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

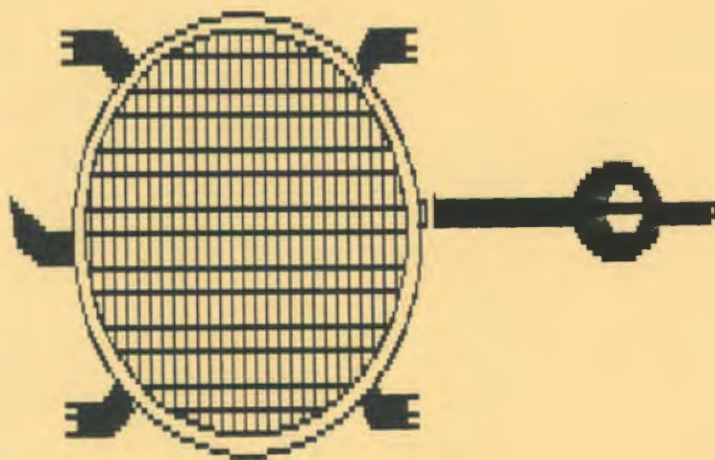


LOGO EXCHANGE

April 1991

Volume 9 Number 7

Carved turtle design on Senufo wooden door
Ivory Coast, Mali, Upper Volta
By Frances Elliot
New York City



Zamble mask
Guro, Ivory Coast
By Benita McDougal
New York City

International Society for Technology in Education



Publications



LOGO EXCHANGE

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

**ISTE BOARD OF DIRECTORS
1990-91**

Executive Committee

Gary Bitter, President

Arizona State University

Bonnie Marks, President-Elect

Alameda County Office of Education (CA)

Colin Harvey, Secretary/Treasurer

Educational Solutions, Inc. (Mississauga, ON)

Dennis Bybee

*Department of Defense Dependent Schools
(Washington, DC)*

Sally Sloan

Winona State University (MN)

Directors

Roy Bhagaloo

Muscogee County Schools (GA)

Cyndy Everest-Bouch

Christa McAuliffe Educator (HI)

Susan Friel

University of North Carolina

Jenelle Leonard

*Computer Literacy Training Laboratory
(Washington, DC)*

Marco Murray Lasso

*Sociedad Mexicana de Computacion
en la Educacion*

Alan November

Educational Technology Consultant (IL)

Paul O'Driscoll

Oregon Total Information Systems

Barry Pitsch

Heartland Area Education Agency (IA)

David Walker

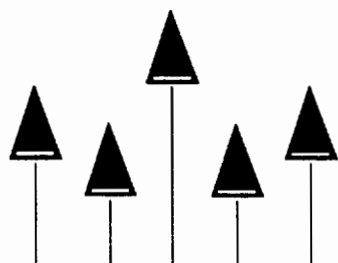
Apple Computer Europe (France)

Executive Director

David Moursund

Associate Executive Director

C. Dianne Martin



LOGO EXCHANGE

Volume 9 Number 7

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

April 1991

Founding Editor

Tom Lough

Editor-In-Chief

Sharon Yoder

International Editor

Dennis Harper

Contributing Editors

Eadie Adamson

Gina Bull

Glen Bull

Frank Corley

Doug Clements

Sandy Dawson

Dorothy Fitch

Judi Harris

Mark Horney

SIGLogo Board of Directors

Gary Stager, President

Lora Friedman, Vice-President

Bev and Lee Cunningham,

Secretary/Treasurer

Publisher

International Society for

Technology in Education

Dave Moursund, Executive Officer

Anita Best, Managing Editor

Talbot Bielefeldt, Associate Editor

Ron Renschler, Assistant Editor

Mark Horney, SIG Coordinator

Lynda Ferguson, Advertising Coordinator

Ian Byington, Production

Advertising space in *Logo Exchange* is limited. Please contact the Advertising Coordinator for space availability and details.

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.

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.

Contents

From the Editor—It Can't Happen to Me Sharon Yoder	2
Monthly Musing—Replacing the B-Word Tom Lough	3
Logo Ideas—Counting to a Million Revisited Eadie Adamson	4
SHUFFLE: A Logo Primitive to Rearrange the Items in a List Charles Crume	8
Beginner's Corner—Logo-to-Logo Exchange Dorothy Fitch	11
Logo LinX—The Handwriting on the Screen Judi Harris	14
Questions Please! Frank Corley	17
MathWorlds—Filling in the Gaps with Logo Software Ihor Charischak Sandy Dawson, Editor	19
Logo Connections—Logo and Telecommunications Glen L. Bull and Gina L. Bull	24
Extra for Experts—Using Assembly Language in <i>Terrapin Logo</i> for the Apple II Susan Wells Rollinson Mark Horney, Editor	27
Logo: Search and Research—GeoTools: Logo-Based Geometric Construction Douglas Clements	33

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

© All papers and programs are copyrighted by ISTE unless otherwise specified. Permission for republication of programs or papers must first be gained from ISTE c/o Talbot Bielefeldt.

Opinions expressed in this publication are those of the authors and do not necessarily reflect or represent the official policy of ISTE.

From the Editor

It Can't Happen to Me

I love teaching. I have always loved teaching. It never mattered whether I was teaching sewing or swimming, math or computing, I have just always loved teaching. I have been proud to be a teacher, whether in an informal setting, in the public schools, or at the university level. I have worked hard over the years to be a better teacher, to help raise the standards in my profession, to meet the needs of my students. But during the last few weeks, I have been shocked and embarrassed by what is happening to education here in Oregon and elsewhere.

Perhaps you read the fine print in your newspapers in early February (as I write this). Because of the passage of a property tax limitation measure in Oregon, drastic cuts are being made in all kinds of services, including education. This year higher education was most dramatically affected. If there are no changes in the tax structure, then next year the cuts may affect the public schools and community colleges. "Oh," you say, "another budget cut. We've *all* been through those." No, I fear this is different. The University of Oregon cut its *entire* division of teacher education, including undergraduate, graduate, and all teacher certification programs. Oregon State University removed its College of Education entirely. Other smaller institutions took drastic cuts as well. No longer will the two largest institutions in the state of Oregon train teachers. The cuts here were unprecedented, and seem, as of this writing, to be quite final.

As you might guess, our education faculty and students are in shock. As the word spreads throughout the telecommunications networks, I receive messages of total disbelief from colleagues all around the country. "How," they ask, "can this be happening?" Many here in Oregon are asking the same question. We simply can't believe it's true. At the same time, many of my colleagues report serious (albeit not as disastrous) cuts in their educational funding. Many institutions in this country seem to be in financial trouble, and education seems to be taking many of the most severe cuts.

The tremendous impact of insufficient funding on education reminds me of a little poster that has hung near my desk for years. It says:

it will be a
great day
when
our schools
get all the
money
they need

and the air force
has to hold
a bake sale
to buy a
bomber

If this seems too "anti-military" to you, just substitute any well-funded organization you want for "air force" and grieve with me that public education has to fight so hard in so many places to have the funds to produce quality education.

And what does all of this have to do with Logo? A lot, I think. We in the Logo community are deeply committed to improving education. We believe in the philosophy behind Logo. We believe in educational change and educational reform. Can such change happen without funding? Most likely not. Think how long it takes to train teachers to use Logo well. Think of the inservice costs involved. When budgets are cut, there is less money for teacher training, innovative programs, and educational reform. Without a financial commitment to education, it is unlikely that Papert's dream can be realized in today's schools.

It seems to me that we all need to lobby in our own ways for the cause of education beyond our own classrooms and schools. I know I am certainly getting a lesson in politics as I watch the proceedings here. The state gave the University of Oregon three weeks to decide which part of itself to excise. There was no time for careful reflection. There was no time to consult with individual students and faculty, no time to examine particular courses or programs. Once the wheels were in motion, there seemed to be nothing that could stop them. Will any of our programs survive? Will any of the faculty have teaching positions here in a year or so? Will current students have a chance to finish their degrees? We just don't know right now. And it's not clear when we will know. Clearly, we should have been thinking about how to prevent those wheels from ever rolling in the first place. We here all seemed to think "it can't happen to us" but this time it did happen to us. Let's hope that the next time the budget cuts don't hurt you or your courses or your program.

Sharon Yoder
ISTE
1787 Agate Street
Eugene, OR 97403
Ph: 503-346-2190
BITNET: YODER@OREGON

Monthly Musing

Replacing the B-Word by Tom Lough

The following is a true story. At a certain educational computing conference not too long ago, I heard a presenter say, "...and so I wrote a few basic Logo procedures and..." Having been an active participant in "Logo vs. BASIC" computer language discussions for the past several years, I was struck by the presenter's statement. I could not shake off the mildly unsettled feeling brought on by the phrase "basic Logo procedures." Naturally, it got me thinking.

Somehow I could not get rid of the capital letters. I kept thinking about BASIC Logo procedures instead of basic ones. And the more I thought about them, the more uncomfortable I became with that (pause) word sitting right next to Logo. Suddenly, I just couldn't take it any longer. I became determined to get rid of the B-word and replace it with another one. Maybe I could find something a little less provoking.

But what replacement should I select? After prolonged and rather exquisite mental anguish, I finally settled on "fundamental." Yes, that would do nicely, at least for the time being. If that presenter had said "...fundamental Logo procedures..." I believe I would have smiled to myself and inclined my head ever so slightly in a nearly imperceptible nod, the way I learned to do at country auctions. "Fundamental" seems to have a sort of Charles Kuralt coziness to it.

OK, so what do I do with "fundamental"? What does it really mean? Ideas came tumbling in from all over. In music, fundamental is related to the chord based on the keynote. In physics, it refers to the part of a complex wave with the lowest frequency and also to the lowest natural frequency at which the air in a pipe vibrates. These and all the other dictionary definitions seemed to point to an importance stemming from some sort of foundational value. Not bad, but I wanted more.

Then I remembered a story told by Seymour Papert. He described a scene in a school where a group of children was coming out of a Logo class just as another group was going into the room.

The incoming children were curious. "What were you doing in there?"

The reply said it all. "It was fun. It was hard. It was Logo."

Fun. Sometimes we forget about something like that. Fun is definitely an important aspect of Logo, especially when fundamental is right next to Logo in a sentence!

But what else did those young students say? "It was hard." Hmmm. That means they had to think hard about what they were doing and what they were learning.

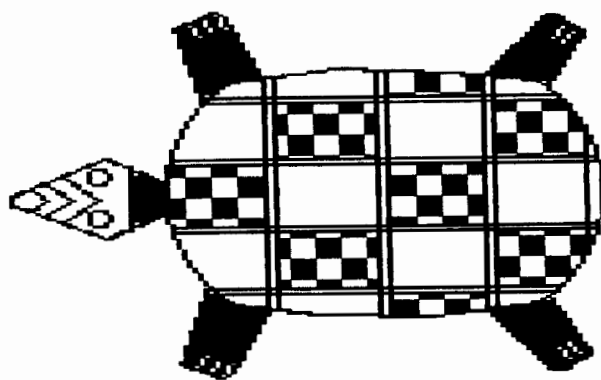
Then, there it was. Logo... fun and mental. Yessss! That feels just about right. At least for now.

After all this mental work and fun had settled down, though, I had to ask myself if it really made any difference. Was I going to produce bumper stickers or organize a protest march or picket the Department of Education?

No, none of the above. But the exercise certainly sharpened my awareness of the B- word. Nowadays, each time I see or hear it being used, I find myself smiling and whispering "Fundamental, fun and mental," even before I can catch myself. I just hope that doesn't get me into any trouble.

Fundamentally,
FD 100!

Tom Lough
Founding Editor
PO Box 394
Simsbury, CT 06070



Afro.turtle
Dahomej
By Frances Elliot & Jose McLean
(see page 7 for details)

Logo Ideas

Counting to a Million Revisited

by Eadie Adamson

In the December 1989 issue of *LX*, I wrote about an exploration of numbers in the context of a computer class. We had done a rather unsuccessful exploration using Logo to time the computer counting to a million. On an Apple IIe our procedure ran for several weeks without completing the task. Then someone turned it off overnight, which put an end to the experiment.

This year I am at The Dalton School, where I have the opportunity to work with several middle school math teachers developing meaningful Logo connections for the topics being studied in their classes. One fifth-grade class, taught by Debbie Coons, was exploring the concept of a million. Using ideas derived from last year's exploration, we decided to add a Logo connection to her class's explorations. By taking this idea examined in one context and looking at it in another, I found many new dimensions to the issues I raised in the December 1989 issue. These ideas form the beginning of what I hope during the coming year will be a series of mathematical explorations using *LogoWriter*, *LogoExpress*, or *LegolLogo*.

Tools or Not?

The first issue involving Debbie's class was a pedagogical one. We were working with Logo as an investigative tool. Although the students were familiar with *LogoWriter*, this was not a programming class. Thus, seeing what they did with Logo tools and helping them to use these tools to draw conclusions was of more importance than asking them to create the tools themselves. Because the tools were simple, I did not put a file on their disks ahead of time. I simply used a short handout for their reference and the large display monitor for them to refer to as we added to the tools or changed them. Again, whether to give the students a tool or ask them to create it themselves depends upon the purpose of the project. It's important to be aware that there are instances when creating a procedure or debugging a program obscures the purpose of the exercise. Teaching in Debbie's class was such an instance. I did not know most of these students, and I had no idea how much Logo they actually knew.

Dimensions of the Problem

Whether counting objects, letters, or numbers, it quickly becomes apparent that counting to a million is no small task. One student had already experimented with collecting a million paper clips and concluded, "A million paper clips is a very large amount." A million is a very large number on the

computer also. Logo's repeat command will not accept 1000000 as input. Nor will forward accept such a large number. *LogoWriter* will, however, print or show 1000000.

The students in this class already had preliminary experience with the concept of a million. They already had tried acquiring objects. They were thus aware of problems such as storage space. What will you do with a million paper clips if you can manage to assemble that many? They were interested in the time it would take to measure a million inches or feet. They were curious about how long it would take a computer to count to a million.

From my own prior experience, I already knew some problems we might face in such an experiment. If the number counted is printed on the *LogoWriter* page, the page runs out of space in a fairly short time. In the experiment described in my December 1989 article, our focus was more on dealing with programming problems such as these, although the time the computer took to count to a million was the problem we were researching. In planning how to look at the problem with a math class, both my prior experience and the prior experience of the math students were useful.

New LogoWriter Programs Help Explore a Million

I had some new tools to work with this year: *LogoExpress* and *LogoWriter* for the Macintosh, both of which contain a clock primitive. This primitive reports the date and time in *LogoExpress*, and the time in Macintosh *LogoWriter*. *LogoExpress* is accurate to the minute, while Macintosh *LogoWriter* is accurate to the second.

LogoExpress can run on Apple IIe, IIC, or IIGs computers, but unless a clock card is installed in the IIe or IIC, clock will report a string of zeroes. We also might have used *LegolLogo*, which has a timer primitive. (See the *LegolLogo* procedures at the end of this article.) Nonetheless, *LogoExpress* and Macintosh *LogoWriter* provided room for several interesting comparisons as well as a change in the way the problem might be approached. Now it would be possible to have the computer report the finish time. That meant that for a long test, the computer could be unattended, something which we could not do before if we wanted to know exactly when the computer finished counting.

The Starting Point

After discussing what we were going to do, I asked for some guesses. First, how long did the students think it would take a person to count to a million, and then how long would it take a computer to do the same thing? It seemed important that we formulate some beginning estimates. Whatever the

guesses, it would be instructive to return to them when we had an answer. The initial guesses could serve as a reference point: "Here's what we thought before we tried it, here's what we found out, and here's how different our initial picture was from the result." Surprisingly, a number of students thought a person could count faster than a computer.

The First Procedure

Since I already knew that to print the number would cause the page to fill too rapidly, I gave the students this procedure:

```
to countup :num
if :num > 999999 [print 1000000 stop]
show :num
countup :num + 1
end
```

By using `show :num`, the number would appear in the Command Center. When the Command Center is full, *LogoWriter* automatically clears the oldest entries. The students could watch the numbers as they increased.

Working in pairs, the students entered the procedure and got ready to try it out. One person in each pair of students was asked to count aloud while his or her partner watched the numbers on the screen. Oops! Quick revision of the initial ideas: it didn't take long to see that the computer counts far faster than humans can. It was time to think about other problems. Speech is slower than visual display! Just try saying 777,777 out loud. In general, the larger the number, the more time it will take us to pronounce it (in any language). Even saying "a million" or "one million" will take more time than it takes the computer to display the number and move on to the next. (Try changing the stop rule in line two of `countup`, above, to a larger number and start counting at 999999. Challenge someone to say "one million" before the computer has zipped far beyond it.)

Speeding Things Up

One group tried to race against another group. Since we had set all the GS computers to "fast" speed, this was not particularly meaningful, since the processors should all run at about the same speed. One student whispered to me, "How can we make ours go faster?" I quietly took her aside and explained that each thing the computer had to do extended the time of the operation, just as each syllable we had to pronounce in counting took time. Therefore, one simple change that would speed up the process was to remove the line that displayed the number, i.e.,

```
show :num
```

I helped her add a control key that would report her number when she wanted to see it:

```
when "z [show :num]
```

By this time, the rival group was aware that something was afoot. I promised to explain it to them, but suggested we see what difference the changes made by comparing the new process with the old one. It was, of course, clearly faster. *Everyone* decided to change the procedure and try the new method.

Soon came another question: "Isn't there another way to make this faster?" I quietly suggested not checking the progress so frequently. Time to relate this to human tasks: every time you are interrupted when doing something, the longer it takes to complete your task. Pressing the Control key and "z" to get a report interrupts the procedure very briefly, but when something runs as quickly as this counting procedure runs, even that small interruption is significant. Back to the computer to test this as well. Sure enough, the students who waited longer to check the progress were getting larger numbers.

Becoming Practical

It was becoming clear that the counting task would take more time than we had available. True, we could find enough free time to set up a test on a few computers. The big question then became: How much time would we need?

I asked for suggestions. Eventually the word "estimate" came up: we could estimate how long it would take. There were a few groans, because this really sounded like math!

"But wait. The computer can do the math for you," I said. "We just need to decide how to tell it what to compute. What should we be looking for?"

We decided to count to a small number several times and then average the results. Then we could determine what fraction of a million that number represented and then.... Clearly a little Logo help was needed. I suggested a way to begin.

We adjusted the `countup` procedure, deciding to count to 1000 instead of 1000000. We changed the stop rule so that it also would print the time when the computer finished counting:

```
if :num = 1000 [(print clock 1000)
stop]
```

We also needed to get a report of when the process began. I suggested we write a new procedure that used our `countup` procedure as its last step. This procedure would set up a control key to report the number and also print the starting time, making it possible to have the start and finish times on the screen for easy reference later:

```
to countmillion
  when "z [show :num]
  print clock
  countup 1
end
```

Here's another interesting question: When you start counting, do you start with 0 or with 1? If you look at this as a two-tiered process, is it "fairer" to use `countup 0` so that the `countup` procedure does all the work, including adding 1 to the initial 0?

Now we had several options open to us. We could rely on the GS clock, the Macintosh clock, or a watch or wall clock to time our experiments. Obviously only the GS or Macintosh solutions would display the start and finish time on the screen. Some students working with the GS quickly concluded that reports to the nearest minute would lead to large inaccuracies over a long period of time and therefore preferred to use a watch with a second hand.

Each group collected at least three estimates. Now we needed to find the average. I showed them the appropriate math symbols,

```
+    to add
-    to subtract
*    to multiply (the asterisk)
/    to divide
```

and let the groups figure out strategies for dealing with minutes and seconds. We eliminated writing procedures here, simply using `show` in the Command Center to get our results, adding a pair of numbers at a time. (All this could clearly be another math lesson later on!)

Note: Some versions of Logo are picky about infix notation and require spaces before and after these symbols. It's easiest just to assume your version requires them. Then procedures will transfer between versions and you won't need to worry about which rule applies to which version.

Once the estimates were done, it became clear that more than two class periods would be needed to run the test. It was also clear that with a fast GS or a Macintosh, this was not a "Logo Overnight" experiment!

There's More

In due time we ran our tests and got our results. We compared our original estimates with the results and then revised our ideas about computers, counting, and time. This time it was to be a "battle of the procedures," with competing methods timed against each other. There was another version of counting to try, a variation on the one Michael Tempel, of LCSU, had suggested earlier (see "Logo Ideas" in the December 1989 issue of *LX*):

```
to count.a.million
  make "num 0
  print clock
  repeat 1000 [repeat 1000 [make "num
    :num + 1] ]
  print clock
end
```

What makes this different is that this is not a recursive procedure. Why use `repeat 1000 [repeat 1000....]`? Our initial investigations had found that *LogoWriter* rejected large numbers for `repeat`. This is a way around the problem.

I shall leave it to you to try this test and compare the results. You might also try simply typing these commands in the Command Center and testing that against running a procedure. Would this be faster? Slower? Would they take the same amount of time? Can you explain why?

Next year we plan to use these experiments as an introduction to the "millions" project, rather than as a concluding exercise, since it clarified the concepts for many of the students. Besides, it was fun. And isn't that what real learning is about?

Procedures for Counting to a Million with *Lego/Logo*

Before you use the `countup` procedure below, type these commands in the Command Center:

```
ht
resett
```

These two commands will accomplish two important things: `ht` will protect you from inadvertently printing on the flip side of the page (a turtle command forces the screen to flip to the front; *Lego/Logo* has no `flip` command), and `resett` will set the timer in *Lego/Logo* to zero. Then `countup` will print the value of the timer when the procedure finishes. Unfortunately, you cannot use events (control keys) with *Lego/Logo*, so you'll need to shorten the count, rewrite the procedure so that it shows the count after each 1000, or find yet another solution:


```
to countup :num
if :num > 999999 [print :num print
timer stop]
show :num
countup :num + 1
end
```

Write **countmillion** as follows:

```
to countmillion
ht
resett
show timer
countup 1
end
```

Write **counta.million** like this:

```
to counta.million
make "num 0
ht
resett
print timer
repeat 1000 [repeat 1000 [make "num
:num + 1]
print timer
end
```

LogoLogo's timer reports time in tenths of a second. You'll probably want to express that value in minutes and seconds, but the accuracy of the report may actually be greater than with the other software versions above!

LogoExpress Note for Users of the New York LCSi Board

There is now a board called "tools" on the New York LCSi bulletin board. We have been posting pages of tools along with ideas on how to use your tool collection. Access the board by connecting with the New York host, then type

```
join "tools
checkpost
```

You will get a list of all the postings. If you've just joined the board and want to collect all the posts, be sure you have space on your disk since many of the postings are *LogoWriter* pages. To collect the first six postings, for example, you type

```
getpost [1 2 3 4 5 6 ]
```

When you are finished with the "tools" board, type

```
join "bbs
```

to get back to the main board.

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

About the Cover ...and more!

The beautiful drawings that grace the cover of this month's issue and that are interspersed throughout its pages were done by the students of Orlando Mihich. Orlando writes:

This school year, my students visited two sites in the New York City area: a thermoelectric facility and a waste-water treatment plant. They collected brochures and manuals, and now they are creating *LogoWriter* screens to explain the operation of these facilities. They will enter their projects in a city-wide competition and science fair.

These projects keep them, and me, very busy.

I like very much the screens my students created in "Optical Art," which were shown on the cover of the February 1991 issue; but I like even more the "Logo-Writer Afro Art" masks they created during last year's Black History Month. The masks are a painstaking creation; students go from one screen to the next and try to create the whole object, which is then visible only when printed.

Orlando Mihich, Technology Coordinator
Joan of Arc JHS
118 Magnet School
New York, New York

SHUFFLE:

A Logo Primitive to Rearrange the Items in a List

by Charles Crume

Two of Logo's more powerful features are recursion and the ability to work with words and lists. These capabilities make it is easy to write Logo programs that present each item in a list. For example, one could display the following list of fruits:

```
[APPLE BANANA BLACKBERRY PEAR PLUM
ORANGE]
```

and ask a child to enter the color of each fruit as its name is displayed.

However, once a child uses such a program several times he or she may begin to remember the order in which the fruits are presented and the corresponding answer. This would probably reduce the usefulness of the program. Rearranging the elements of the list every time the program is run would inject a degree of randomness and help prevent the child from remembering which answer comes next. Although most dialects of Logo contain a random number function to generate random numbers, no dialect contains a primitive to rearrange the elements of a list randomly. This article presents such a primitive.

There are, in fact, many real-life instances in which the same list of items appears in a different order each time the items are encountered. A prominent example would be a deck of playing cards. A deck of cards consists of four suits (hearts, diamonds, clubs, and spades) with each suit containing the values 2 to 10, Jack, Queen, King, and Ace. Before playing any game, the deck is shuffled.

The following *LogoWriter* procedures will rearrange (i.e., shuffle) the contents of a word or list. The primary procedure is called SHUFFLE. SHUFFLE requires the sub-procedure BUTITEM (Crume, 1989). The code for both procedures is shown below:

```
TO SHUFFLE :MYLIST
IFELSE (COUNT :MYLIST) = 1 [OUTPUT
  FIRST :MYLIST] [MAKE "MYLIST LIST
  (RANDOM (COUNT :MYLIST)) + 1
  :MYLIST OUTPUT SENTENCE ITEM FIRST
```

```
:MYLIST LAST :MYLIST SHUFFLE
BUTITEM FIRST :MYLIST LAST
:MYLIST]
```

END

```
TO BUTITEM :MYITEM :MYLIST
IFELSE :MYITEM = 1 [OUTPUT BUTFIRST
:MYLIST] [OUTPUT SENTENCE FIRST
:MYLIST BUTITEM :MYITEM - 1 BUT-
FIRST :MYLIST]
```

END

Note: It is beyond the scope of this article to discuss the way in which SHUFFLE works, but those having questions should send them directly to the author. (Please include a self-addressed, stamped envelope.)

The procedure SHUFFLE requires one input—the word or list whose elements are to be shuffled. SHUFFLE reports the word or list after its elements have been randomly rearranged. For example, the command

```
PRINT SHUFFLE [A B C D E F]
```

might produce any of the following as output:

```
[F C E B A D]
[C D A B F E]
[C A F B D E]
[E A B F C D]
[C B E A F D]
```

The example above is, of course, not very useful in everyday life. A more substantial example, which shuffles an electronic deck of cards, is shown below:

```
TO DEMO
PRINT SHUFFLE DECK [] 3
END
```

```
TO DECK :DECK.OF.CARDS :FIRST.SUIT
IFELSE (COUNT :DECK.OF.CARDS) = 52
[OUTPUT " ] [OUTPUT SENTENCE
:DECK.OF.CARDS ALL.SUITS
:FIRST.SUIT DECK :FIRST.SUIT + 1]
END
```

```
TO ALL.SUITS :SUIT
IFELSE :SUIT > (:FIRST.SUIT + 3)
[OUTPUT " ] [OUTPUT SENTENCE
THIS.SUIT :SUIT 1 ALL.SUITS :SUIT
```

```

+ 1]
END

TO THIS.SUIT :SUIT :N
IFELSE :N > 13 [OUTPUT "] [OUTPUT
  SENTENCE CARD :N :SUIT THIS.SUIT
  :SUIT :N + 1]
END

TO CARD :VALUE :S
IF :VALUE = 1 [OUTPUT WORD "A CHAR
:S]
IF AND (:VALUE < 11) (:VALUE > 1)
[OUTPUT :VALUE CHAR :S]
IF :VALUE = 11 [OUTPUT WORD "J CHAR
:S]
IF :VALUE = 12 [OUTPUT WORD "Q CHAR
:S]
IF :VALUE = 13 [OUTPUT WORD "K CHAR
:S]
END

```

The procedure DECK requires two inputs. The first input is an empty list that will contain the electronic deck of cards after the procedure finishes. The second input, which is the number 3, is the value whose graphic representation is the heart symbol on an IBM/PC. The graphic representations of the numbers 4, 5, and 6 are diamonds, clubs, and spades, respectively. This can be demonstrated by the command

```
(PRINT CHAR 3 CHAR 4 CHAR 5 CHAR 6)
```

which will produce

```
♥ ♦ ♣ ♠
```

The procedure DECK produces a list of electronic cards as shown below:

```

A♥ 2♥ 3♥ 4♥ 5♥ 6♥ 7♥ 8♥ 9♥ 10♥ J♥ Q♥ K♥
A♦ 2♦ 3♦ 4♦ 5♦ 6♦ 7♦ 8♦ 9♦ 10♦ J♦ Q♦ K♦
A♣ 2♣ 3♣ 4♣ 5♣ 6♣ 7♣ 8♣ 9♣ 10♣ J♣ Q♣ K♣
A♠ 2♠ 3♠ 4♠ 5♠ 6♠ 7♠ 8♠ 9♠ 10♠ J♠ Q♠ K♠

```

The author challenges users of the Apple dialect of *LogoWriter* to duplicate the display of DECK using the features of Apple *LogoWriter*.

Reference

Crume, C. E. (1989). A new primitive for *LogoWriter*. *Logo Exchange*, 8(4), 25-26.

Charles E. Crume has over 15 years of experience in the electronics and data-processing industry. Currently, Mr. Crume is an independent computer consultant running his own company on Florida's Emerald Coast. Previous positions include senior technical consultant, software systems analyst, telecommunications analyst, and programmer. Mr. Crume has been using Logo and *LogoWriter* for several years and has taught courses in *LogoWriter* at the college level. He can be reached at

FACTOTUM Software
P. O. Box 8874
Reno, Nevada 89507

NECC '91

National Educational
Computing Conference
June 18-20, 1991
Phoenix, Arizona

NECC '91 provides a broad, rich forum for discussion among individuals interested in educational computing. This year's theme, *Solutions*, refers to both human and electronic solutions. NECC's goals:

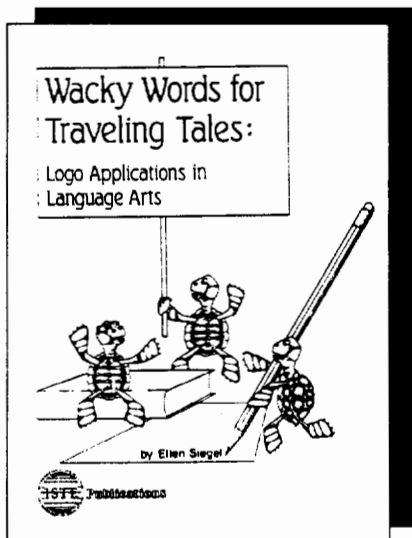
- To present a spectrum of major work regarding computers in education
- To promote interaction among individuals using computers in education at all educational levels
- To develop and coordinate the activities of professional groups involved with computers
- To produce the proceedings, documenting the status of educational computing.

For more information, contact:

NECC '91/ISTE

1787 Agate St., Eugene, OR 97403-1923
Ph. 503/346-4414 FAX 503/346-5890

Wacky Words Work Wonders!

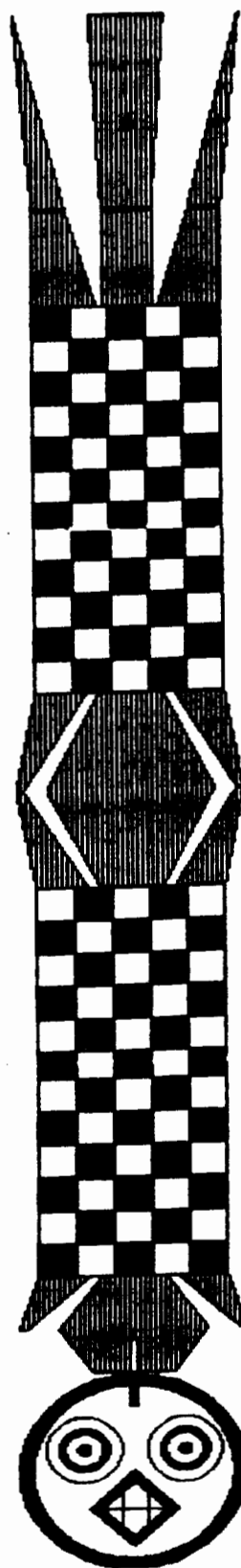


Wacky Words for Traveling Tales: Logo Applications in Language Arts

Add the powerful Logo environment to your current Language Arts activities. Students use or edit a set of Logo tools to create their own sniglet dictionaries, dynamic poetry, and choose-your-own-adventure fairy/folk tales. The activities—on-computer and off—emphasize discovery and collaborative learning, and are appropriate for a variety of levels in the elementary language arts curriculum. Each unit contains lesson plans, overheads, instruction sheets, and student worksheets. *LogoWriter* "tools" on disk can be used by students who have no knowledge of programming and minimal knowledge of *LogoWriter*. (Apple *LogoWriter* required, not included.) By Ellen Siegel; \$25.00 plus shipping; 120 pages, 1 data disk.

ISTE
1787 Agate Street
Eugene, OR 97403-1923

Bwa plank mask
Village of Boni,
Burkina Faso.
By Michael Nevarez
(see page 7 for details)



Beginner's Corner

Logo-to-Logo Exchange by Dorothy Fitch

One of the challenges Logo enthusiasts must face at one time or another is how to deal with the differences among versions of Logo. You may want to use a Logo program in your computer education magazine. But alas, it is not written in your version's dialect, so it does not work! Your version of Logo does not understand all the commands of another version.

My current count indicates that there are (or have been) at least eight versions of Logo for Apple computers, four versions for the Macintosh, two for the Commodore, three for the IBM, and one each for other brands of computers (Atari, Amiga, Adam, Digital, and so on). There are also a number of foreign language versions of Logo for various machines. These are all complete versions of Logo, with both graphics and list processing capabilities. You may know of even more!

Each version of Logo has its own set of commands. Fortunately, a large number of the commands are the same in all versions, and many more are similar. Once you know where the differences lie, it is not hard to translate a program from one version of Logo to another.

This month we'll take a look at some of these differences, particularly those that beginners are likely to encounter. Rather than list the equivalent commands for more than 20 versions, we'll look at the most commonly used commands and versions of Logo.

First, here is a brief historical overview of Logo, which might help you understand why these differences exist.

Some books refer to MIT Logo syntax. This is the original version of Logo developed at the Massachusetts Institute of Technology in the 1970s under the direction of Seymour Papert. When MIT made it available for commercial distribution, two companies—Terrapin and Krell—both won rights to it. Since that time, Terrapin has substantially added to the features of the original language. *Terrapin Logo for the Apple*, *Logo PLUS*, *Krell Logo*, and *Commodore Logo* are all derived from this original Logo and hence have the same general command structure.

Shortly after the first commercial release of the MIT version of Logo, a group of programmers (some of whom worked on the original Logo at MIT) produced *Apple Logo*, which was sold through Apple Computer, Inc. This group, Logo Computer Systems, Inc., also produced a number of

versions of Logo for other machines. More recently, they produced *LogoWriter* and *LogoExpress*. These Logo languages use what is often referred to as LCSI syntax. However, even among the languages made by this company, the commands are not identical.

Other publishers have used a blend of these syntaxes in their versions of Logo. As newer languages offer even more features, the line between LCSI and MIT syntax becomes fainter and fainter.

Here is an attempt to organize the various command sets so that you can more easily convert programs from one version of Logo to another. The tables below list the most commonly used versions of Logo. If you have another version, one of the commands listed will most likely work for you. Note: *Terrapin* refers to *Terrapin Logo for the Apple* and *Logo PLUS*; *LogoWriter* refers to *LogoWriter Version 2.0* and *LogoWriter for the Macintosh*; *Apple Logo* refers to the version of Logo (no longer available) sold by Apple Computer.

Note: If you are not sure which command to use for your version of Logo, the easiest way to find out is to type the name of the command. If your version has that word as a command, you will see a message that says something like

_____ NEEDS MORE INPUTS

RESULT: _____

I DON'T KNOW WHAT TO DO WITH _____

Or you may not receive any message at all. If your version of Logo does not have the command, you will see a message that says something like

THERE IS NO PROCEDURE NAMED _____

I DON'T KNOW HOW TO _____

Graphics Commands

These graphics commands work in the same way in all *current* versions of Logo. Their abbreviations are shown in parentheses. Some versions of Logo do not include the complete command name, such as *HIDETURTLE* or *PEN-DOWN*. The abbreviations will always work.

FORWARD	(FD)	RIGHT	(RT)
BACK	(BK)	LEFT	(LT)
PENUP	(PU)	PENDOWN	(PD)

Table 1

	<i>Terrapin</i>	<i>LogoWriter</i>	<i>Apple Logo</i>
To clear the screen and place the turtle at home:	DRAW	CG	CLEARSCREEN CS
Clear the screen without moving the turtle:	CLEARSCREEN CS	CLEAN	CLEAN
Change the turtle's pen color:	PENCOLOR PC	SETC	SETPC
Reverse the pen color:	PENCOLOR 6 PC 6	PX	PX
Change the screen's background color:	BACKGROUND BG	SETBG	SETBG
Move the turtle to a coordinate position:	SETXY <i>x y</i>	SETPOS [<i>x y</i>]	SETPOS [<i>x y</i>]
Find out the turtle's coordinate position:	LIST XCOR YCOR	POS	POS

Table 2

	<i>Terrapin Versions</i>	<i>LogoWriter</i>	<i>Apple Logo</i>
To determine if the word or list is empty:	EMPTY?	EMPTY?	EMPTY?
To determine if a word or list is a member of another word or list:	MEMBER?	MEMBER?	MEMBERP
To print the word or list, not followed by a return character.	PRINT1	TYPE	TYPE

Table 3

	<i>Terrapin</i>	<i>LogoWriter</i>	<i>Apple Logo</i>
Reports TRUE if all expressions are true:	ALLOF	AND	AND
Reports TRUE if any expression is true:	ANYOF	OR	OR
Stop all procedures and return to top level:	TOPLEVEL	STOPALL	THROW "TOPLEVEL

Table 4

	<i>Terrapin</i>	<i>LogoWriter</i>	<i>Apple Logo</i>
Reports a keystroke from the keyboard:	READCHARACTER RC	READCHAR	READCHAR
Reports a list from the keyboard:	REQUEST	READLIST	READLIST
Reports whether a key has been pressed:	RC?	KEY?	KEYP

PENERASE	(PE)	HOME	
HIDETURTLE	(HT)	SHOWTURTLE	(ST)
HEADING		SETHEADING	(SETH)
XCOR		YCOR	
REPEAT			

Some versions offer these commands, which will always work the same way:

FILL
COLORUNDER

The graphics commands given in Table 1 differ from version to version. (Inputs to commands are shown only when their forms vary.)

Words and Lists

Most of Logo's word and list commands are identical in all versions. Here are the commands you can count on to work in same way:

FIRST		LAST	
BUTFIRST	(BF)	BUTLAST	(BL)
COUNT		ITEM	
SENTENCE		WORD	
LIST		PRINT	(PR)
MAKE		LOCAL	
ASCII		CHAR	

The word and list commands in Table 2 differ from version to version.

Conditionals, Logical Operators, and Control

These commands work the same way in all versions:

NOT	OUTPUT	(OP)
RUN	REPEAT	
STOP		

However, the commands in Table 3 differ from version to version.

The IF statement may need changing when you try to convert a procedure from one version to another. In example A of each syntax below, the commands are run only if the expression after IF is true. In example B, the first set of commands is run if the expression is true, and the second set of commands is run if the expression is false.

Terrapin: A: IF *something.is.true* THEN *commands*
(THEN is optional)
B: IF *something.is.true* THEN *commands*
ELSE *commands*

LogoWriter: A: IF *something.is.true* [*commands*]
B: IFELSE *something.is.true* [*commands*]
[*commands*]

Apple Logo: A: IF *something.is.true* [*commands*]
B: IF *something.is.true* [*commands*]
[*commands*]

Getting Input from the User

Table 4 shows how the commands for getting input differ from version to version.

Summary

There is occasional talk of devising a Logo standard, a set of commands to which all versions of Logo would adhere. It is not clear who would do this or how it would happen. It seems that to get the Logo community and publishers to agree to a set of commands and then to get the publishers to rewrite their software, documentation, and materials would be nigh unto impossible! So for the time being, we must make the best of this situation. Can we claim to be multilingual?

P.S. Terrapin's Technical Tip #25 contains guidelines for converting programs between *Logo PLUS* and *Terrapin Logo for the Macintosh*. Send a self-addressed stamped envelope to Terrapin for a free copy.

A former education and computer consultant, Dorothy Fitch has been the director of product development at Terrapin since 1987. She can be reached at:

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

Logo LinX

The Handwriting on the Screen

by Judi Harris

Handwriting...ugh! That was the part of the upper elementary curriculum I dreaded teaching the most. I wanted my students to experience learning as an intellectually stimulating, interesting, challenging, *relevant* endeavor. Handwriting skills may be exciting for eight-year-olds, but my prepubescent pupils found them, at best, boring and repetitive; at worst, a frustrating four years of building fine motor skills.

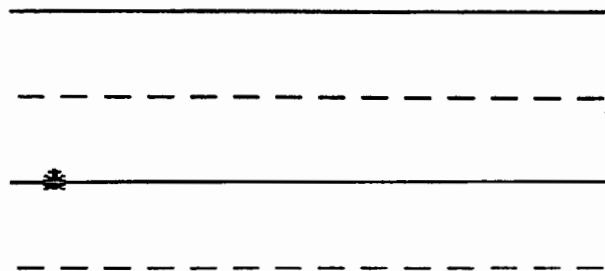
How might I make penmanship, at best, intriguing and pleasant or, at worst, tolerable? Eleven years ago, when our school microcomputers were still being used for BASIC and *Bank Street Writer* (Version 1), I taught my sixth graders to do simple calligraphy. I suspected that if handwriting could be seen and practiced in a new way (as an art form) by these reluctant pen-wielders, the readability and general aesthetic appearance of their handwritten work might improve voluntarily instead of under teacher-imposed duress. After all, I reasoned, they had been practicing Zaner-Bloser letters for more than one-third of their lives; it wasn't perfect letter formation that I should target but decreased teacher eyestrain, right? Readability should be the prime directive in upper elementary and middle school penmanship practice, right?

Well, at least my principal didn't stop the calligraphy lessons. (He had bad handwriting himself; had his handwritten notes to faculty been more readable, I might not have been permitted to restructure this curricular requirement so "creatively.") The students were mildly intrigued with the ancient-looking letters, and there was some improvement in the graphic clarity of their non-word-processed papers (as long as they didn't write them on the bus). But, as you might predict, the students with the best cursive writing were also the ones who presented beautiful calligraphied homework papers. The problem of perspective had been addressed, but attitude was improved with another alphabet rather than with the prescribed cursive content.

Now that Logo, *LogoWriter*, and *Logo PLUS* are common to intermediate-level classrooms, handwriting can be explored in another divergent way: as a vehicle through which to explore arc radii and degree measurements. In a way, the process-centered assumption behind this instructional suggestion is similar to my decade-old calligraphic curricular rebellion. If students are encouraged to examine cursive letter construction in conscious detail and from a new perspective, perhaps their handwriting will improve by default.

Draw It on the Line

Remember the special penmanship paper that made you feel like such a grown-up in those early years of school? Reproducing it on the *LogoWriter* screen is an interesting beginning-level programming challenge. My lines look like this, and are saved in a picture file called LINES.

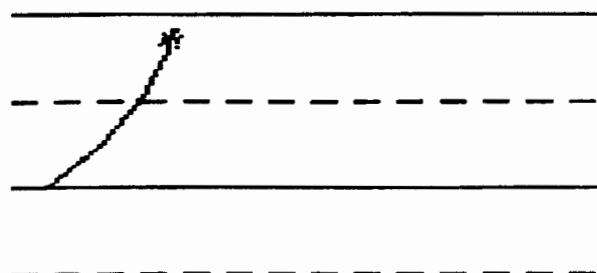


The STARTUP procedure on the CURSIVE.WRITING *LogoWriter* page on my disk includes a series of commands that clear the screen, recall the "LINES" picture, and position the turtle so that it can begin to form cursive letters:

```
CG
LOADPIC "LINES
PU
SETPOS [-120 -15]
PD
ST
```

The Zaner-Bloser *Transition Recorder* (Hackney, Myers, & Bloser, 1969), used in many elementary schools to assist children in switching from manuscript writing (printing) to cursive writing, presents the letters *h* and *i* as the first steps in crossing the bridge to handcrafted script. In this case of mathematics actively explored through practicing turtled penmanship, the bridge is curved into arcs of various radius sizes and degree amounts. Together, the two introductory letters spell a friendly word that the friendly turtle can be taught to write: *hi*.

Let us begin with the first stroke of the *h*, an "undercurve stroke":



In *LogoWriter*, this becomes

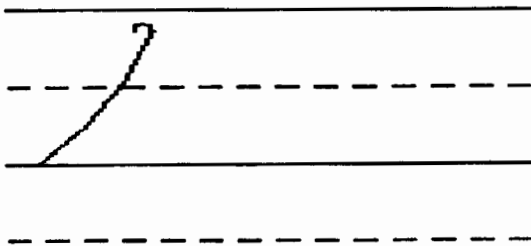
```
RIGHT 50
ARCL 200 30
```

ARCL is a tool procedure on the *LogoWriter* page TURTLE.TOOLS, included on the *LogoWriter* language diskette. It draws a user-specified degree section of a circle of user-specified size that turns to the left:

```
TO ARCL :RADIUS :DEGREES
MAKE "STEPS (2 * :RADIUS * 3.1416 /
36)
MAKE "REM REMAINDER :DEGREES 10
REPEAT :DEGREES / 10 [LEFT 5 FORWARD
:STEPS LEFT 5]
IF :REM > 0 [FORWARD :STEPS * :REM /
10 LEFT :REM]
END
```

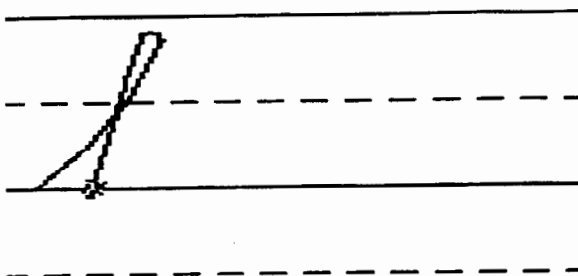
Now for the top of the *h*. If you look closely, you will see that it is really a 180-degree arc from a comparatively small circle. In *LogoWriter* commands, this can be expressed as

```
SETH 10
ARCL 5 180
```



Thus far, the cursive letter *h* has begun to take shape as a 30-degree arc from a 200 turtle-step circle and a 180-degree arc from a 5 turtle-step circle. The next stroke is a straight one, for which the turtle should be slightly repositioned.

```
RIGHT 5
FORWARD 90
```

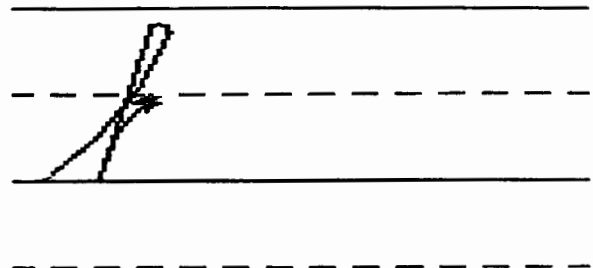


Now the turtle must partially retrace its steps, reset its heading, and draw an arc in the opposite direction to the previous two. The ARCR procedure, also included on the TURTLE.TOOLS page, is identical to ARCL with the exception of the turn commands:

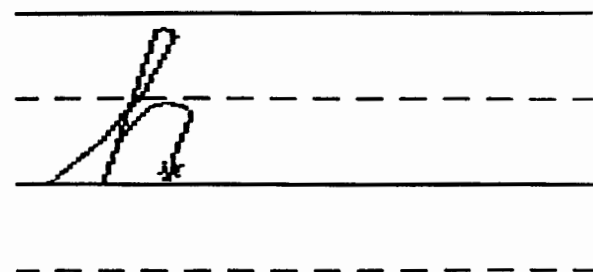
```
TO ARCR :RADIUS :DEGREES
MAKE "STEPS (2 * :RADIUS * 3.1416 /
36)
MAKE "REM REMAINDER :DEGREES 10
REPEAT :DEGREES / 10 [RIGHT 5 FORWARD
:STEPS RIGHT 5]
IF :REM > 0 [FD :STEPS * :REM / 10
RIGHT :REM]
END
```

Using ARCR, the next steps in the formation of our cursive *h* can be taken:

```
RIGHT 180
FORWARD 25
RIGHT 10
ARCR 60 25
```



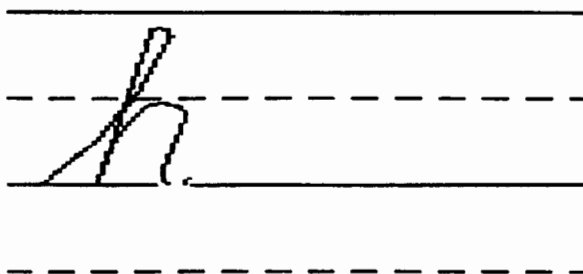
If you look closely, you will see that this section of the *h*, although appearing as one uninterrupted curve, is actually the concatenation of two right arcs from circles of very different radii:



I drew this portion of the letter with these commands:

```
ARCR 12 105
RIGHT 43
FORWARD 35
```

Now the turtle must reverse its heading while drawing another arc. Notice that the radius of this arc is of almost the same size as that of the arc that formed the uppermost portion of the *h*:



```
ARCL 6 180
```

Voila! A beautiful cursive *h*. The *i* should be a much simpler mathematical matter.

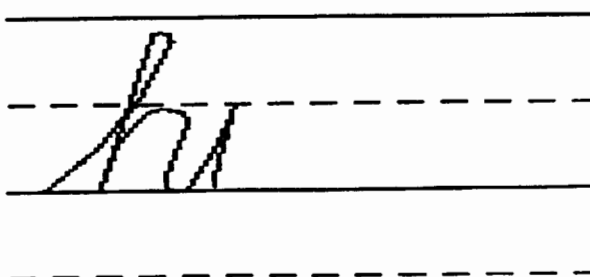
An Eye for an *i*

Here is another relatively straight undercurve stroke. I changed the turtle's heading and drew it like this:

```
RIGHT 15
ARCL 150 15
FORWARD 10
```

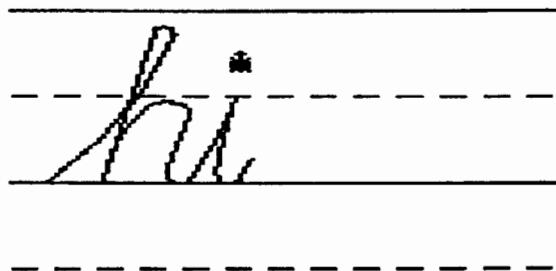
According to the Zaner-Bloser model, the uppermost part of the *i* is then retraced, leading into a slight left arc:

```
RIGHT 180
FORWARD 4
ARCL 150 15
```



Only two steps remain—putting the “tail” on the *i* and drawing its dot:

```
ARCL 6 180
RIGHT 25
FORWARD 10
```



Turtled cursive writing may be time consuming, but it tinkers with a familiar form that is most mathematically satisfying. Want to try it? Exploring the geometrical structure of letters can be an intriguing way to restructure a tedious portion of required curriculum.

Reference

Hackney, C.S., Myers, E.H., & Bloser, P.Z. (1969). *Off we go! Transition recorder*. Columbus, OH: Zaner-Bloser, Inc.

Judi Harris works as an assistant professor of educational technology at the University of Nebraska at Omaha, where she helps teachers learn to use computer-related technologies in their teaching. She is very glad to have use of numerous microcomputers there so that she does not have to form cursive letters very often with her fingers, and she is especially thankful that she does not have to teach penmanship at either the undergraduate or graduate level.

Bitnet: JHarris@unoma1
Internet: JHarris@Zeus.unomaha.edu
CompuServe: 75116,1207

Questions Please!

by Frank Corley

This month we're having a clearance sale on questions. I'm cleaning out my file folder to get all the unanswered questions and all the unquestioned answers printed by the end of the year. I am using up my supply, so I hope I'll have something to fill next month's column. Herewith are the questions and answers.

Pete Gerum of Kamehameha Schools in Honolulu sends these queries:

1. I use *LogoWriter 2.0* and like to write programs for my class using the RANDOM function. I type the class list and Logo randomly picks a name. Naturally some names get picked frequently and others do not. I'd like to know how to write a procedure that randomly picks a name but then drops that name from the list and then continues to pick a name at random from the remaining names in the list until the list is exhausted. Then I'd like to be able to restore the full class list.
2. I haven't found a solid reference book for *LogoWriter*. I'm looking for something more than just the manual of commands that LCSi provides. Their primary and secondary series aren't very satisfying either. I'm looking for *LogoWriter* list-processing projects in order to learn that side of *LogoWriter* better. I'm seeking a book like *Nudges* by Bull or *Apple Logo* by Watts in *LogoWriter* format.

Jackie Griffith, a third-grade teacher in Tucker, Georgia, sends these requests, which have a familiar ring to them:

In January we had four IBM computers networked. That gives us eight computers—with three Apples and a TI/99/4A. With the old TI, we had 20 sprites that could all carry colored balloons or balls and throw them all over the screen; my teenage students could draw joggers jogging across a field or on a highway with a yellow line and a bird flapping overhead.

3. We had a canned book for the TI written by Diane R. Musha; is she still around?
4. I am finding it difficult to run *LogoWriter* with just simple instructions. My children are forever getting lost or don't quite know what to do. I know it will get better but my question is: Is there a primary guide already written other than the material in our kits? I have written some sheets, but has someone already produce other resources?

5. I would also love to go to a Logo conference or workshop. Are there any in the Southeast? I did go to a Summer Math program at Mr. Holyoke two summers ago and enjoyed doing Logo again.

I also have a supply of "big" questions left over from earlier columns. Here they are.

6. Many people have many ideas about the results that Logo, at least in theory, has on its students. Are these in fact true, or even realistic? What must take place in order for the lofty ideas about reasoning skills and higher-order thinking and analysis to be realized? What about assessment? Is there a need for assessment? What can be assessed? What kinds of assessment are there?

The question above is obviously a key one for many educators. I am sure that studies have addressed this question, and perhaps we can have a summary of the research in this column or have a more extensive article written up for this journal or *The Computing Teacher*. I hope that where different studies offer conflicting views, the points of disagreement will be explained.

7. Other than objective, standardized sorts of tests, how does one write tests for computer classes to give grades in a Logo class? What are some good test questions and formats? How do you grade work in Logo?

The question above is an important one for computer science teachers, especially those who teach at schools that offer advanced placement courses in computer science.

8. How does one use Logo as an introduction to other languages, in particular, Pascal and LISP?

I think the next question is very interesting. It has obvious ramifications for social studies classes and in that context the answers may be very complicated. As a math teacher, I think of the famous Four-Color Theorem, and I can envision some exciting Logo activities that involve such material.

9. How do you get a teacher to the point where he/she can help a student with such activities as drawing a map to go with a written project in Logo?

I am not quite sure what this next question means, but it does seem important. I invite the asker to elaborate on the question because it seems like he or she has a more specific issue in mind. I will transcribe it here exactly as it was posed to me.

10. Where are the "tools" to extend the ideas that maybe are already existent in another piece of software?

Glenda Bentz of Oak Hill School here in St. Louis sends an answer to an old question, which first appeared in the October 1990 issue and again in the December/January 1990-1991 issue of *LX*. The question was: In *LogoWriter*, how can I override the `Printtext80` command to print 10 characters per inch and fill the page leaving one-inch margins on both sides of the paper? The answer is:

```
TO PRINTCOLUMN :WIDTH
.DEPOSIT 38547 :WIDTH - 1
PRINTTEXT80
END
```

Glenda says that this appeared in *LCSI Logolink* a couple of years ago. She shortens the procedure to `PC :WIDTH`. If an *ImageWriter II* is used, PC 85 will give one-inch margins if the paper is aligned properly.

Another potential answer to an old question comes from Dorothy Fitch of Terrapin. The question is: What do I do with a full disk? We assume that the user has thought of the usual answers like "They make great coasters for the coffee table" and "Have you tried them as Christmas ornaments or using six for a birdhouse?" Here are Dorothy's suggestions:

ProDOS only lets you save 55 files on a disk without the use of subdirectories. There is no more room in the directory file (where the catalog of the disk is stored), even though there may still be free space on the disk. Files can still be read and used, but not saved to the disk anymore. This is a PRODOS problem, not a Logo problem. If you can find one file to erase, or two to combine, then you can create a directory. (Its name also takes up room in the directory file.) Then you can switch to that directory and save your files there. If the questioner is still uncertain, s/he should contact Dorothy at Terrapin.

I hope that next month I will be able to respond to many of these questions so that we can wrap up the year's volume of issues nicely. Until then, I hope that all your questions are answered, and if they're not, please send them to me.

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

CALL FOR PAPERS

Two Conferences in San Jose, Costa Rica

V INTERNATIONAL LOGO CONFERENCE

November 4, 5, and 6, 1991

INTERNATIONAL MEETING ON EDUCATION AND TELECOMMUNICATION TECHNOLOGY

November 7, 1991

Invited Speakers

Presentations of Papers

Demonstration of Programs and New Applications

Videotape Exhibitions

Hands-on Experiences and Workshops

Social and Cultural Activities

CLOSING DATE FOR SUBMISSION OF PAPERS

May 15, 1991

Full versions of the papers, in a 8 1/2 x 11 inch format, with an abstract no longer than 200 words should be sent before May 15, 1991. Individuals willing to present videotape exhibitions, hands-on experiences, workshops, or other types of demonstrations should send a detailed description of their activity before May 15, 1991. Submitters will be notified of acceptance or rejection before July 31, 1991. Closing dates apply to both conferences.

Send submissions to, or for more information contact:

Prof. Francisco Quesada, Comité Organizador
CONFERENCIAS 1991

FUNDACION OMAR DENGÓ, Apartado 1032-2050

San Jose, Costa Rica

FAX (506) 22 16 54

Internet: quesada@huracan.cr

BITNET: huracan!quesada@uunet.uu.net

MathWorlds

edited by

A. J. (Sandy) Dawson

The Council for Logo in Mathematics Education (CLIME) is an affiliate group of the NCTM. Ihor Charischak and George Bright were instrumental in obtaining affiliate group status for CLIME. Consequently, much credit must go to these gentlemen for their efforts in awakening mathematics teachers in North America to the importance of the creation of Logo-like environments. As president of CLIME, Ihor has been attempting to collect and disseminate software that is Logo-like in its conceptualization and manifestation. The article below describes his recent efforts in this area.

Filling in the Gaps with Logo Software

by Ihor Charischak,
President of CLIME

As you may already know, CLIME is collecting, developing, and sharing Logo *microworlds*. These are Logo applications that teachers and students can use to explore a variety of math topics in a Logo-like way. What you probably don't know is that this collection almost didn't get off the ground because we were stuck deciding whether each application was worthy of being called a *microworld*, that is, whether it was *microworldly* enough (see *CLIME News*, 2,[2] for a discussion of microworlds). Fortunately, we realized that (1) microworldliness is a matter of opinion and (2) most people don't really care whether an application is a microworld or not, so long as it is interesting, useful, or both! So we stopped nitpicking and put together the contributions our readers sent us. If you get the disk from us, what you will discover is a collection of activities that each author found useful and interesting. Along with the disk we include printed instructions and information on how the author used the activity and why he or she found it useful.

Since some of these examples have the look of commercial software, I collectively call them (for lack of a better name) *Logo software*. Logo software comes in many flavors, including puzzles, simulations, tutorials, quizzes, and, yes, even (heaven forbid) drill and practice (done creatively, of course). The advantages of Logo software over other software are that

- The code is accessible. Teachers and students can use the code to learn more Logo.
- The code is modifiable. The teacher or student can change the software to meet individual needs.

- The code is never done. Like a chameleon the program will adapt and change with the needs of the people using it.
- It is cheap! Once you've paid for the Logo program, all the other programs you create with Logo are free (well, almost). Except, of course, if you wish to write your own, then it will cost you some time.

Though many CLIME members use the microworlds in their teaching, the reality is that the computer revolution has touched the lives of only a small percentage of math teachers. Even if these math teachers would like to use computers, most of them are not willing or able to go out of their way to get computers into their classrooms. Yet from my experience of talking with many of them, they would welcome the opportunity to learn. So how can we make computers and, subsequently, the power of Logo available to them? What follows is a description of a possible scenario that currently has my attention.

Computers in Mathematics Education: A Project Approach

The Center for Improved Science and Engineering Education (CIESE), which is based at Stevens Institute of Technology in Hoboken, New Jersey, is helping five school districts in New Jersey integrate computers into their mathematics curriculum in grades 7 through 12. There are 60 experienced mathematics teachers in the project, all of whom are new to computing. Since the project's philosophy is not to endorse any specific way of using computers, teachers get to experience a variety of approaches to using computers and computer languages, including Logo, during summer workshops and at monthly meetings during the academic year. Though many of them responded favorably to Logo, most felt that it would be easier to begin with CAI or graphics software. After reviewing many pieces of software and thinking about how they would use the computer in their classrooms, many of them found examples that they felt were useful in beginning their experimentation. They also discovered that there was a lack of good software, particularly in the area of seventh- and eighth-grade geometry. (I'm eagerly looking forward to seeing *Logo Geometry*, by Clements and Battista, (1991); I think it will fill some gaps.) They all saw that Logo had potential here, but only a few volunteered to learn enough Logo to move in that direction. I noted who the interested Logo folks were and began to talk to them about Logo software projects.

The Evolution of an Idea

During our workshops last summer, one of the high school teachers complained that she could not find a program

to introduce students to the ideas of coordinate geometry. "I want a program that not only draws the plots of functions but also can plot individual ordered pairs. I want students to understand that every point on a line has meaning," she said. At this point, I showed her Henri Picciotto's (1989) *Logo Math Tools and Games*, which she said was fine, but she wanted something *simpler*. Though it wasn't clear to me exactly what she meant, I suggested that she and I work together on a Logo project. The suggestion appealed to her since she had some previous Logo experience. For the next few hours, she began work to create what she wanted. Later we collaborated and improved her work. By the next day, we had a working version of a graphing program that was ready for trial with her students. Next month I will be meeting with her and visiting her classroom to see how she is doing. Probably we will do more revision and development. Hopefully, when we finish, we will be able to share her Logo software with the other schools in our project. They probably will want to make changes in the software and to adapt it to their specific needs. In these cases, teachers may not be interested in making the changes themselves, so another member of the project team would do the revision. At Stevens we have skilled graduate students who do this kind of programming work. In this case, the teacher becomes the designer of the software and works closely with the person who actually writes the code. The reactions of students are actively solicited, and they become partners in the development of the application. Who knows, even some of the students may become inspired to take on some of the programming chores! What's interesting is that in each district this graphing unit will take on a different look reflecting the needs and teaching styles of the people involved.

This project approach to creating Logo software is a push towards developing a rather radical approach to learning within traditional schools. Students become partners with the teachers in these classroom experiments. It is a way to get Logo into the secondary schools in a more realistic and plausible manner. It is also a move to extend the Logo environment beyond the walls of the classroom to include university people, administrators, computer teachers, and parents as collaborators on developing a context for learning where Logo is just one of the significant pieces.

Seymour Papert introduced us to the turtle as a transitional object or bridge that connects the world of ideas with physical objects. I see Logo software projects as a vehicle for helping teachers to broaden their approaches to teaching and begin viewing their job differently. In this context they see themselves as creative directors and managers of learning environments, where they take advantage of a variety of

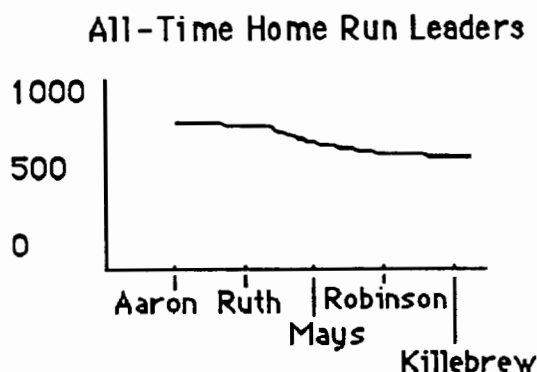
resources, experiment with new ideas and strategies, and work towards unraveling the mysteries of learning.

This will, of course, take patience, energy, and perseverance over a long period of time. The CIESE staff is aware of the difficulty of this task, particularly since most projects succumb to a natural resistance to change and a tendency to drop a project when a newer, more exciting one comes along. So the CIESE staff, along with Stevens' president, who is a strong supporter of the project, will continue to do whatever it takes to make sure that computers become an established part of what mathematics teachers do in their classrooms. And through Logo software projects, teachers can fill in the gaps where there is a lack of other adequate software and make Logo experiences available to students.

A Logo Graphing Project

When I teach a graphing unit in Algebra I, one of the concepts many of my students don't seem to fully understand is that when they draw a graph of a function or relation by connecting a set of points with a smooth curve or line, every point on that curve or line has meaning. In other words, as they slide their pencils from one point to the another, what they are really doing is plotting an infinite set of points, each one associated with a unique pair of numbers that makes true the function they are drawing. For some reason they see the line as just a connector of meaningful points. I find myself constantly reminding them about the significance of the line.

One of the reasons I think children have a problem with this is that when they are introduced to line graphs in elementary school, lines are used as connectors to indicate trends. For example, the graph below, taken from a popular piece of software, shows an instance where the line has meaning only at certain points; the trend represented by the line is meaningless:



One of the goals of creating a Logo graphing experience is to focus the students' attention on the significance of lines. Logo graphing experiences also give students an opportunity to explore ideas from analytical geometry in a Logo-like way.

Setting a Context for the Logo Experience

In a recent article, Thomas Seddon (1990) states that he believes the most important aspect of classroom teaching is the context in which we present ideas. "Unless the children see a purpose in what they are doing, the activity will not make much of a difference," (p. 418) he says. Now this doesn't necessarily mean that the context has to be limited to the real world, much of which students may find irrelevant and boring. The context can appeal to students' imagination, sense of humor, and fantasy. It is also a good idea to use metaphors to which students can relate. A scenario (described below) that I have found useful in teaching graphing to students involves the metaphor of family homes.

Getting Started

Before I begin, I load the Logo graphing microworld in my computer, and I use a projection device to enlarge the screen image on a wall or screen. A summary of Logo procedures used in this lesson and included on the CLIME disk appears in the box below.

GRID	clears the screen and draws an x and y Cartesian coordinate system.
PLOT x y	highlights the point (x, y) on the screen.
LARGE	works with PLOT and makes PLOT highlight points by placing a circle around the point.
SMALL	turns off LARGE and allows PLOT to plot small dots.
GRAPHS [m b]	takes two inputs:
	(1) a list containing the coefficients (slope and y-intercept) of a linear equation ($y=mx + b$) and
	(2) the increment between x values.
C (color)	sets the color.
HURKLE	plays a coordinate guessing game.

I first tell the students that they will be drawing graphed "pictures" of the homes of families who live at the intersection of a pair of perpendicular lines in a town called Coordinateville.

I type GRID, and Coordinateville is projected onto the wall. The names of the two lines are x and y. Like real people, members of these imaginary families have first and last names. In Coordinateville, the names are numbers. For example, if you wish to find out where (4, 5) lives, you type

LARGE PLOT 4 5

The point (4, 5) is now highlighted on the screen. I do a few more examples and then invite the students to join me in the following activities.

Activity: Hurkle

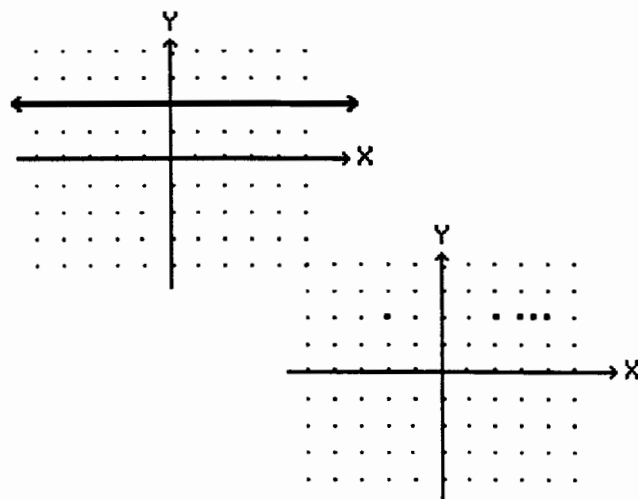
After the students have a feel for plotting points using PLOT, they can then play HURKLE, a game that asks the player to find the imaginary Hurkle who is hiding somewhere in Coordinateville. This game helps students learn how to locate number pairs on the coordinate system.

Activity: Family Plots

The last number of each pair is the "family" name. Hence, (2, 3) and (3, 3) belong to the "3 family." I ask the students to draw several members of the "3 family." If students wish to identify different family members by color, they can type C followed by a number from 1 to 5 to change the color of each point.

After the students plot a few points, I ask them to make a conjecture about the nature of the "3 family." They probably will notice that all the points fall on a horizontal line. "What would the complete family look like?" I ask.

A lively discussion often ensues, including a discussion related to the density of the number line!



Next I ask students to predict what the "4 family" or the "-2 family" looks like. To actually draw the family using PLOT is very tedious; in fact, it is impossible since it is infinite! It is useful, however, to take advantage of the power of Logo to draw a more complete family portrait.

Next I give students the following assignment: Write a procedure that will draw any family living in Coordinateville.

One solution might involve a "brute force" method like this one:

```
to family 4
  plot -8 4
  plot -7.5
  plot -7 4
  plot -6.5
  :
  etc.
end
```

A more powerful solution would look like this:

```
To family :number
  plot.points -8 :number
end

to plot.points :x :number
  if :x > 8 [stop]
  plot sentence :x :number
  family :x + .2 :number
end
```

For example, the "5 family" will start plotting points at (-8, 5) and go to (8, 5), plotting a point at intervals of 0.2 along the x-axis. For a denser line, experiment with increments less than 0.2.

Function Families

Sets of ordered pairs can belong to other functional families as well. Here are some ordered pairs that belong to a functional family: (4, 5), (2, 3), (7, 8). Can you figure out their functional relationship? Notice that the last name is always one more than the first name. In algebraic language this relationship can be written as $y = x + 1$ where y represents the last name and x represents the first name. So the equation means that the last name is equal to one more than the first name.

An appropriate question for the class might be: Is (-1, 0) a member of this functional family? (The answer is yes.)

Activity: Guess My Rule

There are many varieties of this game, some fancier than others. Basically the game works as follows. The teacher (or a student) thinks of a rule (a linear function), and sketches an empty x,y table. Students are then asked to suggest a number for x . Let's say that number is 2. The teacher (or student) who made up the rule writes down the corresponding y number. The students continue offering numbers for x , and the ruler (person who made up the rule) writes down the corresponding y values. If a student thinks he or she knows what the rule is, the ruler challenges him/her with three x values for which the student must come up with the corresponding y values. If the student can come up with the correct y value three times in a row, he or she is given credit for knowing the rule and can share it with the group.

Here's an example:

x	y
2	4
3	7
4	10

If a student thinks he or she knows the rule at this point, the ruler can challenge the student with these values:

5	?
10	?
-3	?

The chosen rule was 3 times x minus 2 ($y = 3x - 2$).

A useful side exploration here is to find a systematic way of determining the rule for linear functions.

Activity: Mini Green Globes

GRAPH is a procedure available for drawing relational families as densely as is possible on the computer screen. The student needs to know how to use the procedure in order to do the Mini Green Globes activity. For those of you not familiar with Dugdale and Kibbey's (1983-85) *Green Globes* program, here is a quick description. In *Green Globes*, the object is to plot functions on coordinate axes so that they pass through as many green globes (large circles on the coordinate axis) as possible. The globes are identified with ordered pairs. In this activity two random points (or globes) are drawn. The object is to use GRAPH to plot as complete a picture as desired of a functional family that includes and passes through the two globes.

GRAPH, which draws the graph of $y = mx + b$, where m is the slope and b is the y-intercept, needs two inputs. The first is a list (in brackets) which contains m and b . The second input is the difference between x values.

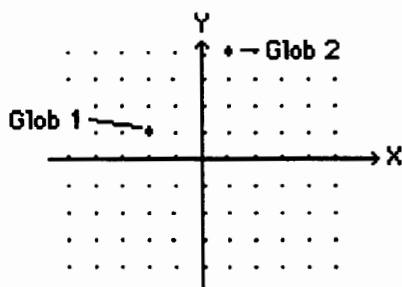
For example, GRAPH [1 1] .5 would draw the picture of $y = 1x + 1$ (or $y=x+1$) starting at $x = -8$ and incrementing by 0.5 until the graph goes off Coordinateville or until $x = 8$, whichever comes up first.

After students practiced using GRAPH, they can tackle the following activity. Type

FAMILY.PLOT

The computer will respond by plotting two points and printing

Draw a family portrait passing through (-2, 2) (1, 4) using GRAPH.



Students can figure out the rule from the two number pairs (GRAPH draws only linear families), or they can determine the slope by counting squares and estimating where the line will hit the y-axis.

Activity: Lines and Points

If you type LINE.POINT the computer will draw a line and a point outside the line and ask the student to plot a family that includes the point and has a slope parallel to the original line.

Please note that I focus on using family language rather than the more precise language that textbooks use. By keeping the context closer to the everyday language of the students, I believe they will better understand the nature of algebraic graphs.

An Invitation to Join the Party

This project is still under construction. I invite you, the reader, to join us in building this unit. What would you bring to the party? How can we build on these ideas? What changes would you make? It's not too late to even tear it down and start all over! The important thing is that we collaborate and have some fun!

References

- Clements, D., & Battista, M. T. (1991). *Logo geometry*. Morristown, NJ: Silver Burdett & Ginn.
- Dugdale, S., & Kibbey, D. (1983-85). *Green globs and graphing equations*. Pleasantville, NY: Sunburst Communications, Inc.
- Picciotto, H. (1990). *Logo math tools and games*. Portland, ME: Terrapin Software, Inc.
- Seddon, T. E. (1990, September) Let's abolish general math. *Mathematics Teacher*.

Please write to Ihor Charischak at 10 Bogert Avenue, White Plains, NY 10606, if you would like information on how to obtain disks with the graphing application described in this column. He can also be reached through BITNET at

TLP_ICHARISC@STEVENS.

A. J. (Sandy) Dawson is director of the Professional Development Program, and an associate professor in the Faculty of Education at Simon Fraser University in Vancouver, BC, Canada V5A 1S6. He can be reached through BITNET as

USERDAWS@SFU.BITNET

Logo Connections

Logo and Telecommunications

by Glen L. Bull and Gina L. Bull

This year we have been discussing practical multimedia suggestions for Logo. There is an international telecommunications network known as *Internet*, which links all major universities and many public schools in the United States, as well as sites in many foreign countries. Some of you may noticed that our Internet/BITNET electronic mail addresses appear beside our names at the end of each column. We often meet interesting Logo users via this network. For example, earlier this year the following message appeared on the screen when we first turned on the computer.

From: EBERG@finujo.bitnet
Subject: Interest about Logo
To: GBull@Virginia.edu

Dear Glen and Gina -

I read your article in Logo Exchange (October 1990) about Logo as a multimedia tool. The idea was very nice and enlarged my thinking concerning to use Logo in learning.

You told in the article of possibilities to import pictures and text from other sources. How you actually do it? Could you give me some guidelines? We use Logowriter and LegoTClogo in IBM compatible environment (AT) and we can use all the painting software which is made for that environment. Will we need some conversion program?

I would be very pleased if I could have some answers to the questions above (through E-mail).

Yours sincerely

Jorma Enkenberg
Research and Development Center for
Information Technology in Education
University of Joensuu - Box 111
80101 Joensuu, Finland
E-mail: Eberg@finujo.bitnet

With some assistance from technical support at Logo Computer Systems, Inc. (LCSI) and from Michael Tempel, we

were able to determine that the graphics format used by the version of *LogoWriter* for IBM-compatible computers does not match any of the standard graphics formats used by paint programs and video digitizers on the IBM. However, it appears that a conversion program to convert standard graphics formats on the IBM to the internal IBM *LogoWriter* format could be developed by LCSI without undue difficulty. With a bit of luck, an "Import Graphics" command might even be added to a future update of the IBM version of *LogoWriter*. We were able to forward this information to Jorma Enkenberg in Finland by return electronic mail.

On another occasion a message from a class in Spain appeared on the screen. Sometimes telecommunications can be the modern-day equivalent of putting a note in the bottle. The message is cast adrift on the network while the user waits to see who may respond.

Subject: Charlottesville-Barcelona Pen Pals
From: EAVERGES@EBRUPC51
To: GBull@Virginia.edu

Hello penfriends,
We are girls and boys of "Princep de Viana II" school in Barcelona in Spain. We are eleven, twelve, thirteen or fourteen years old. We like very much the idea of write you about "Logo". We start to study "Logo" at eleven. We are working Logo in Catalan, our language in Catalonia (Spain). We can't do that in English because we don't know English very well. We are learning English at school, three hours in a week.

We are working "Procs: pictures, games, texts and now we are going to begin Framework (the integrated package with word processor, database, spreadsheet and so on).

We would like very much to write letters with you.

Classes Six, Seven, & Eight
Princep de Viana II School
c/Dubl n s/n 08027 -Barcelona
(Spain)

In this case it was possible to match a sixth-grade class in Charlottesville with the class in Barcelona. The students in Virginia were even able to brush up on their Spanish by sending some of their replies in that language—and soon discovered that the students in Barcelona spoke a specific dialect of Spanish known as “Catalonian.”

In a previous column on Logo robotics, we mentioned the interactions of Becky Fisher’s class at an Albemarle high school with a parallel class using *Logo/Logo* in Moscow. With the help of Sylvia Weir and the Technical Education Research Centers (TERC), a telecommunications link was established between the two schools. Students at one site developed descriptions of construction projects that the other class attempted to replicate.

A Logo User’s Introduction to Telecommunications

Telecommunications can be used to discuss Logo or to exchange actual Logo procedures via the network. To access telecommunications networks, a device called a modem (which connects the computer to the phone line) is needed. A reasonably fast modem that can send data at a rate of 1,200 bits per second (approximately 120 characters per second or 7,200 characters per minute) can be purchased for less than \$100.

In addition to a modem, you will need some telecommunications software to log onto a network. Some telecommunications software, such as the well-known FredMail program, is free or inexpensive. Readers of *Logo Exchange* may be aware that there is also a Logo telecommunications program developed by LCSi called *LogoExpress*. For specific steps in using *LogoExpress*, see the excellent “Logo Ideas” columns by Eadie Adamson in the November 1990 and December/January 1990-1991 issues of *Logo Exchange*.

Once you have a modem and telecommunications software, you will need an account or ID on a computer network. Many local computer users groups have “computer bulletin boards.” A computer bulletin board usually consists of a microcomputer that has been set up to accept phone calls from other computers equipped with modems in that area. Some of the items on *The Teacher’s Lounge*, an electronic conference in the Charlottesville area established by Judi Harris and Sue Anderson, are listed below.

The Teacher’s Lounge

An Electronic Conference

Item	7 (201)	Shareware Requests?
Item	11 (68)	JoAnn’s Mathematical Oddities

Item	16 (58)	Computer Use in the Math classroom
Item	18 (10)	Gender Differences in the Learning of Math
Item	20 (19)	Virginia Museum Tips
Item	24 (22)	Interested in Spanish E-mail Communication?
Item	25 (26)	Bring a Famous Mathematician into Your Class!
Item	29 (26)	Computer Use in the Schools
Item	30 (19)	Retention?
Item	33 (10)	Help with Visual Skills
Item	38 (30)	New BitNet-Connected Educators
Item	43 (35)	A Curricular Fable
Item	47 (14)	Oceanography
Item	52 (21)	Computer Applications in the English Classroom
Item	53 (9)	Student Bloopers
Item	55 (23)	Early Childhood
Item	57 (27)	Wordprocessing in the English Classroom
Item	62 (27)	Earth Science Teachers: Read This
Item	66 (38)	Prejudice is a Handicap
Item	76 (15)	Homogeneous Vs. Heterogeneous
Item	80 (3)	National Teachers of Mathematics Conference
Item	82 (14)	Soviet-American Telecom Conference

As you can see, items on a local bulletin board are likely to cover many areas besides Logo. However, there are two reasons to start with a local computer bulletin board when you first begin telecomputing. The first is that since a local computer bulletin board is likely to be in your phone district, you will not encounter any additional expenses for long-distance phone charges as you experiment and learn. The second reason is that if you encounter any initial difficulty in getting the settings on your telecommunications software set properly, you will be able to find help from nearby users.

If you acquire *LogoExpress*, you will gain access to a bulletin board just for Logo users run by LCSi. There is no

charge for accessing this bulletin board, but it will be necessary to make a long-distance phone call to reach it. However, *LogoExpress* has "macros" (in reality, short Logo procedures) that can quickly transfer all your mail and news from the Logo bulletin board to your local microcomputer, where you can read it at your leisure after the phone call has been terminated. This reduces the long-distance time so that you may only be connected for two or three minutes. (The actual connect time is dependent upon the amount of mail you have and the speed of your modem.)

Commercial telecommunication services, such as CompuServe and Prodigy, are also available. In a commercial service there is a charge for accessing the host computer. Usually a mainframe computer is employed as the host for a commercial telecommunications facility so that many people can simultaneously access the service. CompuServe is the oldest and one of the most extensive of these commercial services, while the rates for Prodigy are lowered somewhat by commercial messages that are interspersed among the other information you receive.

Telecommunications can also be used to transfer Logo procedures from an Apple disk to an IBM disk, or vice versa. A *LogoWriter* program will run on either an Apple or an IBM computer. However, an IBM computer cannot read an Apple II diskette. It is possible to retype all the procedures on the other computer, but this is a tiresome process and often introduces typographical errors.

An alternative to a lot of typing is to use telecommunications to transfer the procedures from one format to the other. This can be done by first uploading the procedures from the Apple II computer to the host computer and then downloading them from the host to the IBM computer (or vice versa). Alternatively, both computers can be connected directly to one another via telecommunications. If the two computers are nearby, the serial ports of the two machines can be directly connected with a cable known as a *null modem* (since it replaces the modems), bypassing phone lines altogether.

The Future of Telecommunications and Logo

In the cases mentioned above (local bulletin boards, the *LogoExpress* bulletin board, and commercial services), a microcomputer connects to a host computer via telecommunications. However, in those instances the host computer is not linked to another computer. In a computer network, a series of computers is interconnected, and it is possible to send a message from computer to computer until it reaches its destination.

At the National Education Computing Conference last year, Senator Gore called for a national public school network.

In Virginia we are establishing a network, Virginia's Public Education Network (PEN), linking public schools to the existing interuniversity network, or Internet. When it is completed, all 2,000 public schools in Virginia will be linked to Internet. This will allow teachers in the public schools to interact with teachers and students around the world, as the examples above illustrate. Conferences such as the *Lego/Logo* conference established by Tom Morgan and Mano Talliaver will allow teachers and students to exchange Logo ideas and procedures.

Connie Stout has just received approval from the Texas Education Agency to link the public schools in Texas to the interuniversity network in a similar fashion. Parallel efforts are also underway in South Carolina, Georgia, Wisconsin, and other states, with encouragement from EDUCOM, the non-profit agency that administers BITNET. In Cleveland, faculty at Case Western have established the Cleveland Free Net and will provide a telecommunications account to any citizen who requests one. (Currently, 20,000 Cleveland citizens have network accounts.) In Montana, the Big Sky Telegraph is an educational network with links to Internet.

Logo Transmissions

Now that we have outlined some suggestions for Logo connections via telecommunications, we must send the column to Sharon Yoder (Yoder@Oregon.bitnet). We will first upload the file, as we do each month, and then transmit it over the network. Possibly this month we should also send a copy to Michael Tempel (michaelt@media-lab.media.mit.edu) to make sure that our description of *LogoExpress* is accurate. Then we also need to send a note to Judi Harris (JHarris@Zeus.unomaha.edu) in Omaha. We need to match a teacher in Charlottesville with a comparable class on Internet in Germany, and Judi has the world's largest list in captivity of teachers on Internet.

Telecommunications has always offered a powerful way to expand Logo horizons. It provides a way to take "electronic field trips." Recently, students in a Virginia science class digitized images of nearby geologic formations, imported them into a Logo file, and transmitted them to a class at another site. Networked Logo classes can collectively address projects that might be beyond the resources of any one class. Logo and telecomputing are both powerful, open-ended instructional tools. When they are combined, classes can go places—electronically.

Glen and Gina Bull, University of Virginia
Charlottesville, VA 22903

Internet/BITNET Addresses

Glen: GBULL@Virginia

Gina: GINA@Virginia

Extra for Experts

edited by Mark Horney

Using Assembly Language in *Terrapin Logo* for the Apple II

by Susan Wells Rollinson

Sometimes only assembly language will do. Perhaps you need a feature that Logo lacks. Or maybe just one little thing is much too slow. Let me tell you how I got started mixing assembly language with Logo.

The highlight of the two-week summer Logo "camp" I teach is putting our designs on T-shirts. Since many of these designs use letters, we needed to be able to flip the screen horizontally before printing with special heat-transfer ink ribbons. Until *Logo PLUS* came along, this capability was not available within Logo. I rummaged through back issues of *Nibble* and found an article by Majka (1987) that described several useful graphics routines, including a horizontal screen flip. Since these routines worked only under Applesoft (Apple's built-in BASIC for the Apple II), we saved the pictures on disk, flipped them in Applesoft, re-saved, and finally printed from Logo. Whew!

You may be thinking, "But assembly language is only for hard-core hackers!" CLEARTEXT! You do not have to be an assembly language wizard to use the Logo Assembler. Naturally, the more knowledge you have in this area the better, but it is easy to adapt published assembly language routines to the Logo environment.

The Terrapin Logo Utilities disk (included with *Logo PLUS* and *Terrapin Logo*) includes an Assembler for integrating assembly language routines into your Logo projects. While not suitable for heavy-duty work, this Assembler is acceptable for incorporating short routines. Documentation is included in the Logo Technical Manual, but allow several readings to understand it.

This article will concentrate on using the Logo Assembler to adapt short, published routines for use with Logo. Consult the references at the end of this article (or another book on 6502/Apple II assembly language) for details about the 6502 microprocessor and assembly language programming.

The Ground Rules

Unfortunately, Logo and Applesoft have conflicting ground rules for creating assembly language routines. Seldom can you use assembly language routines written for Applesoft without modification. Fortunately, such changes are generally quite easy to make.

The greatest source of conflict between Logo and Applesoft is that each reserves different zero-page locations. Since virtually all assembly language routines use the zero-page, you must identify and change these addresses. According to the Logo Technical Manual, the zero-page locations from NARG1 to ANSN4 + 3 and USERPZ through \$FF are available for user assembly language routines. The actual locations for these names are found in the ADDRESSES.LOGO file on the Utilities Disk. For both *Terrapin Logo 3.0* and *Logo PLUS*, these correspond to locations \$A2 through \$C7 (162-199) and \$FC through \$FF (252-255). (For readers unfamiliar with assembly language discussions, numbers preceded by \$ are in hexadecimal, while those without \$ are decimal.)

Another source of conflict between Applesoft and Logo is the space available for user assembly language routines. Applesoft leaves one page (256 bytes) starting at \$300 (768), but clever programmers can place routines virtually anywhere. *Terrapin Logo 3.0* and *Logo PLUS* leave one page starting at \$99A0 (39328). This is the only location available, so your assembly language routines must be short! Fortunately, this is typically the case. An assembly language routine that is not completely relocatable must be reassembled at the new origin. Again, most short routines are relocatable. If in doubt, check the comments and accompanying discussion (if any) for the published routine. If you are advanced enough to be writing your own routines, you already know about relocatable code.

Introducing the Logo Assembler

The Logo Assembler uses four files from the Utilities Disk:

ASSEMBLER.LOGO	Logo assembler. Creates the binary machine-language file.
OPCODES.LOGO	Listing of 6502 instruction set. Associates mnemonic with a number that is the machine-language code for the instruction.
AMODES.LOGO	Helps the assembler figure out the addressing mode used in a statement.

ADDRESSES.LOGO

Gives names to specific locations in the Apple II memory.

ASSEMBLER.LOGO is the file that contains the Assembler. It uses three additional "data" files. **ADDRESSES.LOGO** associates variables with specific locations in the Apple's memory. **AMODE.LOGO** contains information on the various addressing modes used in 6502 assembly language. **OPCODES.LOGO** describes the 6502 microprocessor instruction set in Logo terms.

You may find it convenient to create a data disk (or directory) for Assembler work. Copying the four files mentioned above to a newly formatted disk leaves plenty of room for Assembler procedure files and the assembled binary files.

I find it interesting that an assembler can be written entirely in Logo, and in three printed pages at that! I won't discuss the inner workings of the Assembler. (The dreaded "exercise left for the student" syndrome—however, an annotated hard-copy listing is available from the author.) Your assembly language programs are entered as a procedure in the Logo editor. The format is similar (mostly) to that used by a conventional assembler.

When referring to a named location from the **ADDRESSES.LOGO** file, you should use the name rather than the numerical address. The names will remain the same in new versions of *Terrapin Logo*, but the addresses may not. (Indeed, almost half of all the addresses are different in *Terrapin Logo 3.0* and *Logo PLUS*.) Thus, using names means that changing to a new version of Logo requires reassembly of the routines, not rewriting them!

Modifying OPCODES.LOGO and ASSEMBLER.LOGO

Before using the Assembler, we need to modify the **OPCODES.LOGO** file. This file contains descriptions of most of the instructions in the 6502 chip, but 15 of the less frequently used opcodes are missing. We will add three new opcodes; if you need others, consult the list below.

```
BVS      [REL 112]
CLD      [IMP 216]
CLI      [IMP 88]
CLV      [IMP 184]
EOR      [IMM 73 ZP 69 ZPX 85 ABS 77
          ABX 93 ABY 89 INDX 65 INDY
          81]
LSR      [ACC 74 ZP 70 ZPX 86 ABS 78
          ABX 94]
```

```
PHP      [IMP 8]
PLP      [IMP 40]
ROL      [ACC 42 ZP 38 ZPX 54 ABS 46
          ABX 62]
ROR      [ACC 106 ZP 102 ZPX 118 ABS
          110 ABX 126]
RTI      [IMP 64]
SED      [IMP 248]
SEI      [IMP 120]
TSX      [IMP 186]
TXS      [IMP 154]
```

To modify **OPCODES.LOGO**, make sure you have a clean workspace. The best way to do this is to type

```
GOODBYE
```

at the Logo prompt.

If you are using *Logo PLUS*, you may have to set the ProDOS prefix to the directory containing your assembler files. For example, on my system I need to use

```
SETPREFIX "/HARD1/LOGO/UTILITIES/
ASSEMBLER
```

Logo PLUS users can also use most DOS commands directly. That is, DOS [LOCK **OPCODES.LOGO**] becomes LOCK "**OPCODES.LOGO**"

Unlock **OPCODES.LOGO** by typing

```
DOS [UNLOCK OPCODES.LOGO]
```

Type the following lines:

```
MAKE "ROR [ACC 106 ZP 102 ZPX 118 ABS
110 ABX 126]
MAKE "ROL [ACC 42 ZP 38 ZPX 54 ABS 46
ABX 62]
MAKE "EOR [IMM 73 ZP 69 ZPX 85 ABS 77
ABX 93 ABY 89 INDX 65 INDY 81]
```

(On each line, do not press <Return> until you have typed the final square bracket.) Now save the changes by typing

```
SAVE "OPCODES
```

and lock the file by typing

```
DOS [LOCK OPCODES.LOGO]
```

We also need to modify the ASSEMBLER.LOGO file. This will allow the same "assembly language" procedure to assemble in both *Terrapin Logo 3.0* and *Logo PLUS*. (I won't go into the details, but the problem is a slight difference in the way the two versions of Logo handle THING?.) At the Logo prompt, type

```
GOODBYE
READ "ASSEMBLER
DOS [UNLOCK ASSEMBLER.LOGO]
```

The listing below shows the needed modifications to the SETUP, SYMS.L, and AMODE1 procedures. Changes are shown in boldface. SYMS.L requires that one line be moved. The *Logo PLUS* version of AMODE1 may have a typo in one line. The changes to SETUP are for convenience only.

```
TO SYMS.L :LINE :TOKEN
  IF LIST? :TOKEN RUN :TOKEN OP 0
  IF INS? :TOKEN OP INSLEN AMODE :
    TOKEN BF :LINE
  IF LABEL? TOKEN ASSIGN :DOT OP
    SYMS.L BF :LINE FIRST BF :LINE
  IF NUMBER? :TOKEN OP 1
  ERROR :TOKEN [NOT AN INSTRUCTION]
END

TO AMODE1 :FPART :LPART
  IF :FPART = "# OP LOOKUP BF :LPART
    :IMMS
  IF :FPART = "!" OP LOOKUP BF :LPART
    :ZPS
  IF :FPART = "(" OP LOOKUP BF :LPART
    :INDIRS
  OP LOOKUP :LPART :INDEXES
END
```

```
TO SETUP
  MAKE "$BASE 16
  MAKE "CH 36
  PR [READING ADDRESSES FILE]
  READ "ADDRESSES
  PR [READING AMODES FILE]
  READ "AMODES
  PR [READING OPCODES FILE]
  READ "OPCODES
END
```

After making the changes, save them with

```
SAVE "ASSEMBLER
DOS [LOCK ASSEMBLER.LOGO]
```

As before, *Logo PLUS* users may need to set a prefix first and can use the shorter form of the DOS commands.

Got that done? Let's assemble!

A Word on Assembler Syntax

The listing below shows a Logo procedure, **NEGATIVE**:

```
TO NEGATIVE
  [MAKE "ANSN3 :ANSN4 - 4]
  LDA # [$ "00]
  STA ! ANSN3
  LDA # [$ "20]
  STA ! [:ANSN3 + 1]
  NEG1: LDY # [$ "00]
  NEG2: LDA ( ANSN3 ) ,Y
  EOR # [$ "FF]
  STA ( ANSN3 ) ,Y
  INY
  BNE NEG2
  INC ! [:ANSN3 + 1]
  LDA ! [:ANSN3 + 1]
  CMP # [$ "40]
  BNE NEG1
  RTS
END
```

Compare this listing to the more conventional assembly language source file shown in the following listing published in *Nibble* (Majka, 1987):

```
* HIRES.TRICKS
* BY JOHN R. MAJKA
* COPYRIGHT (C) 1987
* BY MICROSPARC, INC.
* CONCORD, MA 01742

* MERLIN PRO ASSEMBLER

ORG $6000

* NEGATIVE CREATE *

      LDA  #$00
      STA  $FA
      LDA  #$20
      STA  $FB
NEG1   LDY  #$00
NEG2   LDA  ($FA),Y
      EOR  $FF
      STA  ($FA),Y
```

```

INY
BNE  NEG2
INC  $FB
LDA  $FB
CMP  #$40
BNE  NEG1
RTS

```

To the seasoned assembly language programmer, the Logo listing looks a little, but not completely, strange. The Logo Technical Manual describes the syntax required by the Assembler. Here are some highlights.

- The Assembler works with each line as a list and is extremely picky about the placement of spaces, commas, and the like. Characters such as \$, #, and ! are actually defined as procedures and thus require a separating space. ! indicates a zero-page instruction. # specifies immediate mode. \$ converts the following number from hexadecimal to decimal notation.
- Line labels must be the first item on a line and must be terminated with a colon (:).
- Brackets enclose Logo commands. If you need computations, a new Logo variable, or want to include comments, be sure these are enclosed in brackets.
- A number following a label or alone on a line will get incorporated without change into the assembled code. Characters may be entered one to a line by using quotes ("). These methods are commonly used to enter fixed data, such as constants or strings, into the code.

As is frequently the case, studying an example or two can be enlightening. Again, compare the two listings above. And while you're at it, examine the example MUSIC.SRC.LOGO from the Utilities disk.

A Logo Assembler Tutorial

I will use the NEGATIVE procedure to illustrate the conversion of a published assembly language subroutine to one that peacefully coexists with Logo. The routine reverses the colors on the screen and was chosen for its brevity!

First, let's dispense with the \$6000 origin. Logo Assembler automatically sets the origin to \$99A0.

Now let's examine the zero-page usage. Majka's (1987) published routine, given above, uses \$FA and \$FB. We'll choose addresses near the end of our allowed zero-page region. The first line of NEGATIVE creates a new variable, ANSN3, which is four locations below ANSN4. Every place \$FA occurs in the listing from Majka, we'll use ANSN3. Similarly, \$FB will be replaced with ANSN3 + 1.

Again compare the two listings above. Note that direct Logo commands must be enclosed in square brackets. This also applies to the \$ "operator" (actually, it's a procedure here) and any computations with variables. Also, the Logo editor does not support tabs or other "pretty-printing" features that help make the conventional listing more readable. In particular, notice how the labels in NEGATIVE do not stand out.

We're ready to start work. Clear your workspace by typing

```
GOODBYE
```

Type the NEGATIVE procedure shown above. When finished, save the procedure on disk with

```
SAVE "NEG.SRC
```

It is best to save only your procedures: you do not want the procedures and names from ASSEMBLER, OPCODES, AMODES, and ADDRESSES taking up disk space in every routine you write!

It's time to invoke the Assembler. Type

```
READ "ASSEMBLER
SETUP
```

To assemble your routine, type

```
ASSEMBLE "NEGATIVE
```

If all goes well, an assembled listing will scroll down your screen. The next listing shows this assembly listing for NEGATIVE. An OUTDEV 1 command before assembling captures this information on your printer.

NEGATIVE

```

[MAKE "ANSN3 :ANSN4 - 4]
LDA # [$ "00]                                39328=169
                                           39329=0
STA ! ANSN3                                39330=133
                                           39331=192
LDA # [$ "20]                                39332=169
                                           39333=32
STA ! [:ANSN3 + 1]                          39334=133
                                           39335=193
NEG1: LDY # [$ "00]                          39336=160
                                           39337=0
NEG2: LDA ( ANSN3 ) ,Y                      39338=177
                                           39339=192
EOR # [$ "FF]                               39340=73
                                           39341=255
STA ( ANSN3 ) ,Y                           39342=145
                                           39343=192
INY                                         39344=200
BNE NEG2                                   39345=208
                                           39346=247
INC ! [:ANSN3 + 1]                          39347=230
                                           39348=193
LDA ! [:ANSN3 + 1]                          39349=165
                                           39350=193
CMP # [$ "40]                              39351=201
                                           39352=64
BNE NEG1                                   39353=208
                                           39354=237
RTS                                         39355=96

```

If you get an error message, try your best to figure out what's wrong. Errors are probably the worst aspect of assembling in Logo. The error messages give very little help—when they're not downright misleading. If you need to make a change, start from the very beginning, that is

```

GOODBYE
READ "NEG.SRC

```

Make corrections. Then repeat the assembly:

```

SAVE "NEG.SRC
READ "ASSEMBLER
SETUP
ASSEMBLE "NEGATIVE

```

When you have a successful assembly, try out your work. Load a picture into Logo. Hide the turtle. (Failure to hide the turtle can produce some interesting, but usually unwanted, results!) Use the command

```
CALL :ORG 0
```

or

```
CALL 39328 0
```

to test the routine. If it doesn't work, clear the workspace and start again!

When your routine works properly, save the assembled version to disk with the command

```
DOS [BSAVE NEGATIVE.BIN, A$99A0, L$1C]
```

Once again, *Logo PLUS* users must be concerned about the prefix but can also use BSAVE directly. The length parameter for the BSAVE can be calculated in two ways. The easiest is to take the difference between the :ORG and :END variables, PR :END - :ORG. Alternatively, you can look at the assembled listing: length = last byte - first byte + 1. Convert to hexadecimal before using BSAVE.

Try it yourself with one of the other routines from Majka (1987). They include two goodies built into *Logo PLUS* (but not *Terrapin Logo 3.0*): horizontal and vertical flips, and scrolling in four directions—up, down, right, and left. (Contact me for more information about Logo source code and binary file availability.)

I gather from the space-saving techniques used in the Assembler (erases the SETUP procedure, limited OPCODES, no comments) that memory might be tight in a 64K Apple II. The Logo Technical Manual gives some suggestions, including eliminating from the OPCODES.LOGO and ADDRESSES.LOGO files those instructions and addresses you do not need. I have never had any problems on a 128K Apple IIe.

Using Assembly Language Routines in Logo Programs

The listing that follows, NEG.DEMO.LOGO, demonstrates how to use NEGATIVE in your Logo programs:

```

TO NEGATIVE
  MAKE "ORG 39328
  DOS [BLOAD NEGATIVE.BIN]
  .CALL :ORG 0
END

TO MULTIPLE.EFFECT
  ; ASSUMES YOU HAVE TWO BINARY FILES
  ON DISK:

```

```
; NEGATIVE.BIN AND HORZ.FLIP.BIN
MAKE "ORG1 39328
DOS [BLOAD NEGATIVE.BIN]
MAKE "ORG2 39328 + 28
DOS [BLOAD HORZ.FLIP.BIN, A$99BC]
.CALL :ORG1 0
.CALL :ORG2 0
```

END

The procedures assume that a picture has already been drawn. The .CALL routine is a Logo primitive that requires two inputs. The first input is the address of the beginning of the routine. The second is usually 0 but may contain a number to pass to your routine. See the Logo Technical Manual for more information.

You can put more than one assembled routine in memory at a time, provided the total length does not exceed 256 bytes. You must calculate the BLOAD and .CALL addresses yourself. An example of this is shown in the MULTIPLE.EFFECT procedure, which creates a negative image and then flips it horizontally.

A final word of caution: be sure to hide the turtle before using these routines. If the turtle is showing, the turtle image will be inverted, flipped, or scrolled, but the turtle position remains unchanged. The result is seldom desirable!

Where Does This Fit in Your Classroom?

My Logo classroom experience is with gifted students in the fourth through seventh grades. In that situation, I assembled the routines and showed the students how to use them in their programs.

High school students could use the Assembler themselves, especially if they are interested in assembly language programming or other "technical" aspects of computing.

Bibliography

- Hyde, Randy. (1981). *Using 6502 assembly language*. Chatsworth, CA: Datamost, Inc.
- Lancaster, Don. (1984). *Assembly cookbook for the Apple III/IIe*. Indianapolis: Howard W. Sams & Co., Inc.
- Majka, John. (1987). Hi-res tricks. *Nibble*, 8 (5), 41. (Also available in *Nibble Express VIII* and on "Sight & Sound," *Nibble Disk #A10*.)
- Wagner, Roger. (1982). *Assembly lines: The book*. El Cajon, CA: Roger Wagner Publishing, Inc.

Susan Wells Rollinson
849 Lou Ave., Clifton Forge, VA 24422

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)

Logo: Search and Research

GeoTools: Logo-Based Geometric Construction

by Douglas H. Clements

Last month we discussed the Bulgarian Plane Geometry System, a Logo-based geometry construction program for secondary students. That was the first of several columns on Logo and geometry. This month we'll take a look at a related program for elementary and middle school students.

Logo Geometry and GeoTools

Mike Battista and I have finally completed a project we have been working on for four years with support from the National Science Foundation. *Logo Geometry* (Clements & Battista, 1991) is a K-6 geometry curriculum based on Logo and our geometric extensions of Logo. The goal of *Logo Geometry* is to help students progress through the initial levels of geometric thinking as described by van Hiele (see my column in *LX* [1987, vol. 5, No. 6, pp. 20-21] for more information). In much of their initial work, students learn to think about geometric figures as *paths*. They use Logo to raise this thinking to higher levels. For example, to write a parallelogram procedure, they analyze the visual aspects of the rectangle and reflect on how its component parts are put together, an activity that encourages them to move from Level 1 to Level 2 thinking. If then asked to design a "generalized" parallelogram procedure with inputs, students must construct a form of definition for parallelograms, one that the computer understands. This orients students to Level 3 thinking.

This path perspective turns out to be quite valuable, and we shall report our research on it in future columns. However, students also need to learn to make a transition to a more formal, Euclidean perspective (as opposed to Logo's path perspective, one closer to differential and analytic geometry). That is, when students take a path perspective, they see themselves "in" the plane. Thinking of themselves as the turtle, they need only consider their immediate position and heading. But students also need to take the formal perspective of standing "above" the plane, thinking about points and point sets, such as line segments. *Logo Geometry* accomplishes this by providing students with special pseudo-primitives—geometric commands such as those in the Plane Geometry System—written in Logo. Called GeoTools, these pseudo-primitives allow students to work with basic geometric objects, such as points and lines, and to measure lengths and angles. With GeoTools, students can consider geometric objects in terms of their vertices and measurements so that they can explore the metric properties of shapes more fully than in the usual Logo environment.

The point-labeling facility of GeoTools also encourages students to consider "variable" points. That is, instead of having to consider a specific triangle, students can specify an arbitrary triangle with commands that place points on the screen randomly. They can think about a shape formed with such points as representing a *class* of shapes rather than a specific instance. Investigating the properties of such a triangle encourages students to make conclusions about all triangles rather than about a specific instance. The resulting conjectures about triangles and attempts to verify empirical discoveries with informal reasoning lead students to even higher levels of geometric thinking.

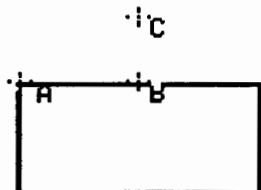
An important benefit of GeoTools is that it combines Logo's turtle with new geometric construction commands. (The *Geometric Supposer*, a well-respected program by Sunburst, has no Logo commands; even in the Logo-based Plane Geometry System, students cannot use the construction commands and the turtle at the same time.) This helps students *integrate* the two perspectives. For example, to make a roof for a building that is not an equilateral triangle, one group of sixth graders first defined a rectangle procedure:

```
TO RECTANGLE
  REPEAT 2 [FORWARD 50 RIGHT 90 FORWARD
    90 RIGHT 90]
END
```

Then they used Logo and GeoTools commands to make some measurements:

```
RECTANGLE
FORWARD 50
LP "A ; "Label Point"; takes a
    letter as an input
RIGHT 90 ; and labels the turtle's
    current position
FORWARD 45 ; with that letter
LP "B
PU
LEFT 90
FORWARD 30
LP "C
PRINT 90 - MEASURE.ANGLE "B "A "C
PRINT LENGTH "A "C
PRINT 90 - MEASURE.ANGLE "A "C "B
```

Note that these students added the "90 - " only after some thought! The following figure was drawn:



and the following numbers were printed:

56.31
54.08
33.69

Using these measures, the students constructed this procedure:

```
TO BUILDING
RECTANGLE
FORWARD 50
RIGHT 56.31
FORWARD 54.08
RIGHT 33.69 * 2
FORWARD 54.08
RIGHT 56.31
END
```

This produced the following figure:

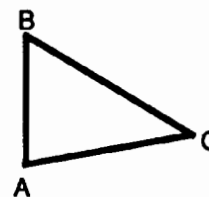
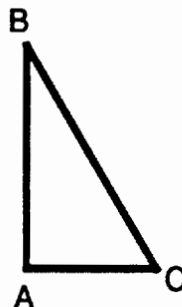


Within the GeoTools environment, students explore points, lines, line segments, and rays; intersecting, parallel, and perpendicular lines; bisectors and perpendicular bisectors; lengths of sides and measures of turns; and finally, general polygons and circles. Examples of specific activities from the *Logo Geometry* curriculum follow.

Activities with GeoTools

In one activity, students learn about classifying triangles by sides (scalene, isosceles, and equilateral) and angles (acute, right, and obtuse), and about certain properties of right and isosceles triangles. To begin, students measure a group of predefined triangles with the GeoTools commands

MEASURE.ANGLE and LENGTH. They record their findings on an activity sheet. Below is a section of the activity sheet; the software provides identical triangles.



T1 length AB 60
length BC 92
length CA
angle A 80°
angle B
angle C

T2 length AB 83
length BC
length CA
angle A 90°
angle B
angle C

At the end of the period, the teacher asks the students to think about ways to group triangles they've measured. The next day, the students cut apart the triangles on one activity sheet and devise ways of classifying them. Students then provide a rationale for their sorting schemes. (Any justified system is acceptable at this time.) From the resulting discussion, the teacher helps students elaborate on the concept of classifying triangles by sides and by angle measure, providing appropriate vocabulary. Finally, the students classify randomly generated triangles produced by a predefined GeoTools procedure. A group of fifth-grade students raised an interesting question the first year of field testing: Will the random triangle procedure generate more acute or obtuse triangles? Time was short, but by the next day a new program, TRI.STATS, was written. This program generated random triangles and kept a tally of their classification by sides and angles. Students predicted which type of triangle the random triangle procedure would generate more frequently and gave justifications for their predictions. The class then ran the program (for 24 hours!) and checked their predictions. (What would your prediction be? Why?) We subsequently built this serendipitous exploration into the curriculum permanently.

In another activity, students explore perpendicular bisectors. They write a Logo procedure to construct the perpendicular bisector of a line segment. For example, a group of sixth graders wrote this procedure:

```

TO PERPB :PT1 :PT2
GOTO :PT1
TURNTO :PT2
PU
FORWARD (LENGTH :PT1 :PT2) / 2
RIGHT 90
PD
FORWARD 200
BACK 400
FORWARD 200
END

```

Teachers then ask students: "If the turtle is anywhere on the perpendicular bisector of a segment, what can be said about its distance from the endpoints of the segment?" Students collect data to answer the question. Similarly, teachers invite them to make discoveries about the perpendicular bisectors of the sides of a triangle and so on.

Initial Research with the GeoTools Environment

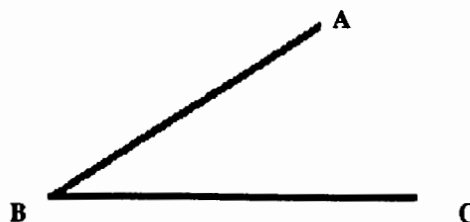
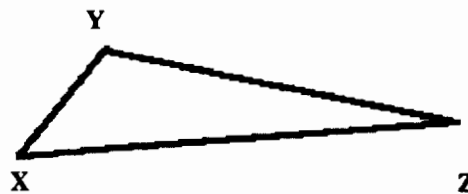
We are just beginning to amass research data from paper-and-pencil assessments, individual interviews, and observations. One brief example of an observation of a GeoTools activity is offered as an illustration.

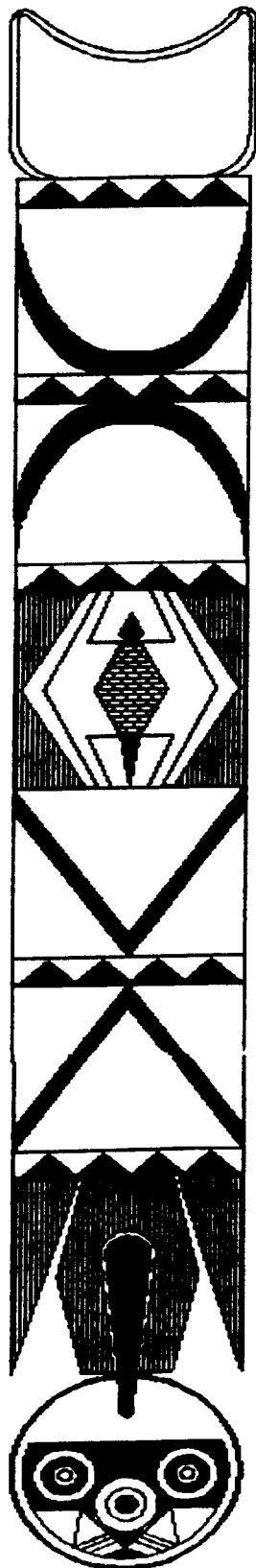
In a fifth-grade classroom, the teacher started the discussion of circles by asking students to identify circular objects in the room. Next, she said that she would be the center of the circle and that the students should stand on the circle, one at a time. As the students started the activity, she asked them how they knew that they were on the circle. The first student did not have much to say. The second student stood directly across from the first at the same distance, but said only that she was directly across. As the third student positioned himself, he said that he had to be the same distance from the teacher as the first two students. The remainder of the students agreed and positioned themselves according to this criterion. The teacher then clarified and defined terms such as center, radius, and diameter. At this point, the students went to the computers in pairs. Students were first asked to make the computer draw a circle that had a given segment as its diameter. Several pairs of students had difficulty at this point. Even though the ideas of center, radius, and diameter had been discussed in class, the students did not understand them well enough to use them on the computer. Thus, the students had to reformulate these concepts that they had formed in the physical environment in the more abstract environment of

GeoTools. Doing so enhanced their overall conceptualizations.

We have completely coded and analyzed only the GeoTools unit test at this point. The following is a table of fifth- and sixth-grade students' percentages correct for several items, for Year 1 (70 students) and Year 2 (205 students) of our field testing.

Items	Year	
	1	2
Angle and triangle classification (11 items)	84	89
Identification of parts of a circle (3 items)	87	95
Perpendicular bisector: "Point X is on the perpendicular bisector of line segment AB, but not on AB. What kind of triangle is ABX? Justify your answer."	88	92
"The circle on the right is circumscribed around the triangle. Explain how you would circumscribe a circle around triangle XYZ." (Only fully adequate answers were given credit; e.g., "I would find the intersection of the perpendicular bisectors of two sides; that's the center of the circle.")	—	58
"Explain how you would construct an angle that is congruent to angle ABC."	—	75





Bwa plank mask
Village of Boni,
Burkina Faso.
By Michael Powell
(see page 7 for details)

These results are encouraging, given the difficulty of these items compared to geometry questions on most current tests. Results from interviews and case studies, however, will likely be much more enlightening.

Final Words

Like the Plane Geometry System, GeoTools offers the combined power of geometric construction programs and Logo. It connects easily to traditional geometric perspectives and remains open for exploration and problem posing. We have found that using new primitives *with* the turtle helps students integrate and synthesize the two perspectives.

Next month we'll summarize research from several studies, including those gleaned from work on other geometric construction programs, such as the *Geometric Supposer*. These have implications for work with both Logo-based geometric construction programs and "regular" Logo.

For more information on *Logo Geometry*, contact

Rachel Auslander, Instructional Technology
Silver Burdett & Ginn
250 James St., Morristown, NJ 07960
(201) 285-8144

Note: Anyone else doing similar work is asked to send information to me so that I can feature it in future columns. (Of course, this also applies to any research work with Logo!)

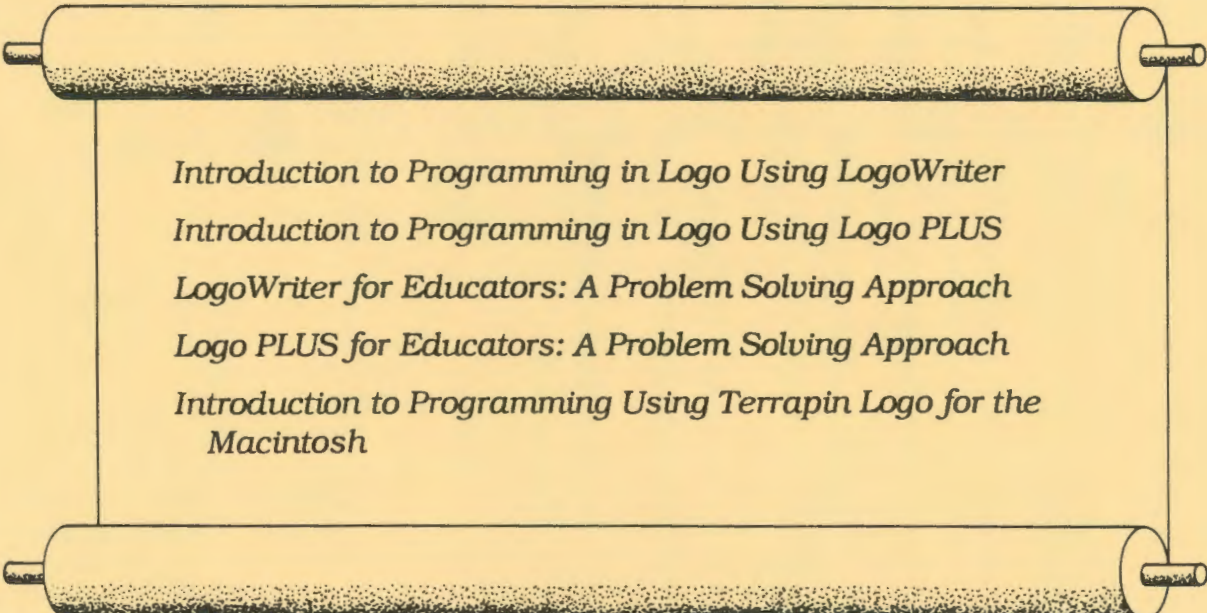
Reference

Clements, D. H., & Batista, M. T. (1991). *Logo geometry*. Morristown, NJ: Silver Burdett & Ginn.

Douglas H. Clements is an associate professor in the Department of Learning and Instruction, State University of New York at Buffalo. He has studied the use of Logo environments in developing children's creative, mathematics, metacognitive, problem-solving, and social abilities. In addition to *Logo Geometry*, his other recent work includes *Computers in Elementary Mathematics Education*, emphasizing Logo, 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

Look at our Logo list!



Introduction to Programming in Logo Using LogoWriter
Introduction to Programming in Logo Using Logo PLUS
LogoWriter for Educators: A Problem Solving Approach
Logo PLUS for Educators: A Problem Solving Approach
Introduction to Programming Using Terrapin Logo for the Macintosh

Logo users at all levels benefit from these ISTE selections.

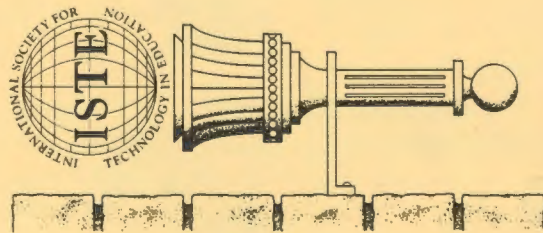
The *Introduction to Programming* books, written by Sharon Yoder, provide beginners with a Logo base to build on and experienced users with a reference to rely on. Both are excellent resources for teacher training or introductory computer science classes.

LogoWriter (Logo PLUS) for Educators: A Problem Solving Approach takes Logo learning to new depths. The focus is entirely on learning and practicing general problem solving skills while using Logo. Great for beginning programming experience. Appendices include keystroke summaries, turtle shape pictures, and a quick reference card. Written by Dave Moursund and Sharon Yoder.

New from ISTE, *Introduction to Programming Using Terrapin Logo for the Macintosh* combines focused lessons with open-ended explorations to introduce Logo syntax and grammar along with programming style and structure. Written by Sharon Yoder.

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

The *International Society for Technology in Education* touches all corners of the world. As the largest international non-profit professional organization serving computer using educators, we are dedicated to the improvement of education through the use and integration of technology.



Drawing from the resources of committed professionals worldwide, ISTE provides information that is always up-to-date, compelling, and relevant to your educational responsibilities. Periodicals, books and courseware, *Special Interest Groups*, *Independent Study* courses, professional committees, and the Private Sector Council all strive to help enhance the quality of information you receive.

It's a big world, but with the joint efforts of educators like yourself, ISTE brings it closer. Be a part of the international sharing of educational ideas and technology. Join ISTE.

**Join today, and discover how ISTE puts
you in touch with the world.**

ISTE

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

**Logo Exchange
ISTE**

**1787 Agate Street
Eugene, OR 97403-1923**