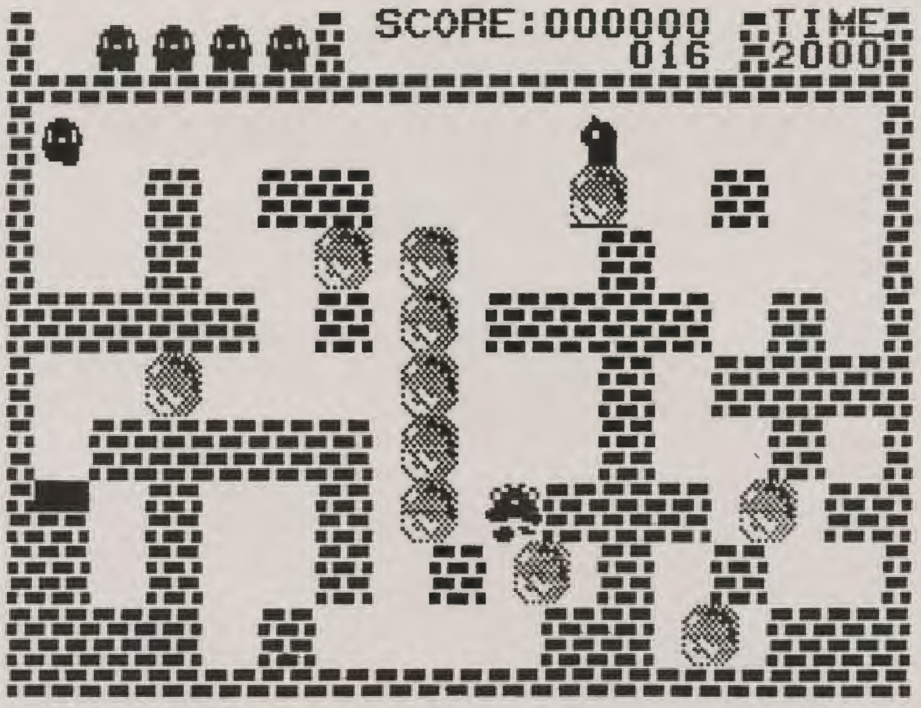
Journal of the ISTE Special Interest Group for Logo-Using Educators

LOGO EXCHANGE

September 1989 Volume 8 Number 1



International Society for Technology in Education

ISTE Publications

LOGO EXCHANGE

Volume 8 Number 1 Journal of the ISTE Special Interest Group for Logo-Using Educators September 1989

Logo Exchange is the journal of the International Society for Technology in Education Special Interest Group for Logo-using Educators (SIGLogo), published monthly September through May by ISTE, University of Oregon, 1787 Agate Street, Eugene, OR 97403-9905, USA.

POSTMASTER: Send address changes to Logo Exchange, UofO, 1787 Agate St., Eugene, OR 97403. Second-class postage paid at Eugene OR. USPS #000-554.

Contents

ISTE Membership (includes *The Computing Teacher*)

	U.S.	Non-U.S.
	28.50	36.00

SIGLogo Membership (includes *The Logo Exchange*)

	U.S.	Non-U.S.
ISTE Member Price	25.00	30.00
Non-ISTE Member Price	30.00	35.00

Send membership dues to ISTE. Add $2.50 for processing if payment does not accompany your dues. VISA and Mastercard accepted. Add $18.00 for airmail shipping.

From the Editor

Welcome Back!

Has your summer been as busy — and short — as mine? Does your list of summer plans still have as many unfinished tasks as mine? Back in May, September seemed so far away and yet, before we know what has happened, the school year is upon us. The pleasant summer days are behind us, but the new challenges of a school year are before us.

It's been a busy summer for us here in Oregon. At the National Educational Computer Conference (NECC) in Boston, the finishing touches were made on the merger between the International Association for Computers in Education (IACE) and the International Council for Computers in Education (ICCE). The new merged organization, the International Society for Technology in Education (ISTE) is now a larger, stronger organization to serve as a support base for those of us in the Logo community who are members of SIGLogo. Take a look at the cover of your first *LX* and you will see indication of the change.

During the months between the time I completed the May issue and began work on September, I have heard from quite a number of you. It's been delightful receiving article ideas, art work for covers, and just friendly notes from readers. In spite of these many varied contacts during the last few months, as I began to think about the September issue, I realized that I missed hearing from our loyal columnists every month. Their cheery notes and delightful writing brightens my day as each new deadline rolls around. Perhaps you too missed the monthly dose of new ideas and challenges.

As we begin the new year, I want to take a moment to welcome two new columnists to the pages of *LX*. First, we are very excited to have Dorothy Fitch of Terrapin, Inc. writing a beginner's column for us. You may know Dorothy for her *Kinderlogo* or her *Logo Data Toolkkit*. As you'll see from this month's column, Dorothy is going to provide you with simple ideas useable by Logo novices. However, I think the more experienced among you will find many things that you can use in Dorothy's work.

Although our other new column won't begin until October, I'm sure you will be pleased by the addition of Eric Brown's work to the pages of *LX*.. Eric, who formerly was with Logo Computer Systems, Inc., will be writing a column to help you with technical problems. Eric has been a <u>great</u> help to me in the past — as he may have been to a number of you! His column will address common difficulties, but you should feel free to send questions about any version of Logo to Eric at the ISTE address below and he will include answers in future columns when possible. We hope that providing some technical help to you through the pages of *LX* will make your use of Logo even easier.

Even though we have added two columnists, we actually have fewer "regulars" than we did last year. We hope to fill those extra pages with material from you, dear reader. We want to know what you are doing in <u>your</u> classrooms. Send an article. Send some material for the cover. Send a brief description of a project with some pictures. Become a part of SIGLogo and *LX*. Only by sharing can we continue to grow together.

I'm looking forward to an exciting new year. With your help, it can be the best ever for *LX* !

P. S. For those of you following this (sad) story, I finally found an agate all by myself! Here is a scanned image of the real thing. There's hope. I may become an Oregonian yet!

Sharon Yoder is beginning her third year as editor of the *Logo Exchange*. A member of the faculty at the University of Oregon, she teaches a variety of computer education courses. In addition to her teaching and editing responsibilities, she is currently developing an order entry system for ISTE in Foxbase (DBASE), serving as coeditor for the "Logo Center" column for *The Computing Teacher*, and working on a book on telecommunications to be published by ISTE. Before moving to Oregon, she taught computing and computer science at the middle and high school level as well as serving as a district wide computer coordinator in Ohio.

Sharon K. Yoder
SIGLogo/ISTE
1787 Agate Street
Eugene, Oregon 97403

Monthly Musings

FD Ever FD
by Tom Lough

Judi Harris is one of my all-time favorite people. Those of you who know her personally have a similar opinion, I'm sure. The license plate on her car is the inspiration for this month's musing. It reads:

```
FD :AMT
```

For me, this simple statement holds a lot of meaning. Perhaps the first thing that comes to mind is FD 100! From the beginning of this publication in September 1982, FD 100! has been a kind of motto. I remember writing quite a number of September back-to-school editorials based upon it. I suggested (and suggest even more today) that FORWARD is the direction of progress, improvement, growth. One hundred percent of the *LX* supports you, the teachers using Logo. The exclamation mark represents the excitement Logo brings to learning.

But :AMT can have values other than 100! Over the years, I have received many letters from teachers telling about their trials, tribulations, and triumphs using Logo with their students. It was not particularly easy for them to learn the more challenging aspects of Logo and to figure out how to apply them effectively in their classrooms. Most of them had to struggle just to stay up with their students. Many times, they would include in their letters a statement similar to, "I usually feel like I am going FD 5 BK 2 instead of FD 100!"

Such a statement always made quite an impression on me. First, I marvelled that they would select Logo commands to provide a description of their feelings. I was also pleased that they were able to communicate their feelings so effectively. Life really is more like FD 5 BK 2 than FD 100! So often, we measure progress in small steps instead of large leaps. And, with success, we sometimes experience minor setbacks, often as a result of trying something else based on our recent success. There is nothing wrong with this sequence. In fact, it is a classic sequence of natural learning. We often fail our way to success in small steps. As always, we must caution ourselves that this is so with our students, too. How difficult it is to be properly patient with them! There is certainly nothing wrong with the value of :AMT being small.

In fact, one of the smallest values of :AMT was key in a major "discovery" for me and thousands of others. When FD 1 RT 1 is REPEATed, things really get interesting! Of all the "classic" Logo problems and procedures, I must have given the ones dealing with circles and arcs the most thought and analysis. For me, it was a consuming challenge to figure out the most straightforward way to draw a circle so that the center was not offset by a little bit.

With a little more thought, we can consider even "smaller" values for :AMT. Think about the classic real number line, with numbers getting larger as you go further to the right along the line from zero. If you go the other way [to the left], the numbers get smaller. If you continue going left <u>through</u> zero, the number values still get smaller, but the numbers themselves begin to get larger again. You are in the realm of the negative numbers! If you and your students have not yet explored commands such as FD -50, you are in for a treat! After I had some experience with this idea, the concept of signed number operations seemed somehow much easier to understand. I associated the addition operation with FD and the subtraction operation with BK. Then I could see that a line such as FD 50 BK -30 FD -60 would correspond to (+50) - (-30) + (-60).

Now let's consider values in the other extreme. Surely every *LX* reader has has a student who has typed in FD 10000 or something similar, just to see what would happen. (And, I'm confident that all *LX* readers have done so themselves!) If the turtle was facing vertically or horizontally, it seemed to undergo some sort of weird but fundamentally uninteresting displacement. But if the turtle was facing any other direction, oh my! For some reason, the behavior of the wrapping turtle going bananas with FD was intriguing to students of all ages. Patterns, plaids, all kinds of graphic designs were produced in enormous quantities. Some teachers refused to let their students play with this wrapping, while others encouraged it. Nevertheless, it seems clear that the combination of FD and a large number offers a peculiar fascination.

In our own lives, I hope that the spirit of the FD command is a central one. Only by going FD do we make progress. The input to FD is certainly best expressed as a variable, such as Judi's :AMT on her Virginia license plate. Sometimes, the input is small; at other times it is larger. The longest journey is made up only of a series of steps. Whether you are trying to learn another programming concept, organize your interim report cards, or aiming for a higher degree, the process is likely to be one which is accomplished one step at a time.

As this school year begins, I wish for each of you and for your students persistent progress in a FD direction. And, as always,

FD 100!

Tom Lough
Founding Editor
Box 394
Simsbury, CT 06070

Logo Ideas

Map Making: Many Variations on a Simple Theme
by Eadie Adamson

For young students or beginners of any age, it's fun to make a drawing program that is a version of the many single key-stroke INSTANT programs often used with small children and Logo. Give your students just the beginnings of such a program, then challenge them to follow the pattern and customize their program. You'll be surprised at the number of variations possible. While they work out the ideas they will also be reviewing for themselves the important Logo commands for moving the turtle about the screen. In addition, creating the program themselves will give them a sense of ownership.

First, show them this simple two-step program, **draw**:

```
to draw
drawit readchar
end

to drawit :key
if :key = "f [forward 5]
draw
end
```

Readchar is a Logo primitive which waits for a key to be pressed. In this case, the key pressed becomes the value of **:key**, the input to **drawit**. The **if** statement in **drawit** then checks to see if a task has been assigned to that key: **if :key = "x**, for instance. If so, Logo does what is assigned (the task in brackets which follows). Finally, procedure **draw** is called. Again, **readchar** waits for a key to be pressed, and the value of the key pressed is passed into the name **key**. If there is no command for that key, again **draw** waits for a key to be pressed. This process continues indefinitely.

I usually give students on a single command:

```
if :key = "f [forward 5]
```

Once students have entered this portion and try it out, they quickly perceive that a drawing program needs more than simply a key for forward in order to be effective. They can then follow the pattern and create their own set of commands. I ask them to think about what their program should do so that they can draw easily, pointing out that even procedures can be included, if they choose. It also helps to remind them that if the forward and back distances are the same, and likewise the right and left turns, it will be easier to move about and get back to a given place than if the numbers are different.

Finally, make a key to stop the program when it finished:

```
if :key = "s [stop]
```

Integrating DRAW into Social Studies?
In the third grade at our school the students study the geography of the states. As a culminating unit, in computer class they create a drawing program similar to this one and use it to draw their own imaginary state. They make shapes to indicate topographical elements, stamp them in the appropriate places, and then use the LABEL keys to mark their maps, often including a key for reading their map. The project makes a wonderful connection to the classroom work, providing the students an opportunity to synthesize what they have learned and to apply it in a slightly different way.

Since we embark on this project near the end of the year, the students usually only have time to complete their "states." When there is time, the imaginary state unit should also include inventing a state flag, writing a state song (words and music), making a page with shapes illustrating the state bird, flower, tree, animal, and so on. There could even be a page of history of the state.

This kind of project could also fit into a larger project on the nations of the world, in which each student might choose a nation to research and then produce a series of pages about it. The pages could all be linked into a larger "data base" type of program (most feasible if you are fortunate enough to have 3.5" drives which can contain many more pages).

Other extensions of this idea
Add a key to change the color of the turtle. Use a **changecolor** procedure like this:

```
to changecolor
cc
type [What color: 0 (black) 1 (white)
   2 (green) 3 (purple) 4 (orange) 5
   (blue): \ ]
setc readchar
cc
end
```

The line added to **drawit** might look like this, assuming you choose Z as the key to press for a color change:

```
if :key = "z [changecolor]
```

Think about using the label command as a flashing sign. Students could program a flashing sign or an arrow which would appear in response to a word typed in the Command Center. For instance:

```
    to capital
    ht
    pu
    setpos [-100 40]
    repeat 10 [label [STAR CITY] wait 5]
    end
```

Here's another, simply an arrow:

```
    to park
    ht
    pu
    setpos [-60 60]
    repeat 10 [label [\-\-\-\>] wait 5]
    end
```

The back slash in the last line above "quotes" the character following it. That is, the back slash causes the character following the \ to be printed "as is." Otherwise, LogoWriter interprets the dash as a minus sign and automatically puts a space before and after it. The same is true of the > sign.

This flashing sign idea provides a good opportunity for teaching how to use SHOW POS to get the location of the turtle. SHOW causes a list to appear in the Command Center. That list can be used as the input to SETPOS, which takes a list of two numbers. Start by showing a sample and perhaps giving a few simple challenges, such as: find out how to make the turtle jump to the upper right corner of the screen, then to the upper left.

Students might add a HELP procedure which tells which locations can be found:

```
    to help
    cc
    type [You can ask for: CAPITAL, PARK,
       TRAINS, AIRPORT.  Type your choice
       and press RETURN.\ ]
    run parse first readlistcc
    end
```

Readlistcc collects a list of information from the Command Center, reporting what has been typed as a list; **first** takes the first item of the list and, in this case, reports it as a word; **parse** makes the word into a list; **run** requires a list as input and so can now run the procedure name typed. Students may also adjust the turns in the drawing program and then use it to create mazes through which they can drive the turtle. One of my students created an interesting variation on this program. He simply set up a control key for each command and then used these keys to draw some marvelous mazes. His procedure was something like this:

```
    to keys
    when "z [seth 90]
    when "x [seth 270]
    when "n [forward 10]
    when "o [back 10]
    when "r [pu]
    when "p [pd]
    when "q [right 45]
    end
```

To set up the controls keys, he simply typed the word "keys" in the Command Center and then pressed Return. That activates the keys and they remain active until the computer is turned off, or until the **clearevents** command is given. He needed no further programming to draw his maze.

Moving on...

When we begin to work with motion, it is a simple task to convert the drawing program to one which controls a moving turtle. If the students are still working with maps, they might want to "drive" the turtle around their state. Copy the **draw** procedure, then change the copy of **draw** to read like this:

```
    to drive
    forward 1
    if key? [drawit readchar]
    drive
    end
```

The **drawit** procedure can stay the same (except that the **stop** command must be changed to **stopall**), or students may wish to make additions to it as well. If their maps have color spots, the turtle can also be programmed to detect the color and either flash a sign or have a location appear in the Command Center. Be sure to pick the turtle's pen up before you drive!

A number of my older students have used a drawing program to quickly add graphics to their hypertext projects. They have added keys to fill and to erase, as well as to change color. One of my students worked on replicating Delta Drawing, using a program like this with four turtles.

What other variations can you and your students dream up?

Eadie Adamson
Allen Stevenson School
132 East 78th Street
New York, New York 10021

Embedded Recursion and Russian Dolls

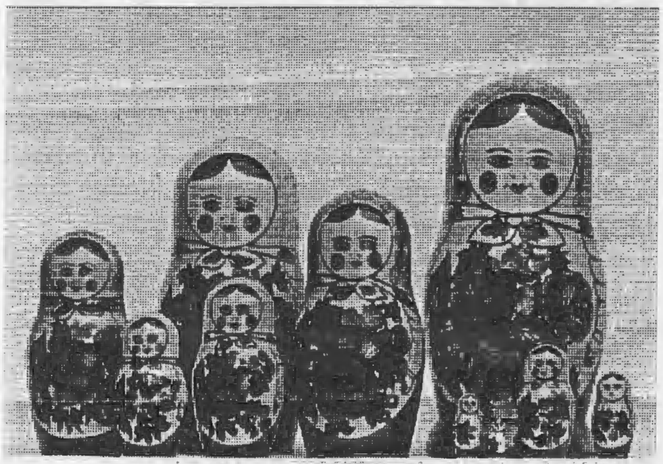
by Jessica Kahn

I teach programming in Logo to teachers. They are able to use tail recursion and have some notion of how it works, but embedded recursion such as the SHAPE procedure below confuse them. Many of my students have difficulty understanding how this procedure works:

```
TO SHAPE :SIZE
IF :SIZE < 5 STOP
FORWARD :SIZE RIGHT 90
SHAPE :SIZE/2
FORWARD :SIZE LEFT 90
END
```

For many of my students the confusion begins with the way they learned tail recursion. While it is true that tail recursion is Logo's way of writing loops, this notion is not helpful when teachers try to understand embedded recursion. If they think in terms of loops rather than using the same procedure again and again, they disregard an important piece of information about tail recursion, namely that as each procedure ends, control returns to the procedure that called it, even if the only command left in that procedure is END. It is possible to work with tail recursion successfully without thinking about this flow of control. However embedded recursion must be understood in terms of flow of control.

One of the principles of Logo is the value of making abstract notions concrete. With this in mind I found a way to help my students visualize what was going on in embedded recursion, with a set of "matrushkas." These are painted wooden dolls of decreasing size, nested one inside another.



Some sets include four dolls, though more elaborate sets may have as many as twelve. They can be found in toy stores that feature wooden toys, and they are sold in ethnic gift shops, including the gift shop at the United Nations Building in New York City.

I brought my matrushka doll into class and I showed my students that the dolls come apart, and that there is another doll inside. I asked them if they could talk out a procedure that would take the dolls apart and put them together again. Here is the procedure we wrote:

```
TO PLAY.WITH.DOLL :OUTER
IF DOLL DOESN'T OPEN STOP
OPEN OUTER DOLL
REMOVE INNER DOLL
PUT INNER DOLL IN OUTER DOLL
CLOSE OUTER DOLL
END
```

This procedure covers one level — that is, it results in taking apart one doll, discovering the doll inside, taking it out, putting it in again and closing up the doll. However the inner doll has a doll within it, and the same thing can be done with this doll. I asked the students where they might put the line that would tell us to do the same thing with the inner doll.

Their first instinct was to put the line at the end, but when we tried to "walk through" the procedure, it was clearly impossible, since the inner doll was back inside the outer doll. Finally they added the recursive call:

```
TO PLAY.WITH.DOLL :OUTER
IF DOLL DOESN'T OPEN STOP
OPEN OUTER DOLL
REMOVE INNER DOLL
PLAY.WITH.DOLL :INNER
PUT INNER DOLL IN OUTER DOLL
CLOSE OUTER DOLL
END
```

This procedure will not run in Logo of course. It only "runs" in English. First I checked to see if the doll opened. The largest doll did. So I opened it, and removed the inner doll. So far so good. Here was the embedded recursive call, to play with doll, but with the inner doll. So I had to put the outer doll down, in two pieces and start a new procedure with the inner doll.

Again I checked to see if the doll would open, and it did. So I opened it and removed the inner doll. We had come to the embedded recursive call again, and again I had to put aside the doll I had taken apart and begin a new procedure with an inner doll. This process continued until I opened the fourth doll and removed the fifth doll. This doll did not come apart.

"Now what?" I asked the class. They could understand that PLAY.WITH.DOLL :INNER at level 5 had stopped. And they also understood that level 4 of the procedure could now be completed —

```
PUT THE INNER DOLL IN OUTER DOLL
CLOSE OUTER DOLL
```

In fact it's impossible to put all the pieces of the matrushka together any other way. So I put the smallest doll in the next smallest doll and repeated, "Now what?" The class instructed me to put the doll in my hand in the next largest doll and close that doll. We continued "picking up" the last lines of this embedded recursion procedure until the whole set was reassembled.

We also used the matrushka to understand the following procedure:

```
TO PRINT.WORD :WORD
IF :WORD = "   STOP
PRINT :WORD
MAKE "WORD BUTLAST :WORD
PRINT.WORD :WORD
PRINT :WORD
END
```

I instructed them to run the procedure using the word "DREAMS and they saw this output on the screen:

```
DREAMS
DREAM
DREA
DRE
DR
D

D
DR
DRE
DREA
DREAM
```

They could explain the first part of the output, down to the blank space. Then I asked them to explain the blank space, and finally someone said that it represented the first time the PRINT :WORD command at the end of the procedure had been executed. I congratulated them and brought out the matrushka again.

This time we added a line to our procedure TO PLAY.WITH.DOLL:

```
TO PLAY.WITH.DOLL :OUTER
IF DOLL DOESN'T OPEN STOP
OPEN OUTER DOLL
REMOVE INNER DOLL
PUT A PIECE OF TAPE ON OUTER DOLL
PLAY.WITH.DOLL :INNER
PUT INNER DOLL IN OUTER DOLL
CLOSE OUTER DOLL
END
```

We walked through the whole procedure again, and I asked whether we would ever have the same set of dolls with which we began. The students could see the piece of tape and that the dolls, like the global variable :WORD, had been changed.

A second look at our PRINT.WORD procedure — and the class noticed that the MAKE statement had changed the value of the global variable :WORD so that at this sixth level of the procedure, all that was left was a blank space. Now they could understand why they did not see the word "DREAMS reappearing at the end of the run of the procedure PRINT.WORD. They understood that it had been changed with a MAKE statement, just as the dolls had been changed with pieces of tape.

Even for adult learners making the abstract concrete can be helpful. Visualizing the nested dolls helped them to visualize the abstract concept of flow of control. When we choose our concrete "objects to think with" carefully they can help our students to grasp abstract principles more easily. My students had little difficulty understanding embedded recursive procedures such as trees after this lesson.

Dr. Kahn teaches in the Graduate School of Education at Beaver College. Her courses address the issues of integrating new technology into the existing curriculum and rethinking curriculum in terms of new technology.

Jessica Kahn
1416 Bryant Lane
Meadowbrook, PA 19046

Beginner's Corner

Little Boxes...
by Dorothy Fitch

Greetings!

Welcome to the Beginner's Corner. In this column, we will explore a variety of Logo ideas that can be used by novice users, both adults and young learners. You don't need to know much about Logo to participate. We'll start with a few very simple procedures that use turtle graphics commands and go on from there. Some of you may want to type these procedures and examples exactly as they appear here; others of you will glance at the pictures and use them as ideas for projects that you do in your own way. This column is designed to help make you feel more comfortable with Logo. The ideas will combine easy-to-use Logo commands and procedures with different subject areas or topics.

There will be those of you who may look at these activities and say, "That's nothing new." You're probably right. But I've also heard that there are no new jokes in this world, only people who haven't heard them yet! So, although these may not be brand-new ideas for all of you, there will certainly be a number of you for whom they are new. If I know that certain ideas originated with a particular person, I will give that person credit. I apologize in advance to those individuals I do not mention—I guess we've had the same good idea!

Let's Get Started

This month, we will take a few simple procedures and explore some interesting things to do with them. This is in a sense a tiny microworld. It is very limited—you can make just one shape, a square, and move the turtle around the screen. But, as you will see, there a number of ways in which you can use these tools to explore math and language. Young children can begin to use Logo with just a few commands to remember and to type.

We will be using the following procedures. You will need to enter them into the version of Logo you are using. Although I use Terrapin Logo and Logo PLUS, these procedures will work using any version of Logo. If you are using a Terrapin or MIT version of Logo, enter the editor by typing EDIT. Then type these procedures and press Control-C to exit the editor and define or remember the procedures. If you want to keep these procedures for use another day, you will need to save them on a formatted disk using the SAVE command. Refer to your Logo documentation for more complete information on editing and saving.

```
TO S
REPEAT 4 [FORWARD 20 RIGHT 90]
END

TO F
PENUP
FORWARD 20
PENDOWN
END

TO B
PENUP
BACK 20
PENDOWN
END

TO R
RIGHT 90
END

TO L
LEFT 90
END

TO E
PENCOLOR 0
END

TO D
PENCOLOR 1
END
```

These procedures are very easy to use, once you have entered and defined them.

Type S and press Return to draw a small square, 20 turtle steps on a side.
Type F to move the turtle forward 20 steps without drawing a line.
Type B to move the turtle backwards 20 steps without drawing a line.
Type R to turn the turtle 90 degrees to the right.
Type L to turn the turtle 90 degrees to the left.
Type E to put the turtle in its erasing mode. The turtle's pen color changes to black to match the background color.
Type D to cancel the turtle's erasing mode. The turtle's pen color changes back to white.

Here is a sample sequence of commands that uses all of the procedures. Typing these commands will help you better understand how they work. Be sure to press the Return key after each line given below. Watch the turtle carefully each time you press Return.

```
S F S F
L
S
E S
R
B
S
B
D S
```

When the turtle has finished following the above instructions, there should be one square left in the middle of the screen.

Spend some time experimenting with these procedures until you feel comfortable with them. You can combine as many commands as you like on one line, as long as there is a space between each letter. Use Control-P (or your Logo's equivalent command) to recall the previous line—this saves typing the same commands again and again. And of course, you can use regular turtle commands in addition to these single letter commands.

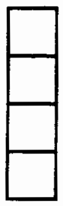
Now that we have these procedures, what can we do with them? Well, lots of things. You can just use them to move the turtle around the screen and create designs made out of the square building block. For young children just beginning to use Logo, it is a way of introducing them to the turtle and giving them some simple commands with which they can explore.

Here is a sampling of activity ideas suitable for a wide range or ages and ability levels.

<u>1. Fun with Patterns</u>

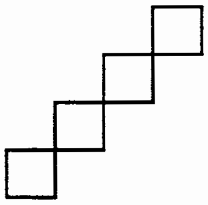
Try to find interesting keystroke patterns using these commands. For example, to make a column of squares, continue this pattern:

```
S F S F ...
```



For a diagonal line of squares, use this pattern:

```
S F R F L ...
```



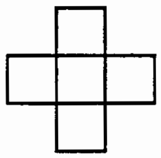
Challenge one child to create a design using a pattern and another try to figure out the pattern of letter commands that was used.

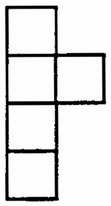
<u>2. Connections</u>

Try to find as many ways of connecting 5 squares as you can. The only rule is that each square must touch another square on at least one side. Below are a couple of examples to get you started.

You can write a procedure (a set of commands with a name) for each figure. Here is a procedure that will draw one of the two examples shown below. You figure out which one!

```
TO FIGURE
S
F
S
F
S
F
S
R
F
L
B
S
END
```

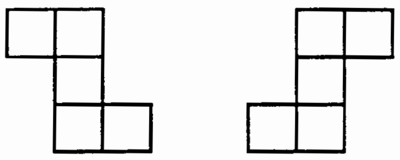




Compare your procedure with a friend's. Does the procedure to draw the same figure use the same commands in the same order? Experiment with different ways of drawing the same figure.

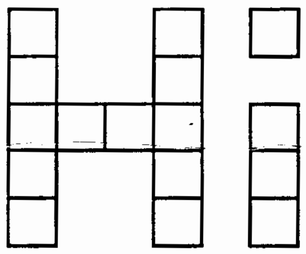
Little Boxes--continued

For a challenge, try to create the mirror-image of one of your designs. Here's a sample pair of symmetrical figures.



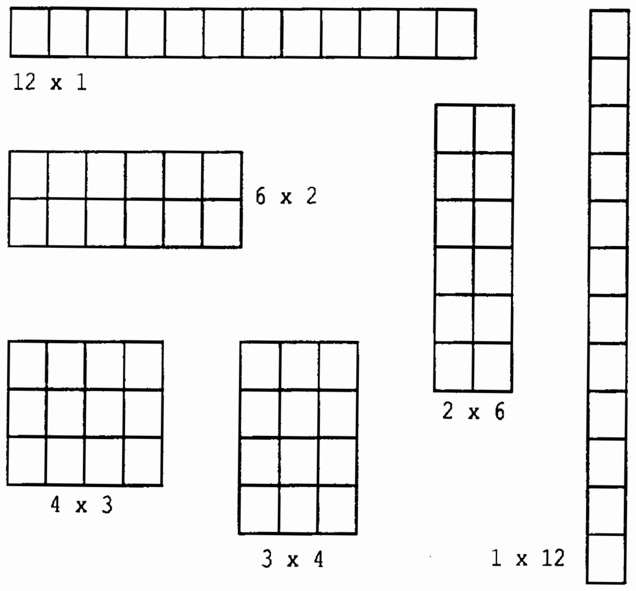
3. Block-letter Words

Try using these letters to create block-style letters. Here's an easy one to try. Students try make their initials or even their first name, if it is short.



4. Factors

Use squares to investigate factors, which makes a good introduction to multiplication. In how many ways, for example, can you place 12 squares in a rectangular form?



Counting the number of squares in each row and column gives the numbers that can be multiplied to reach the product of 12. Experiment with other numbers. Those that can only be represented in a straight line have a special name—prime numbers. Using these squares, younger children can also practice addition and subtraction. (Use the E command to erase a square for "take-away.")

5. Word Puzzle (for Logo PLUS or LogoWriter)

First create a block of squares either 3 or 4 on a side. Fill in the squares with letters that make words when read both horizontally and vertically. Here are a couple of examples:

R	I	M
A	C	E
T	E	N

A	B	L	E
B	R	A	N
L	A	N	D
E	N	D	S

With Logo PLUS, the letters you enter in the GWRITE mode will be centered perfectly inside the squares if you first move the turtle back a few steps. Before drawing any squares, type the following:

```
DRAW
PENUP
BACK 4
PENDOWN
```

Then press Open-Apple-W to enter the GWRITE mode so you can type letters inside the squares to make the puzzle.

6. Filling in the Squares

If your version of Logo has a FILL command, you can create a procedure that makes it easier to fill in a square.

```
TO COLOR
PENUP
RIGHT 45
FORWARD 10
PENDOWN
FILL
PENUP
BACK 10
LEFT 45
PENDOWN
END
```

If your Logo does not have a FILL command, you can write a procedure that colors in the square the way you would with a crayon. Try writing the procedure yourself, then look at this example if you need some help.

```
TO FILL
REPEAT 10 [FORWARD 20 RIGHT 90
   FORWARD 1 RIGHT 90 FORWARD 20
   LEFT 90 FORWARD 1 LEFT 90]
END
```

Set the turtle's pen to the color you want before giving the COLOR command. Use a combination of empty or shaded squares to make interesting quilt patterns, more complicated word puzzles, abstract art designs and so on.

These are just a few ideas for explorations using small squares. I'm sure that you will be able to come up with many, many more!

Dorothy Fitch has been the Director of Product Development at Terrapin, Inc. She first become involved in educational technology in 1981 when the school where she taught music received its first computer. Since that time, as a consultant she has provided schools with inservice training, curriculum development and software customization; taught a number of college courses; and directed a computer classroom for teachers and students. She has also coauthored *Kinderlogo*, a single keystroke Logo curriculum for young learners, and created the *Logo Data Toolkit*. Through her work at Terrapin, she has presented at many local, regional and national conferences, edited many of Terrapin's curriculum materials, and brought Logo PLUS to life. She can be contacted at Terrapin's new address:

Dorothy Fitch
Terrapin, Inc.
400 Riverside Street
Portland, Maine 04103

Logo LinX

Mathematical Magic on the Turtle's BK

by Judi Harris

Rumor has it that a large turtle was responsible for the very first square. As was inscribed in *yih king*, a Chinese book written more than three thousand years ago, a large tortoise crawled out of the Yellow River with some strange markings on her shell. When translated into numbers, the markings look like this:

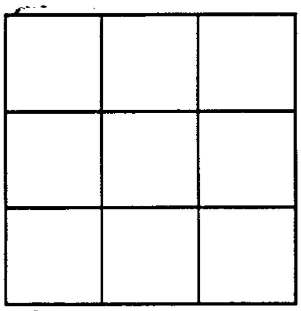
4	9	2
3	5	7
8	1	6

This *MAGIC SQUARE* is assumed to be the world's oldest number mystery. It was used as a divining tool in many Eastern cultures, and during the Middle Ages in Europe, it was believed to drive away disease and attract good fortune. Its apparent magic is reflected in the placement of the digits 1 - 9 so that every row, column, and diagonal sum to the same total.

Magic squares are now used as exercises in mental calculation and motivated practice in mathematical problem-solving. As such, they can be especially effective teaching tools in the beginning of the school year. When used for addition review, paper-and- pencil media are perhaps most appropriate for magic square work. But if you would rather emphasize the problem-solving aspects of magic square work (play?), some simple Logo tools can assist your students' explorations of these ancient puzzles.

Squaring Off

Using several Logo tool procedures, we could program the computer to keep track of students' digit placement trials as they experimented with magic square puzzles. Each time they placed a number in a magic square cell, the computer could show them current row, column, and diagonal totals. This may facilitate puzzle work, since it is difficult for many students to keep nine number placements in mind at once, and having to draw a magic square grid for every attempt at puzzle solution is inconvenient. The grid itself is, in fact, an interesting Logo challenge.



Here's one solution to that challenge. All procedures are shown in Apple LogoWriter 2.0, but minor syntactical adjustments can be made to adapt the code to other versions of the language.

```
TO GRID
CG
CT
PU
SETPOS [-75 -70]
PD
REPEAT 4 [FORWARD 150 RIGHT 90]
REPEAT 2 [FORWARD 50 RIGHT 90 FORWARD
   150 BACK 150 LEFT 90]
REPEAT 2 [FORWARD 50 RIGHT 90]
REPEAT 2 [FORWARD 150 BACK 150 LEFT
   90 FORWARD 50 RIGHT 90]
REPEAT 2 [FORWARD 150 RIGHT 90]
END
```

For ease of use, cells, rows, and columns can be labelled as follows.

A	B	C	___
D	E	F	___
G	H	I	___
___	___	___	___

Values for individual cells and totals for rows, columns, and diagonals can be stored in global variables that are initialized with the following two procedures:

```
TO INITIALIZE.VARIABLES :LIST
IF EMPTY? :LIST [STOP]
MAKE FIRST :LIST 0
INITIALIZE.VARIABLES BUTFIRST :LIST
END

TO VARIABLE.NAMES
OUTPUT [VALUE.A VALUE.B VALUE.C
   VALUE.D VALUE.E VALUE.F VALUE.G
   VALUE.H VALUE.I VALUE.ROW.1
   VALUE.ROW.2 VALUE.ROW.3
   VALUE.COLUMN.1 VALUE.COLUMN.2
   VALUE.COLUMN.3 VALUE.DIAGONAL.1
   VALUE.DIAGONAL.2]
END
```

These two procedures are combined and executed with the command

```
INITIALIZE.VARIABLES VARIABLE.NAMES
```

which assigns a preliminary value of 0 to each of the 17 global variables named in VARIABLE.NAMES.

The Turtle's Accountant

To use the magic square procedures as a puzzle-solving aid, students need only type the letter of the square in which they would like to place a digit, then type the number itself. A separate procedure for each magic square cell can be written, structured like this:

```
TO A
SETPOS POS.A
LABEL :VALUE.A
TYPE [NUMBER?]
MAKE "VALUE.A FIRST READLISTCC
SETPOS POS.A
LABEL :VALUE.A
TOTAL.ROW.1
TOTAL.COLUMN.1
TOTAL.DIAGONAL.2
CC
END
```

POS.A is a subprocedure that outputs the screen position of the printed value inside cell A.

```
TO POS.A
OUTPUT [-45 55]
END
```

Notice that the turtle is directed to set its position to POS.A and print the value of the global variable "VALUE.A *twice*; once to erase the value that was printed there previously, and then again to insert the new value. Notice also that every time a value is changed in a cell, the corresponding row, column, and diagonal sums are adjusted with procedures structured like TOTAL.ROW.1, TOTAL.COLUMN.1 and TOTAL.DIAGONAL.2.

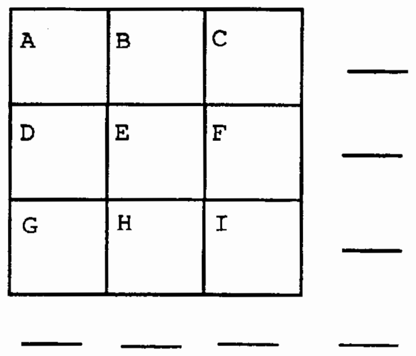
```
TO TOTAL.ROW.1
SETPOS POS.TOTAL.ROW.1
LABEL :VALUE.ROW.1
MAKE "VALUE.ROW.1 :VALUE.A + :VALUE.B
   + :VALUE.C
SETPOS POS.TOTAL.ROW.1
LABEL :VALUE.ROW.1
END
```

POS.TOTAL.ROW.1 is a procedure that outputs the screen position for printing the total of the cells in row 1, written similarly to the procedure POS.A.

You may be thinking that there must be a more economical way to write this program than to have nine almost-identical subprocedures for assigning values to the nine cells, and eight nearly-duplicate subprocedures for tallying row, column, and diagonal totals. You're right. Please let that be a Logo challenge to you.

Magic Turtle Tallies

Let's say that you would like to arrange the digits 1, 2, 2, 5, 6, 6, 7, 8, and 8 in the magic square below so that all rows and columns sum to the same total.



You decide to place the 7 in the center cell, so you type:

```
E
```

and press the Return key. The computer responds:

```
NUMBER?
```

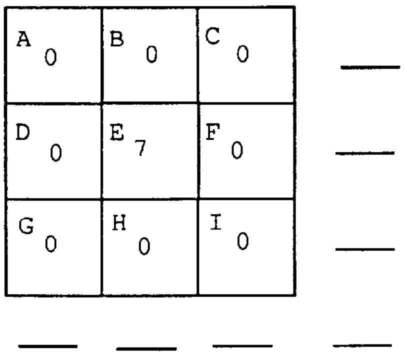
Mathematical Magic on the Turtle's BK--continued

and you type:

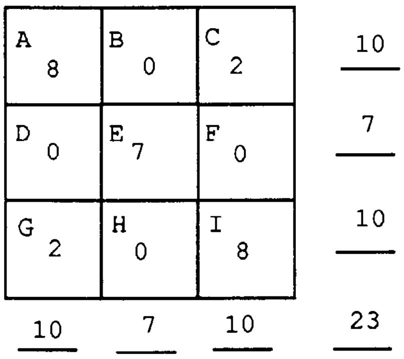
```
    7
```

and press the Return key again.

Now the screen looks like this:



You then try to balance the values of the rows and columns by placing the 8's and 2's in mirror-image placements, like this,



and the computer automatically adjusts the sums to incorporate your digit placements. Now only 1, 5, 6, and 6 remain to be positioned. Can they be arranged in the four remaining boxes so that all row and column totals are the same? If not, cell values for blocks A, C, E, G, and I can be reassigned.

Similar explorations can be conducted with the following digit lists.

List A:	1	1	1	2	2	2	3	3	3
List B:	0	0	1	1	2	3	4	4	6
List C:	2	4	5	5	6	7	8	8	9
List D:	1	1	2	2	3	3	4	5	6
List E:	0	0	1	1	2	2	5	6	7
List F:	2	4	4	4	6	6	6	8	8

In true Logo style, most have several correct positional puzzle solutions.

Magic Code By Mail

Hopefully, you now have enough information to help you to construct a set of Logo magic square tools so that your students can use them for problem solving. The digit lists are by no means exhaustive; I would encourage you to challenge your students to devise other lists (perhaps with two- and three-digit numbers) that will fit the magic square pattern. These can be saved in the computer's memory along with the tools so that other students can use them.

```
TO JASON'S.PUZZLE
OUTPUT [1 2 3 4 5 6 7 8 9]
END

PRINT JASON'S.PUZZLE
```

If you don't have the time or desire to code the rest of the tools, I would be glad to send you a set in either Apple LogoWriter 2.0 or IBM Logo format. Just send a 5.25" disk in a self-addressed, stamped disk mailer to me at the address below. Be sure to include a note specifying which version you would like me to copy onto the disk and return to you.

Judging by the antiquity of this mystical number puzzle, it seems that the turtle has had a long history of motivating active inquiry by revealing the magical in the mathematical. Logo use can certainly preserve and extend this delightful tradition.

Judi Harris taught students in Philadelphia-area elementary through graduate schools to use computers in teaching and learning for six years. She now does similar work at the University of Virginia, where she is completing her doctoral work in Instructional Technology. She can be reached at

Judi Harris
621F Madison Avenue
Charlottesville, VA 22903
CIS: 75116,1207
BitNet: jbh7c@Virginia

"Technology will not transform education to meet the needs of society if teachers do not learn to use it as both an instructional and a mangement tool."
---The National Task Force on Educational Technology

Are computers in <u>your</u> schools not making <u>the impact</u> you had expected?

"At long last! Having the master himself, Seymour Papert talk to teachers about Logo and children and education through video is second only to cloning him."
---Tessa Harvey, Deputy Superintendent District #3, New York

"Inservice educators should treasure ON LOGO because it will make them look good. This is a stellar example of using technology--in this instance, video tapes--well. It not only brings the wisdom of Papert to teachers--but captures the sparkle and immediacy of this master teacher and delivers it to his colleagues anywhere in the world."
---Jim Mecklenburger, Director of ITTE, Nat'l School Boards Association

MIT's SEYMOUR PAPERT HELPS YOU AND OTHER TEACHERS increase the understanding and expertise necessary to make computers effective tools in schools. Now you can bring the "Father of Logo" into your district as a personal instructor with the first video-tape Logo learning system for teachers. Papert's demonstrations on ten half-hour video tapes are designed to help teachers better use LogoWriter or traditional Logo to teach **problem-solving and thinking skills, analyze learning styles and understand how knowledge is best appropriated.**

GRADUATE CREDIT AVAILABLE THROUGH THE UNIVERSITY OF OREGON! The materials are perfect for in-service training, but why not couple the classes with an opportunity for credit? Begin the independent study course anytime and anywhere. Reduced tuition is available for six or more enrollees. Once purchased, tapes and materials may be used over and over again.

TO ORDER: Specify VHS or Beta format. Three-quarter inch tapes $100 additional and available to duplicating license purchasers only. Ten-day preview. Allow three weeks for shipping. Enrollment information on credit opportunities included. Duplicating offer made only to school districts and universities for use within institution.

School districts may acquire one copy of the ten-part video series

for $599

(or about $59 per tape)

or obtain master tapes and a district-wide duplicating license

for $1295

(which includes over 200 pages of copyable support material)

media microworlds inc.--12 clayton terrace--st. louis, missouri 63131--(314) 725-8715

Logo & Company

Logo by Any Other Name ...
by Glen L. Bull & Gina L. Bull

This marks the eighth year we have written a *Logo Exchange* column. The emphasis of the column has changed from year to year. During the early years of the *Logo Exchange*, the column was titled "Microworlds." Later we turned our interest to lists in Logo, in a column titled "Listful Thinking". (Some of these early columns are collected in *Logo: Theory & Practice*, edited by Dennis Harper, just published by Brooks/Cole Publishing Company.)

More recently we have titled the column "Logo Connections" as our interest turned to connections between Logo and other kinds of software and peripherals. Logo Connections has been about using Logo in combination with Appleworks, and paint programs, and using Logo with videodisc players, and touch screens, and robotic applications such as LEGO-Logo. Later this year a special issue of the *Logo Exchange* will be devoted to the theme of Logo Connections. If you are using Logo in conjunction with other kinds of software or peripherals, your contribution is solicited. The deadline for submissions is mid-October, but you may want to send a note to Sharon Yoder (the *LX* editor) describing your idea first.

This year we are broadening our interest to include not only Logo, but "Logo-like" applications. Hence, the change in the title of the column to "Logo & Company." From the beginning Logo has been as much about a philosophy of learning and a culture of ideas as it has been about a specific computer language. In *Mindstorms* Papert talks at length about the ideas embodied in juggling, and notes that one Logo teacher, Howard Austin, adopted analysis of juggling as the topic of his Ph.D. thesis.

At the SIGLogo meeting at this year's National Educational Computing Conference, one of the participants described the most memorable events from the first East Coast Logo Conference (ECLC, 1986). These events included:

- recursive songs
- juggling lessons o contra dancing
- mirrors
- music created with vacuum cleaner hoses
- a guitarist in a bat suit

Interestingly, many of the memorable events included more than Logo procedures and applications. These "Logo-like" activities were included in the conference by design rather than by accident. Later Trent Batson, who discussed his work with networked word processors at ECLC, mentioned that one of the best presentations that he has ever seen was Bob Tinker's demonstration of microcomputer-based laboratories at the same conference.

Sometimes a phrase can create an entire industry, as in the case of the term "desktop publishing." Microcomputers had been utilized in typesetting and publishing for some time before the term was coined, but the phrase crystallized the concept. No comparable phrase has yet been developed for the class of "Logo- like" activities, but there's a definite genre of learner-based tools. This class of software is usually distinguished by control in the hands of the learner, as opposed to CAI applications in which events have been pre-decided by the programmer.

One early book on educational computing described the division as: Tutor, Tool, Tutee. "Tutor" referred to the use of the computer as a teacher, while "Tutee" was used in reference to Papert's description of Logo as a system in which the child teaches the computer. (One teacher jokingly identified a fourth use of the computer as a "toupee," in which technology is used to conceal the fact that nothing much is going on.) However, the term "tutee" never become a popular descriptor for the class of "Logo-like" activities, and as far as we know, there is no word in the English language to describe this class of activities.

In fact, we hereby sponsor a contest for the best descriptor for the class of "Logo-like" software. We offer as a prize for the best word or phrase describing the class of "Logo-like things" the best (in our opinion) Logo book: *Visual Modeling with Logo: A Structural Approach to Seeing*, by James Clayson. This book is the second in Paul Goldenberg's "Exploring with Logo" series, and is available from MIT press. To begin a new decade of Logo, we will offer a copy of this book to the person submitting the best descriptor received by January 1, 1990. The winning entry will be selected by a panel of judges consisting of Paula Cochran, Tom Lough, and Stephen Bull (age 6). Send entries to:

Logo & Company c/o Glen Bull, 405 Emmet Street,
Charlottesville, VA 22903

Until one of our readers develops a better term, we are left with the title "Logo & Company" to describe the class of "Logo-like" activities which we hope to discuss this year. It is worth reviewing some of the changes which have occurred in educational computing during the last decade. As the 1980's began, Visi-Calc and other electronic spreadsheets constituted "tool software." Pilot and BASIC could be used for development of tutorial programs, and Logo represented the entire class of "Logo-like" software. The IBM personal computer did not exist, and neither did the Macintosh.

During the intervening decade, an explosion of educational software has occurred:

- development of paint programs: MacPaint, PC Paint, Mousepaint (for Apple II)

- creation of talking word processors: Talking Textwriter, Kidtalk, Smoothtalker
- use of educational telecommunications: National Geographic Kids Network
- clip art and scanners: MacVision and Computer Eyes (for Apple II)
- hypermedia applications: Hypercard, Linkways, and Hyper Studio (Apple II)
- multimedia programs: VCR Companion and Slide Shop

At the beginning of the decade there was only one Logo-like application: Logo. At the end of the decade there are many Logo-like applications to choose from.

There are two possible responses to this altered state of affairs. One response is to become a member of the steadily shrinking band of Logo "purists" who use Logo, and only Logo (the computer language), no matter what the application. We doubt that any of the *LX* readers fall into this category. The other response is to use a mix of Logo-like tools, including Logo, for a range of educational activities. It is the second approach that we wish to explore in this column. If you are using some "Logo-like" software in your class, we'd like to hear about it.

At some point in the 1990's we would even like to see a change in the title of the journal to reflect this broader class of activities. Originally the *LX* was the *National Logo Exchange (NLX)*. The name was changed when the *National Logo Exchange* and the *International Logo Exchange (ILX)* were combined. At present a new name is not possible because there's no name for "Logo- like" applications - yet. (Somehow we don't think Sharon would agree to the "Logo-like Exchange", or *LLX*. [You might be surprised! ed.]) However, one of you may be thinking of a possible name even as you read this.

Two rapidly expanding areas in educational computing are "hypermedia" and "multimedia" applications. These terms refer to overlapping concepts.

<u>Multimedia</u>
The term "multimedia" refers to the combined use of several different media, including computers, videotape, slides, digital and synthesized speech, videodiscs, and CD-ROMs.

<u>Hypermedia</u>
The term "hypermedia" refers to the ability to travel through a medium in a nonlinear fashion. For example, to get to the middle of a videotape, it is necessary to fast forward through the beginning of the tape first. In contrast, it is possible to go directly to any frame on a videodisc. A book is a linear medium; a Logo adventure story is a non-linear application.

Hypertext is a special category of hypermedia. In a hypertext application, it is possible to select a word or phrase which takes the reader to a new screen with an expanded explanation or a further exploration. For example, if the word "Spain" were selected in the sentence, "The rain in Spain falls mainly in the plain.", a screen might appear with a map of Spain and a paragraph or two on its history and demographics. For a further discussion of hypertext, see Eadie Adamson's column on "Hypertext in Logowriter" in the May 1989 issue of *LX*. If this were hypertext, you could point to the title, and the article would pop up on the page. Instead, in a linear medium, you must thumb through back issues of *LX* to retrieve Eadie's article.

Many hypermedia and multimedia applications were first developed on powerful machines which are scarce in most public schools. Hypercard for the Macintosh is perhaps one of the best known of these applications. Although an increasing number of Mac's are finding their way into schools, they are still a rare commodity in most classrooms. Similarly IBM's Infowindows system (a powerful multimedia system) is more often found in a business environment than a classroom.

However, ideas first developed on machines with a megabyte or more of memory are migrating to classroom computers. For example, at NECC 89 we saw a demonstration of a program by Tutor Tech which embodies many of the concepts in Hypercard on the Apple II. A second program of this kind, HyperStudio, has also been released for the Apple II, and we talked with the author of a third such program which is nearing completion. Thus sophisticated applications first developed on state-of-the-art machines are having a healthy influence on classroom computing.

It is not necessary to have the latest computer and a room full of equipment to participate in this revolution. Next month's issue of *LX* will be devoted to the theme of "Just for Beginners". Our column for that issue will be devoted to "Multimedia for Beginners". We will discuss how you can use Logo and a videocassette recorder (VCR) to begin experimenting with multimedia applications in your classroom.

Glen is a professor in the University of Virginia's Curry School of Education. His BitNet address is GLB2B@ VIRGINIA. Gina is a programmer analyst for the University of Virginia Department of Computer Science. By day she works in a Unix environment; by night in a Logo environment. Her BitNet address is RLBOP@VIRGINIA.

Glen and Gina Bull
Curry School of Education
Ruffner Hall
University of Virginia
Charlottesville, VA 22903

MathWorlds

edited by
A. J. (Sandy) Dawson

Through a series of email messages initially between Tom Kieren and myself, but later joined by David Pimm, I suggested to Tom and David that they should sit down one day and tape record their conversation about the relationship between mathematics and Logo. This they did one evening whilst reclining in Tom's hot tub in Edmonton. They mailed the tape to me, and I had it transcribed. I did some editing on it, and then emailed the result to David in Britain and Tom in Edmonton. Within hours David returned a corrected copy to me by email. Tom's revisions came a few days later. Because of the limitations of space for any one column in *LX*, we have divided the conversation between David and Tom into three sections, the first of which follows.

Two Turtles in a Hot Tub

(Recorded discussion about Mathematics, Logo and Language between David Pimm (DP) and Tom Kieren (TK) held on the 16th of May, 1989 in Edmonton, Alberta, Canada)

Part One

What do Logo learners learn, and who is that does the teaching?

(TK) Do pupils learn any mathematics when doing Logo? It's a reasonable question to ask and many people ask it. It has been said that Logo is an ideal place to learn some mathematics. Nonetheless, the question remains.

(DP) And I think the subsidiary question is who, if anyone, is teaching in an Logo environment and what it is they intend to teach.

(TK) Of course, Papert talks about Logo as being a situation in which a lot of mathematics can be learned. He talks about it being "Mathland", and I think research over the past several years has more or less substantiated that claim. In fact, it's been shown in all kinds of different Logo situations that kids do learn mathematics; such standard stuff as transformation geometry, or ideas about proportionality, or particularly the processes of mathematics like analytic thinking have been looked at by many different people and I think reasonably well substantiated. I think it is probably true to say that mathematics is learned in that environment.

(DP) One question for me is, "Is that intended learning or merely circumstantial learning which comes about from interacting with the machine in this particular way."

(TK) My own experience with it is that it is really very intentional. Let me just mention a couple of instances. Susan Ludwig at the University of Alberta, Edmonton, Canada, did a thesis on teaching motion geometry in a Logo environment. (Ludwig) She invented what I suppose would be called a microworld. Much more simply put, she had kids work with particular primitives that related to geometric transformations, and through a number of hours of work with those primitives it certainly was true on rather ordinary measures of geometry that the students knew a lot of geometry from working in that environment. They had many, many geometrical experiences. I don't think in this case that the learning was incidental. Clearly, if you look at the kinds of tasks posed by Joel Hillel and Carolyn Kieren of Concordia University and the University of Montreal, respectively, then you see that those aren't tasks that are incidental by any means. (Hillel, 1987; Kieren et al, 1986/87) They are very structured tasks, delimiting tasks, where the students have to come to grips with some mathematics in order to accomplish them.

(DP) One of things that interests me is how they actually come to grips with mathematics that are often built into the primitives. The 'multiple turtles' package which the Association of Teachers of Mathematics Logo group has produced in England has a tremendous amount of mathematics actually packed into the primitives themselves. For example, the FACE command where one turtle can be made to face in the same or opposite direction relative to another turtle. There is a lot of trigonometry underlying that, but whether pupils get any access to that through working with primitives alone I am less sure. It strikes me as a similarity to Michael Spivak's book *Calculus on Manifolds* in which the central aim is to prove Stokes' theorem on manifolds. (Spivak) But all the power is in the definitions. The theorem itself is a virtual triviality to prove, because he set up such powerful definitions. It seems to me that the move in mathematics towards increasingly powerful and sophisticated definitions, into which students can have little insight is actually quite analogous to having very powerful primitives that pupils can use very easily without any access to the mathematics that underlie those primitives.

(TK) In my reading of the research, I think there is some cause for concern in that respect. Again, Hillel and Kieren have found that it is not easy to provoke analytic thinking; that is, where students are actually trying to see what mathematics is there. Rather, the students use the primitives almost in a stepwise fashion and construct, or build, or do the geometric things, I wouldn't say in a non-thinking way, but in a way in which it doesn't get them at that which you were calling the hidden mathematics. Hoyle and Noss, in England, are doing some things with proportionality, and students have to see for example, constructing different size letter "N" that there is a proportional relationship between the cross piece of the "N" and the two side pieces of the "N". In that case, the students seem to come in a stepwise fashion to understand proportionality . They first see it as the link of the side piece plus a selected little bit , so they don't really have proportionality, but have a pseudo proportionality. It's not simple to come to the general idea. I think that kids see that there's something there; to get what is there is another question, and I think that is an interesting issue.

(DP) What they're gaining experience is an awareness of what later will come to be the trigonometric functions, so I think, in a sense, they could be working on trigonometry when working on the end task. I think that is one of the strengths of Logo.

(TK) It seems to me that we are ready to tackle the questions of who is doing the teaching; if anybody is doing the teaching in this environment! One might say that the task itself is doing the teaching, or carry's the teaching load. The human teacher is standing back from the situation and instigating the tasks, and probably wandering around and watching students interacting, and in Hoyle and Noss's terms, "nudging" the students along ,if you will. It might be said the task does the teaching.

(DP) I find it difficult thinking about tasks "teaching", partly because I have a very human focused view of teaching. Here it may be helpful to think of Michael Stubbs' notion of teaching as metacommunication. (Stubbs) Metacommunication for Stubbs is all those remarks about the content of the discussion which provides, for him, the definition of what teaching is. If we are talking about turtle geometry, for example, there can be a sense in which the turtle can be said to be teaching by means of the error messages which, to some extent, provides both feedback and orientation to the language that the pupil is generating. However, I have a difficulty with 'nudging' as a technical term, because it sounds like teachers are trying to suggest that when they 'nudge' a student "it isn't very much". But you can actually nudge someone and they fall over. The effect of it can be lot more powerful than was actually attended.

(TK) I'm in agreement with your human orientation, but when I was saying that "task" does the teaching, I meant that the children themselves make something of the task and this, its seems to me, has two implications: (1) the teacher can either say, "here are some primitives do whatever you like with this — do something interesting with those primitives", in which case the primitives don't carry much mathematics; or (2) the further activity might possibly carry a lot of mathematics in which case the need for communication by the human teacher is to say, " Look over there, that might be an interesting thing you might do next." That may be what is really is meant by "nudging". It is pointing in interesting directions so that the students might go from where they are now.

(DP) Right! And so an important part of that is what Ronnie Goldstein insists on, which is the importance of the idea that the pupil is free to reject nudges, to reject the metaphoric twist of the shoulders of the pupil which says, "look over there", and they can say no, and twist back and say, "I prefer to be looking over here". (Ainley and Goldstein)

References

Ainley, Janet & Goldstein, Ronnie (1988) *Making Logo Work.* Oxford: Basil Blackwell.

Balacheff, Nicolas (1988) "Aspects of proof processes in pupils' practice of school mathematics". D. Pimm (ed.) *Mathematics, teachers and children.* London: Hodden & Stoughton .

Hillel, J. (1987) "Multiple perspectives and task-analyses of a Logo activity". J. Hillel (ed.) *Proceedings of the Third International Conference for Logo and Mathematics Education.* Montreal: Concordia University.

Kieren, C., Hillel, J. & Erlwanger, S. (1986) "Perceptual and analytical schemas in solving structured turtle-geometry tasks". C. Hoyles & R. Noss (eds.) *Proceedings of the Second International Conference for Logo and Mathematics Education.* London: University of London School of Education.

Kieren, C., Hillel, J. & Gurtner, J. L. (1987) "Qualitative strategies in Logo centering tasks." J. Hillel (ed.) *Proceedings of the Third International Conference for Logo and Mathematics Education.* Montreal: Concordia University.

Leron, Uri (1987) "On the mathematical nature of turtle programming", *Logo Exchange,* 5 (9).

Ludwig, S. (1986) *A Logo/motion geometry curriculum environment.* Masters Thesis. Edmonton: University of Alberta.

Minsky, Marvin (1985) *The society of mind* New York: Simon & Schuster.

Math Worlds--continued

Pimm, David (1987) *Speaking mathematically* London: Routledge and Kegan Paul.

Spivak, Michael (1965) *Calculus on manifolds* London: Benjamin.

Stubbs, Michael (1974) "Organizing classroom talk." Occasional Paper !9, *Centre for Research in Educational Sciences*, University of Edinburgh.

Vergnaud, G. (1981) "Quelques orientations theoriques et methodologiques des recherches francaises en didactique de mathematique" *Proceedings of PME V.* Glenoble.

About the discussants

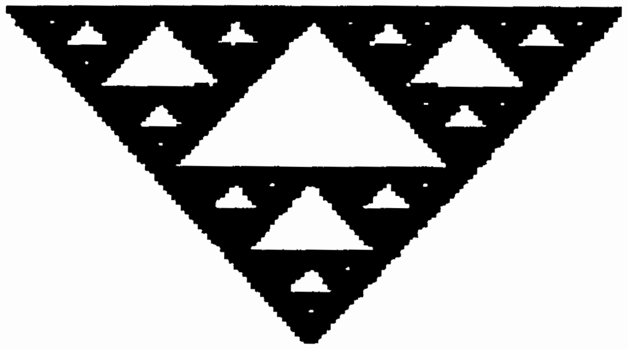
David Pimm is a lecturer in Mathematics Education at the Open University in Britain. Tom Kieren is professor of Mathematics Education, and Distinguished Research Professor, at the University of Alberta, Canada.

A. J. (Sandy) Dawson is a member of the Faculty of Education at Simon Fraser University in Vancouver, Canada. He can be reached electronically through Bitnet as userDaws@SFU.BITNET

Order From Chaos

by Phil Firsenbaum

In the introductory scenes of the the Nova series dealing with chaos as a force in nature (1/31/89)we see a man sitting at a large sketch pad. He has drawn three equidistant points near the edges of the page and labelled each point with a pair of numbers (1,2 3,4 5,6). He has a single die which he begins to toss and with each toss plots a point half way between the last point and one of the original points, one of whose numbers corresponds to the roll of the die. He suggests that continuing this process for 24 hours would yield an interesting result. Not, he suggests as we might expect, a jumble of dots all over the page but a very precise pattern. At this point he proceeds to show the results of a computerized version of this process, and lo and behold, there we see a beautiful pattern consisting of different size triangles very neatly arranged.



Although the type of computer and the language used to write the program were never mentioned, I immediately thought of how it might be done in Logo. A few days later, when I had a some time, I wrote the following procedures, which worked with varying degrees of precision and speed, depending on the size of the shape I used. This solution is written using LogoWriter 2.0 for the Apple, and employs the DISTANCE procedure from the TURTLE.TOOLS page provided on the 2.0 Scrapbook disk. It's a very handy procedure which reports the distance (in turtle steps) between the turtle and another position on the screen, which is required as the input to DISTANCE.

```
to startup
getshapes
gettools "turtle.tools
setup
end

to setup
cg
setsh 2
ht
make "pos1 [-135 85]
make "pos2 [135 85]
make "pos3 [0 -85]
stamp.point :pos1
stamp.point :pos2
stamp.point :pos3
plot :pos1 :pos2 :pos3
end

to stamp.point :point
pu
setpos :point
pd
stamp
end

to plot :pos1 :pos2 :pos3
make "roll random 6
if or :roll = 0 :roll = 1 [pu seth
   towards :pos1 forward (distance
   :pos1) / 2 pd stamp]
if or :roll = 2 :roll = 3 [pu seth
   towards :pos2 forward (distance
   :pos2) /2 pd stamp]
if or :roll = 4 :roll = 5 [pu seth
   towards :pos3 forward (distance
   :pos3) / 2
   pd stamp]
plot :pos1 :pos2 :pos3
end
```

Distance Procedure from the LogoWriter 2.0 Tools

```
to distance :pos
output sqrt (sq(first pos) - (first
   :pos)) + (sq(last pos) - (last
   :pos))
end
```

```
to sq :n
output :n * :n
end
```

I decided to share the problem with some of my more advanced eighth graders. A few students became intrigued with the problem and came up with their own solution! (As a teacher, of course, I find that more exciting than anything I might do on my own.) After experimenting with a number of different custom shapes, without ever finding the "ultimate" shape (apparently due to the resolution of the Apple monitor), they tried modifying the original problem. They rotated the triangle so that the base is no longer parallel to the horizontal of the monitor, they added a number of other points, and most recently they resurrected our copy of Logo II and, using the WINDOW command, wrote a version which was creating the design ten times larger than the original! We haven't yet been able to run this last version long enough to see what it will actually produce, but the experiment goes on.

Perhaps your students would like to experiment as well. I think it might be interesting to see just how long it would take different computers, as well as different version of Logo to complete the design. Why not send the results to the *Logo Exchange* so that we can all share out discoveries with others?

Phil Firsenbaum
Riverdale JHS 141
Bx., NY 10463

About the Cover

This month's cover shows an example of the LOGAME. See the Global Logo Comments column in this issue for more details.

A Report on the LME4 Conference

by Uri Leron

The 4th International Conference for Logo and Math Education (LME4) was organized by the Israeli Logo Centre, and has taken place on July 2-5 in Jerusalem, Israel. (The previous LMEs took place in the University of London and in Concordia University, Montreal.) The participants were researchers working in Logo and math education.

Among the participants and projects were: Sandy Dawson (Vancouver); Andy diSessa (Boxer, UCBerkeley); Feurzeig (Function Machines, BBN, Boston); Joel Hillel (Montreal); Hoyles, Noss ans Sutherland (The Microworlds Project, Univ of London); Uri Leron (The Israeli Logo Centre); Liddy Neville (Sunrise Schools, Melbourne, Australia); Mitch Resnick (LEGO/Logo, MIT).

The conference proceedings include descriptions of the discussion groups, some abstracts and, mainly, papers. The proceedings (249 pages) can be ordered for $15, including surface mail postage, from:

Professor Uri Leron
Department of Science Education
Technion - Israel Institute of Technology
Haifa 32000, Israel.

Following is the detailed contents of the proceedings

Working Groups

Yehoshafat Shafee Give'on: The mathematics of visual recursion: a prelude to recursive programming in Logo.

Nira Krumholtz, Nitza Shafriri: From ideas to practice: A method for analysis of Logo classroom interactions.

Tammy Lapidot: Puzzles in computer science Logo style: Intriguing issues found in Israeli high school students' exams.

Rina Zazkis: Turtle view of geometrical transformations and vice versa.

MIT Media Lab Joint Presentation

Idit Harel: A case study of a young software designer

Mitchel Resnick: LEGO, Logo and math: Learning through and about design.

Ricki Goldman Segall: An ethnographic description of "a culture of mathematics".

Uri Wilensky, Aaron Brandes: The Treasure Game: A Logo-based environment for children's learning about feedback and problem solving strategies.

Paper Presentations

Uzi Armon: Circles and trochoids via POLY and DUOPOLY

Daniel Ben-Sefer: Hebrew grammar explained to non-Hebrew speakers that do know Logo.

Nili Bussel, Ina Berger, Josepha Maoz: Polygons: An experimental math/Logo project.

Jean Cesar: Junior secondary school pupils and solid angle: a theorem in action. Sandy Dawson, David Bell: LEGO/Logo — A study in children's learning.

Tommy Dreyfus, Nurit Hadas: Stereometrix — A learning tool for stereometry

Laurie Edwards: A Logo microworld for transformation geometry.

Wallace Feurzeig: A visual programming environment for math education.

Celia Hoyles, Richard Noss, Rosamund Sutherland: The microworlds project: a meta-report.

Tony Jones: Logo and constructivism in preservice math education: an experimental course and some extensions.

Carolyn Kieran, Joel Hillel: A mathematically oriented computer environment: from conception to implementation Nira Krumholtz: The Israeli Logo Centre: spanning the gap between theory and practice.

Chronis Kynigos: Children's learning of Euclidean geometry with the turtle.

Joao Fillipe Matos: Logo simulations and mathematics: the turtle with sensors.

Liddy Neville: Redefining mathematics education in a Sunrise classroom.

Christophe Parmentier: Who gets something from Logo.

Peter Pereira: Logo and Euclidean geometry.

Ricki Goldman Segall: Videodisc technology as a conceptual research tool for the study of human theory making.

Bruria Sela: The use of music in the teaching Logo.

Anna Sfard: Programmer's ups and downs: teaching top-down strategy with multi-level diagrams.

Nitza Shafriri, Nira Krumholtz: Powerful ideas from theory to practice: A study of Logo classroom interactions.

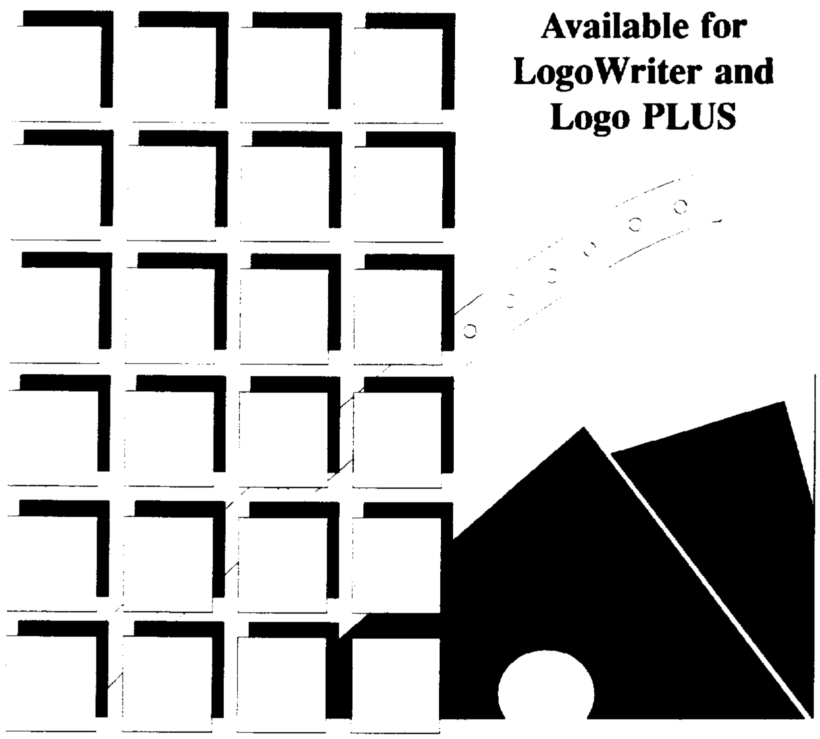
Jose Armando Valente: The use of Logo in public schools in Brazil.

Eduardo Veloso: Logo Geometria: Innovation in the learning of geometry.

Plenary Papers

Andy diSessa: A child's science of motion: Overview and first results.

Yehoshafat Shafee Give'on: Programming in Logo as a mathematical activity.



Available for LogoWriter and Logo PLUS

Introduction to Programming in Logo

by Sharon Burrowes Yoder

Designed for use either in teacher training or in an introductory secondary computer science class. Each section details new Logo primitives or programming concepts and ends with suggested open-ended activities for practice. Numerous appendices include key summaries, a quick reference containing all Logo primitives, and copies of shapes. Specify Logo-Writer or Logo PLUS version when ordering. **$14.95**

Learn from a Leader

Dr. David Moursund's series on
Computer Integrated Instruction for Effective Inservice
is now available through ICCE.

Computer-Integrated Instruction: Effective Inservice for:

- **Secondary School Mathemathics Teachers**
- **Elementary School Teachers**

Based on a National Science Foundation project directed by Dr. Moursund, this series is written for inservice providers. The materials of each title encompass an overview of computers in education and integration of the computer as a tool throughout the curriculum, an introduction to staff development, subject-specific material for inservice providers, and forms to gather information for evaluation.

Purchasers receive a 3-ring binder with a hard copy and a *MacWrite* disk version of the materials, allowing users to easily adapt materials to individual needs.

Each Title is **$40.00** plus **$3.50** shipping. District and Regional site licenses are available.
Available also is a general overview only: *Computer-Integrated Instruction: Effective Inservice* $25.00

To order: contact ICCE, U of Oregon, 1787 Agate St., Eugene, OR 97403 (503) 686-4414

A Logo Challenge...or Two

[Telecommunications has been a wonderful way for me to maintain contact with a wide variety of Logo friends throughout the world. Last spring I shared with you a couple of letters that I received over BITNET. This time I want to share with you a challenge, initially issued by Ken Johnson over the "Logo Friends" network. (You can contact Ken via BITNET at KEN@AIAI.ED.AC.UK.) Editor.]

The Initial Programs

Here is a silly Logo program written in Nimbus Logo:

```
backwards :program :new_name
define putfirst  (putfirst :new_name
rest first text :program) (reverse
rest text :program)

reverse :list
result if emptyq :list [[]] [putlast
reverse rest :list first :list]
```

What 'backwards' does is to reverse a given procedure, e.g. suppose you have a procedure

```
flag :length
forward :length * 3
repeat 3 [right 90 forward :length]
```

and you give the command

```
backwards 'flag 'galf
```

then a procedure 'galf' will be defined for you; its arguments will be the same as those of 'flag' and its content will be the commands in 'flag' in reverse order:

```
galf :length
repeat 3 [right 90 forward :length]
forward :length * 3
```

To understand how it works you need to know only that 'text' is the text of a procedure e.g. 'text flag' (in the example above) would return

```
[[flag :length][forward :length *
3][repeat 3 [right 90 forward
:length]]]
```

Similarly 'define' followed by a list in that format defines a new procedure. If you apply 'backwards' to itself you'll find it leaves itself unchanged!

Ken's Challenges

1. Write this program in your favourite dialect.
2. Write a shorter program that does something (i) useful, (ii) sillier.
3. Find a use for (i) this procedure, (ii) your sillier one.

Ron's Response...

A response quickly appeared from Ron Brasington. (Ron.Brasington @READING.AC.UK)

Well, here is a Mac version (Microsoft) of BACKWARDS anyway. This one has only one argument - the original program name - to save you the bother of reversing it! The cost is an extra pair of procedures, REVERSEWORD and IMPLODE. To be perverse, reverse runs in reverse.

```
to backwards :program.name
define reverseword :program.name
   fput first text :program.name
   reverse butfirst
   text :program.name
end

to reverse :list
if emptyp :list [output []]
output fput last :list
   reverse butlast :list
end

to reverseword :word
output implode reverse :word
end

to implode :list
if emptyp :list [output "]
output word first :list
   implode butfirst :list
end

to test :it
print :it
print reverse :it
end
```

A Reply from Petur...

Soon, another reply came from Petur Thorsteinsson

Ken, here is my reply to your challenges:

1. Write this program in your favourite dialect. (Standard LCSI syntax)

```
to rev :proc :new.proc
define :new.proc fput first text
   :proc rev1 butfirst text :proc
end
```

```
to rev1 :list
if emptyp :list [output []]
output lput first :list
   rev1 butfirst :list
end
```

2. Write a shorter program that does something (i) useful, (ii) sillier.

```
to gsym
local "text
make "text text "gsym
define "gsym lput (list "output "word
   ""g sum 1 last last :text) butlast
   :text
output word "g 0
end
```

gsym returns g0 the first time it is invoked, g1 at its second invocation and so on. It:

a. redefines itself.
b. generates a unique word each time it's invoked.

New Challenge

Write a program 'do.rev :proc' that reverses the activity of a graphical procedure:

If you have the procedure 'flag'

```
to flag :length
forward :length * 3
repeat 3 [right 90 forward :length]
end
```

and say

```
do.rev [flag 100]
```

it should have the same effect as calling a procedure 'galf'

```
to galf :length
back :length * 3
repeat 3 [left 90 back :length]
end
```

with 100 as its input.

Any takers? Send your responses to me at the address given below and we'll publish them in a future issue of *LX* -- as well as sending them to Ken!

Sharon Yoder SIGLogo/ICTE
1787 Agate Street
Eugene, Oregon 97405

Call for Presentations for the Great Lakes/East Coast Logo Conference

Sponsored by: The Educational Computer Consortium of Ohio (ECCO)
Where: Cleveland, Ohio
When: May 4 - 5th, 1990.
(Preconference workshops on May 3.)

Categories of presentations:

Poster Presentations:
These sessions will offer you an opportunity to share a particular idea. Three presentors will be scheduled each hour. Presentations should cover specific teaching ideas, lesson plans, or classroom activities. An Apple IIc or IIe computer and appropriate monitor will be made available during the session.

Session Presentations:
These sessions will be approximately an hour long. Presentations should cover classroom ideas, research project reports, innovative uses of Logo, and Logo connections. An Apple IIc or IIe, overhead projector, and appropriate display screen will be made available.

PreConference Workshops:
These workshops will be six hours in length and will be held on May 3rd. Workshop topics should go beyond beginning turtle graphics and offer participants new ideas and challenges. Either IBM or Apple equipment will be available.

Conference Workshops:
These workshops will be held during the conference and will be approximately 3 hours in length. Workshops can cover either beginning or advanced ideas, but should be limited to material that can be meaningfully presented in a hands-on format in the allotted time. Either IBM or Apple equipment available.

Presentation Proposals are due November 15th, 1989

Contact ECCO for forms to use to submit presentation ideas.
ECCO
1123 S.O.M. Center Road,
Cleveland, OH 44124/
216-461-0800

Adding a Set of Alternative Conditional Primitives to LogoWriter

by Charles E. Crume and Cleborne D. Maddux

In the past, computer languages were written to satisfy the needs of computer programmers. However, as computer education has become more common, ideal languages for use in public schools must meet the needs of teachers and children. Therefore, characteristics such as speed, power, and compactness must be accompanied by simplicity and ease of learning. Logo is exemplary in this regard, since it was designed specifically with children in mind.

There are several dialects of Logo available, however, and later dialects, such as LogoWriter, have incorporated a number of changes to the original syntax. Teachers sometimes complain that favorite primitives from one dialect are not available in another.

The purpose of the this article is to demonstrate how common Logo primitives not included in LogoWriter, can be incorporated by creation of LogoWriter "tool procedures".

Tool Procedures

With most of the early computer languages such as BASIC, FORTRAN, and COBOL, adding primitives is difficult or impossible. In Logo, however, new primitives can be defined by simply writing a procedure to accomplish the desired task. An especially handy feature in LogoWriter is the ability to write tool procedures. Any LogoWriter page can be loaded into memory as a tool procedure. LogoWriter tool procedures are excellent for use with children since they do not appear on the flip side (where procedures are stored and edited). Consequently, they are not routinely accessible for editing (or accidental erasure). Then too, such tools remain in memory throughout a Logo session and take less work space than non-tool procedures.

Using tool procedures in LogoWriter is simple and straightforward. For example, if a LogoWriter page named MYTOOLS contained procedures named TEST, IFTRUE, and IFFALSE, the following command would be entered to load them as tool procedures:

```
GETTOOLS "MYTOOLS
```

As with any LogoWriter primitive, the GETTOOLS command can be entered from the keyboard or incorporated into any LogoWriter procedure. To verify that the procedures have been loaded properly as tool procedures, enter the command:

```
SHOW TOOLLIST
```

LogoWriter should respond:

```
[TEST IFTRUE IFFALSE]
```

TEST, IFTRUE, and IFFALSE Tool Procedures

Most Logo dialects include the TEST, IFTRUE, and IFFALSE primitives. LogoWriter, however, does not include these conditionals. Instead, only IF and IFELSE are provided. The primitive IF allows a list (enclosed in brackets) of actions to be executed only if the condition is true:

```
IF :A = 5
   [PRINT [YES, THE ANSWER IS 5.] ]
```

Some users may wish to designate one list of actions if the condition is true, and another list if it is false. In LogoWriter, the IFELSE primitive must be used:

```
IFELSE :A = 5
   [PRINT [YES, THE ANSWER IS 5.] ]
   [PRINT [SORRY, THAT IS NOT
    CORRECT.] ]
```

In the example above, if variable A equals 5, the actions in the first set of brackets will be executed. If A does not equal 5, the actions in the second set will be executed.

We have found that beginners are often confused by this LogoWriter syntax, and we have often wished that TEST, IFTRUE, and IFFALSE were available. We believe these latter primitives are superior because the logic is more sequential and straightforward, and permits less information to be placed on a single line. In dialects incorporating these primitives, the IFELSE example above would be written as follows:

```
TEST [:A = 5]
IFTRUE
   [PRINT [YES, THE ANSWER IS 5.] ]
IFFALSE
   [PRINT [SORRY, THAT IS NOT
    CORRECT.] ]
```

Therefore, we have written TEST, IFTRUE, and IFFALSE primitives for LogoWriter. These would be entered, named, and saved on a single LogoWriter page. To make these new primitives available, one uses the GETTOOLS command described above. The procedures follow:

```
TO TEST :MYLIST
IF NOT LIST? :MYLIST [(TYPE [TEST
   DOESN'T LIKE] :MYLIST [AS INPUT]
   CHAR 13) STOP]
IFELSE RUN :MYLIST
   [MAKE "TEST.RESULT "TRUE]
   [MAKE "TEST.RESULT "FALSE]
END

TO IFTRUE :LIST.OF.COMMANDS
IF NOT LIST? :LIST.OF.COMMANDS
   [(TYPE [IFTRUE DOESN'T LIKE]
   :LIST.OF.COMMANDS
   [AS INPUT] CHAR 13) STOP]
IF NAME? "TEST.RESULT
   [ IF :TEST.RESULT = "TRUE
   [ RUN :LIST.OF.COMMANDS ] ]
END

TO IFFALSE :LIST.OF.COMMANDS
IF NOT LIST? :LIST.OF.COMMANDS
   [(TYPE [IFFALSE DOESN'T LIKE]
   :LIST.OF.COMMANDS [AS INPUT] CHAR
   13) STOP]
IF NAME? "TEST.RESULT
   [ IF :TEST.RESULT = "FALSE
   [ RUN :LIST.OF.COMMANDS ] ]
END
```

All three of the above procedures require a single input that must be a list (enclosed in brackets). TEST is error trapped to insure that its input (MYLIST) is a list. If not, the error message "TEST DOESN'T LIKE (input) AS INPUT" will be displayed in the Command Center at the bottom of the screen. If the input is a list, TEST evaluates the list (via the RUN primitive) and assigns either TRUE or FALSE to the variable TEST.RESULT.

IFTRUE and IFFALSE are also error trapped to return an error message if their input is not a list. Both of these procedures use the NAME? primitive to verify that the variable TEST.RESULT exists. If it does exist, TEST.RESULT is checked to see if it has been assigned TRUE or FALSE. IFTRUE then runs the list of commands if TEST.RESULT is true. IFFALSE performs similarly if TEST.RESULT is FALSE.

The AND, OR, and NOT primitives as well as internal parentheses may be included in the input to TEST so long as all syntax conforms to LogoWriter rules. For example:

```
TEST [AND (:A = 5) (:B > 3)]
IFTRUE [PRINT "YES]
IFFALSE [PRINT "NO]
```

The parentheses following AND above are optional, just as they would be in any LogoWriter line containing AND. In the example above, if the A variable is equal to 5 and if the B variable is greater than 3, YES will be displayed. Otherwise, NO will be displayed.

Conclusions

We hope the above explanation and sample tool procedures will be helpful to teachers using LogoWriter in their classrooms. We are planning additional LogoWriter tool kits that we believe will simplify the understanding and learning of Logo.

Charles E. Crume, B.S.
Technical Consultant
University of Nevada System Computing Services

Cleborne D. Maddux, Ph.D.
Professor and Chairman
Department of Curriculum and Instruction
University of Nevada Reno

Logo: Search and Research

Learning and Teaching Logo Problem Solving: A Summary

by Douglas H. Clements

Why is problem solving so important to learning? For example, mathematics learning? Because, although students can learn to repeat "facts" and apply procedures, to think mathematically they must build their own mathematical understandings from experience. Problem solving is a critical component of that experience, as much of the building and linking of mathematical ideas occurs when tackling novel problems.

For a year, we've examined Logo problem-solving in detail. In this final column on the subject we'll view the big picture. First, we'll examine a "hot off the press" review of Logo problem-solving research (by authors who apparently understand brevity more than yours truly). Second, we'll ask: When we survey all the research, do any overall suggestions for teaching emerge?

Meta-analysis: A Different Perspective

At the recent NECC convention, Liao and Bright (1989) presented a meta-analysis of research on computer programming and problem solving. Simplified, meta-analysis is a statistical technique for combining the results of several studies ("averaging over" many studies the way most studies "average over" many students). This is done by computing an effect size (ES) for each study:

$$ES = \frac{\text{(Mean of treatment group) - (Mean of control group)}}{\text{Standard Deviation}}$$

These ES's are then averaged. Obviously, this is useful for summarizing. We must not forget it's limitations, however. It can result in combining "apples and oranges," leading to less relevant results. It tends to emphasize numbers more than meaningful interpretation. As a complement to standard reviews (such as those in previous columns), it may nonetheless be worthwhile.

Reviewing a total of 19 studies, the authors found that 84% of the ES's were positive, favoring the computer programming group. The average ES was .41 (an effect usually considered "moderate"), suggesting that students having computer programming experiences scored about 16 percentile points higher on various problem-solving tests than students without these experiences. Variables such as grade level, sample size, and duration of treatment did not significantly affect the size of the ES's (this is unsurprising, given the small number of studies). At the conference, the authors stated that in their most recent version of the meta-analysis, which includes more studies, one variable was significant—Logo programming produced higher ES's than computer programming in other languages.

The meta-analysis did not include many studies we have reviewed over the past year. In addition, like many meta-analyses, it tended not to answer "Why?"

Why Do Some Treatments Generate Larger Effects?

To answer this question, I "re-searched" all the Logo studies reviewed in this column over the past year with this question in mind (Clements & Merriman, 1988). Consistent patterns were shared by most successful Logo teaching environments. Teachers usually would:

- Provide lots of metacognitive prompts and ask lots of higher-order questions. They would emphasize such thinking skills as prediction (If you do RT 120 FD 50, where do you think the turtle will move?), planning (Think about it before you actually try it out. To make the turtle do that, what do we need to do?), monitoring (Did the turtle go where you wanted it to go? Okay, are we on the right track?), and analysis and evaluation (Where do you think you might have gone wrong? Do you know why that's there?).

- Ensure that students are explicitly aware of the strategies and processes that they are to learn.

- Discuss and provide examples of how the skills used in Logo could be applied in other contexts, thus facilitating transfer.

- Encourage transfer directly; for example, "How is that procedure similar to the process we used yesterday to solve the mathematics problem?"

- Provide sufficient time for programming.

- Provide individualized feedback regarding students' problem–solving efforts.

- Ensure that a sufficient proportion of instruction occurs in small groups or 1–to–1 situations.

- Promote both child–teacher and child–child interaction.

- Discuss errors and common misunderstandings.

- Forge links with other knowledge, so that that learned in a programming context becomes integrated with this knowledge.

- In general, mediate their students experiences with Logo to facilitate their use and awareness of problem-solving processes.

Interestingly, two characteristics of the studies did not show a consistent pattern. Despite many claims to the contrary, mastery of programming itself was not a consistently significant factor. Apparently, it is not the programming skills themselves that are being transferred, but the mental processes and strategies. These are what must be well learned.

The second characteristic was the degree of structure in the learning environment. On a continuum from unguided discovery to total teacher-telling, neither extreme appeared successful. Thus, environments that are too loosely or tightly structured may not be effective. However, there was no discernable pattern in the middle of the continuum. So, we can't identify an optimal degree of structure. Perhaps researchers are not being clear about what it is that is being structured. Successful teaching did structure students' experiences. Structure in this sense is not equivalent to the use of lockstep teaching methods, however. It involves facilitating children's employment of, and often awareness of, componential or problem-solving processes. This kind of structure is also not equivalent to "control" of students. Several interventions tightly structured, or controlled, the students' minute–by–minute Logo activities to no avail—what they neglected to do was to structure the environment so as to encourage componential employment. Often children were directed page–by–page through Logo programming worksheets, but were never encouraged to reflect on their activity.

In contrast, successful mediated Logo environments are designed to get children to reflect on their own thinking behaviors, to bring problem-solving processes to an explicit level of awareness, to teach general problem–solving and questioning strategies, and to provide an optimal level of scaffolding to bridge the child's present knowledge with information obtained through a programming experience. As a consequence, they facilitate students' development of the metacomponents of problem solving.

We'll conclude with two vignettes that illustrate the benefits of such environments. A posttest interview of a boy in a recent Logo study was winding down (Clements, in press). As an afterthought, the interviewer asked, "do you think of the homunculi ("little man") when you're solving problems?" The boy replied, "No" (the interviewer inwardly groaned)…"but you always do use them. Like when a problem pops up. But there should be another homunculi. He would get a problem from the problem decider, drop it off to the representer and to the planner. You could even be writing the answer down and this guy could start the problem decider working on a new problem." This boy had independently constructed a function of the monitoring component that we had never discussed: directing the actions of the metacomponents themselves!

The second revealing vignette is taken from Lawler's (1985) study of his son and daughter's almost total immersion in a metacognitive–oriented Logo environment. The two children were discussing the structure of the mind. Mariam proposed an "eraser mind," explaining that thoughts were ideas written on a table that could be erased. Between them, they also postulated "remembering minds" and "talking minds" with "voice boxes" (a reference to a computer's speech synthesizer), and concluded that "You must also have a learning mind, or all your other minds would be empty."

References

Clements, D. H. (in press). Metacomponential development in a Logo programming environment. Journal of Educational Psychology.

Clements, D. H., & Merriman, S. L. (1988). Componential developments in Logo programming environments. In R. Mayer (Ed.), Teaching and learning computer programming: Multiple research perspectives (pp. 13-54). Hillsdale, NJ: Erlbaum.

Lawler, R. (1985). Computer experience and cognitive development: A children's learning in a computer culture. New York: Wiley.

Liao, Y., & Bright, G. W. (1989). Computer programming and problem solving abilities: A meta-analysis. In W. C. Ryan (Ed.), Proceedings of the National Educational Computing Conference (pp. 10-17). Eugene, OR: International Council on Computers for Education.

Douglas H. Clements is an associate professor at State University of New York at Buffalo, Department of Learning and Instruction, 593 Baldy Hall, Buffalo, New York, 14260. He has conducted several research studies on the effects of Logo environments on children's creativity, metacognitive ability, problem-solving skills, and social interaction. Through a National Science Foundation grant, he is currently developing an elementary geometry curriculum based on Logo. His most recent book, *Computers in Elementary Mathematics Education*, was published by Prentice-Hall in September, 1988. An earlier book, *Computers in Early and Primary Education*, was published by Prentice-Hall in Spring, 1985. (E-Mail: CIS: 76136,2027, BitNet: INSDH@UBVMSA)

Global Logo Comments

Edited by Dennis Harper
University of the Virgin Islands
St. Thomas, USVI 00802

Logo Exchange Continental Editors

Africa	**Asia**	**Australia**	**Europe**	**Latin America**
Fatimata Seye Sylla	Jun-ichi Yamanishi	Jeff Richardson	Harry Pinxteren	Jose Valente
UNESCO/BREDA	Faculty of Education	School of Education	Logo Centrum Nederland	NIED
BP 3311, Dakar	Toyama University	GIAE	P.O. Box 1408	UNICAMP
Senegal, West Africa	3190 Gofuku	Switchback Road	BK Nijmegen 6501	13082 Campinas
	Toyama 930 Japan	Churchill 3842	Netherlands	Sao Paulo, Brazil
		Australia		

During the annual three month *Logo Exchange* hiatus, numerous international Logo reports come my way either through the postal system to the Virgin Islands or at conferences I attend. During the last week in May I had the opportunity to visit Bulgaria as part of the Children in the Information Age Conference in Sofia. I was also an instructor for a one-week UNESCO sponsored workshop for educators from developing nations in the Black Sea resort of Varna. The level of Logo sophistication and planning for the use of computers in the K-12 schools of Bulgaria is very impressive indeed. Elementary and secondary schools that I visited were using Logo mainly in the mathematics curriculum and in special Lego/Logo classrooms.

Bulgarian Logophiles are not only doing a tremendous job spreading Logo throughout their country but also training educators in other Eastern Block nations. Especially noteworthy are Logo projects in Hungary, Poland and the Soviet Union. Countries such as Sri Lanka, Nigeria, Kuwait, Algeria, Egypt, Zimbabwe and Vietnam (yes, Vietnam!) are all planning Logo pilot studies in the near future. Working with educators from these countries in Varna and developing their proposed pilot projects was rewarding for me. It is gratifying to know that Logo is entering some nations before BASIC and Pascal.

This month's column reports on Logo Activities in Senegal and the Soviet Union.

Senegal

Recently, the Association of Microcomputing Clubs in Senegal, the Commission for Informatics, and the Ministry of Youth and Sports organized a three-week Logo camp in the country's second largest city, Thies. This was the first time that Logo was used in a city other than the capital of Dakar. This is seen as an important step towards decentralization in the country.

Fifty children from ten to fourteen years old attended the activities. There were two different groups:

1. One group of 30 children was selected according to their high school performance and represented the 10 regions of Senegal. These children were provided with complete scholarships to attend the Logo camp.

2. One group of 20 children whose parents were able to pay an admission fee, attended. These children were admitted to the camp regardless of their school performance.

The objective of this selection was to encourage students with exemplary school grades to encourage slower students to work more seriously.

Computer manufacturers and dealers in Dakar made available ten computers for the camp. Besides experienced Logo instructors, the camp personnel included art, physical education and music teachers whose purpose was to give participants a well-rounded collection of activities. As there were only ten computers, students were split into groups and while one group was working with Logo, others did dance, music, games, etc.

By the end of the three weeks, the children and their parents felt that there should be some follow up activities. This raises a big problem: how is it possible to organize another session for them and how can these activities be expanded to additional children? Of course, the major obstacle for Senegalese authorities is finding the necessary resources.

Another point made clear was that many children participating in the class wanted to go beyond this experience; some parents were even prepared to contribute financially to these types of activities. Whereas the government does not have the means to provide the adequate technological environment for all children, parents and businesses might be able to take initiatives to help their children.

On another Senegal front, the long running Logo Project is being revised to serve as a precursor for a larger project introducing computers into all levels of the Senegalese educational system — elementary through the university. The first phase of this large scale project will cover elementary and secondary schools.

The Computers and Education Laboratory will be used to test and evaluate innovative educational materials, conduct research, sensitize people about the use of computers in education, train teachers in the use of computers in their classrooms evaluate experiments and software, and develop new educational software. The experiments will take place within 40 selected schools in the ten different regions of Senegal. A documentation center will be created and maintenance services provided.

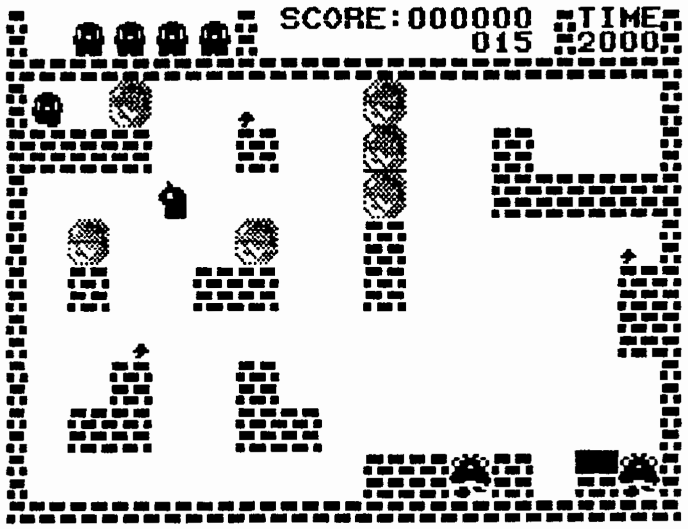
A one-day consultation meeting was organized to study the new project document and set up strategies to launch it. Issues about testing of available educational software, education planning, and the training of administrators, teacher trainers, teachers, and evaluators was discussed. Discussions also centered on questions such as (1) where and when the experiments will take place, (2) how and with what means the experiments will be conducted, and (3) who will be involved?

A consultative committee to deal with pedagogical, administrative and financial problems will be set up to ease the coordination as well as follow up on project activities.

In conclusion, Logo has had a major impact in a large scale decision by the Senegalese government to reform the education system by introducing computers. The plan now calls for the use of Logo on a wider scale. This means a greater challenge for Senegalese Logo teachers and trainers as the use of Logo spreads throughout the country and into every school. It is hoped that teaching methods learned through the Logo Project will be used by many teachers in the rural areas of the country to promote creative thinking. An additional burden on Senegal's Logo community is that neighboring West African countries are now looking at Senegal to help them in their education development.

The USSR

This international column now moves from a large country-wide project in Africa to a small Logo program developed in Leningrad, U.S.S.R. This report deals with a new kind of educational software whose main idea is to connect Logo with an ordinary computer games named LOGAME. LOGAME gives students the opportunity to debug algorithms in order to win the game. This type of software may be used both in computer science classes as well a home video game.



This strategy was developed because it was felt that (1) most students prefer to play computer games to doing homework, and (2) working with Logo is more useful than strictly playing games.

School children work with the system's Logo editor while programmers can have access to the Logo primitives file. The main parts of this microworld are the database and the knowledge base. The database contains static information about external parameters of the microworld. It may be edited by two categories of users: programmers and by teachers. The knowledge base contains algorithms (rules) of interaction of the microworld and may be transformed only by the teacher.

This LOGAME microworld is an imitational representation of a real object, situation or dynamic environment. It is interesting to investigate how students hypothesize using this program and how these hypotheses are verified. LOGAME has been found to be useful because is allows students to investigate algorithms of object behavior, change these algorithms, achieve concrete results. LOGAME has been especially popular with children in the 11-14 year old age group.

In short, the game involves students moving the chief hero, Flappy, through barriers to reach a final destination by changing the Logo knowledge base. In essence, the students play the game with their Logo partner. LOGAME contains a

Global Logo Comments--continued

problem solving, the pupil learns these tasks in a practical manner. These tasks deal with course instructions that include procedures, variables, functions, cycles, and subroutines. Initial research results with LOGAME indicate that it has potential in the Informatics 5-8 course given in Soviet middle schools.

For further information please contact either Dr. Levin Ilya or Alexander A Faibusovich at Bucharestskay Str, 112m Fl, 28, Leningrad, USSR.

Freely Copyable Software from ICCE

First there was FrEdWriter...now there's FrEdBase, a high quality,low-cost* alternative to commercial data-base programs. You're free to make copies of FrEdBase and pass them around. Ideal for a single school, or an entire district!

ONLY $25

FrEdBase (v1.1) Database Manager can have up to 18 fields and can print out column- and label-type reports as well as bar, line, and pie chart graphs. This enhanced version includes 3 disk sides–program side, graphics side, and documentation and sample files –as well as laser-printed documentation. Packaged in a 3-ring binder, FrEdBase was written by Greg Butler and is sponsored by CUE SoftSwap and Apple Computer Australia. ICCE distributes FrEdBase by special arrangement with CUE SoftSwap.Two disks $25.

*The nominal price helps to offset development and production costs. Disks also available from CUE SoftSwap.
Copyright CUE SoftSwap.

Write or call :
ICCE, 1787 Agate St.,
Eugene, OR 97403-9905
ph. 503/686-4414

How to increase your Logo Power

Whether you're a Logo teacher, trainer, or enthusiast, you know that this powerful computer language has the potential to have a significant impact on how teachers teach and how students learn. ISTE's Special Interest Group for Logo Educators (SIGLogo) offers you a forum for the exchange of ideas, concepts, and techniques.

What is SIGLogo? SIGLogo is a professional organization that helps Logo Educators get ahead. We sponsor workshops, providing a support community for Logo-using educators. Novice or expert, you will find helpful information in each issue of our journal, *Logo Exchange*.

Satisfaction Guaranteed. Whether you teach Logo or use Logo to teach, SIGLogo and *Logo Exchange* bring you a wealth of ideas from top Logo educators throughout the world, providing you with current information on Logo research, resources, and methods. We're your personal window on professional Logo activities.

Join SIGLogo Today! As a member of SIGLogo, you will receive the *Logo Exchange* journal nine times per year. SIGLogo members are invited to participate in local, regional, and national meetings and to contribute to the flow of ideas through *Logo Exchange*. *Logo Exchange* is published monthly except for June, July, and August. SIGLogo membership is $25 for ISTE members, $30 for non-members. Add an additional $5 for non-U.S. SIGLogo membership.

The International Society for Technology in Education (ISTE) is the leading U.S. and international professional organization for computer educators. It is non-profit, supported by more than 60 organizations of computer using educators worldwide.

ISTE

ISTE/SIGLogo, University of Oregon
1787 Agate Street, Eugene, OR 97403-9905
503/686-4414