Teaching Young Children to Program in a LOGO Turtle Computer Culture

Cynthia J. Solomon*
80 Ellery Street
Cambridge, MA 02138

What programming knowledge and skills can a young child acquire? This question is no longer shocking although it remains unanswered. Now that the computer presence is clearly a growing part of our lives, the notion that 6 and 7 year olds could learn to program does not bring scorn and doubt into audiences' minds.

This paper describes a particular computer culture and environment in which young children have learned to program. The child as problem-solver is discussed in terms of three identifiable cognitive styles; and finally, some teaching strategies are suggested which take into account these different learning styles.

The Computer Culture

"LOGO" is the name of a programming language, but it is also used as the name of an environment, a culture, a way of thinking about computers and about learning and about putting the two together. The environment is made of ideas, of things, and of people. The things include not only the computer, but computer controlled devices like turtles. There are mechanical turtles which move along the floor and are often equipped with touch or light sensors, and there are also graphics turtles, which live on TV-like screens where they draw in phosphor white or in multi-color. The computer system which gives life to all of this understands the LOGO language. The computer and the programming language play a vital role in creating an exciting atmosphere where programs, people, turtles and other computer controlled devices interact with one another and learn from one another. In the environment people become researchers, and actions and ideas take on animate qualities. Ideas from computer science like naming, procedurization, and debugging become intermixed with anthropomorphic thinking to become lively tools in problem-solving situations.

Different turtle types naturally develop distinct attributes, but there are also common elements. For example, the turtle's state is its position and its heading. Its state can be changed by either telling it to go FORWARD (or BACK) a number of steps or telling it to turn RIGHT (or LEFT) some number of degrees. It can also leave a trace (PENDOWN) of its path or not (PENUP).

*

While the culture is closely tied to turtles, it is certainly more universal. The turtles were invented as vehicles to convey this culture to beginners. They make certain images more vivid and certain ideas more concrete. But the goal is to convey these ideas and images, to make them real, comfortable, personal for a beginner of any age.

A Functional Description

Functionally, the LOGO environment is made up of the following:

(1) a computer

(2) a programming language and an operating system

(3) a collection of computer peripherals, usually including graphics and turtles

(4) a collection of projects

(5) a meta-language - a consistent way of talking about the language, the projects, etc.

(6) a relationship between teacher and learner

(7) a collection of "bridge activities" like juggling, puzzles, etc.

All of these components are interdependent and the special virtues of the environment follow from their coherence with one another. Taken individually, they have no great merit or utility. For example, one would expect very limited educational benefits to come from teaching programming, even LOGO programming, in an "abstract" environment or from using turtles as toys without a vision derived from the computer culture.

The design of the LOGO environment as a whole is strongly influenced by certain general ideas of which three are particularly relevant to work with young children: procedurization, anthropomorphization, and debugging. The following three sections discuss these in turn.

A Procedural View of the World

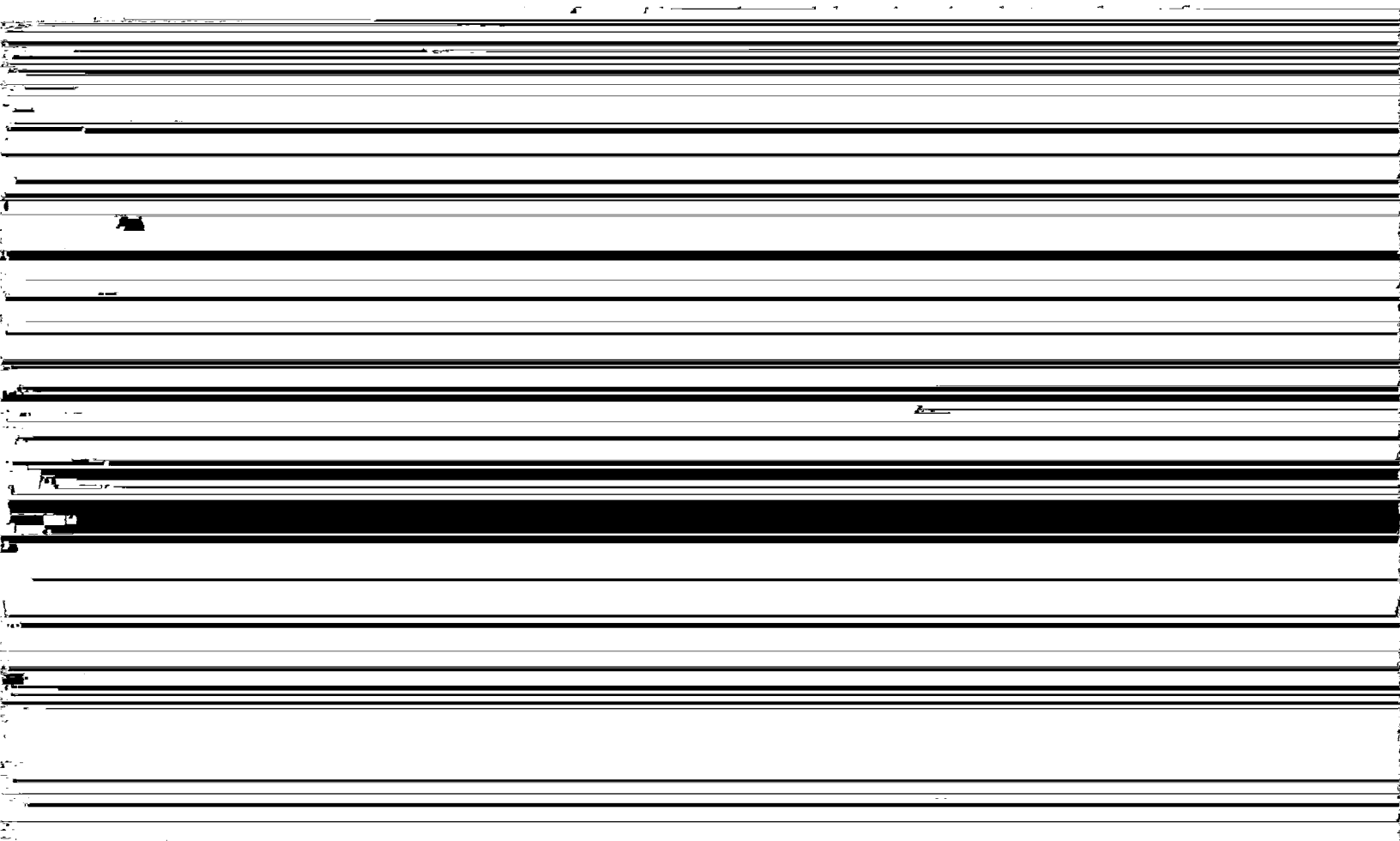
A procedural view of the world touches upon all aspects of our culture. Taken in its simplest sense, a procedure is a description of how to do something, and when applied to the world, leads to a perception of complicated processes in terms of subprocesses. That is, complicated processes are reduced to an interconnected cluster of simpler processes, each of which can be clearly described. In the LOGO world, whether a child is learning to walk on stilts or to juggle three balls or to make the turtle walk in a square, the main intellectual activity is to look for a set of procedures which, when knit together, will do the job. The intellectual environment we are describing is designed to exploit this commonality in order to channel prior real-world procedural knowledge into the service of mastering the computer and also to channel whatever is so learned back into improvements of knowledge about the non-computer world.

As a support to procedural thinking, emphasis is placed on giving words meanings, naming processes, and making descriptions for how to do things. These ideas are embodied in LOGO, the programming language. (A real attempt was made to minimize the formalisms of language so as not to detract from naming, procedurizing, subprocedurizing, recursion. Further work is needed here and become dramatically apparent in work with young children.)

An Anthropormorphic View of the Computer

Anthropomorphizing, "ascribing human characteristics to non-human things", has been a natural way to understand aspects of the world. It can also be seen as a powerful problem solving tool. Its pervasiveness is supported by the fact that we talk about a "LOGO turtle environment" or a "computer culture" or "LOGO worlds", etc., and are understood. Turtles themselves are concrete realizations of this thinking. On a more abstract level, programs as well as turtles are looked at anthropomorphically. This gives rise to powerful teaching strategies such as the use of metaphors like "playing computer", "being the turtle", "being the procedure", "naming the actors and describing their roles", "teaching the computer new words", "teaching the turtle how to do something". A further extension of these teaching strategies is embodied in the idea of encouraging young students to think of themselves as studying turtle behavior or computer behavior in order to learn about themselves--both how they are the same and how they are different. Thinking in terms of using what we know in order to teach the computer requires us to know some of its essential attributes as well as our own, and at the same time feeds into and is supported by a procedural view of the world.

Debugging



never seen before. I do not know in advance what the answers are. One of the most exciting discoveries made by the children is just that: "You mean you really don't know how to do it", exclaimed one child in amazement and in reaction to a hundred remembered situations in which teachers put on the stance of "let's do it together" while really knowing the answer in advance. For some children the prospect of an honest relationship with the teacher is something new and inspiring. This environment is especially good for developing such relationships because it is so "discovery rich". One of my goals is to convey to other teachers the possibility of this "teacher-and-student-as-research collaborators" kind of relationship. The the extent that we can achieve this, we see one way in which the effect of the computer presence goes beyong "using computers". Its real impact is on the total culture of which teacher and child are part.

The Skills a Child Might Use in Programming

Initial studies of young children allow me to construct a plausible list of skills which a child might need in order to construct a program. For example, imagine a child writes a program in LOGO to draw a face like

Such a project involves the following elements:

(1) Setting up a plan for the project

   (a) identifying the parts

   (b) naming each part

   (c) picking a starting state for the turtle (in this case, starting at the center greatly simplifies the plan)

(2) Using procedures conceptually, e.g., CIRCLE procedures

(3) Using inputs, message passing

(4) Scaling figures and rotating figures

(5) Debugging the design, e.g., recognizing deviation from the original plan like eyes too big (so change input to circle); nose too far from center (so either change turtle's heading or change turtle's position before running circle procedure).

(6) Defining procedures formally (without inputs)

(7) Using define procedures as subprocedures

(8) Recursively defining procedures

(9) Debugging procedures, e.g., recognizing that an instruction is missing; recognizing that a command is misspelled; recognizing that the numbers input are revised.

I have observed all these elements in work with my first and second grade subjects. Other elements of LOGO programming which have not been observed in such small children but which seem to be worth trying to teach are:

(a) Defining procedures with inputs
(b) Using conditionals
(c) Using debugging aids

Developing Teaching Strategies in an Anthropomorphic Computer Culture

The development of teaching strategies as well as the accessibility of programming skills are influenced by (and influence) how the system--the language, the devices, the debugging aids--can be used or modified to enhance the learning process. In this process the researcher must decide what key ideas are to be emphasized and must be ready to add to them. This demands sensitive judgment in distinguishing those difficulties a child experiences which are intrinsic to the conceptual material from those difficulties which

the turtle would trace out a square of side length 50. You would, too, if you carried out those commands. Thus "playing turtle" follows from this. Being a turtle is a powerful heuristic and debugging principle and to put it into practice, children are encouraged to walk in a square, observe their own actions, and translate them into turtle commands.

Playing turtle is also useful in encouraging children to "work through" puzzlement or "cognitive dissonance". For example, the set of commands previously given will cause the turtle to make a square no matter where it is positioned or headed. Since the child might see the figure as a diamond or a "skewed square" playing with the procedure creates interesting and provocative situations.

Once the child knows how to describe a square to the turtle, he must give the process a name and link the name and the instructions together. Currently, if the child told the computer to

```
SQUARE
```

the computer would respond

```
I DON'T KNOW HOW TO SQUARE
```

The "standard LOGO" formalism for remedying this to to define a new procedure by typing

```
TO SQUARE
10 FD 50
20 RT 90 etc.
```

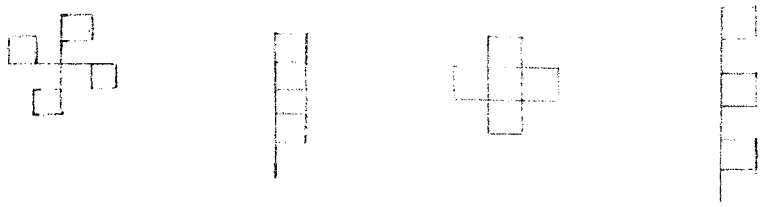
In my work at MIT with elementary school children I noticed that this process compounded two difficulties: (1) the conceptual difficulty inherent in the idea of defining a procedure; and (2) the accidental difficulty of remembering how to do this in LOGO. I introduced the idea of an interactive computer aid for this purpose. The aid is invoked by typing the single word TEACH. It then prompts the child who can, so to speak, "teach" the computer through the following transaction. I underline what the computer types:

```
TEACH
TEACH ME TO SQUARE
STEP 1:  FD 50
STEP 2:  RT 90
      :
      :
STEP 9: END
NOW I KNOW HOW TO SQUARE
```

Now the turtle can make a square and the computer understands the word SQUARE, the child can use it to create a new design where SQUARE is used as another LOGO word:

```
TO FLAG
1  FORWARD 50
2  SQUARE
END
```

And now FLAG can be used to create designs:



As an extension of subprocedurizing, children are introduced to recursion. For example:

```
TO MANY-FLAGS
1  FLAG
2  RIGHT 10
3  MANY-FLAGS
END
```

To understand such a process, we ask children to play a "people procedure game". For example: when I say WOW, raise you hand and then lower it. Now I will say WOW several times. The next step is to change WOW, add a command: this time raise your hand, lower it and tell yourself out loud to WOW.

We play this game for a while and then go back to turtle procedures and apply the same technique to the turtle.

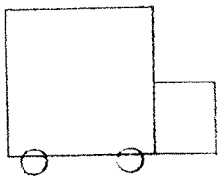
At some point we extend "people procedures" to serve as models for non-turtle activities, "bridge activities" like learning to walk on stilts or juggle or solve puzzles, where we develop procedures, execute them, debug them, and refine them to fit individual learning styles.

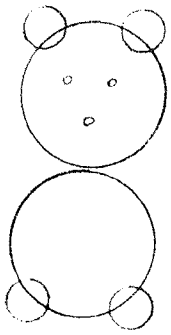
Individual Styles of Learning and Teaching Strategies

In preliminary work, I have observed that different children take over the computer in different ways. They show different learning styles, different paths into the computer work. Undoubtedly this bare statement is true for all learning; what is special here is that the plasticity of the computer allows the process to go further and become more explicit. In working with computers there really are many paths to the same goal. Moreover, there are many equally great goals to pursue. Thus, children really do have to express and explore their own intellectual styles.

Although each child has a unique intellectual personality and the use of the computer allows us to respect it, we do, nevertheless, observe some regularities. I shall describe three learning styles which have emerged particularly clearly not only from my own work with young children, but from work recently completed at the MIT-Brookline LOGO project by D. Watt in his teaching of 8 sixth graders over a six week period.

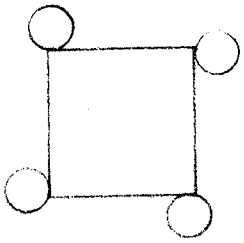
Style 1: This child is a planner. He works from a complete formulation. For example, he will design and implement a truck or a bear:



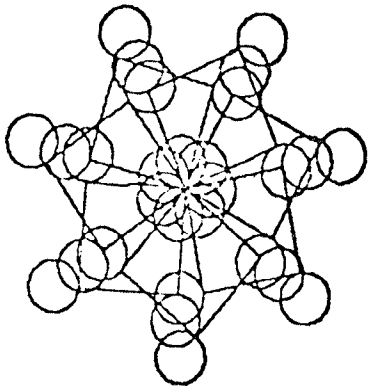


Style 2: This child uses building blocks, subprocedures, and experiments with their possibilities. He arrives at some goal which is not predefined through a series of trial and error steps. For example:

from



to



Style 3: It is, perhaps, most difficult to develop teaching strategies for this child since he defines his own goals which he will not verbalize. What he is exploring and how he does it can easily be misinterpreted. His activities often look like turtle scribblings. He may "revert back" to changing the turtle's state by tiny increments or he may use the same increments to all turtle commands (like FORWARD and LEFT) repeatedly.

The teaching methodology I have developed is based on a model of a child who, in the

But in the same initial session I suggest some concrete goal like: make the turtle walk in a square or, perhaps, having placed some "squares" on the screen or blocks on the floor, I ask the child to make the turtle touch them (knock the tower down, etc.). In this I elicit primarily style 3 with some hint at style 1.

I facilitate style 2 by seizing on something interesting the child has just done and suggesting "teaching" it to the computer. Thus I encourage the child to procedurize, and thereby turn the turtle meanderings into repeatable patterns, procedures, building blocks, and then use these procedures as subprocedures to create unanticipated designs.

The beginning student would very quickly be asked to choose a design from a collection built from a subprocedure familiar to the child or create his own design, and then develop procedures for getting the turtle to make the design. In this way children are exposed to style 1.

I can illustrate both the pervasiveness of these styles and the way in which I work with physical skills as bridge activities by the following anecdote in which we see the same styles in two different domains. Mar and Sco, third grade children from the Roberts School in Cambridge, Massachusetts, were learning to walk on stilts at MIT. Mar had been very resistent to procedural thinking in his computing activities and now when he was learning to walk on stilts he again refused to procedurize. He just wanted to get up and get there and so tried to apply brute-force techniques. Sco, on the other hand, was eager to use procedures in both cases. The result was: Mar, who prided himself on his physical dexterity, was very much surprised when Sco, who was not so "coordinated", learned to walk on stilts very quickly and very well. A side note on Sco: Although he appreciated procedural thinking, he resisted global planning, of developing procedures to accomplish a predetermined goal, until this experience. He was no less surprised than Mar at his "victory" in the race to learn to walk on stilts, and carried the fruits of his triumph for a long time.

References

Brown, John Seely and Richard Burton, Diagnostic Models for Procedural Bugs in Basic Mathematics, BBN Rept. #3669, ICAI Rept. #8, Bolt, Beranek and Newman, Cambridge, MA December 1977.

Davis, Robert B., "Selecting Mini-Procedures: The Conceptualization of Errors in Thinking about Mathematics", J. of Children's Mathematical Behavior, Supplement No. 1, Summer, 1976.

Goldberg, Adele and Alan Kay, Teaching Smalltalk, Xerox, Palo Alto Res. Center, SSL 77-2, Palo Alto, Calif., June 1977.

Inhelder, Barbel, Hermine Sinclair and Magali Bovet, Learning and the Development of Cognition, Harvard Univ. Press, 1974.

Papert, Seymour, "Teaching Children Thinking", Mathematics Teaching, no. 58, Spring 1972.

Papert, Seymour and Cynthia Solomon, "Twenty Things to Do with a Computer", Educational Technology, XII, 4, April 1972.

Papert, Seymour, Uses of Technology to Enhance Education, LOGO Memo #8, Mass. Institute of Technology, Artificial Intelligence Laboratory, Cambridge, MA, June 1973.

Solomon, Cynthia J., Problem-Solving in an Anthropomorphic Computer Culture, A.M. Thesis, Boston University, May 1976.

LOGO Reports Available

The following reports are available in German from the LOGO research group in Darmstadt, Germany. The work was done during the years 1974 through 1978. Please write them at:

Forschungsgruppe CUU
Projekt PROKOP
Frankfurter Str. 24
6100 Darmstadt
West Germany

(1) Kling, U., et.al.: Computer as a tool in processes of active learning, Progress Report. Describes work during the first research period from 1974 through 1976.

(2) Fischer, G.: The solution of complex problems by naive users through interactive programming. 1977 Approx. 250 pages. This thesis defines an integrated view of computer science for naive users which emphasizes problem solvings, model building and learning to learn. Part one analyses the role of artificial intelligence and computer science in educational applications and the case studies of part two illustrate the derival basic concepts of programming and problem solving.

(3) Fischer, G.: Problem solving with the computer: Vol. 1. Introduction to interactive programming, 1975. This collection of lecture notes and handouts was prepared for a course about programming and problem solving with the computer; it has been extended and may be used as a workbook to teach an introductory course in a LOGO-like environment.

(4) Boecker, H. -D. & G. Fischer: Problem solving with the computer: Vol. 2: Problems 1978, approx. 500 pages. This report consists of five parts and presents detailed case studies of complex problem solving with the computer in the fields of mathematics, linguistics, computer science, artificial intelligence and gaming. They are based upon the theoretical work described in part one of (2) and represent a collection of ideas and programming projects.

(5) Boecker, H.-D: LOGO-Manual, 1977, approx. 120 pages.

(6) Laurenze, A., U. Kling: Report on experimental LOGO course with 11 to 13 year old kid. 1976. Approx. 80 pages. Course material, documentation and evaluation.