve their programming styles through the use of model programs and suggestions for debugging, to encourage them to investigate certain areas more deeply, and in general, to help them to consolidate their learning. Particular attention was paid to developing a language for problem solving situations in and out of LOGO work.

The students met at intervals for group lessons where they could share and discuss their work. Each kept a notebook of drawings, written plans, printed records of their procedures, other information, and a brief daily comment about what they had accomplished.

Throughout the year, the teacher made a daily study of each "dribble file"; i.e. the complete printed record, key stroke by key stroke, of interaction with the computer. This was valuable both as a source of data for reporting and analysing the progress of the srtudents and also as a daily guide in planning teaching strategies.

3 Data Collection

Data sources included:

1. Dribble files, that is a complete record of each student's interaction with the computer. Following each class, printouts of the dribble files were carefully annotated by the teacher and/or a regular observer.

2. The teacher's anecdotal records of each student's daily work.

3. A daily compilation of each student's work -- printouts of procedures, hard copy of computer drawings, etc.

4. Observations by Ms. Dunning, conducted on a regular basis; her written reports focused on student-teacher, student-student and student-computer interactions.

5. Regularly conducted observations by members of the MIT LOGO Group.

6. Occasional observations by other members of the MIT faculty, Dr. George Hein, other evaluation consultants and visitors to the LOGO

classes.

7. Informal meetings and workshops with the classroom teachers and the school principal.

8. A series of pre/post student interviews developed by the project staff in collaboration with Dr. Hein and Ms. Dunning, and carried out by Ms. Dunning. Pre-interviews were useful in providing us with baseline information about each student's interests, skills, problem-solving abilities, and attitudes about themselves, their school work and towards computers.

9. Interviews with the classroom teachers, conducted by Dr. Hein, after the conclusion of each series of classes.

10. Seminars and meetings with evaluation consultants who commented on the data.

11. Comments made at parents' meetings, school "open house" etc.

12. An independent study made by Ms. C. Solomon on the diffusion of LOGO knowledge in the school during the year following the work reported here.

4 Themes of Our Research

We present our findings in terms of four themes:(1) can elementary school students learn to program computers? (2) What is the relationship between programming and learning mathematics? (3) What is the relationship between programming and cognitive style? (4) How can we evaluate the work?

The first theme is a set of concerns about whether students can program computers. This cannot be formulated as a yes or no question. If the criterion for what counts as programming is sufficiently trivialized the answer is tautologically affirmative whereas if the criterion for what it is to program is made sufficiently rigorous the answer would almost as clearly be negative. A better formulation would be to ask what kinds of programming can be done by various categories of students. But to give an exhaustive answer to such a question would be a momentous, and perhaps even definitionally impossible task. At best it is a question that can be answered only through a lengthy process of multiple experiments so that experiences in different programming contexts could be compared. This study is a step in that direction. We are possibly the first to have set up a well considered definition of programming (that was not dictated by the accident of a particular programming language simply being there) and to have published (for example in this study) a detailed account of how a representative group of students actually fared with it. This is not the place to delve deeply into the philosophy of design of the definition of programming embodied in the learning goals listed above. For such a discussion the reader must consult our theoretical writings (see the MIT Logo bibliography.) Here we confine ourselves to enumerating a few design criteria for what constitutes a "good programming experience".

(i) As computer science has developed a certain number of powerful ideas have come to be recognized in the field of programming. Among these is the cluster of ideas related to the modularity of pure procedures, the concept of top-down, "structured" programming etc. One of our criteria has been to capture for students as many as possible of these powerful ideas including, in particular, the example cited of procedural modularity. We have, of course, not been able to capture all the powerful ideas of computer science. But we have made an attempt to capture some of them.

(ii) There has grown up in the "computer culture" a rich tradition of using computational ideas as tools to think about other matters. For example, the cognitive sciences now use ideas from Artificial Intelligence to think about psychology. We have tried to provide, within our concept of programming, ideas about thinking styles -- ideas that can be used by students and their teachers as tools to think about other matters of interest.

(iii) Third, we refer to the "holding power" of programming. For many people this activity has an exceptionally powerful quality of engagement of attention. We believe that the search for the proper subset of programming knowledge to give students must take account of such affective criteria. Thus we pose the question: what kinds of programming will be most engaging for what categories of students?

With these criteria in mind, we fixed on one particular vision of programming for the purpose of this study. Our concerns have to do, on the one hand, with whether this approach to programming does engage students, whether in fact they learn to do it and, on the other hand, with whether what they are learning is non-trivial, and whether it captures some of the intellectual and aesthetic content of programming.

This brings us to our second theme, the relation between learning to program and learning other areas of knowledge. In this study we have picked one such area of knowledge: geometry. It is our contention that computational concepts allow some essential parts of geometry to be captured more concretely, more deeply and more intuitively than traditional conceptual frameworks. For example where Euclid uses the static concept of a point we use the dynamic one of a turtle. This allows a more direct and intuitive access to formal geometry. Another, perhaps more immediately obvious example is the difference between the idea of angle in Euclidean geometry and in Turtle Geometry. In the latter an angle is an action, an amount of turning, something that you (the would-be geometer) can do with your own body or with your mental body image. Similarly, the relation between mathematics and physics becomes more immediate in this conceptual framework; for the Newtonian particle turns out to be representable as a species of computational turtle closely enough related to the geometry turtle for each to serve as a means for thinking about the other.

In short,the general concept of this second theme is that of synergistic domains of knowledge. We maintain that LOGO and Turtle Geometry are synergistic in the sense that it is easier to learn both together than to learn either separately. Perhaps the triplet geometry/physics/LOGO is even more powerfully synergistic.

The third theme is the relation of programming to general intellectual skills and styles. The question of the impact of computers on how people think is a controversial one. There is a popular view that "programming teaches you to think logically." There is the view of certain critics who fear that it does this too well, that is to say it encourages an overly "logical" or "analytic" mode of thought at the expense of intuitive, empathic, holistic thinking. There is also the fear that it can encourage isolation of the individual who comes to relate more to the computer than to other people.

The data presented here certainly makes no claim to settle the controversy. The issues are much too big for so modest an experiment. But it does permit us to take some steps towards clarifying them. As background we recall that members of the MIT LOGO group have generally taken a more modulated view than those expressed above. We are suspicious of any statements of the form "programming computers has effect X." The experience of programming and the use of concepts from computation allows for such wide variation that consistent effects would be surprising. The task for educators is to learn what these different effects can be and to learn how to turn them to advantage for intellectual growth.

The first publication from the MIT LOGO group was a paper by Papert called Teaching Children Thinking. The thesis of this paper was not at all that computers would enhance thinking in any automatic sense but rather that exposure to programming could be used to enhance students' ability to think about their own thinking. The stories of the sixteen Brookline students are rich in information relevant to planning strategies for doing this. For example, they show more clearly than previous studies how programming can be a sensitive medium for the expression of different intellectual styles. Different students working on similar projects come up with very different solutions. In other words the computer helps to externalize the individual's style so that it can be confronted by both learner and teacher.

The fourth theme is evaluation: how do we know what really happened and how do we decide whether it is "good?"

Most of this document is devoted to our attempt to describe what was learned insofar as it could be seen in the LOGO environment itself. This is by no means a trivial matter of factual reporting. It involves making interpretations of observed behaviors and conjectures about the nature of the obstacles the students had to overcome. For example, while the count of how many students eventually used the command RIGHT correctly is a mere matter of statistics, much more complex issues are raised when we ask why a particular student uses the command only in the context RIGHT 30 (rather than RIGHT 20, RIGHT 90, etc.) or why another student takes so long to appreciate the synonymity of RIGHT 90 and the sequence RIGHT 45 RIGHT 45.

Two other important classes of evaluative question are touched on, but in a very much less elaborated form. First, the issue of transfer. We obviously would like to know how the knowledge gained in the LOGO environment is integrated by the individual student into other activities in school and out of it. We did not have the resources to study this kind of question in great detail and, in any case, are inclined to believe that the time course of the study was too limited for deep effects to show themselves. We did, however, probe the transfer of a few parts of what is learned in LOGO that is particularly close to topics of study in the normal classroom. For example, we shall report below on a simple experiment to show that working with Turtles leads to a measurable improvement in the ability to estimate angles.

A second question of evaluation about which teachers will want to know much more than we can tell them is how LOGO compares with other approaches to learning to program. For example, is it really better than the much more easily accessible BASIC? The final answer will come from formal or informal comparative studies. In the meantime we think that anyone who has taught BASIC to "average" and "very weak" students at elementary school level will recognize that many of the key breakthrough points in our students' learning simply could not happen in a typical BASIC environment.

Although what we are able to say here about these latter two issues is slight, we have learned a great deal about what kind of study we need in order to elucidate them further. The simplest such conclusion is relevant to our sense of how much exposure to computers can be expected to produce large and generalized changes in intellectual growth. We are at present engaged in a series of experiments on different degrees of involvment, including one in which a group of 8 year old students will have almost unlimited access to computers at home as well as at school over a period of two years. Other conclusions have to do with the need to pay closer attention to affective and social aspects of the learning experience. Some of our students show very strong affect in their relationship with the computer and the entire experience. It has become quite obvious that when we look for "transfer" this ought at least to include transfer of feelings. But it can also be conjectured that this is not simply an additional factor to be looked at as well as the more "cognitive" parts of what is learned. It is increasingly plausible that the way in which the computer experience will transfer to the learning of traditional mathematics is mediated by a change of feeling and is masked by failure to look at the student as a whole person. One strategy for following through on this observation is to look very much more closely at the kinds of experience reported here in our discussion of "exceptional" students.

5. Summary of Findings

5.1 Theme 1: A Core Set of Programming Skills

We approached the project with a definite concept of what we hoped to have the students learn, namely at least the set of objectives in group A and hopefully those in group B. (See section 2.2 for these objectives.)

But we also had a flexible attitude regarding both method and the possibility of modifying specific content goals. Specifically, all the students were introduced to a pre-planned LOGO/turtle learning unit and were under some pressure to follow it. But deviations from the plan were allowed when it became clear to the teacher that this would lead to a more meaningful learning experience for the student and for us.

Which Students Learned To Program?

Of the sixteen subjects we find that two did not achieve the set of skills in group A and so cannot be said to have "learned to program" in any significant sense. This does not mean that they learned nothing. On the contrary, we have reason to believe that the experience profoundly influenced at least one of them.

Our sample size was certainly too small to allow for any general conclusions to be drawn about the two "non-programmers". However, we do note that they were the two students with the lowest scores on the National Achievement Tests: Tina with a total battery percentile score of 3 (!) never learned to write procedures at all while Ray, with a score of 24, learned to write procedures and even used sub-procedures, but did not reach our criterion of being able to do so independently of help from the teacher. Karl, the student with severe learning disabilities who did not have a test score, reached the criterion quite adequately although in an interestingly personal way. (Tina's and Karl's experiences are discussed in chapter 3, below, which focuses on exceptional students. Further details of their work can be found in Part III which contains individual reports written by the teacher.)

We had hoped to be able to demonstrate that ALL students irrespective of level of academic achievement could learn to program. In fact, the strongest statement of this kind that we can make on the basis of the present study is that all students except those in the lowest quartile of school performance did reach our criteria.However we are able to maker a much stronger statement of a somewhat different and much more important kind:

> <u>ALL students irrespective of performance level were engaged by computer activities in the LOGO environment; all underwent significant observed learning and we made significant progress towards developing a methodology of channelling this learning toward mastery of programming.</u>

The most significant idea in the direction of this general methodology is a development of the concept of "micro-world" which has guided much of our thinking over the past few years. The new development of the idea is well illustrated by the experience of one of the subjects in the present study. This is Deborah who will be discussed at several places in the following chapters. Very schematically Deborah's progress could be described as having three phases. During the first she made little progress towards mastery of LOGO. Her behavior was dominated by a lack of security and negative self image. In a second phase Deborah defined for herself a very restricted sub-world of the LOGO/turtle world by placing severe restrictions on what commands she would use and what forms of projects she would undertake. <u>Within this personally constructed micro-world she could feel sufficiently secure to explore and become comfortable with the machine, the turtle and with the formal context of programming</u> The third stage developed just in time for us to observe at the end of the study. Deborah spontaneously broke out from the self-imposed boundaries of her micro-world to begin exploring the larger "official" turtle world.

Deborah's experience was made possible by the flexibility of the LOGO computer system and by the teaching methodology that has developed with it. Study of the details of what took place has led us to a general research orientation of increasing the ease with which this pattern can be followed by other students. From a theoretical point of view it has confirmed our long-standing belief in the importance of creating systems that can be appropriated in a personal way by individual students.

More Advanced Programming Concepts

Since the work in the classes was highly project-oriented each student's needs for concepts in group B varied according to the kind of project each adopted. Thus a count of the students reaching a criterion of mastery of each concept is not an informative measure. We note, however, that half the students reached independent mastery of at least one of these group B concepts. Of these eight students, six also mastered at least one other group B concept, while the other two were apparently very close to doing so. The group B concept most frequently mastered was loops and recursion.

The students who went farthest in mastery of the listed group B concepts were Harriet, Gary, Kevin and Dennis. Reference to the list of achievement scores (Table 1.1) shows rather close agreement between high scores and mastery of group B concepts except for Kevin who had a low total score but a high mathematical score.

We feel that these results confirm us in our separation of the group A and Group B concepts and in approaching programming via the first set. We call attention to the fact that this is made possible by specific and deliberately designed features of LOGO. By contrast, in BASIC, it is impossible to write interesting programs without using some of the "advanced" concepts so that the "non-mathematical" students have difficulty getting started and the gap between "mathematically-minded" students and others is widened.

5.2 Theme 2: Turtle Geometry and LOGO as Synergistic domains

As far as it appeared from our study, the students did not distinguish between learning programming and learning Turtle Geometry. Their initial work at the computer, and in some cases all their work, consisted of developing programs to produce graphic effects on the computer screen. The separation of the knowledge required to develop these programs into programming knowledge, arithmetical knowledge, geometrical knowledge, heuristic knowledge, etc. does not appear to have been of much concern to them. They may be right in their attitude. We, however, felt it necessary to make these distinctions. One reason was to make our work more intelligible to educators used to thinking in terms of subjects each of which has a distinct curriculum. Another, more fundamental, reason was to classify LOGO-related knowledge according to the type and style of intellectual skills required. The boundaries between school subjects are often artificial. For example, it appears to us to be pedagogically and epistemologically wrong to separate "calculus" from "physics" at the introductory levels. The boundaries we have described, on the other hand, reflect real differences in intellectual approach. These distinctions have aided the students and provide a useful structure for discussing their intellectual gains.

In chapters 4 and 5 of this report we shall present detailed analysis of the intellectual content of what one group of students involved in the project actually learned in LOGO/turtle work. That analysis, taken with chapter 6 -- where we show how turtle work brings a beginning student into touch with some fundamental ideas in physics -- shows that a great deal of traditional knowledge is embedded in and exercised by work with turtles.

5.3 Theme 3: Identifying Intellectual Styles

The study has provided us with new data concerning the diversity of intellectual styles, and the effect of learning LOGO on an individual student's image of herself or himself as a learner. For example, even when two students end up with very similar products, an analysis of the dribble file, which shows the process by which they arrived at their result, indicates that there is no sense in which they "just did the same thing". The data from the project has yielded some very fine examples of convergence of different processes and has contributed to further work on classifying intellectual styles. We see this as extremely important both for theoretical psychology and for the development of strategies for teaching and learning.

The most striking dichotomy of styles one sees in a LOGO environment is that between what computer programmers would refer to as "bottom up" as against "top down" programming. The top down programmer is a planner. He starts with a clear model of an end result. His first step is to translate this into a program the details of which are left blank but whose structure has been fixed from the outset. Here, Donald represents the extreme example of a top-down programmer who worked steadily to turn a hand-drawn face into a computer program that drew a face remarkably like it. Over a period of twelve sessions Donald worked to fill in the gaps in a program, HEAD, that had taken on its almost final form by the second of the twelve sessions. (See pp. 2.12, 2.13, below)

Deborah also produced a face. A glance at her first and last drawings, pp. 2.7 and 2.8, will show immediately that she was following a very different process. She has been much freer than Donald in allowing the final product to emerge as she went along. The difference is even more striking when we look at the stages of emergence of her procedure as shown on p 2.8. The procedure was built up line by line as she worked at it with little trace of Donald's preference to structure the whole before beginning on the parts. Following Levi-Strauss, Bob Lawler has proposed to call Deborah's cognitive style a "bricolage" or tinkering. A certain well-established intellectual tradition tends to look down on the style of the bricoleur and see the obsessional planner as the model to follow and impose on students. Research by Lawler and others at M.I.T. as well as the results found in the Lincoln study indicate that the style of bricolage may be much more natural and much more productive than is usually admitted. Certainly we are convinced that the turning point in Deborah's development in LOGO came when the teacher found a way to let her feel supported in developing an extreme form of bricolage in her work. Possibly this was one of the first times when this girl felt that she had achieved success and acknowledgment through work in her own personal style. And her response to this sense of security was so great that she was able to make some tentative essays into the planner's style of work. Thus even educators who do not admit bricolage as a valid mode of work in itself should give it credit as a stepping stone to the ability to experiment with a variety of styles.

(More details of Deborah's work can be found in chapter 2, below, and in her profile in Part III)

5.4 Theme 4: A Tentative Measure of "Transfer"

The results cited in this section are intended only to illustrate a direction of work in progress. They are incomplete and at best on the fringe of statistical significance. Nevertheless, our considered judgment is that they are highly plausible and reflect real trends. We offer them as a guide to others who may want to pursue such questions more rigorously, as we are now doing ourselves. With these qualifications, let us turn to a particular issue: the transfer of knowledge about angles and angular measure from the Turtle context to more general ones.

What should we measure? It is clearly not sufficient to test knowledge of discrete "facts" such as "a square angle is called ninety." We want to measure the use of the knowledge in a context where the student has to apply it less literally. On the other hand, we do not want to confuse the issue of the student's knowledge about angles with problem solving methods in other domains. Our compromise was to study the student's ability to perform on the task shown in figure 1.2.

We have data about this task from three categories of subjects.

Category I: Students who took part in the study reported here.

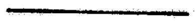
Category II: Students who took part in another less systematic LOGO/Turtle computer experience.

Category III: Students who had no computer experience.

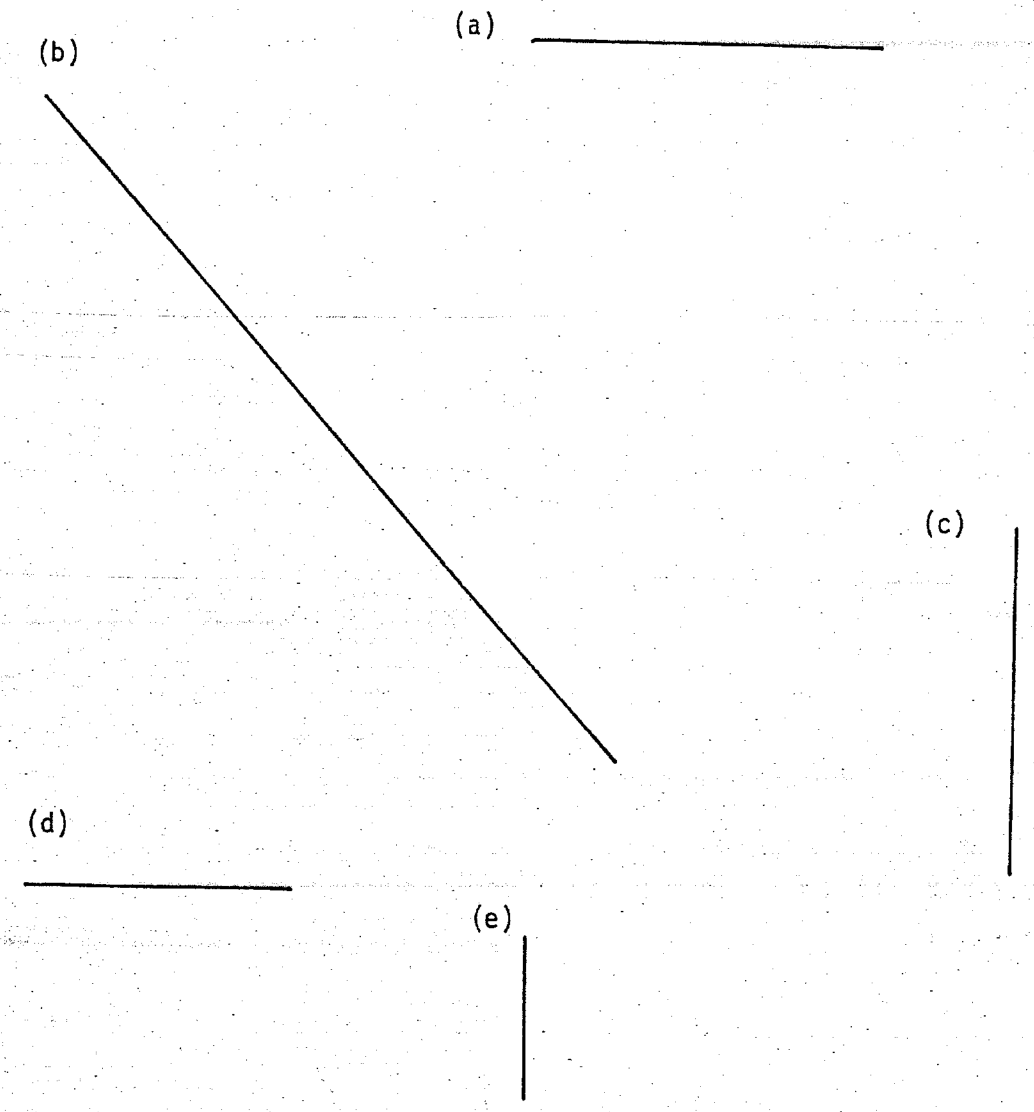
Each subject went through the test twice, before and after the computer experience in the case of categories I and II. We adopted an arbitrary scoring method to give a numerical value to the differences obtained and found (no doubt coincidentally) that category I came out ahead of category II by about the same score difference as category II was ahead of category III.

These differences do not quite make statistical significance but are so convergent with many other observations that we believe them to reflect genuine change. But we need to probe further into the nature of what is changed. For example, it could be improved ability to use numerical estimates in general rather than improved knowledge about angle in particular. The question is elucidated a little by the performance of the same subjects on another task, the one shown in figure 1.1. Here again we find categories I and II doing better than category III. But the differences are less pronounced. One possible interpretation of the comparison between the two tasks is based on a conjecture that may prove to be critical in this kind of study. The students came to both tasks with "intuitive" knowledge that has been built up over many years and in informal as well as formal settings. The new knowledge acquired in the Turtle (or any other) learning environment has to "compete" with knowledge that is already firmly rooted. How firmly it is rooted might influence how much time is needed for new knowledge to displace it. This would account for smaller improvements in the case of estimates of length than in the case of estimates of angle. It would also suggest that sensible measures of change have to allow for longer periods of time than we were able to use in this study. As we have already said, we have begun deeper explorations of such issues in studies with a much longer time course. As an indirect consequence of the length of the new studies, we shall also be able to collect much more varied information about the development of each subject.

We conclude by citing two other tasks, largely with the intention of reminding readers that sixth grade students in public schools have a need for learning intellectual skills that one would expect to be quite easy by that age. The tasks in figures 1.3 and 1.4 are self-explanatory. The test scores we obtained follow a different pattern from what we saw in the estimation tasks: about half of the subjects could do them as well as a sophisticated adult, (whereas no subjects performed "perfectly" on the estimation tasks.) Given this, it will not be surprizing that those students who who did not obtain perfect scores on the pre-test all showed major improvements on the task of figure 1.3 (which is, of course, very close to the turtle work) and about half showed striking improvement on the task of figure 1.4 (which is related to programming but not a direct transposition.)

If this length is 100 

How long are these?



Draw a line which you think will be

(f) 150

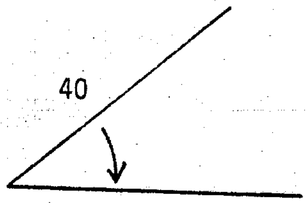
(g) 400

(h) 99

Task 1

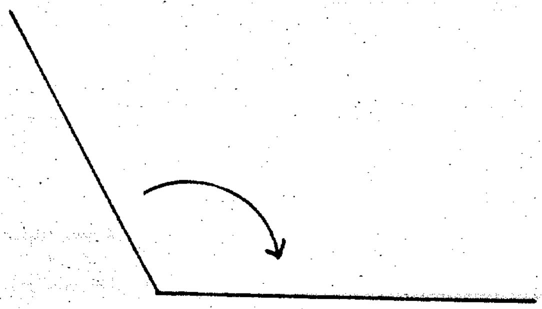
Figure 1.1

If this angle is 40 units

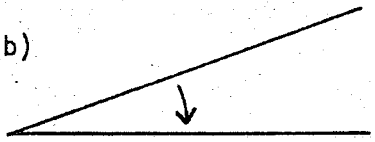


What would you estimate these to be?

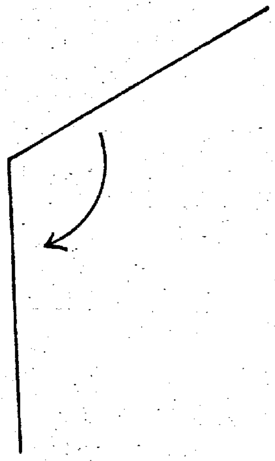
(a)



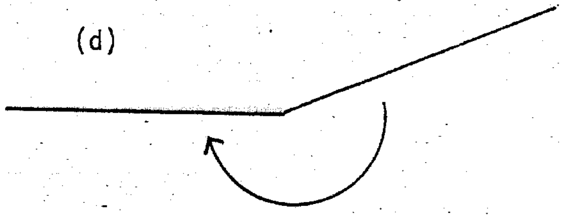
(b)



(c)



(d)



Task 2

Figure 1.2

Can you give one step for the series of Forward and Backward steps given?

Example:

FD 20, BK 10, FD 20 $\longrightarrow$ FD 30

(a) FD 30, FD 40, FD 30 $\longrightarrow$

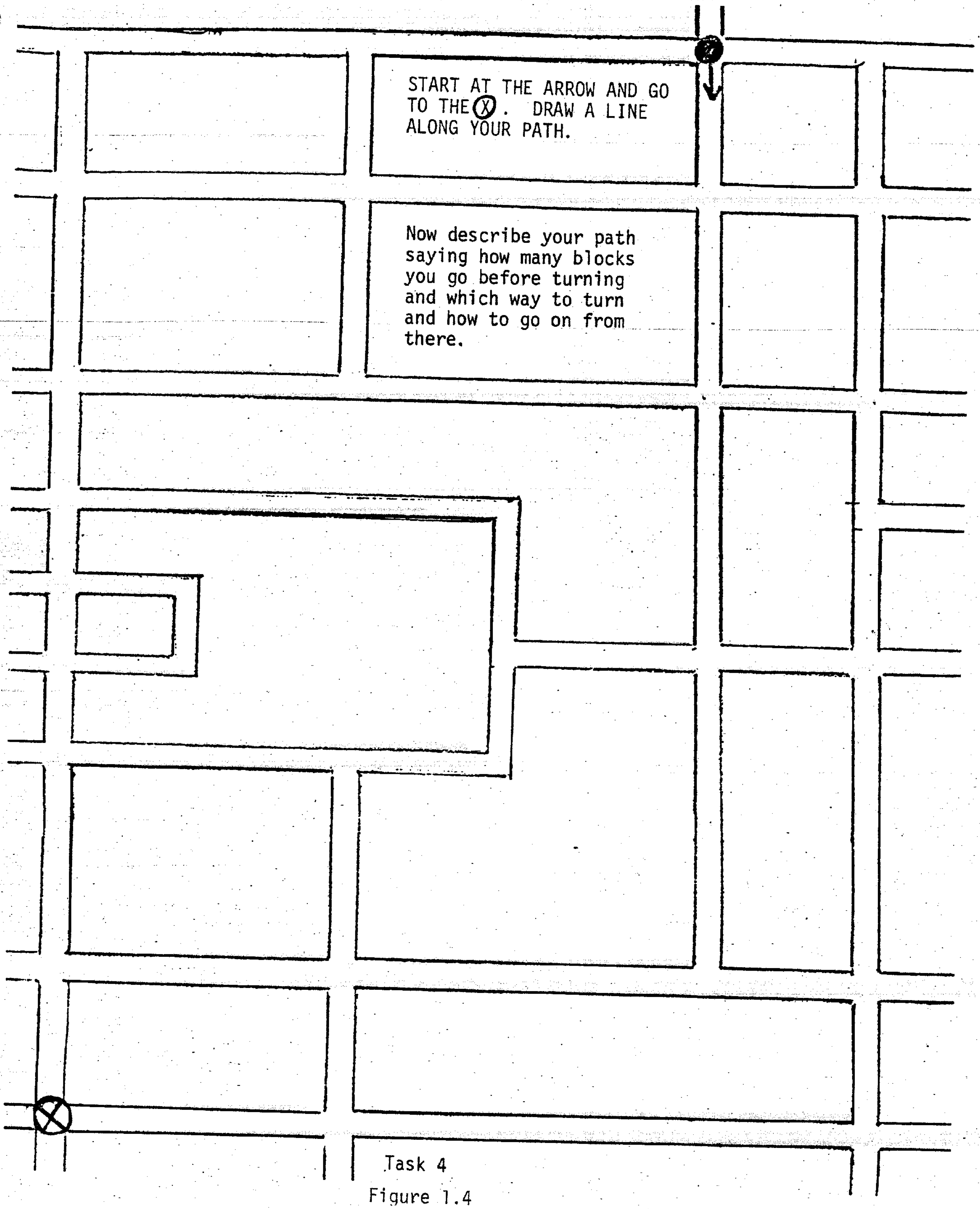
(b) FD 60, FD 80 $\longrightarrow$

(c) FD 80, BK 20, FD 30 $\longrightarrow$

(d) FD 50, BK 40, FD 50, BK 40, FD 50 $\longrightarrow$

Task 3

Figure 1.3



START AT THE ARROW AND GO TO THE ⓧ. DRAW A LINE ALONG YOUR PATH.

Now describe your path saying how many blocks you go before turning and which way to turn and how to go on from there.

Task 4

Figure 1.4

6. Acknowledgments

This project has been a collaboration between the public school system of Brookline, Massachusetts and the MIT LOGO Group. Without the full participation of a large number of people from both systems, the Lincoln School LOGO Project could never have occurred. Dr. Robert I. Sperber, Superintendent of Brookline Schools, and Gerry Cote, the Principal of Lincoln School, are greatly to be thanked for their help and cooperation. All three sixth grade teachers, Lisa Hirsh, Bob Lewis and Florence Regolino, enthusiastically supported the project and the participation of the sixth grade students who were its most immediate beneficiaries. The involvement of students and staff of the Brookline school system was an essential ingredient of the project.

Trial classes at Lincoln School were taught by Dr. Dan Watt, a Brookline teacher whose services were subcontracted to the project on a full-time basis. Dr. Dan Watt also had primary responsibility for collecting data about each student's progress and coordinating all the activities at the school.

The trial classes were planned and materials for them were developed by members of the LOGO Group, Professors Seymour Papert, Harold Abelson, Andy diSessa and Jeanne Bamberger. Dr. Sylvia Weir coordinated data collection and analysis. These members of the LOGO Group also participated in ongoing observation of the trial classes, revision of activities and in a series of project meetings which monitored the progress of the classes. MIT students Ellen Hildreth and Ursula Wolz also observed trial classes, collected data and participated in project meetings.

Our approach to data collection and analysis was developed over many years as an integral part of the research work of the Laboratory. The use of dribble files was initiated by Cynthia Solomon. The tradition of detailed class commentaries was developed by Solomon, Gerianne Goldstein and Kiyoko Okumura. The plan for this study was developed by the project staff in consultation with Professor George E. Hein and Stephanie (Penny) Dunning of the Program Evaluation and Research Group of Lesley College, who also assisted us in developing the pre/post interviews used with our experimental subjects. Ms. Dunning conducted the pre/post interviews, and observed the classes at regular intervals. Prof. Hein conducted classroom observations, interviewed the students' classroom teachers after each round of experimental classes, and participated in several project meetings.

An important aspect of the project was the way in which the computer culture was able to spread within the school. In addition to the experimental classes for

our sixteen subjects, LOGO classes were also provided for the rest of the sixth graders in the school. These classes were taught by volunteers, primarily MIT students, who were supervised by Dr. Watt, and Professors Abelson, diSessa and Bamberger. The volunteer teachers who contributed greatly to the overall success of the project were: Harvey Alcabes, Enno Becker, Kou-Mei Chuang, John Hengeveld, Danny Hillis, Rene Margulies, Margaret Minsky, Brian Schwartz, and Ursula Wolz. An after school computer club, led by MIT graduate student Jose Valente provided an opportunity for seventh and eighth grade students to learn LOGO, and for certain sixth graders to extend and deepen their LOGO knowledge. In addition to teaching, these students contributed beyond measure to the intellectual atmosphere surrounding the project. Since it was in the nature of this project to be "conservative" in developing a version of LOGO that could be integrated into schools in an immediate horizon and in the nature of these students to be "revolutionary" in looking beyond such immediate horizons, many of the student generated ideas have not found a place in this report. They will make themselves increasingly felt in future research directions.

The project's technical components including hardware and software support were carried out by Ron Lebel and Brian Silverman.

Data analysis was carried out by the entire project staff. Our analysis owes a great deal to discussions with Professors Hermine Sinclair-de Zwaart of the University of Geneva and Guy Groen of McGill University, both of whom visited classes, attended project meetings, and commented critically on our first attempts at analysis.

We had a number of visitors to our experimental classes, many of whom offered helpful comments. Two visitors whose comments were especially useful were Cynthia Solomon and Kiyoko Okumura- Montpetit, both highly experienced LOGO teachers and innovators. Cynthia Solomon conducted a follow-up study in the school during the subsequent (i.e. the current) school year whose results will be published in due course. Another kind of insight was contributed by Susan Hartnett who attended a number of classes and filmed one as part of an experiment in the use of film to capture aspects of the students' work. Perhaps the most disappointing aspect of the project was its inability, imposed by lack of resources, to follow up on new insights contributed by visitors. However, many of them have been absorbed into future oriented thinking.

Behind the scenes at all times was Gregory Gargarian who is listed in official papers as "secretary" but who in fact did enormously more than can properly be described by that (or perhaps any other) term. In addition to orchestrating a complex, movement of machines, people and texts so that the right ones would be

in the right places at the right times he was the person who was most in touch with the day to day needs of the project and of the people in it.

An Interim Report, describing the work of the first eight students, and containing our first attempt at data analysis, was written by the entire project staff, and coordinated by Sylvia Weir. This report was published as LOGO Memo #49 in the spring of 1978.

This final project report was written by Prof. Seymour Papert, Dr. Dan Watt, Prof. Andy diSessa and Dr. Sylvia Weir, under the overall direction of Prof. Papert. Part III, containing individual student profiles was written by Dr. Watt.

Editorial contributions were made by John Berlow.

2. Learning Styles in the LOGO Environment

In this chapter and in Chapter 3 which deals with "exceptional" students, we present brief summaries of the learning experiences of several of our experimental subjects. In presenting the different learning processes of these individuals we describe varied approaches to geometry, computer programming, choice of projects, planning and debugging and problem solving that are available to students of LOGO. Each summary has been chosen to illustrate a particular approach to LOGO. The work of these students and of the rest of our experimental sample is described more fully in Part III of this report, in which detailed, comprehensive profiles of the work of each of our sixteen subjects are presented.

We present here a summary of the experiences of four students, Kathy, Deborah, Donald and Kevin. Kathy specialized in a "bottom-up" approach, building up complex designs from a set of modular subprocedures. Deborah also made use of a "bottom-up" approach although hers was based on exploring the effects of long sequences of direct commands, selecting successful designs and copying the steps to create procedures. Eventually Deborah learned to make use of subprocedures and planned and carried out a major project. Donald worked in a "top-down" manner, starting with a definite plan, writing a superprocedure to define his project, and carefully planning each sub-procedure before carrying it out. Kevin's work fit somewhere between Donald's and Deborah's; his plans were more vague than Donald's and he was willing to alter them as he worked, while creating structures that helped him carry out his projects.

1. Kathy: A Modular, Bottom-Up Approach to LOGO Activities.

Kathy was a student with a subtle sense of humor who derived a great deal of intellectual pleasure from her work. When difficulties were encountered, she preferred to resolve them on her own, although without a great deal of persistence. When she did ask for help, she usually accepted the teacher's suggestions, and readily learned new ideas in the context of the projects in which she was engaged.

Kathy carried out dozens of small projects in the course of her LOGO experience. She shifted back and forth between open ended explorations and small goal directed projects. Her favorite activity was to repeat and combine existing procedures to produce unexpected results. Often she would interrupt an exploration to pursue a particular idea which had been suggested to her by the designs she had just created.

One of the ways in which Kathy structured her work was in her choice of procedure names. Her procedure names often indicated the relationship between a new procedure, and the subprocedures from which it had been built. Thus, a symmetrical design was called BARN because it was built by repeating a subprocedure called HORSE. A procedure called WORMY was made by doubling all the sizes in a similar design called WORM. And, in a rare example of top-down naming, a procedure called MONSTER was made up of subprocedures MO, NS, and TER.

Kathy's approach to her work is exemplified by a series of small projects which made use of a BOX and a TRIANGLE procedure as fundamental building blocks. The BOX and TRIANGLE procedures were constructed during periods of careful, goal-directed explorations.

```
TO BOX
1 FORWARD 100
2 RIGHT 90
3 FORWARD 100
4 RIGHT 90
5 FORWARD 100
6 RIGHT 90
7 FORWARD 100
END
```

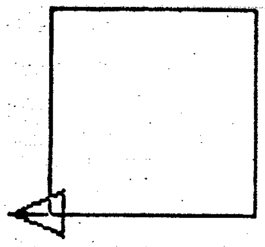


Figure 1.1

```
TO TRIANGLE
1 LEFT 90
2 FORWARD 100
3 RIGHT 120
4 FORWARD 100
5 RIGHT 120
6 FORWARD 100
END
```

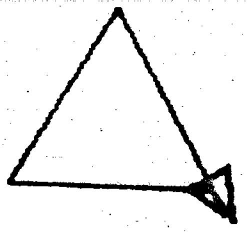


Figure 1.2

Box was the first procedure completed by Kathy and her group, and they immediately followed by constructing a series of figures making use of BOX as a subprocedure. (See examples in section 1.3.2 of Chapter 5). It was quite natural for her to repeat TRIANGLE as well. She was pleased with the result, calling it BUTTERFLY. She then repeated BUTTERFLY six times until the figure "closed". This new design she called 7BUTTERFLY (reflecting an initial miscount of how many repeats of the BUTTERFLY procedure she had used).

```
TO BUTTERFLY
1 TRIANGLE
2 TRIANGLE
END
```

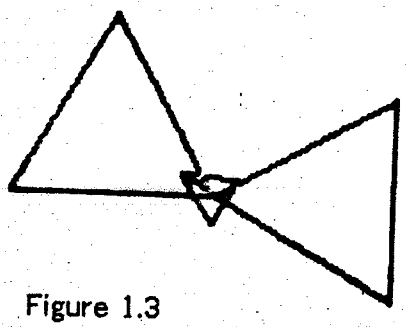


Figure 1.3

```
TO 7BUTTERFLY
1 BUTTERFLY
2 BUTTERFLY
3 BUTTERFLY
4 BUTTERFLY
5 BUTTERFLY
6 BUTTERFLY
END
```

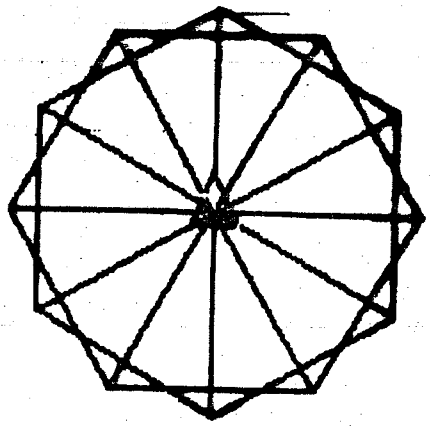


Figure 1.4

Following her initial exploration with triangles, Kathy's teacher suggested that she put her TRIANGLE and BOX procedures together to make a "house". After some goal-directed exploration, the HOUSE procedure resulted. Kathy immediately repeated HOUSE four times (calling this new procedure HOUSE4) until the design closed. Next she wanted to see how her HOUSE4 and 7BUTTERFLY designs would go together. She named the result HB47, indicating its relationship to HOUSE4 and 7BUTTERFLY.

```
TO HOUSE
1 TRIANGLE
2 RIGHT 30
3 BOX
END
```

```
TO HOUSE4
1 HOUSE
2 HOUSE
3 HOUSE
4 HOUSE
END
```

```
TO HB47
1 HOUSE4
2 7BUTTERFLY
END
```

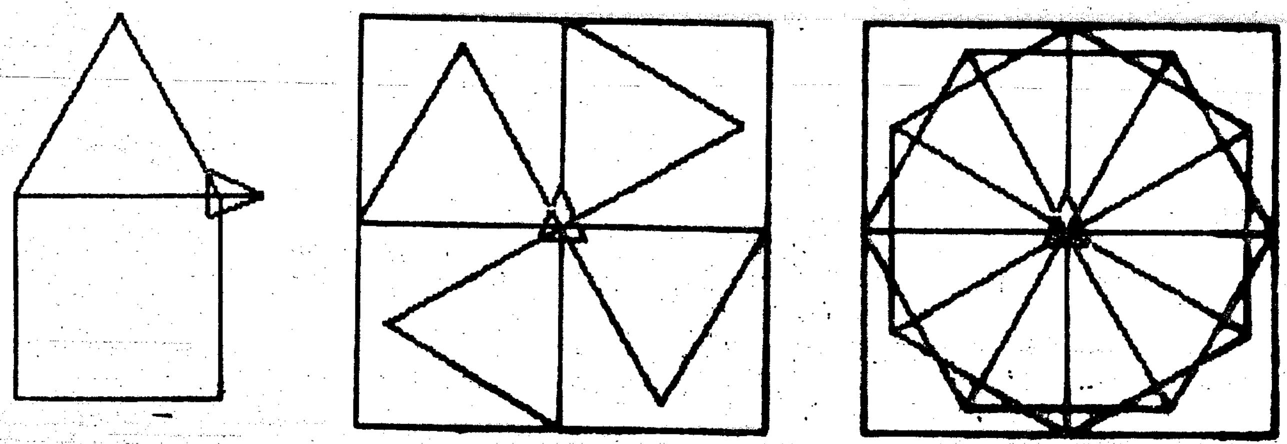


Figure 1.5

This set of projects culminated when Kathy declared that HB47 "looks like a spider," and returned to goal-directed activity, adding a series of circles to the design, to produce the procedure SPI.

```
TO SPI
2 RCIRCLE 30
3 LCIRCLE 30
4 RCIRCLE 20
5 LCIRCLE 20
6 BK 30
7 RCIRCLE 10
8 LCIRCLE 10
END
```

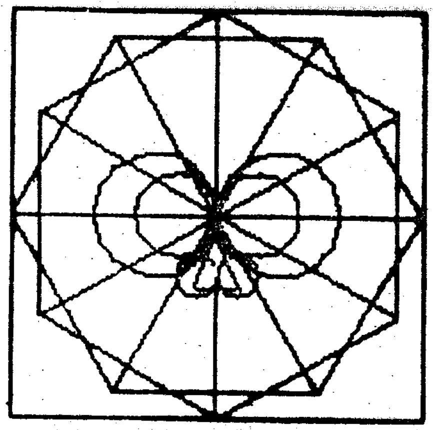


Figure 1.6

Although Kathy had constructed HB47 and its subprocedures by repeating simple shapes over and over to make a symmetrical design, she was also able to make explicit use of both right/left symmetry and similarity of shape in the process of constructing her "spider". It was this combination of more or less random explorations involving existing procedures, with expert use of heuristics such as similarity and symmetry when working in a goal-directed manner, that most commonly characterized Kathy's work.

2. Deborah: A Contrasting "Bottom-Up" Approach

Unlike Kathy, who is a bright student, successful in all of her academic work, Deborah is considered to be a "slow learner". In class she often appears to be withdrawn, indifferent to the subject matter or to her fellow students. When she began work in LOGO, she was totally dependent on the teacher -- requiring his reassurance on matters as routine as when to type a carriage return.

Deborah was able to build her confidence and understanding slowly by limiting her choices of LOGO commands and inputs, limiting the goals of her work, and by working in a way that minimized the chances of error. It was as if Deborah invented an unstated set of rules governing her work in LOGO which helped her to be successful.

Deborah used as few different commands as possible in her work. Basic TURTLE commands along with RARC and LARC were almost the only commands she used. For inputs to TURTLE commands, she used only multiples of 10, up to 100. If a larger effect was needed, she would use additional steps, as in FORWARD 90, FORWARD 30. In fact, Deborah began by using only inputs of 30, and gradually expanded to include other numbers, while continuing to use 30, 60 and 90 as her favorites.

Deborah's patience in a one-step-at-a-time mode of operation was quite remarkable. Her format was quite stereotyped. (1) carry out one TURTLE step (turn, move or penup); (2) check to see if that looks right on the screen; (3) if so, write down the step and continue; (4) if not, clear the screen, retype all the steps previously written down and try another choice for the questionable one.

While Deborah began her LOGO experience by asking for help at literally every turn, she had a deeply engrained resistance to new ideas or concepts. For a long time she rejected the use of subprocedures, although that modification in strategy would have greatly expanded her possibilities. It was as though she deliberately provided herself with a very definite and restricted "microworld" in which to operate.

On the other hand, the "microworld" which Deborah chose for herself, the world of FORWARD 30 RIGHT 30, is very nearly as rich as all of Turtle Geometry. It includes squares, triangles, "circles," stars," "men," "rabbits," and a variety of abstract designs, as well as the mathematical concepts of perpendicularity, inverse operations, the Total Turtle Trip theorem, symmetry, similarity, estimation of lengths and angles, planning and debugging, and procedure writing.

By limiting her inputs to numbers such as 30, 60 and 90, Deborah enhanced the possibility that her explorations would produce interesting results. At the same time, she seemed to have a high degree of visual intuition, often choosing precisely the correct input to produce a desired effect. For example, during one LOGO session the class watched a film which featured computer designs, among them, a six-pointed star. When Deborah came back to class, she drew a six-pointed star with the computer, without making a single mistake. She began by turning the TURTLE RIGHT 30, and used a combination of FORWARD 70s and RIGHT 60s to complete the star. The actual rotations required to construct the star were RIGHT 120 at the points, and LEFT 60 at the inner vertices. The way Deborah accomplished these rotations was quite typical of her work. After each forward step, Deborah would turn the TURTLE RIGHT 60. She kept turning it RIGHT 60, until the TURTLE was headed in the right direction. This required two repeats of RIGHT 60 at each point, and five repeats of RIGHT 60 at each inner vertex. At one inner vertex she missed the correct orientation, and calmly repeated RIGHT 60 for a total of eleven times until the TURTLE was aimed in the right direction. When she copied the steps in her notebook, she copied all eleven RIGHT 60s without any hesitation.

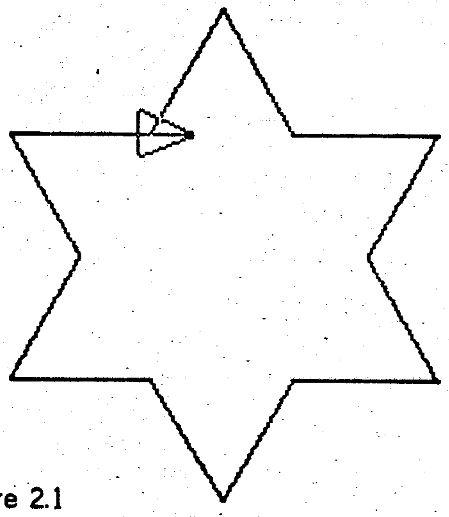


Figure 2.1

It was not until quite late in the series of classes that Deborah was ready to undertake a major project. She drew a picture of a rabbit in her notebook, and asked the teacher if he thought that would make a good project. He suggested modifying the rabbit, to make use of straight rather than curved lines, and redrew the picture for her in more simplified form.

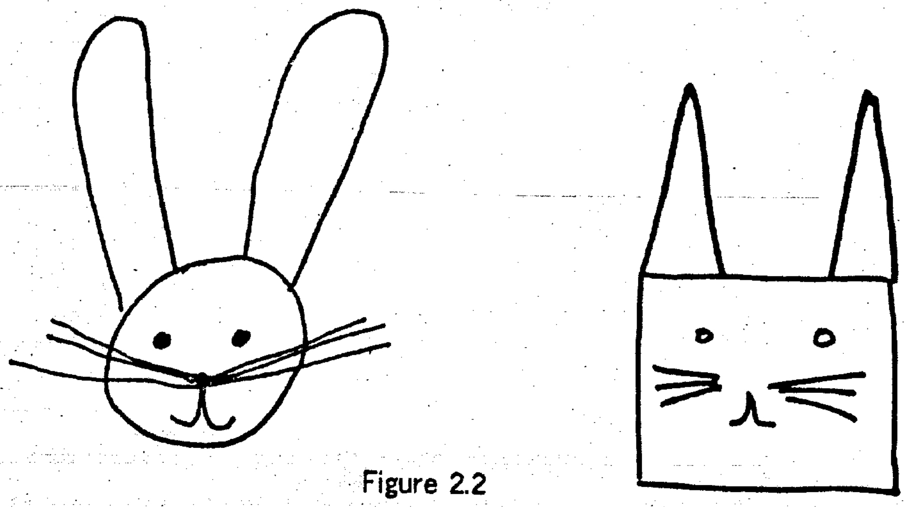
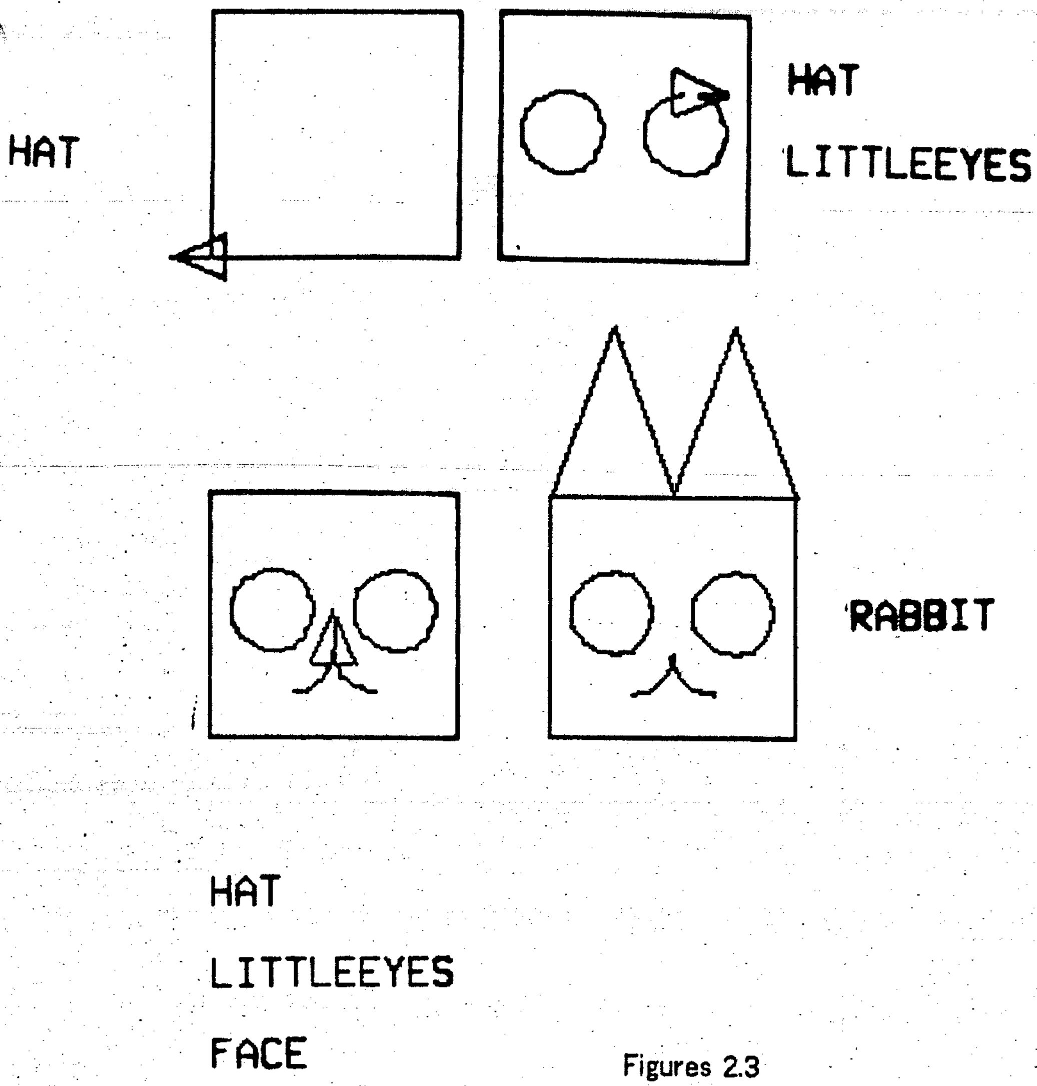


Figure 2.2

Although Deborah began by trying to draw the rabbit as a long series of commands, she quickly accepted her teacher's suggestion that she break the problem into parts, and make each part a separate subprocedure. While her work was directed toward an overall goal, and involved a certain amount of "top-down" planning, she constructed the rabbit piece by piece, in her usual exploratory fashion. Once again her choice of inputs to FORWARD and RIGHT comands were such that it was relatively easy for her to make the design come out the way she wanted. Without any apparent planning, she chose the length for the sides of the rabbit's head (FORWARD 90 FORWARD 30) in a way that made it easy for her to locate the eyes and nose symmetrically. The angles and lengths she chose for the ears -- a departure from her usual 30 or 60 degrees -- resulted in almost perfect symmetry. Details of the project are given in the profile of Deborah's work, in part III of this report.



```
TO RABBIT
5 HAT
10 LITTLEEYES
15 FACE
20 PENUP
25 FORWARD 70
30 FORWARD 3
35 RIGHT 20
40 PENDOWN
45 EARS
50 RIGHT 90
55 FORWARD 50
60 FORWARD 3
65 FORWARD 60
70 FORWARD 5
75 FORWARD 3
80 RIGHT 90
85 RIGHT 20
90 EARS
END
```

Figures 2.3

With the completion of her rabbit project, Deborah had almost totally reversed her initial feelings of dependence and incompetence. She invited her parents, teachers and school principal to visit the computer lab, and in many ways, demonstrated to her visitors and classmates her new found sense of confidence, satisfaction and power.

3. Donald: A Structured, Top-Down Approach to LOGO Activities

Donald provides a striking contrast to both Kathy and Deborah. Donald's work was characterized by a strong component of advanced planning, and the creating of structures within which problems could be solved. At the same time, Donald was quite ineffective at the visually-based, exploratory modes of problem solving which were so useful to Kathy and Deborah. He had difficulty estimating angles, and making use of the visual feedback provided by his explorations to improve his next attempt.

Throughout his work Donald was extremely receptive to suggestions from the teacher, often making use of new ideas before he fully understood them. In this way, he was able to incorporate into his way of working, strategies that would continue to prove useful, as he gradually came to understand them through use in more than one context. He seemed to have the confidence that he could make use of the teacher's suggestions effectively and that he would eventually understand them, even if the concepts were a bit hazy at first.

As an example of both the effectiveness and drawbacks associated with Donald's structured planning approach as well as of his difficulties with visual approaches to problem solving, we consider his construction of a "house" from a square and a triangle, a common LOGO task, tackled by many students at an early stage of their LOGO experience.

At first Donald attempted an exploratory approach to solving this problem. He began by drawing a triangle on the screen, making use of TRI, a state transparent equilateral triangle procedure:

```
TO TRI
1 FORWARD 100
2 RIGHT 120
3 FORWARD 100
4 RIGHT 120
5 FORWARD 100
6 RIGHT 120
END
```

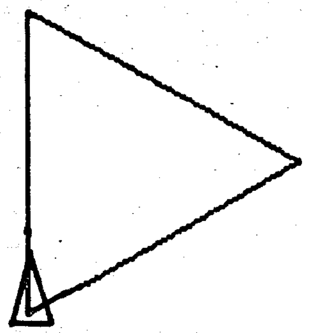


Figure 3.1

Having started with the triangle, the framework Donald established for solving the problem involved changing the "normal" orientation of a "house," to correspond to the initial orientation of the triangle. When Donald was asked to draw a picture of what he was trying to accomplish, he made this diagram:

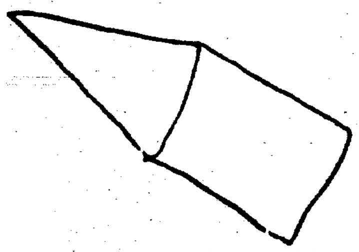


Figure 3.2

Since he was now dealing with two disorientations, the gap between the TRI and BOX procedures, and the tilted orientation of the entire shape, Donald had more difficulty than he could handle, and in an entire period of exploration, he never succeeded in resolving the problem in this form.

At the next class, the teacher suggested that Donald draw the BOX first. This suggestion provided him with enough new insight to devise a plan for solving the problem. Donald's plan allowed him to avoid the usual problem of finding the rotation needed to attach the triangle to the upper left hand corner of the box.

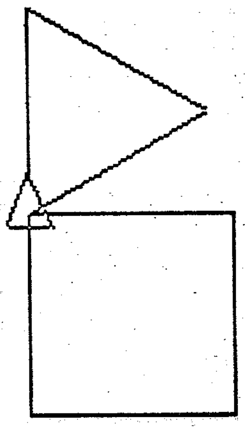


Figure 3.3

Instead, Donald moved the TURTLE to the upper right hand corner of the box, reversed its direction, and then drew the triangle so that its first side was along the top of the box.

```
TO HOUSE
1 BOX
2 RIGHT 90
3 FORWARD 100
4 RIGHT 90
5 FORWARD 100
6 RIGHT 180
7 TRI
END
```

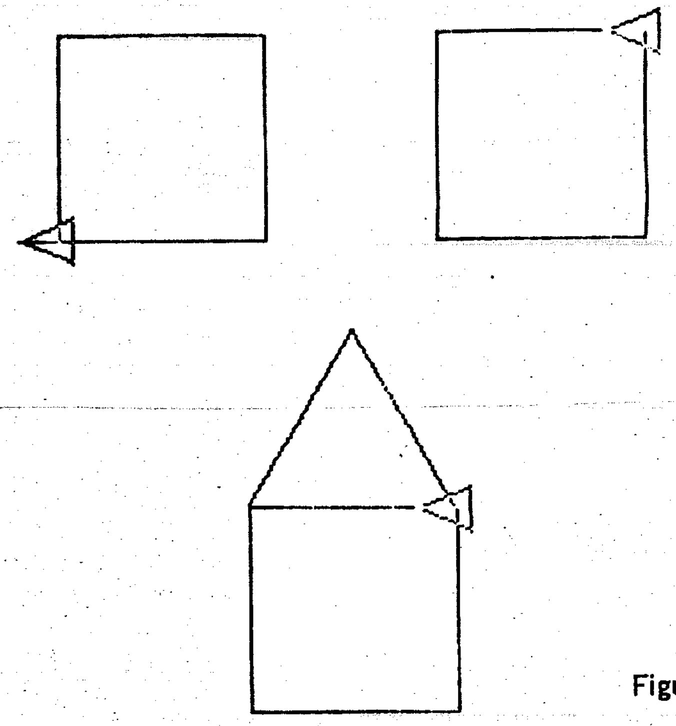
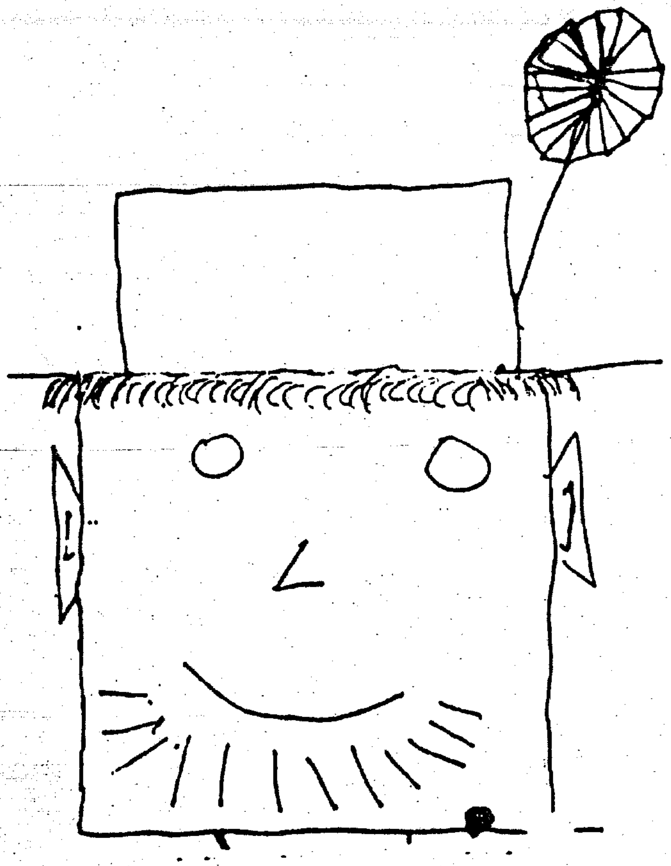
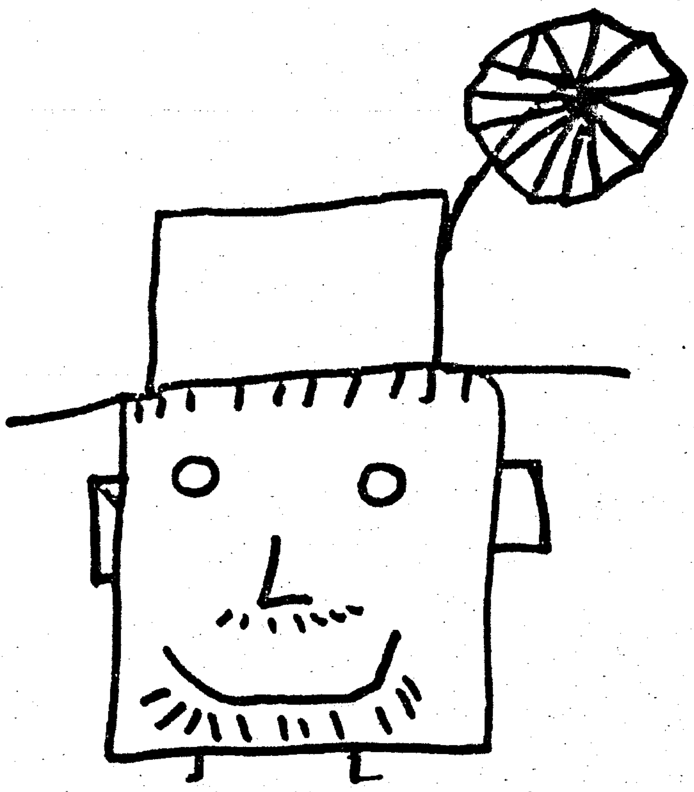


Figure 3.4

Donald had one major project which occupied him for more than 12 class periods -- more time than any other student devoted to a single project. This project began with a plan, a cartoon-like drawing of a man's head, which formed the basis of Donald's work. After one brief session of exploratory work, Donald revised his plan, and worked with his teacher to create a superprocedure, designed to draw the entire figure.



Donald's First Plan
Figure 3.5



Donald's Revised Plan
Figure 3.6

Donald's original superprocedure, TO HEAD, included the first six features of the head: the outside (BOX), EYES, NOSE, MOUTH, BEARD and HAIR. Once the superprocedure was written, each subprocedure became a mini-project, requiring one or two classes to resolve. Each feature of the head required it's own construction plan, a combination of the analysis and exploration needed to carry it out. While working on the features of his head, Donald make use of a great deal of teacher assistance -- especially in developing approaches to geometric analysis that were necessary to overcome his difficulties with visual problem solving.

In the course of his work Donald encountered estimation of distances and angles, the geometry of arcs and circles, the Total Turtle Trip theorem, and the use of both grid-based and intrinsic coordinate systems. He learned to use subprocedures and sub-subprocedures, to use patterned procedures making use of a REPEAT command, to make use of variables to control the size and shape of his "hat" and "flower" and to use a POLY procedure with a conditional stop rule.

Although Donald only "learned" these approaches to the extent necessary to solve the particular problems inherent in his project, each succeeding use of the same concept, reinforced his exposure to it, deepening his sense of mastery.

Donald's final figure, drawn by the superprocedure, HEAD, represents an almost literal translation of his revised plan, figure 3.6, into a computer program.

```
TO HEAD
1 BOX
2 EYES
3 NOSE
4 MOUTH
5 BEARD
6 HAIR
70 EARS
80 HAT
85 FLOWER
END
```

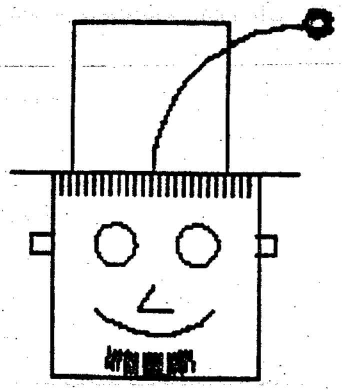


Figure 3.7

4. Kevin: An Expert "TURTLE Driver"; an Intermediate Example

In his LOGO work, Kevin combined certain qualities that were present in the styles of Kathy, Deborah and Donald. What particularly distinguished Kevin's style from that of the other three students discussed in this chapter was his superior ease and comfort in manipulating the TURTLE -- both in moving the TURTLE from place to place on the graphics display screen, and in finding ways of combining and simplifying series of turtle steps, to facilitate his work.

Like Deborah, Kevin worked in a step-by-step fashion, taking careful notes as he worked. Unlike Deborah, Kevin was able to examine his lists of steps, (combining FORWARD 150, BACK 10 into one step, FORWARD 140, for instance) eliminating unnecessary steps. Like Deborah, Kevin chose inputs to LOGO commands and procedures very effectively. Unlike Deborah, he often made connections between the inputs he used. For example, when drawing a right isosceles triangle he began with a rotation of 45 degrees. When asked why he chose 45 degrees, Kevin had two responses: first, "it looked about right" and second, "45 is half of 90".

```
TO OF
1 RT 45
2 FD 100
3 RT 90
4 FD 100
5 RT 135
6 FD 140
END
```

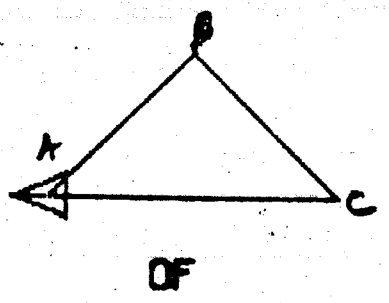


Figure 4.1

Kevin was also like Donald in that he usually engaged in goal directed work, including one long term project, and in that he made use of particular structures to help him resolve issues that arose in carrying out his projects. Unlike Donald, he did not participate in creating those structures for himself, nor did he engage in any significant advanced planning. In his major project, drawing a large "turtle", Kevin learned to use subprocedures to break his problem into manageable "chunks" or when he needed to repeat the same procedure more than once as part of his project. The names Kevin chose for his procedures and subprocedures sometimes had a random quality, such as the name OF for a procedure which drew a triangle, or the name LIFS for a procedure that drew a set of nested squares.

Kevin's project to draw a large "turtle" provides examples of the way in which utilized various structures to help with his work. Having drawn a circle of radius 90, to form the "shell" of his "turtle", Kevin made use of an arc procedure that turns through a variable angle, as a way of moving around the "turtle's" shell. He had noticed that the small dots which appeared when the circle and arc procedures were used occured at intervals of 10 degrees. Using this discovery he created a system of intrinsic coordinates for the purpose of moving around his "turtle's" shell. He would count the dots, and use an angle input of 40 to move a distance of 4 dots along the shell, etc.

Another structure which Kevin used to draw his "turtle", was the creation of the modular subprocedures, FOOT and BKFOOT. Kevin would move the LOGO TURTLE around the "turtle's" shell using an arc command until he reached the point where he needed to locate a "foot" of the "turtle". The two procedures, FOOT and BKFOOT, were equivalent to a state transparent procedure which drew a foot of the "turtle" and returned the TURTLE to the shell, ready to move around to the next point at which a foot would be located.

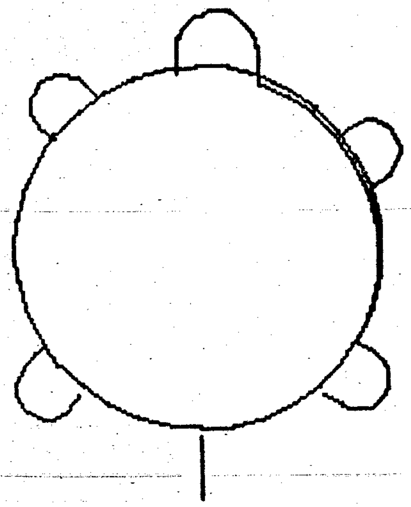


Figure 4.2

The last set of projects that Kevin worked on provides a nice example of the way in which Kevin came to understand the use of variables and stop rules. After having explored the effects of different inputs to a POLY procedure (see Chapter 5, Section 1 for a discussion of POLY), Kevin built a design by keeping the angle constant while varying the size of the POLY. The teacher helped him talk through this procedure:

```
TO TUNNEL :SIZE
10 POLY :SIZE 45
20 IF :SIZE - 105 STOP
30 TUNNEL :SIZE+5
END
```

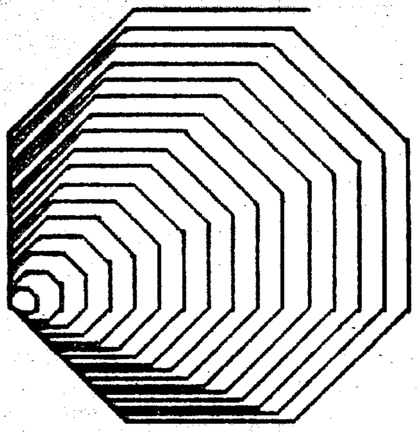


Figure 4.3

Next, he repeated the process with an angle of 90 degrees:

```
TO LIFS :SIZE
10 POLY :SIZE 90
20 IF :SIZE = 150 STOP
30 LIFS :SIZE + 2
END
```

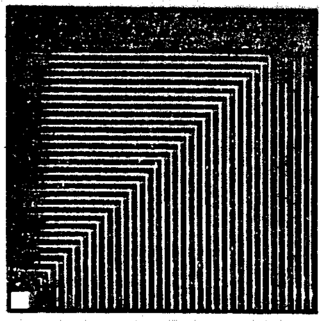


Figure 4.4

Kevin then asked if "the amount the POLYs grow each time" could be changed, and if the largest size could be changed. He picked the variable names "SET and "LARGE for these quantities, and with some syntax help from his teacher, wrote the procedure:

```
TO UFC :SIZE :SET :LARGE
10 POLY :SIZE 90
20 IF :SIZE = :LARGE STOP
30 UFC :SIZE + :SET :SET :LARGE
END
```

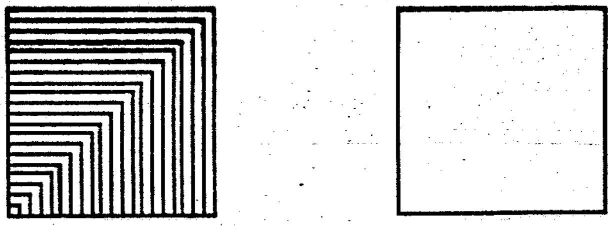


Figure 4.5

In the course of exploring the use of different inputs to this procedure, Kevin was delighted to discover that keeping all the inputs the same had the effect of producing a variable sized square. He understood that the reason the procedure drew only one square was that the starting and ending sizes were identical.

Kevin's major difficulty in using the computer was an initial reluctance to plan ahead or to structure his work more than one step at a time. When new ideas were presented to him in a way that enabled him to simplify his work, he was able to absorb them relatively painlessly and incorporate them into his thinking.

5. Learning Styles of the Other Students.

We have given a description of the differences in learning style among four different students carrying out very similar tasks in order to provide a sense of the differences that existed among all 16 of the students. To offer a better some sense of the scope of these variations among the group, we will attempt a differentiation of the rest of our 16 students based on how their approaches compared with those of our models. Since this greatly oversimplifies the different ways in which these students worked, the reader is urged to read the full profiles in Part III of this report, and the descriptions of the work of exceptional children presented in the next chapter, to get a better sense of the flavor of each student's particular LOGO experience.

Monica and Darlene showed a lot of similarity to Kathy's style of work. Both made a lot of use of repetition to create designs, and preferred small, easily completed explorations to long term projects. Neither had Kathy's interest in relating the procedure names that they used to the ways in which their procedures were constructed. Darlene was unusually curious about the possibilities inherent in the LOGO language, and in the computer system we were using, and explored a lot of different kinds of projects, without settling on any major area of interest.

Although Jimmy was a quicker and more articulate learner than Deborah, his work was similar to hers in many respects. He worked in a linear step-by-step fashion, limiting himself to a few key ideas in carrying out LOGO tasks. He had a great deal of resistance to suggestions for new ways of doing things, and had difficulty making effective use of subprocedures. Jimmy was also like Deborah in his excellent intuitive manipulation of the TURTLE. Jimmy's work had a much finer level of detail than Deborah's, and like Kathy, he was able to make excellent use of symmetry and similarity in planning his steps.

Dennis and Harriet showed a certain amount of similarity to Donald, in the way that they made use of top-down approaches and in the structures they needed to carry out their work. Dennis also had difficulties solving problems by visual exploration, similar to those encountered by Donald. Harriet had no difficulty with visual explorations but preferred to work on challenging tasks in which the visual effects were less important than other aspects of programming. In particular, Harriet carried out two elaborate interactive projects -- designing a tictactoe game, and writing a "madlib" program. Although Harriet needed help with the syntax and programming knowledge to carry out these tasks, she was able to understand how to create complicated structures for her projects, once she was given a model of how they functioned.

Laura probably belongs in a category of her own. Her projects tended to involve large scale designs, which required complex strucures. Although her creative ideas were sophisticated and unusual, she lacked the patience and insight necessary to develop the structures she needed to carry them out. Since Laura did not like to ask for help, or to appear to be having difficulties, she experienced some frustration from time to time.

Karl, Albert and Betsy all had a degree of similarity to Kevin, in that they utilized a mixture of top-down and bottom-up approaches, and that they were able to make use of the particular structures needed to carry out a task without necessarily understanding those structures in a more general way. Karl and Albert tended to prefer shorter projects, while Betsy usually worked on more complex projects, making use of a number of subprocedures.

Gary, Tina, and Ray worked in more unique ways that cannot easily be compared to those of our four "model students," Their work will be examined, along with Karl's, in Chapter 3 dealing with exceptional children.

3. The Experience of Exceptional Children in the LOGO Learning Environment

One of the most striking results of the Brookline LOGO experiment has been the success experienced by exceptional students. We include within the category "exceptional," two groups of students whose education often poses problems within conventional elementary school programs -- intellectually gifted students, and students with significant learning disabilities. While the educational difficulties encountered by these groups of students differ markedly, both groups experience difficulties related to their inclusion in educational programs designed primarily for average students.

Faced with a choice between mainstreaming these students, and isolating them in special programs, schools have usually decided that the disadvantages of isolation outweigh the possible advantages of specialized programs. Even when educational programs are partially individualized teachers continually confront the task of providing challenges and enrichment for gifted students and tutorial and remedial help for the learning disabled, while including them in an overall academic program designed for students of an average range of abilities. The successful LOGO experiences of exceptional students, working side by side with students of average ability, indicates that a LOGO learning environment may prove useful to schools in meeting the problems posed in educating these students.

It should not be surprising that intellectually gifted students were successful in LOGO classes -- many computer education programs have been targeted for bright students who have generally been successful in learning to program computers. What is surprising is that the students with the lowest level of previous academic success should also be successful in an educational context involving a full range of students working together. In this chapter we will describe the work of three exceptional students, Gary, Karl and Tina. Gary was one of three students, in our experimental sample of sixteen students who are considered "gifted" by their teachers. Karl and Tina are two of the three students in our experimental sample who have been diagnosed as having "learning disabilities," and who receive a minimum of one hour of specialized one-to-one tutoring each day.

1. The Work of Gary: An "Intellectually Gifted" Student

Of all our experimental subjects, Gary seemed the most predisposed to success in LOGO. Combining a strong prior interest in computers with a learning style that encompassed both analytical and trial-and-error approaches, Gary was able to successfully carry out projects in a number of different areas.

Gary had had some previous exposure to computers, had attended a personal computer fair, and had been pressuring his parents to buy him a computer. He absorbed new ideas voraciously, and rarely had to be shown something twice. He tackled extremely ambitious projects, and always stayed with a project until some kind of completion was achieved.

1.1 Gary's Learning Style as Seen Through his LOGO Work

Gary's work demonstrated some clear characteristics that set him apart from most of the other students:

Gary easily understood the use of a procedure as an entity, recognizing the usefulness of naming a series of steps, and thereafter considering them as a "unit"; he often wrote procedures without trying out the steps individually first, recognizing that the series of steps could be considered to have a "total effect," as though it were a single command.

Gary had faith in his ability to solve problems by reasoning as well as trial-and-error. He was constantly trying things out "in his head", making use of a number of "abstract principles" to simplify and debug his work as he went along: A series of FORWARD and BACK commands would easily be combined into one command; left/right reversibility would be used to correct an error. At one point, for example, Gary had typed RIGHT 99, and saw from the effect that he should have used LEFT 99. He then used the computer to add 99 + 99, and typed LEFT 198. In writing his procedure later, he simply used the correct command LEFT 99, without ever having tried it explicitly.

Gary tended to "plunge into a problem" impetuously, with very little advance planning, drawing on a quick analysis, based on partially understood ideas. He then enjoyed the process of debugging his original idea, or of moving in a new direction, if his result was significantly different from what he had intended. In the first class, Gary noticed that repeating a simple three step procedure made a "pattern" that looked something like a "circle". When another student suggested making a smaller circle inside the first one, Gary began to make a new circle by using the same procedure with smaller inputs. When his new circle came out larger than the original one, Gary was delighted by the surprise. He then tested another approach -- make all the inputs exactly half of the original, which led to a "circle" of almost the same size as his first one. And so on, until he had tried many variations.

Gary tended to work in a step-by-step fashion, rather than make use of planning. While he generally had an overall idea of what he was trying to do, he tended to incorporate subprocedures one after another, rather than to break down his problem into parts and plan his subprocedures in advance. He showed that he was capable of using a more structured approach when asked by his teacher to rewrite his STARSHIP procedure. He reworked the problem and created a set of simple modular subprocedures to draw his starship design.

Gary often sought out bugs, testing for extreme situations: the largest possible inputs, the largest number of REPEATs, situations which would produce error messages, as a way of understanding both the capabilities and limitations of the computer, beyond the needs of any specific project on which he was working.

Gary's work was usually directed toward ambitious goals. He worked on four major projects, during his seven weeks of LOGO classes. While he enjoyed brief detours, such as the "circle" exploration described above, his work was usually directed quite specifically at his particular immediate goal. Between projects, he often appeared to be restless; once a new task was selected, he was off and running again.

1.2 Gary's LOGO Projects

Gary carried out four major projects, the last of which was still in process when the series of classes ended: a rather elaborate "face" built from a large number of subprocedures; an interactive "math quiz" which gave a user a series of two-digit addition problems; a computer animated "starship" design; and a "morse code translator" which was intended to translate a printed sentence into a line of morse code, and vice-versa.

His FACE project involved learning to use a large number of sub-procedures, and a great deal of Turtle Geometry -- especially arc and circle procedures. Gary used functional procedure names, abbreviated procedure names, and "nonsense" names, all in a rather elbaorate scheme to "hide" the sub-procedures which actually "did the job". (See Figure 1.1) Of course, this complex set of subprocedures was extremely difficult for Gary himself to debug, and he often had to trace through the entire "tree structure" of his project to find a bug in a particular procedure.

```
TO FACE     TO ENM     TO EN      TO M
10 ENM      10 EN      1 EYES     1 MOUTH
END         20 M       2 NOSE     END
            END        END

TO NOSE     TO EYES    TO MOUTH
1 F007      1 F006     10 F008
END         END        END
END

TO F006
1 F005
2 PENUP
3 LEFT 90
4 FORWARD 80
5 PENDOWN
6 RIGHT 90
7 RCIRCLE 45
8 PENUP
9 RIGHT 90
10 FORWARD 160
11 LEFT 90
12 PENDOWN
13 LCIRCLE 45
14 HIDETURTLE
END
```

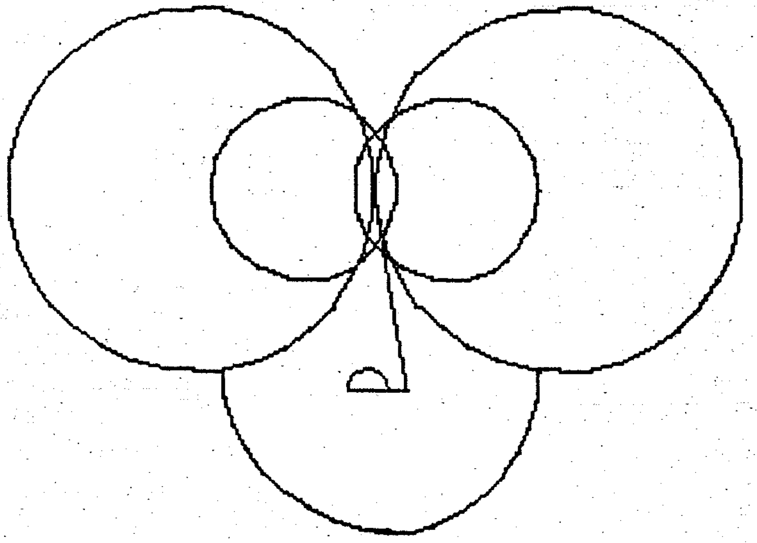


Figure 1.1

```
TO F005
1 LCIRCLE 90
2 RCIRCLE 90
END

TO F008
1 PENUP
10 PENUP
20 FORWARD 70
30 PENDOWN
40 RIGHT 90
50 PENUP
55 RIGHT 90
60 FORWARD 166
70 RIGHT 90
80 FORWARD 70
90 LEFT 90
100 PENDOWN
110 LARC 80
120 HIDETURTLE
END
```

```
TO F007
1 PENDOWN
2 S
3 FORWARD 100
4 RIGHT 99
5 FORWARD 30
6 RIGHT 90
7 RARC 10
8 RARC 10
9 HIDETURTLE
END

TO S
6
10 PENUP
20 SHOWTURTLE
30 LEFT 90
40 FORWARD 80
50 LEFT 90
60 LEFT 9
70 PENDOWN
END
```

Gary's second project, a math quiz, involved the use of conditionals, PRINT statements, the naming of variables, and random numbers. Although he planned to extend the project to include subtraction, multiplication and division, he decided to go on to other activities after completing the addition portion of the quiz.*

Gary's Starship project involved Turtle Geometry once again. (See Figure 1.2) In order to avoid the type of debugging problems he encountered in his FACE project, Gary decided to carry out his starship project by writing one long procedure. This led to a large number of unanticipated bugs as well. Although Gary successfully debugged his lengthy procedure, his teacher suggested that he redo his starship project, making use of simple procedures and subprocedures. This time, having experienced both extremes -- an unnecessarily complex hierarchy of subprocedures, and an unnecessarily long single procedure, Gary developed a set of modular, easily readable and easily debugged subprocedures to carry out his STARSHIP design.

*During the following year, Gary went back to this project and completed it, making use of a LOGO computer provided by his school system.

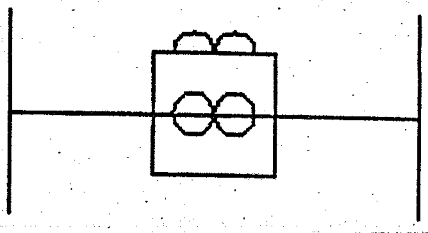


Figure 1.2

Old Starship Procedure

```
TO STARSHIP
10 RIGHT 90
20 FORWARD 100
30 LEFT 90
40 FORWARD 50
50 RIGHT 180
60 FORWARD 100
70 PENUP
71 LEFT 180
72 FORWARD 50
73 LEFT 90
74 FORWARD 100
75 RIGHT 90
90 LEFT 90
95 PENDOWN
100 FORWARD 100
105 RIGHT 90
110 FORWARD 50
120 LEFT 180
```

(continued, next page)

New Starship Procedures

```
TO STARSHIP
10 STA
20 WINGR
30 WINGL
END
```

```
TO STA
5 WRAP
10 C
20 LI 100
END
```

```
TO WINGR
10 MO
20 RIGHT 90
30 LI 50
40 MOVE
END
```

(Old Starship, cont.)

```
130 FORWARD 100
140 PENUP
141 RIGHT 180
142 FORWARD 50
143 RIGHT 90
144 FORWARD 100
145 LEFT 90
155 PENDOWN
160 RIGHT 90
170 FORWARD 30
180 LEFT 90
190 FORWARD 30
200 LEFT 90
210 FORWARD 60
220 LEFT 90
230 FORWARD 60
240 LEFT 90
250 FORWARD 60
260 LEFT 90
270 FORWARD 30
280 PENUP
290 LEFT 90
291 FORWARD 30
292 RIGHT 90
300 HIDETURTLE
310 PENDOWN
320 RCIRCLE 10
330 LCIRCLE 10
340 PENUP FORWARD 30
345 PENDOWN
350 RARC 10
360 RARC 10
370 PENUP RARC 10
380 RARC 10
390 LARC 10
400 LARC 10
410 HIDETURTLE
END
```

(New Starship, cont.)

```
TO WINGL
10 MOV
20 LEFT 90
30 LI 50
35 RIGHT 90
40 HIDETURTLE
END
```

```
TO MO
10 RIGHT 90
20 FORWARD 100
30 LEFT 90
END
```

```
TO MOV
10 LEFT 90
20 FORWARD 100
30 RIGHT 90
END
```

```
TO C
10 SQ.1
20 PENDOWN RCIRCLE 10
30 LCIRCLE 10
40 PENUP FORWARD 30
45 PENDOWN
50 REPEAT [RARC 10] 2
60 PENUP REPEAT [RARC 10] 2
70 PENDOWN REPEAT [LARC 10] 2
80 PENUP REPEAT [LARC 10] 2
90 BACK 30
END
```

```
TO LI :LE
5 PENDOWN
10 RIGHT 90
20 FORWARD :LE
30 LEFT 180
40 FORWARD 2 * :LE
50 RIGHT 180
60 FORWARD :LE
70 LEFT 90
END
```

```
TO SQ.1
5 PENUP
10 RIGHT 90
20 FORWARD 30
30 RIGHT 90
35 PENDOWN
40 FORWARD 30
50 RIGHT 90
60 FORWARD 60
70 RIGHT 90
80 FORWARD 60
90 RIGHT 90
100 FORWARD 60
110 RIGHT 90
120 FORWARD 30
130 RIGHT 90
135 PENUP
140 FORWARD 30
150 FORWARD 30
END
```

For his last project, Gary decided to create a Morse Code Translator as the first step of a project to actually transmit morse code over radio waves, which he had read about in a computer hobbyist magazine. In creating the Morse Code translator he had to make use of LOGO's list and word processing capabilities; of recursive procedures which used the concept of the empty word" and the "empty list" in STOP rules, and of conditionals which were used to decide which particular set of Morse Code symbols to output. (This set of procedures is discussed more fully in Chapter 4, Section 7.3.3)

1.3 Conclusions

Gary absorbed a great deal in approximately 25 hours of LOGO classes. His projects involved a number of different content areas: Turtle Geometry, Interactive programming, animation, list processing, etc. His enthusiasm remained at a fever pitch throughout the series of classes. When the cycle of LOGO classes was finished, Gary helped establish an after school "computer club," so that he could carry on his work.

The computer activities provided the kind of challenge and scope of intellectual activities that Gary wanted and needed to develop his abilities most fully. The fact that this challenge and scope was not always present his regular classes was attested to frequently by his classroom teachers. A LOGO capability in a classroom could help teachers meet the needs of students like Gary.

2. The Work of Karl: A Severely Learning-Disabled Student

Karl is a student who has been diagnosed by the school staff as having severe learning disabilities. Related to his difficulties in reading, writing and arithmetic, are readily observable hearing, speech and motor ability problems which interfere with communication. Karl who is large for his age and somewhat awkward in manner, has few friends among his classmates other than two or three selected "cronies."

In his LOGO work Karl demonstrated the ability to plan and carry out complex projects involving several subprocedures, to understand geometric concepts, to carry out mathematical calculations in his head, and to work in both a well organized step-by-step fashion and in an open-ended exploratory mode. While Karl often needed help with routine tasks such as remembering the spelling of LOGO commands, he was able to make use of reasoning abilities that allowed him to surpass in the LOGO classes what he was normally able to accomplish in either his regular classes or his special tutoring sessions.

2.1 Karl's Working Style in the LOGO Classes

Karl enjoyed the sense of control and accomplishment that he experienced in the LOGO classes. He had to overcome severe typing and spelling problems, and to find ways of organizing his work, in order to achieve this success. While spelling and typing difficulties made his work much slower and more painful than that of his peers, they did not seem to dampen his enthusiasm or impede his learning process.

Karl's work alternated between carefully planned geometric designs and a more random, exploratory use of commands that he did not fully understand. As he began to discover consistencies in the effects produced by different commands, he gradually came to exercise more purposeful control over the outcomes of all his work.

Karl created a number of planned geometric designs making use of direct commands and previously defined procedures. Once a procedure was completed, however, he enjoyed combining procedures and SPIN commands randomly to see the effect. He would then sit for long periods of time watching the different combinations. As the classes went on, he came to have more and more interest in controlling the designs, planning his combinations more carefully, and even editing his "conglomerate" procedures so that random effects were systematically eliminated.

Karl developed an experimental approach to using the computer system and the LOGO language -- he would "ask questions of the computer" by trying things and seeing what happened. He used a method of successive approximations to find the largest possible input to a SPIN command, and to determine the limits of the TURTLE screen. He found the shortest procedure name (one letter) and the "longest" (QWERTYUIOPASDFGHJKLZXCVBNM). He experimented with the adding and subtracting of extremely long numbers, to test the computer's limits in doing arithmetic. Karl also made a point of learning to use all the peripheral devices that were part of the computer system: the Floor TURTLE, plotter and printer. Although seemingly random, it is clear that these explorations were an important part of his effort to establish control over the environment in which he was working.

Karl made a major effort to exercise control over his typing difficulties as well. His typing was characterized by a painfully slow and poorly coordinated approach to using the keyboard. When he wanted to find a particular letter on the keyboard, he would scan with his eyes, moving his index finger back and forth, as his eyes shifted. Often his finger would pass the correct key several times before hitting it. If a wrong key was hit, an error message would result, and the whole process would begin again. Although he gradually improved his typing, he continued to have difficulty finding familiar keys.

Karl used the same random scanning strategy for finding numbers. Although he knew that the numbers were on the top line of the keyboard, he was unable to make use of their inherent order to make them easier to find. This was particularly striking when he was numbering steps in a procedure. Going in a sequence 1, 2, 3, 4, ... he conducted an individual "search" for each number, using his scanning and finger moving technique.

Karl also had difficulty reading error messages and spelling LOGO commands. Although he was gradually able to include the new terminology in his sight vocabulary, he was unable to sound out words, even though he had seen them before. He either "knew" a word, or could not read it. Before long, he had become familiar with the most common commands and error messages, and knew how to respond to them. He continued to have a problem whenever an unfamiliar error message appeared.

Karl developed his own strategies for overcoming these problems. He learned to write short procedure names and abbreviations for commonly used LOGO commands. He kept a notebook of all the commands and procedures that he had learned or created, so that he could easzily find the correct spelling if he couldn't remember it. At one point he named a procedure QWERTYUIOPASDFGHJKLZXCVBNM, utliizing every letter of the alphabet, in their sequence on the keyboard. Since he could move his finger in order across all the letters, this procedure name required no scanning and was easier for Karl to type than even a three letter procedure name such as TAM, which required scanning.

One of the Karl's major activities was an animated "car" project in which the TURTLE moved continuously, while it's motion could be altered interactively as the user typed certain keys on the keyboard. The letters he used to control the motion of the turtle -- Q W E R A S and F -- are all located in a group on the left hand end of the keyboard. Originally he had planned to make a cardboard cover for the keyboard, with a hole cut in it so that only those letters could be seen. He found this to be unnecessary, however. By concentrating his attention on one small corner of the keyboard, he was able to select the correct keys easily, without any of the scanning or memory problems that occurred when he had the entire keyboard as his field.

2.2 Some Examples of Karl's Work

Karl's earliest LOGO procedures were simple geometric designs. His first project, TAM, was a rectangle, and the second, CULL, resulted from repeating TAM four times. His third project, ACE, made use of symmetry, and the properties of circles in a carefully planned format.

```
TO TAM
1 FORWARD 190
2 LEFT 90
3 FORWARD 100
4 LEFT 90
5 FORWARD 190
6 LEFT 90
7 FORWARD 100
END
```

```
TO CULL
1 TAM
2 TAM
3 TAM
4 TAM
END
```

```
TO ACE
1 RCIRCLE 50
2 LCIRCLE 50
3 FORWARD 100
4 RCIRCLE 50
5 LCIRCLE 50
6 BACK 100
7 BACK 100
8 RCIRCLE 50
9 LCIRCLE 50
END
```

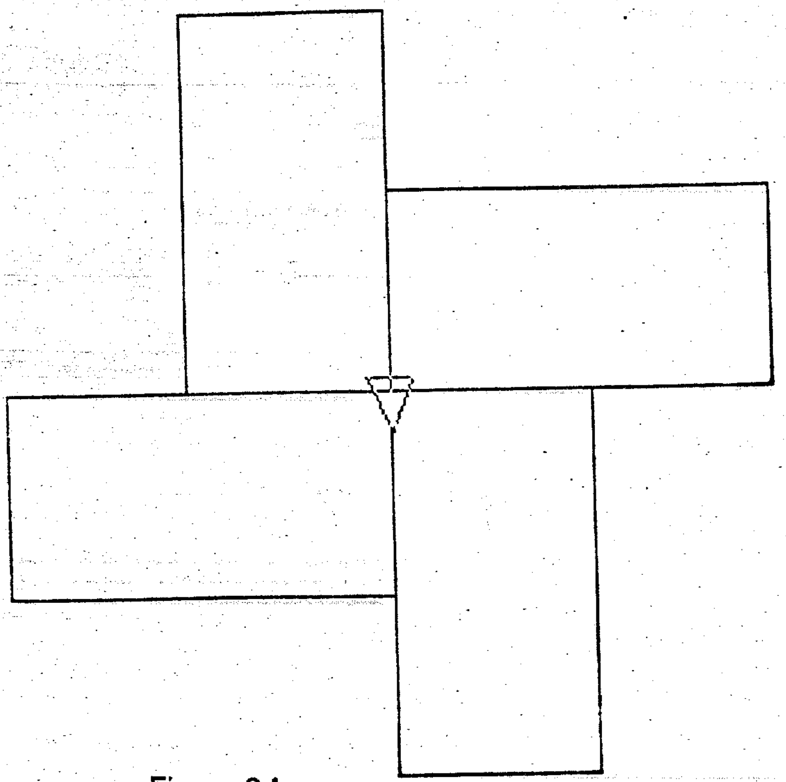


Figure 2.1

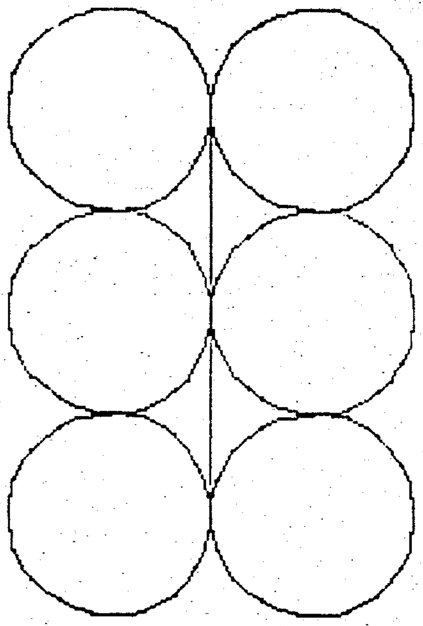


Figure 2.2

Having completed these designs, Karl's work entered a phase in which he created wildly spinning designs by random accumulation of previously defined procedures. ACE2, ME, and ACE3 are examples of this type of procedure which culminated in the procedure BU:

```
TO ACE2
1 SPIN 1020
2 ACE
END
```

```
TO BU
1 SPIN 200
2 CULL
3 ACE
4 ACE2
5 ME
6 NO
7 NO
8 XX78055
9 PLUS
10 TAM
END
```

```
TO ME
1 TAM
2 CULL
3 ACE
4 ACE2
END
```

```
TO ACE3
1 SPIN 1020
2 TAM
3 ACE
4 ACE2
5 ME
6 HIDETURTLE
END
```

At this point, Karl began to assert more control over the effects of his work. He spent an entire class period editing BU, producing an elegant spinning design by going through the procedure step by step, systematically eliminating all random effects.

```
TO BU
1 SPIN 200
2 CULL
3 ACE
4 HOME
9 PLUS
END
```

2.3 Karl's Animation Project

Karl's major accomplishment was a set of procedures which animated the turtle in such a way that he could "drive it around" on the display screen. He was given the initial concepts for the procedures and filled in the specific instructions himself. The procedure ideas he was given were.

```
TO CAR                TO CH
5 PENUP               10 MAKE "LETTER KEY
10 MAKE "D 10         20 IF :LETTER = "R RT 30
20 CH                 30 IF :LETTER = "L LT 30
30 FORWARD :D         END
40 GO 20
END
```

(The command KEY was given as a "primitive" which "tells the computer which letter you type on the keyboard.")

Using this basic idea. Karl was able to define his own system of commands to control the TURTLE's motion:

```
TO CAR                TO CH
1 WRAP                10 MAKE "LETTER KEY
5 PENUP               20 IF :LETTER = "R RIGHT 30
10 MAKE "D 10         30 IF :LETTER = "W LEFT 30
20 CH                 40 IF :LETTER = "F MAKE "D :D + 5
25 WAIT 5             50 IF :LETTER = "S MAKE "D :D - 5
30 FORWARD :D         60 IF :LETTER = "A PENUP
40 GO 20              70 IF :LETTER = "Q PENDOWN
END                   80 IF :LETTER = "E MAKE "D 0
                      END
```

The letters "R" (right turn), "F" (faster), "S" (slower), and "E" (emergency stop), all are abbreviations for their functions while the letters "W", "A" and "Q" were chosen for their position on the keyboard.

Using these seven keys Karl could turn the TURTLE in any direction, make it speed up, slow down, or stop, and could decide whether the TURTLE should draw a line as it moved. Karl discovered that repeatedly pressing "S" would slow the TURTLE down, and eventually make it move backwards. By alternating between the "A" and "Q" keys he could make the TURTLE draw dotted lines. By slowing the TURTLE down and carefully controlling its direction, he found he could use this device to create interesting free-form designs, or to write his name in script.

2.4. Affective Aspects of Karl's LOGO Experience

At the beginning of the LOGO classes, Karl tended to have a "deadpan" expression at all times. This corresponded to his appearance when seen for interviews, or on random occasions throughout the school. Even when he was successful in using the computer for the first few sessions, his expression continued to be blank and non-committal.

As he began to feel successful, Karl became more assertive and curious. He asked what error messages meant, and sought to understand how to use new commands. He made a point of finding out how to use the Plotter and Printer, so that he could make his own "hard copy" of his computer work. At about the same time, he began to express an interest in the work of other children, and to show them his work. He invited a friend to class and swapped programs with him, his behavior demonstrating that he was feeling good and enjoying himself. His face was becoming more expressive, his posture more relaxed.

Changes in Karl's atitude toward his classroom work were noted by his regular teacher. She reported that he was beginning to show that he really cared about his school work, that he had begun concentrating on his work in a way that she had not seen before, and that he seemed to have a great deal more confidence in his ability to carry out academic tasks. She attributed these changes directly to his feeling of success in the LOGO classes.

Karl's success in his LOGO classes demonstrated that with an appropriate educational environment, he was able to function at a higher level of ability than he had demonstrated in schoolwork, even with a great deal of one-to-one tutoring.

By the time of his final interview, after the end of the LOGO classes, Karl had become significantly more articulate. He listed ten uses for a tin can (as opposed to four uses for a brick in the first interview). Instead of carrying out the four color permutation task, he asked "Can I just show you how I do it?" and proceeded to describe a system for finding six permutations that started with each of four colors. With some difficulty, he correctly calculated that there were twenty four possibilities in all, and leaning over to speak directly into the interviewer's tape recorder, he said: "Twenty-four. I'm a brain!"

3. The Work of Tina: A Learning-Disabled Student

Tina is a student with severe learning disabilities, whose academic abilities are extremely low compared with her classmates. Like Karl whose work was described above, she was successful in the LOGO classes, making use of abilities that she was not using to full advantage in her regular classes, or in her daily one-to-one tutoring sessions.

Tina's pattern of computer use was unique among our sixteen experimental subjects in that she was not interested in Turtle Geometry and never learned to write her own LOGO procedures. On the other hand, Tina established an intense, personal relationship with the computer, and, using the computer as a text editor and word processor wrote a series of stories that represented a major achievement in the area of creative writing.

3.1 Tina's Attitude Toward the Computer

In the first class Tina established a special relationship with one of the four computers. She personalized it by giving it a name, Peter, and behaved toward the computer in much the same way a child might behave toward a favorite doll, pet, or much younger child. Tina was extremely possessive of the computer she called "Peter", and would not allow other students to use that particular machine during her classes.

In the early classes Tina made a number of efforts to communicate with the computer, typing questions such as "What's your name?" and responding angrily to error messages with messages of her own.

Gradually, Tina's relationship with the computer tempered somewhat. She was shown how to make the computer communicate, and was helped to write the procedure, WHO:

```
TO WHO
1 PRINT [MY NAME IS PETER]
END
```

The teacher continually stressed the fact that the computer was a "dumb machine", controlled by Tina and her friends, as well as by the teacher and other programmers. Tina eventually developed a more balanced understanding of her role in relation to the computer. Her behavior with the computer became much more matter-of-fact, and although she continued to share credit with "Peter" for her accomplishments, she no longer expressed anger at error messages. As she came to understand the mechanical predictability of the computer's responses to her, and as she began to take more pride in her own accomplishments, "Peter" came more and more to take on the status of a personal fantasy -- one which a child knows is a fantasy, but persists in "playing" sometimes because it's fun.

In her actual work with the computer, Tina had some ideosyncrasies that differentiated her from other students. Although she had a great deal of difficulty starting work at the beginning of a class period, Tina insisted on completing any task before the end of the period. Her work often ended with a rush, or with requests to stay late to finish up. Once a class ended, she never returned to a task regardless of its actual state of completion.

Tina also had a need to have all her work be "correct". When she made a typing error, or received an error message, Tina would clear the screen immediately by typing a long series of carriage returns. This habit interfered with her learning, as it prevented her from maintaining continuity in her work, and elminated the possibility of an appropriate response to an error message. Even completed stories were often removed from the screen before they could be read, much less responded to.

In addition to her compulsive aversion to errors, Tina had a strong desire for neatness. She regularly straightened up the LOGO classroom, reminding other children to put their things away, etc. She loved the printed copies of her work, and always made multiple copies to give to her friends, family and teachers.

3.2 Tina's Use of the Computer as an Editor

Tina devoted most of her time and effort in the LOGO classes to writing, editing and printing copies of a series of "stories" that she wrote. A special program, LETTER, was created for her use, allowing her to type a story directly into the computer without having to write a procedure or use PRINT commands. She used this procedure to write two "letters" and seven "stories" during the course of the LOGO classes. At least three additional stories were discarded without being finished.

Tina wrote about people who were part of her life. Her first two letters were to her aunt and her mother; Tommy, Ann, Sonny, and Donell, subjects of her stories are all Tina's relatives. Harriet, Mr. Lewis and Miss Hirsh, are a classmate and two of her teachers. Each story was written during one class period, and was never continued or changed, once it had been finished.

The letter, HELEN, is representative of Tina's first two attempts to use the computer for writing:

```
DEAR HELEN HOW ARE YOU IN YOUR NEW HOME.  I AM GOING TO GET YOU SOMETHING
  FOR YOUR NEW HOME .AS SOON AS I GET MY MONEY
LOVE TINA
```

In writing this letter Tina regularly asked for spelling and punctuation help. She was concerned that it be gramatically correct, and in proper form. A great deal of time was spent making sure each line was correct, before going on to the next.

Once she shifted from writing letters to stories, Tina became less concerned with spelling and grammar (although she continued to ask for help in these areas). Instead, she was more concerned with the details of the story: the names of the characters, the places they lived, the sequence of events, and the feelings of the characters involved. Tina had deep feelings about her subject matter. Her story, SONNY, is typical of her style and intensity of feeling:

```
SONNY IS A LITTLE BOY HE LIVES WITH HIS AUNT HELEN IN CALIFORNIA
HE HAS BEEN LIVE WITH HER FOR 9 YEARS.  HE IS GOING TO A HOME FOR
LITTLE WONDERS 4 WEEKS AFTER THAT TO
COUPLE A ADOPTED SONNY HE WAS THE HAPPIEST BOY THAT YOU EVERY
SEEN.  I GUESS IF THAT WAS ME I WOULD BE HAPPY IF SOME ONE WOULD
ADOPT.  BUT SEE I AM NOT ADOPT I HAVE MY ON MOTHER AND I AM
GLAD THAT I HAVE MY ON MOTHER.BECAUSE THE KIDS THAT HAVE FEELS
REALLY BAD. THAT'S WHY ALL THE KIDS IN THE WORLD SHOULD BE
GRATEFUL TO THEIR PARENTS. THE END.
```

After Tina had completed SONNY, her fifth piece of writing, she was asked for some samples of writing done in class. "I don't have time to write stories in class," was her reply. "I've got too much work to do." Tina's English teacher and her learning disabilities teacher confirmed the fact that she had done virtually no creative writing in school during the year. Her English teacher explained that she rarely completed any work, and pointed out that her computer stories were among her first finished pieces of work.

After completing each of her stories, Tina printed out between ten and twelve copies to be distributed to her friends, family and teachers. The widespread distribution of her stories was an indication of the pride and satisfaction she felt in her work.

We have attempt to determine the reasons for Tina's success at writing with a computer when many other approaches had resulted in failure. While our findings are somewhat speculative, we feel that we can offer the following insights:

> --Tina is fanatic about work being "complete" and free of errors. Using the computer, she could rubout her errors as they occurred, and correct them without destroying all her previous work. Once she declared a story "finished" she never proofread it or looked back to see if she could find any other errors.
>
> --The teacher in the LOGO classes offered no correction or criticism of her work, limiting his role to answering her questions about spelling, punctuation and grammar.
>
> --The printed computer output had a "professional" quality that Tina liked. She could give away as many copies as she wanted -- thus obtaining positive feedback about her writing from many different people.
>
> --Tina felt that her work was unique, special, and competent. Since only Tina was writing stories using the LETTER proram, she did not have to compare her work with that of any other students.

The combination of these and other factors produced a profound effect on Tina that was apparent to everyone who dealt with her outside of the LOGO classes. Having taken pride in her computer stories, Tina became more conscientious about her other school work, beginning to complete assignments consistently for the first time all year. Having been accepted by the other students in her LOGO class, her "social position" within her class as a whole improved. Thus, although Tina's LOGO experience was perhaps the least conventional of all the students, it seems to have been possibly the most profound.

4. Computer Programming: What the Students Learned

The process of learning to write procedures, particularly more complicated ones which involve subprocedures in an organized hierarchical form, is quite complex. In developing a systematic way of describing student behaviors and student learnings in this area, we have separated our observed behaviors into seven interrelated categories for analysis.

1. Acquiring the sense of command

2. Developing the notion of a procedure as an entity

3. Separating the process from the product of a procedure (how a procedure works versus what the end result looks like).

4. Acquiring flexibility in establishing hierarchies of procedures, including:

   --"playing TURTLE" or "playing computer"

   --becoming aware of stage change equivalence and state transparency

   --functional naming of procedures and subprocedures

   --modularity

   --creating functional procedures

5. Fitting a procedure into a hierarchy: "top-down" versus "bottom-up"

6. Developing patterned procedures using REPEAT, recursion and looping

7. Using variables in procedures

1. Acquiring the Sense of Command

Although the idea of "command" may seem quite obvious to an adult computer scientist, a beginning student may require some time to develop a purposeful sense of deliberately controlling the TURTLE by particular commands. As one might expect, most students develop a sense of command after some degree of initial contact with a computer. Sometimes a student who has developed a sense of command with regard to basic TURTLE instructions, may revert to seemingly purposeless behavior when in initial contact with a new command or idea.

Behaviors Observed:

1.1 A Student sees no connection between commands typed at the computer terminal, and the actions of the TURTLE.

Example: The student looks only at the text display screen. Following a teacher's instruction, s/he may type instructions such as FORWARD 100 or RIGHT 45, without looking at the graphics display screen to see what the TURTLE does. Such a student is likely to be more interested in messages typed by the computer such as "YOU HAVEN'T TOLD ME HOW TO FD19", than in the motion of the TURTLE.

1.2 A student types random Instructions on the Keyboard, and then looks to see what the TURTLE has done.

Example: Having as yet no basis for predicting the types of inputs that might be useful for controlling the TURTLE, a student may type commands such as FORWARD 555 or RIGHT 123, choosing numbers on the basis of previously familiar number patterns, or because they are easy to type. The student then looks at the screen to see what has happened, but is often confused by seemingly random effects or "OUT OF BOUNDS" messages.

1.3 A student who seems to have mastered the sense of command with regard to TURTLE commands such as FORWARD and BACK, RIGHT and LEFT, etc. reverts to the use of random inputs when given the opportunity to make use of a procedure such as POLY or POLYSPI (see Chapter 5, Section 1).

Example. Having not understood the connection of the POLY procedure with FORWARD and RIGHT commands with which s/he is already familiar, the student chooses inputs to POLY such as 123 456 or 555 555, based on their familiarity as number patterns or because they are easy to type. Working in this way, the student may see no consistency between the inputs to POLY and the results obtained, and may temporarily lose a sense of controlling the behavior of the TURTLE.

2. Developing the Notion of a Procedure as an Entity

Merely acquiring the idea of a procedure in a form sufficiently rich and flexible to allow efficient planning, writing and debugging of programs as complex as those produced by our subjects is a significant accomplishment. Learning the proper LOGO syntax is probably the least problematic step. A more interesting issue is coming to think of a sequence of commands as "a thing" having well-defined internal constraints and external properties.

Internally, the strict sequential nature of a procedure is a new experience for most students. Externally one may sometimes consider a procedure to be a command, to invoke a particular image on a graphics screen, for example. Side effects, such as a net move or turn that results from running a procedure, will be discussed in the sections that follow.

Behaviors Observed:

2.1. The student uses a procedure name to run a procedure which s/he did not define.

Example: A POLY or CIRCLE procedure; is given to a student by a teacher or another student. The student may realize that this procedure is derived from direct LOGO commands, but s/he uses the procedure as though it were a direct command, without particular concern for how the procedure itself works; product logic dominates the student's process.

2.2. The student is observed to repeat a sequence of computer commands, achieving the repetition of a particular effect, without giving evidence of thinking of the sequence of commands as an entity.

Example:

2.2.1 A student may draw a square, using direct commands: FORWARD 100, RIGHT 90, etc., until the square is complete. S/he repeats the sequence of commands to draw one or more additional squares, with increasing ease and swiftness, but without defining a procedure.

2.3. <u>The student creates a sequence of steps, and expresses the desire to repeat or preserve those steps for further use</u>.

Examples:

2.3.1 A student may draw a square using direct commands. Then s/he may say, "I want the computer to do that again." Or "Will the computer remember how to do that?" This indicates that the student is consciously aware that a particular sequence of commands can produce a replicable effect.

2.3.2. The student gives a name to a particular sequence of steps, and using proper LOGO syntax (including TO, line numbers and END), creates a LOGO procedure by copying a particular list of direct commands in order to repeat a previous effect.

Having caused the computer to draw a square using direct commands, FORWARD 100, RIGHT 90, FORWARD 100, RIGHT 90, FORWARD 100, RIGHT 90, FORWARD 100, the student writes a LOGO procedure:

```
TO SQUARE
1 FORWARD 100
2 RIGHT 90
3 FORWARD 100
4 RIGHT 90
5 FORWARD 100
6 RIGHT 90
7 FORWARD 100
END
```

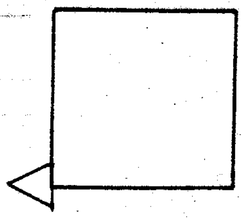


Figure 2.1

2.3.3. The student chooses a name for a procedure, and then uses correct LOGO syntax to write a procedure using a random list of instructions. A procedure is written, and tried out <u>afterwards</u>. One can see this as exercising the abstraction "writing a procedure."

```
TO JOHNNY
1 FORWARD 45
2 LEFT 63
3 FORWARD 22
4 RIGHT 77
5 BACK 99
END
```

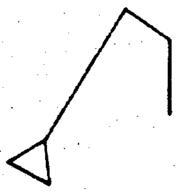


Figure 2.2

3. Separating the Process from the Product of a Procedure

Distinguishing between the internal constraints and the external properties of a procedure involves distinguishing the "process logic" of how a procedure works from the "product logic" of the result of running the procedure. We have observed that most students initially treat a procedure as the picture product it produces. Often the side effects of drawing a particular picture -- a net move or rotation of the TURTLE -- forces a student to relax the boundary between the internal and external aspects of a procedure, in order to appreciate how a particular procedure affects what happens next. This boundary can become particularly problematic when the student needs to create one; for example, in creating process elements (FORWARD 100 RIGHT 90, as "one side" of a square), or in creating subprocedures. This will also be discussssed in section 4.1, below.

Behaviors Observed:

3.1 A Student is Surprised by the Effect of Running a Procedure More than Once

A common occurance is that a student creates a SQUARE procedure, such as that shown in Figure 2.1. Running SQUARE twice produces the surprising effect shown in figure 3.1a. Continuing to run SQUARE results in the shape shown in figure 3.1b, after the fourth SQUARE.

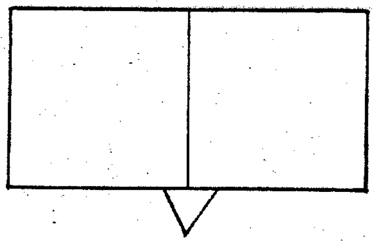


Figure 3.1a

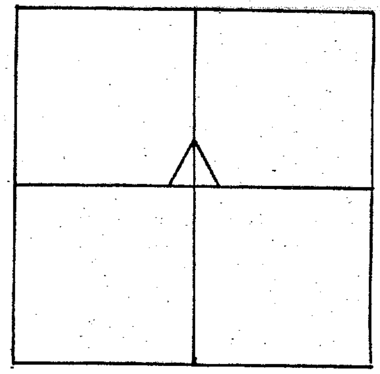


Figure 3.1b

3.2 Failing to account for the process of a procedure

EXAMPLES:

3.1.1 Darlene used a series of commands, FORWARD 40, RIGHT 40, FORWARD 40, RIGHT 40, ... to make a nine-sided polygon. She carefully counted and found that there were nine FORWARD 40 steps and eight RIGHT 40 steps. She tried to repeat the drawing as follows:

```
REPEAT [FORWARD 40] 9
REPEAT [RIGHT 40] 8
```

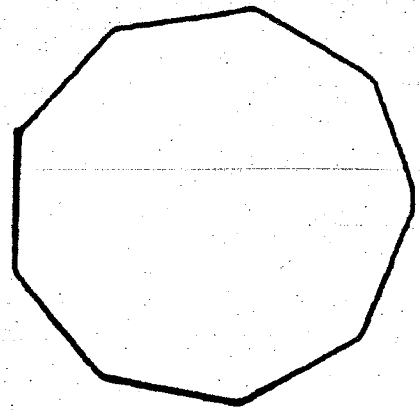


Figure 3.2

Her procedure drew a straight line which resulted in an OUT OF BOUNDS message.

Darlene was not fully aware of the process of her design. From the product point of view, she had probably isolated the FORWARD 40 steps as the only ones which had visible results and treated the RIGHT 40s as "additional ingredients in the recipe." In reality it was the repeat of the process element, FORWARD 40 RIGHT 40, which was needed to produce her desired result.

3.2.2 The following sequence of steps may be suggested to draw a "balloon on a string" or a "lollypop".

```
FD 100

RCIRCLE 20
```

Those steps, however, produce the following figure.

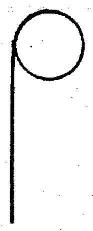


Figure 3.3

Betsy who had a pattern of "product oriented" behaviors, saw this bug as a misplacement of the circle and corrected it by "moving the circle" up and to the left, i.e. moving the TURTLE up and to the left before starting the circle.

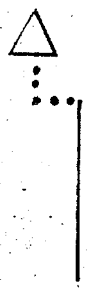


Figure 3.4a

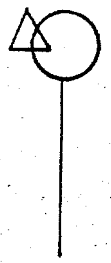


Figure 3.4b

On the other hand, a student who is aware of the process that the TURTLE follows to draw the circle (even if students never look at the steps of the circle procedure, they see it acted out in boring detail each time RCIRCLE is run) notices that the TURTLE repeatedly goes forward and turns, thus drawing the first part of the circle in the direction initially pointed by the TURTLE. A simpler debug is possible with this enriched view -- turn the TURTLE 90 degrees before starting the RCIRCLE process.

```
FORWARD 100
LEFT 90
RCIRCLE 20
```

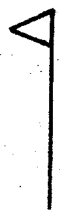


Figure 3.5a

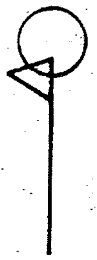


Figure 3.5b

3.2 "Playing Turtle" or "Playing Computer" to Uncover the Process Logic of a Procedure or Subprocedure.

One way to help students resolve this kind of difficulty is to suggest that students "play TURTLE", that is, "put themselves in the TURTLE's place," and physically carry out a set of instructions. This is often used to help students draw a circle with the TURTLE. First the student is asked to walk in a circle, then to separate his motion into distinct forward and turn steps. In doing so, most students realize that a "TURTLE circle" can be drawn by repeating a very small step and a very small turn over and over again.

4. Acquiring Flexibility in Establishing Hierarchies of Procedures

The most flexible and efficient use of procedures in hierarchies, raises a number of issues which many students learned to cope with as part of their LOGO experience.

BEHAVIORS OBSERVED

4.1 Paying attention to state change equivalence and state transparency

This section extends the thread of distinguishing the process and the product of a procedure, begun in section 3, into the development of the concept of TURTLE state, state change, and state change equivalence. From the viewpoint of a TURTLE procedure all that is important about a particular subprocedure is its net change of state (net motion). Any subprocedure with an equivalent state change can be substituted and will not affect in any way the rest of the procedure. Students use this knowledge in a number of ways. For example, a student may realize that a quarter circle is equivalent to going forward the radius, turning 90 degrees and going forward the radius again, and use that in his planning.

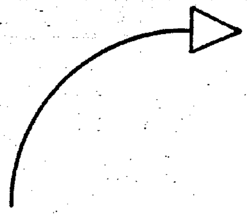


Figure 4.1a

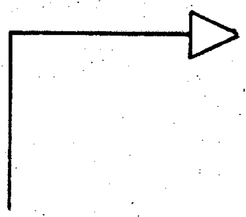


Figure 4.1b

Another way of insuring modularity is to write procedures with no net state change. Such "state transparent" procedures can be introduced or left out at any point of a superprocedure with no effect on the rest of the process.

4.1.1 An Example of Failure to Relate the State of the TURTLE (Process) of a Procedure to the Product

Albert had great difficulty "unpacking" the process of what he had done, once a series of steps were combined into a procedure. Although he was quite competent at moving the TURTLE around the screen, he seemed to lose track of what to do when he had to move the TURTLE from the ending of one to the start of another. This is shown most clearly in the project in which he had the computer draw his initials.

Albert had no difficulty creating the subprocedures, A and J, which drew each initial.

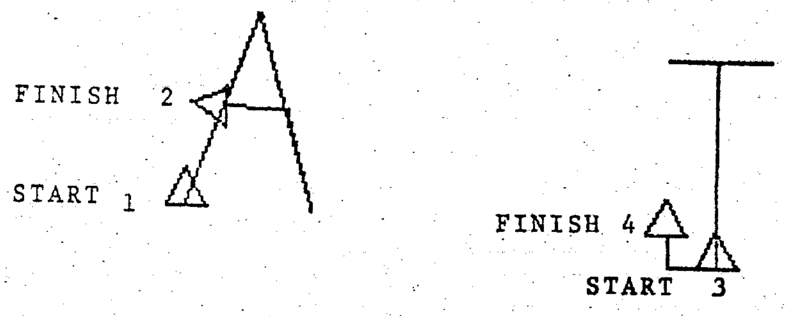


Figure 4.2

The TURTLE finished drawing the A at point 2, and began drawing the J at point 3. Despite a very clear idea of what he wanted to result to look like, he had no idea of how to plan the interface steps. It took a trial and error process involving seven attempts before the A and J were aligned correctly.

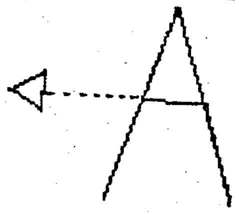
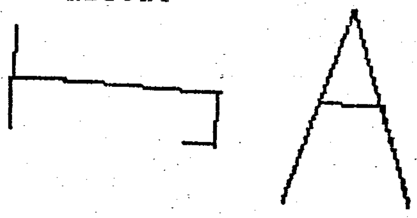
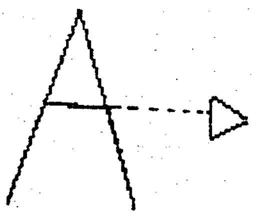
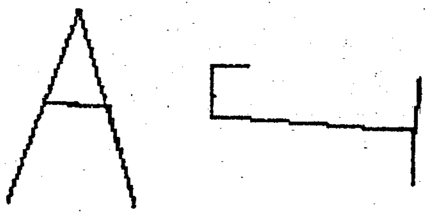
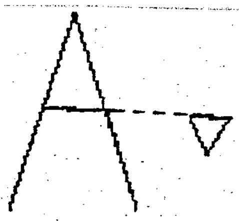
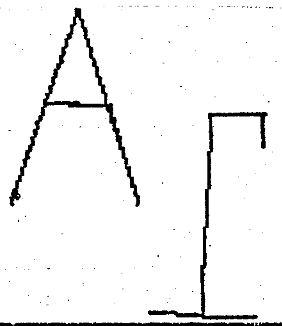
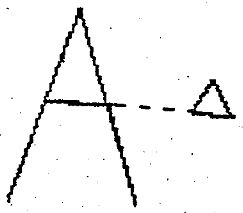
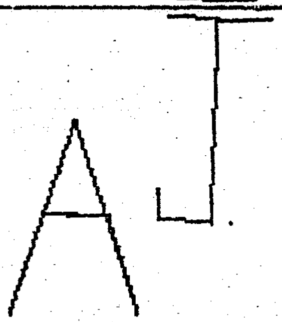
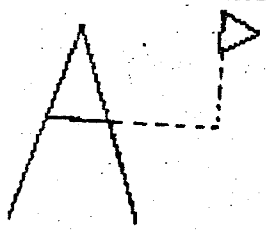
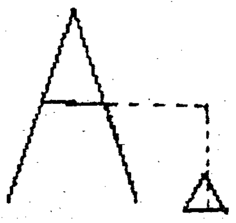
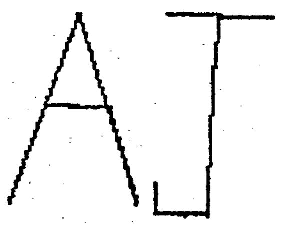
TRIAL	TURTLE MOVED TO	RESULT
1		
2		
3		
4		
5	SAME AS TRIAL #3	SAME AS TRIAL #3
6		THE J WAS "OUT OF BOUNDS"
7		

Figure 4.3

4.1.2 Jimmy's Racetrack

Jimmy, in contrast to Albert, always seemed to know eactly where the turtle had to move to construct the next part of his design. Jimmy had completed the inside of an "oval" racetrack by using the series of commands:

```
FORWARD 150
RARC 40
RARC 40
FORWARD 150
RARC 40
RARC 40
```

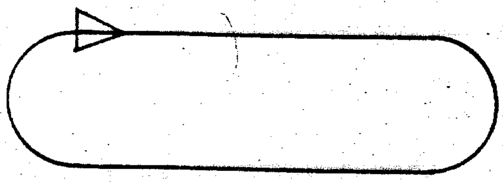


Figure 4.4

Without "visible" calculations or planning, he moved the turtle over as shown, and used these commands to draw the outside of the racetrack.

```
PENUP
LEFT 90
FORWARD 40
RIGHT 90
PENDOWN
FORWARD 150
RARC 80
RARC 80
FORWARD 150
RARC 80
RARC 80
```

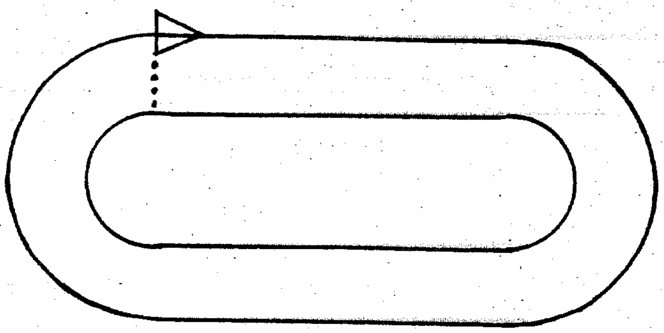


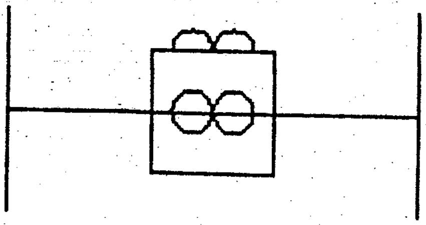
Figure 4.5

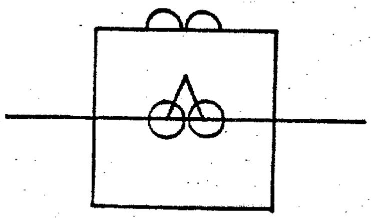
When asked by an observer how he knew where to place the turtle to draw the outside, and how he chose the input for RARC to draw the outside of the track, Jimmy replied that he knew that RARC was the same as "turning a corner," and that since he knew that he wanted the track to be 40 units wide, he made the radius of RARC 40 units larger than the radius he had used for the inner track. This explanation, together with the way he moved the turtle, indicated that he was taking into account the "net effect" of the RARC procedure -- its starting and ending states -- as well as its product -- what it looked like -- in carrying out his plan.

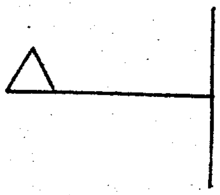
4.1.3 Writing State Transparent Procedures

The clearest way to eliminate confusion between the product drawn by a subprocedure and the steps necessary to put it in the right position, is to make the subprocedure itself state-transparent. A state transparent procedure begins and ends with the turtle in the same state, so that the problem of locating the procedure on the screen can be solved separately from the problem of drawing it. Although the use of state transparency was not consistent among our students, some of them did come to understand and apply this idea.

In Gary's STARSHIP, for example, the three major subprocedures, C, which drew the "cabin" of the starship and WINGL and WINGR, which drew its wings, were state transparent. Each of them ended in the same state it began. For more details of these procedures see section 4.6.1 below.


STARSHIP
Figure 4.6


C
Figure 4.7


WINGR
Figure 4.8

4.2 Functional Naming

In defining a superprocedure clearly it is helpful to give subprocedures names which describe their <u>function</u> in the superprocedure rather than an intrinsic property of the subprocedure itself. For example a certain CIRCLE becomes an EYE in a FACE.

4.2.1 The student may begin by choosing names for procedures according to the <u>shape</u> that they draw, such as SQUARE, TRIANGLE, STAR or DIAMOND.

EXAMPLE:

HOUSE could be drawn using the subprocedures SQUARE and TRIANGLE:

```
TO HOUSE
10 SQUARE
20 RIGHT 90
30 FORWARD 100
40 RIGHT 30
50 TRIANGLE
END
```

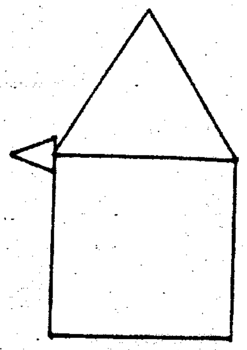


Figure 4.9

4.2.2 Later the student may choose names for procedures according to their purpose or function in a hierarchy.
Examples:

```
TO HOUSE          TO FRAME          TO ROOF
10 FRAME          10 BOX            10 RIGHT 30
20 ROOF           20 RIGHT 90       20 TRIANGLE
END               30 FORWARD 100    END
                  END
```

The FRAME and ROOF subprocedures need not use existing procedures BOX and TRIANGLE. They might include all the steps necessary to draw those shapes.

4.2.3 Darlene made partial use of this approach in her CAT project (see Section 5.2.1).

Her procedure names EAR, EAR1 and TAIL were descriptive of their function in her design, while her procedure names RIBIT, WEE and TURN, were randomly chosen.

4.2.4 Donald's HEAD superprocedure (see Section 5.2.2) made consistent use of functional procedure names.

```
TO HEAD
1 BOX
2 EYES
3 NOSE
4 MOUTH
5 BEARD
6 HAIR
70 EARS
80 HAT
85 FLOWER
END
```

4.3 Modularity

It is usually efficient to use certain procedures as modules, often having several different functions in the same or different superprocedure. This may involve writing the procedure in a more general form (say, with size inputs) than might appear necessary on first thought.

4.3.1 Non-Modularity: Deborah's "Square" and "Diamond"

Deborah, who had worked with squares for a long time, had made the statement, "Now I know all about squares." When shown a picture of a tilted square, Deborah ignored her SQUARE procedure, and attempted to define a new procedure. After a great deal of experimentation, she realized that her new shape required 90 degree turns at each corner, and she eventually defined the procedure.

```
TO DIAMOND
1 RIGHT 40
2 FORWARD 30
3 RIGHT 90
4 FORWARD 30
5 RIGHT 90
6 FORWARD 30
7 RIGHT 90
8 FORWARD 30
END
```

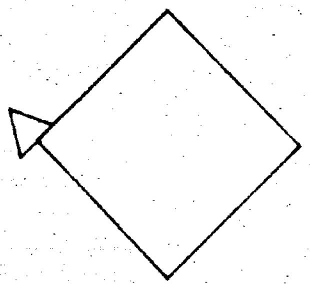


Figure 4.10

4.3.2. Kevin used his TRIANGLE procedure as a subprocedure in two different ways. He repeated it four times to make a FLOWER. Later, He combined it with a BOX procedure to make a HOUSE.

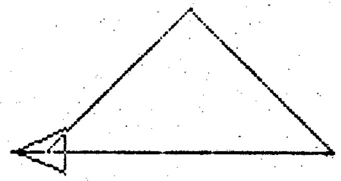


Figure 4.11

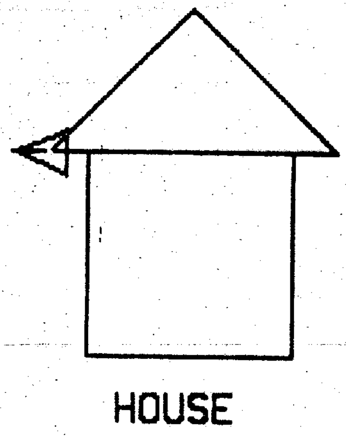


Figure 4.12

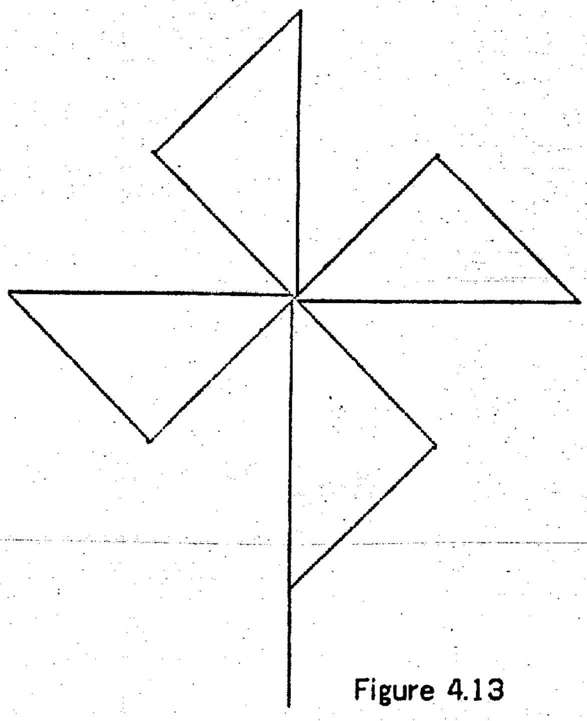
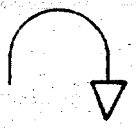
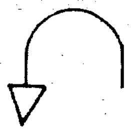
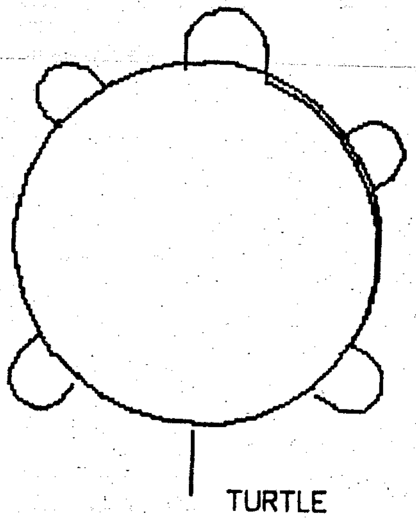


Figure 4.13

4.3.3 Kevin's major project was to draw a "turtle". He used the same pair of subprocedures, FOOT and BKFOOT, four times, to draw the "turtle's" feet.


FOOT
Figure 4.14


BKFOOT
Figure 4.15


TURTLE
Figure 4.16

4.4 Modular Debugging

A student realizes that a subprocedure is an "entity" which can be debugged without changing other elements of a project in which it is embedded. The student will then have to take into account the way in which changing one subprocedure effects the rest of a project.

4.4.1 A student may have written the following buggy procedures:

```
TO BALLOON          TO STRING
10 STICK            10 FORWARD 100
20 RCIRCLE 20       END
END
```

BALLOON produces the picture, shown in figure 4.17a instead of that shown in figure 4.17b as intended.

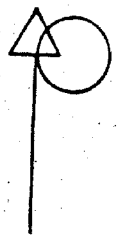


Figure 4.17a

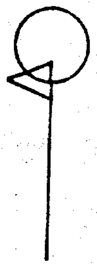


Figure 4.17b

This can be debugged by changing the subprocedure STRING, without changing BALLOON as follows:

```
TO STRING
10 FORWARD 100
20 LEFT 90
END
```

This will produce the desired effect (figure 4.17b) when the command BALLOON is given.

4.4.2 Suppose, however, that BALLOON is itself a subprocedure in another design:

```
TO PERSON          TO ARMS
10 BALLOON         10 RIGHT 90
20 BACK 40         20 FORWARD 30
30 ARMS            30 BACK 60
END                40 FORWARD 30
                   50 LEFT 90
                   END
```

PERSON is supposed to draw:

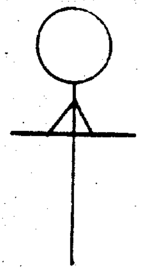


Figure 4.18a

When the original version of
BALLOON is used, PERSON produces a buggy drawing:

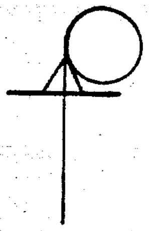


Figure 4.18b

When STRING is debugged, so that BALLOON is correct, PERSON has a different bug:

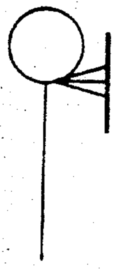


Figure 4.18c

This can be debugged by editing BALLOON:

```
TO BALLOON
10 STRING
20 RCIRCLE 20
30 RIGHT 90
END
```

Both BALLOON and PERSON now produce the desired effect (Figure 4.18a):

4.5 Systems of Procedures

As a student masters a number of these approaches, s/he may be able to combine several of them into a "system" of procedures and subprocedures to carry out a complex task. In particular, the process of a procedure may have parts and even modules which are not evident in the final product. In terms of student behaviors, for example, beginning students rarely separate the interface steps needed to go from one part of a design to another, as an independent subprocedure, though that is a practice which can aid readability and debugging. Some students do begin to use these functional procedures after some programming experience.

EXAMPLES:

4.5.1 The second version of Gary's starship design used functional procedure names, state transparent subprocedures, STA, WINGR and WINGL, and interface subprocedures MO, MOV and the modular procedure LI :LE.

The procedure STA draws the central part of the STARSHIP, while WINGR and WINGL make use of LI, MO, and MOV, to draw the two wings. LI :LE, draws a symmetrical line of any length. MO and MOV shift the TURTLE 100 units to the right and left respectively. (See Figures 4.6-4.8, p. 4.13.)

```
TO STARSHIP      TO WINGR        TO WINGL
10 STA           10 MO           10 MOV
20 WINGR         20 RIGHT 90     20 LEFT 90
30 WINGL         30 LI 50        30 LI 50
END              35 LEFT 90      35 RIGHT 90
                 40 MOV          40 MO
                 END             50 HIDETURTLE
                                 END
```

```
TO LI :LE
5 PENDOWN
10 RIGHT 90
20 FORWARD :LE
30 LEFT 180
40 FORWARD 2 * :LE
50 RIGHT 180
60 FORWARD :LE
70 LEFT 90
END
```

```
TO MO
10 RIGHT 90
20 FORWARD 100
30 LEFT 90
END
```

```
TO MOV
10 LEFT 90
20 FORWARD 100
30 RIGHT 90
END
```

4.5.2 A much more elaborate example of the combined use of these techniques is the system Harriet created for her TICTACTOE game. She needed a large number of subprocedures and variables in order to draw the board and play a game of TICTACTOE, keeping track of the moves, so that the computer could declare a winner. (For a detailed description of the system Harriet created, see the profile of Harriet's work, in part III, CHAPTER 8).

5. Fitting a Procedure Into a Hierarchy: "Top-down" vs. "Bottom-Up"

There are two opposing modes of placing procedures in a hierarchy. The most natural is to take procedures at hand and fit them as primitive elements into a superstructure. This is a "bottom-up" approach. Alternatively one may design the superstructure as an outline first and then implement the necessary new subprocedures. This latter, "top-down" approach, usually allows better fitting of the hierarchy of the process to the logic of the product and hence makes procedures easier to plan, read, and debug. Our examples detail some of the stages in coming to learn both of these complementary approaches and some interesting "pathological" use of the notions as well.

BEHAVIORS OBSERVED:

5.1 The "Bottom-Up" Approach

In this section we present a sequence of successively more complex uses of the "bottom-up" approach.

Examples:

5.1.1 The student uses one of her own procedures as an entity in a design involving simple repetition of that procedure with possible intervening steps.

A student instructs the computer to carry out the following series of steps: SQUARE, SQUARE, SQUARE, SQUARE, or SQUARE, RIGHT 40, SQUARE, RIGHT 40, SQUARE, RIGHT 40.

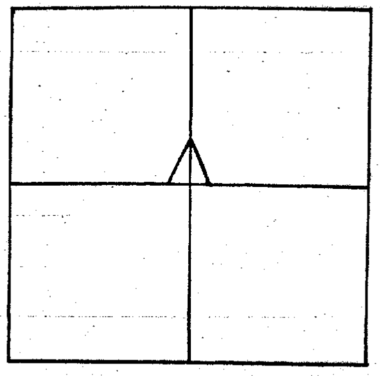


Figure 5.1

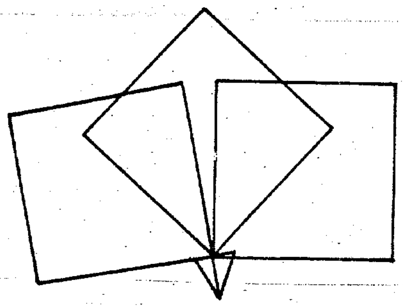


Figure 5.2

5.1.2. The student combines several self-written procedures in one design.

For example: SQUARE, JOHNNY, SQUARE:

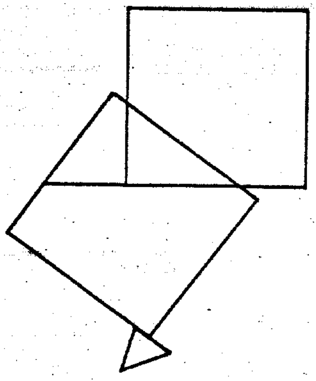


Figure 5.3

5.1.3. The student writes procedures that incorporate previously written procedures:

```
TO WINDOW          or          TO JUNKY
1 SQUARE                       1 SQUARE
2 SQUARE                       2 JOHNNY
3 SQUARE                       3 SQUARE
4 SQUARE                       END
END
```

5.1.4. The student incorporates existing procedures as sub-elements of a planned design. This involves a goal-directed moving and turning of procedure-shapes with TURTLE commands.

Example: A "house" built from a triangle and a square procedure.

```
TO HOUSE
10 SQUARE
20 RIGHT 90
30 FORWARD 100
40 RIGHT 30
50 TRIANGLE
END
```

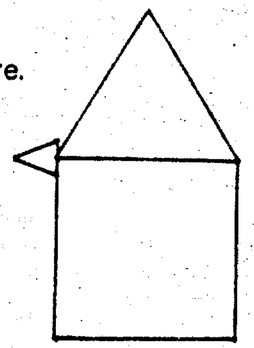


Figure 5.4

A "face" built from square procedures of different sizes.

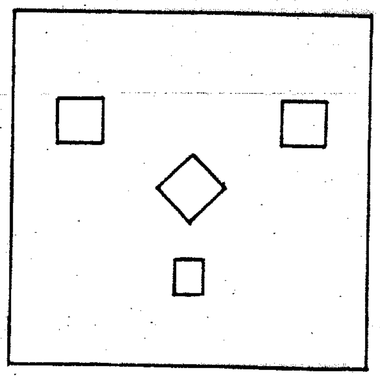


Figure 5.5

Note that using procedures as subprocedures in this fashion is distinct from "free-hand" construction of such designs using only basic TURTLE commands.

5.2. The "Top-Down" Approach

EXAMPLES:

5.2.1. A student divides a task into manageable chunks by breaking a long problem into subparts, which become subprocedures. For example, Darlene's procedure, CAT:

```
TO CAT
1 RIBIT        ;draws two circles for the cat's body and head
2 WEE          ;moves the turtle to draw the cat's left ear
3 EAR          ;draws the left ear of the cat
4 EAR1         ;moves the turtle and draws the right ear
5 TURN         ;moves the turtle to draw the cat's tail
6 TAIL         ;draws the cat's tail
END
```

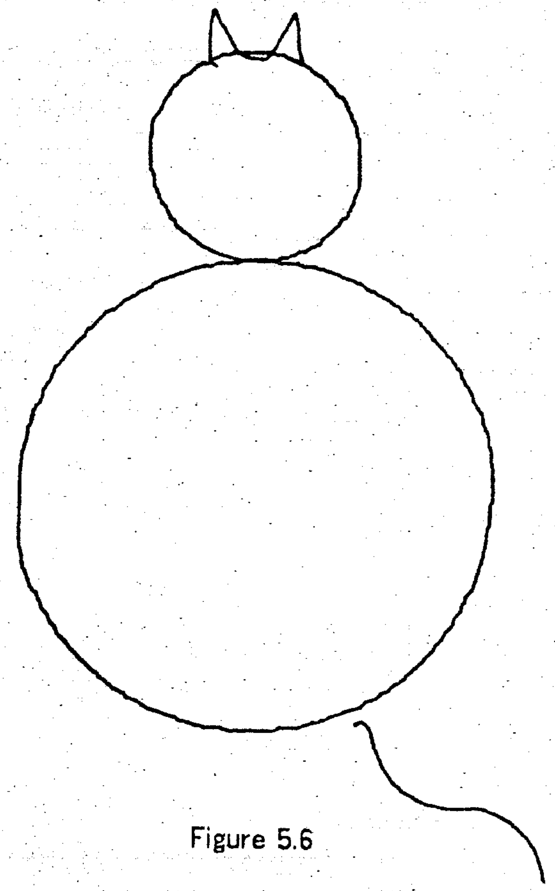


Figure 5.6

Darlene's CAT is atypical of this stage in separating off some interface steps (WEE and TURN) as subprocedures. More typical is what she does in EAR1 which moves the TURTLE and draws the second ear. Note the use of both descriptive, functional subprocedure names, EAR, EAR1, and TAIL, and non-descriptive, non-functional names RIBIT and WEE.

5.2.2. A student breaks a long problem into subparts, but uses a "cumulative hierarchy" structure in which each procedure contains its predecessor procedure as the first step, and then adds additional steps. The following set of procedures draws a person.

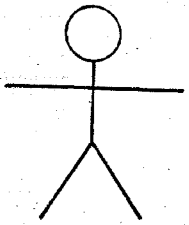


Figure 5.7

```
TO PERSON
10 LEG2
20 RIGHT 135
30 FORWARD 50
40 RIGHT 90
50 FORWARD 50
60 BACK 100
70 HIDETURTLE
END
```

```
TO LEG2
10 LEG1
20 LEFT 90
30 FORWARD 50
40 BACK 50
END
```

```
TO LEG1
10 BODY
20 RIGHT 135
30 FORWARD 50
40 BACK 50
END
```

```
TO BODY
10 HEAD
20 RIGHT 90
30 BACK 50
END
```

```
TO HEAD
10 LEFT 90
20 RCIRCLE 15
END
```

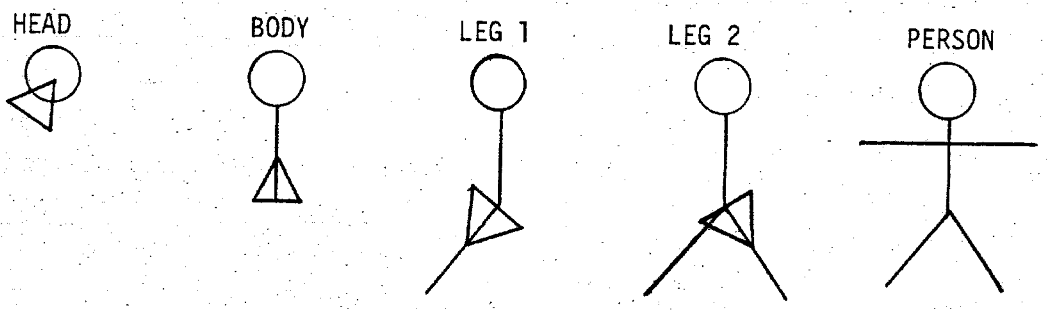


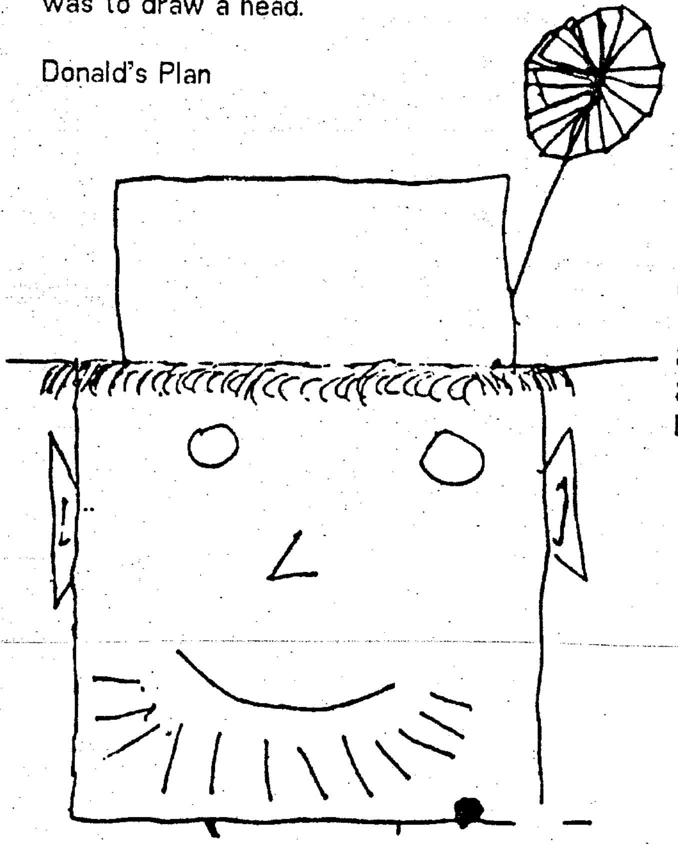
Figure 5.8

Note that the process hierarchy does not match the hierarchy of the product, defeating readability and modular debugging. LEG1 draws a lot more than a leg and legs can't be run by themselves to try them out.

Jimmy used this kind of approach in his long robot project Details are given in Jimmy's profile, Chapter 9, Part III of this report.

5.2.3 A student engages in full top-down advance planning of a project, perhaps even writing procedures into the hierarchy before defining them. Donald's project was to draw a head.

Donald's Plan



```
TO HEAD
1 BOX
2 EYES
3 NOSE
4 MOUTH
5 BEARD
6 HAIR
7 EARS
80 HAT
85 FLOWER
END
```

Figure 5.9

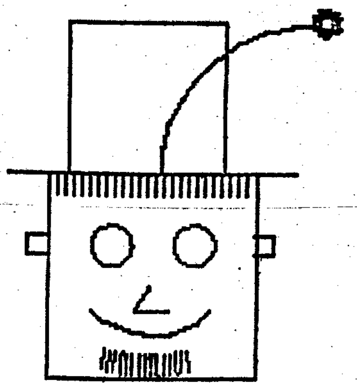


Figure 5.10

First Donald planned his project by means of a drawing. First he wrote the superprocedure HEAD containing the first six subprocedures. Donald then created each subprocedure in order. After completing the head as far as the hair, Donald added the subprocedures EARS, HAT and FLOWER as he completed them. All of the elements were present in his initial drawing. In writing his subprocedures Donald did not separate out the interface steps, but included them at the beginning of each new subprocedure.

6. Developing Patterned Procedures Using REPEAT, Recursion and Iteration

There are three different approaches which can be used by beginning LOGO students to cause the same command or series of commands to be repeated more than once by the computer. Using a LOGO procedure called REPEAT, provided by the teacher; using a procedure name recursively within the procedure itself; and using a process of looping, in which the computer is directed to go back to a previous line of the same procedure.

We refer to a procedure in which a set of steps is repeated as a "patterned procedure", because the student must be aware of a repeating pattern in the process s/he is using, and because such procedures produce visual patterns when they are used in Turtle Geometry. In describing student use of patterned procedures, we will first consider examples of use of the REPEAT command, and of other simple ways that students can cause a repeated sequence of commands before we consider recursion and looping. Initial use of patterned procedures leads naturally to the use of "stop rules" which require knowledge of variables. Thus there will be a certain amount of overlap between this section, and Section 7, "Using Variables in Procedures."

BEHAVIORS OBSERVED:

6.1 Simple Forms of Repetition

Many students begin to repeat a particular set of instructions over and over, before they have even learned to write a procedure, or even to distinguish clearly between the effects of inputs to FORWARD and RIGHT commands. At a later stage, they may make a procedure, and create an unexpected pattern by repeating the procedure. At a still later stage a student who has planned to create a visual pattern with the TURTLE can carry it out by defining a procedure which is then repeated to complete the pattern.

EXAMPLES:

6.1.1 Repetition of a Sequence of Commands

At an early stage of LOGO work, a student may type FORWARD 65, RIGHT 65, FORWARD 65, RIGHT 65, ... Although this may be begun as a random activity, once the student notices the visual pattern s/he may keep typing the same commands until s/he feels the pattern is "complete". This type of behavior often occurs when students are beginning their exploration of LOGO, and have not yet distinguished between the effects of inputs to FORWARD and RIGHT commands (see Chapter 5, Section 1).

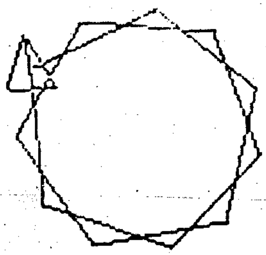


Figure 6.1

6.1.2 Repetition of Procedures

Having defined a procedure, the student may repeat that procedure several times. The emerging pattern may at first be a surprise to the student. Later, s/he may deliberately create a procedure in order to make a pattern by repeating it. For example:

SQUARE
SQUARE
SQUARE
SQUARE

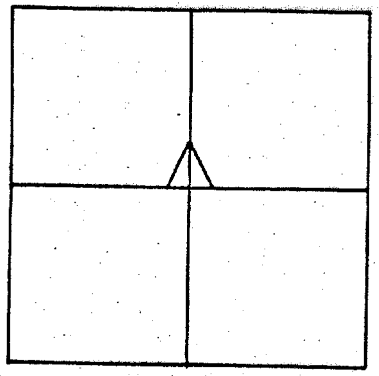


Figure 6.2

JOHNNY
JOHHNY
JOHNNY
JOHNNY
ETC.

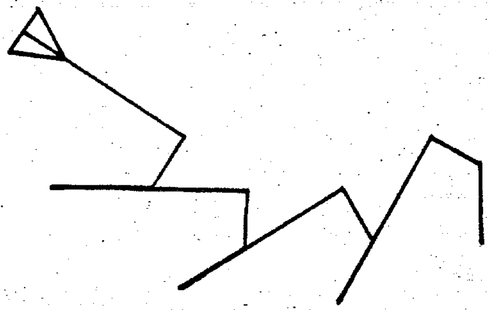


Figure 6.3

6.1.3 Defining a Procedure for the Purpose of Repeating a Fixed Sequence of Steps

A student may define a procedure to aid in drawing a patterned design. Kathy, for example, had decided to make the TURTLE draw a "circle" by repeating FORWARD 20 RIGHT 20 a large number of times. To save typing, she wrote the procedure, ROUND, repeating it four times to complete the "circle":

```
TO ROUND
1 FORWARD 20
2 RIGHT 20
3 FORWARD 20
4 RIGHT 20
5 FORWARD 20
6 RIGHT 20
7 FORWARD 20
8 RIGHT 20
9 FORWARD 20
10 RIGHT 20
END
```

```
TO SHELL
1 ROUND
2 ROUND
3 ROUND
4 ROUND
END
```

6.2 The Use of REPEAT

REPEAT is a LOGO procedure given to our students as a "primitive", indistinguishable to them from a LOGO command. REPEAT requires two inputs: a list of instructions (contained within square brackets, [ ]) and a number, telling how many times the instructions are to be repeated. This allows a student to experiment with the number of times a pattern is to be repeated, as well as to create automatically repeating patterns without having to learn the more complex process of creating a recursive or looping procedure.

EXAMPLES

6.2.1 Using REPEAT to Draw Circles

Darlene used REPEAT to create a variety of circles of different curvature. Using REPEAT she was able to easily explore the effects of varying both size and angle as well as to discover the connection between the angle turned, and the number of turns needed to complete a circle (the "Total Turtle Trip Theorem," see Chapter 5, Section 4). These learnings came at an early stage of Darlene's LOGO activities, before she had the sophistication to incorporate variables in a repeating procedure. Darlene made several circles, two of which she used later as part of her CAT design.

```
REPEAT [FD 3 RT 2] 180
```

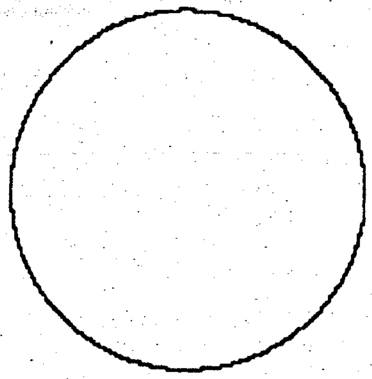


Figure 6.4

```
REPEAT [FD 2 RT 3] 120
```

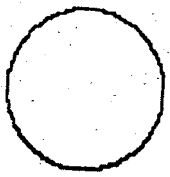


Figure 6.5

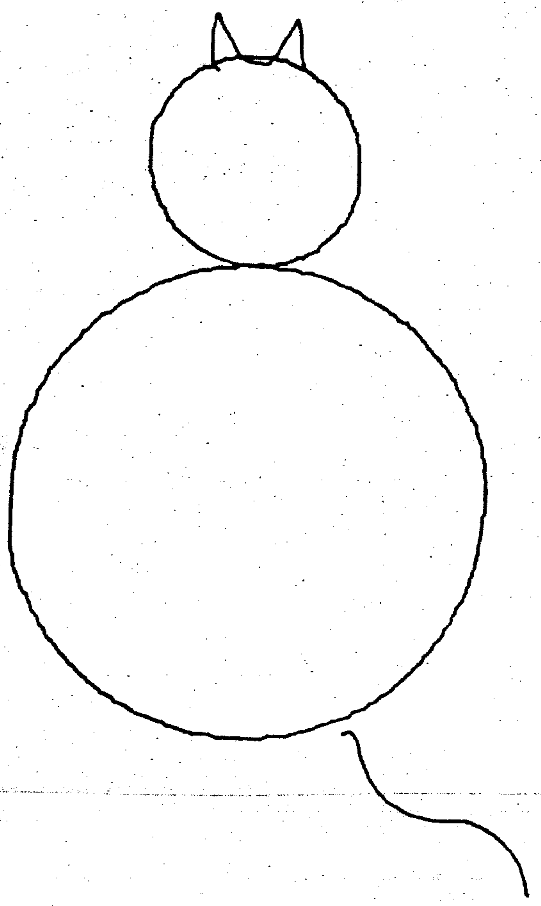


Figure 6.6: CAT

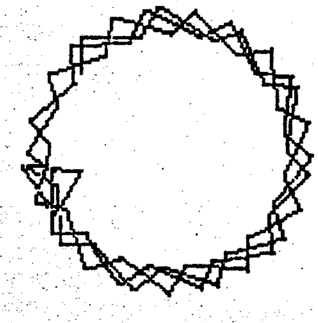
6.2.3 Using REPEAT to Create Random Patterns

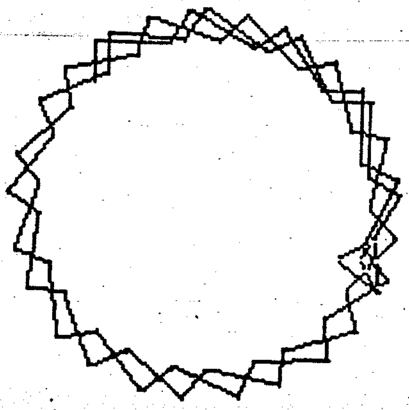
Ray made a number of designs using REPEAT. He would write a procedure with a few random steps, and then experiment with the patterns created by repeating it different numbers of times:

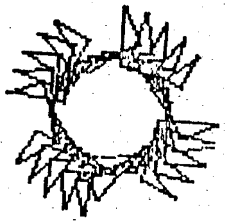
```
TO SAM
1 FORWARD 17
2 RIGHT 90
3 FORWARD 29
4 LEFT 56
END
```

```
TO TIM
1 FORWARD 19
2 RIGHT 90
3 FORWARD 36
4 LEFT 61
END
```

```
TO JOE
1 SAM
2 LEFT 150
3 TIM
END
```


REPEAT [SAM] 30
Figure 6.7

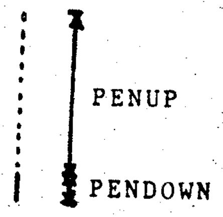

REPEAT [TIM] 30
Figure 6.8

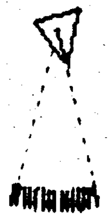

REPEAT [JOE] 30
Figure 6.9

6.2.3 Using REPEAT to Draw Patterned Elements of a Larger Design

Donald used REPEAT to draw both the BEARD and HAIR of his head. His procedure STRING drew one hair of a BEARD, and rotated the TURTLE a little. BEARD used REPEAT [STRING] 15, to draw a complete beard. Similarly, the hair was drawn by using REPEAT [HAIRY] 25, where HAIRY drew a single hair and moved the TURTLE over.

STRING REPEAT [STRING] 15





HEAD

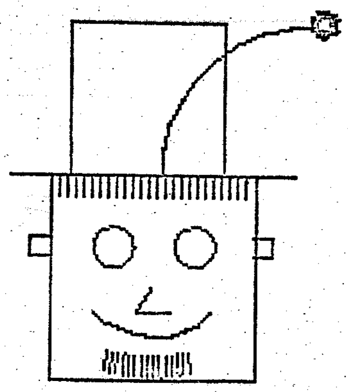


Figure 6.10

Figure 6.11

6.3 The Use of Recursion to Create Patterned Procedures

All the students in our LOGO classes were introduced to the use of recursion at some point during their LOGO experience. Typically, their first use of recursion is "to make something keep happening." In trying to make a circle, for example, they might be shown or may invent the procedure:

```
TO CIRCLE
1 FORWARD 1
2 RIGHT 1
3 CIRCLE
END
```

Many students incorporate this idea into other projects which involve the creation of designs by repeating a fixed series of steps over and over. Students who incorporate this approach into their own work usually go on to include "stop rules" to stop a procedure when the pattern is complete, and variables to allow a greater variety of effects with the same procedure. Some of the students used recursive procedures which increment or decrement a variable. A few students used recursive procedures to manipulate variables -- words or lists, although this was typically after they had had a good deal of LOGO experience. Only the simplest use of recursion is discussed here. Use of recursion with variables and stop rules is discussed in section 7.3 below.

EXAMPLES:

6.3.1 Using Recursion in "cliche" form to Repeat a Series of Fixed steps.

Monica was a student who used simple recursion a great deal. Her procedure, FAN, designed to produce a pattern by rotating a triangle, is a typical example:

```
TO FAN
1 TRI
2 LEFT 10
3 FAN
END
```

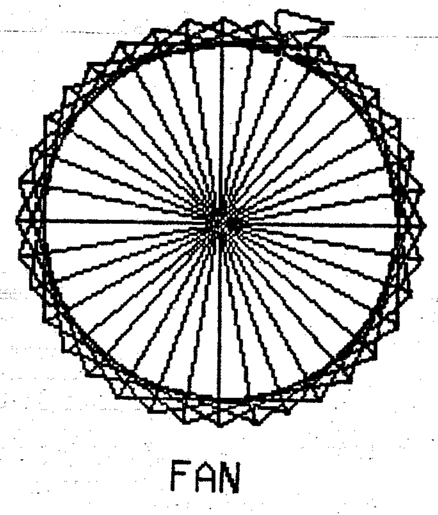


Figure 6.12

6.4 The Use of Looping to Create Patterned Procedures

Looping in LOGO is carried out by using the command GO, followed by a line number, which transfers control to the given line, in the same procedure. For example, the following procedure draws a "circle"

```
TO CIRCLE
10 FORWARD 1
20 RIGHT 1
30 GO 10
END
```

For many applications, the issues involved for beginning students are similar to those encountered in projects involving recursion. Each approach has its own distinct advantages and disadvantages, which will *be mentioned briefly. Except for the use of looping in animation, specific examples of student work will not be presented.*

In procedures involving list and word processing, recursion is usually preferred because it makes it easier for a student to understand the process if each new step is carried out by a new procedure with altered inputs.

Situations in which looping is preferred are those in which a large number of repeats (more than 100) are required. Looping can continue indefinitely, while recursion is limited by the amount of storage space available for new procedures. Looping is therefore to be preferred in the case of a POLY procedure (see chapter 5 section 1) which can require as many as 360 repeats before completing a design, or in an animation, in which a process is repeated indefinitely.

EXAMPLES

6.4.1 The Use of Looping in Animation Procedures

Karl developed a procedure in which the TURTLE was made to move continuously across the screen, and could be controlled by typing commands at the keyboard:

```
TO CAR               TO CH
10 MAKE "D 10        10 MAKE "LETTER KEY
20 CH                20 IF :LETTER = "R RIGHT 30
30 FORWARD :D        30 IF :LETTER = "L LEFT 30
40 GO 20             40 IF :LETTER = "F MAKE "D :D + 5
END                  etc.
```

Karl's work in developing this procedure is discussed more fully in section 7 of this chapter which deals with student use of variables, and in Chapter 3, dealing with exceptional students.

Whereas Karl's procedure CAR "animated" the TURTLE, Gary's procedure MOVESHIP, "animated" his "starship." (see Figure 4.6)

```
TO MOVESHIP
10 MAKE "D 10
20 DISPLAY :SHIP
30 FLY
40 CS
50 FORWARD :D
60 GO 20
END
```

Gary's FLY procedure was similar to Karl's CH, allowing Gary to change the speed or direction of the starship's motion.

7. Using Variables in LOGO Procedures

The use of variables is a major step in the development of mathematical thinking. A variable in LOGO involves the assignment of a name to an object, in such a way that the name of the object can be used in procedures to stand for the object itself. The object might be a number, a word, a list of numbers, words or other lists, or in special cases a "snapshot" of a LOGO drawing. An object may be able to take on a number of different values, which must be specified before a procedure can be carried out.

The use of variables thus represents a higher level of abstraction than the use of direct commands. Instead of being able to see an immediate direct connection between the command and the action it produces, the student must anticipate a variety of different actions, depending on the different values of the variable. A variably sized square or triangle procedure can draw a large number of similar figures, either separately, or one after another, rather than one fixed shape. A procedure can be made to print out a statement, in which the message or messages to be printed have not yet been determined. The use of variables, along with the use of patterned procedures, can unlock the power of a computer for a student.

The LOGO language deals with variables in a way that requires the user to distinguish the name of a variable (indicated by an opening symbol called "quotes") from the value or thing of the variable (represented by a colon: called "dots"). Variables are created in LOGO in two ways. By using the LOGO comand MAKE to assign a name to a particular thing:

```
MAKE "NUM 7

MAKE "MESSAGE "HELLO

MAKE "SENTENCE [HELLO, HOW ARE YOU?]
```

And by including a variable name in the title of a procedure:

```
TO MOVE :DISTANCE
10 FORWARD :DISTANCE
END
```

```
TO SAY :MESSAGE
10 PRINT :MESSAGE
END
```

In using variables, the distinction must be maintained between the use of the name of the variable, and the use of the object or value of the variable. For example, the command

PRINT :MESSAGE results in

HELLO

While the command

PRINT "MESSAGE results in

MESSAGE

If "DISTANCE has a value of 100, then the command

FORWARD :DISTANCE

will cause the TURTLE to move forward 100 TURTLE steps. The command

FORWARD "DISTANCE, however, will result in an error message,

FORWARD DOESN'T LIKE "DISTANCE AS AN INPUT.

Student use of variables in our LOGO classes fell mainly into three major categories having some degree of overlap:

1. The use of variables to change the size and/or shape of a drawing;

2. The use of variables to store information which can later be used in a procedure, or printed as part of a message;

3. The use of variables to control or stop the action of a procedure.

BEHAVIORS OBSERVED:

7.1 Using Variables to Change the Size or Shape of a Turtle Drawing.

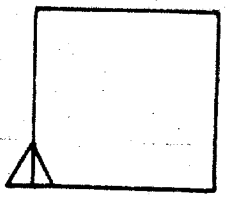
The use of variables to change the size or shape of a geometric figure, provides a concrete introduction to the use of variables.

EXAMPLES:

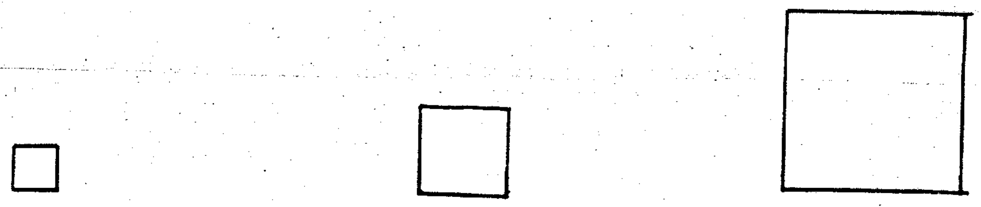
7.1.1 Varying a Fixed Shape

Once a student has written a procedure to draw a square, for example, and has isolated the idea of a square, realizing that many squares of different sizes can be drawn, just by changing the value of all the forward steps in a square procedure, the student is usually ready to understand the use of LOGO variables to accomplish this task with just one procedure. The procedure SQUARE can be "copied" as the procedure NEWSQUARE, by adding :SIZE to the title, and by substituting :SIZE for each forward step:

```
TO SQUARE                    TO NEWSQUARE :SIZE
10 FORWARD 100               10 FORWARD :SIZE
20 RIGHT 90                  20 RIGHT 90
30 FORWARD 100               30 FORWARD :SIZE
40 RIGHT 90                  40 RIGHT 90
50 FORWARD 100               50 FORWARD :SIZE
60 RIGHT 90                  60 RIGHT 90
70 FORWARD 100               70 FORWARD :SIZE
80 RIGHT 90                  80 RIGHT 90
END                          END
```



SQUARE
Figure 7.1

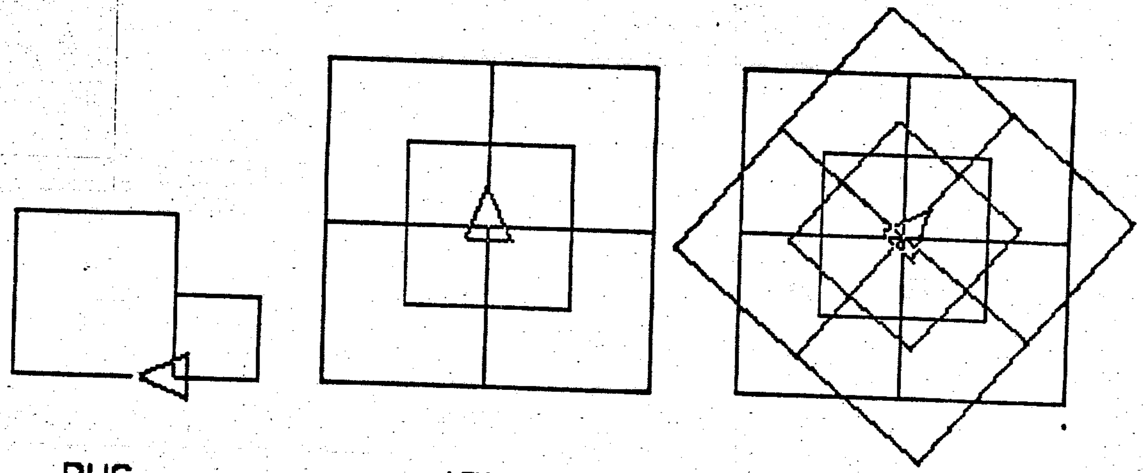


NEWSQUARE 20 NEWSQUARE 50 NEWSQUARE 100

Figure 7.2

Monica and Kathy used a variable square procedure, SQ, to make other designs:

```
TO BUS           TO 4BUS     TO STAR
1 SQ 40          1 BUS       1 4BUS
2 LEFT 90        2 BUS       2 RIGHT 40
3 SQ 80          3 BUS       3 4BUS
END              4 BUS       END
                 END
```

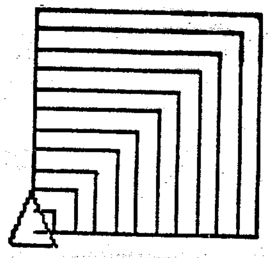


BUS 4BUS STAR

Figure 7.3

and

```
TO WOW
1 SQ 10
2 SQ 20
3 SQ 40
4 SQ 40
5 SQ 50
6 SQ 60
7 SQ 70
8 SQ 80
9 SQ 90
10 SQ 100
11 SQ 110
END
```



WOW

Figure 7.4

Kevin created a similar figure using a POLY procedure with a fixed 90 degree angle, and a variable size. In addition, he made use of a recursive procedure which incremented the value of the size and included a stop rule:

```
TO FU :SIZE
10 POLY :SIZE 90
20 IF :SIZE = 100 STOP
30 FU :SIZE + 5
END
```

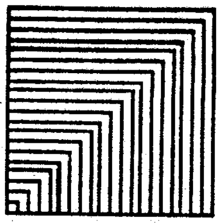


Figure 7.5

Using a variably sized equilateral triangle, THRI1 :SIZE, Dennis created a set of nested triangles by halving the size of the triangle with each repeat, and including a stop rule:

```
TO Q :SIZE
10 IF :SIZE < 10 STOP
20 THRI1 :SIZE
30 FORWARD :SIZE/2
40 RIGHT 60
50 Q :SIZE/2
END
```

```
TO THRI1 :SIZE
1 FORWARD :SIZE
2 RIGHT 120
3 FORWARD :SIZE
4 RIGHT 120
5 FORWARD :SIZE
6 RIGHT 120
END
```

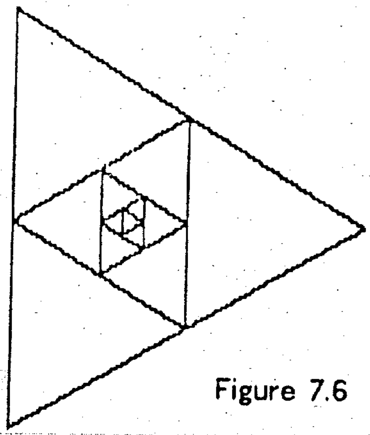


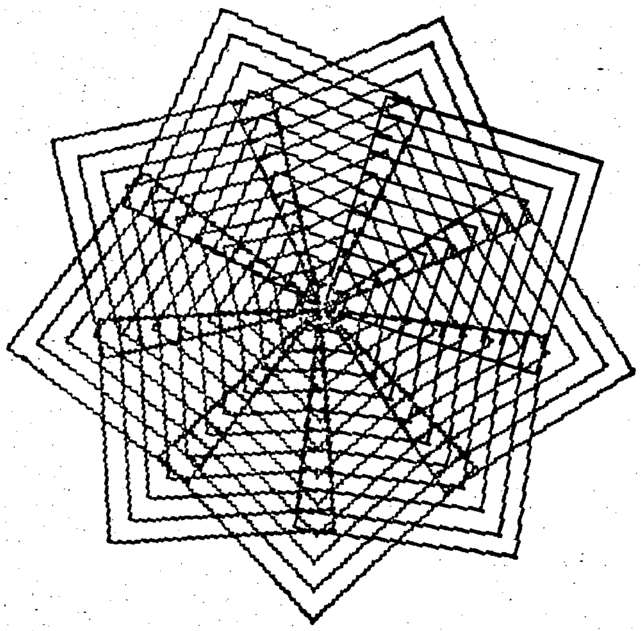
Figure 7.6

7.1.2 Varying the Angle in a Procedure

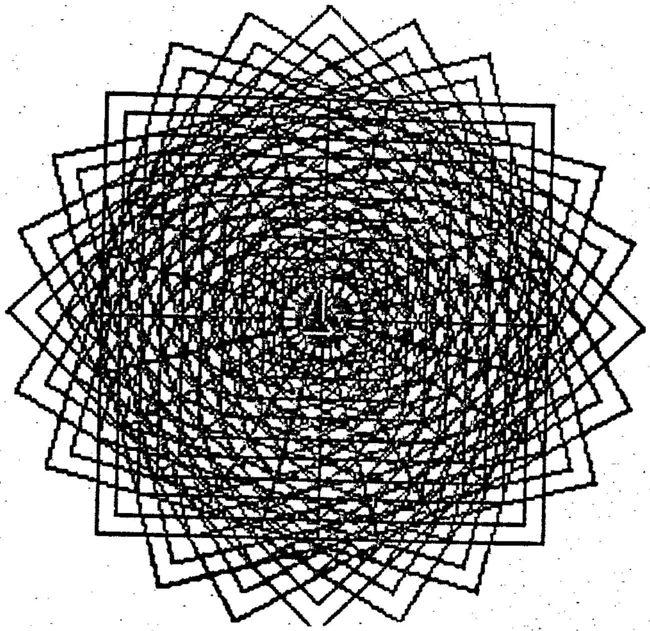
A common theme in Turtle Geometry is to create a design by using a shape, rotation, shape, rotation...motif. Students usually used fixed angles in their first attempts -- Monica's procedure, FAN, (Figure 6.12) for example. An easy next step would be to make the angle a variable, as Monica did in her procedure WISHWOW. In similar fashion, Albert varied the angle of rotation of a series of stars in his procedure AS;

```
TO WISHWOW :ANGLE
10 WOW
20 RIGHT :ANGLE
30 IF HEADING = 0 STOP
40 WISHWOW :ANGLE
END
```

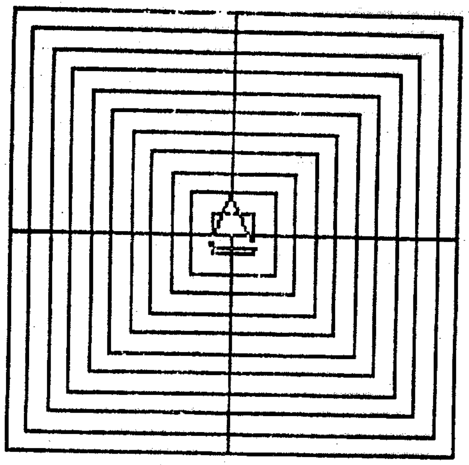
WISHWOW 160



WISHWOW 165



WISHWOW 45



WISHWOW 90

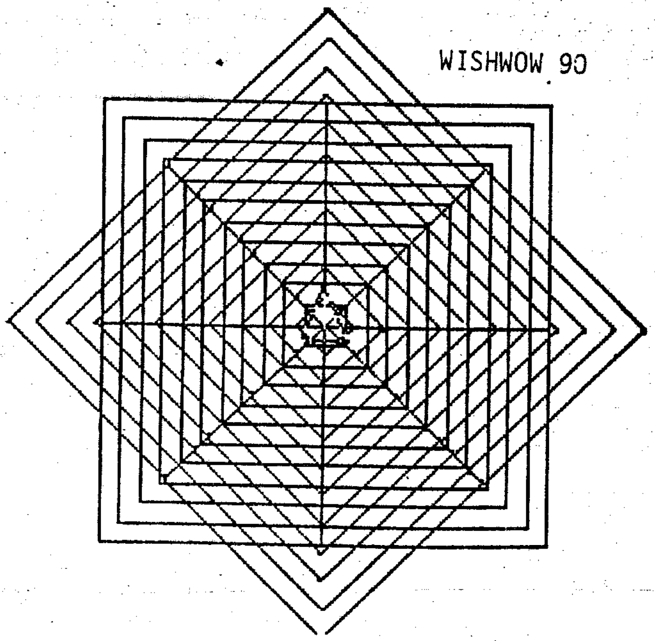
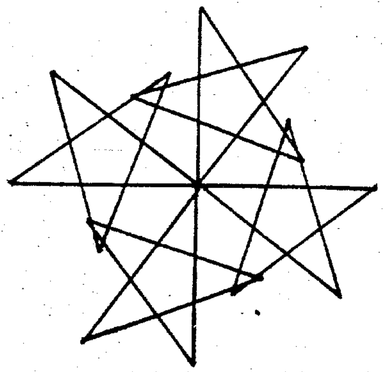


Figure 7.7

```
TO AS :ANGLE
10 STAR
20 RIGHT :ANGLE
30 AS :ANGLE
END
```

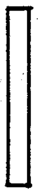

AS 90
Figure 7.8

7.1.3 Procedures with More Than One Variable

One easy extension of this type of variable use is to include more than one variable. For example, a rectangle.

```
TO RECT :SIZE1 :SIZE2
10 FORWARD :SIZE1
20 RIGHT 90
30 FORWARD :SIZE2
40 RIGHT 90
50 FORWARD :SIZE1
60 RIGHT 90
70 FORWARD :SIZE2
80 RIGHT 90
END
```

RECT 20 50


RECT 100 10
Figure 7.9

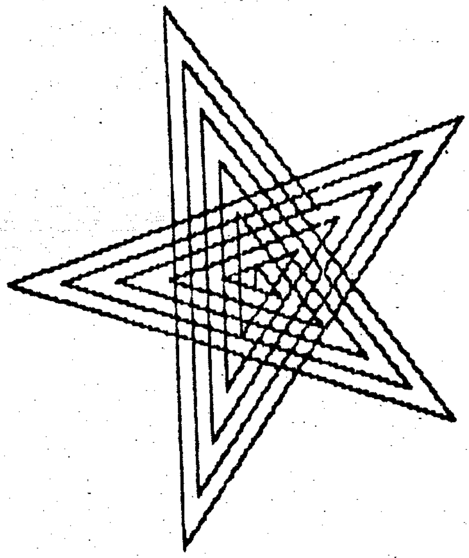
The most widely used procedure involving two variables is the POLY procedure, described extensively in Chapter 5, Section 1.

```
TO POLY :SIZE :ANGLE
10 FORWARD :SIZE
20 RIGHT :ANGLE
30 POLY :SIZE :ANGLE
END
```

(See Chapter 5, Section 1)

Fascinating extensions of POLY can be explored by incrementing either the size or angle variable. Kevin's TUNNEL procedure (Figure 7.15) and his FU procedure (Figure 7.6) are examples of the effect of incrementing the size of a particular POLY shape. More generally, incrementing the size of a POLY procedure's forward step, produces a spiral effect:

```
TO SPI :SIZE :ANGLE
10 FORWARD :SIZE
20 RIGHT :ANGLE
30 SPI :SIZE + 5 :ANGLE
END
```


SPI 5 144

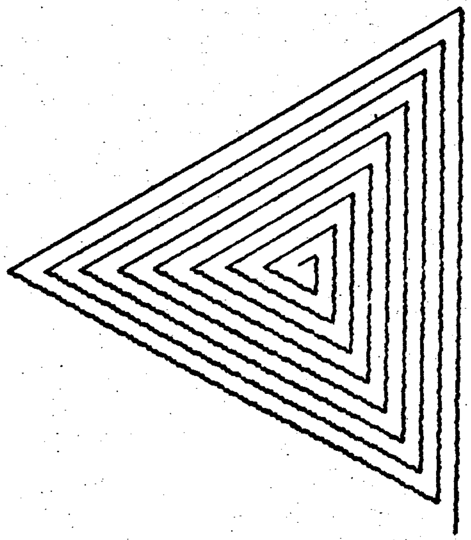

SPI 5 120

Figure 7.10

Incrementing the angle, however, produces a more unusual effect.

```
TO INSPI :SIZE :ANGLE
10 FORWARD :SIZE
20 RIGHT :ANGLE
30 INSPI :SIZE :ANGLE + 10
END
```

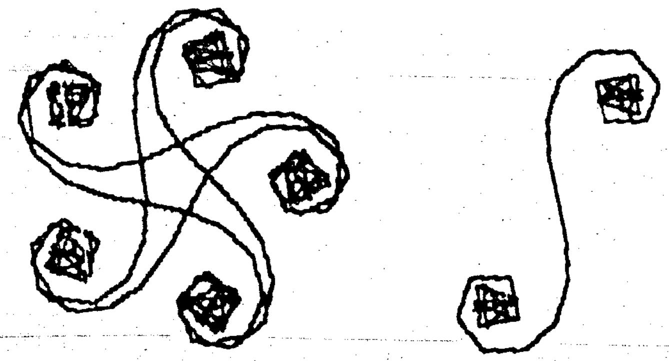


Figure 7.11

7.2 Using Variables to Store Information

Information is stored in the computer's memory by assigning a name to a number, word or list that can be used, updated, or changed at a later time.

7.2.1 Naming Points

The LOGO command HERE, outputs a list of three numbers specifying the x, y and heading coordinates of the TURTLE. A point can be named by moving the TURTLE to a particular point, and using

MAKE: MAKE "POINT HERE.

Later, the turtle can be moved back to the same point by the command

SETTURTLE :POINT,

or the coordinates of the point can be printed out with the command

PRINT :POINT.

This process was used extensively by Harriet in her TICTACTOE game, in which the user could specify that an X or an 0 should be drawn in one of nine boxes. The procedure then moved the TURTLE to a particular point in each box before drawing the X or 0:

```
TO X :POINT
10 IF :POINT = 1 PENUP SETTURTLE :P1 PENDOWN EX
20 IF :POINT = 2 PENUP SETTURTLE :P2 PENDOWN EX
etc.
```

"P1, "P2, ... were points named at an earlier stage of the process. EX is a procedure which draws an X.

7.2.2 Storing a "Message" to be Used Later

The LOGO command REQUEST waits for a user at the keyboard to type a message and carriage return, and then outputs that message as a list. This allows a procedure to store information and present it later.

```
TO HELLO
10 PRINT [HI, HOW ARE YOU TODAY?]
20 PRINT [PLEASE TYPE YOUR FIRST NAME]
30 MAKE "PERSON REQUEST
40 PRINT SENTENCE [IT'S NICE TO SEE YOU TODAY] :PERSON
END
```

In use, the procedure would look like this:

```
HELLO
HI, HOW ARE YOU TODAY?
PLEASE TYPE YOUR FIRST NAME.
<JOHN
IT'S NICE TO SEE YOU TODAY, JOHN
```

A message can also be included as a variable in a procedure title:

```
TO HELLO :PERSON
10 PRINT SENTENCE [IT'S NICE TO SEE YOU TODAY] :PERSON
END
```

In this case, the user at the keyboard would have to input the name in one of two ways:

```
HELLO  "JOHN
IT'S NICE TO SEE YOU TODAY JOHN
?
or
MAKE "PERSON "JOHN
HELLO :PERSON
IT'S NICE TO SEE YOU TODAY JOHN
```

A common use of variables to store information is in a quiz program. Here, the variable must also be compared with another object to see if the answer is correct.

```
TO QUIZ
10 PRINT [WHAT IS YOUR FAVORITE BASEBALL TEAM?]
20 MAKE "ANSWER REQUEST
30 IF :ANSWER = [BOSTON] PRINT [ME TOO] STOP
40 PRINT [I DON'T AGREE! TRY AGAIN.]
50 QUIZ
END
```

7.3 Using Variables to Control or Stop a Recursive Procedure.

A student who creates a repeating design by using recursion often wants to stop the computer after it has completed drawing the design once. At this stage of work a student begins to confront important issues which lead to a richer understanding of the logic of computer programming. Among these issues are the exact wording of a stop rule, and its location in a procedure. The procedure STAR, for example, is intended to draw a twelve-pointed star:

```
TO STAR
10 TRI
20 RIGHT 30
30 STAR
END
```

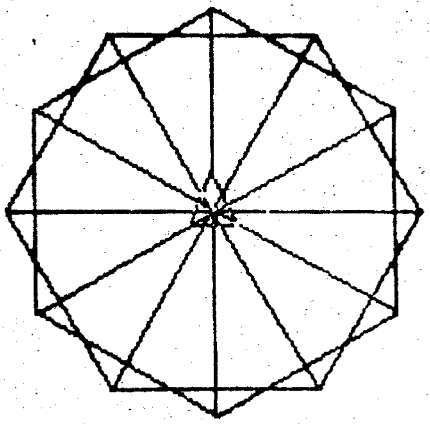


Figure 7.12

When it continues to draw redundant triangles after completing the design, a student can add a stop rule: IF HEADING = 0 STOP", (assuming the TURTLE starts in the "home" position). Beginning students usually assume that the stop rule can simply be added as the last step of the procedure. The reasoning is that since "the last thing the computer has to do is stop, the stop rule should come last". This is a natural consequence of the step-by-step sequential programming they have done up to this point. In the edited version of STAR, however, the stop rule has no effect:

```
TO STAR
10 TRI
20 RIGHT 30
30 STAR
40 IF HEADING = 0 STOP
END
```

The computer continues to re-execute STAR at line 30, and never carries out line 40. Students can learn to debug this kind of error by "playing computer," and "acting out" the execution of the procedure. Once they have realized that "line 40 is never executed," and that the stop rule must be placed before line 30, in the procedure, the problem is still not resolved. Placing the stop rule at line 5:

```
TO STAR
5 IF HEADING = 0 STOP
10 TRI
20 RIGHT 30
30 STAR
END
```

terminates execution of the procedure immediately, if the TURTLE starts at the "home" position. Placing the stop rule at line 15, stops the procedure after drawing only one triangle.

It may require a lengthy process of trial and error, including "playing computer" several times before the student realizes that the stop rule must be placed between lines 20 and 30 in order to have the desired effect:

```
TO STAR
10 TRI
20 RIGHT 30
25 IF HEADING = 0 STOP
30 STAR
END
```

Another bug might occur if the TURTLE's _initial_ *heading is not 0. In a recursive procedure, this can be resolved by initializing the stopping condition before executing the recursive procedure:*

```
TO STAR1                        TO STAR
10 MAKE "START HEADING          10 TRI
20 STAR                         20 RIGHT 30
END                             25 IF HEADING = :START STOP
                                30 STAR
                                END
```

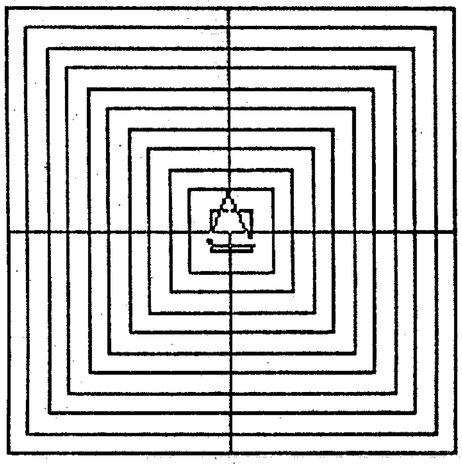
7.3.1 A Stop Rule in a Repeated Design

Monica's procedure, WISHWOW, incorporated a variable angle as well as a stop rule:

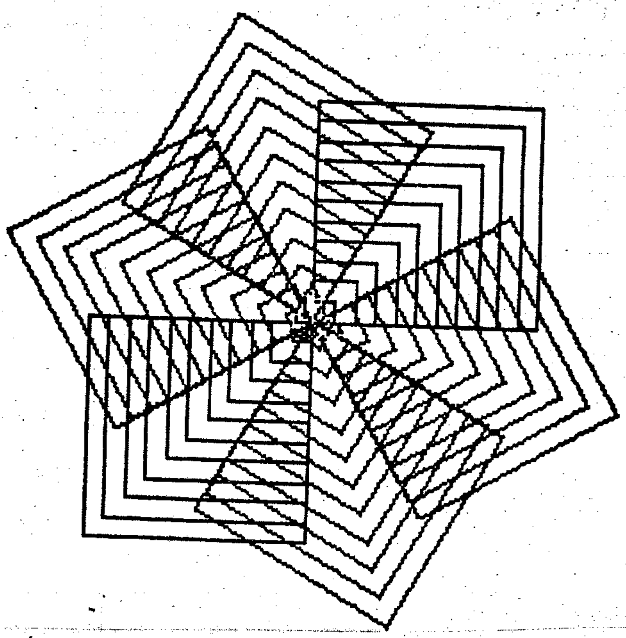
```
TO WISHWOW :ANGLE
10 WOW
20 RIGHT :ANGLE
30 IF HEADING = 0 STOP
40 WISHWOW :ANGLE
END
```

WISHWOW 90 stops after four repeats of the commands WOW, RIGHT 90;

WISHWOW 60 stops after six repeats of WOW, RIGHT 60; etc.



WISHWOW 90
Figure 7.13



WISHWOW 60
Figure 7.14

7.3.2 Incrementing Variables in a Recursive Procedure

Recursive procedures provide a simple mechanism for incrementing and decrementing variables. Students are often introduced to this technique by being shown a sample procedure, COUNTDOWN:

```
TO COUNTDOWN :NUMBER
10 PRINT :NUMBER
20 IF :NUMBER = 0 STOP
30 COUNTDOWN :NUMBER - 1
END
```

Kevin made use of this technique in his procedure, TUNNEL, which was built by using a POLY procedure in which the angle was kept constant (45 degrees) and the size continually increased:

```
TO TUNNEL :SIZE
10 POLY :SIZE 45
20 IF :SIZE = 105 STOP
30 TUNNEL :SIZE + 5
END
```

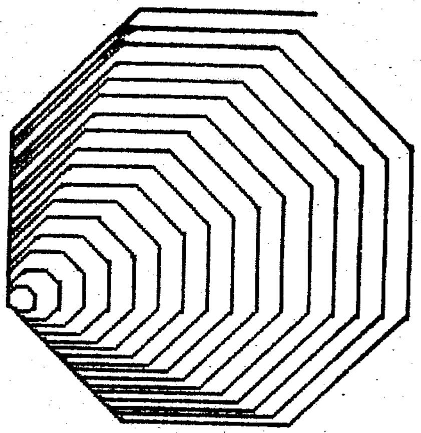


Figure 7.15

7.3.3 Using Recursive Procedures to Manipulate Variables, Words and Lists

The use of recursive procedures to manipulate words and lists was not attempted by many of our students. Although students could have been introduced to these activities in a simple way, our major focus on Turtle Geometry in these classes led us to defer word and list manipulation until students needed it for particular projects -- usually near the end of the series of classes.

In order to manipulate words and lists a student needs to understand the use of the LOGO commands FIRST, BUTFIRST, LAST and BUTLAST. The command FIRST outputs the first character of a word, or the first word of a list. BUTFIRST outputs everything but the first character of a word, or the first word of a list:

```
PRINT FIRST "HELLO
H
PRINT BUTFIRST "HELLO
ELLO
PRINT FIRST [HOW ARE YOU?]
HOW
PRINT BUTFIRST [HOW ARE YOU?]
ARE YOU?
```

LAST and BUTLAST have similar effects on the last element of a word or a list.

Gary made use of these commands in a Morse Code project. Gary's procedure, PRI2 :SENT, took an English sentence as input, and printed the Morse Code for that sentence with single slashes between letters, and spaces between words. The morse code translator was built up by first creating a lengthy procedure, CODE, which output the correct sequence of dots and dashes for any letter or number:

```
TO CODE :LETTER
10 IF :LETTER = "A OUTPUT ".-
20 IF :LETTER = "B OUTPUT "-...
```

And so forth, with one line of the procedure for each letter of the alphabet.

The procedure PRI :WORD prints the correct sequence of letters for an entire word:

```
TO PRI :WORD
10 IF :WORD = "      STOP
20 TYPE CODE FIRST :WORD
30 TYPE "/
40 PRI BUTFIRST :WORD
END
```

```
PRI "HELLO
 ..../././.-../.-../---/
```

The procedure PRI2 :SENT, prints the correct sequence of letters for an entire sentence:

```
TO PRI2 :SENT
10 IF :SENT = [  ] STOP
20 PRI FIRST :SENT
30 TYPE "
40 PRI2 BUTFIRST :SENT
END

PRI2 [HELLO HOW ARE YOU]
 ..../././-../.-../---/ ..../---/.--/ .-/.-./. / -.--/---/..-/
```

In creating these procedures Gary had to understand the difference between words and lists, and how this effected the wording of the stop rules in PRI (which manipulated words) and in PRI2 (which manipulated lists of words). He also had to carry out a process of trial and error to determine the location of the stop rule in each procedure. When the series of LOGO classes ended, Gary was engaged in the process of reversing the code -- that is, writing a set of procedures which would take a string of Morse Code symbols as input, and print out an English sentence.

7.4 Looping Procedures With Stop Rules

To add a stop rule to a looping procedure, one has to consider the location and wording of the stop rule, just as with recursive procedures. To add a stop rule to a procedure to draw twelve-pointed star, we simply write:

```
TO STAR
10 TRI
20 RIGHT 30
25 IF HEADING = 0 STOP
30 GO 10
END
```

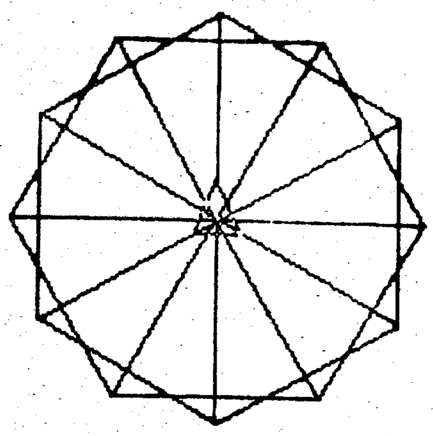


Figure 7.12

To generalize this to a case in which the initial heading is not 0, a line setting the ending condition can be added within the same procedure, while the analogous situation requires two procedures when recursion is involved (see section 5.3.2):

```
TO STAR
5 MAKE "START HEADING
10 TRI
20 RIGHT 30
25 IF HEADING = :START STOP
30 GO 10
END
```

When incrementation is required, however, looping requires an extra step to change the value of the variable. Compare this looping version of COUNTDOWN, for example, with the recursive version, described in Section 7.3.2.

```
TO COUNTDOWN :NUMBER
10 PRINT :NUMBER
20 IF :NUMBER = 0 STOP
30 MAKE "NUMBER :NUMBER - 1
40 GO 10
END
```

5. Turtle Geometry: What the Students Learned

The ways in which students control the movements of the LOGO TURTLE is an important aspect of their behavior in the LOGO learning environment. In this section, we shall describe the range of student behaviors in the "learning environment" of "Turtle Geometry", and relate these behaviors to fundamental understandings in mathematics.

We have organized our observations into six general categories:

1. Qualitative structuring of the number worlds.

2. Quantitative structuring of the number worlds.

3. The "group properties" of the number worlds -- the structuring of mathematical operations.

4 The use of "TURTLE Coordinates" -- the beginnings of differential geometry.

5. The use of coordinate systems -- the "global structure" of the geometric world.

6. Theorems and heuristics -- movement towards formal mathematics.

1. Qualitative Structuring of the "Number World"

The use of numbers as inputs to TURTLE commands forces a student to recognize the different roles for numbers within Turtle Geometry, as well as the different properties of numbers within each role. The primary distinction that a student has to make is between the use of numbers as inputs to FORWARD and as inputs to RIGHT and LEFT. For example, in producing figures, the input to FORWARD determines the size of the figure, while the input to RIGHT determines the shape. As an input to FORWARD, a bigger number produces a "bigger" effect, while as an input to RIGHT, a bigger number usually produces a "different", but not necessarily "bigger" effect. RIGHT 180 reverses the TURTLE's direction, while RIGHT 360 causes no rotation at all (for a T.V. TURTLE!).

Loosely speaking, inputs to FORWARD are mainly "quantitative," while inputs to RIGHT are largely "qualitative." FORWARD is conceptually a "continuous function", while RIGHT is conceptually a "discontinuous function:" FORWARD

90 and FORWARD 91 produce "almost the same effect," while RIGHT 90 and RIGHT 91 can produce extremely different effects in some circumstances.

These discontinuities become apparent when a sequence of instructions is repeated a number of times. It is the effect of a <u>*repeated*</u> *difference in rotation that is "discontinuous" even when two single rotations appear to be very close. Compare two squares made by the commands*

REPEAT [FORWARD 90 RIGHT 90] 4 *and* REPEAT [FORWARD 93 RIGHT 90] 4

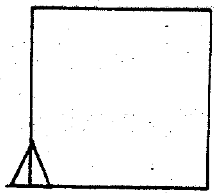


Figure 1.1a

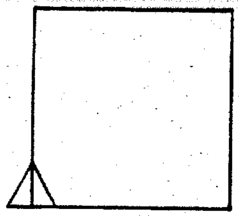


Figure 1.1b

No matter how often these commands are repeated, the results will be similar squares of slightly different sizes. In contrast, consider the effect of a small change in the angle. Compare two figures drawn by the commands

REPEAT [FORWARD 90 RIGHT 90] 4 *and* REPEAT [FORWARD 90 RIGHT 93] 4

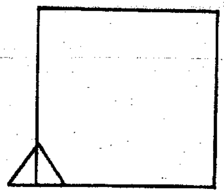


Figure 1.2a

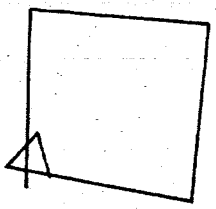


Figure 1.2b

The figure on the right is "almost the same" as the one on the left. If, however, we repeat both FORWARD 90 RIGHT 90 and FORWARD 90 RIGHT 93 a few more times, the differences become more apparent:

REPEAT [FORWARD 90 RIGHT 90] 8 *and* REPEAT [FORWARD 90 RIGHT 93] 8

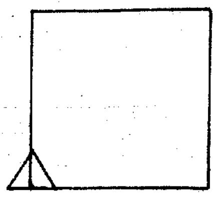


Figure 1.3a

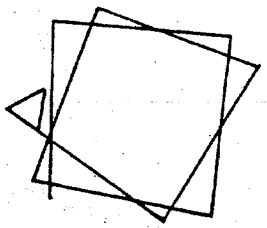


Figure 1.3b

and if we repeat the two pairs of commands a lot more times, the two figures become significantly different in appearance:

REPEAT [FORWARD 90 RIGHT 90] 100 *and* REPEAT [FORWARD 90 RIGHT 93] 100

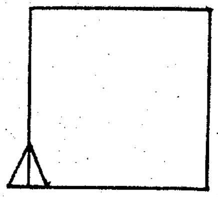


Figure 1.4a

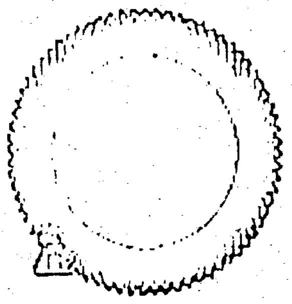
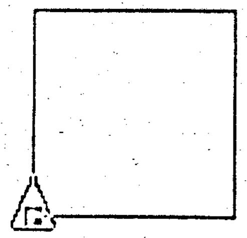


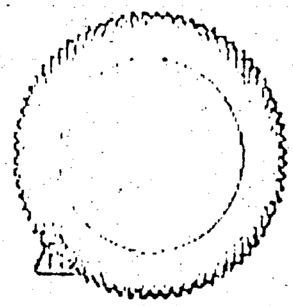
Figure 1.4b

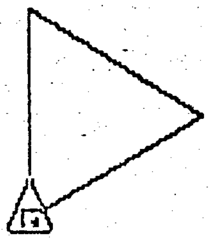
These differences are shown in a more general way by the behavior of the LOGO procedure, POLY:

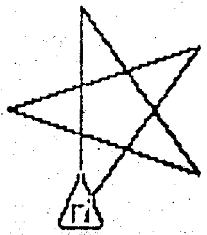
```
TO POLY :SIDE :ANGLE
10 FORWARD :SIDE
20 RIGHT :ANGLE
30 POLY :SIDE :ANGLE
END
```

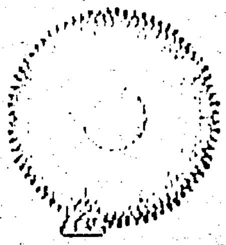
When the command POLY (with two inputs) is given, it will continue to repeat a fixed FORWARD-RIGHT combination, until the computer runs out of storage space. The first input determines the amount the TURTLE will move forward each time; the second determines how much it will turn. Varying the second (angle) input, while holding the first constant, produces dramatically different shapes:

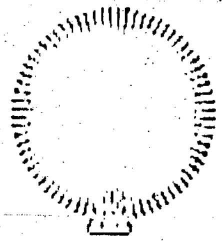

POLY 100 90
Figure 1.5

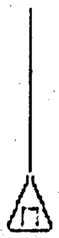

POLY 100 93
Figure 1.6


POLY 100 120
Figure 1.7

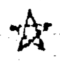

POLY 100 144
Figure 1.8

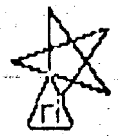

POLY 100 145
Figure 1.9

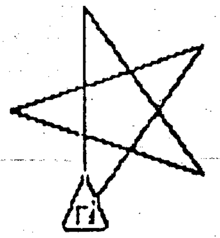

POLY 100 179
Figure 1.10


POLY 100 180
Figure 1.11

On the other hand, varying the first (side) input, while holding the second constant, produces a set of geometrically similar shapes:


POLY 10 144
Figure. 1.12


POLY 50 144
Figure 1.13


POLY 100 144
Figure 1.14

One of the learner's earliest challenges in the control of the TURTLE is to comprehend and make use of this qualitative difference in the effects of numbers as inputs to TURTLE commands. (Other roles of numbers in the LOGO environment have to do with the use of line numbers to order a sequence of steps in a LOGO procedure, and the use of numbers as counters in determining how often a series of steps will be repeated.)

BEHAVIORS OBSERVED:

1.1 Inputs Chosen by Non Geometric Considerations

Numbers used as inputs to both FORWARD and RIGHT commands have no apparent geometric regularity. They may be determined by non-geometric considerations such as ease of typing, or previously familiar number patterns. At this stage of comprehension, interesting effects may be produced, but in an uncontrolled way, subject to unexpected "disasters" such as "OUT OF BOUNDS" messages.

EXAMPLES:

1.1.1 A student may repeat a pair of commands FORWARD 99 RIGHT 99, several times to make an interesting design:

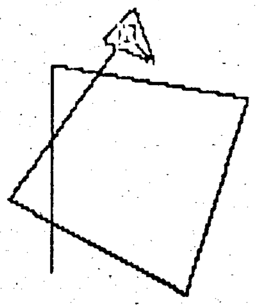
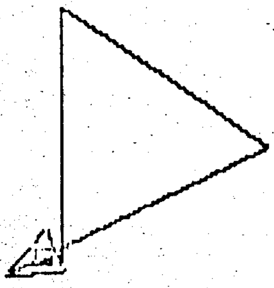
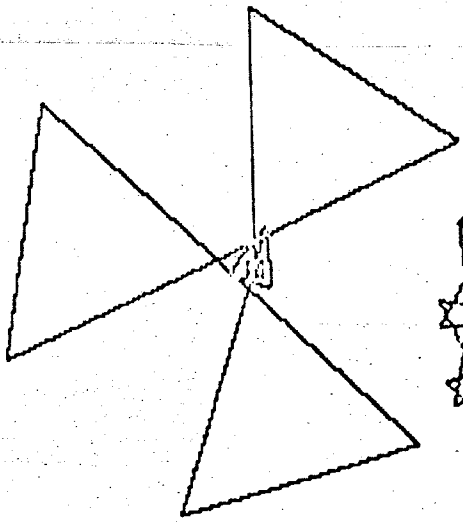


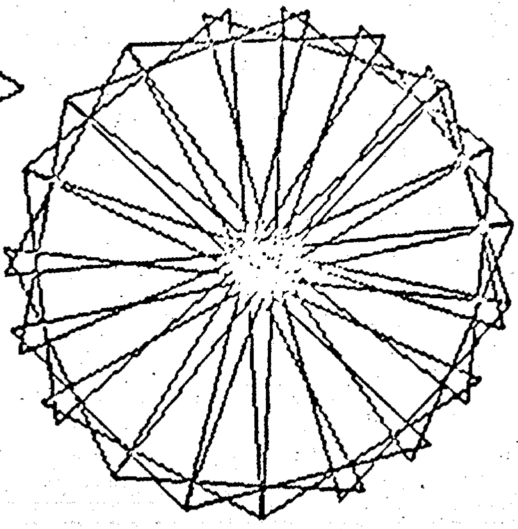
Figure 1.15

1.1.2 A student makes a procedure by using the commands FORWARD 123, RIGHT 123, FORWARD 123, RIGHT 123, FORWARD 123. An interesting design may be made by repeating this procedure a number of times:

```
TO JANE
1 FORWARD 123
2 RIGHT 123
3 FORWARD 123
4 RIGHT 123
5 FORWARD 123
END
```


TO JANE
Figure 1.16


REPEAT [JANE] 3
Figure 1.17


REPEAT [JANE] 30
Figure 1.18

1.2 More Systematic Choices of Inputs

The student's choice of input numbers becomes more systematic, but s/he still does not differentiate between FORWARD/BACK and RIGHT/LEFT commands.

Examples:

1.2.1 One simple regularity consists of reducing all inputs to relatively small numbers (say under 100) in hopes of avoiding "OUT OF BOUNDS" messages.

1.2.2 Another undifferentiated regularity consists of using multiples of ten as inputs, apparently reflecting the observation by the student that small changes in inputs to FORWARD (and to RIGHT and LEFT in cases where polygons are not involved) produce very small effects. At this stage some students repeatedly use a few key "round numbers;"e.g., 30, 50, 90 or 100, for inputs to all turtle commands.

1.3 Differentiation Between Linear and Angular Inputs

The student differentiates between linear and angular uses of numbers. Typically this involves at first the discovery that certain "special angles" produce uniquely desirable effects. Eventually the student clearly differentiates between the two by dropping the use of the "special angle" numbers: 90, 120, 180, etc. as inputs to FORWARD (except in certain isolated particular instances). Some students develop a less significant set of "special inputs to FORWARD": 100, 200, etc. that are not usually used as inputs to RIGHT and LEFT.

EXAMPLES:

1.3.1 The student discovers the significance of 90 degrees as an input to RIGHT and LEFT commands. A typical student project is the construction of a square or rectangle, using 90 degree rotations. (At this point some students still show a blurred differentiation between linear and angular inputs. Many students draw their first squares by typing FORWARD 90, RIGHT 90, FORWARD 90, ...)

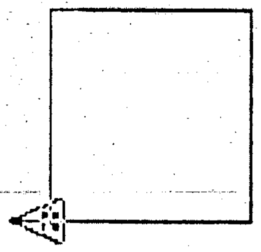
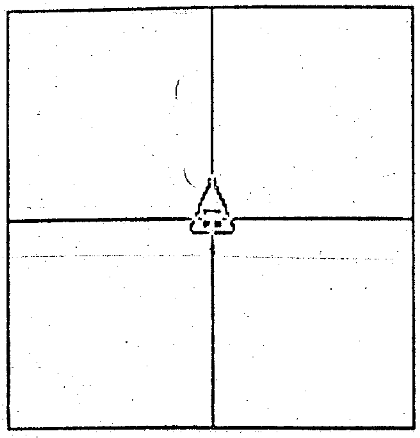
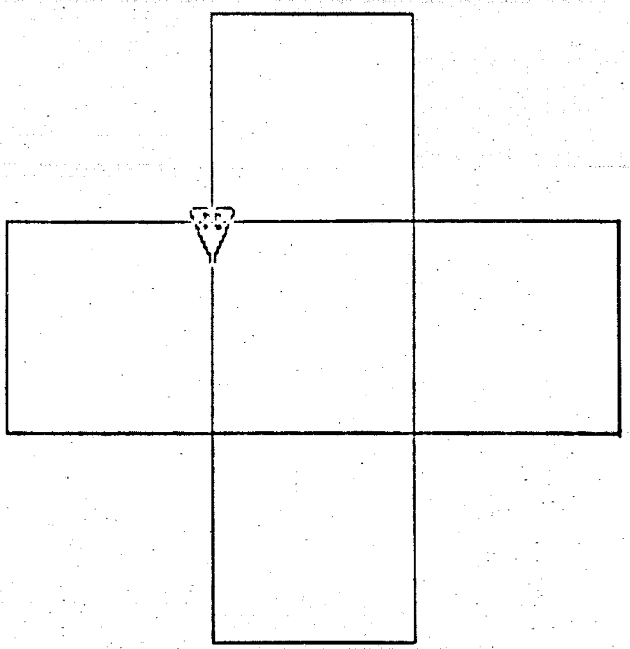
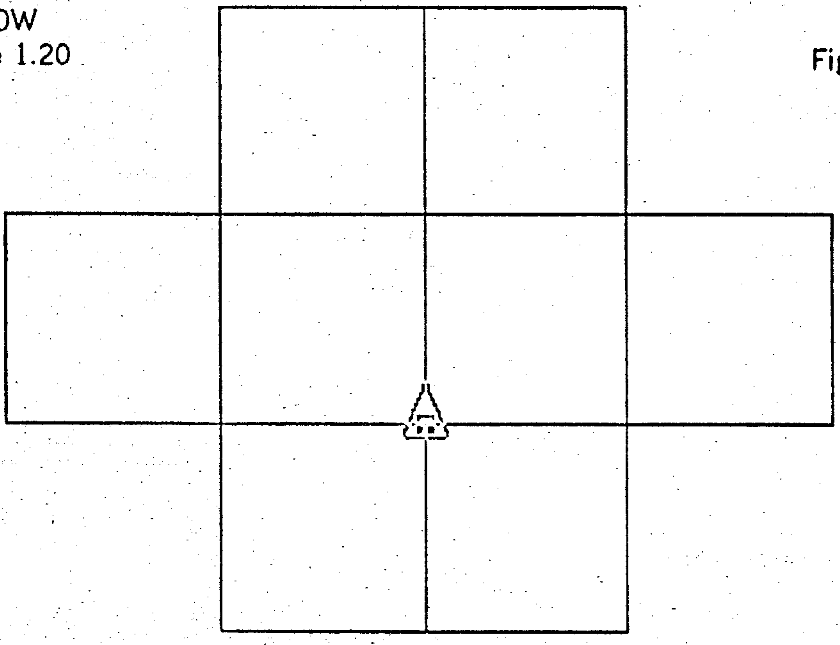


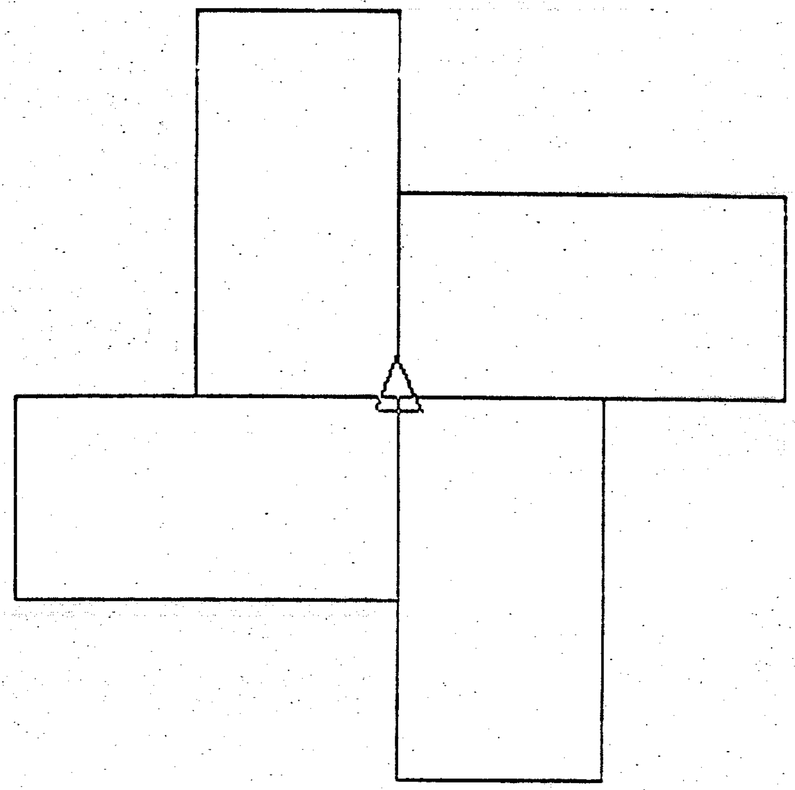
Figure 1.19

1.3.2 A student uses squares and rectangles to make a variety of designs. Student procedures WINDOW, CHAMP, DOUBLECHAMP and CULL are good examples of this type of design:


WINDOW
Figure 1.20


CHAMP
Figure 1.21


DOUBLECHAMP
Figure 1.22



CULL
Figure 1.23

1.3.3 A student uses RIGHT 180 or LEFT 180 to reverse the TURTLE's direction.

1.3.4 The student finds rotations that produce simple closed shapes when repeatedly rotating a figure. Repeating RCIRCLE, RIGHT 60, for example, produces a shape that closes after six repetitions; RCIRCLE, RIGHT 45 closes after eight repetitions.

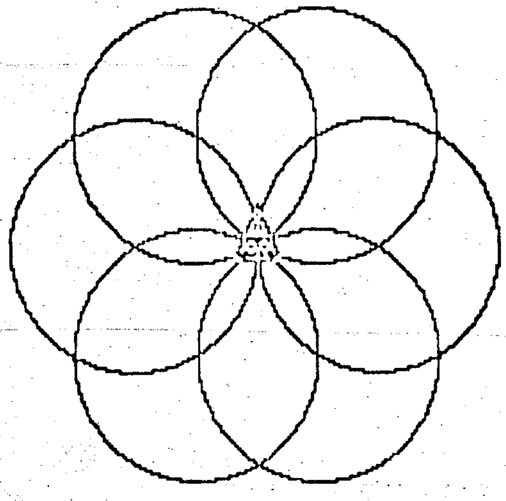


Figure 1.24

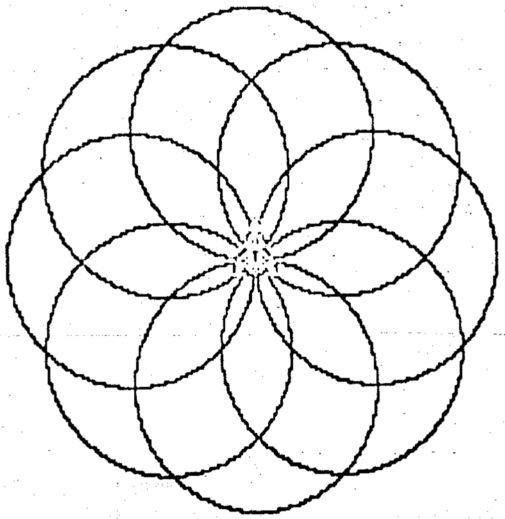


Figure 1.25

1.3.5 The student uses combinations of 45 and 90 degree rotations, or a series of 120 degree rotations to draw a triangle.

```
RIGHT 45
FORWARD 100
RIGHT 90
FORWARD 100
RIGHT 45
RIGHT 90
FORWARD 141
```

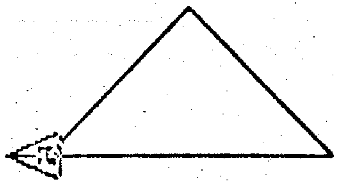


Figure 1.26

or,

```
FORWARD 100
RIGHT 120
FORWARD 100
RIGHT 120
FORWARD 100
```

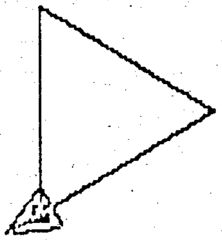


Figure 1.27

1.3.6 A student makes use of "special angles" when carrying out planned designs involving non-right-angles. For example, the "nose cone" of a "rocket" is drawn by making use of rotations of 30 and 120 degrees:

```
FORWARD 100
RIGHT 30
FORWARD 20
RIGHT 120
FORWARD 20
RIGHT 30
FORWARD 100
RIGHT 90
FORWARD 20
```

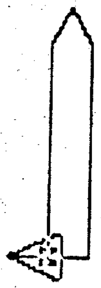
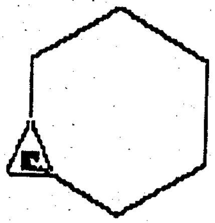
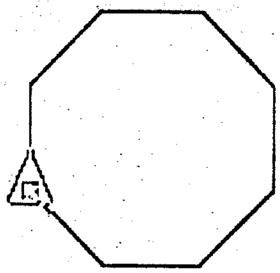
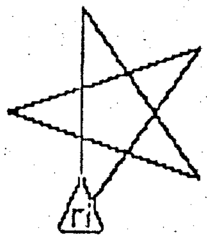


Figure 1.28

1.3.7 A student finds "special angles" useful for drawing regular polygons or stars. Using either a POLY procedure or a sequence of repeated commands, the student finds the angles which will draw a hexagon (60 degrees), an octagon (45 degrees), a five pointed star (144 degrees), etc.


POLY 50 60
Figure 1.29


POLY 50 45
Figure 1.30


POLY 50 144
Figure 1.31

2. Quantitative Structuring of the "Number Worlds"

Many students begin LOGO with a very poor ability to estimate the relative effect of numbers. Splitting the "world of numbers" into "length numbers" and "angle numbers" provides a qualitative structure for that world. Estimating the practical effects of particular numbers provides a quantitative structure. We have observed that most students find it easier to make estimates involving linear numbers than estimates involving angular numbers. The same holds true for other operations with these numbers, to be discussed in section 4 below. This may be due both to a greater ease of visual comparison of linear magnitudes and to the discontinuous effects of angular numbers.

BEHAVIORS OBSERVED:

2.1 Becoming Aware of Limiting Factors

A student's first quantiative structuring of numbers often occurs when he or she becomes aware of certain limiting factors and realizes that certain numbers are too small or too large to be of practical effect in most applications.

EXAMPLES:

2.1.1 Lack of Appreciation of "Orders of Magnitude"

The student discovers that very small inputs to FORWARD/BACK and RIGHT/LEFT produce negligible effects, and shifts to inputs larger than 10 for most FORWARD and RIGHT commands. For a student who has little sense of relative magnitude of numbers, this step may be far from obvious, and may require a great deal of experimentation. Deborah's first LOGO command was FORWARD 10. When she realized that the effect was so tiny as to be almost invisible, her friends urged her to "try a larger number". She responded by typing FORWARD 12. At this point she had no sense how much larger a number had to be to produce an appreciably different effect. (We have found this type of behavior to be more generally prevalent when much younger children (ages 6-9) are introduced to LOGO.)

2.1.2 Coping with "OUT OF BOUNDS" errors

The student discovers that large inputs to FORWARD and BACK may result in "OUT OF BOUNDS" error messages. S/he attempts to predict how large a number can be used as an input without producing an "OUT OF BOUNDS" message.

Eventually s/he is able to create designs that remain "in bounds" by limiting the size of inputs to FORWARD commands.

2.1.3 The Largest Possible Inputs to LOGO Commands

The student discovers upper limits to inputs for RIGHT/LEFT and SPIN comands. In this case, the limits do not correspond to physically observable criteria such as a distance on the display screen, but are based on "arbitrary" limits imposed by the designers of the computer language. Students often find these limits by a process of trial and error.

2.1.4 Finding that 360^0 is the Largest Meaningful Rotation

The student discovers that inputs to RIGHT and LEFT turn the TURTLE more than once around. This is a relatively early discovery for students using a "floor TURTLE" as it can be clearly observed that a rotation larger than 360 degrees turns the TURTLE more than once around. It is more difficult to discover with the "TV TURTLE," because the TURTLE moves instantly to its new position without visibly rotating.

2.2 Estimating Distances

The student develops strategies for estimating the number of TURTLE steps needed to move the turtle to a particular point on the screen. The estimate can be refined by an approach involving successive approximations.

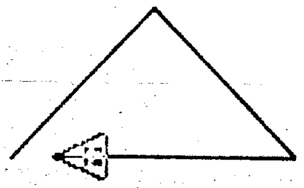
EXAMPLES:

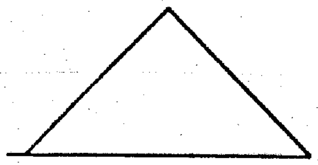
2.2.1 Predicting Large Scale Effects.

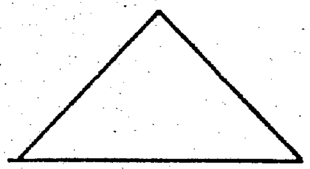
A student begins to be able to predict with some degree of accuracy the effect of particular inputs to FORWARD such as FORWARD 50 or FORWARD 100.

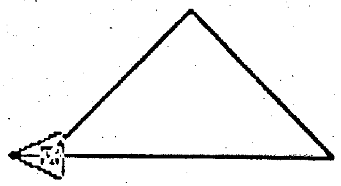
2.2.2 Finding Precise Distances by Successive Approximations

A student moves the turtle to a particular location, by using a trial and error approach. For example, Kevin constructed a right isoceles triangle, each of whose equal sides was 100 turtle steps. To complete the hypotenuse, he used the following sequence of commands: FORWARD 100 (too small), FORWARD 50 (too big), BACK 5 (still to big) and BACK 5 (seemed just right).


FORWARD 100
Figure 2.1


FORWARD 50
Figure 2.2


BACK 5
Figure 2.3


BACK 5
Figure 2.4

A student may also use a method of successive approximations to find the distance from the center, to the edge of the screen. The following sequence of commands is a typical one. Starting with the TURTLE in the center of the screen, a student might type:

```
?FORWARD 150
?FORWARD 100
OUT OF BOUNDS
?FORWARD 50
?FORWARD 50
OUT OF BOUNDS
?FORWARD 20
OUT OF BOUNDS
?FORWARD 1
OUT OF BOUNDS
```

Since the student can not use an input lower than 1, s/he realizes that the TURTLE is now situated at the edge of the screen. By adding all the steps that did not result in an OUT OF BOUNDS message (the steps underlined above), the student concludes that the distance from the center to the top of the screen is

150 + 50, or 200 TURTLE steps.

2.3 Estimating Rotations

The student develops strategies for estimating the amount of rotation necessary to aim the TURTLE in a particular direction. The estimate can be refined by a method involving successive approximations.

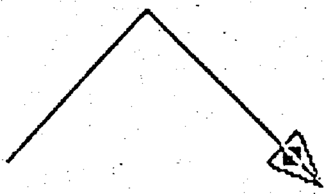
EXAMPLES:

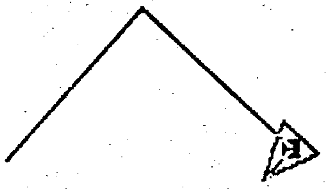
2.3.1 Estimating Rotations Less than 90 Degrees

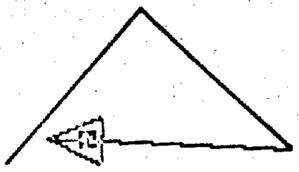
A student estimates rotations of less than 90 degrees with an accuracy of $\pm$ 10 degrees. Trial and error techniques can be used to refine the estimate. A student may want to aim the TURTLE towards the upper right hand corner of the screen. S/he might choose to turn the TURTLE RIGHT 50, knowing that it has to be less than 90 degrees. By moving the TURTLE forward, the student can see how accurate the estimate was.

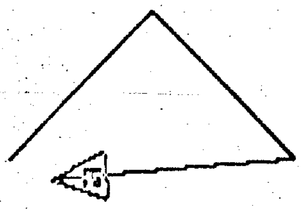
2.3.2 Estimating Rotations Greater Than 90 Degrees

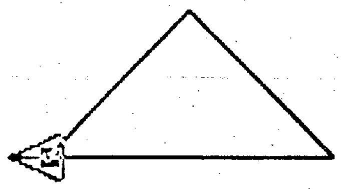
A student estimates rotations of more than 90 degrees by first turning the TURTLE through a rotation of 90 degrees and then estimating how much further rotation is necessary. For example, a student constructing a right isoceles triangle who has reached the end of the second equal leg might try the sequence of rotations shown below. Since it might be difficult for the student to <u>see</u> when the TURTLE is aimed in exactly the right direction, these steps may be combined with some FORWARD commands, in order to see exactly how close to the correct ending point the TURTLE will come.


Starting Point
Figure 2.5


RIGHT 90
(too small)
Figure 2.6


RIGHT 50
(too large)
Figure 2.7

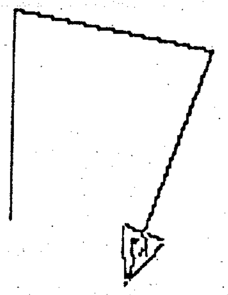

LEFT 10
(still too large)
Figure 2.8


RIGHT 5
(just right)
Figure 2.9

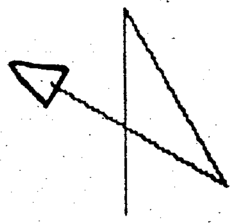
2.3.3 Estimating Repeated Rotations by Trial and Error

A student develops a trial and error strategy for estimating a repeated rotation. Consider the process of constructing an equilateral triangle. The student must first realize that the same FORWARD and RIGHT steps have to be repeated three times. The following sequence of attempts is a typical one for solving this problem:

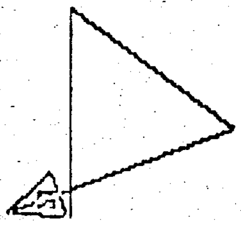
1.
```
FORWARD 100
RIGHT 100
FORWARD 100
RIGHT 100
FORWARD 100:
```


rotation too small
Figure 2.10

2.
```
FORWARD 100
RIGHT 150
FORWARD 100
RIGHT 150
FORWARD 100:
```

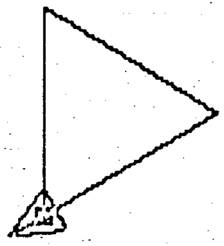

rotation too big
Figure 2.11

3.
```
FORWARD 100
RIGHT 125
FORWARD 100
RIGHT 125
FORWARD 100
```



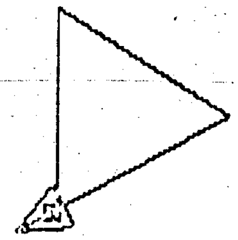
still too big
Figure 2.12

4.
```
FORWARD 100
RIGHT 120
FORWARD 100
RIGHT 120
FORWARD 100
```

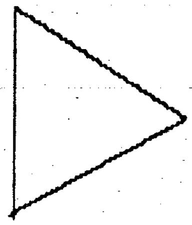


just right!
Figure 2.13

It should be noted that some students using this method may decide that RIGHT 119 or RIGHT 121 is the correct solution to their problem. Unless they "hide" the TURTLE and carefully examine the resulting figure, the differences between such a "solution" and the correct rotation of 120 degrees may not be apparent to them.



REPEAT [FORWARD 100 RIGHT 121] 3
(lines appear to meet exactly)
Figure 2.14



HIDETURTLE
(lines overlap slightly)
Figure 2.15

The difference becomes significant when a student wants to repeat the figure, or relate it to other figures (to construct a "house" for example).

3. The Group Properties of the Number Worlds: The Structuring of Mathematical Operations.

Piaget has demonstrated the significance of the way in which children develop intellectual structures to deal with mathematical operations. The most significant of these operations have been described by Piaget as "composition" (e.g. the additive property of numbers) and "inversion" (e.g. formation of the inverse or negative of an operation). In the LOGO environment we can observe the development (or lack of development) of these intellectual structures in the following ways:

-- The use of "composition" is seen in the ways that students combine or aggregate TURTLE commands so that two commands, FORWARD 50, FORWARD 50, may be combined as FORWARD 100. Angular rotations can be similarly aggregated.

--The use of "inversion" is seen in the ways that students are able to use BACK as an inverse to FORWARD and LEFT as an inverse to RIGHT;

--The combination of these operations is seen when students aggregate a series of commands such as FORWARD 100 BACK 30 FORWARD 10 into one command, FORWARD 80, or a similar series with respect to rotation, LEFT 90 RIGHT 20 LEFT 10, into LEFT 80 (It is our finding that that students tend to combine these operations more readily with linear numbers than with angular numbers);

--The use of inversion is particularly powerful in projects involving SPIN commands

--The use by students of the particular properties of the "rotational group" -- its modularity with respect to 360 degrees.

Among our sample of sixteen sixth grade students, we have observed a wide variety of behaviors with respect to these operations: from students who never made use of composition and inversion, to students who use them inconsistently and tentatively, to students who learned to use them within certain limited contexts, to students who used them "automatically" and regularly as part of their LOGO work. We believe that this particular set of behaviors provides an unusually good "window" through which we can view an important aspect of the children's cognitive development.

Instances of this kind of behavior can often be seen by an examination of a *student's dribble files. Typically a student's exploratory work on a project includes many steps which could either be combined at a later point, or eliminated as unnecessary. By taking note of just how readily the student combines or eliminates these steps, and under what circumstances, we can get a good sense of the student's use of mathematical operations.*

BEHAVIORS OBSERVED:

3.1 Students Who do Not Make Use of Operations

Many students do not easily arrive at the strategy of simplifying their work by combining several commands into one.

EXAMPLES:

3.1.1. Failure to Combine Steps

A student will determine by trial and error that the final leg of a right isoceles triangle can be built by a series of FORWARD steps: FORWARD 100 FORWARD 40 FORWARD 1. (See example 2.2.2 above). When writing a procedure to draw this triangle, the student would include three separate steps to draw the final line:

```
TO TRI
1 RIGHT 45
2 FORWARD 100
3 RIGHT 90
4 FORWARD 100
5 RIGHT 90
6 RIGHT 45
7 FORWARD 100
8 FORWARD 40
9 FORWARD 1
END
```

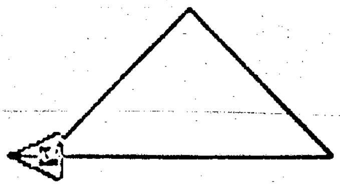


Figure 3.1

3.1.2 The Example of Deborah

The student who exhibited this type of behavior most consistently was Deborah. Although she carried out a number of projects in Turtle Geometry she was never observed to combine steps. If she was pleased with an exploration, she always copied all her original steps exactly as she had first used them. Thus although she seemed to understand that two turns of RIGHT 90 and RIGHT 90 were always needed to "turn the TURTLE around," she never combined these in one step as RIGHT 180. Similarly, when she made a mistake, turning LEFT 90, where she needed a right turn, she would correct it by turning the TURTLE RIGHT 90 to bring it back to its original position, and then turning it an additional RIGHT 90 to achieve the desired direction. While this showed a limited ability to use the inverse operation with regard to angles, it should be noted that Deborah always copied this series of steps as LEFT 90, RIGHT 90, RIGHT 90, rather than as simply RIGHT 90.

Students who, like Deborah, had a limited strategy of copying all experimental steps literally, sometimes had fewer bugs than students who attempted to combine steps and made arithmetical or copying errors.

3.2 Transitional Phase -- Incomplete and Inconsistent Uses of Operations.

Students usually begin combining forward steps before combining rotations (sometimes a good deal before). At first, they may be inconsistent, combining in some cases and not in others. They begin the use of inversion by realizing that a forward step can be reversed by an equal BACK step and that a RIGHT turn can be reversed by an equal LEFT turn. It is somewhat later that students are consistently able to combine FORWARD and BACK steps by subtracting the BACK from the FORWARD inputs. Many students never achieve the further step of combining RIGHTs and LEFTs, by subtracting one from the other.

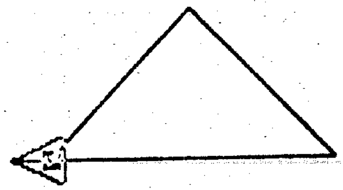
EXAMPLES:

3.2.1 Combining FORWARD Steps but Not Rotational Steps

An example of partial use of composition is seen in Kevin's construction of a right isoceles triangle, described above (section 2.2.2). Compare Kevin's initial steps with his final procedure:

Kevin's Ten Original Steps

```
RIGHT 45
FORWARD 100
RIGHT 90
FORWARD 100
RIGHT 90
RIGHT 45
FORWARD 100
FORWARD 50
BACK 5
BACK 5
```



Kevin's Procedure (Seven Steps):

```
TO OF
1 RIGHT 45
2 FORWARD 100
3 RIGHT 90
4 FORWARD 100
5 RIGHT 90
6 RIGHT 45
7 FORWARD 140
END
```

Figure 3.2

Kevin easily combined four FORWARD and BACK steps in one FORWARD step, making use of both composition and inversion, but at this point he did not combine rotational steps at all.

3.2.2 Jimmy's Boat

A more complex example of inconsistent use of composition and inversion is shown by Jimmy's first project in which he drew the BODY of a boat. First he drew the boat by a series of exploratory steps, arrived at by a process involving a good deal of trial and error. Compare the original steps as written in Jimmy's notebook with his procedure:

Jimmy's 22 Original Steps:

```
LEFT 90
FORWARD 60
RIGHT 40
FORWARD 20
RIGHT 140
PENUP        ┐
FORWARD 70   ├──   A
BACK 70      │
PENDOWN      ┘
FORWARD 70
FORWARD 10
RIGHT 92     ┐
RIGHT 2      ├──   B
RIGHT 4      ┘
FORWARD 10   ┐
FORWARD 4    ┘──   C
RIGHT 180
FORWARD 14
RIGHT 89     ┐
LEFT 178     ┘──   D
FORWARD 20
LEFT 10
```

Jimmy's 15 Step Procedure:

```
TO BODY
1 LEFT 90
2 FORWARD 60
3 RIGHT 40
4 FORWARD 20
5 RIGHT 140  ──────   A   (removed)
6 FORWARD 70
7 FORWARD 10
8 RIGHT 96   ──────   B
9 FORWARD 14 ──────   C
10 RIGHT 180
11 FORWARD 14
12 RIGHT 89  ┐
13 LEFT 178  ┘──      D
14 FORWARD 20
15 LEFT 10
```

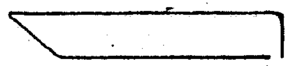


Figure 3.3

Jimmy made use of both composition and inversion in reducing his original list of 22 steps to a procedure of 15 steps. When he used composition in reducing the group of steps labelled B, RIGHT 92, RIGHT 2, RIGHT 4, to the single step, RIGHT 96, Jimmy made a small arithmetical error. When Jimmy combined the steps listed as group A, he made use of inversion to realize that he could eliminate the steps PENUP, FORWARD 70, BACK 70, PENDOWN, but he did <u>not</u> combine the step, FORWARD 70, with the next step, FORWARD 10. On the other hand, he did combine the steps (labeled C), FORWARD 10, FORWARD 4, replacing them with FORWARD 14.

What Jimmy did with the steps labeled D, RIGHT 89, LEFT 178, is typical of many students at this stage. While working, he realized that he had turned the wrong direction when he typed RIGHT 89. In his head he calculated that to compensate for this he would need to turn LEFT 89 twice, for a total turn of LEFT 178, which he used. He did not take the next possible step and replace his two steps by one, LEFT 89.

3.2.3 Dealing With Copying Bugs.

A common difficulty exhibited by students who are at this transitional stage is problems with copying combined with faulty arithmetic. We have seen an example above, where Jimmy combined RIGHT 92 RIGHT 2 RIGHT 4 as RIGHT 96. In this case the results were inconsequential. Often, however, similar situations have caused rather confusing bugs.

Laura usually had difficulty predicting whether to turn the TURTLE to the right or left. Once she made a choice, however, she could easily reverse it. If she mistakenly typed RIGHT 90, for example, she would then follow it by typing LEFT 90, LEFT 90. This might occur several times within one short project. When Laura copied the steps over into her notebook she simplified by writing LEFT 90, eliminating the "unneeded steps," RIGHT 90, LEFT 90. Unfortunately, she often made mistakes in copying, and since she had many opportunities for such errors, extremely puzzling bugs sometimes occured in her final procedures. Since the original steps had been erased from the text display screen, and since her notebook reflected only her "simplified" version, she had no easy way of debugging her procedure, other than starting all over again.

Deborah's approach of copying all steps could have been very useful to Laura. Since Laura was conceptually far beyond Deborah and since Laura was extremely reluctant to do "extra work", she persisted in her scheme of eliminating "unneccessary" steps as she copied. Laura tended to "explain" bugs that arose in this way as inconsistencies of the computer.

3.3 Consistent Use of Mathematical Operations.

Some students came to incorporate these operations into their work quite simply and easily. Although errors might occassionally be made in arithmetic or copying, it usually became quite clear when a student understood these ideas with no difficulty.

EXAMPLES:

3.3.1 A simple example can be taken from Gary's work. The following sequence of statements occured in his dribble file:

```
RIGHT 99
99 + 99
YOU DON'T SAY WHAT TO DO WITH 198
LEFT 198
```

Later, when Gary incorporated this as part of a procedure, he simply replaced these steps by LEFT 99. Gary had turned the TURTLE in the wrong direction and wanted to both undo the wrong turn and make the correct turn in one combined step. He used the computer to carry out the necessary arithmetic and then typed the correct command. When copying these steps over, he remembered the process, and without further ado, used only the correct step, LEFT 99, in his procedure. Compare this with Jimmy, who kept the two steps RIGHT 89 LEFT 178, as part of his procedure, or with Laura's regular confusion in copying.

3.3.2 Sophisticated Uses of Inversion

An example of a more sophisticated use of inversion is the process of moving the TURTLE over without changing its orientation. In this case the LEFT/RIGHT inversion is separated by one or more intervening steps. This is necessary when the student wants to repeat a figure two or more times; for example, three houses could be drawn by the procedure HOUSES, with its subprocedures HOUSE and MOVEOVER:

```
TO HOUSES              TO MOVEOVER
10 HOUSE               10 PENUP
20 MOVEOVER            20 RIGHT 90
30 HOUSE               30 FORWARD 120
40 MOVEOVER            40 LEFT 90
50 HOUSE               50 PENDOWN
END                    END
```

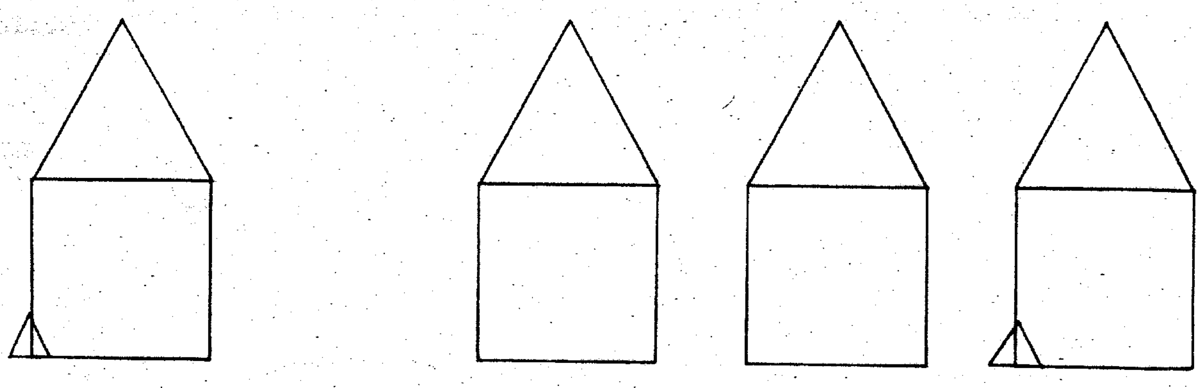


Figure 3.4

As part of his STARSHIP project, GARY created two procedures like MOVEOVER, MO and MOV, each the inverse of the other:

```
TO MO                 TO MOV
10 RIGHT 90           10 LEFT 90
20 FORWARD 100        20 FORWARD 100
30 LEFT 90            30 RIGHT 90
END                   END
```

3.4 The Use of Inverse SPINs

Another example of the use of operations occurs in projects involving the SPIN command. SPIN, followed by a number (e.g. SPIN 100) causes the TURTLE to spin at a fixed rate in the clockwise direction. Once the TURTLE is spinning it will remain spinning about its original center of spin. Any additional non-spin commands will be carried out so as to maintain the rate of spin of the entire design. SPIN commands allow the usual operations of composition and inversion. Additional SPIN commands result in increased or decreased rate of spin in an additive manner. If the sum of a series of SPINs is negative, the TURTLE will spin in a counter clockwise direction. If the sum is zero the TURTLE will stop spinning. The inverse of a SPIN command is that same command with a negative input.

To understand how this works, consider the command SPIN 100 FORWARD 100. This will cause the TURTLE to draw a line of length 100, spinning about the origin. Any other commands or procedures added at that point will also be spinning with the original line.

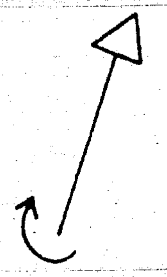


Figure 3.5

EXAMPLES:

3.4.1 A Car With Spinning Wheels

In one of his projects, Dennis drew a car with spinning wheels. This meant that he had to make a circle which would spin about its own center, then return the TURTLE to the center of the circle and make it stop spinning so that the (non-spinning) TURTLE could be moved over to draw the rest of the car. His procedure, WHEEL, made use of inversion with respect to distances, angles and spins:

```
TO WHEEL
10 SPIN 100
20 PENUP FORWARD 40
30 RIGHT 90 PENDOWN
40 RCIRCLE 40
50 LEFT 90 PENUP
60 BACK 40
70 SPIN -100
END
```

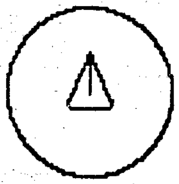


Figure 3.6

This procedure leaves the TURTLE, unmoving, in the center of a spinning circle.

3.4.2 A "ferris wheel" design makes use of positive and negative SPINs. After typing SPIN 100, FORWARD 100 the student types SPIN -100. This leaves the TURTLE facing straight up at all times (a total spin of zero) while moving around at the end of a spinning line. Adding a "car" which hangs vertically makes one arm of a "ferris wheel". Adding seven more "cars" each separated by 45 degrees results in a simulation of the motion of a ferris wheel.

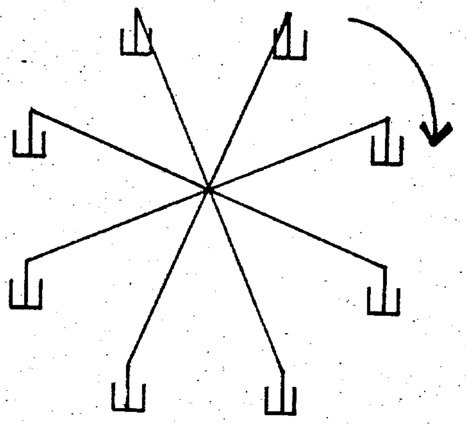


Figure 3.7

3.4.3 A Cowboy's "Lasso"

Harriet's procedure LASSO, (described in detail in Chapter 8 of Part III) required the "undoing" of two SPINs, each spinning about a different origin. This required a fairly sophisticated use of inversion involving combinations of angles, distances and spins, similar to, but more complex than Dennis's use of inversion in his WHEEL procedure.

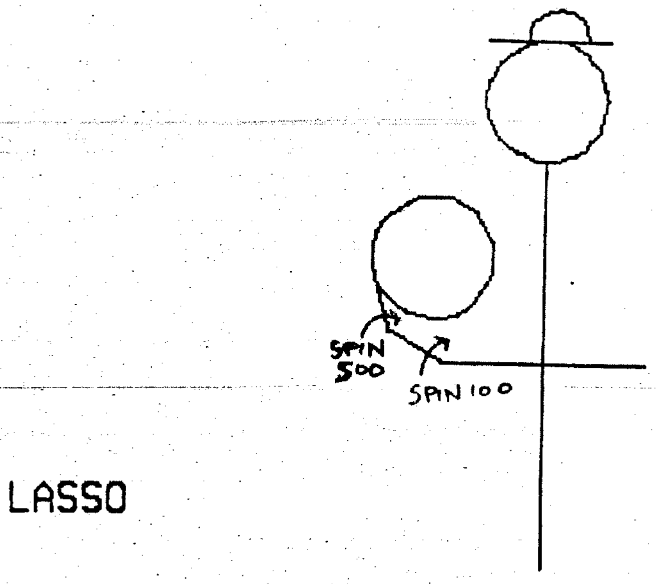


Figure 3.8

3.5 The use by students of the 360 degree modularity of the rotational group.

The group of rotational numbers has a unique property which distinguishes it from a linear number system -- its modularity with respect to a "complete rotation" of 360 degrees. This has some interesting consequences as far as both composition and inversion are concerned. With regard to composition, it has the effect that as rotations are combined, the total rotation increases until it is 360 degrees. Beyond 360 degrees, the orientation of the TURTLE is the same as if it had been turned 360 degrees less. For example two rotations of RIGHT 90 turn the TURTLE twice as far as one RIGHT 90. Adding another RIGHT 90 turns the TURTLE three times a far as one RIGHT 90. Adding a fourth RIGHT 90 however produces no net effect at all! A fifth RIGHT 90 has the same effect as one RIGHT 90, and so on.

With regard to inversion, this modularity means that particular steps can have more than one inverse. The inverse of RIGHT 90 is LEFT 90. It is also RIGHT 270, or LEFT 450. Two steps that are conceptually quite different, for example, turning the TURTLE RIGHT 270 and turning the TURTLE LEFT 90, can have exactly the same effect.

One would expect that some students would take a long time to become *aware of these properties, and even longer to make use of them. Deborah, for instance, was never aware that the same effect could be obtained by one LEFT 90 as by three RIGHT 90s. Gary, at the other extreme, could make use of the equivalence of the two with ease, whenver it was useful to do so.*

We will not discuss specific behaviors here, as most of them belong not to the realm of composition and inversion, as discussed in the section, but more properly to student use of coordinate systems, discussed in Section 5, and on student use of theorems and heuristics, discussed in Section 6 below.

4. The Use of Turtle Coordinates -- The Beginnings of Differential Geometry.

When we discuss the use of Turtle Coordinates, we mean the understanding that the specific consequences of one particular TURTLE action, dependent on a specific position and heading, can be used to predict a more distant effect, achieved by a sequence of such actions. This involves the extrapolation of an immediate effect, into a global consequence, and is an important example of mathematical reasoning. The most common examples of this are the construction of a "LOGO circle," changing the rate of curvature of a circle, and the extensions of a circle -- the POLY and POLYSPI procedures. These activities can help a student develop an intuitive feeling for differential geometry, laying the foundation for an entirely new computational approach to the subject of calculus.*

BEHAVIORS OBSERVED

4.1 The Construction of a "LOGO circle."

Many students who have been introduced to the LOGO TURTLE, and to its basic commands (FORWARD, BACK, RIGHT and LEFT) find themselves asking, "Can the TURTLE draw a circle?"

A teacher's response to this is usually to suggest that the student "play TURTLE:" get up and walk in a circle, and try to make a description of what s/he is doing. Some students spontaneously describe their behaviors as "keep going forward a little and turning a little." These students may then be able to immediately translate this description to a series of TURTLE commands: FORWARD 1, RIGHT 1, FORWARD 1, RIGHT 1, ... or FORWARD 5, RIGHT 5, FORWARD 5, RIGHT 5, ... Repeating such a series of commands, 360 and 72 times, respectively, will produce a many sided polygon that is visually indistinguishable from a circle.

****
*See Abelson and diSessa, Turtle Geometry, MIT Press. To be published in 1980.
****

Other students have more difficulty separating the "forward" and "turn" aspects of walking in a circle. This may be because they move forward and turn at the same time when they walk in a circle. If the teacher reminds the student that s/he has to try to "be the TURTLE" which can only "do one thing at a time," the student is usually able to recognize that forward and turn have to be used alternatively by the TURTLE. Some students still do not realize that the inputs to FORWARD and RIGHT must be uniform, in order to achieve a circle. Often some direct experimentation with the screen TURTLE is necessary before students realize that the TURTLE steps that are repeated must be small and uniform, in order to produce a circle.

For many students, their work in drawing a LOGO circle is also their first contact with writing "patterned" procedures -- procedures that make use of a REPEAT command, looping or recursion. Learnings in this area are described in Chapter 4, Section 6 of this report.

EXAMPLES

4.1.1 Circles of Different Curvature

Most students draw their first circle by using the same input for both FORWARD and RIGHT commands. It is not until they wish to vary the size of the circle, that students begin to realize that the forward and turn inputs can be different and still produce a circle. Typically, a student will try to increase the size of a circle by increasing both inputs the same amount. Surprisingly, FORWARD 2, RIGHT 2... FORWARD 1, RIGHT 1, and even FORWARD 10, RIGHT 10, draw almost identical circles.

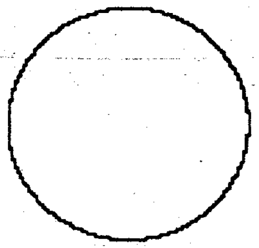


Figure 4.1

It is only when a student can separate the effects of the FORWARD and RIGHT commands that s/he can change the size of the circle.

Darlene's first successful circle used different inputs for FORWARD and RIGHT: RIGHT 2, FORWARD 3, ... When she wanted to make a smaller circle she tried reducing the inputs to RIGHT 1, FORWARD 2. This drew a larger circle. She then tried RIGHT 3, FORWARD 2, which was smaller than her original. Her next circle was RIGHT 3, FORWARD 3, which was larger than the previous one, but smaller than her first circle. RIGHT 4, FORWARD 2, produced the smallest circle of all. In this way Darlene was gradually able to control the curvature of her circles.

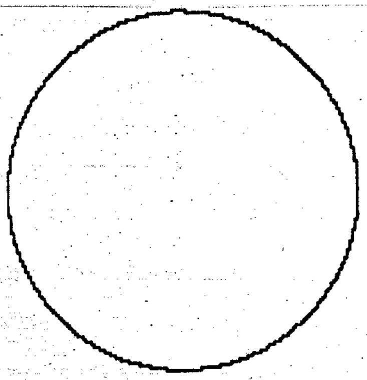


Figure 4.2
RIGHT 2 FORWARD 3

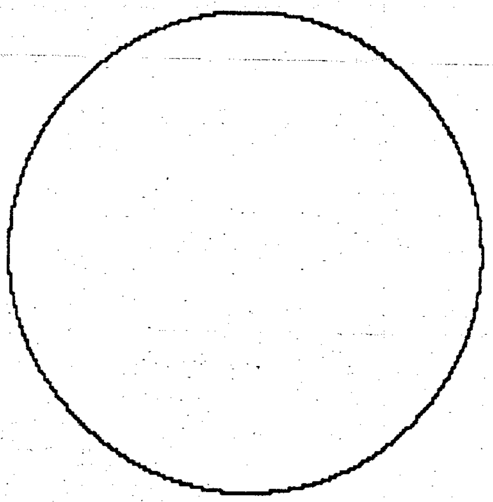


Figure 4.3
RIGHT 1 FORWARD 2

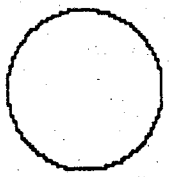


Figure 4.4
RIGHT 3 FORWARD 2

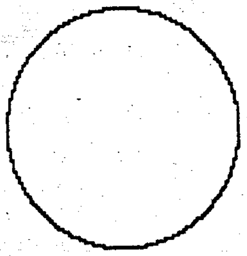


Figure 4.5
RIGHT 3 FORWARD 3

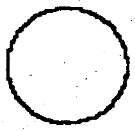


Figure 4.6
RIGHT 4 FORWARD 2

4.2 The Use of POLY Procedures.

The POLY procedure offers students the opportunity to experiment with curvature:

```
TO POLY :SIZE :ANGLE
10 FORWARD :SIZE
20 RIGHT :ANGLE
30 POLY :SIZE :ANGLE
END
```

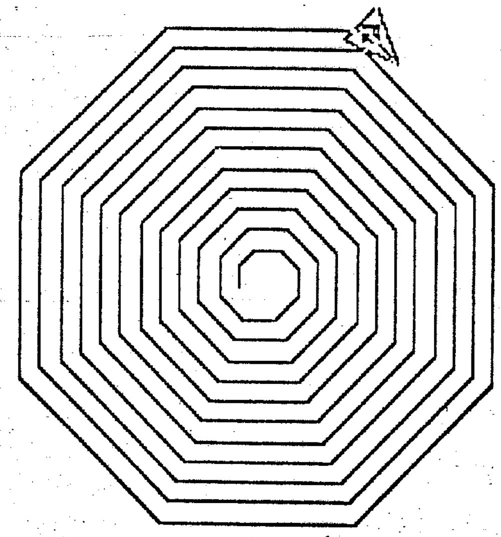
POLY procedures allow a student to construct many different shapes making use of repetition of a fixed distance and turn. This allows the student to experiment more easily with the global effects of local changes and to experience the effects of changing "curvature". Aspects of student work with POLY procedures are also discussed in Sections 1, 3.5, and 6.4 of this chapter, and in Section 6 of chapter 4, which describes student behaviors in learning to write "patterned" procedures.

4.3 Extensions of POLY

Procedures which increment the distance or angle variables in a POLY procedure, provide a further opportunity for investigation of the global effects of local actions. Consider the effects of a procedure called POLYSPI:

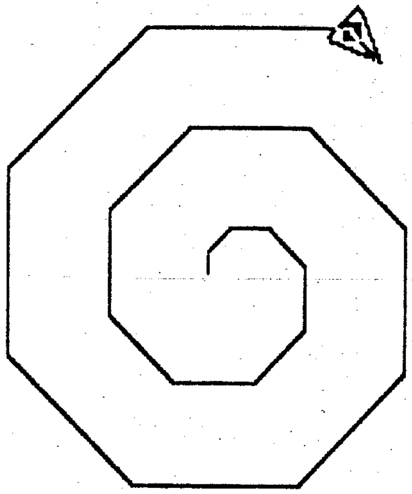
```
TO POLYSPI :SIDE :ANGLE :INCREMENT
10 FORWARD :SIDE
20 RIGHT :ANGLE
30 POLYSPI :SIDE + :INCREMENT :ANGLE :INCREMENT
END
```

POLYSPI produces continuous changes in curvature, as the size of the FORWARD step increases or decreases uniformly. This is similar to the continuous effect of small changes in a FORWARD step discussed in Section 1, above.



POLYSP 10 45 1

Figure 4.7



POLYSPI 10 45 5

Figure 4.8

Another procedure, called INSPI, produces discontinuous changes in curvature, by uniformly increasing or decreasing the angle variable. This is similar to the discontinuous effects of making a small change in a rotational step, discussed above in Section 1.

```
TO INSPI :SIDE :ANGLE :INCREMENT
10 FORWARD :SIDE
20 RIGHT :ANGLE
30 INSPI :SIDE :ANGLE+ :INCREMENT :INCREMENT
END
```

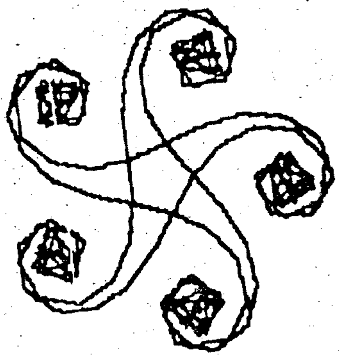


Figure 4.9

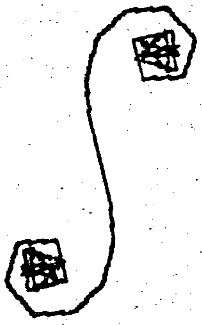


Figure 4.10

5. The Use of Coordinate Systems: the Global Sructure of the Geometric World.

Students working in the area of Turtle Geometry begin by becoming aware of the local geometry of the TURTLE. It is necessary for them to take into acount the TURTLE's position and heading in order to accomplish even the simplest tasks. Through their work on specific projects, they come to make use of -- sometimes even invent -- global structures of their own. They can use these structures to solve problems that require that they take into account aspects of geometry other than the turtle's immediate position and heading. In this section we describe student behaviors that relate to structures that can be used to organize two dimensional space: Turtle Coordinates, domain specific coordinate systems, standard cartesian coordinates, and various types of polar or angular coordinates.

5.1 Drawing a "Bear" Using Three Different Coordinate Systems

As an example of the way in which the same LOGO project could be carried out utilizing different coordinate systems, consider this cartoon drawing of the head of a "bear":

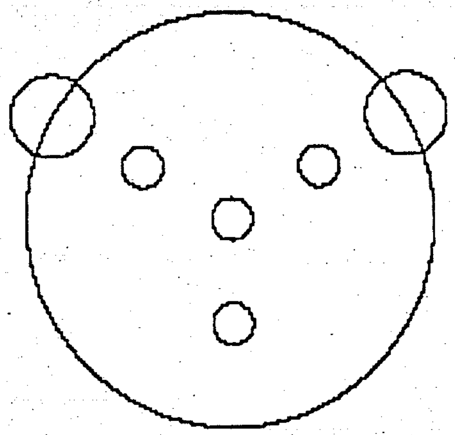


Figure 5.1

The project could be carried out without any use of coordinate systems, by simply moving the turtle to each location where a circle is to be drawn, and using the command RCIRCLE, with the appropriate input to draw circles of different radii. Without some advance planning, however, the result is likely to lack the symmetry of the original plan. Albert's procedure, KEITH and Laura's procedure, FACE, were carried out without benefit of any coordinate system, and are typical of this type of effort.

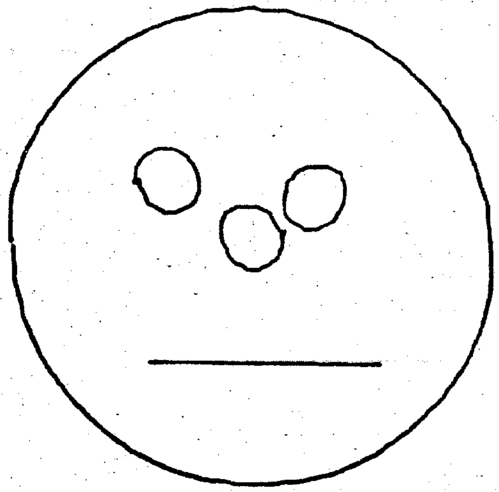


Figure 5.2

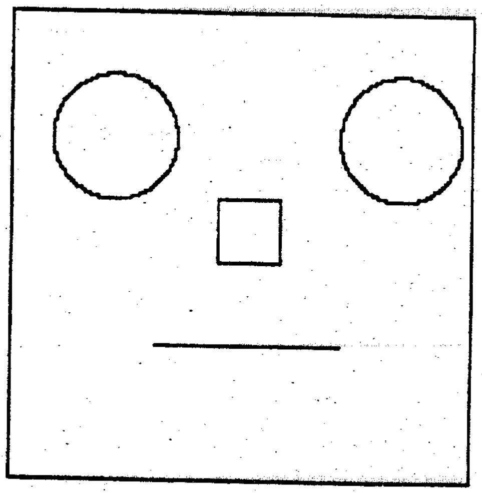


Figure 5.3

To create a symmetrical "bear" it would be simpler to make use of a circle procedure which started from its own center, rather than the procedures RCIRCLE and LCIRCLE which are given to our students as "primitives". Such a procedure can be defined as follows:

```
TO CIRCLE :RADIUS
10 PENUP FORWARD :RADIUS
20 RIGHT 90 PENDOWN
30 RCIRCLE :RADIUS
40 LEFT 90 PENUP
50 BACK :RADIUS
END
```

5.1.1 Drawing the Bear with Domain Specific or Intrinsic Coordinates

Intrinsic coordinates are coordinates developed for a particular figure, and involve developing a structure which allows the TURTLE to move around the figure itself, as part of the process of constructing it. In order to use intrinsic coordinates for this project, a student would require an arc procedure, ARCR :RADIUS :ANGLE, whose two inputs determine the radius and the angle of the arc. This procedure allows the TURTLE to move from point to point along a particular circle, rather than to try to move the TURTLE directly to the desired point. In figure 5.4, the command ARCR 100 120 *allows the TURTLE to move along the arc from A to B, rather than* *straight across the circle.*

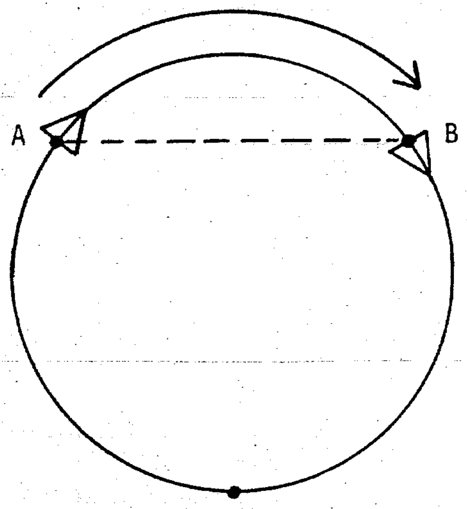


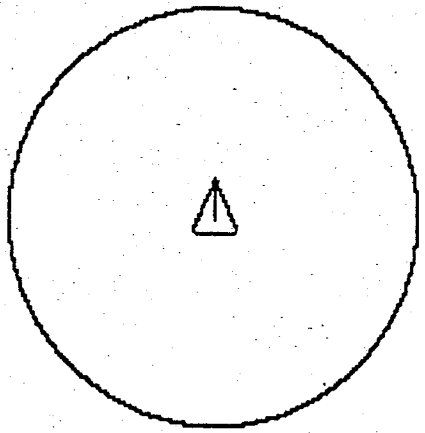
Figure 5.4

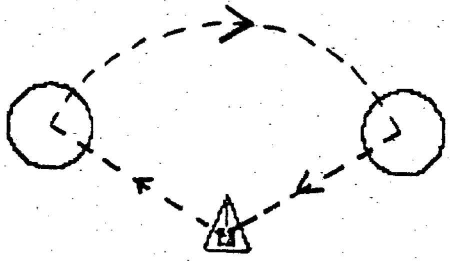
```
TO BEAR1
10 OUTSIDE1
20 INSIDE1
END
```

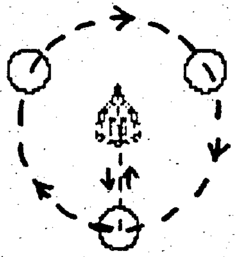
```
TO OUTSIDE1
10 CIRCLE 100
20 EARS1
END
```

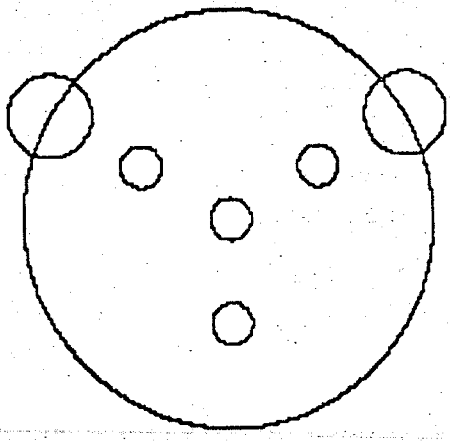
```
TO EARS1
10 LEFT 60
20 PENUP FORWARD 100 PENDOWN
30 CIRCLE 20 RIGHT 90
40 ARCR 100 120
50 CIRCLE 20
60 LEFT 90
70 PENUP BACK 100 PENDOWN
80 LEFT 60
END
```

```
TO INSIDE1
10 CIRCLE 10
20 PENUP BACK 50 PENDOWN
30 CIRCLE 10
40 LEFT 90
50 PENUP ARCR 50 120 PENDOWN
60 CIRCLE 10
70 ARCR 50 120 PENDOWN
80 CIRCLE 10
90 PENUP ARCR 50 120
100 RIGHT 90
110 FORWARD 50 PENDOWN
END
```


OUTSIDE1
Figure 5.5


EARS1
Figure 5.6


INSIDE1
Figure 5.7


BEAR1
Figure 5.8

5.1.2 Drawing the Bear Using Polar Coordinates:

Polar coordinates involve locating points using distances from a common center, and angles measured from a common (vertical) reference line. It would be a simple matter to draw the bear using polar coordinates making use of two commands, PLACE and REPLACE, which place the TURTLE with reference to the center of a circle and return it to the center.

```
TO PLACE :RADIUS :ANGLE
10 PENUP
20 RIGHT :ANGLE
30 FORWARD :RADIUS
40 PENDOWN
END
```

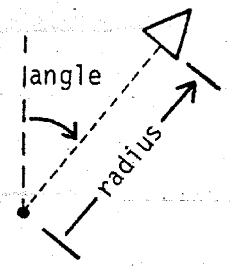


Figure 5.9

and

```
TO REPLACE :RADIUS :ANGLE
10 PENUP
20 BACK :RADIUS
30 LEFT :ANGLE
40 PENDOWN
END
```

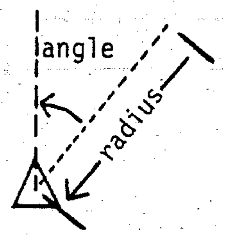
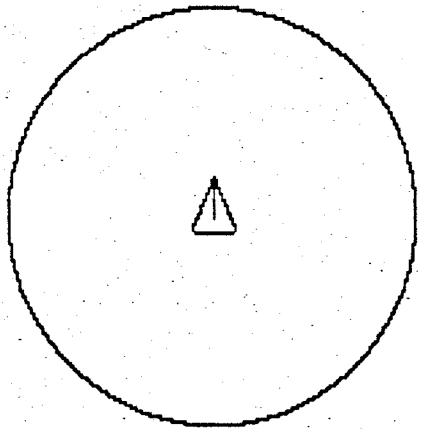
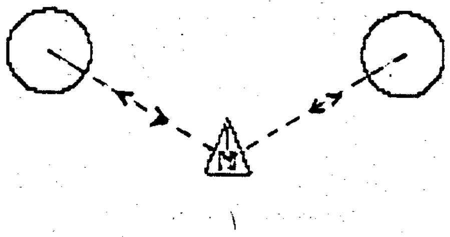
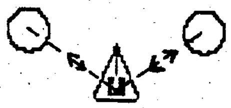


Figure 5.10

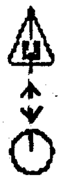
```
TO BEAR2            TO OUTSIDE2         TO INSIDE
10 OUTSIDE2         10 CIRCLE 100       10 EYES2
20 INSIDE2          20 EARS2            20 NOSE2
END                 END                 30 MOUTH2
                                        END

TO NOSE2
10 CIRCLE 10
END

TO EARS2                      TO EYES2
10 PLACE 100 (-60)            10 PLACE 50 (-60)
20 CIRCLE 20                  20 CIRCLE 10
30 REPLACE 100 (-60)          30 REPLACE 50 (-60)
40 PLACE 100 60               40 PLACE 50 60
50 CIRCLE 20                  50 CIRCLE 10
60 REPLACE 100 60             60 REPLACE 50 60
END                           END

TO MOUTH2
10 PLACE 50 180
20 CIRCLE 10
30 REPLACE 50 180
END
```

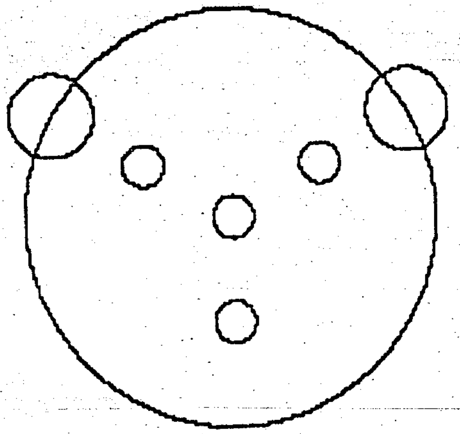

CIRCLE 100
Figure 5.11


EARS2
Figure 5.12


EYES2
Figure 5.13

NOSE2
Figure 5.14


MOUTH2
Figure 5.15


BEAR2
Figure 5.16

5.1.3 Drawing the Bear Using Cartesian Coordinates

The use of Cartesian coordinates involves locating points according to their x-and-y-coordinates. Because of the circular symmetry of the bear's head, cartesian coordinates would be a less likely choice. If cartesian coordinates were used, the x-and-y-coordinates of the eyes and ears would most likely be determined by trial and error, rather than by simple angular relationships. The drawing of a bear, making use of SETXY, might look slightly different as a *result.*

```
TO BEAR3
10 OUTSIDE3
20 INSIDE3
END
```

```
TO OUTSIDE3
10 CIRCLE 100
20 EARS3
END
```

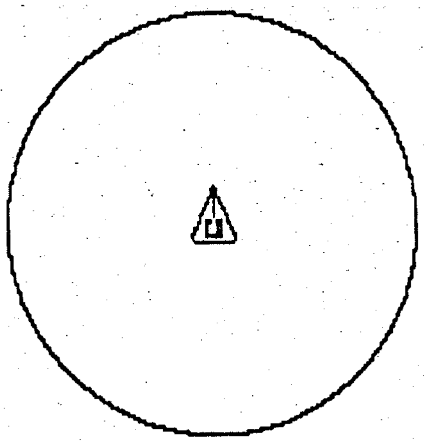
```
TO INSIDE3
10 EYES3
20 NOSE3
30 MOUTH3
END
```

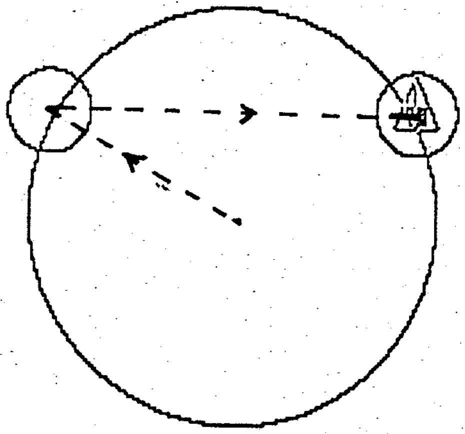
```
TO EARS3
5 PENUP
10 SETXY (-90) 50
15 PENDOWN
20 CIRCLE 20
25 PENUP
30 SETXY 90 50
35 PENDOWN
40 CIRCLE 20
END
```

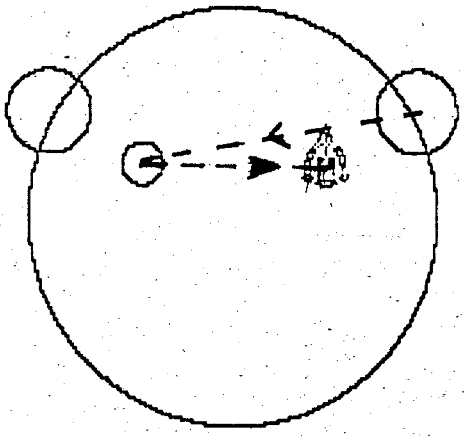
```
TO EYES3
5 PENUP
10 SETXY (-45) 25
PENDOWN
20 CIRCLE 10
25 PENUP
30 SETXY 45 25
35 PENDOWN
40 CIRCLE 10
END
```

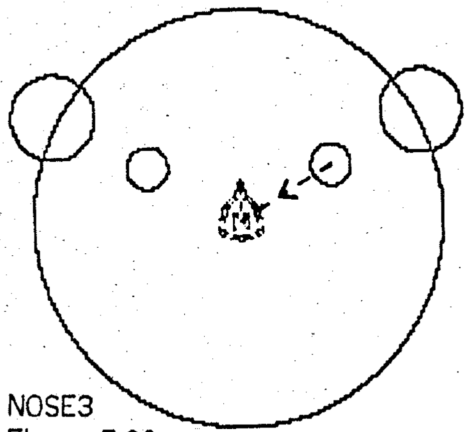
```
TO NOSE3
5 PENUP
10 HOME
20 CIRCLE 10
END
```

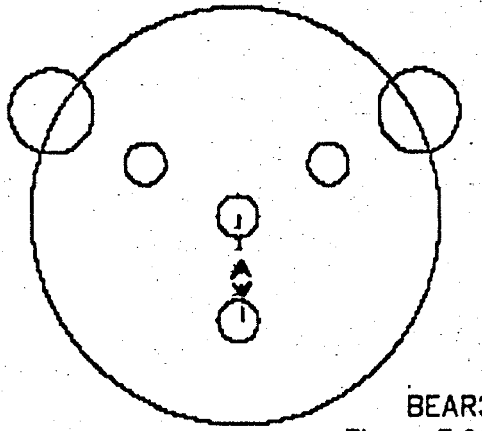
```
TO MOUTH3
5 PENUP
10 SETXY 0 (-25)
15 PENDOWN
20 CIRCLE 10
25 PENUP
30 HOME
END
```


CIRCLE 100
Figure 5.17


EARS3
Figure 5.18


EYES3
Figure 5.19


NOSE3
Figure 5.20


BEAR3
Figure 5.21

BEHAVIORS OBSERVED:

5.2 Student Use of Domain Specific Coordinate Systems.

Sometimes in the course of a particular project, a student will develop a way of structuring the geometry, to aid in solving the problem. The student is not usually aware that s/he is using a "coordinate system," but merely feels they have figured out a kind of "trick" to help with one aspect of a problem.

EXAMPLES:

5.2.1 Kevin's Intrinsic "Arc Coordinates"

Kevin's major project was to draw a picture of a "turtle" on the graphics display screen. He was shown an arc procedure which allowed both radius and angle as inputs. The circle procedure that he used to draw the outline of the "turtle's" shell made use of 10 degree steps, and had a radius of 90. At the end of each step, a tiny dot appeared on the screen. Kevin could maneuver the TURTLE along the "turtle's" shell, by counting dots, and using the appropriate input to ARCR. For example, after drawing the "turtle's" head, Kevin wanted to place four feet and a tail on the "turtle's" shell. He simpy oriented the TURTLE along the shell, counted three dots between where the TURTLE was, and where he wanted to start drawing the first foot, and used the command ARCR 90 30, to move the TURTLE there. The first input, 90, was the radius of the circle, and remained constant. The second input was the angle. Since Kevin knew that three dots represented three 10 degree steps, he used the second input, 30, to move that far along the circle.

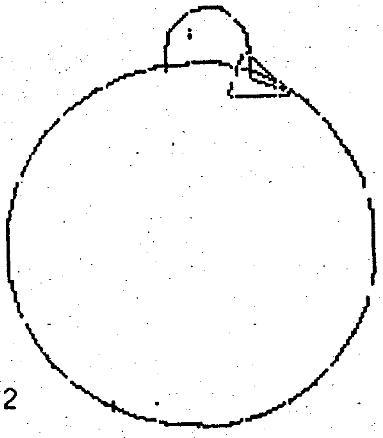


Figure 5.22

SHELL HEAD LT 70

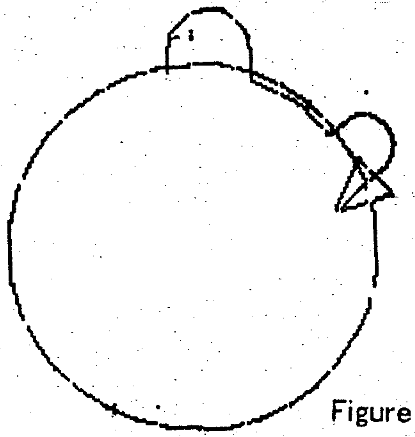


Figure 5.23

ARCR 90 30
LT 90
FOOT

5.2.2 The "Hat" on Donald's "Head"

Donald's extended project was to have the computer draw a complex head, complete with beard, hair, hat and flower: At one point, he had to figure out where to locate the "hat" on the head, and how large to make it, so that it would appear symmetrical. Since Donald had difficulty estimating sizes, and solving problems visually (see a summary of Donald's work in section chapter 5 of this report, and a full profile in Chapter 6 of Part III), he often resorted to analytical strategies to help solve his problem. In this case he made use of the "hairs" which he had already drawn as part of his head. By counting hairs, from each side of the head, he was able to determine when the hat was sufficiently symmetrical to satisfy him. A drawing in his notebook was used as part of the process.

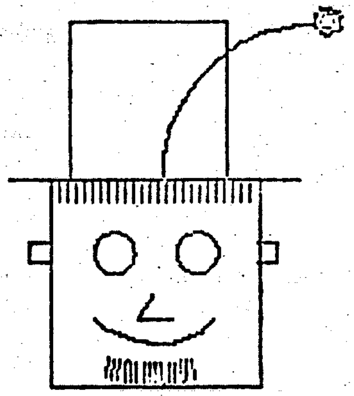


Figure 5.24

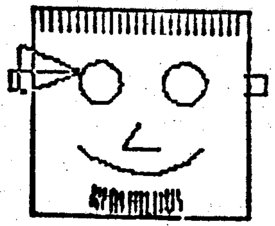


Figure 5.25

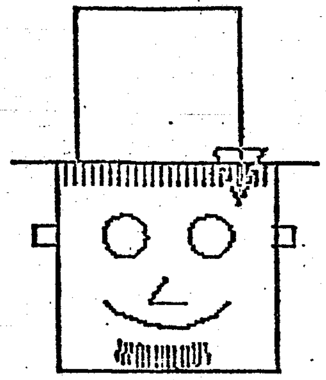


Figure 5.26

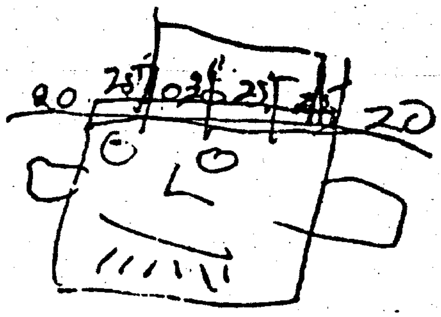


Figure 5.27

5.3 Student Use of Cartesian Coordinates

The cartesian coordinate system is the most commonly used global geometric structure in most courses on analytic geometry, trigonometry, calculus, etc. In LOGO activities, cartesian coordinates have a more limited applicability, because of the compelling immediacy of local Turtle Coordinates as a way of solving the simplest geometric problems which arise in Turtle Geometry. The cartesian description of a circle, for example, $x^2 + y^2 = r^2$, requires the user to understand algebra, the use of exponents, and possibly the pythagorean theorem. It specifies the radius of the circle, and gives a formula enabling the user to find any number of points on the circle, relative to the center. The LOGO description, FORWARD 1, RIGHT 1, ... on the other hand, tells exactly how to draw the circle, without specifying how large it is, or where the center is, or where any particular point is located.

LOGO is designed to simplify the use of cartesian coordinates for specific applications. Every point on the display screen has an x-and-y-coordinate. The LOGO comands XCOR and YCOR output the x-and-y-coordinates of wherever the TURTLE happens to be. SETXY is a command requiring two inputs, will move the TURTLE directly to the point on the screen which has the x-and-y-coordinates given. To totally specify a TURTLE position, it's orientation or "heading" must also be given. The LOGO commands HEADING and SETHEADING are analagous to XCOR, YCOR and SETXY.

The LOGO command HERE outputs a list of three numbers, the x, y and heading coordinates of the TURTLE. The command, HERE, allows a user to give a name to a point on the screen, by moving the TURTLE to that point and using the MAKE comand, as in MAKE "POINT1 HERE. To return the TURTLE to that point later the user types SETTURTLE :POINT1. Points on the screen can also be named without moving the TURTLE as in the command MAKE "POINT2 [100 100 0]. "POINT2 then represents a TURTLE position whose x-coordinate and y-coordinate are both 100, and whose orientation is straight up on the screen.

Many students who do not use cartesian coordinates explicitly do make use of a sort of implicit grid system, in moving the TURTLE from one place to another on the display screen. Students are often introduced to the coordinates in a situation in which the naming of a specific point, and the later return of the TURTLE to that point are needed for a particular project. Cartesian coordinates may also be encountered in a situation in which the user wants to know whether the TURTLE is inside or outside of a particular region of the screen (as in an animated "race" or a "target game.") Another *region of the screen (as in an animated "race" or a "target game.") Another use of cartesian coordinates occurs when the student wants the TURTLE to draw a grid on the screen -- for example, to draw a tictatoe board.*

EXAMPLES:

5.3.1 The Use of an "Implicit Grid" In Moving the TURTLE

Many students find it easier to move the TURTLE around the display screeen in horizontal and vertical "steps", than to accurately estimate both the distance and direction of a point to which the TURTLE is to be moved. To move the TURTLE to one corner of the screen, for example, many students spontaneously learn to use this set of commands: RIGHT 90, FORWARD 200, LEFT 90, FORWARD 200 (rather than RIGHT 45, FORWARD 282, which would move the TURTLE directly to the same spot.)

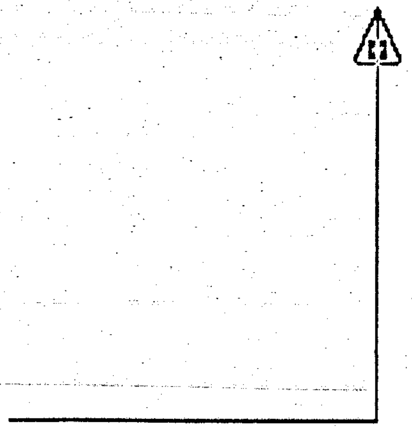


Figure 5.28

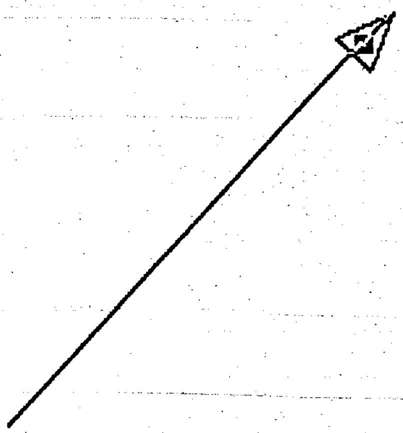


Figure 5.29

Albert used this approach for almost all TURTLE moves (other than drawing curves or triangles). Using a series of alternating FORWARD and RIGHT 90 or LEFT 90 commands, he could move the TURTLE anywhere on the screen. His inputs to FORWARD usually started with multiples of 50 or 100, which "fit" nicely into the total screen size of 400 TURTLE steps.

A student who uses this approach is not using "coordinates", but is in a sense "drawing a mental grid," creating a structure in the mind which will later make a

5.3.2 Naming Points Without Coordinates

There are a few different ways that a student might make use of the capability to name point and set the TURTLE to that point in the context of a particular project, a target game. The idea is that a target is drawn on the screen. The TURTLE is then placed at the center of the screen (or another point), aimed at the target, using RIGHT and LEFT commands, and then "shot" at the target. The computer has to tell the player if the target has been hit.

First, the student might use point naming to draw a state transparent circle that starts from the center of the circle:

```
TO TARGET
10 MAKE "P1 HERE
20 PENUP FORWARD 40
30 PENDOWN RIGHT 90
40 RCIRCLE 40
50 PENUP SETTURTLE :P1
END
```

The student must now draw the target at a particular point on the screen. The TURTLE is first driven to the desired point, and the point given a name, say "P2. To set up the target for a game, the student can make a procedure.

```
TO SETUP
10 PENUP SETTURTLE :P2
20 TARGET
30 HOME
END
```

In a game, the TURTLE is aimed at the target, and fired by a procedure called SHOOT, which checks to see whether the TURTLE has hit the target.

```
TO SHOOT :D
10 MAKE "P3 HERE
20 FORWARD :D
30 IF DISTANCE :P2<40 PRINT [YOU'VE HIT THE TARGET!] STOP
40 SETTURTLE :P3 PRINT [YOU MISSED.  TRY ANOTHER SHOT]
END
```

(DISTANCE :POINT is a LOGO procedure which calculates the distance between the

TURTLE and any point, and which could be given to the student as a "primitive")

5.3.3 Naming Points With Coordinates.

Involvement in a project such as the target game described above leads quite naturally to the desire to select many different points for the location of the target, and for the initial position of the TURTLE before shooting. After gaining some experience with points by using MAKE "POINT HERE, the student can acquire some experience with the coordinates themselves by typing PRINT HERE. S/he will then see that the computer keeps track of the TURTLE's location by means of a list of three numbers corresponding to its x and y coordinates and its heading. Instead of moving the TURTLE to a particular point, "P2, in order to define it, the student can define it by giving its coordinates, MAKE "P2 [50 50 45]; or by having the computer choose coordinates for the point in a random way:

```
MAKE "P2 (SENTENCE 10*RANDOM 10*RANDOM 20*RANDOM)
```

5.3.4 Constructing a Grid Using Cartesian Coordinates

A student could easily construct a simple grid, a TicTacToe board, for example, using x and y coordinates, and the command SETXY:

```
TO TICTACTOE
10 PENUP SETXY (-75) 150
20 PENDOWN SETXY (-75) (-150)
30 PENUP SETXY 75 150
40 PENDOWN SETXY 75 (-150)
50 PENUP SETXY 150 75
60 PENDOWN SETXY (-150) 75
70 PENUP SETXY 150 (-75)
80 PENDOWN SETXY (-150) (-75)
90 PENUP HOME
END
```

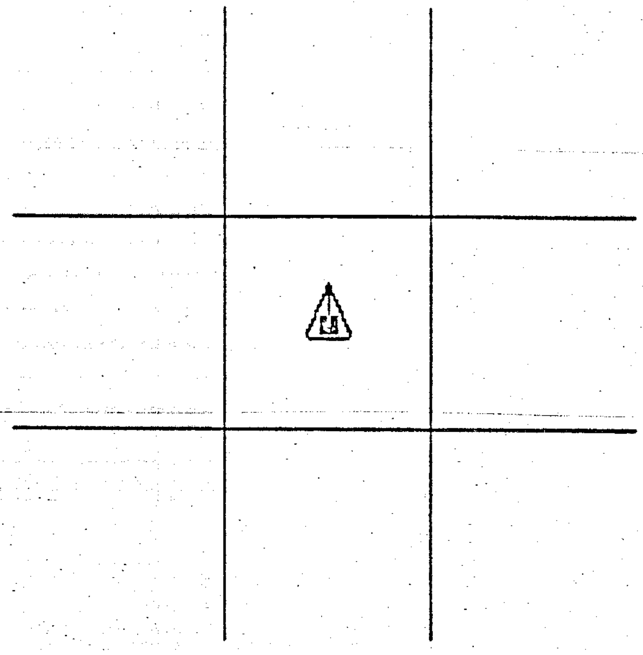


Figure 5.30

The same result could be achieved in a straightforward manner using FORWARD, BACK, RIGHT and LEFT, but would require many more steps.

5.4 The Use of Polar Coordinates

In a polar coordinate system any point is located by specifying its distance from a fixed origin and its angle of orientation with respect to a fixed line through the origin. To use the TURTLE with a polar coordinate system, one can set the orientation of the TURTLE to the desired heading with respect to the vertical, and move it forward the desired distance. Polar coordinates can be useful in LOGO projects in which the TURTLE always returns to a fixed point, but changes its orientation, to carry out a sequence of actions.

Students who make use of this kind of approach as part of a LOGO project, do not consider it a formal coordinate system. For them, a structure based on the TURTLE's orientation can be thought of as another form of "intrinsic" or "implicit" coordinate system, a "trick" developed for its usefulness in a particular case. Students who make use of this particular approach develop an intuitive understanding that may later help them understand a more formal use of polar coordinates.

5.4.1 Creating a "Beard" for Donald's Head

Donald used a procedure like the following to draw a beard for his head project.

```
TO BEARD
10 RIGHT 15
20 REPEAT [STRING] 15
END
```

```
TO STRING
10 PENUP
20 FORWARD 80
30 PENDOWN
40 FORWARD 10
50 PENUP
60 BACK 90
70 LEFT 2
END
```

The fixed point from which each "string" emanated was located near the top of the head, by trial and error.

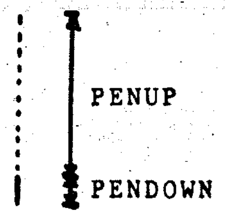


Figure 5.31

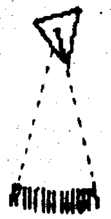


Figure 5.32

5.4.2 Angular Coordinates With Spinning Designs

Many students create spinning designs which end with a HOME command, to complete the design and "undo" the spin. Karl called his design, NO. A "flower-like" design he made by repeating NO, was called XX78055.

```
TO NO
10 SPIN
20 FORWARD
30 SPIN
40 FORWARD
50 SPIN
60 FORWARD
70 HOME
END
```

```
TO XX78055
10 NO
20 RIGHT 10
30 NO
40 RIGHT 20
50 NO
60 RIGHT 30
etc.
```

If NO had not contained a HOME command, Karl's XX78055 procedure would have been of the shape, rotation, shape, rotation... type of design, and would not have involved polar coordinates. Since NO ends with a HOME comand, the TURTLE ends up at the origin, in a vertical orientation every time NO is executed. Therefore, the TURTLE's rotation had to be increased before each successive NO, in order to produce the design Karl wanted.

5.4.3 Polar Coordinates in a Symmetrical "Face"

The "bear's head" described in Section 5.1, is an example of a project which lends itself readily to Polar coordinates.

6. Theorems and Heuristics: Towards Formal Mathematics

In formal mathematics classes, students are generally taught that a "theorem" is a mathematical proposition, already known to be "true" that has to be "proven" by a series of logical statements based on axioms and previously proven theorems. The task of a working mathematician, on the other hand, is to formulate propositions, to try to discover the predictable regularities of a particular "world"; then to test, extend and revise or discard such propositions based on the tests; and only finally, after a theorem and its usefulness is quite well established, to attempt to prove or disprove it in a logical sense.

*Students learning LOGO have the opportunity to function like a mathematician, rather than like a student in a mathematics class.**

While solving their own problems, students begin to discover some of the regularities of the mathematical world in which they are functioning. Such regularities may be used by the students as "heuristics" -- strategies or "rules of thumb" that may be helpful in problem solving. Heuristics used by LOGO students include general approaches to problem solving, such as breaking a large problem in small easily solved parts, specific techniques such as "playing TURTLE" to understand which way to move the TURTLE in a specific instance or to plan a shape like a circle, and specific geometric design ideas such as "keep repeating a shape and an interesting design will occur."

*When a student who has discovered and used a heuristic such as "repeat a shape" begins to be aware of certain regularities that occur whenever that heuristic is applied, s/he is on the track of a "theorem". For instance, the student may realize that "all repeated designs eventually close;" that is, the TURTLE begins to retrace its path, if a sequence of steps repeated often enough. Some students become extremely interested in verifying such a proposition. Although the formal "proof" of such a proposition is beyond the interest or ability of an elementary school student, it might be an interesting math project for a high school or college student studying LOGO.***

****

*See "Teaching Children to be Mathematicians vs. Teaching Children About Mathematics," S. Papert. LOGO Memo #4, MIT Artificial Intelligence Laboratory.

**See "Turtle Geometry" by Abelson and diSessa, to be published by MIT Press, 1980.

If the student shows further interest in this idea, s/he may begin to compare specific shapes with the number of repeats required before they close. Such an investigation could lead to another theorem: "any time the TURTLE has completed a series of steps and returned to its exact starting position and heading, it has rotated through 360 degrees, or an integer multiple of 360 degrees." This is such a common and useful LOGO theorem that it has been given a name, the "Total Turtle Trip" theorem, or "TTT" for short. The TTT applies to situations that do not involve repetition, but it is most commonly discovered, and used in cases that do involve repetition of a series of fixed steps.

In this section, we describe student behaviors involving theorems and heuristics that arise in the context of Turtle Geometry. Many of these relate in some way to the properties of the rotational group, described in Section 3. Theorems and heuristics we will consider in some detail include repetition; the Total Turtle Trip theorem and a special case, the POLY theorem; the concept of similarity; and the use of symmetry.

BEHAVIORS OBSERVED:

6.1 The Use of Repetition

Once a student has written a first procedure to draw a shape on the display screen, a teacher usually suggests that the student repeat the design. Students quickly adopt this idea, and use it to create many fascinating and unexpected designs.

EXAMPLES:

6.1.1 Repeating a square.

A very common early LOGO project is to draw a "box". Typically, a student draws a box in seven steps:

```
TO BOX
1 FORWARD 50
2 RIGHT 90
3 FORWARD 50
4 RIGHT 90
5 FORWARD 50
6 RIGHT 90
7 FORWARD 50
END
```

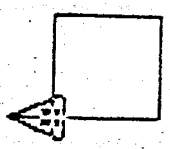


Figure 6.1

Repeating BOX, produces this "surprising" result:

```
BOX
BOX
```

```
BOX
BOX
BOX
```

```
BOX
BOX
BOX
BOX
```

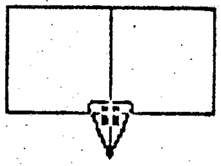


Figure 6.2

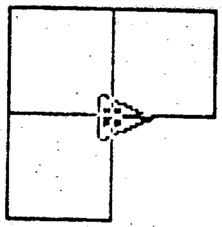


Figure 6.3

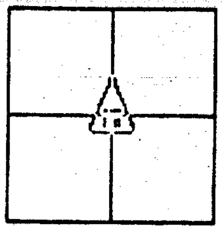
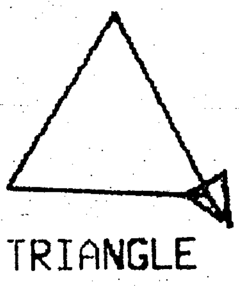


Figure 6.4

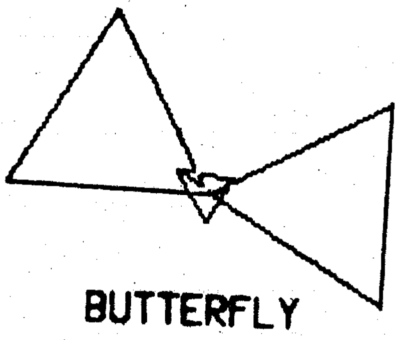
A fifth repeat of BOX retraces the first BOX, starting the cycle again.

6.1.2 Repeating Other Shapes.

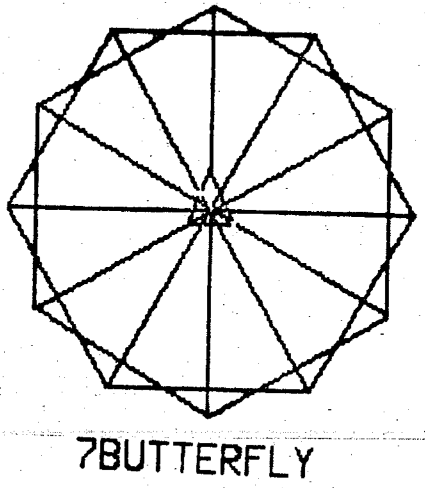
Kathy enjoyed repeating shapes. Her TRIANGLE, repeated twice, became a BUTTERFLY. BUTTERFLY, repeated until it closed, became 7BUTTERFLY.



```
TO TRIANGLE
1 LEFT 90
2 FORWARD 100
3 RIGHT 120
4 FORWARD 100
5 RIGHT 120
6 FORWARD 100
END
```



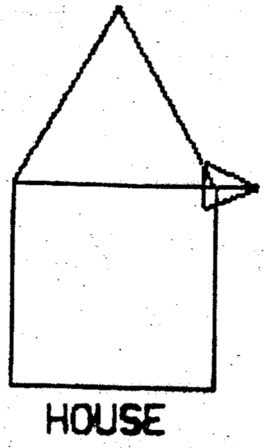
```
TO BUTTERFLY
1 TRIANGLE
2 TRIANGLE
END
```



```
TO 7BUTTERFLY
1 BUTTERFLY
2 BUTTERFLY
3 BUTTERFLY
4 BUTTERFLY
5 BUTTERFLY
6 BUTTERFLY
END
```

TRIANGLE BUTTERFLY 7BUTTERFLY
Figure 6.5

When Kathy made a HOUSE procedure, using her TRIANGLE and BOX procedures, she promptly repeated it four times to make HOUSE4.



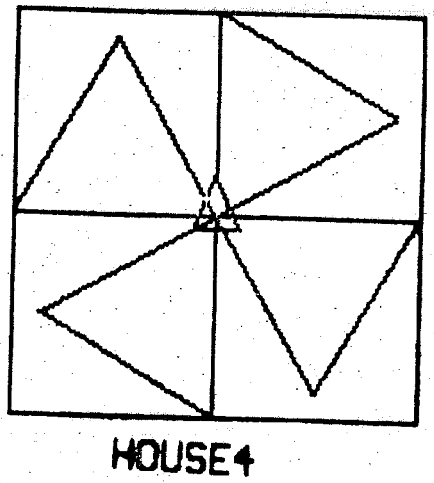


Figure 6.6

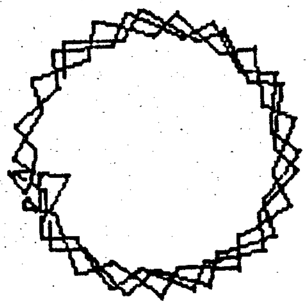
6.1.3 Repeating a "Miscellaneous" Shape

Ray was one of many students who enjoyed "seeing what would happen" if he made up a miscellaneous set of commands, gave those commands a name and repeated them an arbitrary number of times. His procedures, SAM, TIM, and JOE are examples of this.

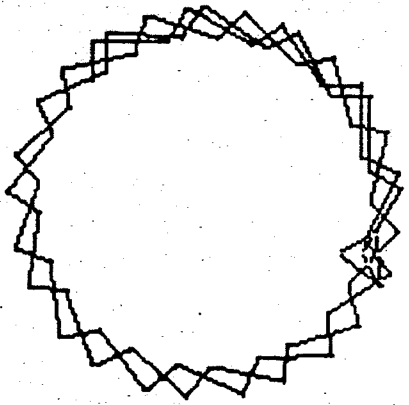
```
TO SAM
1 FORWARD 17
2 RIGHT 90
3 FORWARD 29
4 LEFT 56
END
```

```
TO TIM
1 FORWARD
2 RIGHT 90
3 FORWARD 36
4 LEFT 61
END
```

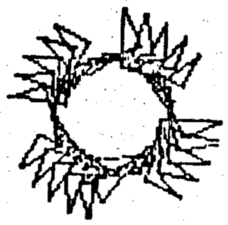
```
TO JOE
1 SAM
2 LEFT 150
3 TIM
END
```



REPEAT [SAM]30



REPEAT [TIM]30



REPEAT [JOE]30

Figure 6.7

6.1.4 Repeating a Shape and a Rotation

Many students develop a process of repeating a shape and a rotation. If a procedure is "state transparent" -- that is, the TURTLE returns to its original position and heading when it completes the figure -- then repeating it causes it to retrace itself. One way to make a more complex design with such a shape is to rotate the TURTLE a fixed amount after each repetition of the shape. Monica used this approach with many of her basic shapes. Here are some of the designs Monica made, using her state-transparent TRI procedure, with different rotations.

```
TO TRI4
1 TRI
2 LEFT 90
3 TRI
4 LEFT 90
5 TRI
6 LEFT 90
7 TRI
END
```

```
TO TRI42
1 LEFT 40
2 TRI4
END
```

```
TOTRI442
1 TRI 4
2 TRI42
END
```

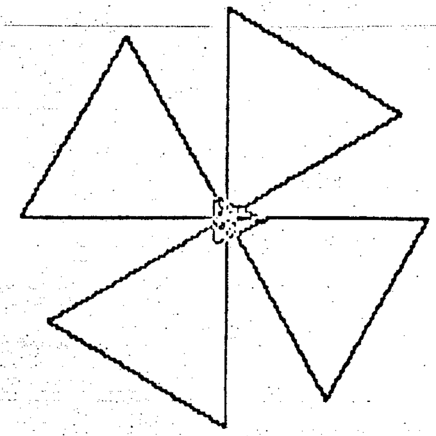


Figure 6.8

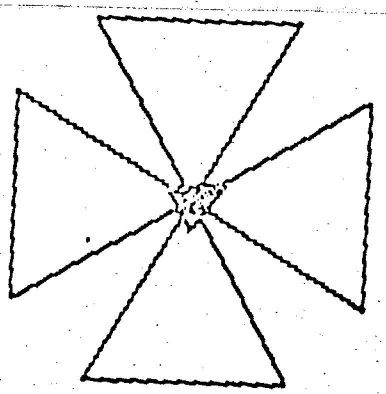


Figure 6.9

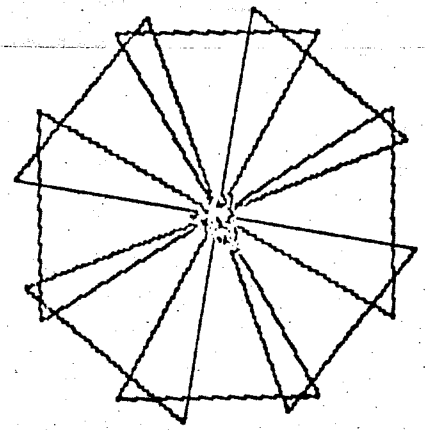
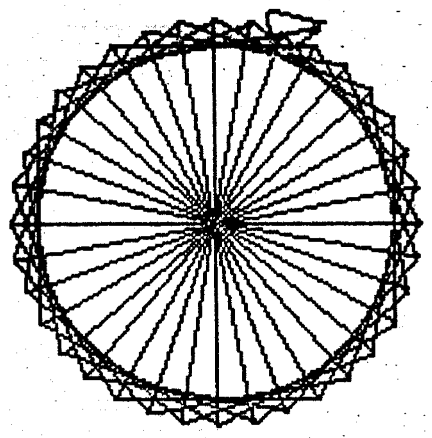


Figure 6.10

```
TO FAN
1 TRI
2 RIGHT 10
3 FAN
END
```



'FAN

Figure 6.11

6.2 Determining the Number of Repeats Needed to "Close" a Particular Shape

Some students quickly realize that when repeating "shape, rotation," certain "special angles" produce fairly simple closed figures, while other angles produce complex figures which take a long time to close, or which "fill up the screen" before closing. Focussing on the particular angles which make the simpler shapes can be an important step, leading to understanding the significance of 360 degrees and to the TTT theorem. A particular case of this is the process of making the TURTLE draw a circle by repeating FORWARD something, RIGHT something a certain number of times.

EXAMPLES

6.2.1 Using "Special Angles" Without Analysis

Deborah limited most of her inputs to turtle commands to the numbers 30, 60 and 90. Therefore, when she began to use the idea of rotating a shape, it was quite natural for her to use RIGHT 60 as the rotation to produce her FLOWER:

```
TO FLOWER
1 CIRCLE
2 RIGHT 60
3 CIRCLE
4 RIGHT 60
5 CIRCLE
6 RIGHT 60
7 CIRCLE
8 RIGHT 60
9 CIRCLE
10 RIGHT 60
11 CIRCLE
END
```

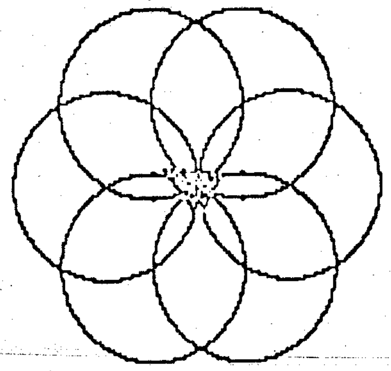


Figure 6.12

Kathy made a star using a rotation of 45 degrees. Her procedure BUS4, achieved by repetition, was the basis of her STAR procedure.

```
TO BUS
1 SQ 40
2 LEFT 90
3 SQ 50
END
```

```
TO 4BUS
1 BUS
2 BUS
3 BUS
4 BUS
END
```

```
TO STAR
1 4BUS
2 RIGHT 45
3 4BUS
END
```

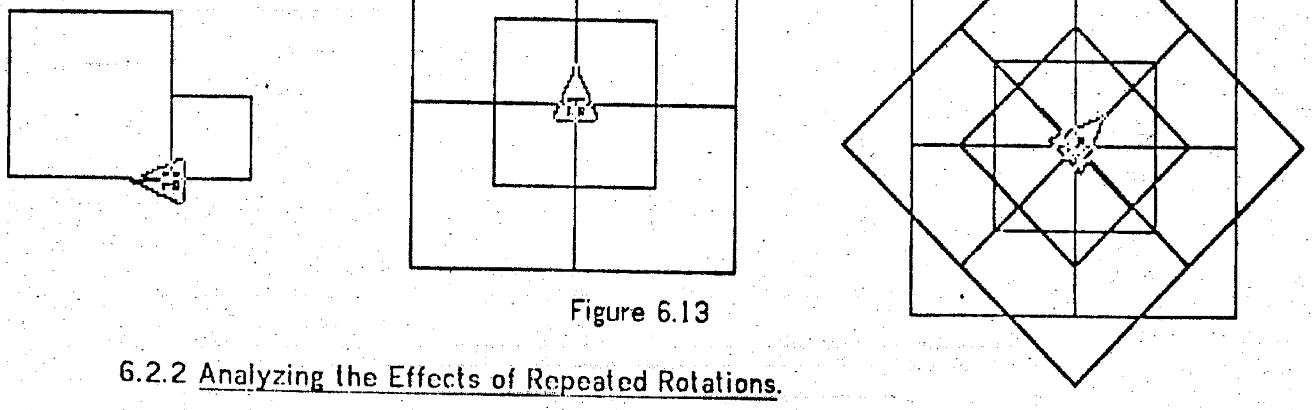


Figure 6.13

6.2.2 Analyzing the Effects of Repeated Rotations.

Monica, enjoyed rotating shapes, but had very little idea of the relationship between the angle she chose, and the resulting shape. Her teacher suggested using the angle of rotation as a variable, and taking notes on the effects of different rotations. She was helped to write the procedure WISHWOW to rotate her procedure, WOW, a collection of nested squares.

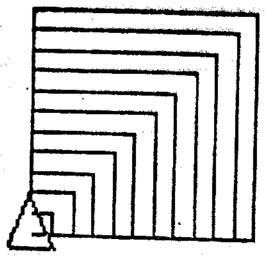
```
TO WISHWOW :ANGLE
10 WOW
20 RIGHT :ANGLE
30 IF HEADING = 0 STOP
40 GO 10
END
```

Monica's notes show that although she was beginning to notice regularities and to connect the number of repetitions with the rotation used, she did not have a systematic understanding of these effects.

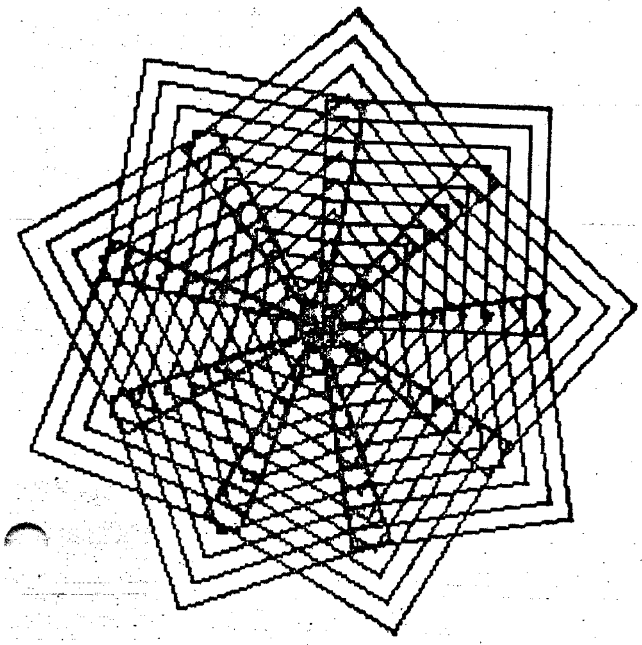
"WISHWOW 160 looked the same as WISHWOW 40. It had thin cones and there were 9 of them.

WISHWOW 165 had thin webbed cones and you couldn't really see them that good. WISHWOW 190 had cones but they looked like they didn't close up. And it was fatter than other ones. It had more squares and cones. The cones were thin. And close together.

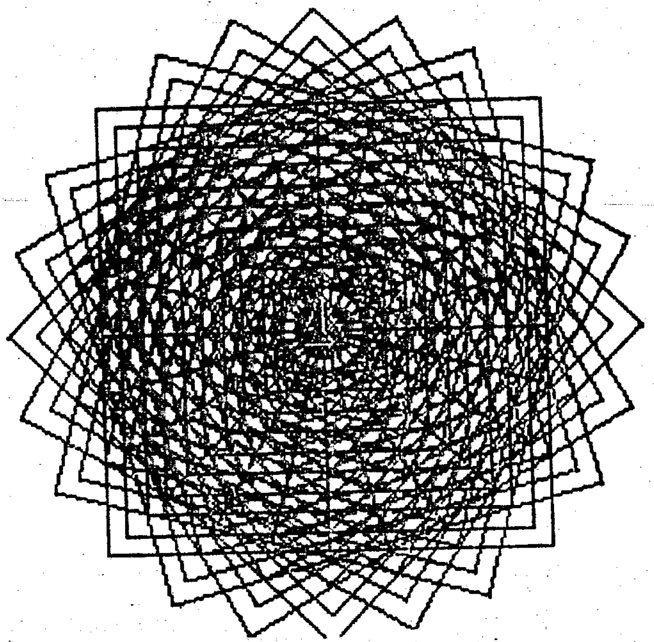
WISHWOW 45, WISHWOW 90. These 2 look almost the same but WISHWOW 45 looks like it goes twice around instead of once. And the cone shaped things on the sides are bigger than the WISHWOW 90 ones."



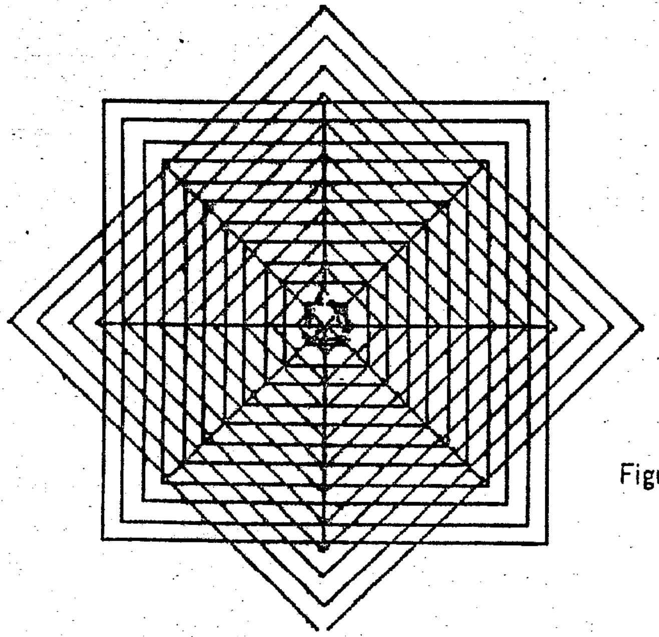
WOW



WISHWOW 160

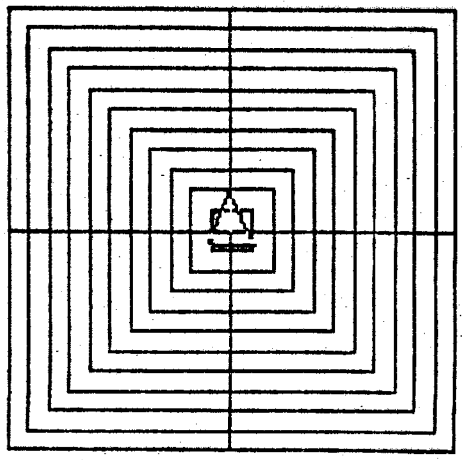


WISHWOW 165



WISHWOW 45

Figure 6.14



WISHWOW 90

6.3 "Discovery" and Use of the "Total Turtle Trip" Theorem

Most students make use of the "Total Turtle Trip" theorem in the context of a concrete project rather than in complete generality. Once the student understands the idea in a general way s/he can apply it to the solution of a different specific problem.

EXAMPLES:

6.3.1 Constructing a LOGO Circle Using Repetition

Darlene had a systematic approach to problems that she encountered. She was always interested in exactly how many repeats it took for a shape to "close". When she drew a "circle", using the REPEAT command, she experimentally determined exactly how many repeats were required. With careful observation she found that the command REPEAT [RIGHT 2 FORWARD 3] 180 would draw a closed circle with no overlap. Once she realized that the TURTLE had turned exactly 360 degrees as it drew the circle she was able to draw smaller circles by using the commands:

REPEAT [RIGHT 3 FORWARD 2] 120 and REPEAT [RIGHT 4 FORWARD 2] 90

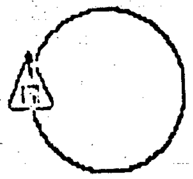


Figure 6.15a

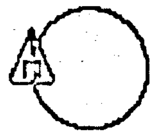


Figure 6.15b

6.3.2 Constructing an Equilateral Triangle, and Other Regular Polygons.

Once a student has drawn a square using the TURTLE, it is natural to attempt the construction of a triangle. There are many ways to construct a triangle using trial and error approaches. Some of these have been discussed above. Another approach is to make use of information derived from the process of constructing a square and apply it to constructing a triangle. The reasoning involved is far from trivial, and it is often the case that a student and a teacher will work together on this process, rather than a student figuring it all out independently.

The reasoning goes like this. A square is constructed by repeating the same "thing" four times. The "thing" that is repeated is the pair of steps, FORWARD something, RIGHT 90. In doing so, the TURTLE has rotated a total *of 360 degrees. The student also knows that a rotation of 360 degrees will turn the TURTLE all the way around. In order to draw a triangle, the TURTLE will also make a shape in which it turns all the way around, this time in three steps. The amount of rotation needed for each step is thus 360/3 or 120 degrees. The student can now draw a triangle by repeating FORWARD something, RIGHT 120 three times.*

At this point, the student might explore the generality of this approach by trying to construct a 6, 8 or 10 sided regular polygon. In each case, the process as described above will lead to a successful result. (The process breaks down if the student wants to construct a 7 sided polygon, however, since 360 is not evenly divisible by 7.)

6.3.3 Application of the TTT to Solving a Particular Problem

A common student project is to draw a "leaf" for a plant, using quarter arcs:

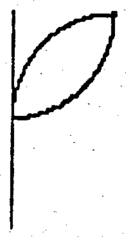


Figure 6.16

The student knows that the RARC and LARC procedures cause the TURTLE to rotate a total of 90 degrees. In order to draw a "leaf" and return to the point at which it started (which is useful in getting the TURTLE back on the plant's "stem") the student reasons that the TURTLE must draw an arc, turn some amount, draw another arc and turn the same amount. Since the two arcs cause the TURTLE to rotate a total of 180 degrees, the TURTLE needs to turn a total of (360-180), or 180 degrees at both ends of the leaf. Thus the TURTLE must turn 90 degrees at each end of the leaf. This procedure will draw the leaf:

```
TO LEAF
10 RARC 100
20 RIGHT 90
30 RARC 100
40 RIGHT 90
END
```

6.4 Calculating the Number of Points in a Star or Polygon, Using a POLY Procedure.

Another way of approaching the "total turtle trip" is by analysing the results of using different inputs for POLY. (See section 2) A student might be asked by a teacher to keep a chart showing the angle input for POLY, the number of points in the resulting star or polygon, and the calculated total rotation turned by the TURTLE in completing the design. A student who has carefully carried out a number of POLY experiments will find that for any polygon, the turtle will turn through total rotation of 360 degrees, and that for any star, the TURTLE will turn through a total rotation of a multiple of 360 degrees. (A five pointed star, produced with an angle input of 144 degrees turns the TURTLE a total of 720 degrees. An eight pointed star, produced with an angle of 135 degrees turns the TURTLE through a total of 1080 degrees, etc.) This is one particular form of the Total Turtle Trip Theorem.

6.5 The Use of Similarity

There are a number of ways for students to encounter and make use of a "LOGO Similarity Theorem". A proportional change in all the FORWARD and BACK steps in a sequence of TURTLE commands, while holding the angles constant, will change the size, but maintain the shape of the figure drawn by those commands. While few students come to understand this principle in its full generality, there are many ways in which students encounter it in simpler forms and use it in their LOGO projects. The desire to create similar designs often provides students with their first use of variables, as they try to create "different sized squares," for example.

6.5.1 EXAMPLES:

6.5.1 Similarity In Regular Shapes.

Almost every student encounters regular similar shapes when they work with RCIRCLE and LCIRCLE commands, squares of different sizes, and the shapes made by a POLY procedure. Since all the sides of a regular shape are the same, making the length of that side a variable, rather than a fixed distance leads immediately to similar figures.

Laura's design, AROUND was built of randomly selected circles:

```
TO AROUND
10 LCIRCLE 90
20 LCIRCLE 58
30 LCIRCLE 48
40 LCIRCLE 20
50 LCIRCLE 10
60 LCIRCLE 96
70 LCIRCLE 50
80 LCIRCLE 33
90 LCIRCLE 66
END
```

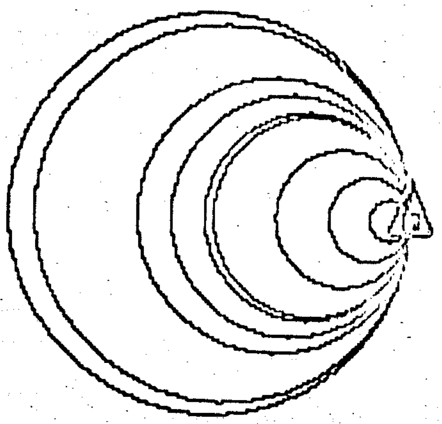


Figure 6.17

Monica's procedure, WOW, was created using a variable sized square procedure:

```
TO WOW
1 SQ 10
2 SQ 20
3 SQ 30
4 SQ 40
5 SQ 50
6 SQ 60
7 SQ 70
8 SQ 80
9 SQ 90
10 SQ 100
11 SQ 110
END
```

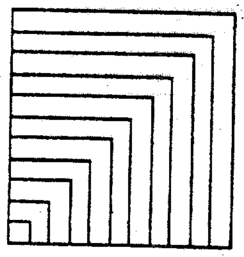


Figure 6.18

Kevin's TUNNEL, using a POLY procedure was a more general use of the same idea.

```
TO TUNNEL :SIZE
10 POLY :SIZE 45
20 IF :SIZE = 105 STOP
30 TUNNEL :SIZE + 5
END
```

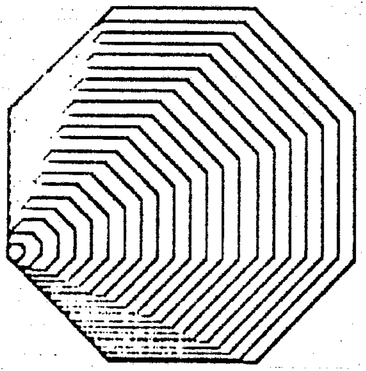


Figure 6.19

Dennis' nested triangle procedure, Q, made use of similar triangles in a very different way:

```
TO Q :SIZE
10 IF :SIZE = 10 STOP
20 THRI1 :SIZE
30 FORWARD :SIZE/2
40 RIGHT 60
50 Q :SIZE/2
END
```

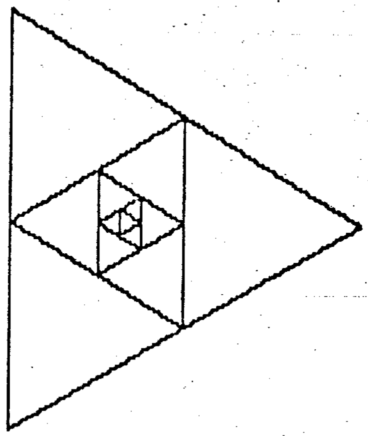


Figure 6.20

One of the most elaborate "regular figures" was Betsy's SUN design. After first creating the procedure with fixed sizes, Betsy made a new set of procedures which took a variable :SIZE as input:

```
TO SUN2 :SIZE
10 REPEAT [RAY2 :SIZE] 9
END

TO RAY2 :SIZE
10 BOX2 :SIZE
20 LEFT 180
30 RIGHT 20
END

TO BOX2 :SIZE
10 RARC :SIZE
20 LARC :SIZE
30 RARC :SIZE
40 LARC :SIZE
END
```

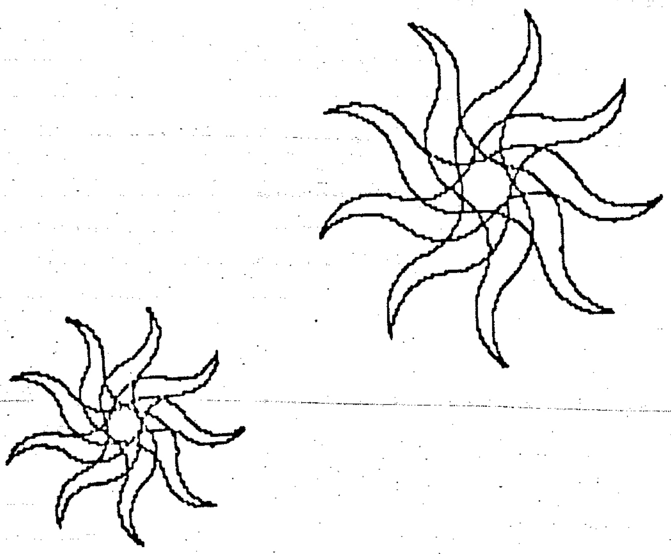


Figure 6.21

6.5.2 Similarity in Non-Regular Shapes.

The more general principal of similarity, in which all the sizes in a shape are multiplied by a constant factor, occurred less frequently in our trial classes. One noteworthy example was Kathy's pair of procedures WORM and WORMY:

```
TO WORM
1 RARC 30
2 LARC 30
3 RARC 30
4 LARC 30
5 RCIRCLE 10
END

TO WORMY
1 RARC 60
2 LARC 60
3 RARC 60
4 LARC 60
5 RCIRCLE 20
END
```

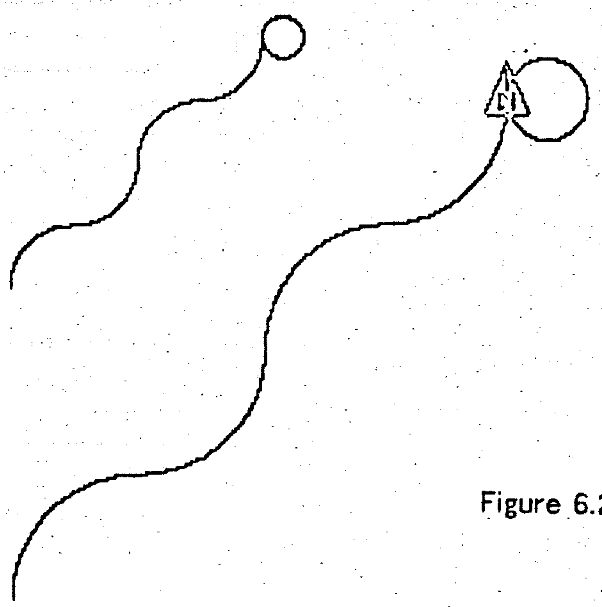


Figure 6.22

6.6 The Use of Symmetry

The idea of symmetry is one which most students encounter as part of their LOGO experience. A "LOGO symmetry theorem" might be stated as follows: "If all the right and left commands in a sequence of TURTLE commands are reversed, without changing any of the other commands in the sequence, the resulting design will be a mirror image of the original design." The reversing of RIGHT and LEFT, is one approach which students use to create symmetrical designs. Another is the use of an "implied axis of symmetry," usually a vertical line down the center of a design, in which both sides are identical but in which the symmetry is produced by "working across from one side" of the design rather than by starting from the middle and reversing RIGHT and LEFT commands. Some students make use of one of these approaches to symmetry in connection with a design that is not fully symmetrical, or in a design that is mostly symmetrical, but which has one or two dramatic asymmetries. Finally, many students carry out projects involving rotational symmetry, examples of which have been considered above.

EXAMPLES:

6.6.1 Symmetry by Right/Left Reversal

Symmetry by right/left reversal is usually preceded by the realization that a RIGHT turn can be eliminated by an equal LEFT turn, and reversed by a double LEFT turn. This involves the use of right and left as inverses of each other, and was discussed in Section 4 of this chapter. The most common example of designs which make use of right/left reversal are those made by using arc and circle primitives, RARC, LARC, RCIRCLE and LCIRCLE, since these immediately produce symmetrical designs.

Two simple examples of this type of symmetry were Deborah's procedure called EYES, drawn using RCIRCLE and LCIRCLE, and the NOSE procedure she used for her "rabbit" in which she used RARC and LARC as inverses of each other.

```
TO EYES
1 RCIRCLE 90
2 LCIRCLE 90
3 RCIRCLE 40
4 LCIRCLE 40
END
```

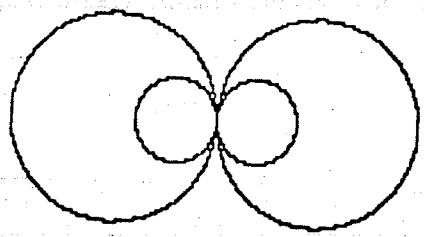


Figure 6.23

```
TO NOSE
1 RARC 30
2 RIGHT 90
3 RIGHT 90
4 LARC 30
5 RIGHT 90
6 RIGHT 90
7 LARC 30
8 RIGHT 90
9 RIGHT 90
10 RARC 30
END
```

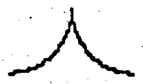


Figure 6.24

A somewhat more elaborate example was Karl's ACE procedure which involved vertical symmetry produced by using FORWARD/BACK reversals, as well as right/left symmetry.

```
TO ACE
1 RCIRCLE 50
2 LCIRCLE 50
3 BACK 100
4 RCIRCLE 50
5 LCIRCLE 50
6 FORWARD 100
7 FORWARD 100
8 RCIRCLE 50
9 LCIRCLE 50
END
```

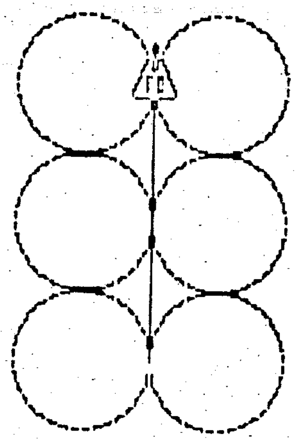


Figure 6.25

Kathy's BIRDMAN procedure was an elaboration of the "eyes" approach, using inverse arcs as well as RCIRCLE and LCIRCLE.

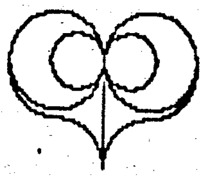


Figure 6.26

Gary regularly made use of the "LOGO symmetry theorem" in his work. An example discussed fully in Section 5.5 of Chapter 4, was his STARSHIP procedure, in which right/left symmetry was built into the entire process.

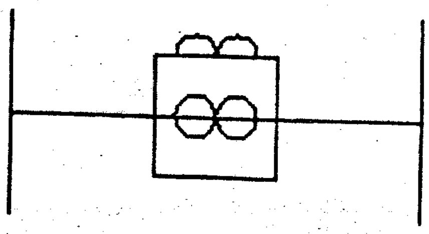


Figure 6.27

6.6.2 Projects Involving an Implied Axis of Symmetry

Many students created symmetrical designs without reversing right and left commands. Such a design is likely to be a drawing of a face, a rocket, a house, etc. Such designs may also have one or two dramatic asymmetric features.

When Kathy added a hat on her BIRDMAN design (Figure 6.26), she made use of an implied axis of symmetry.

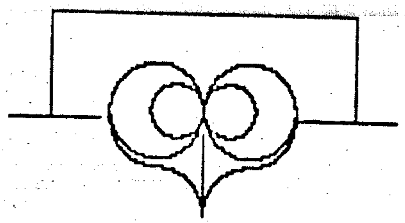


Figure 6.28

Gary's first major project was a FACE which looks as though it was constructed using right/left symmetry. Actually Gary used a more elaborate approach which involved estimating the sizes of circles and placing the TURTLE so that it could draw the circles starting from the outside. The "Nose" is the only asymmetric feature.

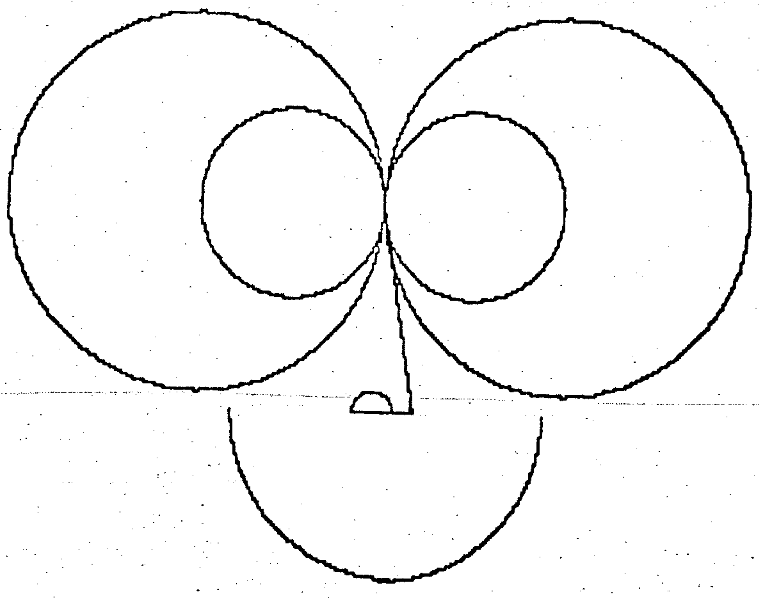


Figure 6.29

Donald's HEAD was also worked out without the use of right/left symmetry. Although his symmetrical placement of the "hat" was not exact, it was clearly intended to be so. Once again the "nose" is deliberately asymmetic, as is the flower.

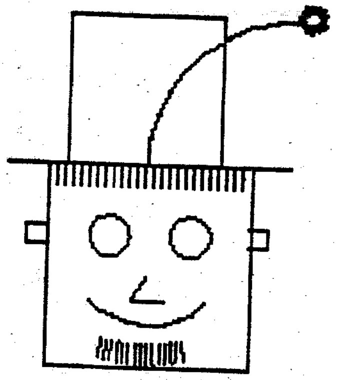


Figure 6.30

6.6.3 The Use of Symmetry in Non-Symmetric Designs.

Dennis used the "LOGO symmetry theorem" to create windows for his car. Once he had drawn the rear window, WIN1, he reversed all the angles to draw the front window, WIN3.

```
TO WIN1
2 FORWARD 60
3 RIGHT 60
4 FORWARD 20
5 RIGHT 90
6 FORWARD 40
7 RIGHT 90
8 FORWARD 50
END
```

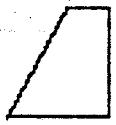


Figure 6.31

```
TO WIN3
1 LEFT 30
2 FORWARD 60
3 LEFT 60
4 FORWARD 20
5 LEFT 90
6 FORWARD 40
7 LEFT 90
8 FORWARD 50
END
```

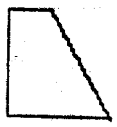


Figure 6.32

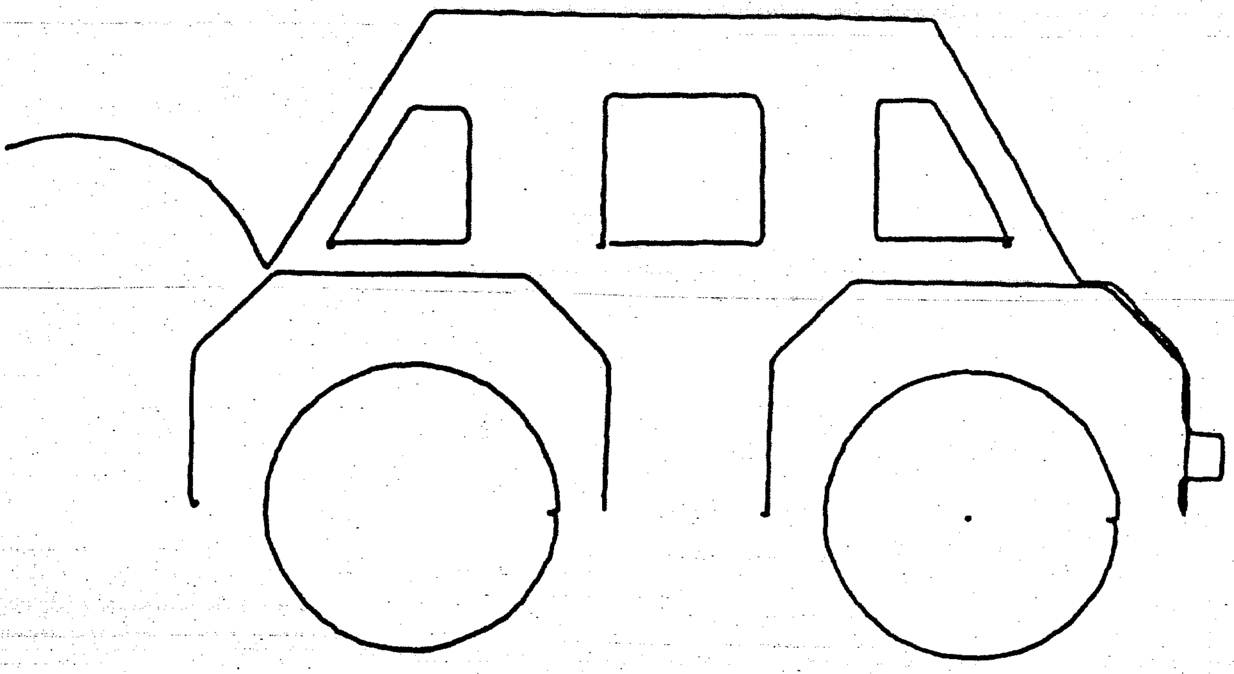
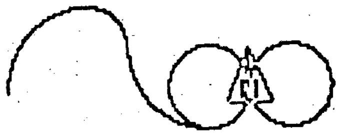


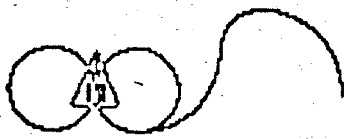
Figure 6.33

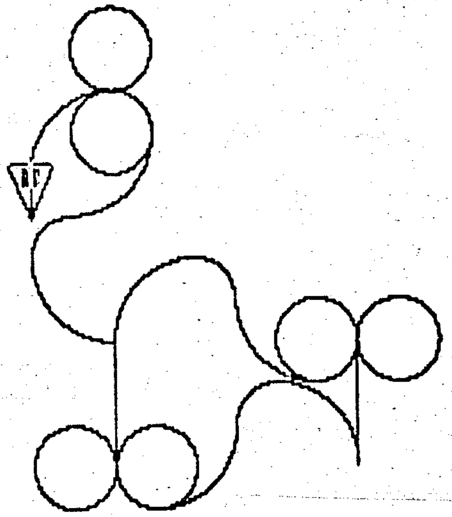
As a final example of student use of symmetry we examine Kathy's procedure MONSTER. MONSTER was produced by an exploration involving arcs and circles. When Kathy divided this into subprocedures MO, NS, and TER, MO and NS were symmetrical, while MO and TER were identical. Although Kathy probably used a symmetric process in working out her design, the asymmetry of the design as a whole obscured this aspect of it.

```
TO MO
1 RARC 40
2 RARC 20
3 LARC 40
4 LARC 20
5 LCIRCLE 20
6 RCIRCLE 20
END
```

```
TO NS
1 LARC 40
2 LARC 20
3 RARC 40
4 RARC 20
5 RCIRCLE 20
6 LCIRCLE 20
END
```


MO (and TER)
Figure 6.34a


NS
Figure 6.34b


MONSTER
Figure 6.35

6. Dynamics: Learning Physics with a Dynaturtle

During the last two weeks of the second round of classes, students were given an option of extending their LOGO world with a new kind of TURTLE, dynaturtle. With the exception of Tina who was deliberately encouraged to stay with her other work, the students found dynaturtle attractive enough to choose to spend a significant amount of their remaining time with it.

1. Description of Dynaturtle:

A dynaturtle, like the ordinary LOGO TURTLE, is a graphics entity which can be moved around on the computer display with commands typed at the keyboard. Like the geometry TURTLE, dynaturtle responds to commands, RIGHT or LEFT, by instantly turning in place. While motion for the geometry TURTLE is caused by the command FORWARD, a dynaturtle never changes position instantly, but can acquire a velocity with a KICK command which gives it an impulse in the direction the dynaturtle is currently facing. To effect real time control, one normally directs a dynaturtle with keystroke commands, R, L, and K which stand for RIGHT 30, LEFT 30 and KICK 30. Provisions were made to allow students to augment this small set of instant commands at their pleasure.

Two model games were provided for the students. The one of relevance here was called TARGET. Its goal was simply to direct a dynaturtle with K's, R's, and L's to hit a target, but to do so with a minimum speed at impact. A qualitative scoring together with impact speed was printed out when the target was reached. The initial configuration had the dynaturtle at rest aimed directly up the screen, and the target, as indicated is Figure 1.1, positioned at a bearing of 45 degrees from the dynaturtle. A single K command would cause dynaturtle to travel the distance between initial position and target in about 15 seconds. The introduction to dynaturtle given to students was a brief description of commands together with an illustration, applying a few "kicks" to a tennis ball on a table using a small wooden mallet.

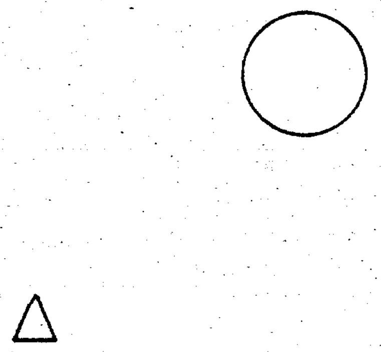
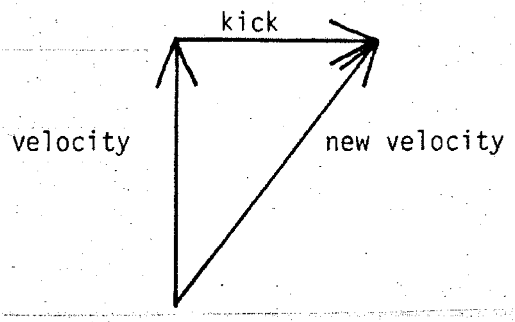


Figure 1.1
Initial Configuration of Target

Dynaturtle and the model games provided were designed with the intent of introducing the students to the Newtonian notion of controlling an object by changing its velocity. The fundamental principle is that an object has a velocity (both direction and magnitude) as part of its state, which is preserved until an interaction (KICK) changes the velocity. KICKs on the other hand act in a very simple way which can be described by vector addition (Figure 1.2): The old velocity is changed into a new one by adding a KICK vector whose length is the input to the KICK command and whose direction is the current heading of the turtle.



Vector Addition

Figure 1.2

It was expected that after initial familiarity with the system was gained, students would modify the model games or use the dynaturtle in more flexible ways to suit their own purposes. This did indeed happen, but the initial period of coming to understand dynaturtle was so rich and suggestive by itself that we decided to concentrate in this report only on that aspect of the students' encounter with dynaturtle. In particular, we have chosen to look carefully at the data to try to answer two questions:

1) What and how are the students learning from dynaturtle?
2) How might that relate with more curriculum centered notions of learning physics?

2 Overview of Results:

To set a perspective on students experience with dynaturtle we will first make a brief oversimplified sketch of the experimental results. In a sentence, almost everybody did essentially the same thing. This was particularly surprising in view of the striking differences in abilities and style which the students exhibited in their other work, and it is the basis for the suspicion that there is a fundamental phenomenon at work here.

In more detail the dominant theme of these students' encounters with the dynaturtle is their exposure to, and learning to control, a Newtonian object. This in turn, without much interpretation, can be seen as the confrontation of an essentially Aristotelian theory of physics with a Newtonian reality. For our purposes here we use the term "Aristotelian physics," to mean that objects simply move in the direction you push them. In similar terms Newtonian physics states that pushes change velocity. The conflict is whether force correlates with changes in velocity (Newton) or with changes in position (Aristotle).

The germ of the conflict resides in a simple situation which all the students encountered and all regarded as problematic. Suppose a dynaturtle is moving upward and one wants it to move to the right (Figure 2.1a). The Aristotelian strategy is simply to aim to the right, then kick in that direction, and the expectation is as shown in Figure 2.1b. "Kick to the right means move to the right." In contrast, the Newtonian dynaturtle moving upward has momentum in the upward direction which is not affected by the sideways kick and thus takes a "compromise" path following the kick as shown in Figure 2.1c.

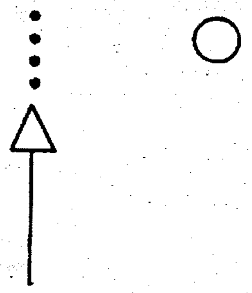


Figure 2.1a
Moving Upward

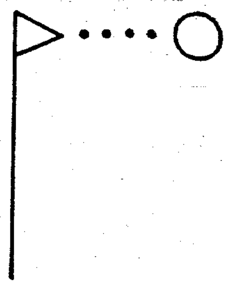


Figure 2.1b
Expected Result
of a sideward kick

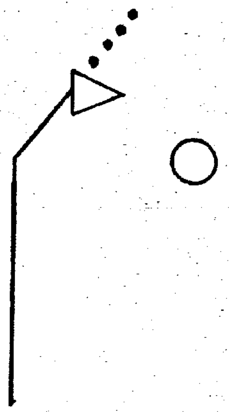


Figure 2.1c
Actual Result

All the students spontaneously generated the sideways kick as a means of making a right turn and expressed surprise and consternation at the result. Complaints that the machine was not working right at this point were commonplace. The robustness of the students' theory is attested to by the fact that, though many of the students had made significant progress in the two weeks of exposure, none proved to completely shed the Aristotelian disposition. We are thus led to consider this as a situation potentially involving significant learning and cognitive restructuring.

3. Detailed Observations of Two Students

3.1 Jimmy's Experience With Dynaturtle

The following gives more detailed account of our observations of two of the students who were observed at greatest length. It is interpreted principally in suggesting a frame for appreciating the importance of the observations both in terms of knowledge (cognitive) structure and in terms of physics pedagogy.

Jimmy's early work with the dynaturtle was typical in the sequence of strategies, successes and failures. His initial strategy with the target game was a simple AIM and SHOOT, the almost universal starting place of all subjects:

(1) Turn the dynaturtle with Rs and Ls until it's facing the target
(2) Shoot using Ks

AIM and SHOOT is one of the principal strategies involved in playing with the geometry TURTLE, and it may therefore be straightforward transfer to find it immediately and clearly implanted in this slightly different environment. Though it seems very natural to the situation, one should remember that children this age are notorious for simply "messing about". A clearcut and essentially instantaneous use of a strategy, no matter how simple, is not the general state of affairs.

This strategy fails as it stands: the target is at a 45^{o} bearing and R and L cause the dynaturtle to turn in 30^{o} increments, thus AIM and SHOOT necessarily carries the dynaturtle off to the left or right of the target. Once Jimmy saw the failure, he summarily dropped the strategy. There is every reason to believe, especially considering his experience with the geometry TURTLE, that Jimmy understood the problem and simply looked for alternatives. This is in sharp contrast to what happened with his next strategy.

The alternative plan Jimmy (and most others) adopted was to move straight up the screen, then, when the dynaturtle was at the same height as the target, make a right hand 90^{o} turn and run into the target. (Figure 2.1) *It is possible that the universality of this step with the students is due to the fact that they have already had significant experience with the geometry TURTLE in as much as this corner movement is a very frequently observed strategy in that domain. It is used to achieve accurate positioning (as in positioning parts of a picture). On the other hand that strategy had been neither taught nor even named or remarked* *upon.* It is only a slight abstraction to see this as achieving independence of control of two degrees of freedom (horizontal and vertical positioning). Thus it is an important "proto-coordinatizing" with similar but somewhat different implications in dynamics compared to the "proto-coordinatizing" discussed earlier in the geometric case. (See Chapter 5, Section 5.3.1, p. 5.48)

The CORNER strategy, of course, brought Jimmy quickly to the very heart of the Aristotelian-Newtonian controversy. The Dynaturtle skipped diagonally away from the target rather than toward it. His first instinct on failing was to try it again and again. Then he applied more kicks at the right angle turning point. AIM and SHOOT had failed for an understandable reason -- he did not complain when it did. But CORNER had no good reason for failure in his eyes, and he complained and appeared frustrated.

At this point an intervention was made discussing with him the essential difference between TURTLE and dynaturtle. Out of the discussion arose a new strategy which was neither explicitly proposed to him nor entirely spontaneous on his part: at the corner, stop the dynaturtle with kicks in the direction opposite to its motion, then AIM and SHOOT directly into the target. (The stopping kicks which canceled initial kicks were named *antikicks* by another student, a name which we appropriate.) Jimmy understood and quickly applied this strategy, which we will call a *Newtonian Corner* (Figure 3.1).

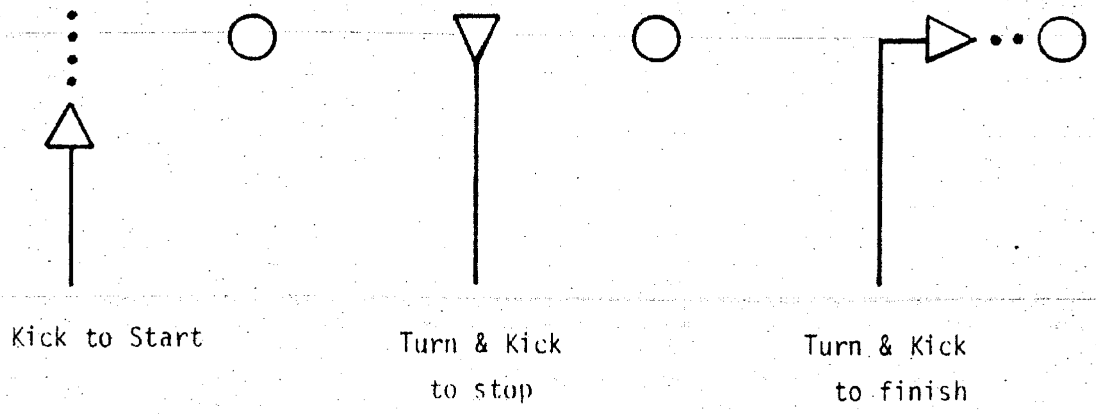


Figure 3.1

There are two significant points to make at this juncture. Jimmy never did exhibit any confusion between turning and kicking; that they are independent actions (perhaps modeled on the independence of forward and turn commands for the geometry TURTLE) was taken for granted. Thus he began turning around immediately after kicking to start the dynaturtle, without expecting to see any change in motion until new kicks were applied. This is important as it shows that he did not have trouble disassociating aiming from moving in his switch from geometry TURTLE to dynaturtle. Without this fact one would be tempted to attribute Aristotelian expectations to a simple carry-over of the fusion of direction-pointed and direction-moved which characterizes the geometry TURTLE. The lack of difficulty in differentiating direction of motion and direction of pointing was true of the other students as well, and may indicate a transfer from the qualitative structuring in Turtle Geometry noted in Chapter Five.

Secondly, Jimmy knew without being told and before experimenting that the number of kicks he needed to give to stop the dynaturtle was the same as the number he gave to start it. He did not worry about timing but only about number. *Again we do not know to what extent this is a carry-over of learning the power of inverse operations in LOGO. As cited in earlier sections of this report, LOGO Turtle Geometry students often give clear indication of coming to grips with inverses in a form we can exemplify with the equation RIGHT 90 LEFT 90 = no change. If kick-antikick inverses with the dynaturtle are due to previous LOGO* *training, we have discovered in this a possible test for transfer of spontaneous learning in the LOGO environment.*

Having developed a failproof strategy which he understood, Jimmy concentrated for an extended time practicing and elaborating it.

3.2 Darlene's Experience with Dynaturtle

Darlene started out with the same AIM and SHOOT as Jimmy but was more patient in trying to debug it. Because of her care not to give too many kicks she in fact succeeded in hitting the target, but not reliably. Trying to follow Jimmy's CORNER path (she could see his screen) she fell into the same, Aristotelian trap. Again at this point an intervention was made, illustrating side kicks and the resultant diagonal trajectory with the tennis ball and mallet. This appeared to engender a state of disequilibrium. She made it clear with facial expressions that she was quite dubious about this "experiment," grabbing the mallet and trying it herself several times. "There must be a way," she said, twisting the mallet as she hit the ball. She was shown Jimmy's Newtonian CORNER strategy of a hit to stop, re-aim and new hit, but still indicated a wish to see the corner accomplished with one kick. A diagonal backward kick was suggested and demonstrated (Figure 3.2), but she refused even to consider that. "I like Jimmy's strategy," whereupon she returned to try it out on the computer.

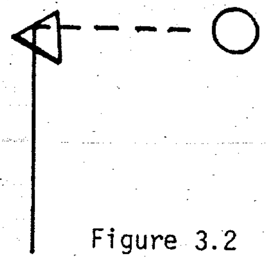
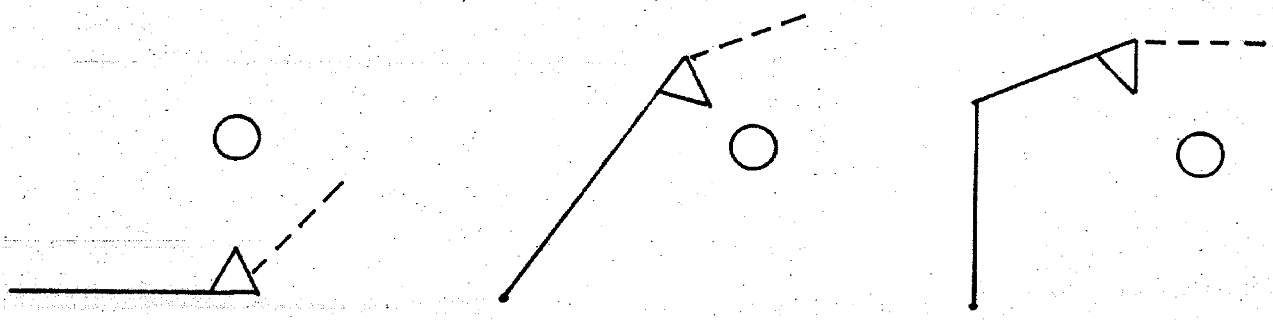


Figure 3.2

Diagonal backward kick
turns the corner

Darlene was not content as Jimmy was to stay with one method. Over the next few days she tried many others. In particular, the next day she tried the CORNER strategy starting out horizontally rather than vertically (Figure 3.3a). Of course it failed. She tried to correct an AIM and SHOOT by kicking at the target as the dynaturtle passed a point on its trajectory perpendicular to the line to target (Figure 3.3b). It failed as well. She tried and failed at correcting the Aristotelian Corner's defects with a second kick, but using the same "kick toward the target" strategy (Figure 3.3c). These are very interesting and crucial experiments. Though to a "Newtonian" she is merely rehashing a known result -- the CORNER strategy fails -- a little more detailed thought suggests some very important developmental function in these activities.



a. Horizontal Aristotelian Corner

b. Correcting AIM and SHOOT another Aristotelian Corner

c. Another Aristotelian Corner

Figure 3.3

Let us ask how Darlene (or we ourselves) could understand that these different experiments are really the same. There are two possibilities. One is that she understands the set of operations (rotations and reflections) which leave the experiments "the same". These are not much different from those which leave a geometric configuration the same -- a book remains perceived as a book whatever its orientation. The gedanken experiment which makes the argument is the observation that turning something around is just like looking at it from a different standpoint -- which shouldn't change anything substantial. On the other hand, with motion and trajectories in the real world, gravity introduces a forceful anisotropy. It seems quite plausible that someone would want to experiment to really test the group of symmetries.

The second possibility for understanding the identity of the experiments is that they are described in an invariant language, *That is to say, the description is unchanged if a different point of view on the experiment is taken.*, a language which assumes frame of reference (up, down etc.) has no functional significance. Indeed, that's the sort of description we have been using in phrases like, "kick perpendicular to an established trajectory." [We do not mean to imply that Darlene's internal description is verbal, but for the sake of this short discussion we will pretend that it is so.] It is quite likely that, though the geometry TURTLE was a help, Darlene could not initially generate such a precise invariant description. More likely her description would have been something like, "It doesn't go straight when you kick it sideways," or even, "Sometimes it doesn't go straight." In this frame of mind it's natural to try experiments of the sort that she did.

Seen in this way Darlene's experiments make utterly clear scientific sense as a way of developing and refining an invariant description. In fact, invariant descriptive techniques along with a few paradigmatic phenomena and strategies (the CORNER strategy's failure and Jimmy's start and stop method are two; we'll mention more shortly) could constitute the basis for a reasonably complete and efficient understanding of dynaturtle.

The symmetry group versus invariant description routes to understanding the conceptual integrity of the Aristotelian CORNER are probably not really alternatives, but each valuable constructs for understanding part of Darlene's growing understanding. We expect that the intuitive roots for even beginning the route to full appreciation of invariance and symmetry lie partially in both camps: the feeling that rotating the thing (phenomenon) doesn't make much difference to it interacts with the proto-invariant sensation that there <u>is a thing</u> (i.e. invariant object) to be studied.

Darlene was presented another workable refinement of the Aristotelian CORNER strategy in addition to Jimmy's start and stop Newtonian CORNER: "Cut the corner," turn and kick sideways early. We call this the EARLY strategy (Figure 3.4). This makes good intuitive sense; just thinking of dynaturtle as being slow to respond is a good heuristic.

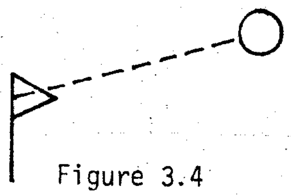


Figure 3.4

Kick Sideways EARLY

The strategy brought along with it, without prodding, the corrective feedback loop, "If you miss by getting to the target *too late*, (meaning x-coordinate position "gets to the target" after y-coordinate *It is interesting that such a formal description hides the trivial obviousness of kicking too early or too late as perceived by the students.*) then kick *earlier*," and vice versa. Kicking "too late" or "earlier" in fact are expressions universally used by our subjects to describe the phenomena and their own intent. It doesn't seem difficult to understand the naturalness of such a strategy. For example, in getting to school at a certain time one may employ the early-late conceptualization and feedback loop with respect to the question of when to leave home. But in any case, despite the fact that an intervention was necessary to cue the EARLY strategy, it is important that EARLY found a natural home in this situation. Contrast Darlene's refusal to consider a suggested diagonal backward kick to enact a CORNER.

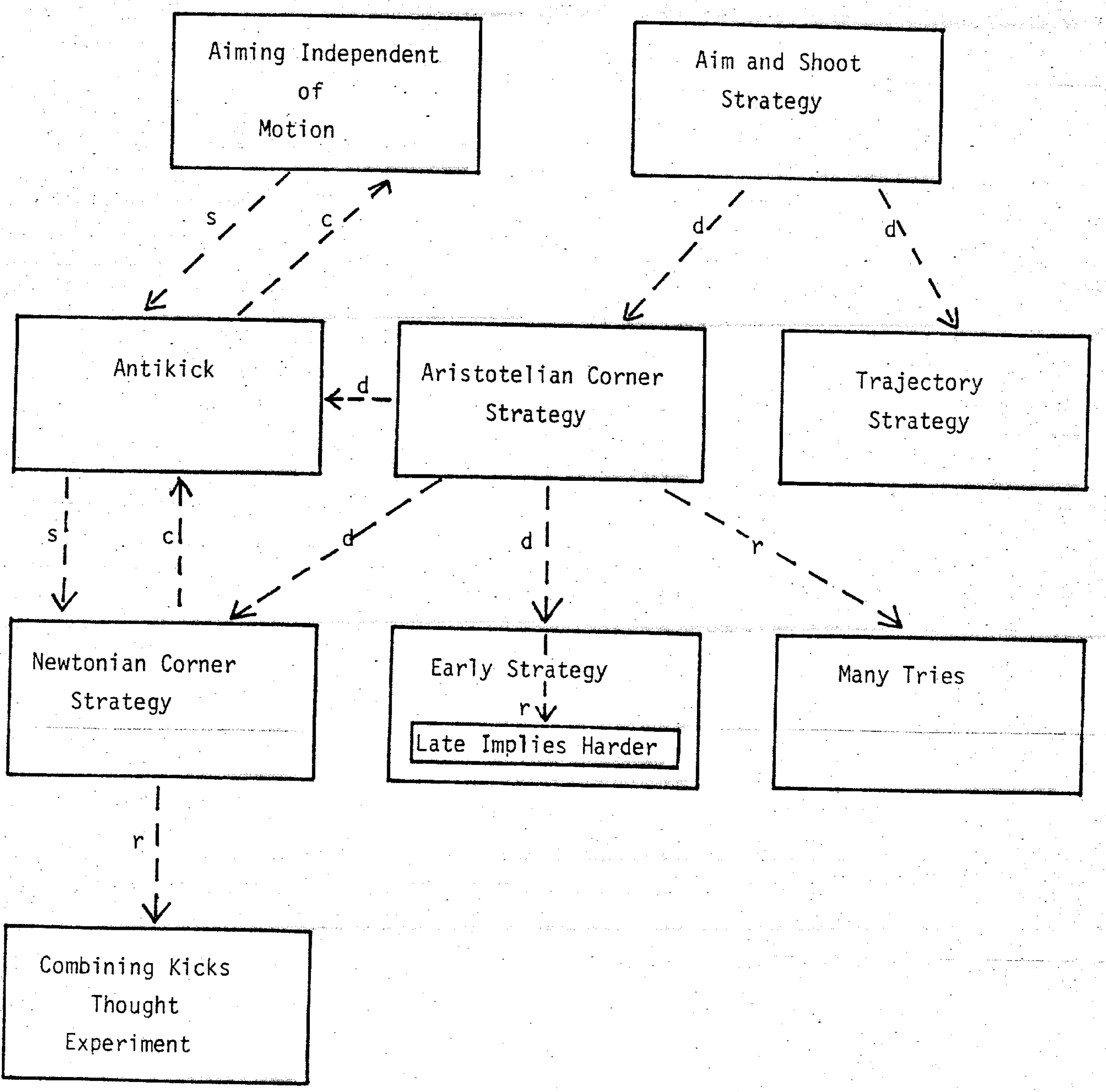
A final episode of Darlene's experience with the dynaturtle is worth mentioning here. After quite a bit of play (much of which has been described above) another attempt was made to bring her to understand the single kick method for making a CORNER (Figure 3.2). She was asked to think of using Jimmy's Newtonian CORNER method (two perpendicular kicks, one to stop, re-aim, another kick toward target) but with the kicks coming very close together in time. Now think of a single kick having the same effect as the two. "I know" she said. "You want to kick at 45 degrees!" *She meant a 135 degree turn as became evident at the computer.* She was in fact anxious at this stage to try out the method. This is a solid leap. Now she was making explicit a new, developing intuition -- vector addition of kicks. This contrasts markedly with her pre-experimentation overt rejection of this same suggestion, one which she can now make on her own.

4. A Learning Paths Chart for TARGET

Chart I summarizes the student development in understanding in terms of insights and strategies. It is based on work with a number of students, both children and adults. Generally speaking a lower point on the chart indicates a later development.

CHART I

Learning Paths Chart for TARGET


Aiming Independent of Motion
Aim and Shoot Strategy
s
c
d
d
Antikick
d
Aristotelian Corner Strategy
Trajectory Strategy
s
c
d
d
r
Newtonian Corner Strategy
Early Strategy
r
Late Implies Harder
Many Tries
r
Combining Kicks Thought Experiment

Annotation of the Chart:

Strategies and Phenomena

Aim and Shoot Strategy - "Turn the dynaturtle toward the target and kick" is the universal starting point. Its failure, due to size of turns incommensurate with 45 degree bearing of the target, was accepted and never problematic.

Aristotelian Corner Strategy - This is the classic non-Newtonian strategy described earlier involving the assumption that the dynaturtle travels in the direction of kick. Its failure always resulted in great surprise, repeat tries and sometimes extended consternation. Note that while the interpretation made in the last section might suggest Aim and Shoot and the Aristotelian Corner are both expressions of the same theory, they are included as separate elements. This is done because their different contexts provide strongly different strategic outcomes. One satisfies expectations; one does not.

Trajectory Strategy - This strange strategy did not occur frequently, but often enough to warrant including. To debug Aim and Shoot, some seemed to posit a curved path approach and, as a mechanism for obtaining such, a systematic and repeated pattern of K's and R's were used, e.g. K R K R K R. "Curving" and "starting to curve" were frequent verbal accompanying descriptions. There seemed to be the assumption of a simple relation between the turn-kick combinations and the amount of curving. In any case, the rate and pattern of keystrokes were the parameters varied in an attempt to refine this strategy into a working one.

Aiming Independent of Motion - The default assumption made naturally by everyone was that turning would not affect the direction of motion until a kick was given. Despite this, various circumstances called this hypothesis explicitly into question. The most important of these circumstances was the context of trying out the antikick idea (perhaps worrying about a presumed "minor" effect interfering with exact cancellation of kicks). It is interesting to note that despite Aiming Independent of Motion, students frequently associated reaiming with the subsequent kick, and did reaiming just before the kick, even if there was much empty wait time preceding the reaim-kick combination.

Antikick - The "kicking the opposite way to cancel a kick" phenomenon was a spontaneous idea in most cases and an immediately accepted suggestion in the rest. Perhaps its importance lies in the function of achieving the stopped state. Three subspecifications are important: 1 Kick-antikick starts and ends at rest, superposition on an initial velocity is not conceptually possible at this stage. 2

It is assumed that any number of kicks will be exactly canceled by the same number of antikicks, the timing of the kicks not being an issue. 3 Interspersing kicks in other directions between kicks and antikicks disqualified the antikick strategy.

Newtonian Corner Strategy - The canonical debug of the Aristotelian Corner (see Figure 3.1) was almost always associated with right angles (as the Aristotelian Corner was).

The Early Strategy - (Figure 3.4) This appears to be a more sophisticated strategy than the above as it did not occur in many of the children's protocols. The refinement of the Early Strategy marked Late Implies Harder, meaning "if you are late, you can kick harder (more)," notably did not occur simultaneously with the principal Early Strategy.

Many Tries - The overt trying over and over of the Aristotelian Corner Strategy in many orientations as carried out by Darlene's was not as frequent as it was striking when it did occur. Usually students fell into trying the strategy in new circumstances, seemingly by accident, without noticing initially that they were doing the same thing again.

Combining Kicks Thought Experiment - (See description in section 3.) This is included not because it was universal, if fact it only occurred once, but because it marks a striking advance over initial inclinations. This is the sort of intervention result one would like to be able to produce reliably.

Links

The links shown on Chart I are a preliminary attempt to map relations between strategies and phenomena, particularly those relations having to do with time sequencing, though not explicitly including the latter. Time sequencing is not included explicitly since there were many small variations.

d: debug or find another plan - The failure of a strategy often seemed to have a contextual influence on what was tried next. Most of these relations seemed straight forward "minimal changes to effect success," i.e. debuggings a la Aristotelian Corner → Newtonian Corner. Some were more subtle such as Aim and Shoot → Trajectory, and may in fact be only the student's way of answering the question "What else could I do?"

s: structural relation - When a phenomenon played a key role in a strategy, or in other cases when one element of the chart played the role of prerequisite to another, the relation is designated as structural. The relation of antikick to the

Newtonian Corner Strategy is paradigmatic.

c: contextual - Certain elements of the chart seemed to become active as a result of concern in another element. These were designated as contextual relations. Working out the idea of antikick not unexpectedly served as a context for worry about the absolute validity of the Aiming Independent of Motion principle. The natural symbiosis of structural and contextual links is evident in Chart I and is easy to understand: An element (strategy, phenomenon) which is prerequisite for success in pursuing (or understanding) another element is naturally cued to concern by that other element. However, it is not at all impossible to imagine circumstances where contextual and structural links are not dual to one another in this way.

r: rehearse and refine - Running over a particular strategy, presumably in order to solidify and improve performance, perhaps consciously changing parameters, was a frequent occurrence. An interesting variation is the relation of Many Tries to the <u>failure</u> of Aristotelian Corner, where rehearse and refine served to elaborate understanding of the circumstance under which Aim and Shoot failed, i.e. served to distinguish successful (in the sense of satisfying expectations) Aim and Shoot Strategy from the unsuccessful Aristotelian Corner Strategy.

5 Linking to Physics

Pointing toward future work with dynaturtle, we speculate a bit in this section on the relation of the student learning phenomena charted above with more standard views of learning physics. The section consists of two plausibility arguments.

A. Controlling Dynaturtle *is* Physics:

One can make the case that learning to manipulate a dynaturtle is *prima facie* learning physics. Certainly the students are learning to deal with (simulated) physical phenomena and to the extent to which one does not insist that that contact be mediated by symbols and formalisms, that is physics. Furthermore, dynaturtle was designed to mirror at least 1 1/2 of Newton's 3 laws: 1 Straight line constant velocity in the absence of interaction; 2 The Second Law without the effect of mass, i.e. change in velocity is proportional to strength of force (number of kicks).

So for the moment let's assume this point of view, that controlling dynaturtle is physics and see what kind of perspective the charted student learning behaviors puts on learning to control it. For contrast let's initially consider a more *a priori* point of view by first looking at dynaturtle from the perspective of an expert physicist. Try to put aside one of the important experimental results implicit in what has been said so far, that average sixth grade students can learn to drive dynaturtles.

A parsimonious description of dynaturtle can involve a vector component of state (velocity) and a state changer (KICK) which increments velocity by vector addition. The task analysis might quite reasonably begin with the notion of instantaneous velocity and vectors (including vector addition, component decomposition, etc.). Thus we see appearing a familiar list of prerequisite studies (for college students!) including perhaps analytic geometry, trigonometry, etc. Can we seriously think a fifth or sixth grade student can understand dynaturtles and manipulate them without months or years of study, or "at best" by learning by rote incomprehensible algorithms?

Now contrast the following rich and realistic task analysis based on the TARGET learning chart which we summarize in the form of a sequence of natural (and in some cases trivial) abstractions of experience with dynaturtle.

(1) The remark that aiming does not affect motion

(2) The warning that AIM and SHOOT fails when the dynaturtle is in motion

(3) The phenomenon of an antikick and its powerful use in producing a true Newtonian CORNER strategy

(4) The EARLY strategy and its refinement, Late Implies Harder

(5) The thought experiment of combining kicks as at the Newtonian CORNER (or the reverse, thinking of a diagonal kick as a backward kick to cancel present motion plus a sideways kick to establish a new direction)

(6) Establishing an invariant recognition capacity of dynamical phenomena, as Darlene seemed to be doing in rehearsing the Aristotelian Corner

This list is essentially a path through the learning paths chart which happens to be both a sort of "average" observed path, and a seemingly natural pedagogical path. For reference we will call this particular path the *modal path*.

B. Physics beyond controlling dynaturtle

A small further abstraction brings the modal path experience much closer to recognizable, textbook physics.

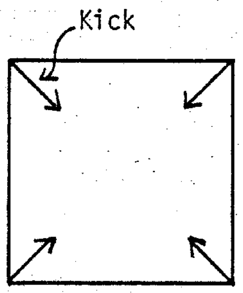
(1) and (2) combined are a strong affirmation that force and direction of motion are uncoupled.

(3) proposes a very special case of vector addition, $v - v = 0$ or $v + v + \ldots + v - v - v - \ldots - v = 0$, which in the context of action and "undoing" counteraction seems very intuitive. It is important to note that this intuition is lost if confounded by intervening kicks or even if it is superimposed on an established velocity.

(4) The EARLY strategy and especially its refinement can easily be seen to involve qualitative versions of vector addition, in this case vector addition of an established velocity with an impulse. It is especially nice that students naturally did this with right angle kicks so that the pedagogically important special case, orthogonal composition, is exercised.

(5) Combining kicks at the Newtonian Corner is another step toward understanding the full implications of vector addition. In this case two kicks are added to each other to produce a theoretically equivalent kick. (Note in Figure 5.1 how close a series of combined kick Newtonian Corners is to the extremely counter-intuitive circular motion caused by kicks toward the center of the circle.) Again the natural

right angle context is pedagogically advantageous.



Motion in a square path caused by inward diagonal kicks, mimics centripetal acceleration of a circle.

Figure 5.1

(6) has already been abstracted to this level in our description of it.

By itself this modeling has proposed an interesting and detailed refinement of what in a standard curriculum might go into a monolithic chunk entitled vector addition (of velocities, forces and impulses). The list is refined in at least two important respects. It singles out particularly "easy" and particularly "difficult" special cases, both of pedagogical interest. It introduces context, e.g. vector addition of kick-to-kick versus vector addition of kick-to-velocity, in cases where context seems to make a difference to the students.

In incorporating an experience with dynaturtle one may establish a rich base of phenomena and common experience with Newtonian physics in a open-ended environment. Following that experience, students should have a great head start in understanding the formalism of mechanics, principally through being able to interpret the formalism in their own experiences. This latter stage may be greatly enhanced by organizing the formalism phenomenologically, as suggested above.

It is natural to ask, why can't the students' experience in the real world serve the same purpose as an experience with dynaturtle? In the first instance, note that experience in moving things around did not suffice for Aristotle (or any other pre-Galilean) to acquire the sense for the Newtonian laws of motion represented in dynaturtle. In part this may be accounted for in the striking ability of humans to hold theories of their own action which contradict what they do in fact. Dynaturtle's advance over naive experience, then, lies in the explicit and unambiguous actions taken to control it. Experience with dynaturtle is mediated by a very narrow channel of kick and turn commands as opposed to interpretation of complex muscle actions actually used by humans in moving things around.

But a better explanation why the real world doesn't teach Newtonian mechanics probably lies in understanding how good a non-Newtonian theory like "kicking in the direction of intended motion" can be. Certainly it suffices for cueing up a billiard ball and works whenever impulse dominates existing momentum. Further, in many circumstances one simply <u>arranges for</u> the theory to work. Compare a soccer player who stops a ball as a matter of course before kicking again, to the Newtonian Corner strategy.

Finally, in the real world, friction has two confounding effects, one supporting Aristotle and one denying Newton. By rapidly bringing velocities near zero it allows an Aristotelian plan to be more generally effective, thus mitigating the need for refinements. More fundamentally, friction denies Newton's First Law by its very presence; the world is *prima facie* non-Newtonian. Since friction is omnipresent and with no visible agent causing it, why should one either implicitly or explicitly treat the "dying away" of motion, so much like other inescapable things, as other than a primitive phenomenon (law) of nature? It is only by coming to understand the Newtonian stance that one even acquires a reason to separate friction as another force to be included in the analysis. And beyond the First Law, of course, the Second Law doesn't work without frictional forces being explicitly included. Summarizing this line of reasoning, a Newtonian frame of analysis seems necessary to make sense of the notion of friction as a force, rather than as a fundamental and universal phenomenon intrinsic to motion. Yet a Newtonian frame is only possible after one has separated out friction as a force to be added to the analysis. In the present case, the bind is not inescapable, as we can simply remove the confounding element from the (simulated) world. Dynaturtle is a pure representation of Newton's Laws, unfettered by friction.

We note in closing this section that though it would be easy to think that the qualitative understandings involved with dynaturtle are superfluous and in all probability trivially implied by a university level physics course, the evidence available suggests that, to the contrary, such qualitative reasoning as that involved in the relationship of force and velocity is in many instances little effected by physics teaching. Viennot [Viennot, 1978], for example, found clearcut confusion between force and acquired velocity in simple situations. (Compare (1) and (2) at the beginning of this section.) This confusion occurred in roughly 50 percent of students from last year of secondary school to third year university! Thus another suggestion is made that learning in computational environments like Turtle Geometry and dynaturtle can allow earlier and more natural access to important mathematical and physical ideas, but, as well, provide a <u>deeper</u> influence on the students' thinking patterns than conventional curriculum.