MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

# LOGO BOOK DRAFT

Seymour Papert
Professor of Applied Mathematics

November 2, 1970

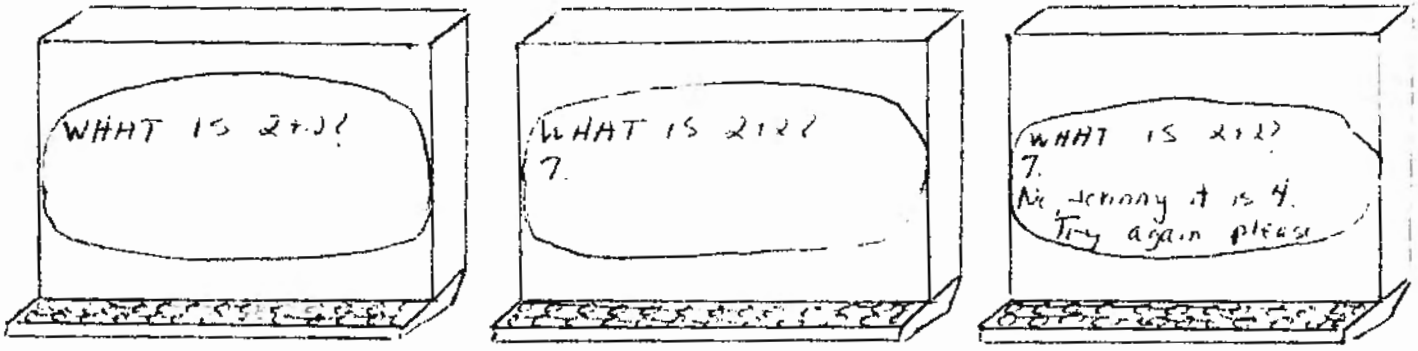
This paper is dedicated to the hope

that someone with power to act will one day see that contemporary research on education is like the following experiment by a nineteenth century engineer who worked to demonstrate that engines were better than horses. This he did by hitching a 1/8 HP motor in parallel with his team of four strong stallions. After a year of statistical research he announced a significant difference. However, it was generally thought that there was a Hawthorne effect on the horses.

LOGO DRAFT
October 31, 1970

## INTRODUCTION 1

The most pointed complaint about the use of technology in education is that it so often reduces to inventing clever new gadgets to teach the same old stuff in a thinly disguised version of the same old way. When the technology uses computers, the complaint is aggravated by apparently high cost and fears of "dehumanization". People who bring these charges are usually thinking of something like the teaching machine represented in the figure as a model of how computers would be used in education.




The frames show successive stages in an interaction between an invisible computer and an invisible child via a device known as a Display Console. Its essential visible parts are a TV tube and a keyboard. It is connected to a computer which can cause words, pictures or what you will to appear on the tube.

the specific content it is useful to say something about the level of aspiration.

Behind much of the thinking described below is an attempt at formulating an experiment to probe the limits of the educational capacities of children- It is crucial to our purpose to realise thatthere is almost no theoretical or experimental knowledge with a direct bearing on the simplest questions one might ask about the inherent learning capacity of chilren or about the relative degrees of difficulty of different subject matters. Take mathematics as an example, though any other would do as well. Ask the psychologists, the teachers or whoever you will, whether they think it would be possible so to construct an educational system that 99% of the present first grade children would become proficient in mathematics at the level necessary, say, to major in it at a good college, or even at the level necessary to do significant research in topology. Of course, it is irrelevant to discuss the desirability of creating a few million topologists. Is it possible?

Now, I maintain that no one is in a position to venture any responsible guess! One can certainly point to the fact that very few actually achieve true competence even at the level of elementary school math. But it would be the crassest empiricism to deduce from this that mathematics is intrinsicially difficult, or that most children are not "mathematically minded." One might as well deduce that French is intrinsically difficult and that most children are not "Frenchly minded" from the poor competence achieved by most children in school French. Yet we know that, had the

talent and access to material resources judge that the thesis is plausible enough to make it worth pursuing with the kind of dedication given by the pioneers of aviation (or any other new field) long before this dedication produced proof that they were right.

The style of writing and the selection of material and examples is personal. But I believe that I am writing as spokesman for a community of people who certainly will not agree with many details but who share the fundamental ideas expresssed through these details. A hard decision was whether to discuss a large number of papers, experiments, data and so on. This might have given a more "scholarly" appearance. But it would have diluted the vision in the way that "averaging" good things does not ever make better. So most of the examples and details are taken from the work I know best because I am deeply involved in it.

The general plan is based on the following classification of computers [To be elaborated for the November 18 version]:

(1) The children themselves are able to use the computer as computer scientists in their own right.

(2) Computers programmed by others as Teaching Machines, Editing Systems, Simulators etc. etc.

(3) The theory of computation as a source of ideas, concepts and theories for children and about children.

# CHAPTER 2

## GLIMPSES OF MATHLAND

To ask whether there could be an IQ-land is a far cry from finding one, and is scarcely very original. Many educational reformers have dreamt the dream and their failure must be taken by any hard-headed observer as at least indirect evidence against the reality of the vision. But I shall argue that we have the technical means to carry out what the earlier reformers only wished they could carry out. We can actually lay down a first sketch of how to construct a real mathland. Indeed some pieces of it are already built and occupied by children. Naturally our first sketch will not be thefinal and decisive one; I claim only that it shows we can cross a certain barrier whose importance has been widely recognized by many clear minds devoted to education. To define the barrier and our relationship to our predecessors let's hear for a moment the voice of John Dewey.

### Digression: A Word from Dewey

For Dewey the dilemma of schools is expressed in the following passages contrasting schooling with the way he supposes children learn in less developed societies:

> For the most part, they depend on the children learning customs by sharing in what the elders are doing...To savages it would seem preposterous to seek out a place where nothing but learning was going on in order that one might learn.

better. I suggest that the difficulty is really not that it is badly taught, but that it is fundamentally unteachable. I do not mean to deny the obvious fact that heroic measures of superlative teaching will get somewhat better results. What I mean to suggest is that this scholastic math (new or old!) is not a natural route into mathematical thinking! A natural route would probably be more like the participation in Dewey's hunt. I believe (following Piaget) that all children acquire a great deal of mathematical knowledge through activities of this that lie outside the classroom. I believe that some children go far enough along such routes to cross a threshold of mathematical comprehension which enables them to work in the academic context (until, as often happens, they are turned off by it!).

To give substance to this series of expressions of personal believe I shall describe another context for mathematical work. When we have it as a concrete example we can come back to the general issues.

A Fifth Grade Child and his "Turtle" on the Fringes of Mathland

A LOGO Turtle

As a first groping step to building up an image of Mathland let's pursue the analogy with learning French in France. This idea leads by an easy step to the fantasy of a companion who spoke only mathematics, and

thereafter the turtle will leave a line wherever it goes, until this command is countermanded by

PENUP.

Thus the series of commands:

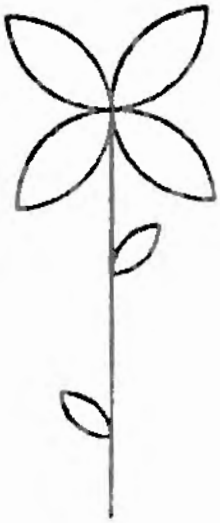
```
PENDOWN
FORWARD 200
PENUP
FORWARD 100
PENDOWN
FORWARD 200
```

will cause the turtle to draw the perfectly dull pattern:





To make the turtle do anything more interesting one has to learn how to use a programming language in order to instruct the computer how to combine the elementary turtle commands to produce complex patterns. The flowers and stick men are examples



| | |
|---|---|
| TO MARCH | The program defines a new command: MARCH. It does it by giving directions saying how to march. |
| 1 TEST TOUCH | These say do you feel anything? |
| 2 IFTRUE BACK 10 | if you did, back off a little |
| 3 IFTRUE RIGHT 180 | if you did, turn around |
| 4 FORWARD 10 | whether you did or not, go forward. |
| 5 MARCH | And this is perhaps the tricky one It says: when you have marched, march again. This makes marching a self-perpetuating process! |

Notice how even this very simple program requires a little observation of behavior patterns so that one is almost tempted to see it as a model made by the child of a component of his own behavior. The temptation becomes much stronger when the children give their turtles more complex behavior using the principle of feed-back to program the turtle to avoid obstacles, to search a room until it finds its "nest", to exhibit tropisms, to play tag with children or with other turtles and so on without bound.

Examples of this kind of work are developed in more detail in Chapter ____.

I believe the term Turtle for
members of this species was first used
by Gray Walter who constructed some
influential turtle-like cybernetic machines

level that is unlikely in an elementary school and, moreover, boring in execution. Even more serious is that such devices do nothing until they work properly. The situation is sharply different when we consider "information machines" in place of "matter machines" or "energy machines." Suddenly there is the prospect of systems with great and interesting complexity, made of parts each of which functions interestingly by itself and, above all, the coupling between the parts does not require mechanical precision or understanding obscure conditions for "matching." In the extreme case there is nothing but "information": the system and its parts are all programs in a large powerful computer. Even when working with turtles brings in physical manipulations and even putting together physical parts such as attaching a new sense organ it remains basically true that the interface between the parts is informational.

The important point is that developing programs can provide even very young children with an experience very much more like that of a research project than is anything now available in the schools. This is not meant to suggest that "programming" is in itself the key to a new education. Writing a "machine language" program is "binary" to add a series of numbers is as abstract and, to most children, almost as boring as learning long division. Good programming languages lead to better results and are essential to many of the projects we shall discuss. But programming is a mere vehicle (the tool, the raw material) for what we have in mind. The educational experience is a project involving building a system of programs of a particular kind -- we shall later on develop

pen to draw the numbers?; time's up;
disconnect everything; put it away...

By contrast...writing a program to do
the job needs no special knowledge
and no special devices. The partial
or complete program is always there
without fuss.

```
TO  RTIME
PRINT  "WHEN YOU SEE A STATEMENT APPEAR
       TYPE 'M' IF IT IS TRUE 'N' IF IT
       IS FALSE
       BE AS QUICK AS POSSIBLE"

PRINT  "IT IS FALSE THAT 17 IS NOT A PRIME NUMBER"
MAKE
   NAME "START"
   THING CLOCK
REQUEST
MAKE
   NAME "END"
   THING CLOCK
PRINT (SENTENCE "THAT WAS":END-:START "SECONDS")
END
```

A LOGO Program for Another
Cognitive Reaction Time Experiment

# Teaching Thinking & Learning

This chapter is about trying to teach children how to think and how to learn. Some of the ideas in it have no logical connection with computation. The idea of heuristic is used in very much the same way as in Polya's books. But I maintain that programming provides by far the best examples on which students -- particularly elementary school children -- can work to develop a sense of heuristic problem solving. So, though there is no logical connection in this case there is a strong heuristic one.

Other ideas are more obviously related to computation. In particular we shall discuss how programs can be used as models of one's own thinking. This will bring us into direct contact with an important set of ideas from the branch of computation known as Artificial Intelligence.

The first sections of the chapter deal with some very general questions about education.

## The Don't-Think-About-Thinking Paradox

It is usually considered good practice to give people instruction in their occupational activities. Now, the occupational activities of children are learning, thinking, playing and the like. Yet, we tell them nothing about those things. Instead, we tell them about numbers, grammar and the French revolution; somehow hoping that from this disorder the really important things will emerge all by themselves. And they sometimes do. But the alienation-dropout-drug complex is certainly not less frequent.

In this respect it is not a relevant innovation to teach children

## The Pop-Ed Culture

One reads in Piaget's books about children re-inventing a kind of Democritean atomic theory to reconcile the dissapearance of the dissolving sugar with their belief in the conservation of matter. They believe that vision is made possible by streams of particles sent out like machine gun bullets from the eyes and even, at a younger age, that the trees make the wind by flapping their branches. It is criminal to react (as some do) to Piaget's findings by proposing to teach the children "the truth." For they surely gain more in their intellectual growth by the act of inventing a theory than they can possibly lose by believing, for a while, whatever theory they invent. Since they are not in the business of making the weather, there is no reason for concern about their meteorological unorthodoxy. But they are in the business of making minds -- notably their own -- and we should consequently pay particular attention to their opinions about how minds work and grow.

There exists amongst children, and in the culture at large, a set of popular ideas about education and the mind. These seem to be sufficiently widespread, uniform and dangerous to deserve a name, and I propose "The Pop-Ed Culture." The following examples of Pop-Ed are taken from real children. My samples are too small for me to guess at their prevalence. But I am sure very similar trends must exist very widely and that identifying and finding methods to neutralize the effects of Pop-Ed culture will become one of the central themes of research on education.

as not having the required aptitude, rather than by diagnosing the specific deficiency of knowledge or skill.

(d) Caterpillar Theories. The theory that explicit formulation harms performance seems to be itself a central part of Pop-Ed. It is akin to the Blank Mind Theories and is felt by children to be confirmed by examples of skills they can perform although they cannot (in their opinion, and probably in reality) describe them.

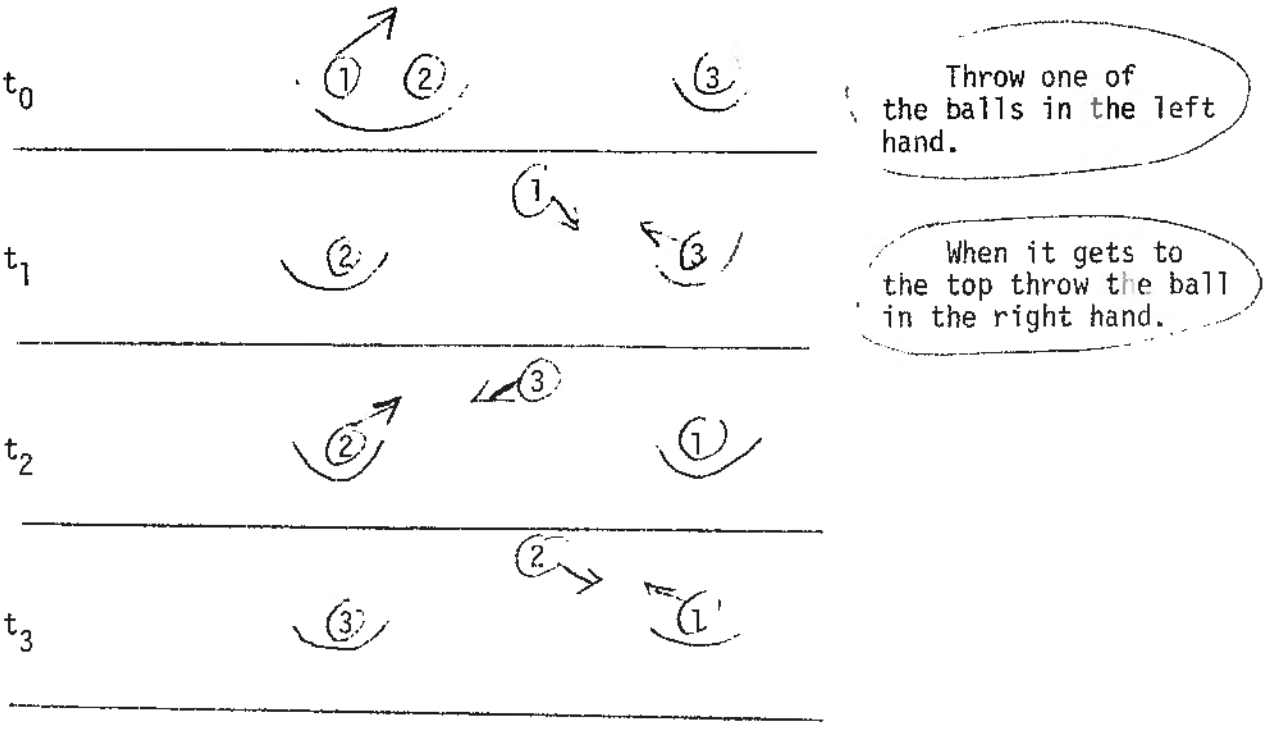
The Computer vs. Pop-Ed

I do not believe that much good can come out of merely preaching to the children about these bad ideas. Such a widespread set of beliefs is bound to reflect deep aspects of the children's intellectual structures. In particular these are largely determined by the poverty of our culture in means of conceptualizing and discussing complex processes of any sort, and mental ones in particular. This must be remedied by exposing the children to more than verbal contact with the new ideas we wish to give them. We must find ways to provide a set of activities through which children can develop an easy fluency in the new concepts and integrate them into widespread structures of concept and intuition and attitude. I see the primary use of computation in this process as providing suitable activities. Secondarily, it will provide children with explicit models of some cognitive processes.

> In a good Piagetian sense I'd rather say: you cannot give someone an idea; education consists of creating the conditions for him to invent it himself.

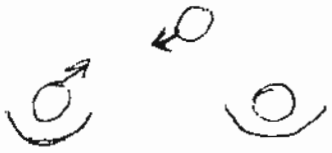
I shall develop the point by taking a real example of a physical activity, juggling, which, in many ways, is not altogether unrelated to the walking problem for millipedes.

The following is an incident from a juggling lesson. I am the teacher. The pupil is a seventh grade girl beginning her third lesson. We are working on a three ball exercise which can be described as follows: The little pictures each show two hands and three balls. The balls are numbered (1) (2) (3) in the diagram to enable us to talk about them. The diagram presupposes a regular time beat, with pulses at, $t_0$, $t_1$, $t_2$, $t_3$.... Little arrows attached to a ball show its direction of flight if it's in the air, and indicate that it is being thrown if it is in a hand. It is understood that "throw" always means, "throw over to the other hand".




My pupil watched and said: "You throw 1, then 3, then catch 1, throw 2,

back to enjoy the problem of describing the juggling sequence. Our usual heuristics advises us to make a clear decision about whether we are setting up the recursive part of the procedure, or a beginning or end part. The recursive part is the interesting one here. We are then told to identify and name the action on the problem scene: two hands and three balls. Let's call them "LEFT", "RIGHT", "B1", "B2", "B3". Now let's look at a round:



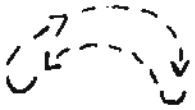
The ACTION part of our procedure will say:

THROW :LEFT AND CATCH :RIGHT

Do we need to give throw a second input saying which ball to throw? This is a delicate point. A mistake would land us back in the complexity of describing sequences of ball names. But we have good heuristics that boil down to saying: there is a choice of hands, so this must be an input; there is no choice of balls so the ball need not be an input. Indeed we need never mention the ball by name.

Pushing the same kind of thinking a step further we see that we don't need to know if a hand is left or right, but only if it is the hand towards which a ball is flying. And this last thought leads to a different, and different kind, of naming. Instead of the names "LEFT" and "RIGHT" being

But though tolerable, there is a definite difference between these procedures - a difference that will seem subtle and perhaps not verbalizable to someone who does not have a conceptual framework to express it. This difference is what a musician would call the meter - and I believe that music teachers complain bitterly about the difficulty expreienced with this notion by many music students. The difference between two throws to the bar and one throw to the bar might not be visible in a student's juggling. But it would hit him like a sledgehammer if we asked him to introduce a difference between the paths of the balls, say like:



If he doesn't see the meter it is just very much harder to modify his patterns. Of course one can represent the meter in many differnet frameworks: programmers, musicians and jugglers all have their somewhat different ways to think of it. But the structure is surprisingly the same and the same as what we teach in computer classes as methods of handling recursive problems whether or not they represent a process in physical time. They surely all at least have these features in common:

> Recursive thinking i.e. they do not represent the process in time since the beginning but rather the generative rule.
>
> Division into rounds, bars, cycles or whatever the particular specialist calls his metric subdivisions in a way that neglects the identity of the balls i.e. noone sees it as 6 throw to the bar. (which my pupil would have done had her memory held out!)
>
> Proper identity of the elements set in the program by naming, in music by a stress or some other identifier of a reference point in the bar. A count, at any rate , amongst beginners at juggling.

So 35 + 35 should give us 6 10 and what's wrong with that? Whose fault is it that the idiotic decimal place rotation won't allow us to write 6 10 without confusing it with 610?

Perhaps my general thesis can be stated crudely as: if you can't think things like that previous paragraph, you shouldn't be doing that kind of arithemtic; if you can think such things effortlessly, you will find the arithmetic equally effortless. The ecuational problem (for arithmetic) is about helping children attain that kind of sophistication. The problem is not to get them to understand numbers as such (if there are such things as numbers as such). It is rather to develop this ability to think about people doing mathematics and especially about themselves. The components of this sophistication are in turn things like:

The idea of a procedure and, especially, the idea of following the procedure literally.

> This distinction is crucial. Most children will say that they do use the little procedure for adding digit by digit. By using a procedure they mean using it with the appropriate additions of good sense. Consequently when they get into trouble by using it literally, they are unable to account for their difficulty by blaming the bugs in the procedure.

The idea of representation and especially feeling free (and being able) to make up a representation when one wants one with some special property.

Debugging Programs as a Culture Medicine for Growing the Attitude of Objectivity

A program that fails to do what was intended nevertheless does something. One can study its behavior, and understand exactly why it does what it does. The term "bug" (as opposed to the morally charged "error") and especially the use of an active verb "to debug" reflect the elements of objectivity in this.

We observe in children the sequence of behaviors:

Initial: reluctance to put a procedure to the test. The attitude is leave well alone once it works for the input or conditions the child used to develop it.

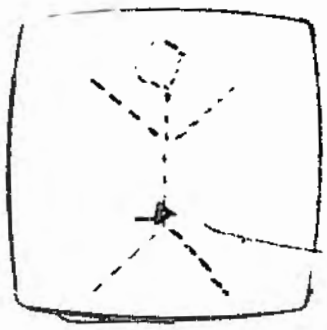
Intermediate: Typical incident -- a program to "reverse" a word gave correct results for 8 out of 10 words and was declared to be a great success with a grade of 80%.

Winning: Treating the cases where it behaves deviantly as the most interesting. Why does the procedure think that

REVERSE "EAGLE" = "EAGLE"

when it gave REVERSE"CAT" = "TAC", REVERSE "LION" = "NEIL" and so on for ELEPHANT, COW, HORSE, DOG and many others.

With these sub-procedures we easily construct the man. Note where the turtle starts and ends.



Dotted figure shows where man will be.

Turtle starts here

```
   TO MAN :SIZE                          Comments
1  LEFT 90                               Turtle faces upward
2  FORWARD :SIZE                         Draws body line
3  VEE :SIZE                             Draws arms
4  FLAG :SIZE /3                         Head and neck, the length
                                         of the neck is a third of the body
                                         length.
5  BACK :SIZE                            Retraces body. We could PENDOWN,
6  LEFT 180                              PENUP but it hardly seems worth the trouble.
7  VEE :SIZE                             The procedure VEE expects the turtle
8  LEFT 90                               to face "into" the V.
END                                      Restoring heading for
                                         tidiness.
```

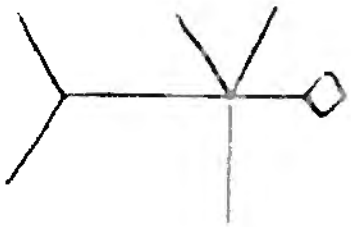
Some Bugs

These display programs provide excellent exercises in debugging. Here are some deformations produced by likely bugs. Try to find what errors in the programs (MAN, VEE, or GROWMAN) might nave produced them:

(a) MAN 50

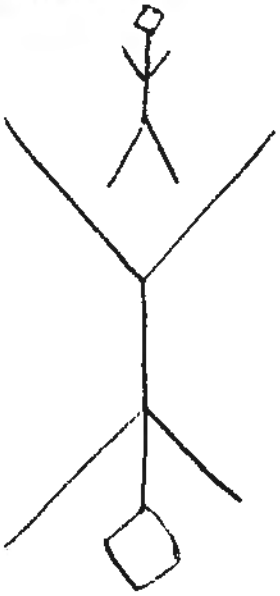
Line 7 of Man missing

(b) MAN 50



Line 7 of VEE has RIGHT in place of LEFT

(c) GROWMAN 50



Line 8 of MAN missing

## Languages and Computers

The next pages will describe the elements of a language, called LOGO, used for instructing computers to carry out actions of various sorts. These actions can be "calculations" in the usual sense, but this is a very special case. In the real world computers fly airplanes, control milling machines, set type and so on ad infinitum. Curiously the educational use of computers has fastened onto verbal exchanges via teletypes and display consoles. The turtles discussed earlier show one way to break this hold. We'll see lots of others.

To see the simplest example of LOGO in operation imagine yourself sitting at a console like that on the first page and typing the command

PRINT 3+4

just as if the console were an ordinary typewriter. If the console is suitably connected to a computer that "understands" LOGO, you will immediately see

7

appear on the screen.

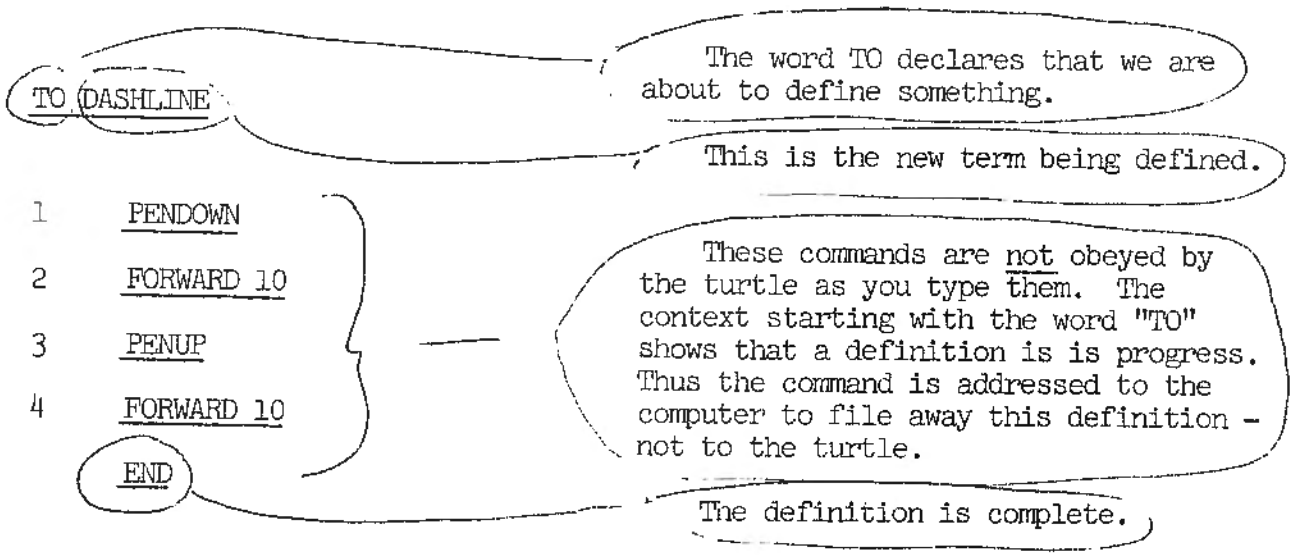
Even in this simple case the computer did more than perform the calculation. It also caused the result to appear on a TV tube! Most of the actions of computers to be discussed here involve producing physical effects of an even less abstract kind than displaying a number on a screen. More typical examples are: causing a turtle to move, causing a music-box to emit a sound and drawing a shape on the display tube.

One can learn to produce very complicated actions of this sort without knowing anything about "how computers work." One merely needs to

move in a circle you will see on the screen a message such as:

DASHLINE NEEDS A DEFINITION

So we have a chance to see how you can "tell the computer" what you want DASHLINE to mean --- in other words you will say how to dashline. The LOGO idiom for this is:




If you now type

DASHLINE

the turtle will drop its pen, move forward 10 units, raise its pen, mov another 10 units and stop. If you type

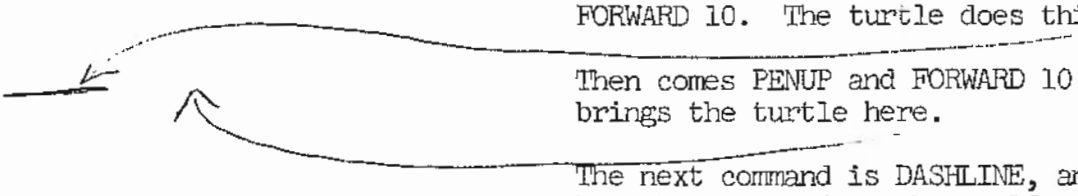
DASHLINE

once more, the same actions will be repeated. Many repetitions of typing DASHLINE when the turtle stops will produce a line like:

Now observe that you acted in a very mechanical way in typing DASHLINE every time the turtle stopped. Almost as if the computer were giving

DASHLINE

This has a definition so the machine reacts by following the definition. This is done by successively carrying out the commands in the definition. The first is PENDOWN. The next is FORWARD 10. The turtle does this.



Then comes PENUP and FORWARD 10 which brings the turtle here.

The next command is DASHLINE, and this has the same effect when self-administered by the computer as when it was administered by us. So exactly the same process is carried out again, except, of course, that the turtle makes the FORWARD 10 movement from where he was left on the previous round.

Another example:

CIRCLE

CIRCLE NEEDS A DEFINITION

TO CIRCLE

FORWARD 5

RIGHT 5

CIRCLE

END

CIRCLE

The effect of this is to cause the turtle to draw a circle. The geometric reason for this should not be obvious to you! From the form of the program you can see that the turtle will go forward a little, turn right a little, go forward a little, turn right a little and go on

definition of DASHLINE is

FORWARD 10 ← This is the input of FORWARD.

and we clearly need to eliminate the "10". But what can we put in its place? Surely not another number! What we need to express is something like:

FORWARD the-number-that-happens-to-be-specified

We don't know what this number will be. But we can talk about it by giving it a name. We could choose the complicated descriptive one with lots of hyphens, or we could choose an arbitrary one, say "JOE". Or we could choose a sensibly mnemonic one like "STEP". Let's do the latter. The definition can then be written:

```
TO DASHLINE :STEP
PENDOWN
FORWARD :STEP
PENUP
FORWARD :STEP
DASHLINE :STEP
END
```

The ":" means something like "whatever happens to be called ___." Thus the line can be read as To dashline a-number-that-is-called "STEP," do as follows. Go forward the-number-of-units-that-is-called "STEP". the same number again and since we want an exact repetition, we say so.

Playing with Procedures

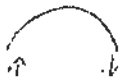
This section will try to give the flavor of how one idea can lead to another in work with the computer. It will develop a series of programs in a more concentrated way than in the previous programs. It may be difficult to follow in detail at a reading pace. It would be much easier if you were developing details of the procedures yourself in interaction with a machine and a turtle. So, if you do not want to work at it, try to get the general feeling by glossing over the details of some procedures --- perhaps the best one to accept or trust is SEGMENT. This command is illustrated by the following examples of its use. The drawings are turtle traces.

SEGMENT 20 180

Notice 2 inputs.

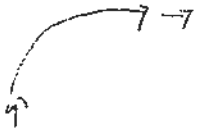
The arrows are not part of the turtle trace. They show the direction it faces at the beginning and end. This would be apparent with a real turtle.

SEGMENT 40 180



This input seen to fix the size.

SEGMENT 40 90



So, the second input specifies a part of a circle in degrees while the first specifies the size.

```
TO PETAL   :SIZE
1     SEGMENT   :SIZE  90
2     RIGHT  90
3     SEGMENT   :SIZE  90
END
```

The command

PETAL 10

causes the turtle to draw

The ways the turtle faces at the beginning and end will be exploited by the next procedure. But in the chapter on bugs it will be seen that this is untidy and dangerous.

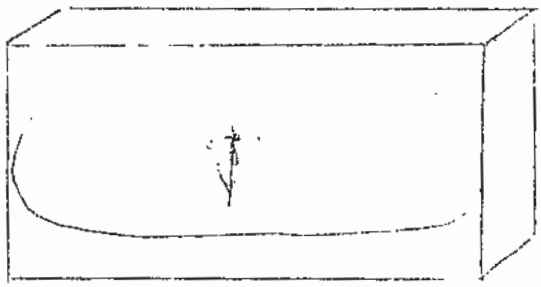
```
TO FLOWER   :PSIZE   :STEP
1     FORWARD   :STEP
2     PETAL     :PSIZE
3     PETAL     :PSIZE
4     PETAL     :PSIZE
5     PETAL     :PSIZE
6     BACK   :STEP
END
```

FLOWER 2 50

FLOWER 4 10

Display Turtles and Movie Making

So far we have talked about using a real solid, physical turtle to draw real lines on a real floor. Some new dimensions of activity are opened (and some closed) by replacing the real turtle by a "simulated turtle," which appears on a display tube, moves about the tube face obeying the same commands as the real turtle. So our program FLOWER can be used just as it stands to draw a flower on the display.



New prospects come from quickness of action and from commands like:

ERASE

which are difficult to realize in the physical case. A useful modification, also meaningless for the physical turtle is to give the command PENDOWN on input. The effect of PENDOWN :TIME is as if the turtle wrote with magic ink that fades away after a while--in fact, after :TIME time units. Thus we can make a charming movie of growing flowers.

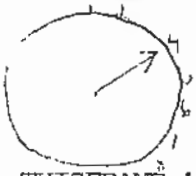
Let's start by making the successive frames by hand by typing:

```
PLANT 5
ERASE
PLANT 10
ERASE
PLANT 30        etc.
```

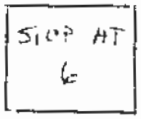
So the little movie program is

```
TO MOVIE   :THISFRAME   :ENDFRAME
IF   :THISFRAME = :ENDFRAME  STOP
PLANT   :THISFRAME  * 10
MOVIE   :THISFRAME + 1   :ENDFRAME
END
```

Another Picture of the Inputs



:THISFRAME is like a moving clock. It ticks up one after each frame.




:ENDFRAME is like a time posted up at the beginning of the show to tell the projectionist when to stop.

Comments on the Inputs:

We think of a movie as a process. As it goes on we need to know two things: where we are and where we are going. The two inputs are set up for this. The first is :THISFRAME. It starts at 1 and increases by 1 on each round. It is the frame number. The second input remains constant during the showing of the movie.

Computer Psychology: Developing a Game-Playing Program

This program to play a simple game in the NIM family brings the LOGO class into contact with another dimension of planning. Projects like making the flower movie could grow from the bottom up. Even if the goal of making a movie was in view at the time the PETAL was written, there was no need to coordinate the parts very tightly. The NIM program will use much more systematic planning and so serve as a model for organization.

The program has the important feature of lending itself to the use of anthropomorphic mentalist language. One cannot really resist saying "it's dumb", "it's getting smarter". Certainly the children talk so and thereby open up discussion on the broader analogies between intelligence and programs, teaching and programming, debugging a program and working on the weaker points in ones own abilities.

In working with the children we try to present the problem so as to create as many opportunities as possible for subdivision into meaningful partial solutions. The key idea for subdivision of the problem is to write a series of programs, each of which is "smarter" than the previous one. The first program will know nothing about the strategy of play. It will not generate moves, but ask each of two human players in turn what move to make. For example, it might act as a score-keeper, just keeping track of the number of sticks without bothering about whether the move is legal. From score-keeper the machine could advance to referee. This means that it checks the moves for legality and eventually declares the game over and announces the winner. After we have a working mechanical referee we will start making a mechanical player. The first version of a player will choose legal, but not necessarily good moves. Indeed, it will generate a move randomly, use its ability as a referee to decide if it is legal, and then accept it or generate another random move.

variations so that no child's final program is a mere subset of a more advanced one. The teacher's computer culture can be very relevant in this delicate kind of situation. Although the richness of programming permits children to generate many fertile ideas, sensitive filtering by the teacher can enormously improve the achievement-to-frustration ratio.

> Examples of individual frills to a referee program: timing moves, declaring the winner a move or two ahead(!), allowing a player to take a move back, printing a score sheet, giving advice (!), allowing the players to be at two teletypes (if the system permits), establishing and imposing handicaps (!), changing the rules, etc., etc.

## 2.0 The Rules

A move consists of taking one, two or three match-sticks from a given pile. Two players move alternately. The player who takes the last stick wins.

## 3.0. First Steps with the Children

The first step is to see that everyone knows the rules and understands what the first program will do; for example, by imitating its function or by writing imaginary scripts. In the course of discussing this we would introduce some names (so as to be able to talk about what we are doing!).

## 4.0. A Simple Score-keeper

If this is the first game-playing program, we might give the class an almost ready-made procedure. We build up to it by asking some standard questions:

What shall we call the procedure? (Let's say "NIMPLAY")

What must NIMPLAY do?

What must NIMPLAY know?

Possible answers are:

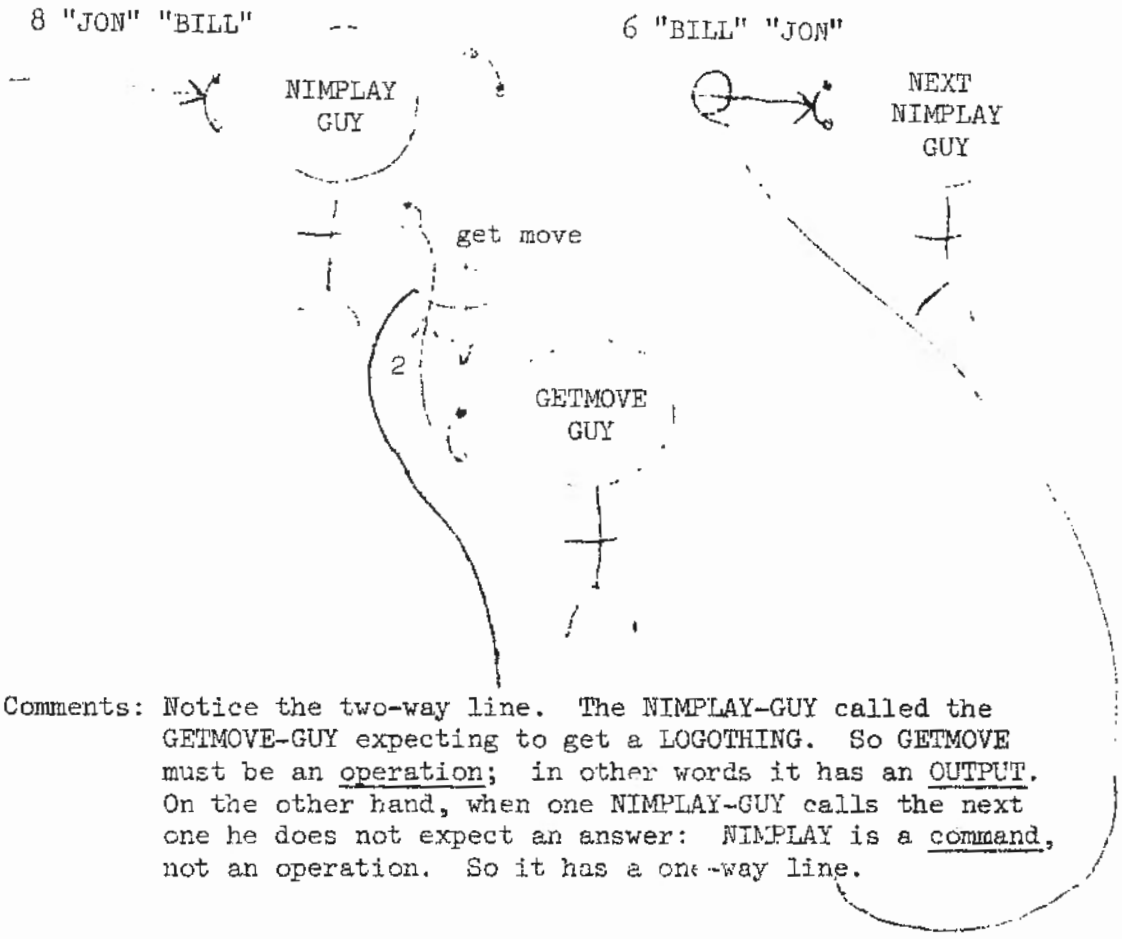
1. Announce the remaining number of sticks
2. Announce the player to move
3. Get his move and make all the modifications
4. Recur.

To do this NIMPLAY must remember :STICKS , :PLAYER , and :OPPONENT from the previous round and get :MOVE by asking for it. The first three THINGS must be told by one NIMPLAY-GUY* to another, so they should be inputs. On the other hand, :MOVE comes from the human player, so it can be gotten by REQUEST and need not be an input. If one looks ahead one might notice that later on, :MOVE will sometimes come from a procedur

---

*The anthropomorphic metaphor is related to the little-men pictures in an earlier section. The use of the anthropomorphic language might be a little precious, but the concept of a separate agent for each program-call is enormously valuable. The children did not seem to resent terms like "MAN" or "GUY".

A little-man picture of a round:




Comments: Notice the two-way line. The NIMPLAY-GUY called the GETMOVE-GUY expecting to get a LOGOTHING. So GETMOVE must be an operation; in other words it has an OUTPUT. On the other hand, when one NIMPLAY-GUY calls the next one he does not expect an answer: NIMPLAY is a command, not an operation. So it has a one-way line.

## 6.0. The Simplest Mechanical Player

How can the machine choose a move? The simplest way is by using RANDOM. For example, we could allow GETMOVE* the choice: if a person is to play use REQUEST, if the machine is to play use RANDOM. But it has to be told whether the player is human or the computer. So it must have an input.

```
TO GETMOVE :PLAYER
TEST IS :PLAYER  "COMPUTER"
IFTRUE  MAKE
          NAME   "MOVE"
          THING   RANDOM
IFFALSE  PRINT "YOU MAY TAKE 1, 2, OR 3 STICKS"
IFFALSE  MAKE
           NAME:  "MOVE"
           THING:  REQUEST
         .
         .
         .       (as before)
```

At this stage the SLIP-BY bug might become serious. One way to to kill it is to tell GETMOVE about :STICKS and have it try-again if :MOVE comes up greater than :STICKS . To do this we change the title line to:

```
TO GETMOVE :PLAYER  :STICKS
```

and add a pair of lines (in the TRY-AGAIN form) after the two MAKEs.

```
TEST GREATERP :MOVE  :STICKS
IFTRUE  OUTPUT GETMOVE :PLAYER  :STICKS
```

* Notice this anthropomorphism. We find it useful to talk of procedures as agents, of their "state of knowledge," of "telling them" of having them "talk to" one another. But we present this to children as a deliberate metaphor which they might find useful.

Question: What do we test for?

English Answer: Whether there are 1, 2, or 3 sticks.

LOGO Answer: TEST MEMBER :STICKS "1 2 3"

We recall the procedure MEMBER shown by the examples:

MEMBER 6 "1 2 3" = "FALSE"

MEMBER 2 "1 2 3" = "TRUE"

Question: What is the action if the test is passed?

English Answer: Take all the sticks .

LOGO Answer: OUTPUT :STICKS

Question: What if it is not passed?

English Answer: Move just like before.

LOGO Answer: MAKE
NAME "MOVE"
THING RANDOM

Putting this together to make a procedure to make the move:

Question: What must the procedure know?

Answer: :STICKS -- so it needs an input.

Question: Operation or command?

Answer: Operation, because it will give us :MOVE as its output.

```
TO MAKEMOVE :STICKS

TEST MEMBER :STICKS "1 2 3"
IFTRUE  OUTPUT :STICKS
IFFALSE  OUTPUT RANDOM
END
```

MAKEMOVE is an easy name to remember.

The procedure is used in place of RANDOM in GETMOVE. So don't forget to change GETMOVE!

Now extra lines can be added. For example:

```
TEST IS :STICKS "5"

IFTRUE  OUTPUT "1"
```

So there we are! The smart invincible NIMplayer is made by replacing

MAKEMOVE by SMARTMOVE.

```
TO SMARTMOVE :STICKS
MAKE
   NAME:  "REM"
   THING   REM :STICKS  4
TEST IS ;REM  Ø
IFTRUE  OUTPUT 1

IFFALSE  OUTPUT :REM
END
```

This LOGOTHING is the main actor, so name it.

It really doesn't matter in this case.

Other Frills

Vary the range of a legal move, e.g., instead of 1-3, allow 1-5; vary the strategy, make the loser take the last stick; add a pile; etc.

We include a listing of MEMBER and REM, but assume that they were written before the NIM unit.

```
TO MEMBER  :IT  :LIST
10  TEST IS  :IT  :EMPTY
20  IFTRUE OUTPUT "FALSE"
30  TEST IS  :IT  FIRST  :LIST
40  IFTRUE OUTPUT "TRUE"
50  OUTPUT MEMBER  :IT  BUTFIRST  :LIST
END
```

```
TO REM  :N  :D
10  TEST GREATERP  :D  :N
20  IFTRUE OUTPUT  :N
30  OUTPUT REM DIFFERENCE  :N  :D
END
```

# TURTLE BIOLOGY

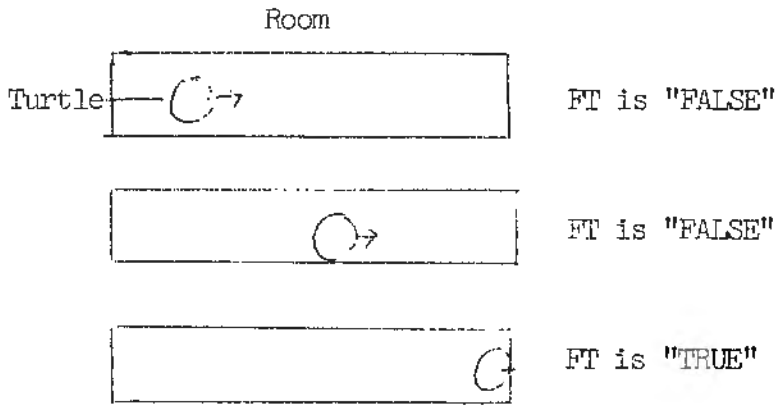
The conceptually deepest advantage of real turtles over display turtles comes out when we exploit their sense organs to produce significant behavior involving interaction with the rest of the world -- objects, people and other turtles. It is easy to provide a turtle with sense organs to allow it to touch, to hear and, in a limited way, to see. Taste and smell are a little exotic in terms of present knowledge but sense of balance based on a principle like the semi-circular canals is easy and instructive.

The examples of work with turtle behavior are selected again on the basis of clarity of principle rather than the spectacular movements.

The use of the touch information in LOGO is illustrated by the following program intended to cause the turtle to walk up to the wall and stop.

```
TO MARCH
1   IF FT STOP
2   FORWARD 5
3   MARCH
```

FT is a predicate, i.e. is either "TRUE" or "FALSE," if FT can be read as "if the Front Touch sensor is activated."



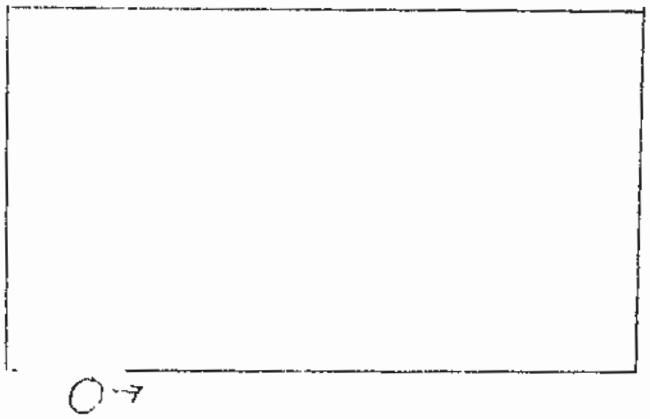

To make the turtle go back again we change MARCH thus:

```
TO MARCH
1   IF FT LEFT 180
2   FORWARD 10
3   MARCH
END
```

To escape from all possible jams we need some more subtle behavior. There is scope for different turtle "personalities". A very methodical turtle might react to any contact by rotating itself until the contact is squarely head-on, and then back away. Of course, even that could lead to trouble in a narrow corridor, where a more appropriate behavior is to rotate until the touch is centered to the left or right.

A different approach is to move randomly when in trouble. This reduces the chances of getting caught in a "trap state"; but it produces complications in carrying out any overall goal since the turtle cannot forget where it is.

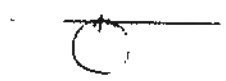
Circumnavigation



The turtle is to go around the large object.

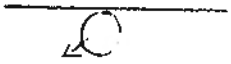
The easiest procedure uses the important concept of Feedback in a very simple way.

In bad situations the turtle does silly things.

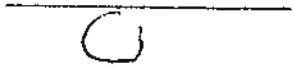


LFT is "FALSE" so the turtle thinks it is away from land and turns right in the hope of getting there.

FORWARD produces no results. The turtle will turn right again since LFT is not coming true.



Eventually the turtle frees itself enough to move away - in the wrong direction.



But it keeps turning right and will go right around a circle and will stay in this trap indefinitely.

One can choose between several courses of action to deal with this problem:

(1) Ignore it on the grounds that the turtle will probably never get into such a position.

(2) Abandon ROUND and construct a more systematic procedure.

(3) Try to debug ROUND, for example, by adding:

```
IF FT  RIGHT 90
```

## Examples of More Advanced Turtle Projects

These examples are mentioned merely to set the reader's mind going.

### Playing Tag

To do this well we need two turtles equipped with some means for directional sensing at a distance. The simplest scheme used light with a directional photosensor attached rigidly to the turtle so that the sensor will see a bright source of light directly in front of the turtle. To find a source of light the turtle would rotate, thus scanning the azimuth for bright sources.

> Slightly more elaborate is a set of 8 sensors arranged like the touch sensors in the previous section. A single rotating tube is not much harder and can scan faster than rotating the turtle, with more resolution than separate fixed sensors. But these engineering details are boringly simple.

The problem is to write programs based on strategies for following and for avoiding the other turtle. The competitive aspect will give additional impetus for some children. As in the case of circumnavigation the simplest strategy is very simple, so something works quickly. And the opportunities for more complex strategies are boundless.

# The Computer as Assistant

The theme of this chapter has a slightly paradoxical overtone illustrated by the following science fiction fantasy. A robot from outer space on a visit to Earth encounters the phenomenon of schools for the first time. On its own planet biological life has vanished long ago, leaving its race of robots with enough intelligence to maintain and advance their level of technological and scientific knowledge. In particular the robots can reproduce themselves in the fullest sense of making new ones complete with all established knowledge and all known skills programmed into them at the time of manufacture. So there is no need for institutions like schools where the individual carries on activity that looks like work, but is done to learn rather than for the product of the work. The robot, which we thereby see to be intelligent but not infallibly so, mistakenly supposes that schools are a kind of sweat-shop in which child labor is exploited to carry out computations. As a parting gift it presents every human child with a miniature computer capable of doing all this work for it. The gift actually was motivated by kindness. Nevertheless some historians maintain that the robot's real intention was to sabotage the educational system so as the inhibit the development of human science to a competitive level.

The question we have to face is whether something like the robot's gift would be sufficiently beneficial to children to warrant our giving it to them without waiting for his visit; or whether it would be so harmful that we must at all cost prevent its development.

the sentence? Unfortunately this is likely to mean re-writing the whole essay. So, he lets it stand! The harm done is worse, much worse, than the presence of a bad sentence in a child's essay. The real harm is the inhibition of the habit of experimental modifications of the text in the spirit of trying several versions to see which pleases most.

Contrast this situation with another, fantastical, one in which the child has a super-secretary who will instantly retype the essay with any changes he wishes. It is admittedly difficult to know what the consequences would be. Surely the child would be more experimental. Surely a better essay would be produced. I conjecture firmly that the child will acquire a deeper and more sensitive feeling for language and for the organization of thinking and exposition. I conjecture a little more tentatively that he would eventually become less "lazy". This last claim will seem perverse to those who think that the supersecretary would "spoil" the child by making him "lazier" than before. Both outcomes are possible in principle. I base mine on the assumption that the child will eventually acquire the taste for literary experiment to a sufficient degree to motivate rewriting the essay ten times by hand if the supersecretary happens to be unavailable. But that is a matter for experiments. The prior question is whether we can give the child a supersecretary.

Well, presumably not a human one, but we can make a machine perform a large part of this function. Indeed, such machines exist and are used to perform just this kind of task for people whose time is recognized as being valuable enough to justify labor saving devices.

It would not be difficult to program the computer to notice unlikely spelling and even some unlikely syntax. Thus if the child types "dificult" the machine would observe that this is not an English word, but is very close to one. So it might indicate this by displaying the word in a brighter form on the console. The child would have the option of asking it to be changed or (as would be appropriate here, since I meant to write "dificult"!) leave it as it stands.

> I am sure many computer experts will exclaim: but that's much too difficult. It is indeed too difficult to achieve right now <u>if</u> we want it to be infallible. But the task is much easier if we think that it would do a lot of good by <u>very often</u> picking on a child's mistakes and <u>only seldom</u> picking on a non-mistake.

An important area to automate is using a dictionary. It really is boring and/or <u>distracting</u> to leave off writing to look up a word in a dictionary —— particularly if the dictionary is large and cumbersome. For the child composing his essay at his console the dictionary reference is very much easier. At his command a summarized dictionary entry appears on the screen. If he wants more information or any part of it he says so and that part is elaborated.

And as an extra bonus the machine can print out for him at the end of the day or week a list of all the spelling mistakes he corrected, the dictionary entries he referenced, etc. If he or his teacher happens to like that sort of thing, the machine could even be made to subject him to a "review quiz" on this material.

Seymour Papert
November 1, 1970

# THE COMPUTER AS TEACHING MACHINE

## 1. What's a Teacher For?

We surely need to answer the question before we can engage in sensible discussion about whether we can or cannot or should or should not automate all or some of the teacher's work. But the sad truth is that there is very little firm knowledge about the role (or rather, the multitude of roles) of teachers in learning.

The crudest models of the teacher see him as imparting information by explicit statement, as correcting errors, in short as deliberately molding the behavior of the student. It is a matter of trivial semantics whether we want to use the word "teaching" for this kind of activity and exclusively so. But if we do, we must accept the consequence that much, or most, learning seems to take place without teaching. An example commonly cited nowadays in support of this position is the acquisition of language. Some parents do engage in teaching of the most explicit sort. One can discuss whether this helps the child. But it is more important to observe that many parents do not attempt any explicit teaching of correct usage. Nevertheless their children acquire the speech patterns of their linguistic culture. Even more impressive is the evidence amassed by the Piagetian school. Most adults, including parents and teachers, find it hard to belive that children of five and six lack conservation, transitivity and other intellectual constructs defined by Piaget. And if the parents do not even recognize the absence of a concept, it can

consisting of those that appear most plausibly attributable to children. This subset includes two teaching functions on which designers of Teaching Machines have expended the greatest energy, namely the teacher as lecturer and the teacher as drill master. It also includes three functions that have not received serious attention from workers in this area (or as far as I know from research in more traditional branches of educational and cognitive psychology.) These I call (though, of course, the names are merely suggestive and mnemonic) the teacher as diagnostician, the teacher as model, and the teacher as problem.

The next five sections will make a brief preliminary comment on each of these functions, mainly to ensure that they are sufficiently well defined to support more technical discussion later.

## 1.1 Super-books: Automating the Teacher as Lecturer

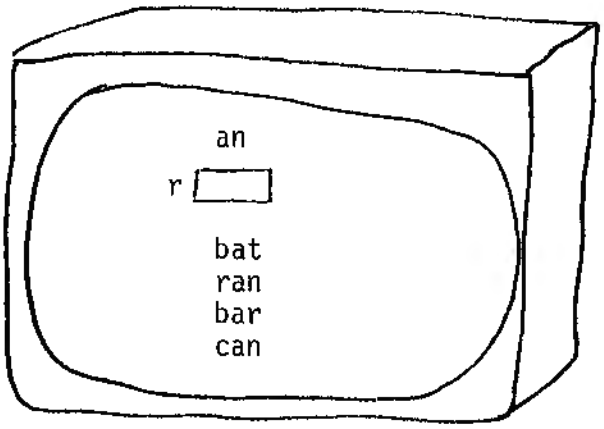
Delivering formal, set lectures in front of a class is widely recognized as a sub-human occupation. It obviously could be done better by a suitable combination of books, movies, feelies and other "multi-media", including the computer.

The word "obviously" is essential to the context of the previous sentence. I think it is important not merely to see that such things are possible, but also to see that this is quite independent of the psychological theories sometimes used (for example, byt followers of B. F. Skinner) to justify the special value of such methods. Nor is it an experimental question. Nothing is more absurd than the host of reports purporting to

## 1.2 Drill and Practice and Drill and Practice and Drill and Practice

The automation of the function of Teacher-As-Drill-Master raises some deeper questions of educational theory. No one has serious doubts about the need to impart information -- whether by book or teacher spouting by computer controlled film strip or whatever it may be. In that case the function was not in question, merely the means of achieving it. In the present case, the function itself is definitely questionable: the need for repetitious exercises is hotly contested, with some extremists calling for their abolition and others calling for all knowledge to be "programmed" into minute cognitive atoms suitable for learning by techniques analogous to those used in conditioning pigeons and rats. Between the extremes there are sober people who argue that while the way-out discussions are in progress about the ideal educational system of the future, right now the computer can make an im prtant immediate contribution by automating the large slice of drill-and-practice embedded in present day elementary education. The claim is seriously made that automated administration of practice activities, for example in reading skills, or elementary mathematical operations can make this work less boring for children and for teachers. At the same time it can make it more effective, not only because it reduces the boredom and consequent bad effects on teacher-student relations, but also because it can exploit some simple principles of learning such as reducing the delay between the learner's attempted performance and the feed-back from the teacher.

that they would find excruciatingly dull in another context. This might be partly the little understood but undeniable fascination of the machine. But a deeper insight comes from observing that it is only in a crudely behaviorist sense that the child's activity is the same in the two contexts. Consider, for example, a typical incident in a program designed to teach elementary reading skills. The child is presented with a display like:




The child moves a word into the box with a light pen. If the word chosen is "ran", the program moves on to the next little problem. If it is "can" the program chosen to emphasize the initial letter, and so on.

It's all very simple. So much so that one might well wonder whether it can do any good, and if it does,one will still ask why a computer is needed. Why not have the child mark the chosen word with a pencil?

It does do good! In fact the Stanford program on reading skills has shown that using machines in just this way for as little as 40 hours spread over a year is enough to enable children who were lagging a year behind their grade level in reading to make two years worth of progress in

gone on to another.

Or even:

Let's see how long I can wait and still get it to go on without complaining about taking too long.

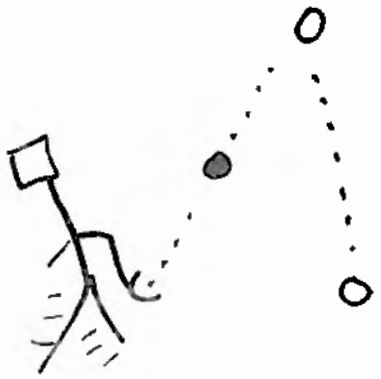
Or:

A bat ran. A bat can. A canned bat. Why's it saying HURRY UP? I was hurring.

If we did, in fact, see such goings on in the child's mind we should have mixed reactions. For although we'd understand the drill phenomenon better, we'd also be impatient with it for not making better use of the child's active mind. But once you shift to complaint, then you are on the way to accepting the computer as instructor of reading and my game is won.

## 3.0 Teacher As Diagnostic Trouble Shooter

I do use teachers as drill-masters. For example, I am learning to juggle and have not found any way to acquire precision in throwing and catching the juggling balls without repetitive practice. Much of this I do alone. Nevertheless I find it useful sometimes to have someone to



And finds he has to step back to catch the ball, he spoils his stance, balance, hand position.

He gets the impression that the balls are always coming at his head.

It might take him a long time and much trial and error and drill and practicd to hit by accident on the proper connection. This is one reason for needing thousands of repetition. A good teacher can spot the trouble and give him a corrective exercise.

would find difficult. Consider the following speculative machine.

The machine is equipped with electronic eyes to enable it to track continuously the actual position in space of the juggling balls and of my hands. This is well within the present state of the art of computer vision.

Once it does this I can immediately use it as a SUPER-MIRROR to see exactly what happened at critical moments, either at the time or on re-run later. Part of this function can be served by using video tapes. But only part, and even then tediously. For I can ask the computer to show me what happened not merely as a picture, but also in analyzed form. So, for example, it could report to me that the time between two balls was gradually deminishing up to a certain throw, when suddenly it increased in a drastically over-compensated way. Then, a little more subtly, it could act as a rapid scratch pad, taking notes of my fleeting thoughts which I express for it in twitches of the ears or lips or sounds or whatever code is most congenial and effective.

A further step, still easily within reach of current technical means, is for it not merely to record but to interpret my signals and signal back appropriately. An interchange might be:

| | ME | MACHINE |
|---|---|---|
| Code for | "Those balls are getting close. I'll push the leading one on the next round." | Code for " Wrong. The trailing ball is out of timing." |

as produced by substituting the expression for one element (Mary's age last year) for another element (Mary's age now). Hence the comment. But if the student had said

$y = 3(x-1)$

it would have changed its diagnosis from "confusion about age when" to "confusion between Mary and Jane".

SAY THIS IN WORDS.

MARY IS TWICE AS OLD AS JANE

IS MARY TWICE AS OLD AS JANE NOW?

NO.

WHEN WILL MARY BE TWICE AS OLD AS JANE?

NEXT YEAR

EXPRESS BOTH NEXT YEAR AGES IN X AND Y.

MARY WILL BE x+1 JANE y+1

SO WHAT'S THE EQUATION?

x+1 = 2(y+1)

GO ON

WE HAVE
x+1 = 2(y+1)
x = 3y-3

SOLVE

The student can command the machine. But the machine need not obey.

HOW DO YOU KNOW IT CAN BE SOLVED?

2 UNKNOWNS AND TWO EQUATIONS

OK
x = 9 y = 4

CAN YOU CHECK THIS?

On the Uses of These Teaching Programs

If one looks at the algebra word program literally (rather than in one of several indirect senses I'll mention in a while) one ought to be struck by a certain lack of taste and sense of proportion. Such a program could easily exist and might actually help some students. But it is absurd to take it as a model of a teaching program in a sound educational system. The kind of cognitive difficulty it tries to cure in the student simply should not exist, and would not exist if the student had been taught properly at an earlier stage. And the situation is not much better if one thinks of it as a remedial measure to help individual students who happen to have been badly treated by the existing system. It would be much better pedagogically to go back to fundamentals; for example by letting them have some turtle-programming experience.

My reasons for mentioning it include wanting to make this point; but there are some less devious ones. The bad feature of the program is not on a non-fundamental topic. It seems more likely that there is a future for programs of this sort to help students with their first elementary LOGO programming.