ISSN 0734-1717

The National LOGO Exchange

Volume 2 Number 2 FORWARD 100! October 1983

LOGO Mess Ups: Process vs. Product

What kind of a model do we provide for our students? I used to protray the image of the all knowing teacher. I knew how to read better than my students, remembered how to work math problems 100% of the time, and corrected their spelling mistakes. They saw the product of 18 years of school - not the process.

I love to learn, and I mess up constantly when I'm doing it; yet I had never showed that part of me - the real me - to my students until last year.

I work with children who have been classified as "learning disabled." By the time they enter my junior high school resouce room, they have experienced six years of school failure. Some of them can no longer confront the possibility of failure, so they actively avoid doing schoolwork. This presents quite a teaching challenge, to say the least.

ENTER LOGO

Then I found out about Logo. Its constructive approach to dealing with initial failure seemed to encourage students to learn by using it. Just what I needed. I decided to try it.

We began learning Logo together last September. I usually stayed ahead of my students in skills because I had a computer at home to practice on. Frequently, though, they asked me questions I couldn't answer right away. I had to look up the information or ask a colleague who had worked with Logo for several years. My students heard me say, "I don't know," many times, watched me rummage in books for answers, and knew that I had to ask others for help.

Logo changed my image. Now my students know that I learn the same way as they do. They place the same demands on me that I place on them: to try to achieve an attainable goal.

I want my students to experience the joy of learning. I want them to choose meaningful goals and learn to overcome those inevitable obstacles. I believe this pursuit of learning is taught best by example.

TURTLE OPERATIONS

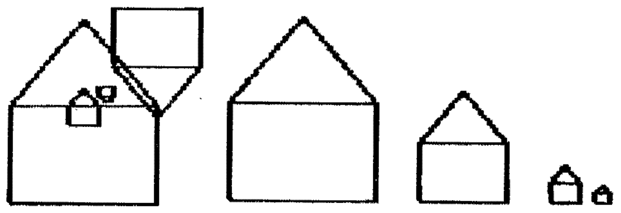
For instance, when we couldn't figure out how to program our robot turtle's touch sensors, my students saw me call M.I.T. for the information. We also operated together on our cybernetic pet's motors. How can I maintain a flawless appearance when I'm on my knees before them, gripping a screwdriver, reciting, "Righty tighty, lefty Lucy," to remember which way to turn the darn screw?

Additionally, they've seen me remain ignorant. Our robot turtle had to be told "TRT 115" for a 90 degree turn, because I had put off learning the assembly language routine that would cure it. And I knew my students would be disgusted with me when they returned to school and found out I hadn't fixed it yet. They'll pressure me to quit procrastinating, and then I really will have to do something about it!

CRAZY HOUSES

On another occasion this past winter, my students succeeded in writing a Logo procedure for the screen turtle to draw a house. I then asked them to begin creating a superprocedure with it.

That evening at home, I decided to create a superprocedure of my own to draw a row of houses of different sizes. As usual, I messed up at first. But, instead of discarding my mistake, I saved that procedure and showed it to the class the next day, along with my final product.



We all laughed at my crazy neighborhood with houses upside down and on top of each other! I can't say for sure, but I think it finally dawned on them that I really do learn the same way they do. And I realized how unfair it had been for me to show them only my final products (since my mess ups occurred at home), when their mess ups were available for public scrutiny on the classroom monitors.

SEPTEMBER STARTUP

When I decorated my room in preparation for the beginning of the 1983-84 school year, I wanted to build on my experiences from last year. I put up copies of both my original and final Logo "neighborhood." To provide my English students with a writing model, I put up my drafts of this NLX article: pages of printouts, paragraphs scribbled out with a red pen, sentences scrawled vertically in the margins.

Next to the drafts, I stapled my final copy. If that one makes it to the presses, I'll display the published version. Thus, both the final product <u>and</u> the sometimes painful process are on display for all to see.

By the way, I really meant to work on that robot turtle, but I waited too late. On the first day of school, James embarassed me in front of the class by asking me if I had fixed it. I said, "No, but I'll try ... if you will help me."

Thus, accompanied by his beautiful grin, we began this school year by confronting and solving a problem together, and thereby set a pattern to follow for the rest of the year.

Nancy Sopp is a special education teacher in the Fairbanks (AK) North Star Borough School District, and has published articles in national educational computing periodicals.

From The Editor

An elementary teacher told a thought provoking story at a meeting I attended recently. She had been using a word processor with a young girl in her writing class. The student had made several mistakes, and felt badly about them. As a result, she grew more frustrated.

Thinking that a demonstration might help the situation, the teacher entered a few sentences into the computer. However, she pressed some of the wrong keys quite by accident. The student suddenly grinned, pointed to the screen and shouted, "You messed up, Teach, just like me!!"

From that point, there was no holding her back. It was OK to make mistakes. Her teacher had done it, and so could she.

What can we learn from this situation? Perhaps we could get a better understanding of the value of the learning process, compared with the learning product, as seen by the student. This idea is key to the Logo learning philosophy.

As Nancy Sopp points out in her article in this issue, students usually see in the teacher the product of 18 (or more) years of education and do not consider the processes involved to attain that station. Hence, it is not obvious to them that teachers learn in the same way students do - by messing up.

"Ah, but if my students see me make mistakes, what will they think of me?" This is a sensitive question for many, and understandably so, given the traditional role of the teacher. For years, the teacher has been viewed as the great oracle, from which all wisdom and knowledge flows in a fountain to be soaked up by fledgling student sponges. If a teacher dared to utter, "I don't know," to a student's penetrating question, or tried a new kind of activity which was not effective, he or she might be branded for life!

Logo gives an opportunity to change this teaching style to a more natural one, in which the teacher becomes a guide to assist the students. But, as importantly, the teacher becomes an unashamed learner, working alongside the students.

For some, such a change is not possible. Personalities, habits, self concepts combine to prohibit innovative behavior. For others, the challenge to relax a little and not be afraid to "mess up" might be taken on successfully.

In that case, students would be able to see more of the process of education in their teachers and might become more responsible participants in their own learning processes.

Shall we give it a try?

MESSUP 100! Tom

Calling All Turtles!

Are you a member of the Young Peoples' Logo Association? If not, you should find out about this upbeat organization. They offer many publications and activities which are of value in the classroom, including a software exchange for members. All popular computer languages (including Logo) are encouraged, but learning in the Logo style is emphasized. To obtain more information, write Jim Muller, Young Peoples' Logo Association, PO Box 855067, Richardson TX 75085.

Logo and Drive 2

Rob Muir of Claremont CA sent this tip for using both disk drives 1 and 2 with MIT Logo (Terrapin/Krell) on the Apple computer. The command .DEPOSIT 43624 followed by a 1 or a 2 will do the job. He also contributed this set of utility procedures for other NLX readers.

```
    TO SETDISK :DRIVE.NUMBER
       IF  ANYOF  (  :DRIVE.NUMBER  <  1  )
(:DRIVE.NUMBER  >  2  )  THEN  MESSAGE  ELSE
.DEPOSIT 43624 :DRIVE.NUMBER
       END

    TO MESSAGE
       PRINT [ SETDISK REQUIRES 1 OR 2 AS
INPUT ]
       END

    TO DISK
       PRINT SENTENCE [ LOGO IS CURRENTLY
USING DRIVE ] .EXAMINE 43624
       END
```

TIPPS for TEACHERS

by

Steve Tipps

The Truth about Numbers

The emphasis on Turtle graphics has often obscured the arithmetic and text capabilities of Logo. Some people do not know that Logo can perform arithmetic operations. Logo comes with the basic arithmetic operators and trigonometric functions; procedures can be written to do many other operations.

Another feature hidden within the arithmetic operations is the ability of Logo to detect equality and inequality between numbers. This might be a very useful tool for teachers who wish to have children explore number concepts.

WHAT ARE THE RESULTS?

Logo performs the four major arithmetic operations of addition, subtraction, multiplication, and division. In MIT (Terrapin and Krell) Logo, Logo signifies the calculations with RESULT: and the answer. Apple Logo is a little less friendly with I DON'T KNOW WHAT TO DO WITH and the answer. Both RESULT: and I DON'T KNOW WHAT TO DO WITH are signals that a calculation has been performed and the answer has not been used to do anything. The result can be passed as input to some other Logo command such as FORWARD or PRINT or RIGHT.

The usual plus (+) and minus (-) signs are for addition and subtraction. The star or asterisk (*) is used for multiplication and a slash (/) means division. Here are some examples of arithmetic:

```
3 + 5
RESULT: 8
9 * 20
RESULT: 180
14 - 300
RESULT: -286
1000 / 33
RESULT: 30.303
```

More than one operation can also be performed on a line. However, the results are not always what is expected or desired. Consider the following:

```
10 - 5 * 3 + 2 / 4
```

If the calculations are done strictly in order from left to right, the result would be 4.25. However, Logo returns

```
RESULT: -4.5
```

After exploring a number of these arithmetic chains, children discover the arithmetic precedence of operations. Logo is being used as a discovery tool to find what rules govern arithmetic. A mnemonic I have used for the order of precedence is "My Dear Aunt Sally," to indicate multiplication and division are performed first, followed by addition and subtraction.

Children will also discover the exception to this order: parentheses. Parentheses can be used in Logo just as they are in arithmetic to cause the operation within them to be performed first. It is fun to make up chains like the following, predict the answers, and compare them with Logo's results.

```
39 + 14 * 3 / 6 - 15
39 + 14 * ( 3 / 6 ) - 15
39 + ( 14 * 3 ) / 6 - 15
```

USING RESULTS

On occasion, you may need a calculation for an input to a turtle command. Logo will do the arithmetic and give the answer to FORWARD and RIGHT. For example,

```
FORWARD 100 * 2 / 3 RIGHT 77 / 9
```

moves the turtle 66.67 steps forward and then turns it right 8.55 degrees. While this sample calculation may not seem useful immediately, figuring out the third side of a 45 degree right triangle is a more common type of problem encountered. If the two sides are 60 steps each, then, by the famous law of Pythagoras, the third side would be

```
FORWARD SQRT ( ( 60 * 60 ) + ( 60 * 60 ) )
```

The turning angles for the 45 degree triangle might come as a surprise to some. Here is one procedure to do this.

```
    TO TRIANGLE.45
       FORWARD 60
       RIGHT 90
       FORWARD 60
       RIGHT 135
       FORWARD SQRT ( ( 60 *  60 ) + ( 60 *
60 ) )
       RIGHT 135
       END
```

Converting this to an all-purpose variable sized 45-degree triangle requires substitution of a placeholder variable, such as :SIZE for 60 and the addition of the placeholder in the naming line. Students may find a way to draw triangles of any size angle and side.

Arithmetic operations can also be combined with PRINT statements as children create addition problems for themselves.

```
PRINT1 [ 200 + 90 + 1 = ]  PRINT 200+90+1
```

gives the result

```
200 + 90 + 1 =291
```

This can be "cleaned up" with parentheses,

```
( PRINT [ 200 + 90 + 1 = ]  200+90+1 )
```

to get

```
200 + 90 + 1 = 291
```

The two ways of printing are the same except for a space after the equals sign. In the first, PRINT1 is used to keep the answer on the same line as the problem. In the second, parentheses are used to tell PRINT that two (or more) things are to be printed together. (For a fuller description of children's arithmetic programs, see NLX, March 1983, Tipps for Teachers.)

WHAT IS EQUAL?

In the addition problem, the equal mark was simply printed on the page. However, when the equal mark is used outside the brackets, something different happens. The equal mark in Logo serves as a detective to find out whether a statement is true or false. In the immediate mode, an arithmetic sentence can be typed in. When RETURN is pressed, Logo responds with the results of its decision: TRUE or FALSE.

```
3 = 3
RESULT: TRUE
5 + 7 = 13
RESULT: FALSE
300 * 41 = 12000 + 300
RESULT: TRUE
300 - 100 + 6 * 4 = 236
RESULT: FALSE
```

The TRUE meaning of equals is demonstrated in Logo because the numerical equality is judged. The equal sign is active in determining equality, rather than a meaningless placeholder.

LESS OR MORE?

Knowing that expressions are equal based on their values needs to be matched with recognition that they can also be unequal. Many classroom activities have been designed for GREATER THAN or LESS THAN signs to masquerade as hungry alligators or voracious sharks wanting to eat the larger number. With Logo, the simple inequality statement is tested for truth.

```
3 < 5
RESULT: TRUE
20 > 25
RESULT: FALSE
10 + 5 < 15
RESULT: FALSE
10000 > 49
RESULT: TRUE
```

Children can play with numbers of any size and in many combinations to verify the relationships based on number. The three possibilities are clearly outlined. With two quantities, one is equal to the other, it is more, or it is less. There are no other choices.

THE PLAIN TRUTH

Playing with equality and inequality signs in Logo supports learning in arithmetic. As children grow to understand that number is an abstract property or descriptor of a set, equality is seen as a fundamental idea. The equality and inequality detectors could be used for exploring beginning number concepts, place value, and operations. At the same time, understanding of equality and testing the relationships is fundamental to work in Logo with TEST, IFTRUE, IFFALSE, and IF ... THEN.

The traditional "guess a number between 0 and 25" game is a good first TEST project for children to work out. The problem scheme would begin with playing the game in class, writing down the dialog, and tracking the decisions. Then, the challenge is to make the computer pick the number and respond with clues about too high or too low. This program can be written in only 10 to 15 lines, but it requires thinking skills and provides opportunity for many creative variations.

Working with operators, functions, and inequalities in Logo should help teachers and students see the truth about numbers. They represent an exciting world within Logo, as vital as drawing on the graphics screen or printing words.

Steve Tipps is a professor in the University of Virginia's School of Education, and presents Logo workshops for school systems throughout the eastern United States.

Logo Study Available

NLX subscriber Reinhold Wappler has just completed an excellent study comparing the capabilities of the MIT and LCSI Logo versions for the Apple computer. This detailed comparison concentrates on the needs of students in grades 2 through 4. However, much of the information applies to many other grade levels also.

Reinhold has agreed to furnish a copy of his study free to any NLX subscriber who sends him a long stamped self-addressed envelope. Send your request to:

Reinhold D. Wappler
MIT/LCSI Logo Study
252 Carter Street
New Canaan, CT 06840

Reinhold also passes along the following tip for MIT Logo users. To clean up your workspace quickly, type the following sequence:

```
ED ALL
CTRL G
ER ALL
ED
```

ED ALL brings all procedures in the workspace to the editor screen. CTRL G exits the editor, leaving all the procedures in the editor buffer. ER ALL erases the entire workspace, but not the edit buffer. ED brings back all of the procedures to the editor screen. Unwanted procedures can be eliminated with CTRL K.

CTRL C will define the procedures remaining in the editor. Thus, you will have in your workspace only those procedures you want. This technique is especially useful for tidying up cluttered disk files.

MICROWORLDS

by
Glen Bull

Numbers and Lists

In last month's column, I described a procedure for calculating average test scores. Although the procedure worked, I later received a challenge to add some safeguards to make it more foolproof.

You see, I made the mistake of putting the sentence, "Foolproof at last!" at the end of the column. Your editor immediately suggested two more ways to make the program fail. My only answer is that, although I tried to make the procedure foolproof, I obviously did not make it Lough-proof!

THE MULTIPLICATION SIGN DOESN'T LIKE LISTS

The reason Tom was able to make my procedure fail has its origin in the relationship between numbers and lists. When a number is entered through the keyboard using REQUEST (or READLIST in LCSI Logo), it is acquired as a list.

[A list, you may recall, consists of anything between brackets, such as a set of numbers, words, Logo commands, or even this sentence!]

Arithmetic cannot be performed directly on a list of numbers. If you enter the following, for example,

```
MAKE "NUMBERS REQUEST
23 34 56
PRINT :NUMBERS
```

Logo will respond with

```
23 34 56
```

But, if you try

```
PRINT 7 * :NUMBERS
```

Logo comes back with

```
* DOESN'T LIKE [ 23 34 56 ]  AS INPUT
```

The reason that the multiplication sign (*) doesn't like [23 34 56] is clear. It doesn't know whether to multiply 7 times the 23, the 34 or the 56, or possibly times all three numbers. (For those of you who teach matrix operations, this situation presents some interesting possibilities.)

In order to gain the cooperation of the multiplication sign, it is necessary to identify the number in the list on which it should operate. For example, you could tell it to multiply 7 times the first number in the list, 23.

```
PRINT FIRST :NUMBERS
23
PRINT 7 * FIRST :NUMBERS
161
```

Of course, it is possible to have a list consisting of just one number by entering something like

```
MAKE "NUMBER REQUEST
50
```

Then, PRINT :NUMBER will return 50.

In that case, it would seem that the times sign could bend its rule a little. If there is only one number in the list, it should be obvious which number to multiply seven by. However, the times sign is polite, but insistent. If the number is actually a list [between brackets], it would still like for you to point out which number to perform the multiplication upon - even if there is only one number!

```
PRINT 7 * :NUMBER
* DOESN'T LIKE [ 50 ]  AS INPUT
PRINT 7 * FIRST :NUMBER
350
```

A PROCEDURE TO INPUT NUMBERS

A procedure could be written which would always extract the first number in a list.

```
TO INPUT.NUMBER
   MAKE "NUMBER REQUEST
   OUTPUT FIRST :NUMBER
   END
```

This procedure can be used just as FIRST REQUEST (or FIRST READLIST) would be.

```
MAKE "VALUE INPUT.NUMBER
50
PRINT 2 * :VALUE
100
```

VIEWING THE OUTERMOST SET OF BRACKETS

At this point, it may be helpful to verify one aspect of Logo. Logo normally does not print the outermost set of brackets which surround a list. This is clearly seen in a PRINT statement.

```
PRINT [ THIS IS A LIST ]
THIS IS A LIST
```

The contents of the variables "NUMBER and "VALUE appear to be identical when they are printed, because Logo drops the outermost set of brackets around the contents of "NUMBER.

```
PRINT :VALUE
50
PRINT :NUMBER
50
```

However, the brackets can be seen if a list of all the global variable names is printed out. This can be done with the command PRINTOUT NAMES (PONS in LCSI Logo). When all the NAMES in the workspace are printed out, the brackets around the contents of "NUMBER can be seen.

```
PRINTOUT NAMES
"VALUE IS 50
"NUMBERS IS [ 23 34 56 ]
"NUMBER IS [ 50 ]
```

FILTERING OUT LETTERS

The procedure INPUT.NUMBER provides a framework for guarding against other potential problems. For example, Tom suggested that my procedure would complain if a letter was typed when the procedure expected a number. This type of problem can be seen in the following sequence.

```
MAKE "VALUE INPUT.NUMBER
B
PRINT 7 * :VALUE
* DOESN'T LIKE B AS INPUT
```

The reason the multiplication sign does not like the letter B is clear. But, it is not uncommon for a letter to be typed when a number is intended. For example, the letter "L" is frequently used in place of the number "1" by office personnel who are used to a typewriter keyboard. Another common error is typing the letter "O" in place of the zero "0". Since INPUT.NUMBER will only be used when a number is expected, a test to filter out letters can be inserted.

```
TO INPUT.NUMBER
   MAKE "NUMBER REQUEST
   TEST NUMBER? FIRST :NUMBER
   IFFALSE OUTPUT INPUT.NUMBER
   OUTPUT FIRST :NUMBER
   END
```

The Logo command NUMBER? (NUMBERP in LCSI Logo) asks whether the value which follows is a number. If it is not, the procedure INPUT.NUMBER is recursively executed, until a number is encountered.

PROTECTION AGAINST EMPTY BRACKETS

Although the procedure is improved by the filter to screen out letters, it still has a potentially fatal bug. To see why, it is necessary to observe the effect of the list operator FIRST on several types of Logo lists:

```
PRINT FIRST [ 23 34 56 ]
23
PRINT FIRST [ 73 ]
73
PRINT FIRST [ ]
FIRST DOESN'T LIKE [ ]  AS INPUT
```

The first item in a list of the three numbers [23 34 56] is 23. The first item in the list containing one number [73] is 73. However, what is the first item in a list that does not contain anything? The question clearly does not make sense. For that reason, FIRST will explain that it doesn't like empty brackets as input.

In the case of INPUT.NUMBER, occasionally an individual may press the RETURN key without typing either a letter or a number. When that happens, an empty list without anything in it is created. As soon as FIRST encounters the brackets with nothing between them, it delivers the error message shown above, and the procedure explodes. Try running the last version of the procedure INPUT.NUMBER, and press the RETURN key in response to REQUEST. What happens?

The way to guard against this occurrence is to insert a test to see whether the list is empty before FIRST is allowed to see the list. If the list is empty, the procedure INPUT.NUMBER is recursively executed.

```
TO INPUT.NUMBER
   MAKE "NUMBER REQUEST
   TEST :NUMBER = [ ]
   IFTRUE OUTPUT INPUT.NUMBER
   TEST NUMBER?  FIRST :NUMBER
   IFFALSE OUTPUT INPUT.NUMBER
   OUTPUT FIRST :NUMBER
   END
```

This revision would appear to be nearly both foolproof and Lough-proof, but is it? There is at least one condition in which this procedure works functionally, but is deficient from an esthetic viewpoint. This condition involves destruction of the formatted text on the screen.

If a letter is typed in place of a number, the cursor does not return to its original position. Instead, the cursor moves to the next line, destroying text to its right on the preceding line.

A partial fix can be adopted by recording the position of the cursor before asking for input. If the input is not a number, the cursor can be returned to its original position before requesting another input. The position of the cursor can be determined by examining memory locations 36 and 37 in MIT (Terrapin and Krell) Logo. The same information can be obtained through use of the command CURSOR in LCSI Logo.

This is the final version which seems to correct all the bugs with which I am familiar. However, I am sure there is at least one more. Who will be the first to find it?

```
TO INPUT.NUMBER
   MAKE "HORIZONTAL .EXAMINE 36
   MAKE "VERTICAL .EXAMINE 37
   MAKE "NUMBER REQUEST
   TEST :NUMBER = [ ]
   IFTRUE CURSOR :HORIZONTAL :VERTICAL
   IFTRUE OUTPUT INPUT.NUMBER
   TEST NUMBER? FIRST :NUMBER
   IFFALSE CURSOR :HORIZONTAL :VERTICAL
   IFFALSE OUTPUT INPUT.NUMBER
   OUTPUT FIRST :NUMBER
   END
```

The corresponding LCSI version for the Apple is

```
TO INPUT.NUMBER
   MAKE "POSITION CURSOR
   MAKE "NUMBER READLIST
   TEST EMPTYP :NUMBER
   IFTRUE [ SETCURSOR :POSITION ]
   IFTRUE [ OUTPUT INPUT.NUMBER ]
   TEST NUMBERP FIRST :NUMBER
   IFFALSE [ SETCURSOR :POSITION ]
   IFFALSE [ OUTPUT INPUT.NUMBER ]
   OUTPUT FIRST :NUMBER
   END
```

Glen Bull is a professor in University of Virginia's School of Education, and teaches Logo courses at both the graduate and undergraduate level.

MECC '83 Conference Announced

The Minnesota Educational Computing Consortium announces its MECC '83 Educational Computing Conference, scheduled for November 18-22, 1983. A series of 2-day Logo workshops will be offered, as well as Logo activity presentations. For more information, write MECC '83, 2520 Broadway Drive, St. Paul MN 55113, or call 612-638-0683.

New Mexico Students Think Logo is VAN-tastic

The Alamogordo Space Center and New Mexico State University have joined forces to produce a Logo program that is really rolling, in every sense of the word! A van called "The Computer Experience" brings Logo to elementary students and teachers at schools throughout the state.

The van is equipped with 10 TI 99/4A computers, and is staffed by two professional educators. They can set up a temporary computer facility which handles up to 30 students at a time, providing them and their teachers with a familiarization of computers in general and of Logo in particular. Once all the classes and teachers of a school have rotated through the introductory activities, the computers are loaded back into the van and "The Computer Experience" roars off to its next destination.

Perhaps this innovative concept is one which could be used in your state or local situation. For more information on "The Computer Experience," write to: Robert F. Content, Executive Director, The Computer Experience, Alamogordo Space Center, PO Box 533, Alamogordo NM 88311.

LEARNING WITH LOGO
Arrives at Last

Learning With Logo, by Daniel Watt, is a likely candidate for a "Logo book of the year" award. Its 365 pages are full of activities which are designed to help beginners learn Terrapin or Krell Logo on the Apple computer.

To say that the book has been long awaited is an understatement. Its release finally came in late May. When Watt stood up at the NLX Birds-of-a-Feather Logo Session at NECC in Baltimore and raised the book above his head in triumph, he was greeted with a sustained round of warm applause. No words were necessary.

"Learning With Logo" takes readers through turtle graphics first, with basic commands, recursion, and variables. A study of the classic POLY is made, followed by work with numbers, words, and lists. The last half of the book is devoted to extending these ideas, using special tools and reference material.

Watt drew upon his extensive Logo experience and designed his book so that readers could learn by personal exploration. He has written many procedures to facilitate this exploration. These microworlds contain important learning concepts and are easy for beginners to use.

Watt stresses that, for maximum benefit, the book should be used with a diskette containing these procedures. A listing of the procedures, from which a diskette can be made, appears in the book. Or, the diskette could be ordered for $15.95 additional.

One of his sets of procedures is an activity called SHOOT, which helps learners gain experience in estimating turns and distances. A circle is drawn on the screen and the turtle is displayed some distance away, pointing in a random direction. The learner turns the turtle until it is pointing toward the circle, and then "shoots" the turtle a specified distance forward. If the turtle stops inside the circle, a congratulatory message is displayed. Otherwise, the learner is given the opportunity to make adjustments.

Watt points out that young children need a lot of practice estimating distances and directions. Activities such as SHOOT will help give them the turtle control skills necessary for more advanced work. He also encourages more experienced Logo users to modify his procedures into more complex ones, and gives extensive hints and suggestions for doing so. It is here that the richness and depth of the book shows through.

Teachers will find these suggestions to be gold mines of ideas for other Logo projects. In addition to being a valuable reference, portions of the book could be used as an individual text for children as young as 10 or 11. A 13-year-old reported to me that about one-third of the book was "moderately difficult" to read, and the rest was easy.

An article, "Learning With Logo," on page 40 of the April 1983 issue of Classroom Computer News magazine (now Classroom Computer Learning) contains excerpts from his book. This gives a more detailed foretaste of what to expect.

Watt has four other similar books on the way for learning with Apple (LCSI) Logo, Atari Logo, Commodore Logo, and Texas Instruments Logo.

Learning With Logo, Daniel Watt. BYTE/McGraw-Hill, $19.95.

Logo Notes

Dagobert and Lissa Soergel have written a detailed comparison of the commands in the MIT and LCSI Logo versions for the Apple computer. To receive a copy, send a long self addressed envelope with 54 cents of postage attached to:

Dagobert and Lissa Soergel
MIT/LCSI Logo Comparison
College of Library Info Sci
Univ of Maryland
College Park, MD 20742

Have you ever wanted to make a printout of your LCSI Logo procedures without having the .PRINTER 0 appear at the end? Here is one way to get around it. Enter the following all on one line.

```
.PRINTER 9 ( POALL ) .PRINTER 0
```

From Four Bugs to a Society of Turtles

by
Michael Burke

The cover of the August 1982 BYTE magazine (the LOGO issue) depicts an Escher-like sketch of the paths of four bugs. The paths evolve from placing the bugs at the corners of a square and having each bug always moving towards its clockwise neighbor.

The LOGO program that simulates this motion uses property lists to keep track of the positions of the bugs and transports the turtle from one bug position to another to draw small portions of the path. (Ed. note: see pages 236 and 238, BYTE magazine, August 1982)

PROBLEM SOLVING WITH BUGS

While the statement of the problem and the projected solution can be simply stated, the published program is incredibly complex in comparison. The program is useless in solving similar problems. For example, what paths evolve if we have five bugs at the vertices of a five sided figure? No part of the published program can be used in this new program.

The complexity of the published program arises because of the unnecessary use of local and global variables, the MAKE command, and the entanglement of bug representation and bug motion. There is also no need to ever think of the location of the bug as a point in a cartesian coordinate system. It is the kind of solution that a progessional programmer would come up with; it is intimidating to the non-professional.

A problem solving approach leads to a cleaner solution and a powerful mechanism for dealing with similar problems. The whole philosophy behind problem solving implies that we can learn a great deal on the way to the solution.

MANY BUGS TO SOLVE A PROBLEM

A bug has the same properties as a turtle, so let's think in terms of multiple turtles. Each bug can be represented by a turtle. Let's also assume that our turtles have names: FRED, SALLY, JANE, and SAM. Finally, we assume that the turtles are positioned at the corners of a square as follows:

```
FRED          SALLY



SAM           JANE
```

The simulation in LOGO is easily specified in this setting where :FRED, :SALLY, :SAM, and :JANE are assumed to be variables representing turtles.

```
TO MOVEBUGS
   MOVEBUG :FRED FINDLOC :SALLY
   MOVEBUG :SALLY FINDLOC :JANE
   MOVEBUG :JANE FINDLOC :SAM
   MOVEBUG :SAM FINDLOC :FRED
   END
```

FINDLOC is a procedure which returns the location of the named turtle, and MOVEBUG moves the named turtle a small distance towards that location.

BUG QUESTIONS

There are several questions that need to be addressed here: What is a turtle? How do I control multiple turtles? How can I represent more than one turtle in a LOGO that only provides one?

The second question is the important one. That is, we need to direct messages like FORWARD 10 to a specific turtle. One way is to require a name of a turtle as an input to each turtle command. However, extra inputs require extra keystrokes, and we would soon tire of typing in the name if we are sending a long sequence of messages to the same turtle.

WHERE DO BUGS COME FROM?

We have not yet mentioned how turtles are created in Mattel LOGO. Actually, they are HATCHed. HATCH "FRED will create a new turtle whose name is FRED (and will associate the new turtle with the variable :FRED) at the current turtle's position.

Thus, FRED, SALLY, JANE, and SAM could be created either in calculator mode or programmatically by defining a function which hatches four turtles at the desired positions. For example,

```
TO HATCHBUGS
   HATCH "FRED
   FD 30 RT 90 HATCH "SALLY
   FD 30 RT 90 HATCH "JANE
   FD 30 RT 90 HATCH "SAM
   END
```

So what needs to be done to simulate multiple turtles, when you only have one? You need to have a representation in mind (e.g. property lists), a way to keep track of several turtles (a list of turtle names associated with a global variable) and you need to define functions that act like HATCH and ASK. The not-too-difficult details are left as an exercise in problem solving.

FOUR BUGS FINISHED

When you have accomplished this task, you not only have a solution to the four bugs problem, you have a powerful mechanism that can be employed to solve other multiple turtle problems. A by-product of problem solving should always return more than the solution to a specific problem.

The ASK primitive of Mattel LOGO suggests more than a polite way of commanding a turtle to do something. It suggests that each turtle is an independent object that will listen politely, but is free to act in any way it deems appropriate.

Mattel LOGO has multiple turtles and the right idea on how to manage several turtles. There is always a single turtle that is listening for messages. When we want to send a message to another turtle we simply ASK it to respond. For example,

```
ASK :FRED
FORWARD 50
RIGHT 90
ASK :JANE
BACK 20
LEFT 30
```

will cause FRED to move forward and turn right, while JANE will move backward and turn left. When only one turtle exists, all messages are sent to the existing turtle.

The definitions of FINDLOC and MOVEBUG are then easily designed using the Mattel primitives ASK, POSITION, FACE, and FORWARD.

```
TO FINDLOC :BUG
   ASK :BUG POSITION ?
   END

TO MOVEBUG :BUG :LOC
   ASK :BUG FACE :LOC FD 10
   END
```

FINDLOC asks the named turtle for its location. POSITION is a primitive function that requires either a list of two numbers or a "?" as its only input. If the input is a "?", then the position of the current turtle is returned. If the input is a list of two numbers, the turtle will be moved to the corresponding point.

MOVEBUG asks the named turtle to FACE the location input (its leftmost neighbor) and move forward 10.

The possibilities here are endless. Imagine a family of turtles each of which is programmed with a certain behavior (reactions to certain kinds of messages). We now have a microworld of a society. We might even be able to change the behavior of a turtle if we could communicate to it in the right way!

Michael Burke teaches graduate and undergraduate courses in mathematics and computer science at San Jose State University, and conducts LOGO workshops throughout California.

ERIC Announces Computer Newsletter

ERIC's Clearinghouse on Elementary and Early Childhood Education has begun publication of a bimonthly newsletter called "Micronotes." It contains information on applications of microcomputers for instructional, creative, recreational and administrative uses, and usually mentions several Logo activities, among other things. The cost is $8 per year. ERIC/EECE, College of Education, University of Illinois, 805 West Pennsylvania Avenue, Urbana IL 61801, or call 217-333-1386.

New Series of Logo Booklets Announced

Kathleen Martin and Donna Bearden have announced a new series of booklets which give teachers Logo based activities for use both on- and off-computer. Each booklet in the series concentrates on a separate group of mathematical concepts. The activities show students how to use Logo to explore and understand these concepts.

The first volume in the series, "The Rule of 360," contains 50 pages designed for 10 to 12 year old students who have had some Logo experience. The second volume, "Polyspi Inspi," will be available by late October. Each booklet costs $7.95, and may be purchased with a Logo-specific diskette for $4.95 additional. Diskette versions available include Apple (LCSI), Terrapin/Krell, Atari, TI, and Commodore.

For more information, write Martin-Bearden Inc., 1908 Sandy Lane, Irving TX 75060, or call 214-253-6579.

Logo Research Directory to be Compiled

The National Logo Exchange is compiling a directory of all persons conducting any research relating to Logo. If you would like to be listed, send your name, address, telephone number, position, and a brief summary of your research interests and activities. Copies of the directory will be made available in mid-1984. Mail your information to Research Directory, Attn: NLX, PO Box 5341, Charlottesville VA 22905.

Start a Local Logo Exchange

Do you feel as if you are alone in the Logo world? Would you like to be able to talk with Logo folks from other schools once in a while? Do you want to know someone nearby whom you can call for Logo help? Then think about forming a Local Logo Exchange.

Local Logo Exchanges (LLX) are made up of Logo teachers and other workers who want to meet periodically to exchange ideas and share Logo activities in an informal setting. The NLX supports the spontaneous formation of such groups by furnishing suggested guidelines for organizing, giving publicity in the NLX newsletter, and helping to identify other Logo workers nearby as prospective members.

If you are interested in organizing such a group, then write to NLX Local Logo Exchange Program, PO Box 5341, Charlottesville, VA 22905. Tell us something about your situation, and what you hope to do. We will send you a copy of the LLX Guidelines and suggestions for getting started.

What better way to learn more Logo and get to know others in the field at the same time?

Logo Summer Courses

If you are teaching a Logo course at the college or university level during the summer of 1984, please send information to the NLX as soon as possible. Our readers would be interested in the location, dates, number of hours, prerequisites, cost, and general content of the courses. We will include your course announcement in our spring 1984 issues. Send information to: 1984 Logo Courses, Attn: NLX, PO Box 5341, Charlottesville Va 229205.

The National Logo Exchange, copyright 1983 by Posy Publications, a part of The Posy Collection, all rights reserved. Published monthly, September through May, $25 (US) per year, mailed FIRST CLASS from Charlottesville, Va. $5 (US) additional per year for addresses outside the USA, Canada, and Mexico. The opinions expressed by the authors are not necessarily those of The National Logo Exchange. Permission is granted for libraries and others registered with the Copyright Clearance Center (CCC) to photocopy articles herein for the flat fee of $2 per copy of each article. Payment should be sent directly to CCC, 21 Congress Street, Salem, MA 01970. Address other correspondence to: The National Logo Exchange, Box 5341, Charlottesville, VA 22905. ISSN 0734-1717

Editor..Tom Lough