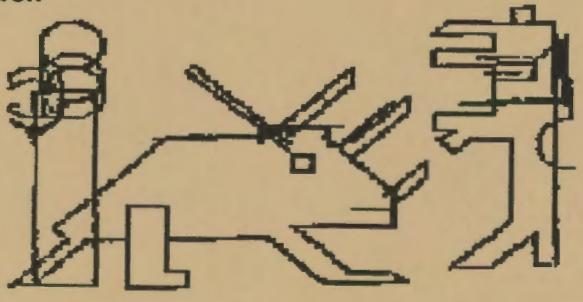
Journal of the ISTE Special Interest Group for Logo-Using Educators

LOGO EXCHANGE

May 1991 Volume 9 Number 8

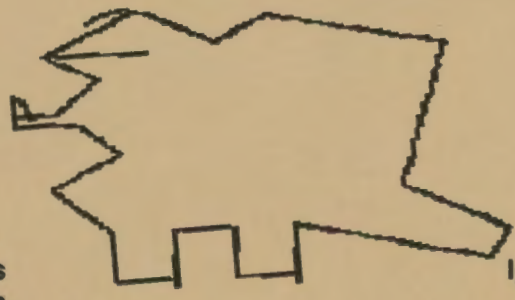
Dinosaurs, Second-Graders, and Logo

Triceratops meets Tyrannosaurus by Aaron



Corythosaurus by Marcus

Stegosaurus by Aaron



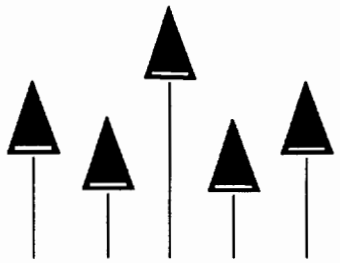
Iguanodon by Oscar

International Society for Technology in Education

ISTE Publications

LOGO EXCHANGE

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



LOGO EXCHANGE

Volume 9 Number 8 Journal of the ISTE Special Interest Group for Logo-Using Educators May 1991

Logo Exchange (ISSN 0888-6970) is published monthly September through May except for a combined issue for December/January by the International Society for Technology in Education (ISTE), 1787 Agate Street, Eugene, OR 97403-1923, USA; 503/346-4414. Subscription rates: Membership dues for SIGLogo are $16.00. $10 of this amount is for the subscription to *Logo Exchange* for one year. This publication was produced using Aldus *PageMaker*®.

POSTMASTER: Send address changes to *Logo Exchange*, ISTE, 1787 Agate St., Eugene, OR 97403-1923. Second-class postage paid at Eugene OR. USPS #000-554.

ISTE is a nonprofit organization with its main offices housed at the University of Oregon.

Contents

Individual ISTE Membership: $46.00
Dues support the development, coordination, and delivery of ISTE services, including 8 issues of the *ISTE Update* newsletter, either 8 issues of *The Computing Teacher* or 4 issues of the *Journal of Research on Computing in Education*, full voting privileges, and a 10% discount on ISTE books and courseware. Add $20 for mailing outside the USA.
Individual ISTE Members may join SIG Logo for $16.00.
Dues include a subscription to *Logo Exchange*. Add $10 for mailing outside the USA.

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

From the Editor

Endings, Changes, and a New Beginning

Once again I find myself full of mixed emotions at the end of a publication year. On the one hand, I'm relieved that I will have a break from the deadlines that always seem to loom on the horizon. On the other hand, there's a twinge of sadness. Another year has gone past in such a rush that I can't even remember what the September issue of *Logo Exchange* looked like! So much has happened both here at the University of Oregon and in the world that at times it has been hard to focus on our *LX* readers. Have we met your needs? How could we have done better? What can we do better in the future?

Next year marks the tenth year of the publication of *LX*. Tom Lough discusses *LX*'s remarkable history in his "Monthly Musings" column this month; and, as Tom indicates, it is indeed amazing that *LX* has remained a viable publication for so long. How many computer-related magazines do you read that have been published for ten years or longer? There just aren't very many. But *LX* is one of them, and we hope to keep it strong.

Changes

As we enter our tenth year of publication, there will be a number of changes we hope will help *LX* prosper for yet another ten years. First, *LX* will come to you four times a year instead of eight. (The length of your subscription will be adjusted accordingly.) This will put *LX* on the same publication schedule as all other ISTE SIGs. In addition, each issue will be somewhat longer than the monthly issues you now get.

No doubt, you are asking, "Why?" The reasons are primarily economic. I am unable to continue to produce eight issues a year. I simply do not have enough time. When we began thinking about searching for a new editor, we quickly found that asking anyone to volunteer to be an unpaid editor for eight issues of a high-quality, 36-page journal like *LX* was simply unreasonable. Then why not hire an editor? Quite frankly, the membership of SIGLogo is not high enough for ISTE to be able to afford such a financial commitment, especially for an eight-issue journal. We were simply stuck between financial realities and the amount of time that qualified potential editors would have give to *LX*. By switching to four issues per year, I can continue to serve as editor, at least for this coming year. And we also have a better chance of finding a willing editor when the time comes for a change.

Can the four-issue decision be changed? No, not for the coming publication year. The decision was made by ISTE's executive board in January, and the board doesn't meet again until NECC, in June. By then we will already be working on next year's issues of *LX*. In the longer run, if memberships in SIGLogo were to increase, then it would be possible to reconsider the decision on the number of issues.

The other significant change is that *LX* will become a refereed journal. There is a strong feeling at ISTE and among many people in the Logo community that *LX* is a high-quality journal that *should* be refereed. What does that mean to you? It means better articles. It means higher quality writing. It means articles from a wider range of authors. Judy Kull, of the University of New Hampshire, has agreed to be "applications editor" and will oversee the process of refereeing articles submitted for publication. She plans to use practicing teachers as reviewers of submitted articles. (See Judy's comments on page 4.)

For many of you, I suspect that talking about "refereeing" implies a change to a dry, academic style for the journal. That is certainly not our plan. We certainly don't intend to take the fun out of *LX*.

How Can You Contribute?

Logo Exchange is what you make it. The more you give, the better quality journal you receive. Here's a list of ways you can help.

- Contribute questions (or answers) to Frank Corley's "Questions Please" column.
- Write letters expressing your views to the editors (Judy, Tom, Sandy, Dennis, Mark, or myself) or to the columnists (Eadie, Glen, Doug, Dorothy, or Judi).
- Write an article. Articles about using Logo in the classroom should go directly to Judy Kull at the address given on page 4. Do you have an advanced Logo application? Send it to Mark. Do you have a neat math idea? Send it to Sandy. How about a report on international issues? Write to Dennis. Feeling thoughtful or philosophic? Drop a note to Tom. If you're not sure where to send it, send it to me.
- How about some art? Have your students done something unusual, exciting, or really striking with Logo? Send it to me.
- Suggest a column or a series of articles. Write to me and tell me about your ideas.

- Provide editorial help. Contact either Anita Best or me at ISTE if you are interested in helping in an editorial role.

- Help us celebrate!—see below.

Let's Celebrate a New Beginning

We plan to begin the new publication year with an extra-big, extra-special, tenth anniversary issue. We'll look back and we'll look forward. All of your favorite columnists will still be there. There will be lots of useful articles. Don't miss this special first issue of the new year!

Why don't you contribute something for this special issue? Tell us about your own Logo experiences over the years. Send them to me *immediately* so we can meet the mid-May deadline.

I know some of you will be unhappy with the changes we are making in *LX*. Change is always somewhat painful. But change can also be a new beginning. So let's all pull together to make *LX* bigger and better. Let's make it a publication that we can all be proud(er) of.

Sharon Yoder
ISTE
1787 Agate Street
Eugene, OR 97403
Phone: (503) 346-2190
CIS: 73007,1645
BITNET: YODER@OREGON

About the Cover

Judi Menken writes of her second graders:

The dinosaurs came after six months of studying every possible facet of dinosaurs except actually drawing them on the computer. Seven-year-old Aaron wondered why no one had ever tried to draw one. We'd pretty much done everything else, from graphing them with the kids' bodies to making papier-mache models. I told Aaron to feel free to try, and these drawings resulted. I put them together for the cover of our invitation to parents for our last-day class celebration.

Judi Menken
East Harlem Block School
1615 Madison Avenue
New York, NY 10029
212-860-7967

From the "Other" Editor

by Judith Kull

Tim: Soon after I started to play around, I came upon some problems that I had to solve before I could move on. That would mean that I would have to stop and look at the book or ask someone for the correct information. With this new information I would continue until a new problem arose. The circle is endless.

Corinne: After playing on the computer for several different time periods and debugging programs, I believe that I am having fun using it!

Tim and Corinne are graduate students who are studying to become classroom teachers. The Learning and Logo class in which they are enrolled has provided their first experience with Logo as well as their first experience with the educational uses of computers in general. For them, Logo has become a place for exploration and fun—a place to pose and solve problems, a place where shared ideas and new ways to look at things are welcomed, a place where new knowledge provides a springboard for moving on. Their work with Logo has prompted Tim and Corinne to think about the kind of experiences they want to provide for children in their own classrooms—experiences that promote exploration and sharing of new ideas. Another student teacher, Diana, writes:

Directions must be precise, and the turtle will not follow implicit commands. The turtle follows "do as I say, not as I intended for you to interpret what I said."

Diana is thinking about her own objectives for children in her classroom and how she will accomplish and communicate them in clear, meaningful ways.

It is in this spirit of exploration, sharing, communication, and *fun* that I invite you to contribute your ideas and experiences to *Logo Exchange*. As the new *LX* Associate Editor, and as a teacher of teachers and children, I am anxious to publish articles that give specific ideas for Logo problems/solutions. I am equally enthusiastic about publishing descriptions of teachers' and students' experiences with Logo as well as how Logo complements other educational activities. Please submit your articles or ideas for articles to me, Judy Kull, at the University of New Hampshire. We need to have *two* hard copies and would prefer the document on disk in Microsoft *Word* for the Macintosh. If that is not possible, please send us a disk containing an ASCII or text file for the Macintosh, IBM, or Apple. We have a small staff of reviewers who are looking forward to working with you. The review process will take several weeks and will draw upon the expertise of a number of "Logo people" in school and university classrooms. Reviewed articles will then be discussed with authors and revised, if necessary, prior to publication. The process ensures that we learn from each other while maintaining a high quality of publication. We hope to hear from you soon!

Judith A. Kull
Associate Editor, Logo Exchange
Department of Education
Morrill Hall
University of New Hampshire
Durham, New Hampshire 03824

Monthly Musing

Cross My Heart
by Tom Lough

Have you ever eagerly anticipated something over a long period of time? Can you remember the exquisite sweetness of the instant just before your anticipation became a reality?

I am experiencing a similar sweetness at this very moment because I have looked FD to writing this particular editorial for a long time. So excuse me if I pause just a little while to savor this moment before moving on.

It was back in the spring of 1982 when I began to dream about this moment. I was a student in the very first Logo class being taught at the University of Virginia. Professors Steve Tipps and Glen Bull were helping us learn about the wonders of this fantastic computer language. (Okay, so it was TI Logo! I still enjoyed it!)

During one particular class, I began thinking about the impact Logo was going to make on education. I realized that not every teacher would be able to take a university course to learn Logo. Moreover, not every teacher would be aware of what other teachers were doing with Logo. It was then that the idea of a Logo newsletter for teachers came to me. I told Steve and Glen about the idea and asked them if they would be interested in writing columns for the newsletter.

As the idea became more of a certainty, I had to choose a name. The most important role for the newsletter was to provide a way for teachers to exchange information with each other. The word *exchange* had a nice ring to it. So, the *National Logo Exchange* came into being.

The September, 1982, issue was a monstrous eight pages long. It was printed from dot-matrix, camera-ready copy. When the commercial printer had the big web press all ready to go, he turned to me and asked how many copies to print. Since I had a total of 32 paid subscribers by then, I calmly replied, "Please run off 1,000 copies!" The dot-matrix look dominated the entire first volume, and the newsletter soon expanded to 12 pages.

My dear wife, Posy, pitched in to help with the administrative end of things, freeing me to concentrate on editorial matters. We bought a daisy-wheel printer, bringing a new look of professionalism to Volumes 2 and 3, which had grown to 16 pages per issue. We were blessed with help from many additional contributors and friends. Finally, beginning with Volume 4, we entered the laser printer age!

Beginning with Volume 5, the *National Logo Exchange* newsletter and the (then) new *International Logo Exchange* newsletter were combined into the present *Logo Exchange*, at which time Meckler Publishing took over the operation. The magazine was transferred to its current home at ISTE in 1987 and has been published there since Volume 6. Sharon Yoder took over as editor at that time.

During this entire period, the word *exchange* seemed to dominate the name and content of the newsletter. But it was the letter X that emerged as the representation of the *exchange* idea. Long-time readers will remember the references to *NLX* and *ILX*, Robs Muir's NLXual Challenges, Griff Wigley's LXionary, and, of course, Judi Harris's incomparable Logo LinX.

And that brings me back to the present. It is time once again to celebrate the X. You see, we are at the end of Volume 9. The next issue will belong to Volume 10 (or X, if you will). My long wait is now over. What a satisfying moment it is!

For an educational computing publication to survive for nearly ten years, it must have something special going for it. In the case of *LX*, we have the incredible Logo language, several active Logo companies, and an excellent support staff. But most of all, we have you, the classroom teachers across the country and around the world. Without your continued encouragement and support, *LX* would have ceased publication years ago.

Thank you for helping yesterday's dream become a reality. *LX* has evolved and changed through the years, and you can expect more changes. But I trust that you will continue to find the same spirit of *exchange*, the X-ness, if you will, in each and every issue.

FD 10!

Tom Lough
Founding Editor
PO Box 394
Simsbury, CT 06070

Letter to the Editor

Dear Sharon:

I read with interest your article entitled "Why Logo?" in the December/January 1990-91 issue (Vol. 9, No. 4, p. 2) of *Logo Exchange*. I agree that it is the Logo philosophy that is important, not the programming language. I did "catch the dream," but BASIC was the language.

When I worked as a kindergarten teacher I felt that computers had no place in the primary classroom. Concrete experiences, I thought, were the only appropriate learning experiences for young children.

Against my better judgment I took a computer class in the summer of 1982, not because I wanted to—I'll admit I was curious—but because it fit into my schedule. The class was in BASIC.

The materials provided for the course were entitled TLC (Thinking–Learning–Creating) for Young Minds. The authors of the materials taught the class at the Museum of Science and Industry in Chicago. The text we used provided step-by-step instructions so that just by reading you could teach yourself BASIC. When questions were included in the text, direct answers were not given. The reader was guided toward discovery and exploration. The projects that were given were open-ended. Individual students could be as creative as they liked when completing the assignment.

Time was set aside for discussion so that the teachers involved could talk about incorporating the computer into the classroom situation. The course stressed open-ended teaching that cut across the curriculum. I left that two-week class feeling empowered. I felt as though I knew all there was to know about microcomputers and BASIC. I later found out how naive I was, but this initial class gave me the confidence to take other computer courses.

As an educator, it was more than the computer knowledge and BASIC that intrigued me. I was awakened to the power of these machines and excited about the potential they had for children. I am so grateful that I had the opportunity to "catch the dream." The instructors of the class were teaching the "Logo philosophy," and Logo wouldn't even work on those 16K computers.

I eventually became aware of the Logo programming language and began to understand how to use that language. My understanding grew after I went to the "Logo 86" conference at MIT. I was so affected by the Logo philosophy that I pursued other computer courses, received the State Computer Endorsement, and for the last eight years have been the building computer coordinator in an elementary school that serves Grades 1-8. All that time I've been teaching Logo and, more recently, *LogoWriter*. I am convinced that my students' use of Logo has improved their problem-solving and math skills. Because I work with students for eight years, I can chart their progress, and I believe the results have been very favorable.

As an adult learner, I know that using the computer has been a wonderful exercise for my mind. I feel totally absorbed when I work on the computer. Time seems suspended. Hours go by without my being aware of time.

In the book *Insult to Intelligence*, by Frank Smith, an adult diagnosed in childhood as having a learning disability, "credits the computer with making it easy for him to write fluently and extend his thinking about thinking and learning in general" (p. 52). I believe he was referring to word processing as expanding the mind. Perhaps the Logo philosophy could be applied to many other areas of computer use.

My students continue to amaze and inspire me. I caught the dream almost a decade ago, and I continue to dream. I hope I've transferred the dream to my students. Let's all dream on.

Carole Rivera, Computer Teacher
Nettelhorst Elementary School
3252 North Broadway
Chicago, IL 60657

Beginner's Corner

Logic Game
by Dorothy Fitch

Here is a logic game that is great for all ages to play—even young learners. It's also a fun programming project for you or your older students. As an extra bonus, the programmer gets to play too—during debugging!

How to Play the Game

The computer picks a shape, color, and size at random. For example, it might choose CIRCLE, ORANGE, and LARGE. Your job is to guess what the computer picked by asking Logo to draw various shapes that you describe. Logo then tells you how close you are to the right combination.

For example, you could ask the computer to draw a SMALL BLUE TRIANGLE or a LARGE GREEN SQUARE. Logo will tell you if your guess is COLD (no parts right), WARM (one part right), HOT (two parts right), or RIGHT! (all parts right).

Here is a sample game. (Text in bold indicates the response of the computer.)

```
SMALL  BLUE  SQUARE
WARM
MEDIUM  ORANGE  SQUARE
COLD
SMALL  GREEN  CIRCLE
WARM
LARGE  PURPLE  CIRCLE
HOT
LARGE  BLUE  CIRCLE
RIGHT!
YOU GUESSED IT IN 5 TRIES!
```

Thinking About the Program

Have you already started to think about how to write a program for this game? You need a procedure for each color word, each size word, and each shape word. You also need a way to store Logo's answers and your guesses. You need a way for Logo to pick its answers at random. Then Logo needs to compare your guess to its answers and give you a report. If you want, you can also get Logo to keep track of how many guesses it took to find the right combination. As with any Logo project, there are many ways to write this program. We'll use a way that hopefully is easy to understand.

The natural way to describe a shape for Logo to draw is by typing the adjectives first, followed by the name of the shape. We want to be able to type an English phrase, like SMALL GREEN TRIANGLE, not TRIANGLE GREEN SMALL or SMALL TRIANGLE GREEN. We also want to be able to type either SMALL GREEN TRIANGLE or GREEN SMALL TRIANGLE. It shouldn't matter whether we type the size or color first. The procedures we write must store the information so that Logo can look up how to draw the shape we describe.

Writing the Procedures

Note: The procedures below are written in *Terrapin Logo* for the Apple/Logo PLUS dialect. The only changes for other versions will be in the pen commands and IF statements. See last month's "Beginner's Corner" (Vol. 9, No. 7, pp. 11-13) for instructions on converting the procedures to another dialect.

Let's start with three sizes: SMALL, MEDIUM, and LARGE. The size will relate to the length of each side of the shape that is drawn. We can use 20 for a small shape, 50 for a medium-sized shape, and 80 for a large shape. Our size procedures can store a value for the length of the side in a variable called SIZE, like this:

```
TO SMALL
MAKE "SIZE 20
END
```

Write a similar procedure called MEDIUM that makes the size 50, and write a LARGE procedure that makes the size 80.

We also need procedures to handle the color words. (If your version of Logo does not support color, you can find an alternate, such as PenPattern.) These procedures will store a number for the color, using a variable called COLOR, like this:

```
TO WHITE
MAKE "COLOR 1
END
```

Next, write the procedures GREEN (2), PURPLE (3), ORANGE (4), and BLUE (5) in the same way. You could also use BLACK (0), although it might be confusing for younger users!

Now for the shape words. We'll write procedures to draw a SQUARE, TRIANGLE, CIRCLE, HEXAGON, and OCTAGON. They will use the variables for size and color that were set up by the procedures above. For example, here is a SQUARE procedure. (Your version of Logo might use SETPC or SETC instead of PENCOLOR.)

```
TO SQUARE
PENCOLOR :COLOR
REPEAT 4 [FORWARD :SIZE RIGHT 90]
END
```

As you write similar procedures for the other shapes, you will discover that a small octagon looks about the same size as a large square! We need to modify the SIZE variable for each shape so that they are all about the same size for each value, for example,

```
TO SQUARE
PENCOLOR :COLOR
REPEAT 4 [FORWARD :SIZE * 1.25
   RIGHT 90]
MAKE "SHAPE "SQUARE
REPORT          ; we'll get to REPORT in a minute
END

TO TRIANGLE
PENCOLOR :COLOR
REPEAT 3 [FORWARD :SIZE * 1.5
   RIGHT 120]
MAKE "SHAPE "TRIANGLE
REPORT
END

TO HEXAGON
PENCOLOR :COLOR
REPEAT 6 [FORWARD :SIZE RIGHT 60]
MAKE "SHAPE "HEXAGON
REPORT
END

TO OCTAGON
PENCOLOR :COLOR
REPEAT 8 [FORWARD :SIZE / 1.5
   RIGHT 45]
MAKE "SHAPE "OCTAGON
REPORT
END

TO CIRCLE
PENCOLOR :COLOR
REPEAT 18 [FORWARD :SIZE / 4
   RIGHT 20]
MAKE "SHAPE "CIRCLE
REPORT
END
```

The Finishing Steps

Now we need to get Logo to pick the size, shape, and color answers. We can use a PICK procedure to select them at random from the available options:

```
TO PICK :OBJECT
OUTPUT ITEM 1 + RANDOM COUNT :OBJECT
   :OBJECT
END
```

The instruction

```
PICK [WHITE GREEN PURPLE ORANGE BLUE]
```

reports one of the colors in the list after PICK. The instruction

```
PICK [SMALL MEDIUM LARGE]
```

reports one of the sizes in the list. However, when we start checking the answers, it will be simpler if we use the numbers for these colors and sizes instead of the words.

This procedure sets up the game and gets Logo to pick the winning combination:

```
TO GAME
CLEARSCREEN
MAKE "SIZE.ANSWER PICK [20 50 80]
MAKE "COLOR.ANSWER PICK [1 2 3 4 5]
MAKE "SHAPE.ANSWER PICK [SQUARE
   TRIANGLE CIRCLE HEXAGON OCTAGON]
END
```

To test this procedure, type GAME. Then type

```
PO NAMES
```

or

```
PONS
```

to see what Logo picked for the right answers. (Don't advertise this trick or game players can cheat!)

Think about what Logo needs to do to give you clues. It must set a correct guess counter to 0; check your size, color, and shape guesses against its answers; add 1 to the counter for each correct guess; and give you a clue based on how many correct guesses you made.

```
TO REPORT
MAKE "CORRECT 0
IF :SIZE = :SIZE.ANSWER MAKE "CORRECT
   :CORRECT + 1
IF :COLOR = :COLOR.ANSWER MAKE
   "CORRECT :CORRECT + 1
IF :SHAPE = :SHAPE.ANSWER MAKE
   "CORRECT :CORRECT + 1
IF :CORRECT = 0 PRINT [COLD]
IF :CORRECT = 1 PRINT [WARM]
IF :CORRECT = 2 PRINT [HOT]
IF :CORRECT = 3 PRINT [RIGHT!]
END
```

Keeping Track of Guesses

If you want Logo to keep track of the number of guesses, add this line to the end of the GAME procedure:

```
MAKE "TRIES 0
```

Add this line to the beginning of REPORT to make the number of tries one bigger each time you make a guess:

```
MAKE "TRIES :TRIES + 1
```

Then add this line to the end of REPORT:

```
IF :CORRECT = 3 (PRINT [YOU GUESSED
   IT IN] :TRIES [TRIES!])
```

Abbreviations

You can make this game even easier for younger children to play by writing procedures that let them use abbreviations:

```
TO S
SMALL
END

TO B
BLUE
END

TO TRI
TRIANGLE
END
```

Now they can draw a small blue triangle by typing:

```
S B TRI
```

Extensions

Here are some challenges for you and your students:

- Add an instruction or Help screen so your user will know how to play.
- If you use the right strategy, in how many guesses should you always be able to win? What is a good strategy?
- Add extra shapes, sizes, and colors (if your computer offers more).
- If you guess the right answer the first time, get Logo to say:

  ```
  You guessed it in 1 try!
  ```

  instead of:

  ```
  You guessed it in 1 tries!
  ```

This game was modeled after a logic game I wrote for *Kinderlogo*, a single-keystroke Logo curriculum for young learners. That package also contains the inverse game where the child selects the size, color, and shape, and the computer (through the process of elimination) finds the right combination. There's another challenge for you!

Happy Logo adventures!

Dorothy Fitch has been the Director of Product Development at Terrapin since 1987. She first became involved in educational technology in 1981, when the school where she taught music received its first computer. Since that time, 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 workshops at many local, regional, and national conferences, edited several of Terrapin's curriculum materials, and coordinated the development of Logo PLUS. She can be contacted at:

Dorothy Fitch
Terrapin Software, Inc.
400 Riverside Street
Portland, ME 04103
(207) 878-8200
CompuServe address: 71760,366

Logo Ideas

In the Spirit of Play: Playful Polygons

by Eadie Adamson

The end of a year is often a difficult time to plan your classes. Major projects are nearly complete. Time is often too short to create another significant activity. This time of year is, however, a wonderful time to encourage some playful "messing about" with Logo. This month, in that spirit of play, I offer some ideas with which students may explore and play at many levels.

For Starters

If you haven't explored growing polygons, begin with a simple procedure like this:

```
to poly :size :angle
forward :size
right :angle
poly :size :angle + 5
end
```

To use this procedure, students specify the starting size and the angle for the turn. Suggest to your students that they begin with a size of 5. Use the Stop keys to stop the action.

A Smart Stop Rule

Add a stop rule so that the turtle doesn't go beyond the edge of the screen. Find the maximum x and y coordinates of your screen by moving the turtle to the top and then to the right side of the screen. At the top type

```
show ycor
```

At the side type

```
show xcor
```

Use this information for your stop rule. It might look something like this:

```
if or xcor > 140 ycor > 85 [stop]
```

Translated into English, the above line says that "if *either* the x-coordinate is greater than 140 *or* the y-coordinate is greater than 85, stop." Now experiment with angles. What designs are pleasing? Collect a list of interesting inputs.

Play with Color, Too

What patterns emerge when color is added? Add some color:

```
setc color + 1
```

In *LogoWriter* the primitive **color** is a reporter that outputs the current color of the turtle. The line of code given above simply adds 1 to the color number each time the poly procedure is run.

You might also encourage your students to play with random color:

```
setc 1 + random 5
```

For an Apple computer, this will eliminate black. For computers with wider color ranges, simply change the number following **random** to match your highest color number.

There's More

There are many ways to experiment with this procedure. Try negative inputs. If your version of Logo has multiple turtles, see what happens when several turtles are active and you run the **poly** procedure. Make symmetrical and asymmetrical designs. Change the stop rules so that the designs created by **poly** are kept small. Write a procedure to display a series of your favorite **poly** designs: generate the first one, add a **wait** command, then clear the graphics before drawing the next polygon.

Add a dice toss to your favorite **poly** procedure—if a die comes up one way, pick up the pen; otherwise, leave it down:

```
ifelse (random 2) = 1 [pu] [pd]
```

Strange Polys and Other Ideas

If you have a copy of Abelson and diSessa's (1981) *Turtle Geometry*, you will recognize some of these ideas. If you go back to look at the first few chapters, you may find other ideas to try.

How about a variation on the simple **poly**? Rewrite **poly** so that it places dots at the vertices of the designs. A command of **forward 1** will produce a dot if your version of Logo does not contain **dot** as a primitive.

```
to dot
pd
forward 1
pu
end
```

Note that in some versions of Logo **forward 0** will produce a dot. Test your version by clearing the screen and typing

```
pd
forward 0
pu
hideturtle
```

Use **dot** in a **poly** procedure like this:

```
to poly :size :angle
pu
forward :size
right :angle
dot
poly :size + 4 :angle
end
```

Notice the interesting patterns that emerge.

Reverse!

Now try reversing colors by using **px**. Write a procedure **reverse** whose input is a list containing **poly** and its inputs. **Reverse** will reverse the the colors of the designs just drawn:

```
to reverse :design
pu
home
px
run :design
pu
home
pd
end
```

Use **reverse** by typing

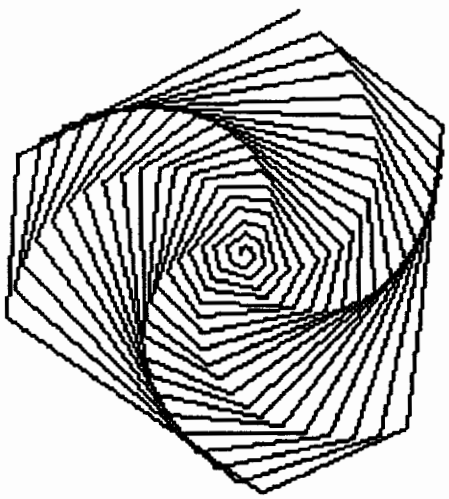
```
reverse [poly 0 90]
```

Replace the **px** command with **pe** (penerase) in the reverse procedure. Which effect do you like better?

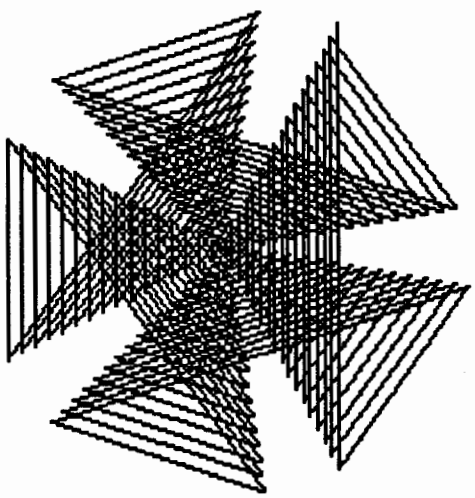
More Strange Polys

Following Abelson and diSessa's lead, I generated some more ideas for you to explore. The first procedure below requires three inputs: a number for each of two sides and an angle that will remain constant. The trick comes in the last line of the procedure. The first side increases by 1, the second side increases by 2. Surprising images result!

```
to double :side1 :side2 :angle
if or xcor > 140 ycor > 85 [stop]
forward :side1
right :angle
forward :side2
right :angle
double :side1 + 1 :side2 + 2 :angle
end
```



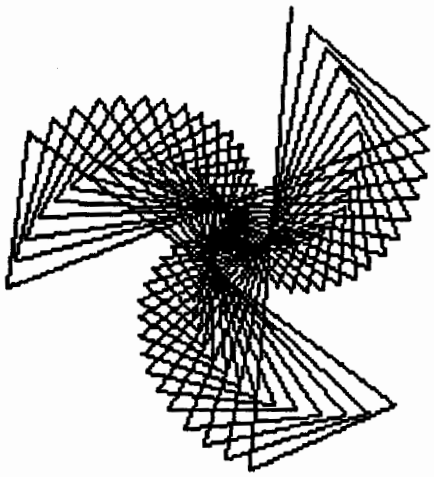
double 2 2 59



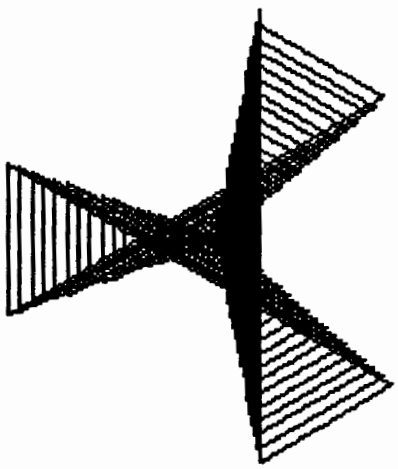
double 2 2 144

Because I didn't want to add more inputs, I wrote a **triple** procedure (it might better be called "three"), which increased the second side by 3.

```
to triple :side1 :side2 :angle
if or xcor > 140 ycor > 85 [stop]
forward :side1
right :angle
forward :side2
right :angle
triple :side1 + 1 :side2 + 3 :angle
end
```



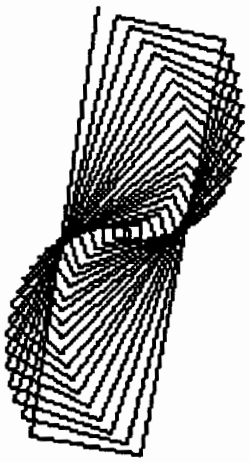
triple 2 2 119



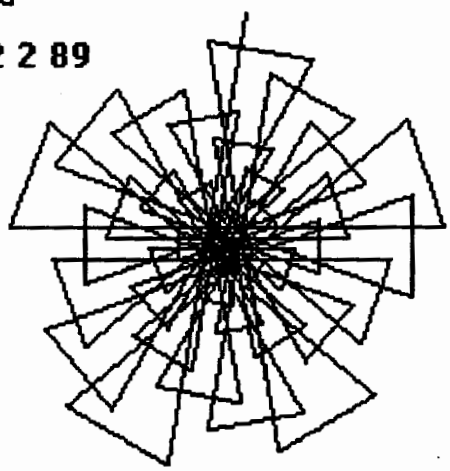
triple 2 2 120

Then I added a **quad**, which increased the second side by 4.

```
to quad :side1 :side2 :angle
if or xcor > 140 ycor > 85 [stop]
forward :side1
right :angle
forward :side2
right :angle
quad :side1 + 1 :side2 + 4 :angle
end
```



quad 2 2 89



quad 2 2 250

Strange Polys with Color Added

I wondered what would happen if I added color to these strange polys. Not surprisingly, this set me off into another series of investigations as I pondered the resulting displays (in GS *LogoWriter*) of warm and cool colors when I used an input of 90 for the angle but played with negative and positive numbers for all three inputs:

```
to double2 :side1 :side2 :angle
if or xcor > 140 ycor > 85 [stop]
forward :side1
right :angle
setc color + 1
forward :side2
right :angle
setc color + 1
double2 :side1 + 1 :side2 + 2 :angle
end

to triple2 :side1 :side2 :angle
if or xcor > 140 ycor > 85 [stop]
forward :side1
right :angle
setc color + 1
forward :side2
right :angle
setc color + 1
triple2 :side1 + 1 :side2 + 3 :angle
end

to quad2 :side1 :side2 :angle
if or xcor > 140 ycor > 85 [stop]
forward :side1
right :angle
setc color + 1
forward :side2
right :angle
setc color + 1
quad2 :side1 + 1 :side2 + 4 :angle
end
```

But There's More Yet

One of my students came up with another fascinating idea. First, we need to backtrack a bit and look at procedures that generate specific polygons. My student wrote an unusual growing-square procedure. Notice that the first input to repeat is not 4 but 6. See if you can predict what this will look like:

```
to sqs :size
if :size > 150 [stop]
repeat 6 [forward :size right 90]
setc color + 1
sqs :size + 2
end
```

Fascinated by the result of sqs, I explored the effects of similar changes in the repeat number on other polygons, until I could get each one to produce the effect of nearly concentric designs. Except for a circle, I found I could choose the same increment of 2 and get the result I wanted. I also found I needed to change the stop rule for each design in order to allow them all to grow to a similar size on the screen. What's the best number for a circle? I'll leave the experimenting to you!

And Yes, There's Still More!

Finally I put the really strange polys together into a procedure that displayed them all. Are they really polys? Why not argue that point with your students? First I wrote a tool procedure that paused at the end of each drawing before clearing the graphics:

```
to c
wait 30
cg
end
```

Then I wrote the reverse procedure, which is the same as the one above,

```
to reverse :design
pu
home
pe
run list :design 0
pu
home
pd
end
```

and put everything together (I leave the rest of the programming to you!):

```
to geos
ht
tris 0
c
sqs 0
c
pents 0
c
hexes 0
c
septs 0
c
octs 0
c
cirs 0
c
```

```
tris 0
reverse "tris
sqs 0
reverse "sqs
pents 0
reverse "pents
octs 0
reverse "octs
cirs 0
reverse "cirs
end
```

One Final Idea

So far, all the examples have drawn designs from the inside out. Here's a final challenge: add procedures that draw from the outside in. What would you need to change? How could you use this? What happens when you layer a design drawn from the inside with another design drawn from the outside? Can you figure out how to have two turtles draw, one from the outside and the other from the inside? Try one procedure with **px** and the other with **pd**, and notice the difference in the effect when two turtles cross paths.

Happy "messing about"!

If You Were Stumped

Here are some of the procedures used for gens:

```
to tris :size
if :size > 150 [stop]
repeat 5 [forward :size right 120]
setc color + 1
tris :size + 2
end

to pents :size
if :size > 100 [stop]
repeat 7 [forward :size right 72]
setc color + 1
pents :size + 2
end

to hexes :size
if :size > 90 [stop]
repeat 8 [forward :size right 60]
setc color + 1
hexes :size + 2
end
```

```
to septs :size
if :size > 80 [stop]
repeat 9 [forward :size right
   51.4286]
setc color + 1
septs :size + 2
end

to octs :size
if :size > 70 [stop]
repeat 10 [forward :size right 45]
setc color + 1
octs :size + 2
end
```

My solution for the circle:

```
to cirs :size
if :size > 15 [stop]
repeat 54 [forward :size right 10]
setc color + 1
cirs :size + .25
end
```

Reference

Abelson, H., & diSessa, A. (1981). *Turtle geometry: The computer as a medium for exploring mathematics*. Cambridge, MA: MIT Press.

Eadie Adamson is a project collaborator and consultant for the New Laboratory for Teaching and Learning at The Dalton School, where she teaches middle school computer courses.

Eadie Adamson
The Dalton School
108 East 89th Street
New York, NY 10128

Why Use Logo?

Would you like a booklet to help you explain to others why you use Logo and why they should use it too? *Why Use Logo?* is designed to help teachers, administrators, and parents understand Logo's place in education.

Why Use Logo? is strictly informational and is not directed toward any specific version of Logo. It gives a general overview of the Logo language and learning environment, emphasizing how Logo can be used as a problem-solving tool in many curriculum areas. Some of the myths it helps dispel are that Logo is just for children and that it is not a full programming language. It explains why students should not learn BASIC and supports the use of Logo to introduce other structured languages, like Pascal.

The 28-page booklet contains 12 chapters that address the following topics:.

What is Logo?
A quick lesson in Logo.
Why is Logo good for learning?
Who is using Logo and for what?
How does Logo fit into the curriculum?
What do students and teachers say about Logo?
What does Logo research tell us?
What is the role of the teacher?
How can parents get involved?
How does Logo compare with other educational software?
How does Logo compare with BASIC and Pascal?
How can I learn more about Logo?

Why Use Logo? also includes a Logo resources directory containing information about Logo books, math texts that incorporate Logo, computer education journals and magazines that feature Logo, national Logo organizations and user groups, and curriculum materials. You may request your free copy of *Why Use Logo?* from

Terrapin Software, Inc.
400 Riverside Street
Portland, ME 04103
(207) 878-8200

Logo LinX

Did Nietzsche Know Logo?
by Judi Harris

"A man has no ears for that to which experience has given him no access."

"Education has very little to do with explanation. It has to do with engagement, with falling in love with the material."

One of these statements was written in 1888. The other was recorded in 1983. The ideas that they express are remarkably similar, despite nearly a century of elapsed time between their public expression. Both were crafted by philosophers, although the tools they used were strikingly different.

One was written by Papert; the other by Nietzsche. Were it not for the difference in language, it might be difficult to identify the first statement as the older one.

"Time ripens all things"
—Cervantes

Papert is fond of saying, "What's good for thinking is good for thinking." This is true across learner cultures, among pedagogical methods, and over time. Have you experienced an eerie sense of deja vu while perusing the pages of *Mindstorms*?

1a. "Errors benefit us because they lead us to study what happened, to understand what went wrong, and, through understanding, to fix it."
—*Mindstorms*, p. 114

1b. "Mistakes are their own instructors."
—Horace

2a. "The kind of mathematics foisted on children in schools is not meaningful, fun, or even very useful."
—*Mindstorms*, p. 50

2b. "Arithmetic is where the answer is right and everything is nice and you can look out the window and see the blue sky—or the answer is wrong and you have to start all over and try again to see how it comes out this time."
—Carl Sandburg

3a. "By working with small parts...bugs can be confined and more easily trapped, figured out."
—*Mindstorms*, p. 102

3b. "Nothing is particularly hard if you divide it into small jobs."
—Henry Ford

4a. "The gulf must be bridged between the technical-scientific and humanistic cultures. And I think that the key to constructing this bridge will be learning to recast powerful ideas in computational form, ideas that are as important to the poet as to the engineer."
—*Mindstorms*, p. 183

4b. "A mathematician, like a painter or poet, is a maker of patterns."
—G.H. Hardy

5a. "The Logo teacher will answer questions, provide help if asked, and sometimes sit down next to a student and say: "Let me show you something." What is shown is not dictated by a set syllabus."
—*Mindstorms*, p. 179

5b. "[People] must be taught as if you taught them not, And things unknown proposed as things forgot."
—Alexander Pope

It seems that many of the most powerful ideas expressed by educational philosophers are written with indelible ink in the pages of time. Perhaps that is one testament to the truth of their messages.

Eloquent Reminders

During a visit to a central Virginia kindergarten, I noticed this quote, posted high on a wall, just above the students' pictograph of favorite animals:

"When you are dealing with a child, keep all your wits about you, and sit on the floor."
—Austin O'Malley

My initial reaction was one of confusion. Certainly, this master teacher realized that her students were not able to decode, much less decipher, the delightful idea posted near the ceiling! And then, the poster's position revealed the flaw in my thinking. These words were a reminder to the *adults* in this classroom culture.

Are you a "quote collector"? As you can tell from the above quotes, I am; and I'd like to share a few more of my favorites with you. They are selected for their relevance to what we generally call "Logo philosophy." But as their author listings will indicate, they originate in the common base of

human experience from which Logo itself sprang, along with many other calls for reform in education throughout history.

On Learning

"I hear and I forget.
I see and I remember.
I do and I understand."
—Chinese proverb

"Let early education be a sort of amusement; you will then be better able to find out the natural bent."
—Plato

"What is learned...depends far less on what is taught than on what one actually experiences in the place."
—Edgar Friedenberg

"You come to know a thing by being inside it."
—Edmund Carpenter

"Wonder rather than doubt is the root of knowledge."
—Abraham Heschel

"To learn is a natural pleasure, not confined to philosophers, but common to all [people]."
—Aristotle

"The creative mind plays with the objects it loves."
—Carl Jung

On Problem-Solving

"There is always more chance of hitting upon something valuable when you aren't too sure what you want to hit upon."
—Alfred North Whitehead

"The great difficulty in education is to get experience out of idea."
—George Santayana

"Failure is, in a sense, the highway to success, inasmuch as every discovery of what is false leads us to seek earnestly after what is true."
—John Keats

"The best ideas are common property."
—Seneca

"(A mistake is) evidence that somebody has tried to accomplish something."
—John Babcock

"Give me a good fruitful error any time, full of seeds, bursting with its own corrections. You can keep your sterile truth for yourself."
—Vilfredo Pareto

"Every solution of a problem is a new problem."
—Goethe

"Problems are only opportunities in work clothes."
—Henry J. Kaiser

On Education

"There is a grave defect in the school where the playground suggests *happy* and the classroom *disagreeable* thoughts."
—John Spalding

"If a man does not keep pace with his companions, perhaps it is because he hears a different drummer. Let him step to the music which he hears, however measured or far away."
—Thoreau

"Most students treat knowledge as a liquid to be swallowed rather than as a solid to be chewed, and then wonder why it provides so little nourishment."
—Sydney Harris

"What is more wonderful than the delight which the mind feels when it KNOWS? This delight is not for anything beyond the knowing, but is in the act of knowing. It is the satisfaction of a primary instinct."
—Mark Rutherford

"School is not preparation for life...school is life."
—John Dewey

"Children have to be educated, but they have also to be left to educate themselves."
—Ernest Dimnet

"An education which does not cultivate the will is an education that depraves the mind."
—Anatole France

"I have learned much from my teachers, and from my colleagues more than from my teachers, and from my students more than from all."
—Haggadah

On Teachers and Teaching

"It is the supreme art of the teacher to awaken joy in creative expression and knowledge."
—Albert Einstein

"Our chief want in life is somebody who will make us do what we can."
—Ralph Waldo Emerson

"The students are alive, and the purpose of education is to stimulate and guide their self-development. It follows as a corollary from this premise, that the teachers should also be alive with living thoughts."
—Alfred North Whitehead

"The object of teaching a child is to enable him to get along without his teacher."
—Elbert Hubbard

"To teach is to learn twice."
—Joseph Joubert

"No man can reveal to you aught that which already lies half asleep in the dawning of your knowledge."
—Kahlil Gibran

A Final Quote

Mark Twain once observed, "What a good thing Adam had—when he said a good thing he knew nobody had said it before."

Education is too often a process of reinventing wheels. Viewed only as an application of computer technology, Logo is newborn and time-bound. Viewed in historical and philosophical perspective, Logo is a natural outgrowth of progressive educational thought, set in a particular time window of technological advance. Let us recognize the foundations of many of our most powerful pedagogies, and build upon them.

Bibliography

Lincoln, W., & Suid, M. (1986). *The teacher's quotation book: Little lessons on learning*. Palo Alto, CA: Dale Seymour Publications.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York: Basic Books.

Tripp, R.T. (1970). *The international thesaurus of quotations*. New York: Harper and Row.

WGBH Transcripts (1983). *NOVA #1011: Talking turtle*. Boston: WGBH Educational Foundation.

An earlier version of this article appeared in the May, 1988, issue of *Logo Exchange*.

Judi Harris currently works as an assistant professor of educational technology at the University of Nebraska at Omaha. Her teaching, research, and service interests include Logo (of course), design of developmentally appropriate interactive hypermedia materials, computer-mediated educational telecommunications, preservice teachers as classroom ethnographers, and expressive qualities of children's computer-assisted artwork. She has also served as an elementary classroom teacher, computer specialist, mathematics specialist, and teacher telecommunications network facilitator.

Internet: JHarris@Zeus.unomaha.edu
Bitnet: JHarris@unoma1

ISTE Preconference Activities at NECC Registration Form

Name _____

Organization _____

Home Address _____

City _____

State/Country _____

Zip or Postal Code _____

Annual OA Workshop and Meeting—free to one OA Representative and $25 additional person. (Please indicate your organization above.) $ _____

Planning & Implementing International Educational Technology Projects $85 $ _____

Educational Designers' Forum $85 $ _____

The Critical Connection: Education/Business Partnerships. Public Sector $50 for 1st person, $25 additional person from the same school/agency. Private Sector $100 for 1st person; $75 additional person from the same company. ($125 and $100 after June 1, 1991.) $ _____

_____ Yes, I plan to attend the OA/ISTE Board Year-End Meeting Saturday night representing formally or informally _____.

Payment:

☐ **Payment enclosed.** (Address checks to ISTE. US funds only, please.)

☐ **Bill me.** (ISTE will add $2.50 order processing)

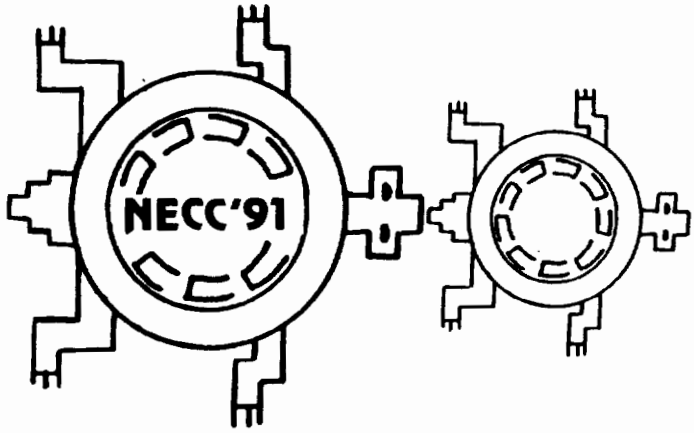
☐ VISA ☐ Mastercard ☐ Discover Card

Name on Card _____

Exp. Date _____

Mail form to:
ISTE/NECC, 1787 Agate St., Eugene, OR 97403-1923

Call if you want to register after June 1, 1991.
Phone: 503/346-4414.



ISTE Activities at NECC

The following special sessions are ISTE activities rather than NECC events and will not be listed in the NECC Advance Program.

ISTE International Committee Workshop

Planning and Implementing International Educational Technology Projects: Strategies and Experiences
Sunday, June 16, 1991 • 9:00 am - 4:00 pm

Many international educational technology projects fail for lack of attention and planning in critical areas. This workshop will gather prominent participants in functioning international projects to share experiences and describe proven successful and unsuccessful strategies. Projects under discussion will include telecommunications, collaborative research, and vocational and educational training.

Contact: Marco Murray-Lasso, Rembrandt 53, Mexico 19, D.F., 03910, MEXICO; ph. 905/563-3740; Fax: 905/761-5329; Bitnet: Murray@UNAVM1.

Educational Designers' Forum
Monday, June 17, 1991 • 9:00 am - 4:00 pm

Due to its success at NECC '89 and '90, the Designer's Forum will be a full-day ISTE preconference event in Phoenix. Designers will gather to discuss current design and educational issues. Specific topics will include networks, telecommunications, designing for multimedia, developing tools for educational reform, and others. Participants, panelists, and moderators will represent publishing houses, hardware companies, university projects, and K-12 professionals involved in product development.

Contact: Mary Cron, Forum Chair, at 213/544-4827.

The Critical Connection: Education/Business Partnerships
Monday, June 17, 1991 • 9:00 am - 4:00 pm

Education and business have a lot to gain from partnerships with each other. What are some of the ways of participating to effect change in education that don't involve large cash donations? How can links be established between education and business communities that overcome the cultural differences between the two? This symposium will:

- recognize and honor exemplary education/business partnerships that help integrate technology into schools.
- discuss types of partnerships.
- cover the how-to of goal setting and necessary steps to begin, maintain, and evaluate partnerships.

Contact: Anita Best, ISTE, 1787 Agate Street, Eugene, OR 97403; ph. 503/346-4414.

Questions Please!

by Frank Corley

I would like to begin this final column of the year with an expression of thanks to those who have helped me during this past year. Sharon Yoder, Dorothy Fitch, and Brian Harvey have answered many questions for me. Many Logo teachers, from Hawaii to Massachusetts, have fed this column invaluable queries and responses. Without contributions from these people, there would have been no column. To one and all, thank you.

During the past year, a number of people have expressed gratitude for a forum of this sort. It appears that a need has been addressed, and I hope that some questions that were raised have been satisfactorily answered. If there is sufficient interest—in the form of both questions and answers *from you*—this column will be continued next year. However, I can't do it alone. I need your input. Take a few minutes at the end of the school year and send me your questions. By then I will already be working on the *second* column for the Winter issue of *Logo Exchange*.

This column will address a number of questions from last month and will include Brian Harvey's replies to some questions asked in the March, 1991 (Vol. 9, No. 6), column.

A Correction

A question in the December/January, 1990-91 (Vol. 9, No. 4) issue concerned checking input and branching to one of two procedures depending on the response entered by the user. According to one reader, the answer previously printed in *LX* did not produce the desired result. The key line of code in *LogoWriter* is:

```
IFELSE  (EQUAL?  (FIRST  READLIST)
   "TESTWORD) [RIGHT.PROC] [WRONG.PROC]
```

I incorrectly used IF instead of IFELSE, a common *LogoWriter* error, I'm told.

Answers to Questions for April, 1991

I'm going to try to vary the versions of Logo used in the answers. There are many versions of Logo products on the market; however, the most common versions are *Terrapin Logo 3.0*, *LogoPLUS*, and *LogoWriter*, from LCSI. Some people use these trademark names as synonyms for the word Logo, which is actually the name of a computer programming language.

Logo Conferences (Question 5)

A recent issue of *Terrapin Times*, Terrapin's free newsletter, calls for announcements of such conferences from its readers. Since conferences are generally not dialect-specific, users should see that publication for announcements of conferences that interest them. Readers should send announcements of conferences directly to Terrapin. Of course, you should also see *LX* and *The Computing Teacher* for conference information.

Testing and Assessment (Question 6)

Sharon Yoder says that the International Society for Technology in Education (ISTE) has a book on Logo assessment and will soon publish another one, authored by Daniel Lynn Watt and Molly Lynn Watt. Sharon says she hasn't "tested" Logo in years, but she wrote an article on the subject in the "Logo Center" column of the August/September 1989 (Vol. 17, No. 1) issue of *The Computing Teacher*. You can write or call ISTE to request a catalog of their publications, as well as to order back issues of *The Computing Teacher*.

Brian Harvey Responds

Here are Brian Harvey's answers to several specific questions asked in the March, 1991 (Vol. 9, No. 6), "Questions Please!" column.

There are local variables in some versions of Logo. Generally they are declared with the command LOCAL, as in:

```
TO  PROC
LOCAL  "VAR
. . .
```

Brian explains that you can define a procedure within a procedure by using the command DEFINE, which creates locally defined procedures. The companion commands TEXT and DEFINE, which do not exist in all versions of Logo, allow you to define a procedure from a list of lists or to convert a procedure into a list of lists. DEFINE is a good way of dealing with copious subprocedures that have no use outside of their defining procedure. I must admit that I didn't know that these commands existed.

Brian also defined a CASE statement for us, and scolded me for not writing one of my own. A CASE statement procedure might look like this:

continued on page 35

MathWorlds

edited by
A. J. (Sandy) Dawson

Many years ago—too long ago to really think about—I took a non-Euclidean geometry course as part of my Masters program at Washington University in St. Louis. What was remarkable about this course was that I failed it! Yes, I really did. I failed the course. I simply could not construct any mental images that would allow me to comprehend these rather *weird* worlds, or so I thought at the time. Oh, how I wish that Logo had been invented then, and that someone like Gary Martin was around to write a non-Euclidean Logo microworld to help me learn. I am sure experience with his R-Logo and L-Logo microworlds would have enabled me to gain an intuitive understanding, and subsequently some mental images, of how lines behave in these worlds. However, that insight had to wait many years until late one night in a pub in Kingston, Ontario, when Dick Tahta started drawing diagrams with a flow pen on the surface of a bottle.

In his article below, Gary Martin notes the potential importance of microworlds toward the implementation of the *Standards* document. I think, indeed, that Gary's microworlds would provide an incredibly rich experience for secondary school students, giving them direct experience with alternate axiom systems. I hope Gary's description will be sufficient to whet your appetite to explore these microworlds further, and that you write to Gary to obtain the code for them. His ideas certainly helped me enrich my personal experience with non-Euclidean geometries!

Non-Euclidean Logo Microworlds

by W. Gary Martin

The *Curriculum and Evaluation Standards for School Mathematics* of the National Council of Teachers of Mathematics (1989) suggests a fundamental restructuring of what geometry is studied and how it is studied in Grades 9-12 (see especially Standards 7 and 8). One aspect of these proposals that has received very little attention is the final section of Standard 7, which states that college-bound students should "develop an understanding of an axiomatic system through investigating and comparing various geometries" (p. 157). To really understand what an axiomatic system is, the argument goes, one must examine more than one example of an axiomatic system. Moreover, since Euclidean geometry appears to students as being quite self-evident, it is hard to convince them of the need to explicate basic assumptions, such as the Parallel Postulate.

While attention to alternative geometries may seem quite clear and reasonable, several problems exist with including them in a curriculum. A fundamental problem lies in dealing with a geometry that contradicts both the students' intuitions and their previous schooling. Many theorems in non-Euclidean geometry appear to be, in the words of Gauss, "paradoxical and, to the uninitiated, absurd" (quoted in Wolfe [1945, p. 47]). The inherent difficulty of developing understanding at this level is confirmed by the Van Hiele model of geometric thinking, the highest level of which is *rigor*, in which students understand the nature of a geometry as a geometric system, including the existence of multiple geometries. Testing of students who had completed a high school geometry course found that less than 30% had attained either of the highest two levels (Usiskin, 1982).

A second difficulty is the small number of materials that deal with non-Euclidean geometries at a level appropriate for high school students. For example, among 10 geometry textbooks reviewed, only one (Jacobs, 1987) includes more than passing reference to alternatives to Euclidean geometry. In particular, it is difficult to locate easily accessible models for the alternatives to Euclidean geometry. Krause (1975) suggests that to be instructive, a model should have the following three qualities:

1. It should be similar to Euclidean geometry in its axiomatic structure.
2. It should have meaningful applications.
3. It should be understandable to a person who has studied Euclidean geometry.

Logo microworlds provide a very accessible means for students to enter the world of non-Euclidean geometry. This article briefly describes two such microworlds that can run with minor modifications in any dialect of Logo. R-Logo and L-Logo model Riemannian and Lobachevskian geometries, respectively. In each of these microworlds, the students use the usual commands (such as FORWARD and RIGHT) to direct the turtle around the screen. The microworlds meet Krause's criteria described above:

1. Both Riemannian geometry and Lobachevskian geometry are very similar to Euclidean geometry; they differ only in varying the Parallel Postulate. In terms of Logo, they use the same commands as the standard Logo.
2. Many meaningful applications of the models can be found. The usual tasks of drawing regular polygons and other geometric figures can be undertaken.
3. The models are readily understandable to anyone who has played around with standard Logo. Both R-Logo

and L-Logo have been used extensively by the author with preservice and inservice teachers and with high school geometry students.

The Models

A brief discussion of the two microworlds and examples of how they can be used follow. When using either model, students should first have an opportunity to explore the environment, to understand how things work. Students usually first experience a sense of bewilderment that "things don't work right," followed by some attempts to make sense of that experience.

In R-Logo, which models Riemannian geometry, students intuit fairly quickly that they are moving the turtle around on a sphere. As the turtle approaches the edges of the screen, it appears to move less far than when it is in the center of the screen. When the turtle advances far enough, it appears to begin returning to the center of the screen, although its path is now gray, suggesting that the turtle is now "under" the screen. But the full impact of this strange new Logo does not become apparent until students attempt to use their usual Logo commands to draw figures.

For example, consider this task: "Draw an equilateral triangle where each side is 150 steps long." Students who have had experience with Logo will confidently type something like

```
REPEAT 3 [FORWARD 150 RIGHT 120]
```

Amazingly, this command does not work; the turtle does not return to its starting position, and the figure is not closed. At this point, many students will want to throw up their hands in despair; it is the teacher's job to encourage them to debug their efforts. Most students realize that the only viable possibility is to change the amount of the turn, and that the amount of turn needs to be decreased, since the turtle crossed over its own path. Trial-and-error can be utilized to determine an approximate value for the turn, (in the scaling used in this model, about 111 degrees).

A further perturbation of their sense of propriety occurs with the natural follow-up task: "Draw an equilateral triangle where each side is 200 steps long." Students make the obvious leap—111 degrees is the amount of turn needed to draw an equilateral triangle in this strange Logo, so let's type

```
REPEAT 3 [FORWARD 200 RIGHT 111]
```

Again, this fails; the turn is too large. Trial-and-error can again be used to find a turn that works.

Drawing several additional equilateral triangles may lead to generalizations such as these:

1. The sum of the turns must be less than 360 degrees.
2. The sum of the turns is not fixed, but changes according to the size of the triangle.
3. The larger the triangle, the smaller the sum of the turns.

As has been noted by several authors (e.g., Kynigos, 1989), Logo experiences do not necessarily translate to (or from) explicit geometric knowledge. The teacher needs to encourage students to explore what this implies in terms of geometry. In the case of an exploration of equilateral triangles, the statements about sums of turns can be related to statements about sums of exterior angles of triangles. These, in turn, can be related to statements such as these about the interior angles of triangles on the sphere:

1. The sum of the interior angles of a triangle must be more than 180 degrees.
2. The sum of the interior angles is not fixed, but depends on the size of the triangle.
3. The larger the triangle, the larger the sum of its interior angles. This generally surprises students, but, as one student philosophized, "Things are different on a sphere."

One additional exploration (out of many possible ones) that may serve to highlight the differentness of Riemannian geometry from normal geometry involves drawing two perpendiculars to the same segment, as in

```
RIGHT 90
FORWARD 200
BACK 200
LEFT 90
FORWARD 50
RIGHT 90
FORWARD 200
```

Students quickly realize that if the two perpendiculars were to be extended they would intersect; indeed, they can have the turtle draw sufficiently long segments to observe the point of intersection. The teacher can develop these observations into a discussion about the nonexistence of parallel lines on the sphere, its relation to the Parallel Postulate, and so forth.

The development of L-Logo follows a path similar to that described above for R-Logo. Beginning with exploration of the environment, students will realize that, as in R-Logo, dis-

tances contract as the turtle approaches the edge of the screen. Unlike R-Logo, the turtle does not "wrap around" but continues to move smaller and smaller distances as it approaches the edge. Actually, only a disk in the center of the screen can be accessed by the turtle. This is the Poincare model for the Lobachevskian plane (cf. Smart, 1988). While students find this quite puzzling, they eventually develop enough intuition about how the turtle will behave so that they can effectively deal with the environment; one student characterized it as "sort of an inside-out sphere."

Explorations with equilateral triangles can again be carried out, this time noting that the amount of turn is greater than in the usual Logo, rather than less than the usual as was the case in R-Logo. Statements parallel to those about R-Logo can then be developed for L-Logo. Again, the teacher must give attention to translating Logo experiences into inferences about geometry. Explorations with lines prove interesting as well; not only will two lines perpendicular to the same line be parallel, but a line at an 85 degrees angle may also be parallel to a perpendicular line. This may be expanded to discussions of how corresponding angles might work in the Lobachevskian plane—if two lines are parallel, can we be sure that their corresponding angles are congruent? An interesting exploration is to find the range of values that an angle formed by one line with a transversal may have in comparison to the measure of its corresponding angle on a second line so that the two lines are parallel. (A tidbit for the eager reader—it depends on the distance between the two lines.)

An effective summary task for students after using either (or both) of the microworlds follows: "Summarize the differences we have found in Riemannian (and/or Lobachevskian) geometry, compared to Euclidean geometry. What else do you think will be different?" This frequently generates a lot of discussion and can suggest additional explorations within the microworlds.

While the account presented in this column has admittedly been sketchy, it is my hope that you have captured a bit of a taste of how Logo microworlds might be useful in promoting the necessity of explicating the axiom system in which one is working, as suggested in the *Standards*, and in gaining experiences with alternative, non-Euclidean geometries within an environment in which students can develop and explore ideas for themselves. Moreover, it is my experience that these microworlds need not be limited to the best and brightest "college-intending students," as suggested by the *Standards*; but that these microworlds can make non-Euclidean geometry can come alive for a full range of students.

References

Jacobs, H. R. (1987). *Geometry* (2nd ed.). New York: W. H. Freeman and Company.

Krause, E. F. (1975). *Taxicab geometry*. Menlo Park, CA: Addison-Wesley Publishing Company.

Kynigos, C. (1989). Intrinsic versus Euclidean geometry: Is the distinction important to children learning with the turtle? In G. Vergnaud, J. Rogalski, & M. Artique (Eds.), *Proceedings of the Thirteenth International Conference for the Psychology of Mathematics Education* (pp. 194-201). Paris: City University.

National Council of Teachers of Mathematics. (1989). *Curriculum and evaluation standards for school mathematics*. Reston, VA: National Council of Teachers of Mathematics.

Smart, J. R. (1988). *Modern geometries* (3rd ed.). Pacific Grove, CA: Brooks/Cole Publishing Company.

Usiskin, Z. (1982). *Van Hiele levels and achievement in secondary school geometry*. Chicago: University of Chicago.

Wolfe, H. E. (1945). *Introduction to non-Euclidean geometry*. New York: Holt, Rinehart and Winston.

Dr. W. Gary Martin is the director of the Geometry Learning Project within the Curriculum Research and Development Group (CRDG) at the University of Hawaii-Manoa, Honolulu. His address is 1776 University Avenue, Honolulu, HI 96822. He can be reached through Bitnet at
gmartin@unCCUX.BITNET.

A. J. (Sandy) Dawson is on leave as director of the Professional Development Program, Faculty of Education, Simon Fraser University, Vancouver, BC, Canada V5A 1S6. He was recently a guest of the University of Hawaii's Curriculum Research and Development Group (CRDG). Dr. Sid Rachlin is the chair of the Hawaii Algebra Learning Project within the CRDG, and Sandy wishes to thank Sid and his colleagues for their hospitality during his stay.

Sandy Dawson
Faculty of Education
Simon Fraser University,
Vancouver, BC, Canada V5A 1S6
Bitnet: UserDaws@SFU.BITNET

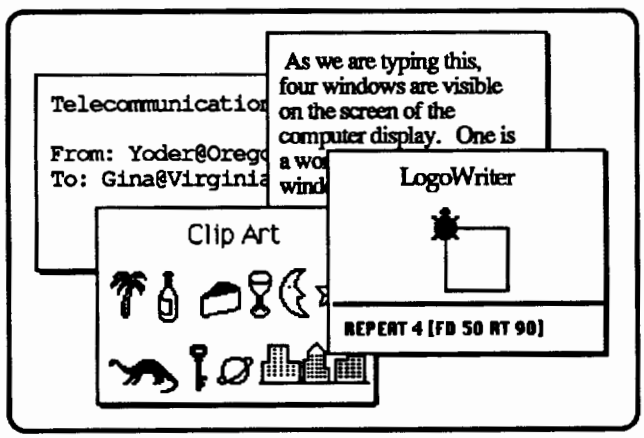
Logo Connections

A Turtle in Your Window
by Glen L. Bull and Gina L. Bull

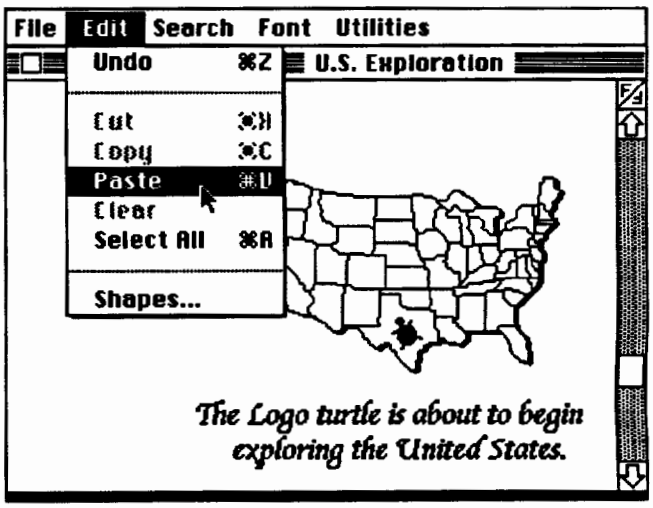
This year we have been discussing multimedia suggestions for Logo. Multimedia applications involve the combined use of audio, video, computer graphics, and other media. The first versions of Logo were developed with Teletype-like terminals that had no graphics capability and did not include the familiar Logo turtle. Later versions of Logo were created for the Texas Instruments 99/4 microcomputer, the Apple II, the Atari, the Commodore 64, the Acorn, and many other microcomputers. The capabilities of each computer influenced the characteristics of Logo. For example, the Apple II+ could display only six colors, while the original IBM PC could display only four colors at a time.

The next major advance in microcomputers will inevitably affect what we have come to think of as "Logo." Computers with *windows* affect the way computing takes place. As we are typing this, four windows are visible on the screen of the computer display we are using. One is a word-processing window. A second is a telecommunications window that allows us to send or receive electronic messages from around the world. A third is a graphics window with a display of clip art. *And the fourth window contains LogoWriter*. With the click of a button we can move graphics or text from any window to any other window. As you may have guessed, we are using *LogoWriter* for the Macintosh.

The illustration below was taken from a *LogoWriter* window on the Macintosh. To move the *LogoWriter* screen into the word processor, we simply selected the graphic, copied it, moved to the word-processing window, and pasted the graphic into the word-processed document you are now reading. With windows, you can create materials for a class in *LogoWriter*, and then paste examples from *LogoWriter* screens into your word processor for lesson plans or handouts that you can distribute to your students. In the example below, the *LogoWriter* window contains a map of the United States, with a Logo turtle, perched on Texas, about to begin a cross-country trip.



This *LogoWriter* page was created by selecting and copying a U.S. map that appeared in the clip art window. The map was then pasted into the *LogoWriter* window.



Of course, *Logo Exchange* readers may recall from our November, 1990, (Vol. 9, No. 3, pp. 20-23) column, "Imported Graphics" that it is also possible to import a graphic into Logo from another program on the Apple II computer. There are two important differences in this instance.

- The first difference is that on the Macintosh it was not necessary to exit *LogoWriter* to go to the graphics program. Both *LogoWriter* and the graphics program

can remain in separate windows on the screen at the same time.

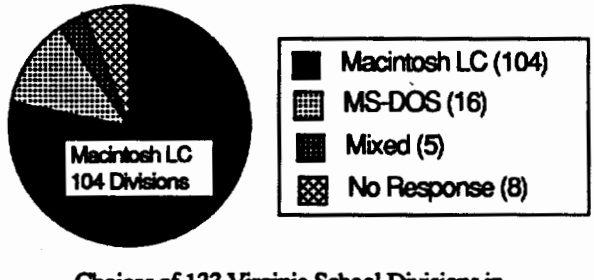
- The second important difference is that any properly written program on the Macintosh can exchange graphics with any other program via the Clipboard. The Clipboard allows programs on the Macintosh to freely exchange graphics without the necessity for the cumbersome conversions that are sometimes necessary on the Apple II. (An explanation of some graphics interchange formats on the Apple II can be found in the article "Secular Conversions," by Judi Harris, in the September, 1988 *LX*.)

These sorts of differences will have a profound effect on the way that we think about computers and use them in our daily lives.

A Window in Your Future

Until recently there have been relatively few Macintoshes in the public schools, largely because Macintoshes have been too expensive for schools and were incompatible with existing Apple II software. This spring, Apple addressed these issues by offering schools the Mac Classic for less than $1,000 and by introducing the Macintosh LC, which can run Apple II programs with the addition of an Apple II option card. It appears that these adjustments will result in the introduction of significant numbers of Macintoshes in schools.

In Virginia, the Governor's Educational Technology Initiative is designed to ensure that Virginia teachers have access to appropriate technologies for their classrooms. In the current round of the initiative, school systems were offered a choice of either Macintosh LC computers or MS-DOS systems. According to the response to a survey conducted by the Department of Education through Superintendent's Memorandum 137, approximately 80% of the school systems participating in this initiative chose the Macintosh LC.



Choices of 133 Virginia School Divisions in Response to Superintendent's Memo No 137.

In the future, windows capabilities can also be expected for IBM and IBM-compatible computers as well. The expected successors to the MS-DOS operating system for those machines, Microsoft Windows and OS/2, are both based on the model of the windowing concept popularized on the Macintosh. The Microsoft Windows operating system is already in widespread use in many universities and businesses, although it requires hardware beyond the economic reach of most public schools. However, when these computers become inexpensive enough for public education, it seems likely that versions of *LogoWriter* would be developed for these systems as well. (It is already possible to run *LogoWriter* for the IBM and most other MS-DOS programs under Microsoft Windows 3.0, but a special version of these programs is required to take advantage of the features of the windowed operating system.)

Thus, in a sense, the first version of *LogoWriter* for an operating system with windows is like a time machine that allows us to see what future versions of Logo may offer. For this reason, the characteristics of this program are of particular interest.

The Evolution of Logo and Windows

LogoWriter is not the first version of Logo for the Macintosh. From where we sit, a copy of *Microsoft Logo* for the Macintosh is visible on a shelf across the room. This version of Logo was developed by Logo Computer Systems, Inc. (LCSI) and distributed by Microsoft soon after the introduction of the Macintosh. Since there were very few Macintoshes in the schools, sales did not justify its continued support, and the product was discontinued. Coral later developed *Object Logo* for the Macintosh, a powerful version of Logo that utilized an object-oriented approach. However, two years ago Apple purchased the company to obtain another language that Coral developed, *Object LISP*, and *Object Logo* was taken off the market at that time. Recently, another firm, Paradigm Software, has obtained rights to *Object Logo* and reintroduced it to the market. We have received a copy and hope to review it in a future column.

Terrapin also developed a version of Logo for the Macintosh. *Terrapin Logo for the Macintosh* is not currently compatible with MultiFinder. Therefore it is not possible to use multiple windows containing other programs in concert with *Terrapin Logo on the Macintosh*. However, this spring *Terrapin Logo for the Macintosh* is scheduled for an upgrade that will make it compatible with MultiFinder. A review of *Terrapin Logo for the Macintosh* may be found in the August/September, 1990, (Vol. 18, No. 1, pp. 28-31) issue of *The Computing Teacher*.

First Impressions

Revision of a program like *LogoWriter* for another computer system poses some interesting problems. On one hand it is desirable to maintain compatibility with previous versions of the program on other computers. A user who has previous experience with *LogoWriter* on an Apple or IBM computer should be able to use *LogoWriter* on the Macintosh without undue difficulty. On the other hand, the program should take advantage of the capabilities of the newer, more powerful computer system. This design puzzle is complicated by the fact that developers may have more programming expertise and experience with the previous computer systems with which they are familiar.

LogoWriter for the Macintosh appears to do a number of things right. As we have mentioned, it is possible to import and export graphics from other programs. Although all successful Macintosh programs have this characteristic, this capability cannot be taken for granted in programs new to the Macintosh. This opens a universe of clip art, digitized images, and graphics to the Logo user. *LogoWriter* also operates properly under MultiFinder, and does not generate any error messages or problems even when multiple windows with other programs are opened. This enables the user to take full advantage of the power of the Macintosh with *LogoWriter*.

Interactive Videodiscs

LogoWriter for the Macintosh has documented commands to send and receive characters through the serial port. This makes it possible to control external devices attached to the Macintosh. The first thing we did was to try to use *LogoWriter* for the Macintosh with a videodisc player. This two-line test program to control the videodisc player worked the first time:

```
TO PLAYVIDEO
SEND "PL
SEND CHAR 13
END
```

Readers of the Logo Connections column, "Logo and Videodiscs," in the March, 1991, issue of *LX* (Vol. 9, No. 6, pp.11-15) know that it is also possible to control a videodisc player using *LogoWriter* on an Apple II.

However, with *LogoWriter* for the Macintosh the serial commands necessary to do this are, for the first time, officially sanctioned and documented in the manual. Since access to the serial port is essential for control of a variety of peripheral devices, this is a significant advance. In one of our future projects, we intend to try using *LogoWriter* to explore use of a home control system that turns on lights, monitors visitors, controls motors, and so forth. This should be fertile ground for a variety of robotics projects.

Although it is easy enough to write short procedures such as the one above to control videodisc players with *LogoWriter*, this can sometimes present a barrier to a first-time user. We would recommend inclusion of *LogoWriter* procedures to control popular videodisc players such as the Pioneer 2200 and 4200 and the Sony 1200, with appropriate documentation, in the *LogoWriter* package. Relatively little expense would be involved, and it would make it clear to every *LogoWriter* user (even those who do not read *Logo Exchange*) that *LogoWriter* can easily be used to control videodisc players and other multimedia systems.

Talking Turtles

Several of the recently introduced Macintoshes, including the Macintosh LC, have built-in sound recording capabilities. *LogoWriter* for the Macintosh takes advantage of these capabilities and includes commands to record and play back digitized speech and sound. Imagine the turtle counting turtle steps "One ... two ... three ... four" as it moves forward. This capability will be very welcome in elementary classrooms with younger children. Other applications programs have used the sound-recording capability to meet a variety of instructional goals, from storage of bird calls to playback of diagnostic engine sounds in a vocational education class.

The sound digitizer is actually a 20 kilohertz analog-to-digital converter. In theory, not only a microphone but other devices, such as temperature sensors, light detectors, motion sensors, and so forth, could also be connected to this input. This would open the door for exploration of a variety of microcomputer-based laboratory (MBL) applications in science education.

A Timer

LogoWriter for the Macintosh features a working clock and timer. The CLOCK command tells the current date and time, while the TIMER command reports the duration (to the nearest 1/60 second) since the timer was last reset. Analog and digital clocks are popular Logo projects; now it will be possible for them to tell the actual time. Another project might consist of a *LogoWriter* page with a calendar in which the turtle moves to the correct date when the page is opened.

We have observed that a timer is often one of the most popular features in programs such as the Broderbund *Science Toolkit*. Students use it for studying reaction time and for many other projects. However, without rudimentary pro-

gramming capability, the range of potential projects is limited. With *LogoWriter*, for example, it might be possible to use the timer and a light sensor to time the downhill speed of wooden race cars in a pine-box derby. (If you are using *LogoWriter* on the IBM PC and are interested in the date-and-time programs, see the article "Date, Time, Palette, and Diskspace: Four New Primitives for *LogoWriter* on the IBM/PC," by Charles Crume, in the February, 1991, issue of *Logo Exchange* [Vol. 9, No. 5, pp. 28-29].) On the Apple II, timing loops can also be used to create a timer that can be calibrated as a project for students.

Shapes and Colors

If you have a color Macintosh, such as the Macintosh LC with a color monitor, 256 colors are available on *LogoWriter* for the Macintosh. A variety of projects, from design of stained glass windows to development of color-coded diagrams, are afforded by this capability. (*LogoWriter* for the IBM also offers 256 colors on an IBM PS/2 computer with an MCGA monitor.)

The number of user-defined turtle shapes available is increased from 30 to 90 in *LogoWriter* for the Macintosh. More importantly, it is possible to have a different set of shapes on every *LogoWriter* page. This overcomes a significant limitation in prior versions of *LogoWriter* that could only be overcome through cumbersome work-arounds involving the creation of multiple ProDOS subdirectories.

LogoWriter for the Macintosh also allows the user to track the position of the mouse cursor as well as the turtle position. One of the first programs we created was a short procedure that allowed a whirring helicopter to be guided across the screen with the mouse. We are sure that other projects involving the mouse will suggest themselves.

Limitations

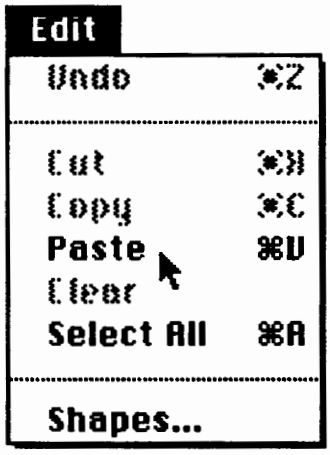
As you can tell, we are very enthusiastic about *LogoWriter* for the Macintosh. It is a significant breakthrough. As might be expected with the first release of a program for a new machine, there are a few limitations. Keep in mind that as we write this, we have only had access to *LogoWriter* for a week. It is possible that in some cases we simply may have not discovered the correct command or procedure yet.

Fonts

One of the most striking aspects of the Macintosh is access to a variety of fonts that appear on the screen exactly as they appear in print. Fonts can be accessed within *LogoWriter*, just as they can be within any other Macintosh program. However, in the present version only one font appears to be available on each *LogoWriter* page.

Graphics Interface

Macintosh users have become accustomed to being able to copy or paste text or graphics from any application to any other application. *LogoWriter* for the Macintosh allows either text or graphics from another application to be *pasted* into a *LogoWriter* page. However, the reverse is not true. Copy and Cut on the Edit menu are disabled for graphics and only function with text.



This deficiency can be circumvented by using the SAVEPIC command to save a picture on a *LogoWriter* page to disk. The graphic saved in this file can then be imported into another program. The limitation of this method is that it leads to a proliferation of temporary graphics files on disk, and cannot be used to select a specific portion of a *LogoWriter* image.

Most Macintosh users will expect Copy to function with both text and graphics. This could be achieved in a future version of the program through provision of a selection tool consisting of a dashed rectangle that can be used to indicate the portion of the graphic that is to be copied.

Macintosh Resources

Most Macintosh programs have a *resource fork* and a *data fork*. Fonts, icons, cursor shapes, sounds, and other resources are stored in the resource fork. The *LogoWriter* manual indicates that the new *LogoWriter* primitive PLAY can be used to play a sound resource stored within a *LogoWriter* page. There are literally thousands of public domain sound resources available to Macintosh users, so we were eager to play some of them from within *LogoWriter*.

However, the Macintosh *Resource Editor* did not display a resource fork for the *LogoWriter* pages we created, suggest-

ing that sound resources may be stored within *LogoWriter* in a nonstandard way. The PLAY command can be used to play global sounds stored in the system folder. However, in our experience, allowing users in a public laboratory to modify the system folder on a Macintosh is an invitation to disaster. In addition, use of this method means that the program will promptly quit working properly when it is shared with another user on a different computer, since the sounds stored in one system folder will not necessarily be the same on other machines.

Extending the *LogoWriter* metaphor to include not only pages but books could provide one way of addressing this issue in a future revision. A *LogoWriter* book could consist of a number of *LogoWriter* pages that share common resources, including sound resources. This would provide a means of allowing all the pages in a given *LogoWriter* book to access the same resources, and at the same time allow the application (and all of its associated resources) to be shared with other users.

Many of the teachers with whom we work have long wanted a way of arranging *LogoWriter* pages in a specific order on the table of contents, and possibly the concept of the *LogoWriter* book might include this capability as well. This would open the way for primitives such as NEXTPAGE or PREVPAGE, which would take the user to the next page or the previous page in the book.

LogoWriter for the Macintosh is a major advance; it is far more powerful than any other version of *LogoWriter* available thus far. Those who have used *LogoWriter* on Apple, IBM, or Commodore computers will have no difficulty using this version of *LogoWriter* and will find it even easier to use in many respects.

A Decade of *Logo Exchange*

Next year will mark our tenth year of columns for *Logo Exchange*. Although we have not been keeping an exact count, in the coming year we will be approaching our 100th column. When we first began using Logo, we stored our programs on audiocassette tapes on the TI 99/4 microcomputer. The changes that have taken place since then are remarkable, with the advent of hard disks, CD ROMs, videodiscs, memory measured in megabytes rather than in kilobytes, and many other advances.

The decade to come will be even more remarkable. In the future, all computers will have an environment with windows and a graphical user interface. Multimedia capabilities for intermixing sound, video, and computer graphics on a single monitor will be built into many educational computers. *LogoWriter* for the Macintosh provides a hint of these possibilities. Over the summer we will be exploring some of the capabilities we have described above. In the coming school year we hope to offer a new series of columns titled "Windows on Logo." If you are developing or exploring an application involving Logo and windows, we would like to hear about it. Have a good summer!

Glen and Gina Bull
University of Virginia
405 Emmet Street
Charlottesville, VA 22903

E-Mail	Internet	BITNET
Glen:	GBull@Virginia.edu	GBull@Virginia.bitnet
Gina:	Gina@Virginia.edu	Gina@Virginia.bitnet

Extra for Experts

edited by Mark Horney

Object Logo 2.51: A Brief Review

by Der-Thanq Victor Chen

Have you ever complained about the slowness of Logo? Did you ever want to compile your Logo program? Did you ever feel frustrated when chasing a bug in your Logo program? Did you ever want to access the Macintosh toolbox through Logo? *Object Logo* allows you to solve all the above problems, offering a powerful Logo environment for you to explore.

Object Logo is designed for the Apple Macintosh computer with at least one megabyte of memory. It has expanded Logo's "ceiling" to a higher and wider dimension in two ways. The first is that more handy tools and primitives have been added to the "traditional" Logo structure. The second is the creation of an object-oriented programming environment. With these two major improvements, *Object Logo* puts more direct control of the Macintosh in the hands of the user.

The newest version, *Object Logo* 2.51, was released in February, 1991, by Paradigm Software. It fixes a few bugs that were reported in Version 2.5 and adds support of 24-bit color and hierarchical menus.

This review examines the features of *Object Logo* that are not part of most of today's versions of Logo. In addition, it looks briefly at the object-oriented features of this new version.

The Package

The *Object Logo* package includes:

- two double-sided (800K) disks containing *Object Logo* 2.51 along with a number of other useful files,
- the *Object Logo* reference manual,
- a registration card,
- a license agreement, and
- a list of available reference books.

The price of *Object Logo* ranges from $149 for single users to $819.50 for lab packs. You may not make unlimited copies of *Object Logo*. If you buy the *Object Logo* 20-pack license, you may run it on up to 20 Macintosh computers in a single lab setting. You are, of course, allowed to make one copy solely for archival purposes.

The Manual

Generally speaking, the manual is well written. However, as the manual itself says, "This is a reference manual, not a primer." It assumes that the reader has experience with Logo. Most of the examples in the manual are quite understandable. Without further instruction an experienced Logo user should be able to easily utilize the new features.

However, an experienced Logo user is not necessarily an experienced object-oriented programmer. The chapters about object-oriented programming are more difficult for average readers because they lack explanations of the overall theme of object-oriented programming. The average Logo user would have to learn concepts such as property, class, and inheritance from sources other than this manual.

Enhanced Features in *Object Logo*

It is now possible to build stand-alone applications with *Object Logo*. These "double-clickable" applications can be launched from the Finder just like any other application.

Object Logo offers four groups of debugging primitives, which allow the user to debug in different ways. *Tracing* primitives provide an overview of a procedure, *stepping* primitives provide an opportunity to pursue details, *watching* primitives display information about variables, and *pausing and continuing* primitives provide opportunities to examine bugs and correct errors.

The ability to insert comments into *Object Logo* is now available. Comments are an important way of making a program more understandable. In *Object Logo*, comments can be inserted into any position in a program. Also, *Object Logo* provides a continuation symbol that allows the user to break long command lines into several lines. These tools make already easy-to-read code even easier to understand.

Object Logo solves a problem inherent in a number of current versions of Logo. In these versions of Logo, most arithmetic operators are available in infix form only, while the other primitives are prefix form. For example, the syntax of

```
and  (3 > 9)  (5 = 3)
```

is often confusing. Most students ask why they can't write the previous "and" statement as

```
(3 > 9)  and  (5 = 3)
```

or, to be consistent,

```
and  (> 3 9)  (= 5 3)
```

The **and** is a prefix reporter, while the > and = are infix reporters. The ability to use > and = in an infix form benefits beginners—especially younger children—because these arithmetic operators are exactly like those taught in math classes. However, this mixture begins to cause confusion when students start writing more advanced programs. The syntax rules do not seem to be consistent.

To avoid this problem, *Object Logo* provides another set of equivalent prefix arithmetic reporters. For example,

```
Sum 5 3
```

is equivalent to

```
5 + 3
```

and

```
Equalp :X 5
```

is equivalent to

```
:X = 5
```

These reporters allow the programmer to maintain a consistent syntax and thus alleviate confusion when learning formal syntactic rules.

Object Logo adds two new arithmetic data types: ratio and complex number. Examples of these are:

```
print 3 + 2/3 (note that there are no spaces around the / sign)
11/3

print complex 3 5
3+5i
```

These two new types will make the creation of Logo microworlds for use in math classes even easier.

There is also special group of data types, called MacType, which allow you to easily manipulate complex data types. For example,

```
Make "MyPict Snap 30 50
```

will store a bitmap (a MacType) of width 30 and height 50 to the variable **MyPict**. This bitmap represents the portion of the graphic on the current screen that has the turtle at the upper left corner and is 30 pixels wide and 50 pixels long. Instead of storing the whole picture, you can now simply store the complex bitmap in one variable. MacType also includes such types as cursor, region, and polygons.

Object Logo offers several new control structures: **Forever, DoUntil, DoWhile, Until,** and **While**. These control primitives are similar to the **Repeat until** and **While do** structures in Pascal. Most versions of Logo have forced programmers to use tail recursion to write iterative programs. It is now possible to write a simple loop without using recursion.

Traditional Logo:

```
To MoveNoMore50
If YCor < 50 [Forward 1 MoveNoMore50]
End
```

Object Logo:

```
To MoveNoMore50
DoUntil [Forward 1] [YCor > 50]
End
```

The Object Environment

The object environment is the most powerful and probably the most difficult component of *Object Logo*. An object is a collection of procedures and variables. You may ask an object to do things based upon the procedures defined in that object.

An object can be defined from an existing object by using the primitive **OneOf**. The newly defined object will inherit all the properties from its parent object:

```
Make "GeometryTurtle OneOf Turtle
```

You now have a **GeometryTurtle** that "knows" all the procedures that a regular turtle, **Turtle**, knows. You may now "ask" this object to do things it knows, for example,

```
Ask :GeometryTurtle [Forward 50]
```

This new turtle will then move forward 50 turtle steps. Note that the primitive **Ask** is used whenever you want to define something in an object or to ask an object to do something.

You may now define a procedure **square** for **GeometryTurtle** to carry around:

```
Ask :GeometryTurtle [to square :size]
repeat 4 [forward :size right 90]
end
```

You will see a message that says **square** is defined in **GeometryTurtle**. When you type

```
Ask :GeometryTurtle [Square 20],
```

the **GeometryTurtle** will then draw a square of size 20.

You may also override (redefine) an old procedure in an object:

```
ask :GeometryTurtle [to back :size]
forward :size
end
```

Remember the **GeometryTurtle** was made using the turtle object. It knows how to back. But now when you type

```
Ask :GeometryTurtle [back 30]
```

instead of going back 30 steps, the **GeometryTurtle** will go forward for 30 steps.

You can also have a variable carried by an object for later use:

```
Ask :GeometryTurtle [make "scale 10]
                run some other commands here
Ask :GeometryTurtle [forward 5 *
   :scale]
```

After the first command line, the **GeometryTurtle** will "remember" the variable **scale** with value 10. In the third line, the **GeometryTurtle** will recall the value of **scale** (10) and then go forward for 50 (5*10 = 50) turtle steps.

Object Logo offers a wide range of predefined objects so that you don't have to create some common objects that you might want to use. Some of these predefined objects are **Speaker, Window, Driver, MenuItem, InputStream, Menu, OutStream, Turtle, Conductor, MIDICoder,** and **MIDIScore**. Even Logo itself can be treated as one of the objects.

Because it is object-oriented, *Object Logo* makes a natural connection to the potential of the Macintosh. It treats the Macintosh as an object-oriented environment. This makes it easy to access the low-level routines in Macintosh. For example, you can type

```
PaintOval [50 50 100 100]
```

to access the Macintosh tool box routine to draw an oval.

Summary

Object Logo's strengths include the following:

- The syntactic rules are more consistent than most current versions.
- The ratio and complex number data types are useful for math learners.
- The MacType data type allows users to manipulate complex data type easily.
- More choices of control structures are provided.
- Multiline commands and comments are useful in increasing the readability of a program.
- The improved debugging tools are powerful.
- The object-oriented features shorten the distance between the user and the low-level routines in Macintosh.
- It is now possible to build stand-alone applications.

Here are some of *Logo Object*'s weaknesses:

- More object concepts should be introduced in the manual. Studying the manual alone doesn't help the user understand the concept of object-oriented programming.
- The manual should include the proper application of object-oriented features. For example, when is it appropriate to create a new object and when is it appropriate to simply write a procedure? In what kind of situation should the user use a procedure instead of an object? What conventions should be followed?
- Although *Object Logo* expands Logo's "ceiling" to a higher level, its "threshold" doesn't seem to improve a lot. Compared with *LogoWriter*, the user interface of *Object Logo* is less intuitive. For example, if the user closes the Listener window by accident, it is not intuitive that the user should

press a key in order to bring up that window again. Most likely, the user will be forced to use the mouse and "thumb" through the menus to find out how to open the Listener.

- This version runs only on the Macintosh series of computers. A version of *Object Logo* that runs on other powerful computers would be welcome.

In short, the most recent version of *Object Logo* is an exciting one. It adds more useful tools to the traditional Logo structure. It puts more of the power of the Macintosh under the user's control. It provides a powerful object-oriented programming environment. It makes sophisticated Logo programming easier than ever before.

Der-Thanq Victor Chen is a doctoral student specializing in computers in education at the University of Oregon. He has been a graduate fellow at the UO since Fall, 1990, assisting your *LX* editor in teaching her classes (including Logo, of course!) He can be reached at:

CATE
University of Oregon
1787 Agate Street
Eugene, OR 97403

BitNet: Derthanq@Oregon

EUROLOGO 91

Parma, Italy
27-30 August 1991

Final Call for Papers
Third European Logo Conference 1991

LIPI
LABORATORIO INTERPROVINCIALE
DI INFORMATICA

Universita' di Parma
Regione Emilia-Romagna
Provincia di Parma

THEMES
The EUROLOGO 91 themes reflect a wide variety of topics:

1. Classroom experiences with Logo from primary to higher education including special education and professional schools.
2. Logo prospects and research. Logo and other languages. Logo and Artificial Intelligence.
3. Programming environments. Microworlds. Logo and subjects in school curriculum.
4. Teacher training in relation to Logo and Logo use.

OFFICIAL LANGUAGE
The official language of EUROLOGO 91 is English. However, the organization committee is pursuing the possibility of having simultaneous English-Italian translations.

DATES TO NOTE

Deadline for registration	31 May 1991
Deadline for submitting papers	31 March 1991
Deadline for discount registration	31 March 1991

EUROLOGO 91
LIPI
B.go G. Tommasini, 8
I-43100 Parma (Italy)

Special Interest Group Meetings at NECC '91

Phoenix, Arizona June 18-20, 1991

At this year's NECC conference, ISTE's Special Interest Groups will meet during these times:

12:30 - 2:00 pm, Tuesday, June 18

SIGTC **SIG/Tel**

12:30 - 2:00 pm, Wednesday, June 19

SIGCS **SIGTE**

6:45 - 8:45 pm, Wednesday, June 19

HyperSIG **SIGLogo**

Everyone is welcome to attend these meetings. Check for room numbers at registration or at the ISTE booth.

Logo: Search and Research

Geometric Construction Programs: What Have We Learned?

by Douglas H. Clements

For the past couple of months we've been discussing Logo-based geometric construction programs for middle school and high school students. This month we'll summarize research on other geometric construction programs. This research has implications for work with Logo-based geometric construction programs and "regular" Logo.

The Point

Construction programs such as the *Geometric Supposer* series help students make and test conjectures. The *Supposer* programs allow students to choose a primitive figure, such as a triangle or quadrilateral (depending on the specific program), and perform measurement operations and geometric constructions on it. The programs record the sequence of constructions and can automatically apply it to other figures. In this way, students test the generality of the consequences of constructions.

The (Bottom) Line

Research confirms that students can use the *Supposer* effectively. In one study, *Supposer* students performed as well as or better than control students on geometry exams (Yerushalmy, Chazan, & Gordon, 1987). In addition, students' learning went beyond standard geometry content. For example, they could reinvent definitions, make conjectures, pose and solve significant problems, and devise original proofs. Making conjectures did not come easily to students—there was much frustration at the beginning of the year. By the end, however, nearly all students were successfully making conjectures and felt the need to *justify* their generalizations. On specially designed tests, *Supposer* students produced the same or higher level generalizations than the comparison group.

Students often have difficulty understanding proof. For instance, they often do not distinguish between two sources of knowledge about geometrical statements: measurement evidence and deductive proofs. They believe that measuring examples "proves" a statement true for all members of an infinite set and that deductive proof pertains to one example only (Chazan, 1989). *Supposer* activities designed to change these beliefs move students away from considering measurement evidence as proof (Chazan, 1989; Wiske & Houde, 1988). Some students, however, still think that there might be counterexamples to deductively proven results. Students believe that, unlike textbook theorems, *Supposer*-generated theorems need to be proved before they can be accepted as true (Lampert, 1988a).

There is, however, some conflicting evidence. For example, Bobango (1988) found that instruction using the *Supposer* significantly raised students' van Hiele level of thought. The only substantial increase, however, was from Level 1 to Level 2. Instruction did not raise achievement in standard content or proof writing.

Overall, it appears that students using the *Supposer* can come to understand the importance of formal proof as a way of establishing mathematical truth. Students focus more on the *illumination* or *explanation* function of proof than on the *verification* function (Yerushalmy, 1986).

Constructing a Path to Learning: Effective Teaching Strategies

Implementing the *Supposer*'s guided inquiry approach requires effective teaching strategies. Successful teachers connect students' inquiry with the curriculum. They encourage inquiry as a way to learn what needs to be known.

Teachers use a variety of strategies to accomplish these goals. For example, they label students' activities with words such as "know," "explain," "problem," and "answer." By basing some lessons on students' inquiries, they legitimate that approach to knowledge acquisition. They build formal language in phases, from firmly situated "pointing" at the computer screen to natural language descriptions of patterns of data to mathematical terms to symbolic expressions (Lampert, 1988b).

In one study, teachers first using the *Supposer* often promoted a rigid sequence of data collection and analysis, conjecturing, and proof. This led to rote data gathering and obscured the difference between representations of specific instances during data collection phases and more general representations during conjecture and proof phases. Students did not appreciate the different levels of generality these phases represented. Successful teachers start *Supposer* activities as investigations of a geometric relationship or concept (Yerushalmy et al., 1987).

Finally, good teachers pose good problems. Research suggests several effective strategies (Yerushalmy, Chazan, & Gordon, 1988). For example, in writing materials, teachers should:

- state the goal of the problem clearly,
- provide explicit process instructions focusing on the students' role as inquirers, and
- use diagrams to exemplify written construction instructions, once students understand diagrams as models.

Teachers should also avoid small-scale problems with charts that direct students to make specific measurements. Students find this uninteresting and have difficulty seeing patterns. Instead, expand the problem. For example, ask students to add auxiliary lines necessary to write a proof. For students who do not have the necessary background for deductive arguments, start with construction problems and other problems with specific solutions (and avoid diagrams at this early stage). Finally, teachers should model inquiry strategies.

Teachers in regular classrooms face multiple difficulties using the *Supposer*. These problems include:

- deciding how to proceed, from subjugating students' inquiries to the traditional curricular sequence to building new sequences (not in "logical" order) from students' work (this demands considerably greater command of the subject matter, especially to maintain connections between students' and teachers' goals);
- freeing students' from concerns about the correctness of their conjectures;
- dealing with the consequences of the innovation in terms of time, assessment, and authority; and
- facing dilemmas, such as viewing knowledge as content versus process and learning as social versus individual (Lampert, 1988b; Wiske & Houde, 1988).

Struggles such as these, however, seem to have the potential to change teachers' classroom practices and their beliefs about knowing and learning geometry. With experience, teachers get better at resolving problems, though difficulties do not disappear. Use of the *Supposer* demands hard work from, and causes some frustration in, both teachers and students, but benefits are evident (Lampert, 1988a; Wiske & Houde, 1988; Yerushalmy et al., 1987).

Parallel Lines of Inquiry

Several findings are intriguingly consistent across studies using different computer environments. First, researchers and teachers consistently report that in such contexts students cannot "hide" what they do not understand. That is, difficulties and misconceptions that traditional approaches mask emerge and must be dealt with. This leads to some frustration on the part of both teachers and students, but it also leads to greater development of mathematics abilities (Clements & Battista, 1989; Schofield & Verban, 1988; Yerushalmy et al., 1987).

Second, at least at the high school level, students can become confused regarding the purpose of different components of a course. A single location for computer work, discussion, and lecture may alleviate this confusion. A monitor or projector for group discussions is essential. Problems of arranging and managing hardware and software must not be underestimated.

Third, evaluation of learning in such environments must be reconsidered. Traditional approaches did not assess the full spectrum of what students learned. In some cases, these approaches make little sense (e.g., when students worked on self-selected inquiries).

Fourth, two students working cooperatively at a computer seems ideal. One of the strengths of these computer environments is the spontaneous generation of cooperative learning and teaching (Clements & Battista, 1989; Lampert, 1988a; Yerushalmy et al., 1987).

Shaping Ideas about Geometry

Certain computer environments allow the manipulation of screen objects in ways that assist students in viewing them as representatives of a class of geometric objects. This develops students' ability to reflect on the properties of the class of objects and to think in a more general and abstract manner (Clements & Battista, in press). Thoughtful sequences of computer activities and teacher guidance are critical.

Inquiry environments such as the *Supposer* and Logo-based environments have the potential to serve as catalysts in promoting change. Most important, teachers and students can change their ideas about what it means to learn and understand geometry. Fundamental changes demand considerable effort on the part of teachers. They also call for extensive support from teacher educators, peers, and ultimately, the greater school system and culture.

References

Bobango, J. C. (1988). Van Hiele levels of geometric thought and student achievement in standard content and proof writing: The effect of phase-based instruction. *Dissertation Abstracts International, 48*, 2566A. (University Microfilms No. DA8727983)

Chazan, D. (1989). Instructional implications of a research project on students' understandings of the differences between empirical verification and mathematical proof. In D. Hergert (Ed.), *Proceedings of the First International Conference on the History and Philosophy of Science in Science Teaching* (pp. 52-60). Tallahassee, FL: Florida State University Science Education and Philosophy Department.

Clements, D. H., & Battista, M. T. (1989). Learning of geometric concepts in a Logo environment. *Journal for Research in Mathematics Education, 20*, 450-467.

Clements, D. H., & Battista, M. T. (in press). Geometry and spatial reasoning. In D. A. Grouws (Ed.), *Handbook of research on mathematics teaching*. Reston, VA: National Council of Teachers of Mathematics/Macmillan.

Lampert, M. (1988a). *Teachers' thinking about students' thinking about geometry: The effects of new teaching tools. Technical Report*. Cambridge, MA: Educational Technology Center, Harvard Graduate School of Education.

Lampert, M. (1988b). *Teaching that connects students' inquiry with curricular agendas in schools. Technical Report*. Cambridge, MA: Educational Technology Center, Harvard Graduate School of Education.

Schofield, J. W., & Verban, D. (1988). Computer usage in the teaching of mathematics: Issues that need answers. In D. A. Grouws, T. J. Cooney, & D. Jones (Eds.), *Perspectives on research on effective mathematics teaching: Vol. 1* (pp. 169-193). Hillsdale, NJ: Lawrence Erlbaum.

Wiske, M. S., & Houde, R. (1988). *From recitation to construction: Teachers change with new technologies. Technical Report*. Cambridge, MA: Educational Technology Center, Harvard Graduate School of Education.

Yerushalmy, M. (1986). Induction and generalization: An experiment in teaching and learning high school geometry. *Dissertation Abstracts International, 47*, 1641A. (University Microfilms No. DA8616766)

Yerushalmy, M., Chazan, D., & Gordon, M. (1987). *Guided inquiry and technology: A year long study of children and teachers using the Geometric Supposer: ETC Final Report*. Newton, MA: Educational Development Center.

Yerushalmy, M., Chazan, D., & Gordon, M. (1988). *Posing problems: One aspect of bringing inquiry into classroom*. Cambridge, MA: Educational Technology Center, Harvard Graduate School of Education.

Douglas H. Clements is an associate professor at the State University of New York at Buffalo, Department of Learning and Instruction. He has studied the use of Logo environments in developing children's creative, mathematics, metacognitive, problem-solving, and social abilities. Through a National Science Foundation grant, he has codeveloped a K-6 elementary geometry curriculum, *Logo Geometry*, published by Silver Burdett, and Ginn. His most recent book, *Computers in Elementary Mathematics Education*, emphasizing Logo, was published by Prentice-Hall in 1989.

Douglas H. Clements
State University of New York at Buffalo
Department of Learning and Instruction
593 Baldy Hall
Buffalo, NY 14260
CIS: 76136,2027 BITNET: INSDHC@UBVMS

Questions, Please!

continued from page 20

```
TO CASE :CHOICE
IF :CHOICE = first.one [list of actions STOP]
IF :CHOICE = second.one [list of actions STOP]
IF :CHOICE = third.one [list of actions STOP]
:

END
```

I'm still unclear about the use for such structures. I suppose I should try to learn more about these ideas. We all really appreciate Brian's comments here at Priory. They certainly provoked a lot of conversation. Thank you, Brian.

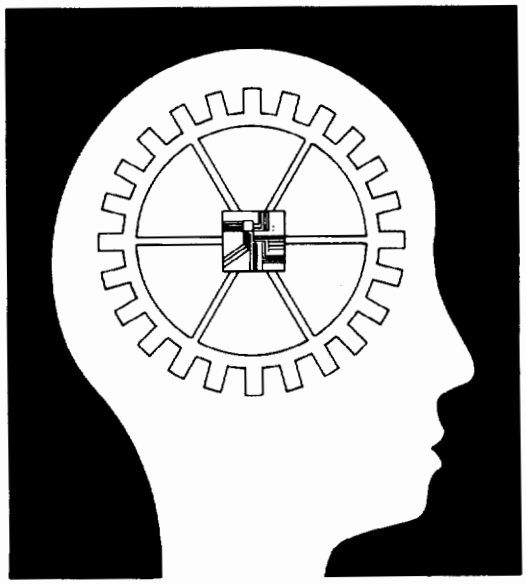
Farewell, and Please Write

So long for the year! I have really enjoyed working on this column. It has been my first venture into writing. I probably won't make it to NECC this year, so come up with some questions for me at the SIGLogo meeting. Have a good, inquisitive summer.

Frank J. Corley
St. Louis Priory School
500 South Mason Road
St. Louis MO 63141

Fantastic Journey Through Minds and Machines

by Michael Muir



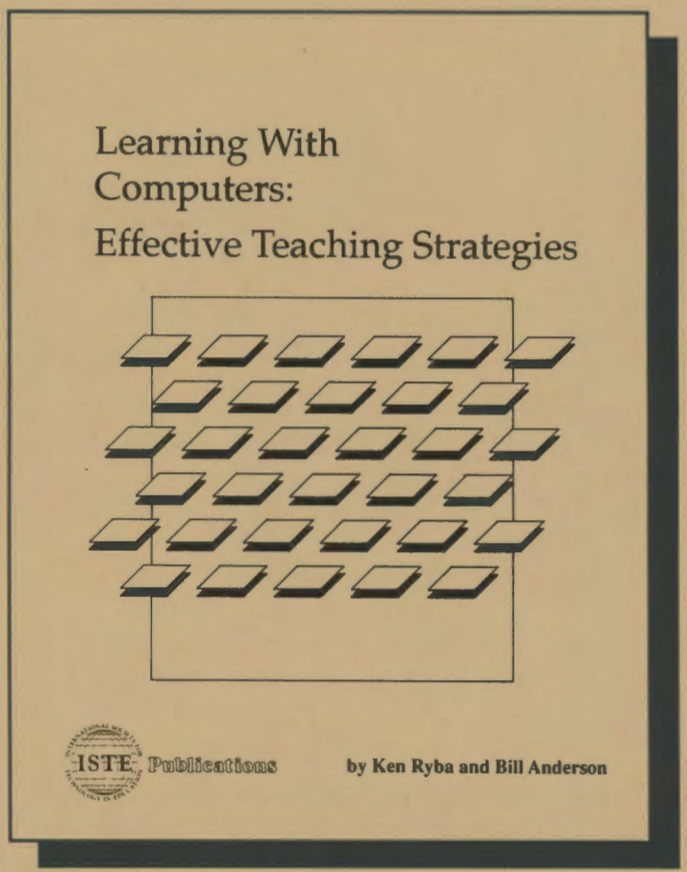
Artificial Intelligence (AI) calls up visions of supercomputers and gigabytes of computer code. Michael Muir gives AI a human dimension and a practical application in the secondary classroom.

Even noncomputing readers will enjoy his entertaining chapter introductions on the important history and people of this technical and intellectual frontier. Muir explains the essential challenges of creating robots, games, natural language, machine intelligence, and expert systems.

For students and teachers of computer science, each chapter also includes a complete AI program in Logo (both *LogoWriter* and *Logo Plus* versions). The programs range in sophistication from a simple maze to an inference engine with natural language interface. The code is documented line by line, so that even novice programmers can follow the logic and development. Suggestions for program extensions offer expert-level challenges.

Fantastic Journey Through Minds and Machines is $13.95 plus $3.25 shipping.

To order contact: ISTE, 1787 Agate St., Eugene, OR 97403-1923; ph. 503/346-4414



Learning With Computers: Effective Teaching Strategies is filled with suggestions to help students become independent thinkers and productive members of learning teams. Excellent for university classes and as a resource for K-12 teachers. Drawing on cognitive theory and research, Ken Ryba and Bill Anderson outline specific methods that teachers can use with a variety of computer applications (Logo, word processing, adventure games, databases, spreadsheets, telecommunications) to enhance social and cognitive development. Includes strategies for cooperative learning.

Learning With Computers Effective Teaching Strategies is $20.00 plus $4.50 shipping.

To order contact:

ISTE, 1787 Agate St., Eugene, OR 97403-1923; ph. 503/346-4414

Logo Exchange
ISTE
1787 Agate Street
Eugene, OR 97403-1923