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

LOGO EXCHANGE

Spring 1993

Volume 11 Number 3

In this issue:

My Friendly Kangaroo—A Study in Fascinating Fibonaccis

Also—

Turtle Morse Code

The Greater Scheme of Things

International Society for Technology in Education

LX Volume 11 Number 3

Editorial Publisher
International Society for Technology in Education

Editor-in-Chief
Sharon Yoder

Assistant Editor
Ron Renchler

Founding Editor
Tom Lough

International Editor
Dennis Harper

Contributing Editors
Eadie Adamson
Gina Bull
Glen Bull
Doug Clements
Sandy Dawson
Dorothy Fitch
Mark Horney
Robert Macdonald

Production
Donella Ingham

Cover Design
Donella Ingham

SIG Coordinator
Dave Moursund

Logo Exchange is published quarterly by the International Society for Technology in Education (ISTE), 1787 Agate Street, Eugene, OR 97403-1923, USA; 800/336-5191. This publication was produced using Aldus *PageMaker*®

Individual ISTE Members may join SIG/Logo for $16.00. Dues include a subscription to *Logo Exchange*. Add $10 for mailing outside the USA. Send membership dues to ISTE. Add $2.50 for processing if payment does not accompany your dues. VISA, Mastercard, and Discover accepted.

Advertising space in *Logo Exchange* is limited. Please contact ISTE's advertising coordinator for space availability and details.

Logo Exchange solicits articles on all topics of interest to Logo-using educators. See the back cover of this issue for submission guidelines. Opinions expressed in this publication are those of the authors and do not necessarily represent or reflect the official policy of ISTE.

POSTMASTER: Send address changes to *Logo Exchange*, ISTE, 1787 Agate St., Eugene, OR 97403-1923. ISTE is a nonprofit organization with its main offices housed at the University of Oregon. ISSN# 0888-6790

LOGO EXCHANGE

Volume 11 Number 3 Journal of the ISTE Special Interest Group for Logo-Using Educators Spring 1993

Contents

Logo Programing Style... again

by Sharon Yoder

It seems like I'm always writing about programming and whether it should be formally taught and at what level. There are days when I am sure that programming is an essential part of being a competent computer user, and there are times when I'm not so sure. A recent encounter with some Logo microworlds caused me to rethink this issue again.

These microworlds were brilliantly conceived and probed deeply into concepts of math, science, and problem solving. I was truly excited by this set of materials.

We all know there are lots of wonderful microworlds "out there," and we in the Logo community have talked for many years about collecting them. We are thrilled when we see really good ones. But somehow those collections have never emerged—for one reason or another.

When I saw this delightful group of microworlds, my first reaction was to want to share them with the whole world. But things were not as delightful as they seemed at first glance.

After my initial excitement as I looked over the printed materials, I went to my computer to try the programs. That's when the trouble began. As I was working with the first microworld, Logo began to run increasingly slowly. That seemed strange, especially on my high-speed Macintosh. So I decided to take a look at the code. And there it was—the argument for teaching some computer science to those who want to write programs.

First of all, the programs were filled with groups of instructions like this:

```
make "an readlist
make "nu :an
make "nu first :nu
```

Those of you who have programmed in BASIC will recognize the syndrome. Two-character variable names and lots of assignment statements give it away. But in Logo we could write

```
make "nu first readlist
```

But most Logo programmers would use more meaningful names:

```
make "number first readlist
```

Certainly the code I was looking at was difficult to read and understand because of the naming of variables and the many extra statements.

The particular microworld I examined had an "Instant" type program at its heart. The code for this part of the program looked something like this:

```
to instant
make "key readchar
if :key = "c [cg instant]
if :key = "s [square instant]
if :key = "t [triangle instant]
if :key = "f [forward 10 instant]
if :key = "r [right 10 instant]
if :key = "p [printscreen instant]
if :key = "q [stopall]
instant
end
```

There was the reason my Mac was slowing down. The embedded recursion being generated by the program was enormous! All of those instants at the end of each line were using up more and more of the memory available to Logo. Sooner or later, an "out of space" message would appear.

Those of you who don't think programming is all that important are probably saying, "Well, who cares about variable names and embedded recursion?" After all, the microworlds were wonderful and the students never needed to see the code, so it doesn't matter.

That indeed is the heart of the matter. What does it matter how the code looks if the ideas are profound and powerful? And if you talk to 10 Logo experts, you will likely get 10 different answers. But this *is* an editorial, so I get to air *my* opinion.

I would argue that the teaching of programming style and technique should occur at all levels when you are teaching Logo. No, I would not be discussing embedded recursion and passing parameters with fourth graders, but I certainly would be emphasizing meaningful procedure names. As students get older, I think that the com-

Welcome!

puter science concepts should be introduced gradually. If Logo is continued into high school, students should learn a full range of computer science ideas.

Logo teachers? What about them? Again, I'd like to see ongoing teacher training that would help teachers learn more and more about the programming techniques that make for good, sound Logo code. I have debugged innumerable programs for teachers who could have done their own debugging if they had simply had the opportunity to learn some computer science.

But we don't live in an ideal world. There's not enough time and money for the inservice we would like to provide. Logo is not as popular as it once was—even though it still has a great deal to offer. School districts are unwilling to provide the kind of support that would result in wonderful microworlds with wonderful code supporting them.

So my best advice to teachers of Logo is to learn some computer science. Increase your *own* skills as a programmer. And model those skills for your students. The result will be easier-to-debug code and much more "publishable" products for yourself and your students.

Sharon Yoder
170 Education, DLIL
University of ORegon
Eugene, OR 97403

With this issue, we welcome a new columnist, Robert Macdonald. If that name sounds familiar, it's because Robert has published in the *Logo Exchange* in the past. Robert is a recently retired classroom teacher who says he is thoroughly enjoying his retirement.

I first met Robert as a student in the ISTE Logo Independent Study Course. Robert's lessons were always full of wonderful ideas. It was clear that his classroom always bubbled with new ideas and that his students benefited tremendously by their time with him.

After Robert completed the course, we continued to hear from him. He took ideas from LX and adapted them to his classroom and then wrote to us to tell us about his experiences.

I briefly lost track of Robert when he retired and moved but was delighted when we again made contact. Retirement hasn't seemed to dampen Robert's enthusiasm for Logo in the classroom. I continued to hear from him about his delightful ideas. So I suggested a "retirement column." That is, Robert would contribute as long as he had new ideas and when it didn't interfere with the serious business of enjoying life. I am now inundated in articles from Robert, so you will be hearing from him for a number of issues to come.

So check the pages of LX starting this month for his new column called "Musings."

Don't Miss Global Connections!

ISTE is proud to present the Second International Symposium on Telecommunications in Education, November 10-13, 1993, in Dallas, Texas.

This unique conference for educators, policy makers, and researchers is designed to let you see demonstrations of some of the rapidly-advancing developments in telecommunications. You'll also have the opportunity to debate key policy issues and exchange ideas and information with your peers.

Don't miss out on the exchange!

If you're interested in presenting at the conference, call ISTE today to receive a call for participation! The deadline for submissions is March 31, 1993.

503/346-4414

The Greater Scheme of Things

by Tom Lough

I received a package in the mail recently that took me back about six years in time to two incidents I had all but forgotten.

It was during the early, exciting, growing years of the *National Logo Exchange* newsletter. Robs Muir was our programming-problem columnist and wrote a wonderful series of Logo problems under the headline of "NLXual Challenges." Robs had sent me a note strongly recommending that I purchase a copy of a book called *Structure and Interpretation of Computer Programs*, written by Harold Abelson and Gerald Jay Sussman, with Julie Sussman (MIT Press, 1985).

Heeding Robs' suggestion, I obtained the book and learned that its title was the required entry-level subject in computer science at MIT. The teaching in the book made use of Scheme, a powerful dialect of the Lisp programming language. It was fascinating. Since I knew a little Logo, I was able to follow several of the programming ideas. Unfortunately, I did not have time for more than a cursory reading of the book, nor did I have access to a Lisp machine. Reluctantly, I put it aside.

The following summer at the Logo '86 conference at MIT, I had the good fortune to participate in a Scheme workshop presented by Harold Abelson. This revived my interest in this fascinating language. But, again, I had no opportunity to follow through.

The package I opened contained disks to set up a Scheme programming environment on my own Macintosh! But it also gave me even more—a vision of yet another alternative for high school computer science, with close ties to Logo.

Although Pascal has been designated as the programming language for the computer science Advanced Placement test for high school students, some educators feel it is too limiting and cumbersome for general computer science applications. They are seeking alternatives for Pascal, and have begun focusing attention on Scheme.

Now that Scheme is available for Macintosh, MS-DOS, and Atari computers, you may want to explore this option for high school application. Paired with an introductory Logo unit, Scheme could provide a powerful offering for computer science courses of study at the secondary level. Or, even better, high school courses using Scheme could build on the several years of Logo experience students might bring with them from the elementary and middle school grades.

For more information on Scheme, write to Tarry Kaufman, Schemers, Inc., 4250 Galt Ocean Mile, Suite 7-U, Ft. Lauderdale, FL 33308.

FD 100!

P.S. If you missed Natasha Chen's "High School Computing: The Inside Story" in the May, 1992, issue of *The Computing Teacher*, read her article to learn about Scheme (and Logo) from the point of view of a student.

Tom Lough
Founding Editor
PO Box 394
Simsbury, CT 06070

Experience the Magic of Technology!

Hosted by the University of Central Florida, College of Education, NECC '93 promises to be the technnology event of the summer. Discover what wizardry educators and administrators in the field of learning technology have been conjuring up in their labs. Magicians and apprentices alike will be charmed by the powerful presentations and exhibits given at this four-day conference.

Don't miss out on the enchantment of the newest cutting-edge products and ideas shown at NECC's "Magic of Technology."

Call 503/346-2834 for more information.

_ ••_ •_• _ •_•• •__ ___ •_• ••• •_•_ ___ _•• •

or Turtle Morse Code

by Dorothy Fitch

Do you wish your turtle could speak another language? It can, if you teach it how. Let's try an easy language—Morse Code!

International Morse Code is made up of dots and dashes. Messages are sent using short and long sounds, or short and long flashes of light. Perhaps you can invent another way to send signals.

There are several ways you could program Morse Code in Logo. You could teach the turtle to draw a dot and a dash. You could print dot and dash symbols as text. You could flash the turtle on the screen by showing it and hiding it. Or you could use long and short sounds. Let's try a few of these ideas.

This chart shows each letter and its International Morse Code equivalent:

A	•_	N	_•
B	_•••	O	___
C	_•_•	P	•__•
D	_••	Q	__•_
E	•	R	•_•
F	••_•	S	•••
G	__•	T	_
H	••••	U	••_
I	••	V	•••_
J	•___	W	•___
K	_•_	X	_••_
L	•_••	Y	_•__
M	__	Z	__••

Setting Up the Codes

The first thing we need to do is to enter all the codes into Logo. We could use variables, and store the information like this:

```
MAKE "A ".̲
```

However, this method can get complicated for beginners, and it forces us to work with the codes in a particular way—with symbols. Let's try a different approach. We'll use a reporter named for each letter of the alphabet. The reporter will tell us the pattern of dots and dashes for its letter.

Remember that a Logo reporter is a type of instruction that gives us back information. You may be familiar with the reporter HEADING, which tells us the direction the turtle is pointing, or the reporter RANDOM, which gives us a random number based on the number we give as input.

To write a reporter, we need to use the Logo primitive OUTPUT, which is a command. OUTPUT "kicks" its input out of the procedure. It also stops the procedure. Logo will not run any instructions that are after the instruction line containing the primitive OUTPUT.

Here is how we can write the procedure for the letter A, whose signal is • __:

```
TO A
OUTPUT [DOT DASH]
END
```

If you type A, Logo will tell you something like

```
Result: [DOT DASH]
```

or

```
I don't know what to do with [DOT DASH]
```

If you type PRINT A, Logo will display:

```
DOT DASH
```

Logo will report the list [DOT DASH] every time we refer to A. Can you write the other 25 procedures for the rest of the letters? Each procedure should use OUTPUT to report a list of the appropriate words, according to the chart shown above.

Using the Codes

Now that Logo knows the codes for all the letters, what can we do? For starters, we can print the codes using our new reporters.

You probably know that SOS is a universal help signal. Its Morse Code message is DOT DOT DOT, DASH DASH DASH, DOT DOT DOT.

In Logo, you could show it by typing:

```
(PRINT S O S)
DOT DOT DOT DASH DASH DASH DOT DOT DOT
```

All the right words are there, but the dots and dashes run together. You can't tell where one letter's code ends and the next one begins. A simple solution is to add a space between the letters. Use an empty word like this (be sure to leave a space after each quotation mark):

```
(PRINT S " O " S)
DOT DOT DOT  DASH DASH DASH  DOT DOT DOT
```

You might even want to leave two spaces:

```
(PRINT S " " O " " S)
DOT DOT DOT   DASH DASH DASH   DOT DOT DOT
```

Turtle Dots and Dashes

Now you see how you can use the information that your reporters report. Let's program something that looks and acts more like the real Morse Code!

Your turtle could draw dots and dashes on the graphics screen. You will need a DOT procedure and a DASH procedure that use turtle commands. The turtle should go forward more for the DASH than for the DOT. Here is a DOT procedure; you figure out the DASH procedure! (If your Logo already has a DOT primitive, use a different word, such as DT or DIT.)

```
TO DOT
FORWARD 1
PENUP
FORWARD 3
PENDOWN
END
```

You may also want a SETUP procedure to prepare your screen. It should clear the screen and hide the turtle at the left edge, pointing to the right. See if you can write a SETUP procedure on your own.

Type SETUP, then

```
RUN S
```

The primitive RUN may be new to you. It takes a list as input and runs the instructions in the list just as if you had typed them at top level. Typing RUN S is the same as typing DOT DOT DOT, because S reports the list [DOT DOT DOT].

Do you see three dots? Try doing the O, then another S. Does the turtle draw all the right dots and dashes? But they all run together, just like the words we printed before. It is impossible to distinguish the letters. Let's write a procedure called SPACE that moves the turtle forward a little with its pen up:

```
TO SPACE
PENUP
FORWARD 6
PENDOWN
END
```

Now, you can make your SOS code easier to read by typing:

```
RUN S SPACE RUN O SPACE RUN S
```

Automatic Morse Code

You must be saying to yourself, "This is neat, but isn't there an easier way to do this? Can't we get Logo to do it for us automatically?" We really shouldn't have to type a RUN instruction for each letter in our message!

The good news is—of course there is a way! The bad news is that this is supposed to be a beginner's column, and the solution isn't what one would consider easy. But we can't stop now, so here goes!

This ENCODE procedure takes a word as input and runs the DOT and DASH procedures for each letter:

```
TO ENCODE :WORD
IF EMPTY? :WORD THEN STOP
   (or IF EMPTY? :WORD [STOP])
RUN RUN (LIST FIRST :WORD)
SPACE
ENCODE BUTFIRST :WORD
END
```

Using this procedure, you can show the code for any word:

```
ENCODE "SOS
ENCODE "LOGO
ENCODE "TURTLE
```

Here's how it works:

The first instruction checks to see if the word is empty (has no more letters). If it is, the procedure stops. The next line is tricky, and, yes, there are two RUN commands in a row. Here's why. The first RUN looks for a list of instructions to run. The second RUN gets its input from the first letter in the word, but first it has to turn the letter into a list. This second RUN reports the list of DOT and DASH words to the first RUN, which then runs those procedures. Finally, SPACE moves the turtle to be ready for the next letter. The last line is a recursive call to the same procedure. It gives the next copy of ENCODE all but the first letter of the message word.

Whew! If you didn't follow all that, don't worry. You can take the procedure as a "black box" and use it yourself or with your students anyway.

If you want to encode a message that has more than one word, add this MORSE procedure. It calls the ENCODE procedure, once for each word in the message:

```
TO MORSE :MESSAGE
IF EMPTY? :MESSAGE THEN STOP
   (or IF EMPTY? :SENTENCE [STOP])
ENCODE FIRST :MESSAGE
SPACE SPACE
MORSE BUTFIRST :MESSAGE
END
```

The two SPACE commands in this procedure put a double space between words.

Try these, first typing SETUP to clear the screen:

```
MORSE [TURTLE MORSE CODE]
MORSE [secret message to follow]
```

If your message won't all fit on one line on the screen, you could write a NEXTLINE procedure that moves the turtle down to the beginning of the next line. It might look like this:

```
TO NEXTLINE
PENUP
SETXY -100 YCOR - 30
   (or SETPOS LIST -100 YCOR - 30)
PENDOWN
END
```

You would have to check in the MORSE procedure to see if the turtle were near the right edge of the screen, so the Morse Code wouldn't wrap. Find the right place in MORSE to add a line like this:

```
IF XCOR > 100 THEN NEXTLINE  ;
   or IF XCOR > 100 [NEXTLINE]
```

Different Dots and Dashes

Remember that you don't have to use the turtle to draw dots and dashes on the screen. Because of the way we have used reporters and procedures called DOT and DASH, it is very easy to change the way we show the code. You don't have to change the ENCODE and MORSE procedures at all. They will work fine no matter what your DOT and DASH procedures do!

Text Codes

For example, if you want to print dots and dashes as text, change your procedures to:

```
TO DOT
PRINT1 ".
   (or TYPE ".)
END

TO DASH
PRINT1 "_
   (or TYPE "_)
END
```

Use PRINT1 or TYPE, depending on your version of Logo, to print a character and leave the cursor on the same line.

You will also have to change the SPACE procedure to print a real space, instead of moving the turtle. A space is character code 32, so that is what you print:

```
TO SPACE
PRINT1 CHAR 32
   (or TYPE CHAR 32)
END
```

A Flashing Turtle

Change DOT so that it shows the turtle briefly. DASH should show it for a longer time. Experiment with the durations until you can understand the codes you enter. Don't blink or you might miss a letter! Change SPACE to hide the turtle for a longer time (to separate letters and words). Use a WAIT command, if you want.

Sound Codes

DOT should play a short sound and DASH should play a longer sound. Use WAIT in your SPACE command to cause a slight delay between letters and words.

Your Turn!

Think of other ways to play with Morse Code in Logo! Send messages to your friends. Change the background color of the screen periodically. Send your codes over a modem! Tap your code on the keyboard, using your computer's internal clock to sense the timing (use TICKS and TICKCOUNT using Terrapin Logo for the Macintosh). Use your imagination!

• • • •   • —   • — — •   • — — •   — • —
—
• — • •   — — —   — — •   — — —
• —   — • •   • • • —   •   — •   —   • •
—   • — •   •   • • • !

Dorothy Fitch has been director of product development at Terrapin since 1987. A former music educator, she has also directed a computer education classroom for teachers and students and provided inservice training and curriculum development for schools. She is the author of *Logo Data Toolkit* and coauthor of *Kinderlogo*, a single-keystroke Logo curriculum for young learners. At Terrapin, she coordinates software development, edits curriculum materials, writes documentation, and presents sessions at regional and national conferences.

Dorothy Fitch
Terrapin Software, Inc.
400 Riverside Street
Portland, ME 04103-1068
CompuServe: 71760,366
Internet: 71760.366@COMPUSERVE.COM
(207) 878-8200

Temperatures!

by Eadie Adamson

As our thoughts turn to spring and we begin to leave winter behind, the changes in the thermometer also herald warmer weather and gradual renewal. Looking back on winter, perhaps all we remember is the cold. Last winter one of my students explored temperatures and created a project to make the relative Fahrenheit and Celsius measurements into a visual display. As temperatures rise, this is an interesting idea to share.

Functions

It all started with writing functions. I was working with a fourth-grade student using Logo as a substitute for his mathematics class. "Seb" was probably capable of study at a seventh-grade level. Seb had already skipped one year in school. Advancing him even more would have been difficult, given all the constraints of schedules, let alone the great age disparity that would exist between Seb and his classmates.

We had already worked with polygons and tessellations. Seb's grasp of mathematics was exceedingly strong. He was also a remarkably perceptive and able learner. Seb often worked by himself for half an hour while I completed another class. I had been working a bit with Phil Lewis' wonderful new book, *Approaching Precalculus Mathematics Discretely*. I thought that Appendix A of Phil's book, with its discussion of writing functions in Logo, was interesting and would provide Seb with an appropriate level of challenges for independent work. It also connected Logo and mathematics in a nice way.

Machines

Phil uses the analogy of machines—procedures—beginning with a procedure to double a number. As Phil explains it, the procedure **double** looks like this:

```
to double :num
output 2 * :num
end
```

The input **num** is an input hopper. We are thinking of machines—procedures—as kinds of processors. In this case they are like food processors that have a hopper in the top where something goes in. Some machines, though, have spouts to output something. These machines are more like juice machines, where the fruit goes in one end, is processed, and comes out at the other end as juice. If you think of a Logo function as a juice machine, you have an interesting and effective metaphor. Something goes in, something comes out, but it needs a destination—probably a bottle or a glass. With no destination, a juice machine makes a mess! When Logo is asked to output something and is given no destination, or instruction about what to do with the result, Logo complains

```
I don't know what to do with...
```

This is true for any version of Logo, although the messages vary slightly.

As Phil suggests, we explored writing functions by spending some time working out the kinds of machines we were creating. Seb liked to draw, so drawing pictures of the Logo function machines had considerable appeal. He liked looking at the processes, determining the kinds of machines that needed to be fit together, and deciding what kind of destination (glass or pitcher ?) was needed for the result.

A Practical Problem

Having progressed through many of Phil's beginning exercises, we arrived at a discussion of a "Transformation Machine" to transform Celsius temperatures to Fahrenheit. As it happened, I had just returned from a trip to Montreal. The day I left it snowed and the weather was rather cold. However, in Montreal temperatures are reported, as they are in Europe and elsewhere, using the Celsius scale.

The Relative Meaning of Numbers

We all are so accustomed to hearing a weather report: "Today's high will be..." and can rather easily decide what kind of clothing we need. If it's winter, chances are we need to decide about hats, earmuffs, and extra layers of clothing, depending on the numbers we hear. As spring comes along, we are delighted with a forecast of a temperature in the 60s, say, which means we probably need only a light jacket or sweater. When we hear the high will be in the mid-80s, we seem to know what to expect. Just how accustomed we are to these numbers becomes apparent when everything is referred to on another scale. We may *know* that 0° Celsius is the same as 32° Fahrenheit, but how cold is -6°

Celsius? How *warm* is 25° Celsius? This is an interesting and immediately engaging problem.

I told Seb about the temperature when I left Montreal the day before. It was -6° Celsius. He worked on writing the conversions so that we could figure out how cold it really was. To do this, he had to create several machines that poured information one into another. Seb followed Phil's instructions and made the following processing machines/procedures:

```
    to mult9 :num
    output 9 * :num
    end

    to add160 :num
    output :num + 160
    end

    to div5 :num
    output :num / 5
    end
```

Next Seb wrote a machine to convert Centigrade to Fahrenheit, using the three machines he had just made:

```
    to ctof :c
    output div5 add160 mult9 :c
    end
```

Seb also made the "undoing" machine, reversing the process he had just created. This machine takes the output of what Seb called **ctof** and outputs the input of **ctof**, the original temperature. This reversing process moves backward through the original machine, producing the original input number.

First Seb built some procedures to undo the processes above. The first process multiplied by 9; he now needed a process to divide by 9:

```
    to div9 :num
    output :num / 9
    end
```

Likewise, to undo the **add160** he needed a **sub160** to subtract 160 from the number:

```
    to sub160 :num
    output :num - 160
    end
```

Lastly, to undo the **div5** he needed a **mult5**:

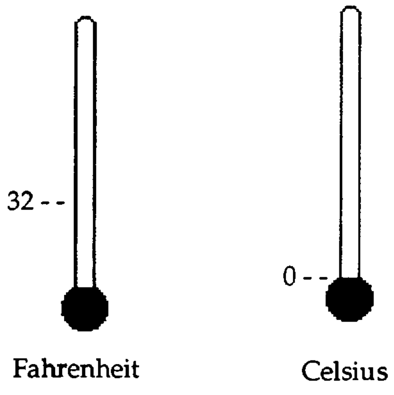
```
    to mult5 :num
    output :num * 5
    end
```

And, with those pieces in place, Seb wrote this converter for Fahrenheit to Celsius:

```
    to ftoc :f
    output div9 sub160 mult5 :f
    end
```

A Problem to Solve

Seb spent some time playing about, making for himself a kind of guessing game: How cold is -16° Celsius? How hot is 25° Celsius? Finally I suggested a project—make a visual display of the temperatures using the processors he had created. Seb created two thermometers on the screen, set two turtles to label each, and then had them draw in the "mercury" to the appropriate level. (The project was created with *LogoWriter* on the Macintosh, but its application would be possible in any version of Logo.)



A Challenge

Seb's representation is interesting, for it shows how he thinks of the temperatures. Notice where the "freezing" mark is on the Fahrenheit scale. There's lot's of room for lower temperatures. But look what he does for Celsius—there's nowhere to go when the temperature goes below zero. On an intellectual level, he clearly understood what we were discussing. Conceptually, and this time visually, Seb "saw" things differently.

Now, here's a challenge for you, the reader. Can you create a program that gives a realistic simulation of the two temperatures? Would you have your thermometers represent the temperature levels as Seb did? Or would you have the 0° and 32° marks level? Why? This was only one flaw in Seb's solution. He also failed to take account of the possibility of having negative Celsius numbers in the procedures he wrote. Anyone living in Canada would need plenty of these!

Send your solutions to me and we'll plan to share them in a future issue of *Logo Exchange*.

Temperatures and Graphs

It occurred to me as I was rethinking this whole problem that such a project is a nice addition to the temperature graphing my fourth-grade classes had done when we shared measurements via *Logo Express* with Carol Goodman's classes at Campbell Hall in California. As a programming project, it would probably be more appropriate for older and more advanced students, but it would be wonderful as a tool for any students to use in order to make the two scales more meaningful.

Bibliography

Lewis, Philip G. (1990). *Approaching precalculus mathematics discretely*. Cambridge, MA: The MIT Press.

Eadie Adamson
1199 Park Avenue
New York, NY 10128
212/876-3276

Email: LCSI LogoExpress BBS, New York and Montreal: EadieA
CompuServe: 73330,3266
AppleLink: EadieA

Call for Papers

1993 European Logo Conference

Place: Athens, Greece.

Dates: 28 to 31 August, 1993.

Main theme: Logo-like learning environments; reflection and prospects.

Principal speakers include: Eric deCorte, Andrea diSessa, Celia Hoyles, Brian Harvey, Richard Noss.

Deadlines: Papers 31 March

Early registration fees: 31 May.

For more information contact:

Triaena Congress,
24 Harilaou Trikoupi str., Athens, 106 79
Greece
Phone +30 1 3609511-15, +30 1 3609552.
Telex 218256.
Fax +30 1 360 7962.
Email: elogo93@grathunl.bitnet

International Logo Conference

Place: Melbourne Australia

Dates: July 4 - 7, 1993

Mission: To create an opportunity for Australian educators and international guests to collaborate, share ideas, and celebrate Logo learning.

Conference Strands:

- Logo and computer science
- Professional development and teacher preparation
- Primary mathematics and Logo
- Secondary mathematics and Logo
- Language arts and Logo
- Logo across the curriculum
- Logo and robotics
- Languages other than English and Logo

Conference includes a free warm-up workshop.

For more information, contact:

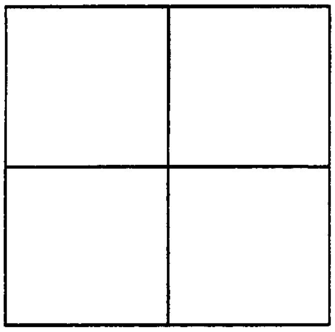
1993 International Logo Conference
MLC Community Education
207 Barkers Road Kew
Victoria 3101 Australia
Phone: 03 810 1412/1426
Fax: 03 819 2345
Internet: K0331@applelink.apple.com

Counting Sequentially and Systematically

by Kay Matsushige

This article presents a math-related, Logo-infused activity with several educational goals:

- to have the students create one type of polygon and "multiply it" so that it shares sides with replications of itself. For example,



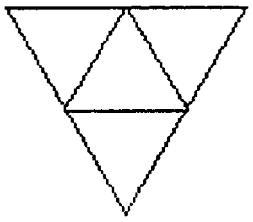
- to have the students learn to count shapes sequentially and systematically.

The lesson can begin by challenging students to come up with a polygon, such as the four squares, shown above. This involves the students in trying to solve the problem of how to place the turtle after it has made one shape so it can make the other one right next to it. A simple procedure such as this solves the four squares problem:

```
TO FOUR.SQUARES
REPEAT 4 [SQUARE 20 RIGHT 90]
END

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

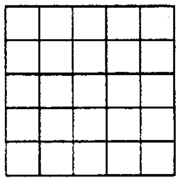
If students have difficulty visualizing a polygon such a FOUR.SQUARES, they should sketch it.

Variables may be introduced so that students can create shapes of different sizes. Another more advanced activity may be to have the students create the large polygon and divide the inside of the polygon into the same shape as the outside.



```
TO TRIS
RIGHT 90
TRI1 90
FORWARD 45
RIGHT 120
TRI2 45
END

TO TRI1 :SIDE
REPEAT 3 [FORWARD :SIDE RIGHT 120]
END

TO TRI2 :SIDE
REPEAT 3 [FORWARD :SIDE LEFT 120]
END
```

Another question you might ask students concerns the number or rectangles in a more complex shape. Introduce a shape that requires sequential and systematic counting, instead of just random counting.



Depending on the students' ability levels, they may number the polygons, color them, or trace them—either on paper or on the computer screen. Students should be encouraged to work in groups to develop cooperation and teamwork. They can then create their own more complex shapes as a group and cooperatively decide on a strategy for counting systematically and sequentially.

After students have had some practice with this idea, they may want to trade shapes among groups. One group could challenge another to find all the polygons in their more complex shape. Students will be motivated to come up with more creative and complicated shapes. Class time should be taken to discuss the strategies used for locating all the shapes. With practice, students should develop better skills for counting in a systemic manner. In the end, this activity may seem to the students more like a puzzle or game than a math lesson.

Kay Matsushige is a former student of Judi Harris at the University of Nebraska at Omaha. She can be reached at

Kay Matsushige
2134 Chardonnay Lane NW
Rochester, MN 55901

Object Logo Student Edition

A student edition of Object Logo is now available for $49.95. It retains the speed of Object Logo and its most commonly used features:

- Full Macintosh color
- Advanced list processing
- Extensive math
- Extensive sample demos
- Multiturtle graphics
- Incremental compiler
- Complete object system with multiple inheritance
- MIDI and LEGO robotics support (with appropriate modules)

This edition contains a 480-page reference manual and a discount coupon toward an upgrade to Object Logo 2.61.

For more information, contact:

Paradigm Software
PO Box 2995
Cambridge, MA 00238
617/576-7675
Fax: 617/576-7680
Internet: paradigm.tec@applelink.apple.com

Logo for the Macintosh, an Introduction Through Object Logo

Logo for the Macintosh teaches the art of programming to first-time programmers. Written for readers 11 years and older, the book approaches programming through explorations in geometry, math, graphics, and language. Topics covered include procedures, variables, turtle geometry, debugging, recursion, and object-oriented programming. Written primarily for users of Object Logo, this book may be used with other implementations of Logo on the Macintosh.
For more information contact

Paradigm Software
PO Box 2995
Cambridge, MA 00238
617/576-7675
Fax: 617/576-7680
Internet: paradigm.tec@applelink.apple.com

My Favorite Kangaroo—A Study in Fascinating Fibonaccis

by Robert Macdonald

Last spring I decided to retire from teaching. Some of the pluses of my action were the number of letters and personal visits of students going back more than 40 years. Two letters in particular come to mind because they reminded me of a mathematical microworld that I had used over the past two or three years.

The letters were from two first graders I taught more than two decades ago: one from a young lady who is now teaching English in Japan, the other from a young man who had returned to his native Germany when he was in the fifth grade. Both former students mentioned the other in their letters. They especially remembered a number of language encounters on which they cooperated when in the first grade.

One of those encounters had been the adventures of a hopping kangaroo they had named KOONY. The illustrations of their text have remained a favorite of mine for years. That was in an age prior to computers and video cameras. However, I was able to record a number of their creations—which they read aloud—on tape. I used those tapes during language workshops on initial reading for years.

Three or four years ago, while teaching fourth grade, I came across a handbook of mathematical encounters for children that detailed the activities of a hopping grasshopper (Greenes, 1979). It brought to mind the strange adventures of my favorite kangaroo KOONY. However, rather than use a grasshopper (I tired of that insect after a science unit earlier in the year), I substituted a kangaroo.

The Problem

A kangaroo wants to hop up some stairs. He has some difficulties. He can only hop up, never down. He is able to hop only one step or two steps at a time. He is unable to manage three or more steps in a single hop. How many different ways can the kangaroo hop to get up some stairs if you know the number of steps in those stairs? If the stairs have two steps, what combination of hops would permit him to get to the top? What if there are three steps, four, five ... fifteen steps?

Hands-On Solutions

Students may choose any mode of arriving at answers to the above questions. Some may prefer to work directly with some physical objects—constructing their own stairs and hopping with bottle caps. Others may prefer to draw pictures of stairs on graph paper and literally trace the hops. The more concrete-thinking students might even resort to finding some stairs and imitating the actions of a kangaroo. A few might be able to visualize the method outline below. (Greenes does offer some possibilities.)

Let's detail the size of hops (1 = a one-step hop, 2 = a two-step hop) for the following stairs:

one step stair	two step stair	three step stair	four step stair	five step stair
1	2	2.1	2,2	2,2,1
	1,1	1,2	2,1,1	2,1,2
		1,1,1	1,1,2	1,2,2
			1,2,1	2,1,1,1
			1,1,1,1	1,2,1,1
				1,1,2,1
				1,1,1,2
				1,1,1,1,1

Providing graph paper and worksheets based on the table below will be essential for most students to gather accurate data. The organization of that data is very important. From the data provided above, make a brief table detailing the following:

Number of steps	Number of different arrangements of hops needed to ascend the various stairs
1	1
2	2
3	3
4	5
5	8

I've always felt that in engaging students in math, one should emphasize patterning. What patterns can you find here? Powers of two, squares of numbers, triangular numbers, or possibly a Fibonacci Sequence among others (AIMS Education Foundation, 1986-87).

A Computer Application

To help students check their work, the following *LogoWriter* program may prove helpful. It can be adjusted in any number of different ways:

```
to startup
commence
end

to commence
show [When you wish to begin type in
   the command <KANGAROO> then touch
   the return.]
end

to kangaroo
clearpage
initialize
fibonacci :numbers :steps
commence
end

to clearpage
if not front? [flip]
rg
ht
ct
cc
end

to initialize
make "numbers [0 1]
print [How many steps do you want my
   kangaroo to jump?]
make "steps readnumber
if :steps = 0 [Print [Please enter a
   number greater than 0.] print [ ]
   initialize]
print []
end

to readnumber
output first readlist
end

to fibonacci :numbers :steps
if :steps < 1 ([insert [The number of
   ways my kangaroo can jump is]
   insert char 32 insert :next.number
   print [.] stop])
make "next.number ((first :numbers)
   + (last :numbers))
make "numbers (sentence last :num-
   bers :next.number)
fibonacci :numbers :steps - 1
end
```

If one names the program KANGAROO and selects it from the Contents Page, the startup procedure will automatically begin the program by asking you to type the command KANGAROO.

When I set up the program for a class, I liked to offer some directions in the work area on the front page of *LogoWriter* as the program began. You can easily do this, then lock the program after it is written. The directions will appear when called up from the Contents Page. One might write:

> Koony the Kangaroo loves to hop up steps. He never hops down. He hops up in only two ways: one step at a time or two steps at a time. He has never learned how to hop up three or more steps at a time.
>
> I would like you to discover how many different ways he can hop up stairs if you know the number of steps.
>
> See your worksheets and use graph paper to work out some jumping patterns. Collect your data and write it down on your worksheet. You should discover an interesting number pattern.
>
> Good luck. Have fun. But think!

The worksheet is shown at the end of this article.

Collecting and Evaluating Data

The second column of our table shown above may now be expanded through 10 steps if the students can discern the patterning.

Number of steps	Number of ways to ascend
6	13
7	21
8	34
9	55
10	89

With this collection of data, the students should analyze how the second column moves forward. It is the famous Fibonacci Sequence named after the brilliant Leonardo of Pisa, who devised it almost 800 years ago (Jacobs, 1970):

```
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89...
```

The sequence begins with 0 and 1. The series reproduces itself by adding the two preceding elements in the series, for example,

```
0 + 1 = 1

1 + 1 = 2

1 + 2 = 3

2 + 3 = 5
```

Further Exploration

Although the *LogoWriter* program listed above begins with the sequence "0 1," a Fibonacci Sequence could begin anywhere.

The *LogoWriter* program produced below may be used by students to set up the first two numbers of a series and then the number of items in the series to be replicated.

```
to fibonacci :firstnum :secondnum
   :counter
if :counter = 0 [stop]
print :firstnum
fibonacci :secondnum (:firstnum +
   :secondnum) :counter - 1
end
```

If one enters the command

```
fibonacci 4 5 12
```

4 indicates the first number in the sequence, 5 indicates the second number in the sequence, and 12 indicates that the first 12 items in the sequence are to be reproduced.

Hence the sequence becomes:

```
4, 5, 9, 14, 23, 37, 60, 97, 157, 254,
   411, 665
```

There is a wealth of publications dealing with the Fibonacci Sequence. Trudi Garland (1987) offers eight chapters brimming with suggestions. If you are as fascinated by the sequence as I am, you will delight in her discussion of Fibonacci numbers in nature, in art, and in architecture. She finds musical applications in works of Palestrina, Bach, and Beethoven, and intriguingly in the formal plan of the first movements of Bartok's *Music for Strings, Percussion, and Celeste.* However, she fails to note the famed Golden Sonata by Henry Purcell, which applies the plan of the Golden Rectangle, which is based on the Fibonacci Sequence.

For the math teacher, Garland's chapter on the mathematics of Fibonacci numbers will provide fodder for the most jaded mathematical tastes.

Consider the following: The Fibonacci Sequence is a recursive sequence because each number is a function of the two preceding numbers. Both of these change as the sequence moves on. This is probably the first time students confront such a possibility. As Garland notes, if the sequence begins with 1 1 2, no common factors can be found in any two consecutive Fibonacci numbers. The sum of any 10 consecutive numbers in the sequence is always divided by 11 evenly. Every third Fibonacci number may be divided by 2, every fourth by 3, every fifth by 5, every sixth by 8, and so on. The divisors are all part of the Fibonacci Sequence.

The possibilities appear endless (Mottershead, 1977). After 800 years, the sequence still fascinates us (Pappas, 1986). It is very likely going to stimulate your class in ways that may well surprise you.

References

AIMS Education Foundation. (1986-87). *Mathematical poster series, Sets A - J.* Fresno, CA: Author. This is a wonderful source for bringing to mind the marvels of mathematical patterning. The work begs for Logo applications.

Burns, Marilyn. (1977). *The good time math event book.* Sunnyvale, CA: Creative Publications. Event 21.

Garland, Trudi. (1987). *Fascinating Fibonaccis: Mystery and magic in numbers.* Palo Alto, CA: Dale Seymour Publications.

Greenes, Carole, et al. (1979). *The Mathworks: Handbook of activities for helping students learn mathematics.* Palo Alto, CA: Creative Publications.

Jacobs, Harold. (1970). *Mathematics: A human endeavor.* San Francisco: W.H. Freeman.

Mottershead, Lorraine. (1977). *Metamorphosis. A sourcebook of mathematical discovery.* Palo Alto, CA: Dale Seymour Publications.

Pappas, Theoni. (1986). *The joy of mathematics: Discovering mathematics all around you.* San Carlo, CA: Wide World Publishing.

Recently retired from teaching, Bob accompanies two choral groups and performs with several chamber music ensembles. He tutors adults with severe reading disabilities, provides tax aid for the elderly and hopes to qualify this spring for the state meet with his Masters Swim Team.

Robert Macdonald
Hawthorne Meadows
10225 Nancy's Blvd.
Grosse Ile, MI 48138

LX Copy-Me Page!

Kangaroo Puzzle

Name: ____________________________________ **Date:** ____________

Use graph paper to plot the data you gather.

See if you can discover a number pattern that will help you solve this puzzle.

After you have gathered your data, check it against the computer program. See how close you came.

Number of Steps	Number of different ways the kangaroo jumps up the steps
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	

Did you discover a pattern? ____________________________________

What is the pattern? ______________________________________

The Star of David and the Spirit of Logo

by Daniel Melman

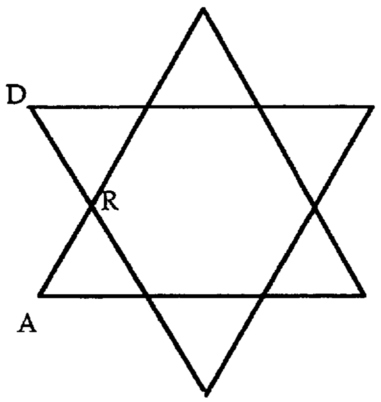
What is the "Spirit of Logo"? While this article will not attempt to define this concept, it will describe a series of activities to exemplify the ideas contained in this phrase.

It all began with a group of teachers exploring ways to draw a Star of David.

One of the teachers tried using two triangles, but she was dissatisfied with the process. After she drew the first triangle at vertex A, there was a problem with bringing the turtle to point D to begin the second triangle at a vertex. It was not easy to calculate the placement of that point.

In order to facilitate the solution to the problem of the connecting procedure, I hinted, "You don't have to calculate. Try to use a part of the turtle path that is already known to both you and the turtle." (A powerful idea!)

After a few minutes of observation and thought, the teacher announced, "I will begin the second triangle not at one of its vertices but at a point R, which is common to both triangles."



This was an interesting and stimulating direction. I remarked that perhaps it was possible to begin drawing the first triangle at the same common point.

Before I had completed my remark, the teacher interrupted me: "... and then I will also finish drawing the first triangle at point R, and at the same point I will begin the second triangle."

Soon, I noticed a commotion around this teacher. She was excitedly telling her classmates about her discovery—the procedure she had written looked very different from any known procedure for drawing a triangle. However, the procedure did produce an equilateral triangle as required, leaving the turtle at a point appropriate for beginning the second triangle.

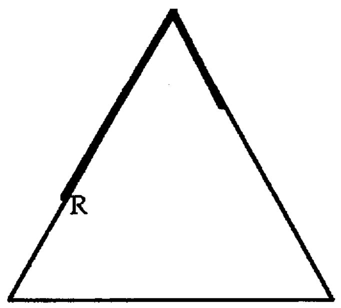
```
TO TRIANGLE
REPEAT 3 [FORWARD 60 RIGHT 120
   FORWARD 30]
END
```

No doubt, this is not quite an "Euclidian" definition for an equilateral triangle. In turtle geometry, however, it is one of the possible solutions, and a most interesting one in that it solves the problem with which we were dealing.

The following "structured" approach was then suggested:

```
TO TRIANGLE
REPEAT 3 [CORNER]
END
```

```
TO CORNER
FORWARD 60
RIGHT 120
FORWARD 30
END
```



The class continued exploring by using the above procedures—trying procedures with variables, testing the relations between the input values for the two FORWARD commands, viewing the new procedure as a broader case of the known procedures for an equilateral triangle, and, of course, completing the solution for the Star of David.

Which brings us to some new problems...

The early activities triggered some different explorations. If you examine the TRIANGLE procedure carefully, you can see that *could* be written as follows:

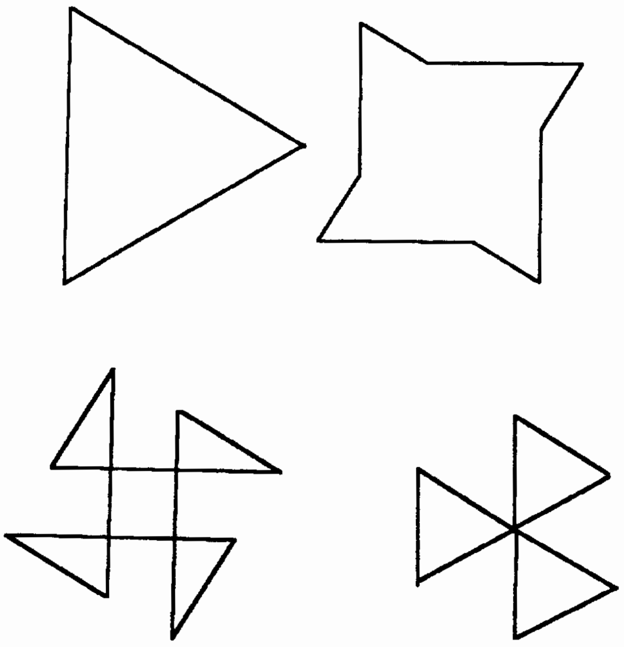
```
TO TRIANGLE
REPEAT 3 [CORNER RIGHT 0]
END
```

The addition of RIGHT 0 does not affect the drawing of the triangle but does pose an intriguing question: What will happen when the input value for RIGHT is other than zero?

To investigate this, it was necessary to use varying values for REPEAT and then define a general procedure with variables:

```
TO PICTURE :NUM :DEG
REPEAT :NUM [CORNER RIGHT :DEG]
END
```

All the drawings that follow were created with the PICTURE procedure, but the family relationship may not be obvious at first glance. Think about what values for NUM and DEG would be needed to produce each of these drawings.



At this point, you have become acquainted with only a few of the representatives of a large "family tree." This family has many fascinating offspring with exceptional personalities, some rather boring offspring with so many lines that they all look the same, and even some unusual offspring, rebels who attempt to break off from the family traditions. It is worthwhile seeking them out and getting to know them. You need only supply the "genetic code"—the values for the variables—to the "Father of the Tribe"—the PICTURE procedure.

This topic can be presented as an open-ended investigation in the following manner.

1. All the drawings shown above were created with the instruction line

   `REPEAT :NUM [(`*`some procedure name`*`) RIGHT :DEG]`

   It is possible to display other drawings from the same family.
   They should all be stimulating!

2. Find the procedure and the appropriate input values for the variables that will produce each of the above drawings.
3. Try the instruction line (with the procedure you found), changing values for the variables, and discover interesting new drawings.
4. Share your discoveries with your colleagues.
5. Suggest a different instruction line with similar traits.

When teachers work on the above task, the activity becomes quite varied and fascinating. Even if new instruction lines are not suggested, interesting questions are posed and point the way to new investigations—for example, "Does the instruction line above represent a function?" The exercises are informative and fun. Try them!

Daniel teaches about the integration of computers in education to preservice and inservice teachers at the State Teacher's college Seminar Hakibutzim. In addition, he is in charge of a team of teachers in a Regional Primayr School. His main areas of interest are Logo and Lego/Logo. He has been working with Lege/Logo with groups of students for more than three years.

Daniel Melman
Nir Yitzchak
Doar Na Negev 85455
Israel

NECC '93 Presents

The Magic of Technology

Hosted by the University of Central Florida, College of Education

Discover what wizardry educators and administrators in the field of learning technology have been conjuring up in their labs.

Magicians and apprentices alike will be charmed by the powerful presentations and exhibits given at this four-day conference.

Share ideas with leaders in the field, and discuss practical approaches to the exciting surprises ahead for education professionals.

Don't miss out on the enchantment of the newest cutting-edge products and ideas shown at the National Educational Computing Conference, "The Magic of Technology."

To receive information as it becomes available, call or write to:
Susan Gayle, NECC '93
1787 Agate St., Eugene, OR 97403-1923
503/346-2834

NATIONAL EDUCATIONAL COMPUTING CONFERENCE

NECC '93
The Magic of Technology
June 27-30, 1993
Marriott's Orlando World Center
Orlando, Florida

Screen Robots

by Glen L. Bull and Gina L. Bull

The LEGO-Logo robotic system never fails to captivate both teachers and children. We had hoped to report on a new LEGO robotics system for the Macintosh that was scheduled to be available about the time this column appears. However, the new system has been delayed until this summer; perhaps we will be able to provide a preview in our next column. In the absence of the hardware that we had hoped to review, we will instead focus on screen robots this month.

The instructions for the existing LEGO-Logo kits are their great strength. Increasingly we are meeting teachers whose first encounter with Logo is through LEGO-Logo. We usually ask teachers to work together in teams of two or three when they begin working with LEGO-Logo. There is almost always someone in the group with LEGO experience; those who never had the opportunity to tinker with LEGO parts as children usually find it satisfying as adults.

In contrast to the LEGO building activity, teachers who have never had experience with Logo sometimes are initially perplexed by the metaphors used. The Logo procedures used to control the LEGO-constructed robots usually work when they are entered into the computer in a rote fashion, but the programming element of the exercise sometimes leaves novices puzzled.

Logic and Programming in a Robotic Environment

This led us to wonder whether we might be able to create an exercise in Logo that would provide a useful pre-LEGO-Logo experience. This led to the thought of "screen robots." Generally when teachers first begin working with LEGO-Logo, we suggest that they begin with one of the simpler activities. For example, construction of a LEGO traffic light is an exercise that most groups can complete in 15 or 20 minutes. More ambitious groups often go on to create a LEGO car that can be programmed to stop when the light turns red.

For those who have not had the pleasure of working with LEGO-Logo, these robotics kits operate through a series of ports in a controller box connected to the computer. For example, if a light were plugged into Port 1, it would be possible to turn it on by entering the following LEGO-Logo commands:

```
TALKTO 1
ON
```

Similarly, a LEGO motor connected to Port 2 could be turned off with the following LEGO-Logo commands:

```
TALKTO 2
OFF
```

Recently a teacher constructed a car whose motor was connected to Port 0, and a traffic light with red, yellow, and green lights connected to Ports 1, 2, and 3, respectively. She wrote the following procedure, which was not working satisfactorily:

```
TO TRAVEL
TALKTO 3
ON
TALKTO 0
ON
WAIT 30
TALKTO 3
OFF
TALKTO 2
ON
WAIT 30
TALKTO 2
OFF
TALKTO 1
ON
WAIT 30
TALKTO 0
OFF
END
```

The problem with the procedure was that the red light was on for several seconds before the car stopped—the teacher wanted the car to stop immediately when the light came on. You may already have spotted the bug in the Logo procedure. However, in a filled classroom with lots of activity it took us a bit longer. First, we had to trace each of the wires back to the LEGO controller box to identify which port was associated with which light or motor. Then we had to remember these port assignments as we searched for the logic error.

We suggested that it might be easier if the program were rewritten in the following fashion:

```
TO TRAVEL
GREEN
ON
CAR
ON
WAIT 30
GREEN
OFF
YELLOW
ON
WAIT 30
YELLOW
OFF
RED
ON
WAIT 30
CAR
OFF
END
```

This approach depends on development of the following type of short subprocedures:

```
TO CAR
TALKTO 0
END

TO RED
TALKTO 1
END
```

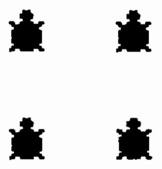
Once the concept of substituting meaningful names in place of port numbers is discovered, programming errors become much easier to identify.

Screen Robots

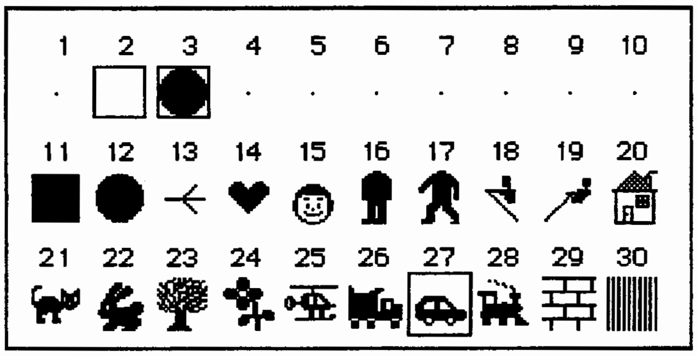
We wondered if we could reconstruct the LEGO-Logo traffic light and car environment in *LogoWriter*. The LEGO-Logo programming environment and *LogoWriter* are similar, so concepts developed in one environment should readily transfer to the other.

LogoWriter provides a total of four turtles. Enter the following command to see all four turtles:

```
TELL [0 1 2 3]
ST
```



Our thought was that the first turtle (Turtle 0) might play the role of the LEGO car, while turtles 1, 2, and 3 served as the red, yellow, and green lights. *LogoWriter* has a Shapes page (which you can access by entering the command SHAPES). Depending on the version of *LogoWriter* you are using (Apple II, IBM, or Macintosh), the exact number that goes with each shape may differ. In the version shown below, the car shape is Shape 27. We also copied the ball shape shown as Shape 12 to location 3, and used the Shape Editor to place a border around it to form the frame of a traffic light. We knew that when we turned the light off we would need an empty square, so we used location 2 to create this shape, as shown below:

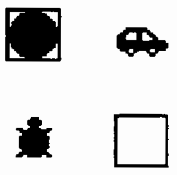


While you are in the Shapes page, you may find the following commands helpful as you copy and paste shapes. <Open Apple> F for Apple II computers (or <Ctrl> F for IBM computers) flips between the overview of all the shapes and the Shape Editor. When you are in the Shape Editor, you can change a dot from white to black (or vice versa) by pressing the space bar. In some versions of *LogoWriter*, Shape 3 may look better if you simply outline the corners rather than placing a border around the entire ball. The following table summarizes some of the Shapes page commands:

Apple	IBM	Command
Open Apple 3	F3	Copy Shape
Open Apple 4	F4	Paste Shape
Open Apple F	Ctrl F	Enter/Exit Shape Editor
Escape Key	Escape Key	Leave Shapes Page

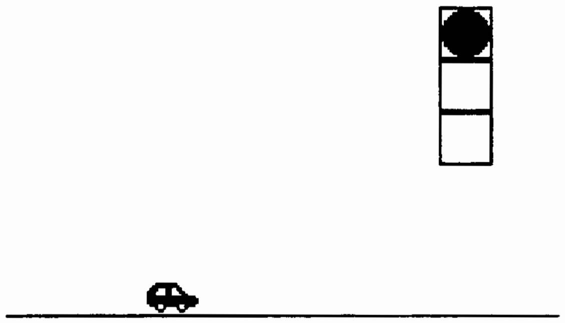
Press the Escape key to leave the Shapes page and return to your previous *LogoWriter* page after you have created Shapes 2 and 3. After you have returned to your original *LogoWriter* page, enter the following commands to set turtles 0, 1, and 2 to the shapes of the car, the traffic light, and the empty border, respectively:

```
TELL 0
SETSH 27
TELL 1
SETSH 2
TELL 2
SETSH 3
```



Constructing a Traffic Light

These elements can be used to create a street with a car and a traffic light. In reality the traffic light consists of turtles 1, 2, and 3 stacked atop one another, while the car consists of Turtle 0.



You may find it useful to develop a procedure to build the traffic light. Depending on the version of *LogoWriter* and the computer you are using, you may have to experiment with the coordinates and shape numbers, but the following procedure illustrates a general format that could be used for a BUILDLIGHT procedure.

Important Tip: You will need to create Shapes 2 and 3 in the LogoWriter *Shape page before they will be available for use.*

```
TO BUILDLIGHT
TELL [1 2 3]
SETSH 2
PU
TELL 1
SETPOS [50 50]
TELL 2
SETPOS [50 30]
TELL 3
SETPOS [50 10]
TELL 0
SETSH 27
PU
SETPOS [-120 -50]
END
```

Once everything is in place, we are ready to control the traffic light. Procedures to allow us to talk to the red, green, and yellow lights will make it easier to develop the program. As you can see, these *LogoWriter* procedures are almost identical to the ones for the LEGO-Logo project described above, except that TELL 1 is substituted for TALKTO 1:

```
TO RED
TELL 1
END

TO YELLOW
TELL 2
END

TO GREEN
TELL 3
END
```

After the procedures that allow us to talk to the lights have been defined, additional commands that allow us to turn each light on or off are needed. In this instance, we will set the turtle shape to Shape 3 to turn it on, and to Shape 2 to turn it off:

```
TO ON
SETSH 3
END

TO OFF
SETSH 2
END
```

Testing the Components

Once you have defined these procedures, experiment with commands to turn the lights on and off:

```
RED ON
RED OFF YELLOW ON
```

Test each light (red, yellow, and green) to be sure it will turn on and off properly. If everything is working, a procedure to send the lights through an entire traffic cycle is straightforward:

```
TO CYCLE
RED
OFF
GREEN
ON
WAIT 60
GREEN
OFF
```

```
YELLOW
ON
WAIT 60
YELLOW
OFF
RED
ON
WAIT 60
END
```

If the traffic light cycles properly, we are ready to focus on moving the car. In LEGO-Logo it is necessary to build the car first, but in screen Logo we simply have talk to the turtle that has the car shape:

```
TO CAR
TELL 0
END
```

LEGO-Logo has a feature that allows a device to be turned on for a specified period of time. The following procedure will simulate the ONFOR command for the car in *LogoWriter*. (The command TONE 40 1 in the ONFOR procedure causes the car to make a noise as it moves forward; if you would like a silent car, substitute WAIT 1 in place of the TONE command. You may also want to experiment with using different frequencies in place of 40 in the TONE command.)

```
TO ONFOR :AMOUNT
SETH 90
PU
REPEAT :AMOUNT [FORWARD 1 TONE 40 1]
END
```

To cause the car to move forward for 60 seconds, the following command would be entered:

```
CAR ONFOR 60
```

The Robotic Simulation

If both the car and lights work properly, we are ready to simulate the LEGO-Logo robotic traffic light and car with the aid of screen robots in *LogoWriter*. The following procedure is our simulation:

```
TO TRAVEL
RED
OFF
GREEN
ON
WAIT 20
CAR
ONFOR 60
GREEN
OFF
YELLOW
ON
CAR
ONFOR 60
YELLOW
OFF
RED
ON
WAIT 90
END
```

The TRAVEL procedure allows the car to stop each time the light cycles to red and continue when the light turns green:

```
REPEAT 2 [TRAVEL]
```

This simulation makes it possible to use screen turtles in *LogoWriter* to recreate some of the elements found in LEGO-Logo. The original Logo turtle was adapted from an experimental robot roaming the halls of M.I.T. Seymour Papert and the Logo development team recognized its instructional potential and developed a screen turtle that mimicked the behavior of the floor robot. By using the multiple screen turtles of *LogoWriter* to copy some of the behaviors of the robotic environment of LEGO-Logo, in a sense Logo is coming full circle.

The value of meaningful names is one of the more important lessons that might be learned in such an environment, we believe. It would be possible to write the TRAVEL procedure using TELL commands that directly address the turtle. However, in that case the procedure would be less comprehensible to humans:

```
TO TRAVEL
TELL 1
SETSH 3
TELL 2
SETSH 2
WAIT 20
TELL 0
SETH 90
PU
REPEAT :AMOUNT [FORWARD 1 TONE 60 1]

    etc.

END
```

The creation of subprocedures such as RED, YELLOW, GREEN, and CAR makes the code of the main TRAVEL procedure far easier to follow and debug. The value of meaningful names and the usefulness of subprocedures holds true for both *LogoWriter* and LEGO-Logo environments.

Extending the Robotic Environment

We hope our screen robots are useful as a precursor to LEGO-Logo. There are other procedures that might be developed as enhancements. For example, in LEGO-Logo the RD command is used to reverse the direction of the motor. An RD command possibly could be developed for the screen robots that would cause the car to reverse its direction and back up. If the traffic light were reduced to its red and green elements, the YELLOW turtle could be pressed into service as a second vehicle—perhaps as a truck, to distinguish it from the car. If your computer and the version of *LogoWriter* have the appropriate colors, you may want to incorporate them into the stoplight.

A more ambitious exercise would be to develop more extensive environments through screen robot simulations with turtles. For example, is it possible to simulate a washing machine or a conveyer belt on a computer screen with LogoWriter. We leave those projects as an exercise for the reader or for the reader's students.

Glen Bull is an associate professor in the Instructional Technology Program of the curry School of Education at the University of Virginia. Gina Bull is a system administrator in the Department of Computer Science at the University of Virginia. By day she works in a Unix environment, by night in a Logo environment.

Internet Addresses: GBull@Virginia.edu, Gina@Virginia.edu
BITNET Addresses: GBull@Virginia, Gina@Virginia

...LogoMaths to LME...

-by A. J. (Sandy) Dawson

Vancouver experienced a particularly warm and pleasant summer in 1992, capped by endless—it seemed!—weeks without the usual summer rains. This fine weather combined with the mountaintop setting at Simon Fraser University (SFU) made the occasion of the Sixth Logo and Mathematics Education (LME6) conference all that more pleasant. Held at SFU from July 16 to July 20, 1992, the conference was attended by 35 colleagues from Australia, Britain, Greece, Spain, the United States, Israel, and Canada.

The conference was a mixture of plenary addresses, workshops, hands-on activities, small-group discussions, individual presentations, and frequent conviviality. After a late morning DimSum, a walking tour of Vancouver took up the better part of one day, topped off with a delightful Greek dinner at a restaurant overlooking English Bay and the North Shore mountains. A Sunday evening Canadian-style BBQ was the only occasion on which Brian Harvey had to retreat indoors to escape an early evening onslaught of tiny, winged beasts intent on making a meal out of the BBQ guests, particularly Brian.

In her plenary address, Janet Ainley (1992) captured the spirit not only of LME6 but also of previous conferences and the people who travel from such great distances in order to attend:

> I want to tell you a story. It is the story of my community, the Elemee people. We are a people who are widely scattered, who gather only rarely. Although the physical distances that separate us are great, the spirit that unites the community is strong. Although scattered, many of us are able to keep in contact through strange rituals in front of screens and keyboards, where the use of mystic codes allows messages to pass between us. Indeed there are some of the Elemee people for whom these rituals assume great importance, and if the codes are lost or they are cut off from the source of the messages, they soon become weak and distracted. They need the care and vigilance of their friends to soothe and comfort them until they can re-create the ritual and renew the connections.
>
> Although physical gatherings are rare, it is through these that the spirit of the Elemee people is kept alive. They travel great distances, and often through great adversity, to meet together and to renew their sense of community. These meetings are times of celebration, marked by much ritual hugging and elaborate planning for the enjoyment of food and drink. The responsibility of the hosts of such meetings is great, as shortcomings as well as triumphs will be remembered and recounted for many years. At these times, our thoughts go out to those of our people who are not able to be with us.

Yes, we did miss those of you not able to attend. And it certainly was the case that each morning before sessions began, and at each break during the day, and well into the evening, Elemeers scurried to the Centre of Education Technology at SFU to sign on to their home computers through Internet, to sit in front of screens and use mystic codes to pass messages to those not in attendance.

In addition to the plenary address given by Janet Ainley, Celia Hoyles (1992) challenged the gathering with thoughts about the future and about the directions Logo and mathematics education might take.

Celia began her plenary address that opened the conference by recounting a conversation she had at LME4:

> It was at the 1989 Israel conference ... that a ... group of people sat on the grass and thought:
>
> The trouble is we're not all working in Logo anymore. Some of us are still working in Logo, but some are working in other software environments. Is there any point in carrying on meeting under the LogoMaths banner?
>
> Very thankfully, I think, we decided yes, and we changed LogoMaths to LME. I think this is quite an interesting mathematical event. Perhaps to begin with in all our minds we were transferring back to its referent—LME meant LogoMaths Education. But I think, as with any symbolization process, LME now has a life of its own. We don't have to translate back to what it actually meant a few years ago.... I think there is something that has stayed from LogoMaths to LME, and this is the business of an evolution in mathematical culture.

> When we moved from LogoMaths to LME, what we maintained, what was invariant in that transition, was the spirit of changing what was going on in school mathematics.

LME6 was very much about changing what goes on, or what might go on, in school mathematics. Both in the workshops and during individual presentations of work done or currently going on, a spirit of exploration and adventure was evident. The Elemee folk were seeking ways to improve the teaching and learning of mathematics, and the reports they gave bode well for the future.

Uri Leron (1992) added a personal note to the deliberations when he spoke in his plenary address about the changes that have taken place within him as he, over the years, progressed from being an undergraduate student of mathematics to being an active member of the Elemee community. In the selections from his address given below, he talks about the stages he went through in making that transition:

> The first stage was when I was an undergraduate math major. I call this my "stage of linear understanding."
>
> By linear understanding I mean that I was able to formulate proofs, which was the main part of the course, in a step-by-step manner, convincing myself that each step indeed follows from the previous steps, and that I was able to write that down in an exam. And I was very happy with that...
>
> Well, actually, when I reflect back on this period I think it's not quite a precise description because this is my recollection if I try to define my cognitive understanding of the period. But I remember that I actually did do some homework exercises and some of them were nontrivial. I doubt that I could have done them if I had only linear understanding. But still that's how I remember the stuff and probably this means that whatever understanding was not linear was totally unconscious—maybe there was something I didn't know about but was still there. And I discovered it not because anyone took the trouble to tell me....
>
> I spent many, many hours doing [exercises in Algebra II] and in the process I discovered a lot of things that I probably wouldn't have discovered if I hadn't the time to do this.
>
> And the thing I discovered was that there is an intuitive side to mathematics. I remember the great excitement with which I discovered this.... I think there was only one other time when I had this similar excitement and that was when I was very little kid maybe five or six...
>
> And then I was determined that when I became a teacher or an instructor at a university or whatever, I would be better.... I would tell my students about these things. So that was the beginning of intuition.
>
> The next stage... I call... something like "a beginning instructor" or "a young instructor." Then the next one is "an experienced instructor," in which I knew this [telling] doesn't work. This was actually a period of frustration because I was realizing that explaining doesn't do the job, but I didn't have any alternatives.
>
> Anyway, I started reading this [Logo] stuff.... I think the only way to describe what happened to me is that I fell in love with the thing.... I really fell in love.... I mean I had a tremendous emotional reaction to what I was reading. I think probably I was quite ready because for many years I had been realizing that things don't work...and here comes this man.... Papert ... and he actually says that things don't work and we have an alternative—we have ways of teaching in a different way.
>
> Anyway, this was the period when I really think the most important things in my development happened.... I started thinking in terms of learning by doing and learning from errors and all of the other things.

I would conjecture that many readers will resonate with Uri's description, and will be able to look back upon their own lives—their own coming into being as mathematics teachers—and note those points where changes occurred within themselves, where they started thinking in terms of learning by doing and learning from errors, and so on.

In organizing LME6, a very concerted attempt was made to provide the time, space, and equipment where the learning by doing and the learning from errors that Uri spoke about could, in fact, occur. One of the places this happened most predictably was in the four workshops. Each workshop was four hours in length, and each participant could attend two of the four workshops. The workshops focused on four pieces of software the organizers and workshop leaders felt would, now or in the future, significantly impact the manner in which learners interact with computers and confront mathematical ideas.

The leaders of the workshops and the software they presented were Andy DiSessa and Laurie Edwards (Boxer), Joel Hillel (Maple), Tom O'Shea (Geometer's Sketchpad), and Uri Leron and Orit Hazzan (Isetl).

Workshop participants had lots of opportunity to become familiar with the software and to engage in oftentimes spirited discussions about the uses that might be made of the particular software under discussion in bringing learners into contact with mathematical ideas.

But the workshops were not the only occasions when Elemeers got to play with and explore new software. Uri Wilenski, Wally Feurzeig, Benoit Cote, David Mitchell, Rina Cohen, Tony Jones, and Chronis Kynigos were among some of the participants who offered mini- and sometimes impromptu sessions that seemed to never want to end despite the arrival of food and drink! In these sessions, they invited others to share in their excitement and insights and experiences of working with learners of many different ages as these learners engaged in mathematical thinking.

But alas, like all good things, the conference had to end. It was left to Janet Ainley to bring us full circle, to focus on that movement from LogoMaths to LME. Below are some excerpts from her closing plenary. They give a flavor not only of the spirit of the meeting but also of the concerns that Elemee folk carried away with them from the meeting.

> In some ways this has been a very different kind of gathering for the Elemee people. The traditions of enjoyment of good food and the renewal of old friendships have been maintained, but although some of the people still wear the sign of the turtle as a form of adornment, the turtle itself has had little influence on the business of our meeting. There has been very little emphasis on the presentation of research and much more on discussion and working groups. More than ever, there has been a tacit assumption of shared values and beliefs underlying both the form and the content of our interactions.
>
> In speaking at the end of a conference I have both the advantage of having the last word and the disadvantage that much I might have wanted to say has already been said. Before this conference began, I had planned to talk about the Elemee people growing up, from our romantic, idealistic but essentially naive beginnings in 1985, through a stage of striving for academic respectability, to a more realistic and sophisticated view of the role of Logo—or other similar computer environments—in mathematics education, but now this story seems too simplistic.
>
> I think we...have a sense of disappointment that the revolution we thought we were leading in 1985 hasn't happened.
>
> Perhaps we have been too impatient. In Benoit's terms, I think we are still very close to the Big Bang. The important factor here is not how long Logo has been around but how long teachers and children have had access to it.
>
> This may not look like the revolution we dreamed of or embody the Logo spirit as we would once have envisaged it. It certainly isn't an integrated Logo/mathematics curriculum, but it seems a realistic aim for all pupils, not just those involved in special projects, to have enough familiarity with and access to computers to use them as mathematical tools. For most schools this is still some way in the future, and I am coming to believe more and more that it won't be achievable until portable machines that sit on your desk alongside other mathematical tools become affordable or until the financial priorities change.
>
> I don' t think that we know enough, still, about how pupils who have grown up with free access to computers will choose to use them, or how their experience of Logo will influence their views of mathematics.
>
> More generally, we know relatively little about the ways in which people use computers for doing—rather than learning or teaching—mathematics.
>
> As a research community, we naturally tend to talk and write to each other. The interest and support of our colleagues are important for our work and for our personal satisfaction. But we are perhaps less expert at communicating our work to a wider community and supporting teachers who are dealing with the realities of Logo and mathematics in the classroom.
>
> And so I return to the history of the Elemee community. It is a story that is not completed, but none of us knows how it continues. I am sure that the Elemee will continue to evolve, and perhaps its meaning and purpose will change.
>
> Vancouver is a city of mountains. If you visit Grouse Mountain, at the summit there is a magical room where you may see and hear many strange things. Amid the visions, words, and music that I experienced there, these words stayed

clearly in my mind—and I saw them perhaps as a message to my people.

This is the greatest transformation of all, to become what you have always been.

And so, at the close of this meeting, I give you, the Elemee people, this thought: become what you have always been, lovable mathematics educators.

Plans for the next LME have not been finalized, nor has its location been determined. But of one thing you can be assured. Somewhere around the world at this very moment an Elemeer is huddled in front of a screen, pecking away at a keyboard, in that strange daily ritual where the use of mystic codes allows messages to pass to other Elemeers similarly occupied in other parts of the world, and each is endeavoring to become what they have always been, and to be a lovable mathematics educator.

References

Ainley, Janet. (1992). Turtling off into the sunset. In A. J. (Sandy) Dawson & Rina Zazkis (Eds.), *Proceedings of the Sixth Logo and Mathematics Education Conference.* Vancouver, BC: Simon Fraser University.

Hoyles, Celia. (1992). School interface and curriculum change. In A. J. (Sandy) Dawson & Rina Zazkis (Eds.), *Proceedings of the Sixth Logo and Mathematics Education Conference.* Vancouver, BC: Simon Fraser University.

Leron, Uri. (1992). Who does what and why: From the diary of a radical applied constructivist. In A. J. (Sandy) Dawson & Rina Zazkis (Eds.), *Proceedings of the Sixth Logo and Mathematics Education Conference.* Vancouver, BC: Simon Fraser University.

Sandy Dawson is an associate professor of mathematics education at Simon Fraser University, and director of that institution's teacher education program. His most recent research interests center on the areas of LEGO/Logo and on the exploration of what mathematics lessons with a humanistic focus might look like.

A. J. (Sandy) Dawson
Faculty of Education
Simon Fraser University
Vancouver, BC Canada V5A 1S6
Email address:Dawson@sfu.ca

When You Are Really Serious About Logo...

Introducing PC Logo 4.0, a powerful new version of the Logo programming language designed for the IBM PC and compatibles. PC Logo 4.0 is versatile and flexible, suitable for novice as well as experienced programmers. With more than 300 built-in commands, PC Logo 4.0 supports all the functions you would expect from a full-featured Logo program.

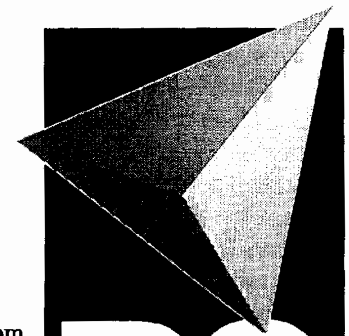
New PC Logo 4.0 features include:

- EGA/VGA screen support
- More than 80 new primitives
- On-line help system
- Full mouse support
- Fully integrated editor
- Laser printing

There's also a growing list of Logo materials, books and curriculum from educators and Logo experts. Low-cost multiple-workstation licensing available, too.

For more information or to order PC Logo, call
800/776-4610

HARVARD
ASSOCIATES, INC.
10 Holworthy Street
Cambridge, MA 02138



PC logo™
Version 4.0

I Know How to Move the Turtle—But What's All This List Stuff?

by John Gough

Turtles catch on quickly when students start learning Logo, However, few students learn Logo in enough depth to learn much about using "lists" in Logo or *LogoWriter*. Making the turtle draw squares and spirals and whiz around the screen is easy to learn because you can *see* what the commands mean and what they do. But list handling is not so visible or so easy to understand. However students can be helped over this hurdle if they are given some sample procedures that do sensible—although not necessarily obvious things—with lists. This article presents some simple procedures that can lead to real skill and insight. Once these have been mastered, as with turtle geometry, the only limits are those of the student's imagination.

So What's a List?

First, what is a "list"? We all know shopping lists and other everyday examples of lists. In Logo a list is an ordered list of words and or lists. It is represented using a pair of square brackets. For example, the standard REPEAT requires two inputs: a number and a *list*. In the well-known REPEAT command to draw a square

```
REPEAT 4 [FORWARD 50 RIGHT 90]
```

the number is 4 and the *list* of commands to repeat is [FORWARD 50 RIGHT 90]. Similarly, the primitive SETPOS requires a list consisting of two numbers. Lists can contain commands, numbers, letters, words, and even other lists. The Logo language contains many primitives that allow us to work with lists.

Lists can be made of Logo words

```
[HOUSE BIG.DOOR WINDOW TREE]
```

or they can be made of other lists

```
[ [CAT DOG] [MOUSE HORSE] [TREE GRASS
   PLANT] ]
```

or they can be made of both words and lists

```
[ [HOUSE [MOUSE HORSE] WINDOW TREE [CAT
   DOG] ]
```

Creating Lists

Now let's write a procedure to make a list. The first procedure, START.FOLDER, gives a user instructions.

```
TO START.FOLDER
PRINT [This builds up a list of what-
   ever words you put in]
PRINT [Type a word and press RETURN]
PRINT [type Q to quit]
MAKE "FOLDER [ ]
BUILD.FOLDER
END
```

The next procedure is recursive and builds a list called FOLDER by adding to the end of the list each new word the user types until the user types q to stop.

```
TO BUILD.FOLDER
MAKE "INPUT READLISTCC
IF :INPUT = [Q] [TONE 300 20 STOP]
MAKE "FOLDER SENTENCE :FOLDER :INPUT
PRINT :FOLDER
BUILD.FOLDER
END
```

Note that the expression " "FOLDER"—with the double quote mark at the beginning of the word—represents the *name* of the variable called FOLDER. By contrast, the expression ":FOLDER "—with the colon at the beginning of the word— refers to the *value* of the variable called FOLDER. Think of FOLDER as though it is a real filing-cabinet folder. Such a folder usually has a name and also usually has something inside it. The name of the folder is related to the contents of the folder but is not the *same* as the contents.

Before a variable can be used it must first be defined. In START.FOLDER, the variable FOLDER is initialized to the empty list. Then in BUILD.FOLDER, the variable has new values inserted by the MAKE command each time the procedure is run.

Spend some time experimenting with these procedures. What happens after you type a few words? Printing the value of lists as they are created in a Logo procedure is the list-handling equivalent of acting out what the turtle does with graphic commands.

Working With Lists

We want to be able to do things with lists, so we need some more procedures. The next two procedures begin to use some of the primitives that allow us to manipulate lists.

The first procedure defines a variable called PACK, which is a list of letters.

```
TO START.JUGGLE
MAKE "PACK [A B C D E]
PRINT :PACK
JUGGLE
END
```

Then the recursive procedure JUGGLE manipulates the list. It can only be stopped by pressing the Stop keys.

```
TO JUGGLE
MAKE "PACK SENTENCE :PACK (FIRST
   :PACK)
MAKE "PACK BUTFIRST :PACK
PRINT :PACK TONE 200 20
JUGGLE
END
```

The second line of JUGGLE redefines the value of PACK to be a new list consisting of the current value of PACK followed by the first item in PACK. The third line uses the primitive BUTFIRST to redefine PACK by removing the first item in PACK. That is, the left-most item is pulled off the "left" and stuck at the "right" end of the list, over and over again. This can be used to bring an item forward so that it can be removed or compared with something or changed. It can also be used to make the turtle appear to juggle turtle-shapes like a circus juggler.

Here is a procedure that shows how lists can be manipulated randomly to make new lists.

```
TO GRAB.WORDS
MAKE "ONE [YELLOW ANGRY PURPLE INNO-
   CENT SMILING FLUFFY OLD]
MAKE "TWO [LETTERBOX GRANDFATHER
   BUTTERFLY COURAGE MUSTARD]
MAKE "GRAB.ONE PICK :ONE
MAKE "GRAB.TWO PICK :TWO
MAKE "LINE SENTENCE :GRAB.ONE
   :GRAB.TWO
PRINT :LINE
END
```

GRAB.WORDS makes use of the procedure PICK.

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

Think of PICK as a primitive until you have more experience with lists. Now run GRAB.WORDS a few times to see what happens.

Did you find that GRAB.WORDS picks one word randomly from each of the two lists and then prints a line containing those words? PICK is used to get the random choice. Then LINE is created by using SENTENCE to put the two selections together.

You can use GRAB.WORDS to make a "poetry machine" that will make random combinations of words from lists of words. Or it could be used to generate all the possible permutations or combinations of items in a list. Here is one simple way of extending this idea so that a user can build two lists of words and then be given some random combinations of words.

```
TO BEGIN JUMBLE.WORDS
CT
HT
MAKE "ADJECTIVES [ ]
MAKE "NOUNS [ ]
MAKE "VERBS [ ]
INSTRUCT
END

TO INSTRUCT
CT
PRINT [Give me some words and I'll
   make jumbled sentences with your
   words.]
PRINT [Type some adjectives, then
   press RETURN.]
MAKE "ADJECTIVES SENTENCE :ADJEC-
   TIVES READLIST
PRINT [Type some nouns, then press
   RETURN.]
MAKE "NOUNS SENTENCE :NOUNS READLIST
PRINT [Type some verbs, then press
   RETURN.]
MAKE "VERBS SENTENCE :VERBS READLIST
MIX
END

TO MIX
MAKE "ADJECTIVE.CHOICE PICK :ADJEC-
   TIVES
MAKE "NOUN.CHOICE PICK :NOUNS
MAKE "VERB.CHOICE PICK :VERB
MAKE "LINE (SENTENCE
   :ADJECTIVE.CHOICE :NOUN.CHOICE
   :VERB.CHOICE)
PRINT :LINE
PRINT [Type the word INSTRUCT to add
   more words, or MIX to get another
   jumbled line]
END
```

Suppose we want to avoid choosing something we have already chosen from a list. We need to build up a "memory" of what has been selected so far, and check whether each new choice is already in that memory.

```
TO START.DRAW.WITHOUT.REPLACEMENT
MAKE "DATA [ONE TWO THREE FOUR FIVE
   SIX SEVEN EIGHT NINE TEN]
MAKE "CHOSEN.MEMORY []
PRINT :DATA
DRAW.WITHOUT.REPLACEMENT
END

TO DRAW.WITHOUT.REPLACEMENT
IF (COUNT :CHOSEN.MEMORY)=(COUNT
   :DATA) [PRINT :CHOSEN.MEMORY
   STOP]
MAKE "CHOICE PICK :DATA
IFELSE MEMBER? :CHOICE
   :CHOSEN.MEMORY
   [DRAW.WITHOUT.REPLACEMENT][MAKE
   "CHOSEN.MEMORY SENTENCE
   :CHOSEN.MEMORY :CHOICE]
END
```

This is what happens in Bingo when you randomly choose numbered balls out of a barrel one at a time and do not put them back.

This is an excellent time to resort to physically simulating what a collection of procedures actually does in order to understand them. The first procedure defines the list DATA and the pool of items from which we will be drawing, and gives an initial value of the empty list to the variable CHOSEN.MEMORY. The second procedure picks an item from :DATA, then uses MEMBER? to test whether this choice is already in CHOSEN.MEMORY. If it is, a new CHOICE is made. If it isn't, CHOICE is added to CHOSEN.MEMORY. The procedure stops when there are as many items in CHOSEN.MEMORY as in :DATA.

Experiment with this last set of procedures. The best way to understand lists is to spend time experimenting with procedures that make use of lists.

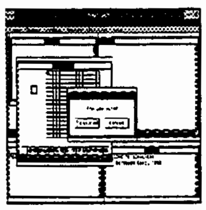
John Gough has worked in tertiary teacher education for more than 15 years. He is the author of *Learning to Be a LogoWriter Writer: A Resource Manual for Adult Learners* (Deakin University Press), a project that grew from nearly five years experience teaching *LogoWriter*. He is interested in *LogoWriter* applications for school mathematics and is currently working on a resource manual and teaching stacks for *HyperCard* and its Logo-like programming language HyperTalk.

John Gough
Lecturer in Education
Deakin University (Toorak Campus)
336 Glenferrie Road
Malvern Victoria 3144
Australia
Phone: 03 8235234
Fax: 03 8225493

Logo keeps getting better with WIN-LOGO™

WIN-LOGO is one of the most advanced Logo language products available today. It represents a new generation of Logo language products using a *Windows*- like graphical user interface and easy-to-use pull-down menus with mouse and printer support. Features include:

- 12 turtles with color, shape, thickness, and different font styles for graphics writing – supports import of PCX images
- Over 300 predefined commands – allows the user defined functions in either C or Assembly language
- Flexible text editor with copying, erasing, search/replace, and interpreting functions
- Interactive debug tool – can set rate and breakpoints for visual analysis
- Supports industry standard color graphics cards – CGA, EGA, VGA 16 OR 256, and MCGA



SOFTEAST ® CORP
2352 Main Street
Concord, MA 01742

For more information or to order, please call 508-897-3172

Ask about the WIN-LOGO Special Introductory Promotion

Technology Learning For a Better Future

A First Course in Programming

in Terrapin Logo, LogoWriter, and PC Logo

This is a complete curriculum for a semester course in programming. It includes student activity sheets, teacher lesson preparation sheets, tests, quizzes, assignments, and sample solutions for all student assignments (hard and softcopy!)

A First Course in Programming is a directed learning environment in structured programming. Its 450 pages emphasize problem solving strategies, critical thinking skills and solid principles of computer science.

Only $150 for a building site license. Call us for further information!

Curriculum written BY teachers FOR teachers!

Logo Curriculum Publishers
4122 Edwinstowe Avenue
Colorado Springs, CO 80907
1-800-348-5646 (FIT LOGO)

Connect with telecommunications leaders from all over the globe.

Join us at Tel•Ed '93—*Global Connections*—The Second International Symposium on Telecommunications in Education and meet with telecommunications leaders from all over the globe.

Be one of the estimated 1,000 educators, policymakers, and researchers who will join together in this unique international forum—for the first time since 1989—to exchange the latest in telecommunications ideas, techniques, strategies, and policy concerns.

Topics will include:
- Multimedia and telecommunications
- Global educational development through telecommunications
- Telecommunications in math, science, and technology education
- Policy and legal issues

To receive attendee information as it becomes available, call Juanita Benzer, 512/471-4014.

Tel•Ed '93
Global Connections

The Second International Symposium on Telecommunications in Education
November 10-13, 1993 ◆ Dallas, Texas, USA at the INFOMART
Tel•Ed '93 is sponsored by the International Society for Technology in Education (ISTE).

Papert on Technology and Megachange

by Julie S. Meredith and Douglas H. Clements

Two people, a surgeon and a teacher, travel from their home in the 19th century to the present day. The surgeon visits an operating room. How do you think he would feel? With the myriad of technological tools and blinking lights, and an apparently dead patient, the surgeon would probably feel bewildered.

The teacher, on the other hand, would immediately recognize the goings on of the classroom. Should the regular teacher be called away, the time traveler could easily take over the class.

What does this tell us about educational progress? So began a recent talk by Seymour Papert at the University at Buffalo. He challenged the audience to consider the following: There has been megachange—a change so significant that the time travelers would indeed feel lost—in transportation, telecommunications, medicine, and construction. Why has there not been megachange in schooling?

One possibility is that megachange just isn't appropriate for schools. For example, while a change in the field of surgery seems inevitable, we have not changed the way we eat. The way babies learn, through interacting with their environment, is not going to change, but according to Papert, "School is essentially a technical act."

Megachange in schools *can, should, and will* come about. If the time travelers went home with kids, they would see some megachange. For example, the kids would be immersed in Nintendo—in itself a radical change. And not just a change in entertainment. Lots of learning occurs, although we may not like it all. There is a concentration of quality learning. There is also quite a bit of knowledge being learned (read *Nintendo Power*)—more knowledge than anyone would dare put in a third- grade social studies unit.

Finally, kids want to learn *fast*. Those who learn the tricks of the new game first, garner the most respect.

This powerful learning experience may give us an opportunity to reach into the kids' world—a point of leverage for our educational interventions. There is an opportunity for conversation about learning. We must be careful, though, for one of the reasons kids love this learning is that they can learn in their own way.

What about a classroom situation? Papert tells about meeting four-year-old Jennifer. Jennifer heard that Papert was born in Africa. She asked him, "How do giraffes sleep? I cuddle my head. My puppy does too. So how does the giraffe sleep? If he can't cuddle his head, can he get enough sleep?" Papert admitted that he didn't know how giraffes sleep and sat down to talk to her about it. A group of children gathered and discussed this for a while. They decided that giraffes must not lie down because they are too tall. Giraffes must sleep with their head in the fork of a tree. Papert asked, "What if there are no trees?" The children replied, "Of course there are trees. Giraffes have long necks so they can eat from the trees!"

We have all read how children invent their own theories about the world, and we love the stories. In education, we feel morally obliged to put them right; but putting them right is putting them down. In addition, when we put children right, the idea of allowing them to find their own way (a la Nintendo) gets lost. This is an essential dilemma in education. We don't want to put them down, but we need to guide their thinking. There is no way around this problem within the current structure of schools.

Papert went home and investigated giraffes' sleeping habits. He looked up giraffe in the encyclopedia. He got sidetracked and ended up learning that all mammals have the same number of cervical bones. So, the giraffes' cervical bones must be huge. This helps explain how a giraffe keeps its head aloft.

Papert claimed that books such as encyclopedias gave him an *extended immediacy*. Jennifer was not yet able to extend her world with the written word. She was dependent on others for knowledge that she could not gain through immediate interaction with her physical environment.

What if Jennifer had a knowledge machine that consisted mainly of a big screen? What if she could conjure up giraffes on the screen and explore them? What if they could even come out as holograms and she could experience them as three dimensional objects?

Papert stated that such machines exist, and it is only a matter of time before they are available to children. The amount of time before children have them depends more on social and economic factors than on available technology. Papert suggested, "If you don't like the idea, you can learn a lot from the ostrich."

The machines' existence will challenge one of the foundations of education: the nature and role of literacy in learning. Presently, nonreaders are dependent on others for information, but reading is going to lose its role as the primary way of learning.

In the past, literacy has been equated with what

Papert calls "letteracy"— traditional reading and writing skills. If Jennifer had this machine, she wouldn't have to achieve letteracy to have access to an unlimited supply of knowledge. Presently, even if she learned to read an encyclopedia, her supply of knowledge would be limited by the encyclopedia company. Papert recounted a story about visiting such a company where a fairly large division, called the space patrol, is devoted to searching for information that can be omitted—so that the next edition would only be as heavy as the last.

The widespread use of the knowledge machine is not necessarily a good thing. Papert warns that we need to consider the implications of the changes that would take place. He hopes that we have learned from our experiences with the environmental and social effects of the automobile—effects we did not anticipate.

This is not to say that change is not desirable—or avoidable. The world as kids see it is so much more dynamic than school. Kids won't continue to sit still for it. Change *will* happen.

Papert believes that if this machine existed, the panic about reading would go away. The opportunity to learn to read would still be there, but the pressure would be off. Papert predicted kids would read earlier because of the decreased emotional pressure.

Letteracy in math can be thought of as the ability to use symbols. There is an incredible vastness of knowledge in the pre-letterate stage, and there is very little connection of this knowledge to letterate math. When real teaching takes place, an attempt is made to make the connections to real life, but even then, the pre-letterate knowledge is too often considered to be subordinate. Concrete materials are only seen as a way to get to letterate math.

If Papert had to describe his career it would be as an explorer in pre-letterate mathematics for children. Computers offer explorations of pre-letterate mathematics for all ages. We can think of programming as a way into the concept of pre-letterate mathematics. When Papert began work on Logo, he thought that children needed an easy way to make the computer do something interesting for them.

His original idea was for children to use the robot turtle as a feedback device, for example, making the turtle go around a table by telling the turtle to turn until it almost hits the table and then turning away until it may be too far and then turning back, and so on. This idea never really caught on. The turtle as a drawing instrument did.

Why? There is continuity and connection with the kid's own world. Kids know about drawing. Turtle graphics is connected to what they are already doing. Also, drawing is important to children. Making the turtle find a way around the table is not interesting to them, but drawing with Logo *is* interesting.

Then he got a new idea—Lego-Logo. There is a new connection to what is interesting to children—to what is their own.

Papert observed some Boston schoolchildren playing with Lego and computers. The boys started making trucks right away. The girls made a house. At first, they traded motors for things they could use to decorate their house. Then one day, there was a light in one of the rooms in the house. The Logo code was simple — **on wait 10 off wait 10.** Later there were several lights, then a lighted Christmas tree that turned around.

This was a soft transition. The girls found their own way without doing violence to their selves.

Another time, two girls made a mother cat and her kitten. The kitten called its mother with a flashing light. To make the mother cat go to the kitten, the girls put a light sensor on each side of her head. They each sensed light in a semicircle. The cat would turn in some vague direction toward the light, back and forth, back and forth, until it reached the kitten *exactly*. This vague knowledge wasn't an approximation; although vague, it was useful because it was interactive. The cat always got to the kitten.

Many people who aren't comfortable with precision and exactness can find a way to acquire knowledge. They can develop their own way of appropriating mathematical and technological knowledge. With Logo, fantasy, technology, mathematics, science, and personal ways of knowing can come together in natural connections rather than stay separate as specialized subjects.

Technology opens up assumptions about what we want kids to know. Old assumptions such as letteracy = literacy, math = precision and symbols, and science = laws must be re-examined.

Douglas H. Clements, associate professor at the State University of New York at Buffalo, has studied the use of Logo environments in developing children's creative, mathematics, metacognitive, problem-solving, and social abilities. He is currently working on an NSF-funded project to develop a full K-6 mathematics curriculum featuring Logo.

Julie S. Meredith is a mathematics education doctoral student at the State University of New York at Buffalo. She has taught secondary mathematics and computer science, gifted math at the middle school level, and mathematics methods courses.

Douglas H. Clements and Julie Meredith
State University of New York at Buffalo
Department of Learning and Instruction
593 Baldy Hall
Buffalo, NY 14260

CIS: 76136,2027 BITNET: CLEMENTS@UBVMS

On Turtles and Cycloids: To SIN or Not to SIN

by Daniel Melman

This article demonstrates an important curriculum process. Melman presents a progression of Logo procedures exploring the creation of a particular geometric shape without resorting to the "advanced" features of his Logo dialect. Now, while such efforts are extraneous in product-oriented projects, in learning activities where process is the central concern, this technique pushes students to deepen their understanding of Logo, of mathematics, and of the procedure whereby the two are joined to accomplish some task. (MH)

At the Fourth International Conference for Logo and Mathematics Education (LME4) held in Israel in July, 1989, Uzi Armon[1] presented an interesting paper on the "turtle Representation of Trochoid Curves". These are plane curves that describe the path of a fixed point on a circle that rolls with no slipping outside or inside another circle.

Armon published his article "The Investigation of the Geometry of Cycloids through Logo Programming" in the magazine *Misparim*.[2] His Logo procedures for drawing the cycloids are based on the use of the SINE function:[3]

```
TO CYC :R
CYCLOID 1 1 (PI / 45) * :R
END

TO CYCLOID :TOTAL :ANGLE :SIDE
IF (:TOTAL > 180) [STOP]
RIGHT :ANGLE
FORWARD :SIDE * SIN :TOTAL
CYCLOID (:TOTAL + :ANGLE) :ANGLE
   :SIDE
END
```

Unfortunately my turtles do not like trigonometric functions, and it was clear to me that it would be difficult for them to understand the use, or reason, for SINE as input to FORWARD. This being the case, I was forced to be content with the basic turtle geometry to draw the cycloids. To carry out this mission, I relied on Armon's own ideas in LME4. In that paper ("Duopoly as a turtle-Representation of Trochoid Curves"), I found the following:

> We have to look at an object which is attached to one point of a rotating circle, where the circle is also participating in another circular movement. Thus, the object takes part in two circular movements simultaneously.

Let's try to understand one unit of that object's orbit. First, it walks a small step along its circle, and then immediately along the other circle. In the article *Misparim*, Armon writes:

> The formal definition of a cycloid is, that this curve describes the path made by a fixed point on a circle which rolls without slipping on a straight line. The point may be seen as a turtle holding a pen and when the circle rolls the turtle draws the resulting curve.
> The cycloidal movement of the turtle might be decomposed into two movements of equal speed. One—a circular movement of the turtle around the center of the circle, and the second—a linear movement of the center of the circle rolling along the straight line.

In order to get the intrinsic representation of a cycloid in Logo without the use of SINE, we can combine all the quoted ideas and suggest the following:

We must look at one object attached to a point on a rolling circle. At the same time, the circle moves forward along a straight line. Thus, the object is involved in two different movements simultaneously.

We will try to understand one "unit" of the orbit of this object. First, it steps one small step along the circle, and immediately following this, one small step in the linear direction along a straight line parallel to that on which the circle rolls.

Translating this idea into Logo gives us the following:

```
TO CYCLOID :ANG
LEFT 90
   (To start drawing along the X-
    axis)
CYC :ANG
END

TO CYC :ANG
LOCAL "DIR
RIGHT :ANG
FORWARD 1
   (One small step around the circle)
MAKE "DIR HEADING
   (Saving the direction of the
    turtle)
SETH 90
FORWARD 1
   (One small step along the X-axis)
SETH :DIR
   (Returning to the original direc
    tion)
CYC :ANG
END
```

It may be claimed that this version is not a "pure" intrinsic description of the cycloid because of the use of HEADING and SETH, even if it is done only to save values.

In the next version there is no reference to the Cartesian coordinates, and thus no extrinsic elements. An extra variable has been added for the size of the step, and also a stop statement for ending the loop after completing one branch of the cycloid (one complete turn of the circle).

```
TO CYCLOID :ANG :STEP
LEFT 90
CYC :ANG 0 :STEP
END

TO CYC :ANG :TOT :STEP
IF :TOT > (360 - :ANG) [STOP]
RIGHT :ANG
FORWARD :STEP
RIGHT (180 - (:TOT + :ANG))
   (Horizontal direction to the
   right)
FORWARD :STEP
LEFT (180 - (:TOT + :ANG))
   (Return to the original direction)
CYC :ANG (:TOT + :ANG) :STEP
END
```

This procedure for drawing a cycloid might be expanded for a circle of any given radius:

```
TO RCYCLOID :RAD :ANG
LEFT 90
CYC :ANG 0 ((PI * :RAD * :ANG) / 180)
   (The value for "STEP" according to
   the given radius)
   END
```

The circle may roll along any other line, like the horizontal one. This is only done to receive a known recognizable cycloid. By altering the input 90 to LEFT in the procedure CYCLOID, we may receive a rotated cycloid.

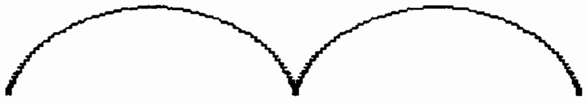
As it would be expected, the cycloid drawn by the procedure CYCLOID is congruent to that received from the procedure Armon suggested for the same radius.

Until now the turtle's steps on the circle and in parallel to the straight line are of the same size. This is the way regular cycloids are generated according to the previous explanation. It would now be interesting to investigate the effect of unequal steps.

To this end we will define a general procedure with a new variable to determine the number of cycles for the cycloid (NC):

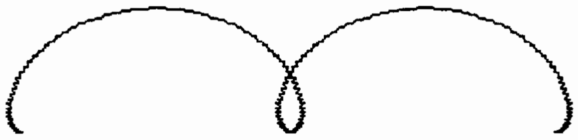
```
TO GCYCLOID :ANG :CSTEP :LSTEP :NC
   (:CSTEP-step on the circle)
LEFT 90
GCYC :ANG 0 :CSTEP :LSTEP :NC
   (:LSTEP-step parallel to the
   straight line)
END

TO GCYC :ANG :TOT :CSTEP :LSTEP :NC
IF :TOT > (360 * :NC - :ANG) [STOP]
RIGHT :ANG
FORWARD :CSTEP
RT (180 - (:TOT + :ANG))
FORWARD :LSTEP
LEFT (180 - (:TOT + :ANG))
GCYC :ANG (:TOT + :ANG) :CSTEP
   :LSTEP :NC
END
```

When :CSTEP = :LSTEP, we will receive the regular cycloid, which describes the path of a fixed point on the circle where its distance from the center is equal to the radius:



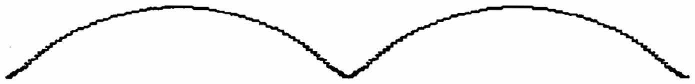
```
GCYCLOID 5 2 2 2
```

When :CSTEP > :LSTEP, the way traveled parallel to the straight line will be shorter than that around the circle. A short cycloid will be received—a "shrunken" cycloid with a loop:



```
GCYCLOID 5 3 2 2
```

When :CSTEP < :LSTEP, the length of the way caused by :LSTEP will be longer than that caused by :CSTEP, and an elongated cycloid is received:



```
GCYCLOID 5 2 3 2
```

It would also be interesting to try either :CSTEP = O or :LSTEP = O, but this is left to the reader.

Thus, what is the connection between the short or the elongated cycloids and the circle which generates them?

It is quite reasonable that the short cycloid is generated when the distance of the fixed point to the center of the circle is shorter than the radius. An elongated cycloid is generated when the distance is longer than the radius.

To check this assumption, we will include a new procedure. The value of :CSTEP will be calculated as usual according to the radius of the circle, and :LSTEP will be determined by the value of :DEV as a positive or negative addition to the radius of the circle:

```
TO C-CYCLOID :RAD :ANG :DEV :NC
G-CYC :ANG 0 (STEP :RAD)(STEP :RAD +
   :DEV) :NC
END

TO STEP :RAD
OUTPUT ((PI * :RAD * :ANG) / 180)
END
```

Now it might be seen that when :DEV = O, a regular cycloid is received, for :DEV < O one receives a shortened cycloid, and for :DEV > O one receives an elongated cycloid.

Footnotes

1. Uzi Armon, The Israeli Logo Center, Technion (Israel Institute of Technology), Haifa, Israel.
2. A magazine for mathematics teachers, published by the Department for Science Teaching, Weizman Institute for Science, Rehovot, Israel.
3. These procedures are written with Terrapin™ Logo for the Macintosh.™

Daniel teaches about the integration of computers in education to preservice and inservice teachers at the State Teacher's college Seminar Hakibutzim. In addition, he is in charge of a team of teachers in a Regional Primayr School. His main areas of interest are Logo and Lego/Logo. He has been working with Lege/Logo with groups of students for more than three years.

Daniel Melman
Nir Yitzchak
Doar Na Haneqn 85455
Israel

Global Logo Comments

by Dennis Harper

Logo Exchange Continental Editors

Africa	Asia	Australia	Europe	Latin America
Fatimata Seye Sylia	Marie Tada	Anne McDougall	Harry Pinxteren	Jose Valente
UNESCO/BREDA	St. Mary's Int. Sch.	Monash Univ.	Logo Centrum Nederland	NIED
BP 3311 Dakar	6-19 Seta 1-Chome	6 Riverside Dr.	P.O. Box 1408	UNICAMP
Senegal, West Africa	Setagaya-Ku	East Kew 3120	BK Nijmegen 6501	13082 Campinas
	Tokyo 158, Japan	Victoria, Australia	Netherlands	Sao Paulo, Brazil

Logo '86, the West Coast Logo Conference, and Logo in Paradise seem like events of long ago. Although Logo conferences have thinned considerably in the United States, EuroLogo and the National Logo Congress of Brazil continue to attract educators abroad. Following is José Valente's report on the recent Brazilian congress.

The III National Logo Congress in Brazil by José Armando Valente

The III National Congress took place at the Universidade Católica de Petrópolis in Petrópolis, a picturesque city nestled in the mountains outside of Rio de Janeiro, Brazil. Petrópolis was the former summer retreat of the royal family and relics of the 19th century permeate the modern-day city. The congress was held from September 15 to 19, 1992. Approximately 300 participants attended from different states in Brazil, Argentina, Uruguay, and Costa Rica.

The main theme of the congress was "Logo, Technology, Consciousness, and Creativity," and it was structured in terms of main talks and paper presentations. There were several contributions from international researchers, such as Antonio Battro, from Argentina; Miguel Rodé, from Uruguay; Douglas Clements, from the United States; and Jean Retschitzki and Jean Claude Bres, both from Switzerland. There were 58 papers, 12 about Logo and special education, 10 on special topics (Lego-Logo, Logo-music, and 3D-Logo), 9 about development of Logo systems, 9 about research on cognitive development in the Logo environment, 16 about Logo teacher training, and 30 about use of Logo in the classroom.

Even though the majority of the papers was about the use of Logo, several papers introduced other topics that were not present in previous Logo congresses. These included the development of Logo systems for PC computers, robotics, 3D-Logo and Logo music, and papers about the cognitive aspects of Logo (especially the ways students cope with recursion, concepts of angle, and ways Logo can help in the acquisition of reading and writing skills).

The other positive aspect of this conference was the keynote speakers. Professor Heraldo Cidade, from the Ministry of Education, gave the opening talk, which described how the computer could help the creation of a new era and new directions in education — this is a dream pursued by many educators in the past. His point was that appropriate use of the computer gives us the seeds to make this dream come true.

Professors Antonio Battro and Lea Fagundes talked about the expansion of the school and the students' consciousness through telecommunications. They introduced this technology in the learning environment preserving the Logo aesthetic. They showed that people can learn from each other and have another dimension of the world they live in by interacting through telecommunications.

Metacognition was a topic mentioned in several talks. Professor Jean Retschizki talked about how learning recursion can help students to acquire metacognitive capabilities. Professor Haimuth Kruger presented a theoretical description of metacognition, illustrating his main points with examples from Logo activities. Professor Douglas Clements talked about the main results of his research on Logo and the development of metacognitive and creative problem-solving skills.

The use of intuition, imagination, and creativity in Logo activities was another very interesting topic.

Professor Maria Candida Albuquerque Limas talked about a school of the future in which the main concern should be the development of the students' consciousness and creativity. Professor Wanda Macedo talked about the use of Logo to study the imaginative archetypes according to Duran's theory. Professor Silvia Bustamante showed how Logo activities break the traditional discipline barriers and create opportunities for the students to use their intuition and creativity skills.

Other talks introduced related Logo topics. These included a beautiful presentation about fractals by Dr. Miguel Rodé; and talks on robotics and the description of the Pangée Project, by Professor Jean Brés; the relationship between Artificial Intelligence and Logo, by José Valente; the use of Logo to help rehabilitation programs for different types of disabled populations, by Professor Andrea Kopel; and two presentations of multimedia, organized by Carlos de Souza and by Heitor Quintela from IBM.

Comparing the third to the second National Logo Congress, which was held in July, 1988, it is possible to identify a major development in the concerns of the Logo community. In the second congress, the paper presentations and the main talks were about the "how to" aspects of Logo. The third congress also featured some of these talks; however, the keynote speeches and the discussions within members of the Brazilian Logo community showed a much deeper level of concern. They were about the Logo aesthetic rather than the "how to" aspects. This shift shows serious and gradually evolving work.

The proceedings of the Congress can be obtained from:

Universidade Católica de Petrópolis
Secretaria de Desenvolvimento
Rua Benjamin Constant 213
25610 - Petrópolis - RJ BRAZIL

Dennis Harper
Olympia School District
1113 Legion Way S. E.
Olympia, WA 98501
206-753-8835

LCSI Introduces *Microworlds*™ Products

MicroWorlds Language Art™ encourages students to explore words, text, form, and the images these words and text evoke. This package makes it possible to write text in any shape, style, color, or direction. Projects include Visual Poetry, Advertising, Haiku, Cinquain, plus more.

MicroWorlds Math Links™ supports many of the Math Standards published by the NCTM by linking math to different subjects in the curriculum. With *MicroWorlds Math Links*™, students *use* math, as opposed to just studying math, to develop projects related to science, art, and social studies. Projects range from creating kaleidoscopic images to examining textile patterns of Navaho Indians.

MicroWorlds Project Builder™ gives students the tools to build their own projects for any subject in the curriculum.

All three of these produces include a student project book, a teacher's manual, a how-to guide, on-disk project starters and samples, and on-line help.

The *MicroWorlds*™ program, the core of all three packages, includes all the best features of *LogoWriter*®. New features include drawing tools, an infinite number of turtles, a music and sound center, multiple text fonts, styles, and text colors, plus objects such as buttons and sliders. The new shapes center includes multicolored shapes, a shape editor, and tools to change the turtles' sizes. In addition, *MicroWorlds*™ lets students use parallel processing to create more realistic simulations and animations.

LCSI will continue to make their products available under their current site and network license policy. The MicroWorlds family of products is available for color Macintosh LC computers. System 7 is required. Demo versions for each product will be available at no charge.

For further information on Logo Computer Systems Inc. or its products, call the toll-free number: 1-800-321-LOGO.

ISTE Books & Courseware Order Form

Name ______________________________

School/Business ______________________________

Address ______________________________

City ____________ State ____________ Zip/Postal Code ____________

Country ____________ Phone ____________

☐ Yes! I'm presently an ISTE member. Membership number must be included to receive discount.
My membership number is ____________ (Discount does not apply to initial membership fee.)

Occupation

Grade Level: ☐ all (K-12) Specific grade level ____________

☐ Teacher (350) Subject ____________

☐ Administrator (320) Title/Position ____________

☐ Higher Education (360) Title/Position ____________

☐ Federal/State Government (310) Title/Position ____________

☐ Educational Information Resource Manager (IRM) (320)

☐ Technology Coordinator (340)

☐ Other ____________

Terms & Conditions

Terms: Net 60 orders must be accompanied by authorized company purchase orders. Orders for individuals must be pre-paid, C.O.D., or charged to a Visa, MasterCard, or Discover card number. All non-U.S. orders must be pre-paid.
Shipping: Shipping method is UPS. Add 5% in addition to shipping rate for first class delivery. ***Allow three weeks for delivery.*** Call for rush orders.
Prices: Subject to change without notice.
Returns: ISTE allows return of merchandise, provided the materials are in resellable condition. There will be a 10% restocking charge. All returns must be accompanied by an ISTE Return Authorization form. If you wish to return merchandise, please contact the Distribution Dept. *GST Registration #128828431*

Shipping & Handling for Books & Courseware

$0-$15.99 (subtotal)	add $3.25
$16-$45.99 (subtotal)	$4.50
$46-$75.99 (subtotal)	$5.50
$76-$100.99 (subtotal)	$6.50
$101 or more	7% of subtotal

Do not include *additional site license fees or subscription costs* when computing shipping rates.

Non-U.S. orders for Books & Courseware are sent surface mail. If you want your Books & Courseware order shipped AIRMAIL, please check here. ☐ ISTE will bill you the additional shipping charge.

Payment Options

☐ **Payment enclosed.** (Make checks out to ISTE.) Non-U.S. orders must be pre-paid with U.S. funds or credit card.
☐ VISA ☐ MasterCard ☐ Discover Card Exp. Date ______

☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

☐ **Purchase Order enclosed.** (Please add $2.50 for order processing. P.O. not including $2.50 fee will be returned.)

☐ **C.O.D.** for U.S. Book orders only. You will pay UPS the total upon delivery (check or cash—ISTE will add $2.75 order processing).

GST Registration Number 128828431

ISTE Books, Courseware, & Nonmember Subscriptions

Qnty.	Title	Member Unit Price	Nonmember Unit Price	Total Price

BOOKS & COURSEWARE SUBTOTAL	
Add Shipping, based on Books & Courseware SUBTOTAL	+
Add $10 S & H for Don't Copy that Floppy	+
Add Additional 5% of SUBTOTAL if shipped to PO Box, AK, HI, or outside U.S.	+
Add 7% of SUBTOTAL for GST if shipped to Canada	+
If billed with purchase order, add $2.50; If COD, add $2.75	+
	=
Membership Total from other side	+
TOTAL	=

MAIL this form to:
ISTE, 1787 Agate St., Eugene, OR 97403-1923 USA
FAX this form with credit card or P.O. information to: 503/346-5890
PHONE your credit card or P.O. order to:
ISTE Order Desk: 800/336-5191

Submission of Manuscripts

Logo Exchange is published quarterly by the International Society for Technology in Education Special Interest Group for Logo. Logo Exchange solicits articles on all aspects of Logo use in education. Articles appropriate to the International column should be submitted directly to Dennis Harper. Advanced articles should be submitted to Mark Horney, editor of the Extra for Experts column. Articles appropriate for the MathWorlds column should be sent directly to Sandy Dawson.

Manuscripts should be sent by surface mail on a 3.5" disk (where possible). Preferred format is Microsoft *Word* for the Macintosh. ASCII files in either Macintosh or DOS format are also welcome. Submissions may be made by electronic mail as well. Where possible, graphics should also be submitted electronically. Please include electronic copy with any paper submissions. Disk are preferred, but electronic mail is also welcome. Paper submissions will NOT be accepted.

Send surface mail to:
Sharon Yoder
170 Education, DLIL
University of Oregon
Eugene, OR 97403

Send electronic mail to:
Internet: YODER@oregon.uoregon.edu

Deadlines

In order to be considered for publication, manuscripts must be received by the dates indicated below.

Volume 12, Number 2	April 1, 1993
Volume 12, Number 3	July 1, 1993
Volume 12, Number 4	Oct 1, 1993

SIGLogo
ISTE
1787 Agate Street
Eugene, OR 97403-1923

Non-Profit
Organization
US Postage
PAID
Eugene, OR
Permit No. 63