

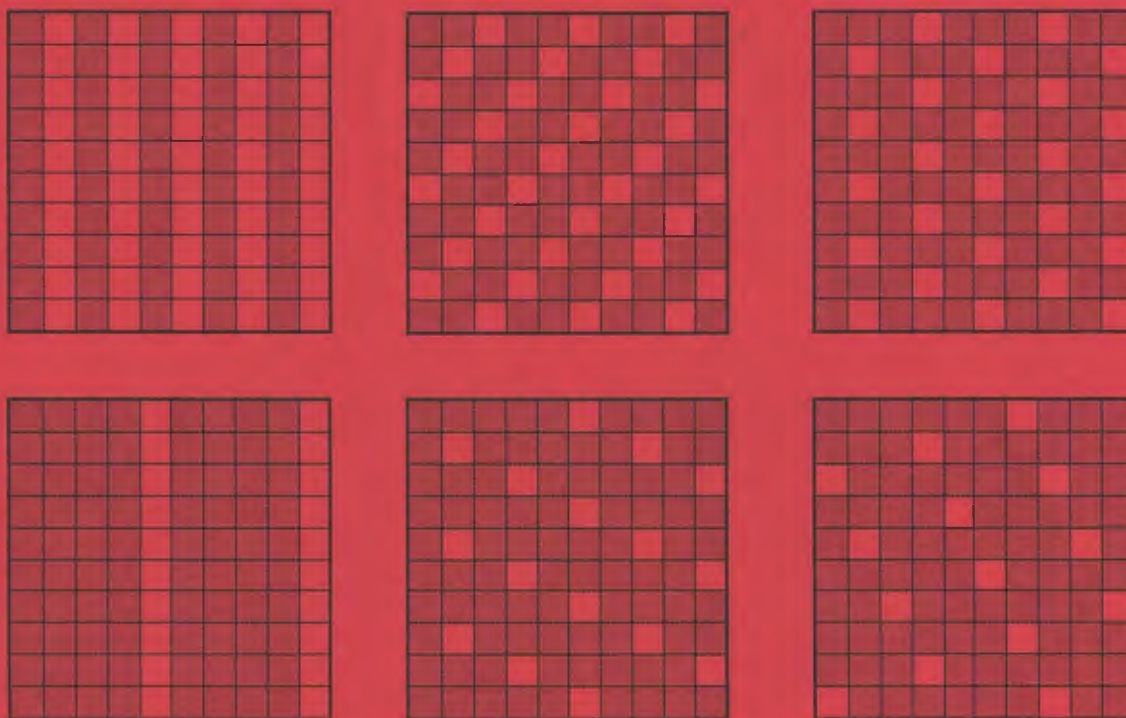


LOGO EXCHANGE

Volume 14 Number 3

Spring 1996

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



Fun with Hundred Board Patterns



In this issue:

**Exploring Recursion
Hundred Board Patterns
Logo and Spatial Abilities
A Circular Romp**

**Exploring Triangles
Logo Action Research
Parallel Processing
Logo and the Internet**

Editorial Publisher

International Society for Technology in
Education

Editor-in-Chief

Dorothy M. Fitch

Copy Editor

Jennifer Lindsey

Founding Editor

Tom Lough

International Editor

Dennis Harper

SIGLogo Officers

Chuck Friesen, President
Steve Sesko, Vice-President
Hope Chaffian, Secretary/Treasurer
Dorothy Fitch, Editor

Production

Kerry Lutz

Director of Advertising Services

Lynda Ferguson

Submission of Manuscripts

Logo Exchange is published quarterly by the International Society for Technology in Education Special Interest Group for Logo-Using Educators. *Logo Exchange* solicits articles on all aspects of Logo use in education.

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, either on disk (preferred) or by electronic mail, with any paper submissions. Paper submissions may be submitted for review if electronic copies are supplied upon acceptance.

Send surface mail to:

Dorothy M. Fitch
3 Derby Road
Derry, NH 03038

Send electronic mail to:

Internet: 71760.366@compuserve.com

Deadlines

To be considered for publication, manuscripts must be received by the dates indicated below.

Volume 15, Number 1	Mar. 1, 1996
Volume 15, Number 2	June 1, 1996
Volume 15, Number 3	Oct. 1, 1996
Volume 15, Number 4	Jan. 1, 1997

ISTE Board of Directors 1995-96

David Brittain, President
MGT. of America (FL)
David Moursund, Executive Officer
Dennis L. Bybee, Associate Executive Officer

Executive Board Members

Cornie Stoul, Secretary
Texas Education Network (TX)
Barry Pitsch, Treasurer
Heartland AEA #11 (IA)
M.G. (Peggy) Kelly, Past-President
California State University - San Marcos
Terrie Gray, At Large
Chico Unified School District (CA)

Board Members

Kathryn Edwards Price
Apple Computer, Inc., Santa Monica (CA)
Terry Killian
Jefferson County R1 School District (CO)
Paul O'Driscoll
Salem-Keizer Public Schools (OR)
Lynne Schrum
University of Georgia
Carla Schutte
Educational Technology Consultants (FL)
Gwen Solomon
U.S. Department of Education (DC)
Neal Strudler
University of Nevada - Las Vegas
Harriet Taylor
Louisiana State University
Brenda Williams
West Virginia Department of Education

Logo Exchange is published quarterly by the International Society for Technology in Education (ISTE), 1787 Agate Street, Eugene, OR 97403-1923, USA, 800/356-5191. This publication was produced using Aldus PageMaker®.

ISTE Members may join SIG/Logo for \$24.00. Dues include a subscription to *Logo Exchange*. Add \$10 for mailing outside the USA. Send membership dues to ISTE. Add \$4.00 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 director of advertising services for space availability and details.

Logo Exchange solicits articles on all topics of interest to Logo-using educators. Submission guidelines can be obtained by contacting the editor. Opinions expressed in this publication are those of the authors and do not necessarily represent or reflect the official policy of ISTE.

© 1995 ISTE. All articles are copyright of ISTE unless otherwise specified. Reprint permission for nonprofit educational use can be obtained for a nominal charge through the Copyright Clearance Center, 27 Congress St., Salem, MA 01970, 508/744-3350, FAX 508/741-2318. ISTE members may apply directly to the ISTE office for free reprint permission.

POSTMASTER: Send address changes to *Logo Exchange*, ISTE, 480 Churnelton St., Eugene, OR 97401-2626 USA. Second-class postage paid at Eugene OR. USPS# 660-130. ISTE is a nonprofit organization with its main offices housed at the University of Oregon. ISSN# 0888-6970

Contents

From the Editor:

Spring Threads *Dorothy M. Fitch* 2

Quarterly Quantum:

Turtles? What Turtles? *Tom Lough* 3

The Recurring Question of Recursion *Jim Muller* 4

Hundred Board Patterns *Ihor Charischak* 7

Using Logo as a "Building Block" to Improve

Spatial Abilities in Primary Grade Students *Nancy Flynn* 12

A Circular Romp with Logo Triumphant *Robert Macdonald* 15

Turtle and Coordinate Geometry in a Secondary Environment *Medhat H. Rahim* 22

Logo Action Research:

Getting Teachers Excited

About Technology *Nan Youngerman and Donna Cutler-Landsman* 27

Recursion, the Problem Solver *Jim Muller* 30

Windows on Logo:

Logo and the Internet—The Turtle and

the World Wide Web *Glen L. Bull and Gina L. Bull* 33

Logo: Search and Research

Parallel Programming *Douglas H. Clements and Julie Sarama* 36

From the Editor: Spring Threads

by Dorothy M. Fitch

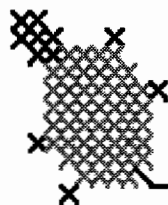
Three themes thread their way through this issue of *LX*: using Logo as a tool for exploring mathematical ideas, for investigating gender differences, and for approaching the latest technologies.

Exploring mathematics, one of the classic ways to use the power of Logo, is the main theme of this issue. Ihor Charischak takes us on a field trip to ancient Greece, where we use Logo to experiment with **times tables** and **factors**. Jim Muller helps remove the mystery from **recursion**. In his first article, he explains recursion with a great hands-on demonstration. His second article gives us a powerful example of putting recursion to work to solve a problem. Robert Macdonald shows how to get kids excited about **circles**. Dr. Rahim helps students explore **triangles** and learn what makes a triangle a triangle.

Also in this issue, we investigate gender differences in two different contexts. Nancy Flynn explores gender differences in spatial abilities using **building blocks** and Logo. Nan Youngerman and Donna Cutler-Landsman look at an action research project focused on gender differences using **robotics**.

To help us look toward the future, Glen and Gina Bull ponder the use of the **Internet** and **World Wide Web** as a distribution channel for Logo ideas, Doug Clements and Julie Sarama explore **parallel programming** and the new types of bugs we must learn to unravel, and Tom Lough ponders the future of **real live turtles** and tells how we can help their cause.

And speaking of turtles (and threads), I couldn't resist sharing this morsel with you.



I recently attempted teaching myself the art of counted cross-stitch. The first lesson in the pamphlet showed how to cross-stitch a turtle, so

naturally I took that as a good omen. But before taking a needle in hand, there I was at the computer, writing this little automatic cross-stitch program in Logo. For a challenge, modify it for other cross-stitch diagrams.

```
to cross.stitched.turtle
cleargraphics hideturtle
pu setx -50 sety 50 pd
; 0 is no stitch (an X with pen up)
; 1 is a light green X (my color # 11)
; 2 is a dark green X (my color # 12)
; B is a custom backstitch procedure
; starting from the upper left corner
row [2 2]
row [2 2 2 0 0 0 2]
row [0 2 2 1 1 1 1]
row [0 0 1 1 1 1 1 1]
row [0 0 1 1 1 1 1 1 1 2]
row [0 0 1 1 1 1 1 1 1]
row [0 0 1 1 1 1 1 1 1]
row [0 2 1 1 1 1 1 1 1]
row [0 0 0 1 1 1 1 1 1]
row [0 0 0 0 1 1 1 1 B]
row [0 0 0 2]
end

to row :list
if empty? :list [pu setx -50 sety ycor - 8
pd stop]
if first :list = 0 [pu x pd]
if first :list = 1 [setpc 11 x]
if first :list = 2 [setpc 12 x]
if first :list = "B [setpc 12 bkstitch]
row butfirst :list
end

to x
right 45 forward 10 back 5
left 90 forward 5 back 10 right 45
end

to bkstitch
pu right 90 forward 7 left 90 pd
left 45 forward 10 back 10 right 45
right 90 forward 7 left 90
end
```

Happy Logo adventures! ▲

Dorothy M. Fitch, *LX* Editor
3 Derby Road, Derry, NH 03038
Telephone: 603-425-2010 Fax: 603-425-6487
E-mail: 71760.366@compuserve.com

Quarterly Quantum: Turtles? What Turtles?

by Tom Lough

Kim: Knock, knock.

Pat: Who's there?

Kim: Leonardo, Donatello, Michelangelo, and Raphael

Pat: Leonardo, Donatello, Michelangelo, and Raphael who?

Kim: How quickly we forget.

The Teenage Mutant Ninja Turtles are history. Gone. Poof.

I have been thinking about turtles recently. No real surprise there. Perhaps because the TMNTs have faded so quickly I am more attuned to turtles in distress. Consider these items.

In a recent issue of the *Hartford Courant*, there appeared at the top of the front page an article about Joseph Ferrara of Wallingford, CT. Joseph had learned that a 14-year-old boy had caught two huge snapping turtles, each weighing about 90 pounds and estimated to be between 75 and 100 years old. The boy and his father had declared their intention to make turtle soup of their captives. Joseph tracked them down and persuaded them to sell the turtles to him for \$200. Joseph then released the turtles back into the wild at an undisclosed location.

Not long after that, I noticed a turtle in the road as I drove to work one morning. By the time I could turn around and drive back, another car had stopped by the turtle. A man got out of the car, picked up the turtle, and placed it down gently in a marshy area a safe distance from the road.

This motivated me to call my friend Carole Allen, founder and chairperson of Help Endangered Animals—Ridley Turtles (HEART). For years, Carole has focused her efforts on helping the Kemp's Ridley sea turtle avoid extinction. She has organized public awareness campaigns for the Kemp's Ridley and has developed educational material for school use.

While Carole was happy to hear my two stories of how turtles have been saved, she is

more concerned with the thousands of sea turtles that wash up on U.S. beaches each year. Many of these turtles had been needlessly drowned when they became entangled in shrimping nets. Carole helps to educate shrimpers about how they can use Turtle Exclusion Devices (TEDs) in their nets to give sea turtles a means of escape. Although thousands of TEDs are now in use, sea turtles are still endangered.

How can we help?

Carole's most recent initiative is a Turtle-Safe Shrimp Campaign, in which shrimpers who agree to use TEDs properly will have their catch certified as Turtle-Safe. Concerned consumers are encouraged to ask their retail grocers and restaurant managers if they are using Turtle-Safe shrimp. Such a campaign can raise public awareness about the plight of the sea turtles and also encourage shrimpers, retailers, and consumers alike to become part of the solution.

Our Logo turtles have been around a long time. We have used them in both subject-specific and interdisciplinary activities. Here is an opportunity for our students to become more aware of the living, breathing variety of turtle and perhaps help to save a turtle or two.

For more information, write to Carole Allen, HEART, P.O. Box 681231, Houston, TX 77268-1231. She would appreciate a small contribution to help with the return postage if possible. You can also contact her through her online address: heartallen@aol.com. ▲

FD 100!

Tom Lough
Founding Editor
P.O. Box 394
Simsbury, CT 06070

P.S. Do you have a story of a turtle being saved? I would love to hear about it!



The Recurring Question of Recursion

by Jim Muller, The Logo Forum

Every Logo fan(atic) knows about recursion—where a procedure calls itself as an element of the solution. At least you would think so.

Beginning Logophiles often know about recursion. They can talk about it among themselves. They can point to lots of examples. But can they explain it to others? More importantly, can you explain it to a young child?

Here's a fun procedure to help you make some sense out of recursion, even though it really isn't Logo.

```
to get.through.life
get.through.today
get.through.life
end
```

That's cute. But think about it for a moment. The procedure says that to get through life, you have to get through today. Once you are through today, you have to move on with your life.

But where?

You cannot go backward. You cannot stop time. You have to get through life. But to get through life, you have to get through today. But each day is different. Wouldn't it be nice if you could save each day so that at the end, you had a record of your life. All you have to do is add a line to the life procedure.

```
to get.through.life
get.through.today
get.through.life
print book.of.life
end
```

Now, as you go through **get.through.life**, a page is written. When the procedure stops, it will print your book of life, starting from the most recent page back to the first.

To help make sense out of this, let's use a real example.

```
to test.recursion :n
print [Is this recursion?]
if readword = "YES [test.recursion :n + 1]
print :n
end
```

[Note: If your version of Logo doesn't include **readword**, use **first readlist** or **first request**.]

In this procedure, the variable **n** is used as a counter. It is used to keep track of your answers to the question, Is this recursion?

```
test.recursion 1
```

When you start the procedure, the first thing you see on the screen is the question,

Is this recursion?

Readword tells Logo to stop and wait for you to type an answer. Then comes the test. If the word you typed was YES, the procedure calls itself and the question is posed again.

Is this recursion?

Type YES a few times when you see the question and then type NO. When you type NO, the procedure comes to the test. This time the word you typed doesn't match the Logo word "YES" and so Logo reads the next line,

```
print :n
```

The result demonstrates what makes recursion different from just a simple loop.

When you first look at this procedure, it seems as if it were going to go around in a loop. Every time it passes the **test.recursion :n + 1** line, the counter is going to add 1. Then, when you type NO instead of YES, you'd think the procedure would simply print the current value of **n**.

Now the question becomes how to demonstrate this so that it makes sense.

How recursion works, hands-on!

You'll need some blank paper, a pencil, and scissors for this demonstration.

Fold the paper in half. Then fold it in half again, and again, then one last time. Crease the edges, then open up your piece of paper. Cut the paper along the folds. You should end up with 16 small pieces of paper. Now number these "pages" from 1 to 16 by writing a small number at the bottom of the page.

Get your pencil ready and type:

```
test.recursion 1
```

First, you see the question: Is this recursion? So write a big 1 on your first piece of paper and put that piece off by itself.

Type YES. The question appears on the screen again and **n** becomes **n + 1**, or 2. Write 2 on your second piece of paper and put that on the pile with your first page, the one with the 1 on it. Type YES again. Write 3 on the next piece of paper and put that piece on the number pile. Do this three more times, writing the new number for **n** each time. Put each piece of paper on the top of your growing pile of papers.

Now, when you type NO, what happens on the screen? You see a list of numbers counting backward from 7 to 1.

Why? Look at the screen. There are seven answers displayed there. You typed YES six times and NO once. You typed seven answers.

You have two stacks of paper now. You should have some blank pages left over in one stack, pages 8 to 16. The other stack has the pages you numbered from 1 to 7.

The stacks of paper are like the computer's memory. Each time you answered YES, Logo filed the current procedure in memory. The current procedure carried the current value of **n**, referred to in the procedure as **n**. When you answered NO, Logo went to the memory stack and asked for all the values of **n** and displayed them on the screen. In doing so, Logo emptied the memory stack.

This is like putting your pages back in order.

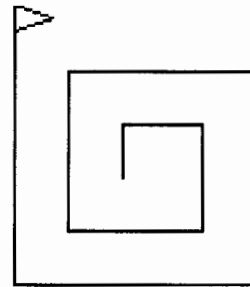
To demonstrate this, erase the 7 from page 7 and put it on top of page 8. Put page 6 on top of page 7, page 5 on top of page 6, and so on until you have all the pages back in a single stack again, erasing the large number each time.

Tail-end versus embedded recursion

Another way to demonstrate recursion is to use the following two procedures. The first uses tail-end recursion, where the recursive call is at the end of the procedure. The second procedure does much the same thing, only the recursive call is embedded in the procedure.

The **maze** procedure definitely looks like a simple loop that stops when the value of **n**, or **n**, is greater than 100. But is it?

```
to maze :n
  if :n > 100 [stop]
  forward :n right 90
  maze :n + 10
end
```



When you type **maze 20**, the turtle moves **forward 20 right 90** and then the procedure calls itself. Only **n** is now 30. The procedure continues with **n** adding 10 each time until it becomes greater than 100. Then it stops.

Now take a look at an example of "embedded" recursion. Will the **amaze** procedure below produce the same picture as the **maze** procedure, or will it be different? Try to picture what will happen before you run it.

```
to amaze :n
  if :n > 200 [stop]
  amaze :n + 10
  forward :n right 90
end
```

Think about how recursion works, about how Logo reads and acts on procedures. Start with **n**



equal to 20. This is how Logo reads the procedure the first time.

```
to amaze 20
if 20 > 100 [stop]
; Since 20 is smaller than 100, Logo goes to
the next line.
amaze 20 + 10
; The next line is written to memory as
"amaze 30" is called.
forward 20 right 90
end
```

Next, you have:

```
to amaze 30
if 30 > 100 [stop]
; Since 30 is smaller than 100, Logo goes to
the next line.
amaze 30 + 10
; The next line is written to memory as
"amaze 40" is called.
forward 30 right 90
end
```

Each time Logo runs the procedure, it never gets the chance to run the last line. So it puts each last line on a "page" of the memory stack. It will keep going, adding pages for each line it did not run from **amaze 40** through **amaze 100**. Then it stops.

Then Logo reads and executes the memory stack: forward 100 right 90, then forward 90 right 90, forward 80 right 90, and back to where it started, forward 20 right 90.

Are the pictures produced by **maze** and **amaze** the same? The end results look the same. The methods demonstrate what makes tail-end recursion different from embedded recursion.

Before we leave the discussion of tail-end versus embedded recursion, look at the **maze** procedure again.

```
to maze :n
if :n > 100 [stop]
forward :n right 90
maze :n + 10
end
```

What will happen if you change the second line to read:

```
if :n > 100 [print :n stop]
```

What will be printed at the end? Why?

This procedure does not result in a long list of numbers being printed. As it is run, every step is executed. There is nothing left over to hold in memory. At the end, Logo will print 110 and stop. Why 110? Only when **n** is greater than 100 does the procedure stop.

Putting recursion to work

Fractals were once thought to be math monsters. No one could figure out what to do with them. But thanks to computers, we now know that these recursive monsters help make beautiful computer graphics.

Here is a very short procedure. It certainly looks simple enough. But how can it produce such a complex graphic? Type **c 5 10** and take a look.

```
to c :size :level
if :level = 0 [forward :size stop]
c :size :level - 1 right 90
c :size :level - 1 left 90
end
```

The **c** curve is fun to watch. But how does it do what it does?

Start off with **c 5 0**. Because **:level** equals 0, the turtle moves forward 5. Once I get beyond this, I need a pad of paper to keep track of what is happening. Those ubiquitous sticky-back notes on the board are a big help. By the time you get to **c 5 3** or **c 5 4**, you will have a lot of notes on the board—and a pretty good idea of how recursion works.

Special thanks to Dr. Kathleen Martin, co-author of *The Turtle's Sourcebook*, for her stack-of-paper demonstration. It made that proverbial light go on so I could begin to see what recursion was all about. ▲

Jim Muller has had a lifelong interest in translating various technologies into understandable and persuasive programs. In 1981, Muller and his son organized the first Logo users group, the Young Peoples' Logo Association, which eventually grew into a worldwide 6,000-member organization. Among other things, the YPLA published *1,2,3 My Computer and Me* and *The Turtle's Sourcebook*. Both of these books are now being expanded for use with current versions of Logo. In 1985, the YPLA merged with CompuServe, where it became The Logo Forum. Today, Muller is a computer training and marketing consultant in the Dallas/Fort Worth metroplex. You can reach him via e-mail at 76703.3005@compuserve.com or on CompuServe at 76703,3005.

Hundred Board Patterns

by Ihor Charischak

[Editor's note: This article is reprinted with permission from CLIME (Council for Logo & Technology in Math Education). It was originally published in 1990 in *Clime Microworlds: Volume II*, a collection of Logo-based math activities. The Logo program provided here is a revision of the original.]

Children enjoy games of "Let's Pretend." Here is an activity that involves children in a historical fantasy.

One of the ways students can have a meaningful learning experience is to go on a math field trip to a place many of the ideas that we study in math were first discussed. That place is Ancient Greece. Now that would be quite a trick, given that Ancient Greece hasn't been around for over 2,000 years. So how can you possibly go back in time? Why, you would take the trip in a time machine!

To get students ready for this adventure, you might start by distributing to your class the following letter:

Dear student,

You are about to embark on a journey back in time. Your destination will be Ancient Greece. When you arrive, you will join an affluent merchant's family as an adopted son or daughter. Being rich in Greece means you don't have to do any back-breaking work that would take up all your time. So you can spend your idle hours on contemplation and study, which, of course, you do not have to do. But before you plan to head for the beaches with your ice cream and candy in hand, you must remember one thing: When in Greece, you will do as the affluent Greeks do, which is to pursue intellectual challenges of the mind! You do have one advantage over the Greeks. You have your computer with you. Your guide will be Eratosthenes. You can call him Beta for short.

Regards,
Your teacher

Next have your students take home and have signed a parent permission form.

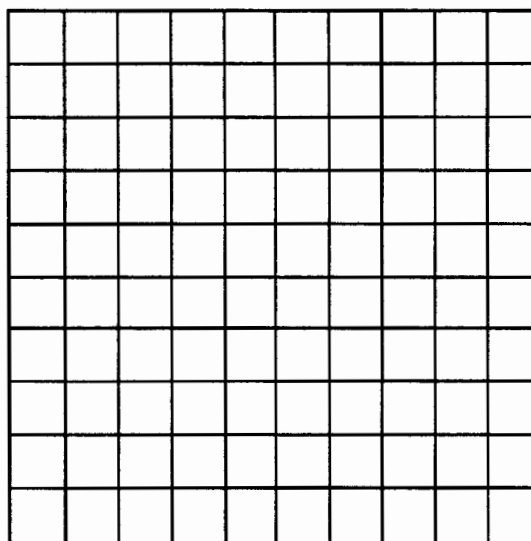
Dear school,

I hereby give permission for my son/daughter
to travel to Ancient Greece.
I waive all rights to sue the school if my son/
daughter gets lost in a time warp.

Sincerely,
A brave parent

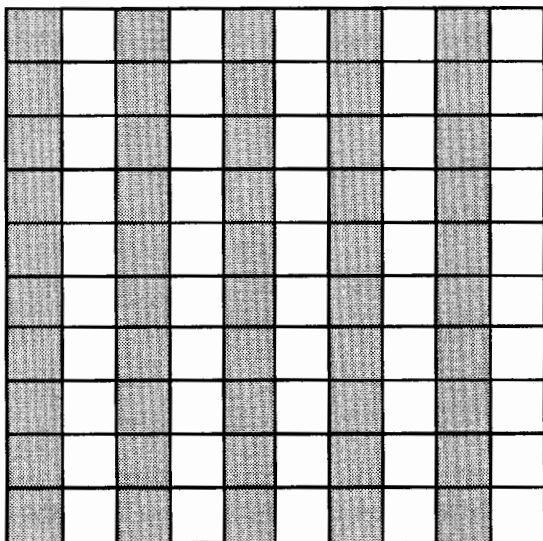
So having agreed to the mission and permission granted, the students climb into the time capsule and take off. The machine is fast, and before too long you arrive at your destination (200 BC). There standing in front of the machine waiting to greet you is Beta. He takes everyone into his house and begins.

"Hi there! Beta's my name and math is my game. Let's start off with something that's fun and easy. We are going to explore patterns on the hundred boards."



A 100 Board

Beta then passes out hundred boards made out of wood to all his guests. He next asks a question, "Do you know what numbers belong to the 2 times table or club?"



The 2 Times Table

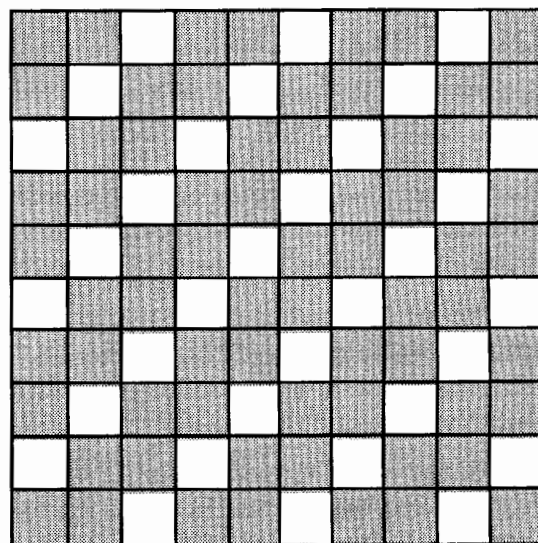
"Sure," says one of the students. "Those are all the even numbers."

"Yes, that's it! Now use a cutting tool to cut out the even squares. When you finish you will have a family portrait of the two club."

Now Beta was noticing something odd. Some of the students brought with them a strange machine. "What is that?" he asked.

"Why, it's a computer and I can do this hundred board activity with it. Watch," responded a student.

The student calls up 100BOARD on the disk, then types **grid** followed by **board 3** to create the 3 times table. Beta was intrigued. He grabbed the computer from the student.



The 3 Times Table

"I'm going to make the 3 times table," he said. Beta and his guests continued until they had completed the 2, 3, 4, and 5 tables.

"I suppose you want us to make the 6 times table as well," asked a student.

"Yes, but you don't have to make one. You can create it from the boards you already have," Beta responded.

He then placed his wooden 2 board on top of the 3 board and held them up to the light. Sure enough, only the 6s emerged.

A student then tried the same experiment on the computer. He cleared the screen and typed **board 2** followed by **board 3**. As expected, the 6s appeared.

Beta continues, "Can you tell me what will happen if you put a 3 board on top of a 5 board?"

"That's easy," said a student, "it's the 15 times table."

"Good. You think you've got the pattern, don't you? Well, what about the 3 table on top of the 6 table?"

The students were surprised to find out that it was *not* the 18 table! It turned out to be the 6 table. They scratched their heads and went back to the drawing board to try and figure out why.

Can you help them? Make the "boards" on the computer and see if you can predict what hap-

pens. Extend your guessing to three boards. For example, what happens if you put the 2s on top of the 3s on top of the 8s? What boards would you use to make the 12 times board or the 18 times board?

The secret of the 100 boards revealed...

You may have already figured out that the specific times table that results is the least common multiple of the tables that created it. So if you should place board 4 on top of board 6 and board 8, you should see the 24 times table.

Creating a 100Board program

Presented here are sets of procedures for two different methods of creating times tables. (These procedures were written using Terrapin's *Logo PLUS for the Macintosh*, but should work with most versions of Logo with little modification.)

Using Method A, students will determine which boxes should be erased to create a times table. Method B is an automated process that doesn't require students to generate the times tables themselves. Choose whichever method you wish to use with your students, depending on your goals and available time (Method B is handy when time is limited).

Method A

After entering the following procedures, type **grid** to display the 100 x 100 grid. Then show your students how to edit the **setup** procedure to make one that will create the times table for any number, as explained below.

```
to grid
penup setpos [-100 -100] pendown
repeat 4 [forward 200 right 90]
repeat 10 [forward 200 back 200 right 90
  forward 20 left 90]
penup setpos [-100 -100] pendown
right 90
repeat 10 [forward 200 back 200 left 90
  forward 20 right 90]
penup setpos [-100 -100] pendown
left 90
penup setpos [-100 80] pendown
end
```

```
to do :list
if empty? :list [nextrow stop]
if first :list = "X [fillsq]
nextsq
do butfirst :list
end
```

```
to fillsq
; use the necessary commands for your
; Logo to fill a 20x20 square
; a quick Terrapin/Mac shortcut is:
; (stamprect 20 20 "true)
setpc blue ; or any color you choose
fillsh [repeat 4 [fd 20 rt 90]]
setpc black
end
```

```
to nextsq
penup
right 90 forward 20 left 90
pendown
end
```

```
to nextrow
penup
left 90 forward 200 right 90 back 20
pendown
end
```

After entering the "housekeeping" procedures listed above, show your students how to edit the following **setup** procedure to make a procedure for any times table.

The **setup** procedure calls the **do** procedure 10 times, with an input that specifies which boxes should be erased in each row in order to draw the correct table. An X stands for a filled square. An O stands for an open or empty square.

Give the students this starting **setup** procedure:

```
to setup
penup setpos [-100 80] pendown
; X for filled, O for open (empty) box
do [X X X X X X X X X X]
do [X X X X X X X X X X]
do [X X X X X X X X X X]
do [X X X X X X X X X X]
do [X X X X X X X X X X]
do [X X X X X X X X X X]
do [X X X X X X X X X X]
do [X X X X X X X X X X]
do [X X X X X X X X X X]
do [X X X X X X X X X X]
grid
end
```



To generate a 2 times table, students would change every other X to O, starting with the second X. The result is the **twos** procedure below. In each row, the even numbered Xs have been changed to Os. When the **do** procedure is run, the Xs will be drawn as filled squares. The Os will not be filled.

```
to twos
penup setpos [-100 80] pendown
; X for filled, 0 for open (empty) box
do [X 0 X 0 X 0 X 0 X 0]
do [X 0 X 0 X 0 X 0 X 0]
do [X 0 X 0 X 0 X 0 X 0]
do [X 0 X 0 X 0 X 0 X 0]
do [X 0 X 0 X 0 X 0 X 0]
do [X 0 X 0 X 0 X 0 X 0]
do [X 0 X 0 X 0 X 0 X 0]
do [X 0 X 0 X 0 X 0 X 0]
do [X 0 X 0 X 0 X 0 X 0]
do [X 0 X 0 X 0 X 0 X 0]
grid
end
```

Following is the **setup** procedure edited into a 3 times table procedure. By comparing it with the **twos** procedure, you can see how to develop additional procedures for other times tables.

```
to threes
penup setpos [-100 80] pendown
; X for filled, 0 for open (empty) box
do [X X 0 X X 0 X X 0 X]
do [X 0 X X 0 X X 0 X X]
do [0 X X 0 X X 0 X X 0]
do [X X 0 X X 0 X X 0 X]
do [X 0 X X 0 X X 0 X X]
do [0 X X 0 X X 0 X X 0]
do [X X 0 X X 0 X X 0 X]
do [X 0 X X 0 X X 0 X X]
do [0 X X 0 X X 0 X X 0]
do [X X 0 X X 0 X X 0 X]
grid
end
```

Some students may discover that they can use **repeat** in their times table procedures if a pattern of filled squares appears. The 10 **do** instructions in the **twos** procedure could be replaced with:

```
repeat 10 [do [X 0 X 0 X 0 X 0 X 0]]
```

To overlay tables, students start by entering **grid**. Then they can enter **twos**, **threes**, or the names of other times table procedures they have created.

Method B

In this method, the computer calculates which boxes should be filled. The students need only to give the desired times table number as input to **board**. First enter the **grid**, **fillsq**, **nextsq**, and **nextrow** procedures from Method A. Then enter the following additional procedures.

```
to board :num
make "list []
compile :num 1
penup setpos [-100 80] pendown
dolist :list
grid
end

to compile :num :counter
repeat :num - 1 [make "list lput "X :list
  make "counter :counter + 1 if :counter >
    100 [stop]]
make "list lput "0 :list
make "counter :counter + 1
if :counter > 100 [stop]
compile :num :counter
end

to dolist :list
if empty? :list [stop]
if first :list = "X [fillsq]
nextsq
if xcor > 100 [nextrow]
dolist butfirst :list
end
```

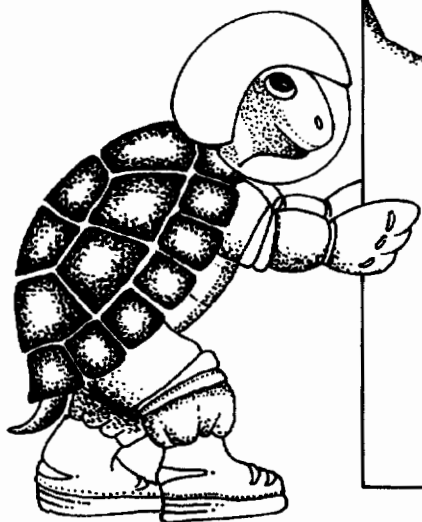
[Note: Some versions of Logo use **ifelse** when the True/False expression is followed by two action lists. Other versions use **then** and **else** before unbracketed action lists.]

Students can now generate any times table automatically by entering **grid**, then **board** followed by the times table they want to display. For example, the following instructions will create the 6 table, as described in the article.

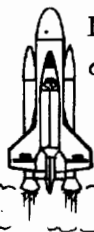
```
cleargraphics
grid
board 2
board 3
```

▲

In addition to founding CLIME in 1987 and supporting it ever since, **Ihor Charischak** is program manager for the Center for Improved Engineering and Science Education (CIESE) at Stevens Institute of Technology. For more information you can write to Ihor at 10 Bogert Avenue, White Plains, NY 10606 or send e-mail to: icharisc@stevens-tech.edu.



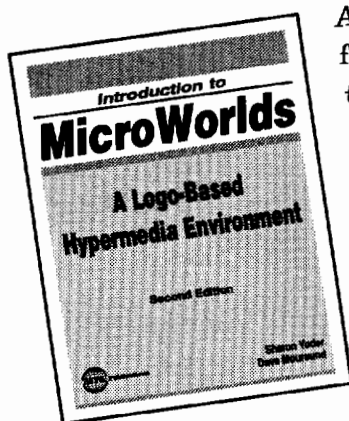
Push the envelope with MicroWorlds



Launch your students toward stimulating new classroom challenges and adventures. They can soar to a galaxy of new skills using MicroWorlds.

MicroWorlds is an exciting application that offers in a single program many of the features available in HyperCard, Kid Pix, and LogoWriter.

You're the pilot. Get ready now to grab the controls, blast off, and explore these exciting new worlds with your class! To help you prepare your crew for this fantastic voyage, Sharon Yoder and Dave Moursund back at mission control have developed a 222-page flight manual titled *Introduction to MicroWorlds—A Logo-Based Hypermedia Environment*.



As a computer application, MicroWorlds contains a wide range of features that make it easy and fun to work with color graphics, sound, text, and animation. As a programming environment, MicroWorlds includes a powerful and modern version of the Logo programming language.

It's time to broaden your universe. The countdown is underway. Have a nice trip!



International Society for Technology in Education
Customer Service Office
 480 Charnell Street, Eugene, OR 97401-2626 USA
 Phone: 800/336-5191 (US and Canada)
 Phone: 541/302-3777 (International)
 Fax: 541/302-3778 Fax-on-Demand: 800/891-0212
 America Online: ISTE CompuServe: 70014,2117
 Internet: iste@oregon.uoregon.edu
 World Wide Web: <http://isteonline.uoregon.edu>

HYPERCARD® IS A REGISTERED TRADEMARK OF CLARIS CORPORATION, SANTA CLARA, CA, U.S.A. KID PIX® IS A REGISTERED TRADEMARK OF BRØDERBUND SOFTWARE, NOVATO, CA, U.S.A. LOGO®, LOGOWRITER® AND MICROWORLDS® ARE REGISTERED TRADEMARKS OF LOGO COMPUTER SYSTEMS, INC., HIGHGATE SPRINGS, VT, U.S.A.



Using Logo as a “Building Block” to Improve Spatial Abilities in Primary Grade Students

by Nancy Flynn

“Can we do *Kid Pix*? Yeah, I want to do *Kid Pix*!” came a chorus of first graders upon my first meeting with them in the computer lab.

“No,” I said casually. “We’re going to learn Logo.”

“Logo? Is that a new computer game? What do we do with it?” they asked.

Typical of first graders, they were full of questions and many doubts, as this was something brand new. I began the first lesson with a story that I wrote about the Logo turtle, to introduce what we would be doing for the next four weeks.

I then proceeded to introduce programming using *KidTool*, which is included with *Primary LogoWriter for the Macintosh* from Logo Computer Systems, Inc. (LCSI). This program very closely resembles a toolkit I wrote for kindergartners and first graders a few years ago. We began with the basic commands of F for forward, B for backward, R for right turn, and L for left turn. All the children had to do was type a letter, a space, and any number from one to ten, which would be multiplied by ten in *KidTool*.

Turtle orientation

After my first session of Logo with the first graders I was a bit discouraged. I found that in most cases, the girls had a lot more difficulty with the spatial concept of turtle orientation, and in general, taking charge of the computer. Most of the girls needed constant attention to stay on task. They asked a lot of questions relating to orientation, the most often of which was “How do I make the turtle go down? Is it D for down?”

For example, a child would get two sides of a square ending with the turtle heading 90 degrees. By starting with the turtle in the home position, the child would be able to program **F 4 R 9 F 4** because the first right turn is visualized as a right turn when initially heading at 0 de-

grees. The next logical move would be **R 9** again, but a right turn was not what most of the children, especially the girls, could comprehend because it looked to them as if the turtle had to turn up or down instead of right or left, as it had in the first turn.

I would then explain that there were only four commands that we would be using to move the turtle and I would ask the children to recite those four commands. After I heard F, B, R, and L, I would ask the child who asked me how to make the turtle go down which of those four commands he or she would use. This was a very puzzling concept for many of the students because they could not see how turning the turtle right or left would make the turtle go down. I used the analogy of riding a bike. I asked the children what they would do if they were riding their bike and came to the end of the street. Their responses were either turn right, turn left, or turn around and go back. By using analogies of turning I was trying to get the children to think in terms of turns for heading in the desired direction.

I devoted a considerable amount of time teaching turtle orientation through both physical movements of the children themselves and turtle moves because this concept of spatial orientation was frustrating for so many children. I used a maze and a target practice in which the students tried to “hit” a second turtle, had them draw squares and rectangles, and had them trace over pre-programmed shapes.

Spatial abilities: girls versus boys

After a few sessions I began to see a change in the children’s abilities to work independently, figure out the spatial orientation on their own, and take charge of the computer. I especially saw a very significant change in how the girls approached programming. They began to feel

more comfortable working independently and with peers without constantly asking for help.

I began to look for explanations as to why there was such a difference in the spatial abilities between the boys and the girls. By observing how they worked through their programming challenges I decided that it could be based on how much time the children spent on spatial activities outside of school, such as building with blocks.

Logo building blocks

Given this assumption, one Logo challenge I gave the children, once they had mastered drawing squares and rectangles, was to "build" with them. At first I noticed that most of the boys had no trouble building with the "blocks" they drew with the turtle, while the majority of the girls had trouble deciding how to build with their Logo blocks. (*LogoWriter for the Macintosh* allows the user to place the arrow on the turtle and, by clicking the mouse button, move the turtle anywhere on the screen without drawing a line, thus eliminating the **pu setpos pd** steps.)

Results of surveys and research

With my new hypothesis linking building experience with spatial ability, I sent a survey form home with the children asking their parents to indicate approximately how many hours per week their child spent building with blocks, Legos, Tinker Toys, and/or Lincoln Logs prior to entering first grade. My assumption of linking building experience with spatial ability proved to be reasonably accurate. I found that, on average, the female child spent no more than two to four hours per week engaged in a building activity, whereas the male child spent a minimum of four to seven hours per week. Some parents indicated that their male child spent more than ten hours per week building.

In her book *Sex Differences in Cognitive Abilities*, Diane Halpern cites a study conducting a direct test of the hypothesis that boys develop better spatial skills because of their experiences with spatial activities. In this study, Sprafkin, Serbin, Denier, and Connor (1983) gave 3½- to 4-year-old children special training with pre-

school toys that are typically preferred by boys, including blocks, dominoes, Tinker Toys, and paper cut in geometric shapes. They believed that these typical "boys' toys" were important in the development of spatial ability. They found that the children who received the special training with these toys scored higher on a test of spatial ability than a control group of children who did not receive the special training. From this study it seems clear that spatial ability can be taught, and that it can be taught to preschool children by increasing their exposure to typical male-preferred toys. While I do not consider Logo a male-preferred "toy," I do advocate using it as a tool to improve spatial ability in all children in the primary grades.

In another study, Schackman and Serbin (1978) tested first-grade children using a special children's version of the Embedded Figures Test, a spatial ability test that sometimes (but not reliably) produces sex differences. They found the usual sex difference, with the boys performing more accurately than the girls. They then provided both the girls and the boys with an "overlay training procedure" in which they used transparent overlays to highlight the way smaller figures were embedded within the boundaries of the larger figures. They found that after this training both girls and boys improved significantly. Most importantly, they found that the sex differential disappeared after the training. They believe that the reason the girls benefited more from the training than the boys is because the girls typically have less experience with spatial activities and therefore would improve more with specific instruction.

The results of these studies provide a strong case for the importance of sex differentiated environmental factors in the development of spatial skills. They also demonstrate that spatial skills are trainable. Because not all children are exposed to toys that enhance the development of spatial abilities, I believe that exposing young children to programming can be very beneficial to their learning development. Logo already applies skills in math, science, and language, and it is actually improving spatial skills as well.



Final results

At the end of four weeks I asked the children to come up with their own programming challenge. I kept a copy of each child's work and when I displayed their work on the wall of the lab it was impossible to distinguish pictures done by children who caught on quickly from those done by children who struggled at the beginning. I'm convinced that those who struggled early on with turtle orientation have improved spatial abilities because of their Logo programming experience.

On the last day I told the children they could do *Kid Pix*. Shouts of glee rang out. They all selected *Kid Pix* from the menu, and within 10 minutes, most were bored, asking if it was all right to do Logo. "I guess so," I said, grinning ear to ear. ▲

Nancy Flynn is a computer specialist in the Saint Paul (Minnesota) Public Schools. She is currently writing a book on multicultural projects using *LogoWriter* and how they can be integrated into the classroom curriculum.

References:

- Halpern, D.F. (Ed.) (1992). *Sex differences in cognitive abilities*. Hillsdale, New Jersey: Lawrence Erlbaum Associates, Inc. Publishers.
- Sprigle, J.E., & Schaefer, L. (1984). Age, gender, and spatial knowledge influences on preschoolers' computer programming ability, *Early child development and care, Vol. 14*, pp. 243-250.

Logosium '96

St. Anthony Park Elementary School
St. Paul, Minnesota
Friday, June 14, 1996

Call for Participation

The third annual Logosium will be a day of Logo workshops, discussions, and presentations hosted by the world renowned St. Paul Logo Project. This year Logosium will include sessions conducted by students sharing their projects with other students and with adults.

Sessions may be one-hour presentations or panel discussions, or two-hour, hands-on workshops on any topic of interest to the Logo community.

If you wish to offer a session, submit a one-page description of what you have in mind to:

Marian Rosen & Michael Tempel
c/o Logo Foundation
250 West 57th Street, Suite 2228
New York, NY 10107-2228
Ph: 212/765-4918
Fax: 212/765-4789
e-mail: mbrosen@oui.com
michaelt@media.mitedu

**The deadline for submissions is
March 1, 1996**

For registration and hotel information contact:
NECC '96
1787 Agate Street
Eugene, OR 97403-1923
Ph: 503/346-2834
Fax: 503/346-5890
e-mail: necc@oregon.uoregon.edu

Logosium is sponsored by the Logo Foundation and ISTE's SIGLogo.

A Circular Romp with Logo Triumphant

by Robert Macdonald

This article details an educational adventure with fourth-grade students enrolled in two elementary schools. The school system has been investigating the possible educational benefits that might accrue with the philosophy of cooperative learning. With this in mind, a mathematics teacher in a separately housed high school decided that one of her second-year unified math classes (comprising algebra, geometry, statistics, and probability) might investigate the value of pi (π) with fourth graders.

One of the fourth-grade teachers asked me to suggest ways in which she might prepare her students for the learning experience. I suggested that she collect compasses and straight edges. With these, the students would construct circles and investigate the relationship of the circumferences of their circles to the diameters and radii. As it is sometimes difficult for fourth graders to manipulate compasses, I suggested that she use tongue depressors in which some holes were drilled. Use one of the holes as the center of a circle by holding it in place with a pencil or pin. One of the other holes could be controlled with a pencil, and the circumference could be sketched out by moving the pencil in a circular path.

Some hands-on activities

I have always delighted in introducing fourth graders to constructions with a compass and straight edge. Perhaps you might introduce such basic geometric constructions as:

- making a line segment equal to a given line segment
- bisecting a line segment
- constructing an angle equal to a given angle
- bisecting an angle
- constructing a line parallel to a given line through a given point off the line
- erecting a perpendicular to a line from a point on the line
- dropping a perpendicular to a given line from a point off the line.

With these basic constructions, students gain some control over their tools and are empowered to build a number of interesting geometric figures.

Measuring what we have drawn

Having drawn a circle with a compass and inserting a diameter with a straight edge, measure the distance of the circumference with a string. Using a metric ruler determine this distance in decimeters. Now measure the diameter with that string. Again determine the length in decimeters.

With a calculator divide the length of the diameter into the length of the circumference. What is your answer? Compare your answer with others in the classroom. You will note something of great interest.

Now take the fraction $22/7$. This improper fraction indicates that 22 is divided by the 7. What is the answer? With all of the values, you have uncovered the value of pi (π). It is the relationship of the diameter of any circle to its own circumference.

As the students work through these activities, they should become more and more familiar with the terminology applied: circumference, diameter, radius, and pi. It is an awareness that has been brought about through active participation in the activities.

A pi day

The high school math class arrived at noon at the first elementary school they were to visit. To accommodate scheduling, the high school students had missed their lunch period. They quickly set up shop with 10 workstations suitably labeled and staffed. Within an hour they were able to service two fourth-grade classes, giving the young students a number of work sheets and handouts in addition to treats in the form of slices of pizza pie. (Leave it to a group of teenagers to dream up a yummy conclusion.)

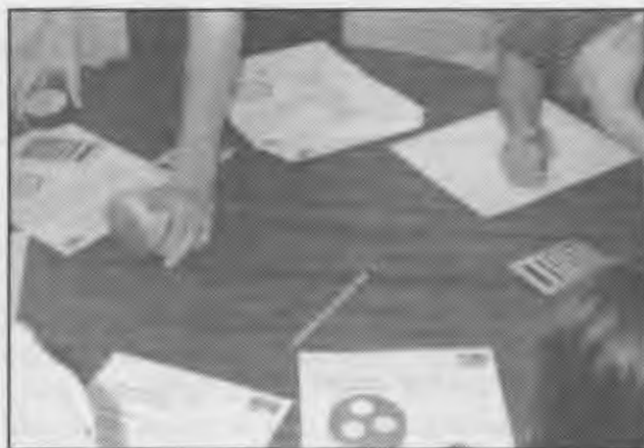




A typical selection of circular material

It was interesting to note the diversity of the offerings at the different stations. The 10 booths provided:

- a review of the concepts of circumference, radius, and diameter
- ample visual aids reviewing the areas of variously sized rectangles
- explorations involving the parts of a circle, then an introduction to the method of determining its area
- an examination of the Greek and English alphabets (this could have been a wonderful Logo project)
- a vast number of balls and much string for measuring many circumferences
- a comparison of the shape of a circle to the shapes of various polygons
- a circle word search
- hands-on experiments for finding the volume of a cylinder
- jelly bean counting contests (all of the jelly beans were in circular jars)
- a student-generated video in which Euclid, the father of geometry, appeared from the mists of time detailing the need for earth measurements.



Let's get down to calculating!

To focus interest, the high school students were costumed and bore name tags providing them Grecian identities, such as Euclid, Eratosthenes, and Aristotle. I enjoyed the presentations as much as the fourth graders. I also recognized a number of former students who seemed to enjoy their teaching tasks.

Several days later one of the fourth-grade teachers and I discussed the effectiveness of the presentation. She thought that, at least for the short term, her students had derived benefits from the visit, although, she cautioned, she didn't know how long it would last or if the students would be able to expand on the experiences on their own.

During our conversation a thought entered my mind: Why hadn't my former students applied computer-generated programs to forward their investigations? The answer was simple enough: They had not used Logo since I had had them in the fourth grade, and the school in which they were guests had never used Logo in its instructional programs, even though the school has a computer lab with 14 Apple IIGSes and two Apple IIs in each classroom.

One factor was in our favor. The second elementary school that the high schoolers were to visit did use Logo as an instructional tool. With that in mind, I got together with one of the fourth-grade teachers at that school to see how we could further her students' investigations.

Logo to the rescue

After first engaging in the hands-on activities described above, the students were soon able to discover that there are different ways of expressing the size of the same circle. There is a fixed relationship between the measurements of a circle. Therefore the students could define the size of the same circle by providing the circumference, the diameter, or the radius.

In their earliest experiences with the turtle, all the students had constructed a circle with the following Logo program, where the circle is drawn with the turtle moving on the circumference.

```
to circle
repeat 360 [forward 1 right 1]
end
```

We can supply programs based on this model that will draw circles using the value we provide for the circumference, the diameter, or the radius.

Drawing circles on the circumference

All of the following programs were written in the Macintosh version of *LogoWriter*. If you do not have a primitive that reports the value for pi, you can write the following tool procedure:

```
to pi
output 3.1416
end
```

Place the following programs on a page that you name **circle.circum**, denoting that the circles are drawn directly from the circumference. (Later in the article we will look at programs that draw circles from a center, as you would expect to do with a compass.)

The programs are very straightforward and should be easily understood by all students who have done the work described previously. The command **circle.c** requires the input of the value of the circumference. The command **circle.d** requires the input of the desired diameter. The command **circle.r** requires the input of the desired radius.

```
to circle.c :circumference
repeat 360 [forward (:circumference / 360)
right 1]
end
```

```
to circle.r :radius
repeat 360 [forward (:radius * 2 * pi) / 360
right 1]
end
```

```
to circle.d :diameter
repeat 360 [forward (:diameter * pi) / 360
right 1]
end
```

One of the first tasks the students undertook was to see if they could construct identical circles by varying the program used. If students thoroughly understand the relationship of the diameter to the circumference, they will have little difficulty and should enjoy themselves immensely.

Early on, the students decided to change the color of their turtle with a **setc** command so that the circumferences generated with each program could be differentiated.

Let's play around with these programs. In the Command Center enter the following commands to turn the turtle (and its pencolor) blue and draw a circle:

```
setc 2
circle.c 300
```

The turtle draws a circle with a circumference of 300 turtle steps.

We wish to enter a diameter that will create a circle whose circumference will be drawn on top of our previous circumference. If we divide 3 (an approximation for the value of pi) into 300 (the value of our circumference), we will get the value of 100 (an approximation for the diameter). These commands change the color to green and draw the circle.

```
setc 3
circle.d 100
```

Our estimate is close. However, the green circumference is a trifle too big. Now let's try the input of a radius. We will divide 6 into the circumference of 300 or divide 2 into the diameter of 100. The estimate of the radius with either calculation is 50. First change the color to fuchsia. Then enter the command with the input of the radius.

```
setc 6
circle.r 50
```

The fuchsia circumference fits right over the green. Our estimate based on our calculations was quite good. But why not have Logo do the mathematics? Try:

```
setc 2
circle.c 300
setc 3
circle.d 300 / pi
```

Our input has caused the circumference to be divided by the value of pi.

```
setc 6
circle.r (300 / pi) / 2
```

We have taken the previous input and divided it in half.

You might need to do some preliminary work on the order of operations with fourth graders. But I have found that they do not consider it a difficult concept. Now try some constructions on your own.

Because people do make mistakes, the students wanted to be able to erase them. So back to the drawing board. We altered each of the three procedures provided above with a **penerase (pe)** command. (The command **pd** cancels the **penerase** mode.) The following procedure shows the changes to the procedure **circle.c** to make an erasing procedure. You can change the other two procedures using this as a guide.

```
to erase.circle.c :circumference
pe
repeat 360 [forward (:circumference / 360)
  right 1]
pd
end
```

For example, give the command:

```
circle.c 200
```

Now erase it with the command:

```
erase.circle.c 200
```

Drawing circles from a center

Within a very short time, some students wanted to draw circles on the screen in the same fashion as they had drawn circles with a compass. It did not take long to decide how that could be accomplished. We put these programs on a page named **circle.center**.

The commands for these procedures, listed below, are similar to those provided for circles drawn from the circumference. However, we add one extra feature: we print out the values of the circumference, radius, and diameter for each circle as a form of mathematical reinforcement.

```
to circle.c :circumference
make "diameter (:circumference / pi)
make "radius (:diameter / 2)
pu forward :radius pd
right 90
repeat 360 [forward (:circumference / 360)
  right 1]
left 90
pu back :radius pd
print.out
end
```

```
to circle.d :diameter
make "radius (:diameter / 2)
make "circumference (:diameter * pi)
pu forward :radius pd
right 90
repeat 360 [forward (:diameter * pi) / 360
  right 1]
left 90
pu back :radius pd
print.out
end
```

```
to circle.r :radius
make "diameter (:radius * 2)
make "circumference (:diameter * pi)
pu forward :radius pd
right 90
repeat 360 [forward (:radius * pi * 2) / 360
  right 1]
left 90
pu back :radius pd
print.out
end
```

```
to print.out
ct
(print [The Circumference is] :circumference)
(print [The Diameter is] :diameter)
(print [The Radius is] :radius)
end
```

Experiment with these new procedures. For example:

```
setc 11
circle.c 350
```

You quickly notice that the turtle moves from the center of the circle and draws a circumference of 350 turtle steps and returns to its center. In addition, you note that the following information is displayed:

```
The Circumference is 350
The Diameter is 111.4085
The Radius is 55.7042
```

If you wish to take advantage of the **print.out** procedure provided in the previous programs, you have the information for the inputs for the programs that use the diameter and the radius. If you do not wish to be provided with this aid, remove the instruction **print.out** from each program. However, its inclusion offers some interesting advantages. For example, enter:

```
setc 46
circle.d 111.4085
```

(We have borrowed the value of the diameter from the printout above.)

A red circumference is drawn and you get a printout on the screen of

```
The Circumference is 350.0001
The Diameter is 111.4085
The Radius is 55.7043
```

Look at the numbers carefully. Logo is doing its math precisely. Now enter:

```
setc 15
circle.r 55.7042
```

(We have borrowed the value of the radius from the first printout.)

A yellow circumference is drawn and we note the following information:

```
The Circumference is 349.9998
The Diameter is 111.4084
The Radius is 55.7042
```

Notice that numbers may vary slightly because of computer rounding algorithms.

I discovered that the students found this type of experimentation fascinating. Many of them took advantage of the opportunity.

We then provided procedures to erase any errors.

```
to erase.circle.c :circumference
make "diameter (:circumference / pi)
make "radius (:diameter / 2)
pu forward :radius pe
right 90
repeat 360 [forward (:circumference / 360)
             right 1]
left 90
pu back :radius pd
ct
end
```

```
to erase.circle.d :diameter
make "radius (:diameter / 2)
pu forward :radius pe
right 90
repeat 360 [forward (:diameter * pi) / 360
             right 1]
left 90
pu back :radius pd
ct
end
```

```
to erase.circle.r :radius
pu forward :radius pe
right 90
repeat 360 [forward (:radius * pi * 2) / 360
             right 1]
left 90
pu back :radius pd
ct
end
```

These procedures mirror the first procedures we introduced for erasing. Therefore no samples of their use will be provided here. You might notice what was omitted or added in these procedures as you compare them with the procedures they are to erase.

"Earth and Moon" game

The following activity may be used to help students firm up their knowledge of circumferences and radii. Two turtle shapes are used: the predesigned shape 12 for the earth and a newly created smaller circular shape 1 for the moon. A knowledge of Cartesian coordinates is assumed.



```

to earth
clearpage
tell [0 1]
pu
setpos [0 0]
setsh 12
setc 11
st
show [Enter the coordinates of the point
      where you wish to locate EARTH.]
setpos readlistcc
pd stamp
end

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

to moon
cc
tell 1
pu
setsh 1
setc 4
st
show [What distance do you wish your moon to
      travel from the surface of the earth?]
make "distance readnumber
show [What circumference do you wish the
      moon to travel?]
make "circumference readnumber
forward :distance right 90
pd
circle :circumference
end

to readnumber
output first readlist
end

to circle :circumference
repeat 360 [forward (:circumference / 360)
            right 1]
end

```

First give the command **earth** and answer the question provided. Then give the command **moon** and answer both questions. There is a very important relationship to be discovered between the two numbers. You will soon find that the circumference must be a little more than six times the radius. Do you see why?

Here is a sample run-through. You would enter the underlined commands. The program displays the bracketed questions.

```

earth
[Enter the coordinates of the point where
 you wish to locate EARTH.]
20 40 (a green earth appears there)
moon
[What distance do you wish your moon to
 travel from the surface of the earth?]
(In other words: at what radius from the center
 of the earth do you wish your orbiting moon to
 travel?)

```

```

20 (a small blue moon appears)
[What circumference do you wish the moon to
 travel?]
140

```



The blue moon safely orbits the green earth. Note that my entry is seven times greater than my radius of 20. I played it safe. A number of children did seem to enjoy getting the moon to "crash" into the earth. This was an easy enough task. Make certain that your radius is somewhat small and then diminish the size of the circumference.

One young fellow expanded upon this and constructed a sunburst that appeared with printed statements announcing detonation when inputs were smaller.

Some concluding observations

In working through this material, students were firming up what they had discovered with the hands-on material. The firming up was the result of enjoying what they were doing. Trying to impose the circumference of a circle exactly upon another with the use of different inputs (circumference, diameter, or radius) was a task many students found fascinating.

Logo had provided the second group of fourth graders with opportunities for learning that had been denied the first group. In a way, this points out some of the problems inherent in American education. When a mode of learning has been proven beneficial, why shouldn't it be shared? But what if the opportunities are ignored? Alas, proselytizing is not always the answer. ▲

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

NECC '96—Call of the North • Minneapolis, MN • June 11-13, 1996



NECC '96 promises to bring you an exciting lineup of speakers from both the K-12 and Higher Education community, and the largest exhibition of educational technology products and services ever featured at a National Educational Computing Conference.

Conference highlights include:

- An exciting educational program—featuring practitioner-led pre-conference workshops and a wide variety of concurrent project, focus, poster, and paper sessions on topics including: Internet, WWW, multimedia, ethics, computer science, curriculum integration, and business/industry/school partnerships.
- “Flavors of Minnesota”—an opening reception co-sponsored by Compaq Computer Corporation that will feature displays and entertainment representing Minnesota’s broad ethnic base.
- The biggest educational technology exhibition yet—more than 600 booths will fill two domes of the beautiful Minneapolis Convention Center.

To receive a copy of the NECC '96 Advance Program containing registration and housing information, please contact:

Donella Ingham, NECC '96/ISTE, 1787 Agate Street, Eugene, OR 97403-1923
(PH) 541/346-2834, (FX) 541/346-5890, necc@oregon.uoregon.edu

NECC '96/TIES, 2665 Long Lake Road, Suite 250, Roseville, MN 55113-2535
(PH) 612/638-8764, (FX) 612/638-8769, necc96@ties.k12.mn.us

For information on exhibiting at NECC '96, contact Paul Katz, Exhibits Manager,
541/346-3537, pdkatz@oregon.uoregon.edu

You can also access information contained in the NECC '96 Advance Program—and all previously published NECC '96 materials—by visiting NECC's home page at:
<http://isteonline.uoregon.edu/necc/necc96home.html>

HOSTED BY: TECHNOLOGY AND INFORMATION
EDUCATIONAL SERVICES (TIES)

NECC '96
Minneapolis

SPONSORED BY: NATIONAL EDUCATIONAL
COMPUTING ASSOCIATION (NECA)

Turtle and Coordinate Geometry in a Secondary Environment

by Medhat H. Rahim

After reading the Fall 1994 issue of *LX*, and in particular Robert Macdonald's article *Turtle and Coordinate Geometry in an Elementary Environment*, I felt that the following would be a nice addition to Macdonald's work, but for a secondary environment.

Using Logo, a computer module will be introduced dealing with the conditions and properties that govern the definition and construction of a triangle whenever three inputs representing measures of two angles and the common side are given.

The main purpose of the module is to engage students in computer dialogues through a sequence of input and immediate feedback focusing on the necessary and sufficient condition for defining, solving, and then graphing a triangle.

The main purpose is two-fold: (a) at the middle school level, the set of Logo procedures used in the module guides the user in a process of investigating and exploring, through a sequence of trials of exchange inputs and feedback, about whether the three numbers *do or do not make a triangle and why*. Whenever the set of three inputs works, the defined triangular region will be solved and graphed on the screen and the calculated information of the unknowns will be available to the user; (b) at the high school level, the Logo procedures are proposed as an interesting problem-solving project that could be assigned for the computing classroom.

The Logo module developed was inspired by the NCTM document on Standards (1989). In this respect in particular, *Standard 12: Geometry* states:

Computer software that allows students to construct geometric figures and determine the measures of arc, angles, and segments creates a rich environment for investigation of geometric properties and relationships. (p. 114)

Development of the module

First, the user needs to choose the starting position and the heading for the triangular cursor (turtle) on the screen. The procedure **start** will do the job. It allows the user to choose an x-y position and a heading and store them as **start.position** and **start.heading**. These variables are provided as a way to return to the initial chosen x-y position and heading if needed. They can be deleted otherwise by removing the two **make** statements at the end of the procedure.

If you are working with *Terrapin™ Logo for the Macintosh®* (or *Logo PLUS™ for the Macintosh®*) the procedure **start** would look like the following:

```
to start
cleartext cleargraphics
showturtle penup
print [You will be asked to provide x-y
      coordinates and a heading for the turtle.]
print []
print [Enter an x-coordinate:]
setx readnumber
print [Enter a y-coordinate:]
sety readnumber
print [Enter a heading for the turtle:]
setheading readnumber
make "start.position pos
make "start.heading heading
pendown
end

to readnumber
make "number readlist
if empty? :number [output 0] [output first
:number]
end
```

Clearly, one may add restriction on the range of the possible values of x and y depending on the level of user's awareness of the possible disappearance of the turtle from the screen. A restriction on the heading may be included too, such as $0^\circ \leq \text{heading} \leq 360^\circ$ (or $-360^\circ \leq \text{heading} \leq 0^\circ$).

This procedure will offer the students visual practice in plotting and setting the direction for the triangular cursor on the x-y plane.

To assist in error handling, the version of **readnumber** provided causes a value of 0 to be entered automatically if the user presses a carriage return with no input.

After a position and a heading for the turtle have been chosen, a defining procedure such as **define.tri**, listed below, requests the user to provide measures for two angles and the common side of a triangle.

```
to define.tri
print [Enter a value for the first angle:]
make "angle1 readnumber
print [Enter a length for the side:]
make "side1 readnumber
print [Enter a value for the other angle:]
make "angle3 readnumber
if not (and (:angle1 > 0) (:angle1 < 180)
(:angle3 > 0) (:angle3 < 180)) [print
[Values for angles should be between 0 and
180. Try again.] define.tri stop]
if or (:angle1 + :angle3 > 180) (:angle1 +
:angle3 = 180) [print [The sum of the two
angles must be < 180. Try again.]
define.tri]
end
```

This procedure offers suggestions when the provided inputs do not work; the user would have opportunities to rethink his or her inputs and possibly adjust them. Otherwise a graph of a representative triangle will be displayed on the screen.

It would be helpful to illustrate the setup of the **define.tri** procedure. The triangle below depicts a general form of a triangular shape related to the ASA (angle-side-angle) case.

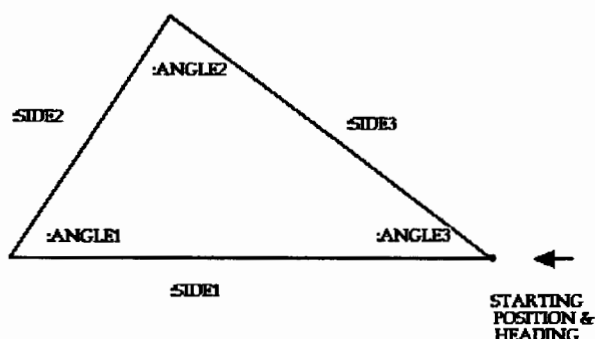


Figure 1

In the middle school level, it would be valuable to introduce a warm-up exercise on the concept that the sum of the angles of a triangle equals 180° . This may be done by having students create cutouts of triangles, cut the individual angles, and then arrange the three angles to form concretely a straight line.



Figure 2

Another activity would be for students to use a protractor and a ruler to draw a line segment as a side. They then could draw a 120° angle at one end, then draw a 100° angle at the other end, and try to create a triangle in which these two angles are both interior angles.

Research findings

Investigating and understanding the conditions involved in the procedure **define.tri** will enable students to handle assignments such as "Choose the set of three numbers that cannot be measures of angles of a triangle," reported by the NAEP (1988). That is, students would come to know that the sum of the three angles of a triangle cannot be more than 180° or less than 180° ; it can only be equal to 180° . They would realize that the angle of a triangle should always be strictly positive and less than 180° ; and the sum of 180° for two angles will make the third angle 0° , and the three numbers cannot make a triangle.

The NAEP (1988) findings on the state of the students' performance in U.S. middle and high schools (Brown et al. 1988a & 1988b) indicate explicitly the *lack of understanding of the relationships among the three angles of a triangle*. The data make a direct reference to the middle and high school students' unsatisfactory performance when dealing with these basic relationships.

For example, on four-choice multiple choice items on "Properties of Figures" concerning the relationship among the angles of a triangle:

"Choose the set of three numbers that cannot be measures of angles of a triangle," only 18% of the seventh-grade students and 17% of the eleventh-grade students with no geometry selected the right answer, while 51% percent of all eleventh-grade students in the sample answered this item correctly, and only 71% of the eleventh-grade students with geometry selected the correct response (Brown et al. 1988b, pp. 340–341).

Finishing the module

The next task is to establish a procedure for solving the defined triangle in order to calculate the remaining two sides and the third angle. The procedure **solve** below is designed for this task. It is not difficult for a user who is familiar with trigonometry to follow the **solve** procedure. However, the next illustration is helpful for understanding it.

For any triangle ABC with **a**, **b**, and **c** as measures for the sides **CB**, **CA**, and **AB**,

$$\frac{a}{\text{Sine } A} = \frac{b}{\text{Sine } B} = \frac{c}{\text{Sine } C}$$

and this relation, according to Figure 1, becomes:

$$\frac{\text{:side1}}{\text{Sine :angle2}} = \frac{\text{:side2}}{\text{Sine :angle3}} = \frac{\text{:side3}}{\text{Sine :angle1}}$$

Figure 1 may be helpful in following the steps included in the procedure **solve** below.

```
to solve
make "angle2 180 - :angle1 - :angle3
make "side2 :side1 * ((sin :angle3) / (sin
:angle2))
make "side3 :side1 * ((sin :angle1) / (sin
:angle2))
end
```

Clearly, the procedure **solve** will calculate all unknown angles and sides of the triangle, and the triangle is then said to be solved.

A new procedure, therefore, using the three inputs provided by the user and the three calculated values, is needed. At this stage, where all sides and all angles of the triangle are known, it

would seem quite easy for the user to visualize how the following procedure, **sketch**, is set to produce a representation for the triangle (see triangle on page 23).

```
to sketch
forward :side1 right 180 - :angle1
forward :side2 right 180 - :angle2
forward :side3 right 180 - :angle3
end
```

It would be helpful to notice that the command **right** (or **left**) creates an exterior angle for the turtle to turn right (or left) while, in this article, **:angle1**, **:angle2**, and **:angle3** refer to the interior angles of the triangle. This should explain why the command for the right turns in the procedure **sketch** is $180^\circ - \text{the angle}$. Note, in addition, that the turtle will at times wrap around the screen. To reduce this possibility the user should try to place graphs not too far from the centre of the turtle screen.

The following **info** procedure will provide all the measures of the angles and sides of the triangle whenever requested by the user.

```
to info
cleartext
print []
(print [Information you entered:])
(print [First side =] :side1)
(print [First angle =] :angle1)
(print [Other angle =] :angle3)
print []
print [Calculated information:]
(print [Unknown angle =] :angle2)
(print [Second side =] :side2)
(print [Third side =] :side3)
end
```

The superprocedure **asa** will activate the whole task. Each of the preceding procedures is considered a subprocedure of the superprocedure **asa**. If desired, the instruction **info** could be removed from the **asa** procedure and left as an optional command for students to use after the triangle has been drawn on the screen.

Therefore, to start the triangle program, enter the procedure name **asa**.

```
to asa
start
define.tri
solve
sketch
info
end
```

Final remarks

The work introduced and discussed in this article will provide students opportunities to consider, explore, and conceptualize—both visually and interactively—the relationships among the measures of the angles that determine whether or not a triangle does exist (*part-to-whole interrelationships*); relationships among the measures of the three angles of a triangle and among their corresponding sides (*part-to-part interrelationships*); and relationships among the measures of the three angles and the type of a triangle they determine (*part-to-whole interrelationships*). Such basic propositions about triangles are often unclear to a large portion of the student population at both middle and high school levels (Brown et al. 1988a & 1988b). This module covers the following aspects of triangles:

1. The sum of 180° degrees of two angles will not make a triangle.
2. The sum of more than 180° degrees of two angles will not make a triangle.
3. Only the sum of less than 180° of two angles will make a triangle.
4. The sum of 90° degrees of two angles will make a right angle triangle.
5. Pythagorean theorem can be applied in solving the triangle whenever the sum of two angles of the triangle is 90° .
6. If the two angles were acute, then their sum of more than 90° and less than 180° will make an acute triangle.
7. If the two angles were acute, then their sum of less than 90° will make an obtuse triangle.
8. Two equal angles make an isosceles triangle. The two corresponding sides will be equal.
9. Three nonequal angles make a scalene triangle. The corresponding sides will be unequal.
10. Three equal angles, 60° each, will make an equilateral triangle (or two angles of 60° each will make an equilateral triangle).
11. Whenever the three angles are equal, the corresponding sides will be equal.
12. The sum of the three angles is 180° .
13. The largest angle will be opposite the longer side and the smallest angle will be opposite the shortest side.

A set of questions such as the following may be given as a handout along with this activity:

Can you make an equilateral triangle?

Can you make an isosceles triangle?

Can you make a scalene triangle?

Can you make a right angle triangle?

Can you make an acute angle triangle?

Can you make an obtuse angle triangle?

Can you make an acute isosceles triangle?

Can you make an obtuse isosceles triangle?

Can you make a right isosceles triangle?

Can you make each of these triangles with one of its sides horizontal and one of the vertices above it?

Can you make each of these triangles with one of the vertices below it?

Can you make each of these triangles with one of its sides vertical?

As a miniproject, it would be helpful for the teachers to suggest an assignment like the following for students to complete and record their observations prior to using computers. The teacher may suggest certain sets of three inputs, as measures for two angles and the common side of a triangle, some of which will not



work. Students may then try the assignment using computers:

**Fill in the last two columns
whenever appropriate**

First Angle	Common Side	Other Angle	Triangle? Y or N	If N, Why?
120°	50	30°		
30°	50	120°		
60°	70	30°		
60°	100	120°		
80°	60	110°		
-30°	80	60°		
70°	50	-60°		
...		

The text of the computer module presented may be modified as desired. The structure of the module does work, however, and is open for computer and Logo enthusiasts, students and educators alike, for modification. ▲

Medhat H. Rahim is currently an associate professor of mathematics and mathematics education at Lakehead University, Faculty of Education, Thunder Bay, Ontario, Canada, P7B 5E1. Over the last decade and a half, he studied and taught Logo at several places: Northern Illinois University, University of Alberta in Canada, and the University of the West Indies in Trinidad and Tobago. His interests include the use of manipulatives and technology in teaching and learning geometry. Dr. Rahim can be reached via e-mail at: MHRahim@Flash.Lakeheadu.ca.

References

- Abelson, H. & di Sessa, A. (1980). *Turtle geometry. The computer as a medium for exploring mathematics*. Cambridge, MA: MIT Press.
- Brown, C., Carpenter, T., Kouba, V., Lindquist, M., Silver, E., & Swafford, J. (1988a). Secondary school results for the Fourth NAPE Mathematics Assessment. *Mathematics Teacher*, April, 241-248.
- Brown, C., Carpenter, T., Kouba, V., Lindquist, M., Silver, E., & Swafford, J. (1988b). Secondary school results for the Fourth NAPE, Mathematics Assessment. *Mathematics Teacher*, May, 337-347.
- Macdonald, R. (1994). Turtle and coordinate geometry in an elementary environment. *Logo Exchange*, 13(1), 21-27.
- NCTM, (1989). *Curriculum and Evaluation: Standards for School Mathematics*. Reston, Virginia.

New Address for Terrapin Software

Terrapin Software, Inc., developer of Logo language programs and publisher of Logo-based curriculum materials, announces a change of location. The new address, as of November 15, 1995, is:

Terrapin Software, Inc.
10 Holworthy Street
Cambridge, MA 02138
tel: 800-972-8200
tel: 617-547-LOGO (547-5646)
fax: 617-492-4610
e-mail: 71760.366@compuserve.com

At its new location, Terrapin will continue to strengthen and expand its offerings through cooperative marketing and joint product development with Harvard Associates, Inc., another major developer of Logo products. Please contact Terrapin at its new address to receive a catalog and free subscription to its new series of newsletters.

Logo Action Research: Getting Teachers Excited about Technology

by Nan Youngerman and Donna Cutler-Landsman

Change threatens people. Those of us schooled during the age of slide rules need a supportive framework to move into the age of digital Zap Cameras, scanners, computer robotics, laser disks, and telecommunications. Participating in collaborative action research brings teachers together in an active environment that promotes growth and change. A stimulus for experimentation, action research energizes teachers to try new teaching practices with a safety net provided by colleagues.

During the 1990–1991 school year 100 teachers were part of the national Logo Action Research Collaborative (LARC), sponsored by the National Science Foundation and co-led by principal investigators Dan and Molly Watt of the Educational Development Center. This article focuses on the impact of LARC on the teaching practices of one of these teachers, Donna Cutler-Landsman.

Getting involved

Cutler-Landsman has taught middle school in the Middleton-Cross Plains, Wisconsin School District since 1974, focusing on the areas of mathematics and science. During 1990–1991 she was team-teaching sixth grade.

Although interested in technology, her limited background included a few computer classes and what she had gleaned from teacher inservices, workshops, and the advice of colleagues. Cutler-Landsman saw a *LEGO® TC logo* demonstration at a “Science on Saturday Workshop” taught by Nan Youngerman and others and proceeded to take the first steps to introduce this robotics program to her sixth graders.

Looking back, Cutler-Landsman says, “I was excited by technology, but frustrated by my lack of understanding of manuals and hardware. I needed support to move past the glitches and to

feel confident in my ability to integrate technology into my teaching.”

Responding to an invitation from Youngerman, the Madison LARC site-leader, Cutler-Landsman soon agreed to participate in the Logo Action Research Collaborative.

Posing a research question

During the initial meeting, as Cutler-Landsman met fellow action researchers and began to learn about the action research process, she focused on an idea for her own research question. With confidence she informed the group she would investigate whether there was a motivational difference between girls and boys who worked with *LEGO TC logo*. She collected data by observing students and quickly realized the answer to her question was YES! Boys in her class created complicated and unique projects, while the girls she observed showed signs of frustration and boredom.

Concerned about what she was seeing and cognizant of studies reflecting the lack of involvement of girls in engineering fields, Cutler-Landsman determined she needed to alter what was happening in her classroom. Her action research question evolved into a more relevant question: How can I increase the motivation and creativity of girls as they use *LEGO TC logo*?

Finding solutions

Posing a new question was easy compared to choosing a course of action. Brainstorming the situation with the LARC group offered new insight into what was happening and possible interventions. Armed with support and suggestions from her action research colleagues, Cutler-Landsman tried the following changes in her teaching methods:



- mixed gender groups
- girls in leadership roles
- student-initiated projects
- more direct instruction for girls.

After several weeks it was apparent that the girls' interest levels, confidence, and competence had increased. Working with a second class of sixth graders, Cutler-Landsman incorporated the changes and observed again. After two weeks of surveying the class, she found a high interest rate with both boys and girls. In addition, girls reported feeling more confidence and comfort in creating original projects.

The action research group supported the evolving *LEGO TC logo* work in Cutler-Landsman's classroom and gave a framework to share more formally with other educators beyond the LARC teachers. In this way one teacher's experience could benefit others.

Teacher turns researcher

Participation in the Madison LARC offered Cutler-Landsman a unique opportunity to expand her role as teacher to one of researcher. It gave her skills necessary to focus on a question, design an assessment tool, collect data, and, most importantly, alter her teaching techniques to directly benefit her students. As a result of her research into gender differences she has become more proactive in her approach to girls and *LEGO TC logo* learning. Her teaching style and the composition of her learning groups has changed, and the involvement of girls with *LEGO TC logo* has increased dramatically.

In a larger context, participation in the action research collaborative has allowed her to develop working relationships with other educators who share similar interests and goals. They have fostered her efforts to try new techniques, offered numerous suggestions to improve her teaching methods, and provided support and encouragement for her work. In addition, the research has allowed her to become involved at the national and international level with others in computer education.

Cutler-Landsman published her *LEGO TC logo* results in *New Paradigms in Classroom Research*

on *Logo Learning* and presented her findings at several National Educational Computing Conferences (Phoenix, 1991; Dallas, 1992; Boston, 1994). Lego Dacta features this article in their school sales representatives' notebook. Since Cutler-Landsman's work, Lego Dacta has also introduced pastel Legos and a line of toys with greater appeal to girls.

Cutler-Landsman participated in a second technology action research collaborative, researching the Internet as a vehicle for original student research. The NEA professional journal, *Teaching and Change*, published this study. This prompted her to submit a proposal along with Youngerman to the 1995 World Conference on Computers and Education in Birmingham, England, focusing on action research as a mechanism for systemic change. This year she hopes to expand her work, integrating technology into her sixth-grade classroom by participating in a telecommunications project stemming from contacts at that conference.

A great deal has changed in technology and in Cutler-Landsman's classroom in a span of five years. "I am no longer intimidated by any aspect of technology," she said. "I know if I have questions, supportive colleagues will help answer them. I continue to learn and to experiment with added confidence." ▲

Nan Youngerman has taught in the Madison, WI, school district since 1971. She is currently at Crestwood Elementary School, teaching science, math, and technology to first through fifth graders. This past year she was the 1994 Wisconsin Presidential Awardee for Excellence in Mathematics and Science Education sponsored by the National Science Foundation. She can be reached during the school year at: Crestwood Elementary School, 5930 Old Sauk Road, Madison, WI 53705; or anytime via e-mail at neyounge@facstaff.wisc.edu.

Donna Cutler-Landsman is a sixth-grade teacher at Elm Lawn Elementary School in the Middleton-Cross Plains School District in Wisconsin. She has taught for almost 20 years.

References

Cutler-Landsman, D. (1993). Bridging the gender gap with *LEGO TC logo*. In D.L. Watt &

M.L. Watt (Eds.), 91-99. *New paradigms in classroom research on Logo learning*. Eugene, OR: ISTE.

Cutler-Landsman, D., & Wrzesinski, C. (1994). Internet: A vehicle for original student research. *Teaching and Change*, 1(2), 135-153.

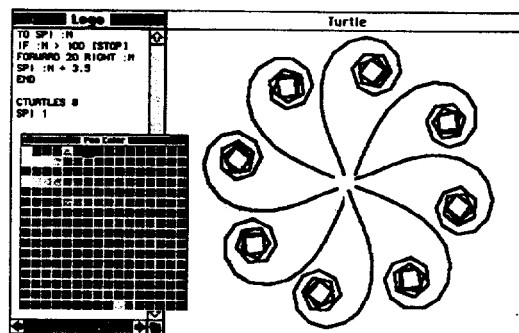
Kemmis, S., & McTaggart, R. (1988). *The action research planner* (3rd ed.), Victoria, Australia: Deakin University Press.

Watt, D., & Watt, M. (1992). *Logo learning: Strategies for assessing content and process*. Eugene, OR: ISTE.

Watt, D., & Watt, M. (Eds.) (1993). *New paradigms in classroom research on Logo learning*. Eugene, OR: ISTE.

Watt, D., & Watt, M. (1993). Teacher research, action research: The Logo action research collaborative. *Educational Action Research: An International Journal*, 1(1). Newton, MA: Education Development Center, Inc. (1991)

How Can Logo Help You Teach?



New Logo PLUS for the Macintosh lets your students explore—in glorious color!

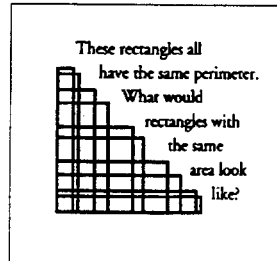


Logo is a teaching tool unlike any other. It is active and hands-on. Kids can explore, making their own discoveries about angles and measurements, patterns and shapes. With Logo, rotations, reflections, translations, and tessellations are a snap.

Students estimate, plan, and create, using a variety of problem-solving and debugging strategies. Students can figure out on their own how to draw arcs, circles and polygons, or they can use the handy built-in tools. They can explore fractions, use variables, plot points, graph functions. In short, your students can explore your entire math curriculum using Logo.

Logo teaches process, the "how-to" of math. It says "explore and see for yourself." And in learning step-by-step, students acquire the sort of knowledge that sends them confidently in many new directions.

With over 50 tools for exploring all areas of math, Logo PLUS is the natural and logical choice of a language for your classroom.



Call or write for our free *Exploring Math with Logo* and *Why Use Logo?* booklets.

1-800-972-8200

Terrapin™

10 HOLWORTHY STREET • CAMBRIDGE, MA 02138
617-547-LOGO (547-5646) FAX 617-492-4610

Logo languages and curriculum materials for all machines.



Recursion, The Problem Solver

by Jim Muller, The Logo Forum

A number of years ago, an editor asked me what was so special about Logo. Among the powerful features I mentioned was recursion.

Ok, what do you do with it?

I could describe it. I could provide all sorts of examples. But I really could not explain it. Nor could I describe a problem I had solved using recursion. After Dr. Kathleen Martin of Texas Christian University showed me her stack-of-paper demonstration (see "The Recurring Question of Recursion" in this issue), it began to make sense.

A recursive procedure is one that calls itself. Another way of saying this is that recursive conditions are those that repeat themselves indefinitely or until a certain condition is met. To narrow the focus even more, recursive conditions are those that repeat themselves until the problem is solved.

The problems and the conditions

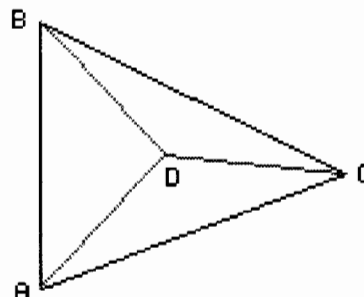
Here is a problem that Logo and recursion can solve much faster and easier than I can.

Draw a circle around a randomly generated triangle so that the edge of the circle touches the three points of the triangle. To solve any problem, start with what you know. Determine what you need to know. Then go find it.

What do you know?

You know that the three points of the triangle are going to be on the circumference of the circle. If you can find a point that is the same distance from each of those points, then you have the center of the circle. The rest is easy.

To make things easier to understand, label the points on the circle: A, B, and C.

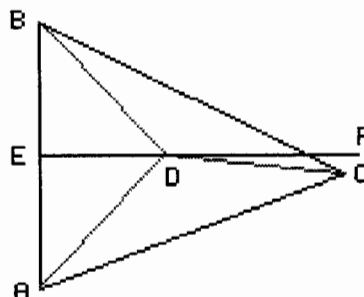


We have to find point D, a point that is the same distance from A as it is from B and C.

D is the center of the circle and the three lines, AD, BD, and CD are each a radius of the circle we are supposed to draw.

Now, how can we prove that?

Draw the line EF so that it is perpendicular to the middle of line AB.



What can you learn from this drawing now?

You have two triangles, ADE and BDE, that share one side and have two short sides that are equal. Therefore, the sides AD and BD must be equal.

Next, if we can find the point on line EF that makes these two lines equal to line CD, we have found the middle of the circle we want to draw.

The random triangle

The first step is to create a random triangle. The procedure **random.tri** sets some restrictive limits (right 120 – random 30) to make the illus-

trations easier to follow. When you expand the limits, the resulting circle sometimes goes off the screen. But the basic concept still applies.

```
to random.tri
make "pointa pos
forward 100 right 120 - random 30
make "pointb pos
make "dist 250 - random 100
forward :dist
make "pointc :pos
home
end
```

This procedure starts from PointA, which is a list defining the coordinates [0 0]. The turtle goes forward 100 and turns right a random angle, somewhere between 120 and 90. This is PointB at coordinates [100 0]. The turtle then goes forward between 150 and 250 and sets PointC. Then the turtle goes Home.

[Note: The point variables can be defined to suit your particular version of Logo. For details, see "Conversion tips" at the end of this article.]

Finding the center

What else do we need to know to complete our circle? We need to find the point D, which is the same distance from B as it is from C. We already know that AD and BD are equal and that each is going to be a radius of our circle.

The first thing we need for that is a distance procedure.

```
to dist :x1 :y1 :x2 :y2
output int dist1 :x1 - :x2 :y1 - :y2
end

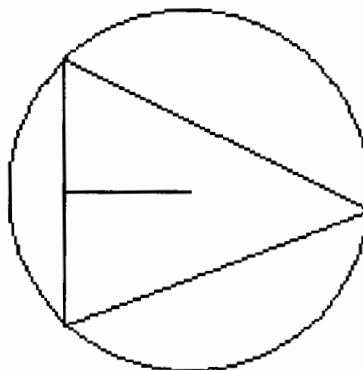
to dist1 :dx :dy
output sqrt (:dx * :dx) + (:dy * :dy)
end
```

The **dist** procedure measures the distance between two sets of coordinates. To keep things simple and easy to compare, the output is an integer.

The next step is to write a procedure that calculates two distances, the distance between B and D, and the distance between C and D. When these are the same, you have the center around which you can draw a circle.

```
to check.distances
make "BD dist first :pointb last :pointb
first :pointd last :pointd
make "CD dist int first :pointc int last
:pointc first :pointd last :pointd
(print [BD = ] :BD)
(print [CD = ] :CD)
if or :BD = :CD abs (:BD - :CD) = 1
[hideturtle circle :pointd :BD] [forward 1
make "pointd pos check.distances]
end
```

The first two lines of **check.distances** calculate the distances BD and CD. Then Logo compares the two numbers. If **:BD = :CD** is true (or if they are within 1 of each other), they are equal (or close enough). Logo draws a circle with PointD as the center and a radius of :BD. If the two distances are not the same, the turtle moves forward 1 and checks the distances again. As processing continues, the values of BD and CD are printed, so you can see what the actual distances are.



```
to circle :center :radius
local "amt
make "amt :radius * pi / 180
penup setpos :center
setx xcor - :radius
setheading 0
pendown
repeat 360 [forward :amt right 1]
penup setpos :center pendown
end
```

Putting it all together

To put the whole thing together, here is a place to start.

```

to start
random.tri
setpos :pointa
forward 50 right 90
make "pointd pos
check.distances
end

```

This problem provides another look at a question that has been around for some time. Are you better off being able to define the problem? Or the solution? I can only speak from my own experience. Using Logo and the computer to define what at first glance seem like totally innocuous problems, has opened up some fascinating solutions.

Conversion tips

These procedures were developed using the freeware Logo for Windows package, *MSW Logo*.

If your computer or your version of Logo runs out of memory because of the large number of recursive calls that can be required, change the third line of the **start** procedure to:

```
forward 50 right 90 forward 100
```

If your version of Logo does not include **pos**, **pi**, **abs**, **int**, or **setpos** commands, add the following procedures, as needed:

```

to pi
output 3.14159
end

```

```

to abs :num
if :num < 0 [output - :num] [output :num]
end

```

```

to setpos :x :y
setxy list :x :y
end

```

```

to int :number
output integer :number
end

```

```

to pos
output list xcor ycor
end

```

You can read more about **Jim Muller** at the end of his other article in this issue, "The Recurring Question of Recursion."



December 5–8, 1996
Tampa, Florida

Tampa Convention Center

International Society for Technology in Education (ISTE) will continue to partner with the *Florida Association for Computers in Education (FACE)* and will add the participation of a second site in Monterrey, Mexico via telecommunications for the Fifth International Conference on Telecommunications in Education.

The world of telecommunications continues to grow dramatically, especially in the field of education. This exciting conference will give leaders in educational telecommunications a forum in which to learn about rapidly advancing developments in telecommunications technologies. Be one of the thousands of educators, policymakers, and researchers who will join together in this unique global forum to exchange the latest in telecommunications ideas, techniques, strategies, and policy concerns.

To receive an Advance Program, to be mailed in August '96, please contact:

International Society for Technology in Education
Tel•Ed '96
1787 Agate Street
Eugene, OR 97403-1923
Phone: 541/346-2411 Fax: 541/346-5890
Internet: Lori_Novak@ccmail.uoregon.edu

If you have Internet access and World Wide Web software, point to URL: <http://isteonline.uoregon.edu> for information on Tel•Ed as it becomes available.

For information on exhibiting at Tel•Ed '96 contact
Paul Katz, Exhibits Manager
Phone: 541/346-3537
Internet: pdkatz@oregon.uoregon.edu



Windows on Logo: Logo and the Internet— The Turtle and the World Wide Web

by Glen L. Bull and Gina L. Bull

For the last 14 years we have contributed an article to each issue of *Logo Exchange*, from the first issue to the present. In a sense we have grown up with the microcomputer-based versions of Logo. *LX* periodically undergoes facelifts to ensure that it remains fresh and vital. While the Logo universe is so rich that there is always another interesting aspect to discuss, transitions in *LX* make this an appropriate time to bring the longest-running *LX* column to a close. Before we do, we would like to use the last two columns of this academic year (the Spring and Summer 1996 issues) to make some final observations about Logo in general and *LX* in particular.

Problems with Logo programs in print

Tom Lough founded *LX* 14 years ago to provide a forum for sharing both Logo ideas and Logo programs. Print proved to be an effective medium for sharing ideas, but it has some limitations for sharing Logo programs. From the beginning we were startled at how often typographical errors would creep into even short Logo programs, no matter how carefully each draft was proofread. We eventually began asking pilot testers to work through each column, entering the code to be sure that it worked as we intended. This helped, but typographical errors still crept in at other stages of the editorial process.

Readers also tend to contribute additional typing mistakes when they enter Logo programs. Using a monospaced font such as Courier makes it a bit easier to identify spacing and punctuation. However, only machines type perfectly—everyone else makes typing mistakes. From time to time we would see *LX* columns used at a workshop or conference, and were

often surprised at how frequently incorrectly entered programs derailed exploration of Logo concepts.

Dialect differences

Different dialects of Logo present another problem. In the beginning there was only one dialect of Logo—but that first dialect of Logo did not even have a turtle. Every improvement and enhancement to the language made newer versions of Logo somewhat incompatible with prior versions. Two or three major versions of Logo—Terrapin, Logo Computer Systems, Inc. (LCSI), etc.—run on several different computing platforms (Apple II, DOS, Microsoft Windows, Macintosh, etc.). In addition there are dozens of other variants, some commercial (*Object Logo*, *PC Logo*) and some freely distributed in the public domain (*UCB Logo*, *MSW Logo*).

Because of these dialects, even correctly entered programs designed for one dialect often will not run in other dialects—at least, not without modifications beyond the capabilities of many readers. For a time we attempted to provide versions of Logo programs in alternate versions of major dialects, such as Terrapin and LCSI. However, this eventually proved to consume too much space.

These issues were the subject of some recent thoughts by *LX*'s newest editor, Dorothy Fitch, reporting on a discussion by *LX* readers at the National Educational Computing Conference:

Although some people thought that *LX* should not print Logo code at all, I feel it is important for several reasons: to provide easily usable projects, to keep the journal less dry, and to provide good models for



programming—after all, Logo is a programming language!

—Dorothy Fitch, *Logo Exchange* Editor,
Electronic Mail Message to *LX* Columnists, Summer 1995.

Dorothy Fitch offered several good suggestions on ways in which Logo programs might be made more accessible to *LX* readers. For example, a chart displaying the names of commonly used commands in different dialects of Logo could assist readers in conversion of programs to their version of Logo.

A network solution

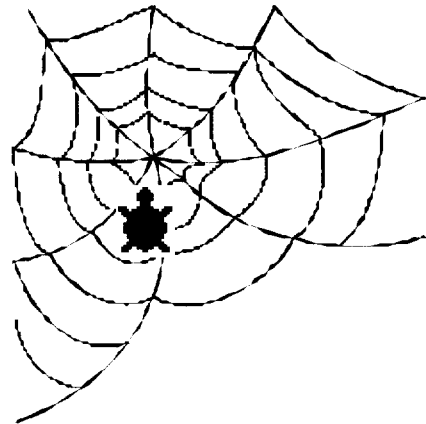
Our own computing experiences suggested one other way in which Logo programs might be made more accessible to *LX* readers. In the early 1980s, about the time that Tom Lough was printing the first issues of *LX* on a dot matrix printer, we began to collaborate with local schools on ways to link teachers through computing networks. Eventually a regional computing network that we developed called Teacher-LINK served as the model for one of the nation's first state-wide public school Internet systems, Virginia's Public Education Network (PEN).

We had hoped that K-12 networks could provide a convenient way to exchange Logo programs without retyping them. The first networks based on older mainframe technologies were simply too cumbersome. We still have a set of instructions developed by Judi Harris to allow teachers to download files. At the time, this process required 38 steps, and was clearly impractical for day-to-day use.

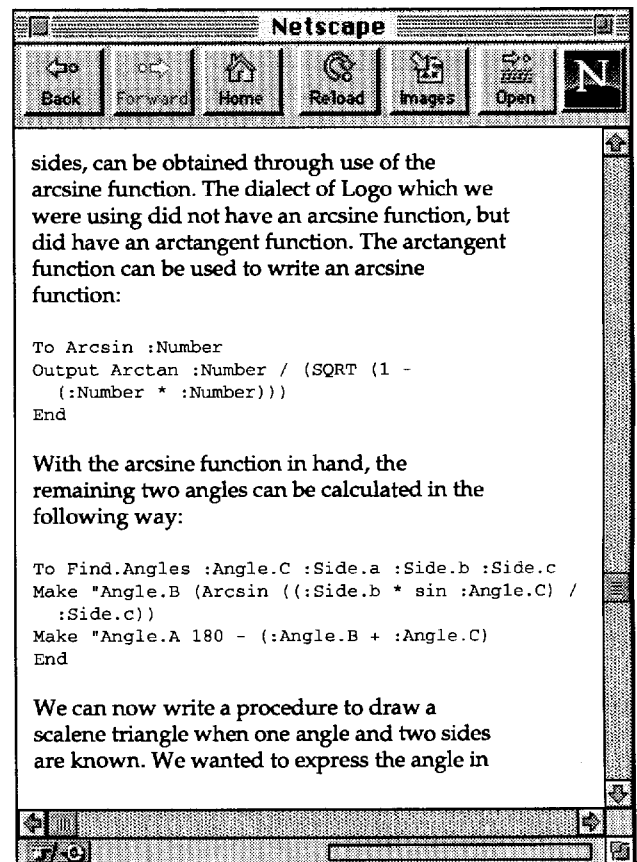
World Wide Web possibilities

The advent of the World Wide Web and client-server computing has changed this equation. With a Web browser such as Netscape, it is possible to use the Save As command to save a file to the microcomputer in a single step. The Web also permits creation of hypertext links that allow readers to download a file with a click of the mouse. Most states are creating K-12 networks that provide access to the Internet. Major information services such as America

Online and Compuserve have also provided gateways to the Internet.



In Spring 1994 we began to experiment with posting *LX* columns on the World Wide Web for our classes. We have included a sample screen below.



When we posted these initial files it was necessary to enter the formatting tags by hand. Today most major word processors such as *Microsoft Word* and *WordPerfect* include an option to save a file in the HyperText Markup Language (HTML) necessary to post documents on the World Wide Web. This makes the process considerably easier than it was before the widespread popularity of the Web.

Distribution of *LX* programs through an International Society for Technology in Education (ISTE) Web site would have several advantages. It would reduce the frustration caused by typing mistakes, and save time. It would also make it possible to store several different versions of the same program in different dialects without taking up valuable space in the printed version of *LX*.

CD-ROM as a distribution option

Once the *LX* materials have been posted on a World Wide Web site, it will also be relatively easy to transfer them to a CD-ROM. Another organization with which we work, the Society for Information Technology and Teacher Education (SITE), distributes a compilation of instructional materials on an annual CD-ROM.

A CD-ROM would have the capacity to hold several years of past *LX* articles and versions of accompanying programs in several different variants. It would also be possible to include other materials, such as public domain versions of Logo. For other educational CD-ROMs, vendors have even provided demonstration versions of programs with the Save function disabled. This allows users to compare the latest features of newer programs.

The title *Logo Exchange* was selected with some thought about the intent of the publication. When *LX* was founded, 300 bps modems were still in common use, and 1200 bps modems were considered state of the art. It would be several years before Bill Gates would sponsor a conference on "The New Papyrus" to publicize the potential of the CD-ROM.

Looking to the future

Electronic supplements to *LX* would make a variety of other options possible as well. We use a variety of Logo programs and utilities in our personal classes that are simply too long for inclusion in *LX*. There is a limit to the length and complexity of a Logo program that can be presented and discussed in a three- to four-page article. An electronic Web site would make it possible for *LX* authors to provide extended and enhanced versions of Logo programs presented in *LX*.

An *LX* Web site could easily include a Comments button that would allow readers to provide feedback. A Submissions button could also allow readers to submit Logo programs that they have developed themselves. We have often found that educators are too modest, believing that materials they have developed for their own classrooms would not be of interest to others. An electronic submissions forum could allow tips and short programs to be shared. These contributions might not warrant the time required to develop a full article, but still could be of interest to others. An Exchange Forum on the *LX* Web site could provide a convenient electronic location to share such programs.

Summary

There will still be a need for paper documents like the print version of *LX* for the foreseeable future. However, new technologies such as the World Wide Web offer new possibilities for exchange of Logo ideas and programs. These newer possibilities make it possible to extend and complement the printed version of *LX*, continuing to fulfill the original vision for the publication.

In our final column in the Summer 1996 issue, we will share some thoughts stemming from the decade and a half that we have spent with *LX*. ▲

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 computer systems engineer in the Information Technology and Communication organization at the University of Virginia.

Internet: GBull@Virginia.edu
Gina@Virginia.edu



Logo: Search and Research

Parallel Programming

by Douglas H. Clements and Julie Sarama

In regular, or sequential, Logo, there is a single process that runs instructions one step at a time. But what if you wish to program a group of animated figures to dance together, or if you want two Lego robots to perform two tasks in synchrony with a graphic on the screen?

One of the many recent innovations in various versions of Logo is the addition of parallel programming. Also called "concurrent" or "multiprocessing," this feature allows programmers to control multiple, interacting processes. Parallel programming is possible in *MicroWorlds* (from LCSi) and in other versions of Logo, such as *MultiLogo*.

Parallel programming with Logo

Mitch Resnick, the inventor of *MultiLogo*, has been working with parallel Logo for many years (Resnick, 1990). His adventure began as he became increasingly aware of sequential Logo's limitations for many tasks. For example, in designing a Lego program, a child might have to write a procedure to recursively check one (front) sensor, then another (back) sensor. With parallel processing, however, students might just tell the computer to check once. Worse, what if you wish to have a car check its front and back sensors and have the computer flash a light for 5 seconds every time it reverses direction? The following might work.

```
to check-sensors
  listento :front-sensor
  if sensor? [rd flash-light]
  listento :front-sensor
  if sensor? [rd flash-light]
  check-sensors
end
```

But, what happens if the car bumps into something while the computer is flashing? It won't change direction until after the flashing stops.

The computer cannot flash light and check the sensors at the same time. It does one or the other.

To ease such problems, Resnick developed *MultiLogo* to use for his research.

Research on parallel programming

Resnick conducted a study with elementary students using *MultiLogo* (Resnick, 1990). Fourth and fifth graders used this Logo to control the concurrent actions of robotic machines. Resnick wanted to see whether students could use parallel processing and if they displayed new types of bugs.

He guided the students through a sequence of projects. The "walker/flasher" projects involved a Lego walking machine with a light on its nose. He asked students to make it walk back and forth and then to make the light flash while it was doing so. One approach might be to set up three processes (or "agents").

```
to walk                                     walker process
talkto :motor-port
repeat 6 [onfor 30 rd]
end

to flash                                    flasher process
talkto :light-port
repeat 20 [onfor 4 wait 2]
end

to walk-and-flash                           manager process
ask :walker [walk]
ask :flasher [flash]
end

walk-and-flash
```

The second project involved the use of Lego touch sensors to control the walker/flasher. This requires the coordination of a larger number of processes. Also, problem decomposition was more difficult. What would each agent be? What would each do? One solution, using some of the previous code, is as follows:

```

listento: walk-sensor      walker process

to sense-and-walk
if sensor? [walk]
sense-and-walk
end

listento: light-sensor     flasher process

to sense-and-flash
if sensor? [flash]
sense-and-flash
end

to sensor-task              manager process
ask :walker [sense-and-walk]
ask :flasher [sense-and-flash]
end

sensor-task

```

For the third project, students control a Lego turtle with two motors.

Findings

Students used the idea of parallel processing and agents easily. They all appeared to understand the power of parallel programming. One student, FB, described it to his partner, NL, as follows:

Remember how we were trying to do that program with the people walking and music and snow falling? Well, here there are different agents, and each agent can do something on the screen at the same time. (p. 159)

However, students did say there were some struggles. Here is NL, FB's partner.

You're asking two different agents to talk to two different things to do two different things. And so everybody's doing something else. And it's confusing to figure out what you are doing and why you are doing it and who told you to do it. (pp. 159–160)

Overall, the projects went well. All students successfully programmed the walker/flasher and touch-sensor projects. In his article, Resnick jumped over the success stories and focused on any new difficulties that parallel programming might bring. Knowledge of this would be important to teachers working with a new parallel

version of Logo, such as *MicroWorlds*. They tell us how students think about such processes and how we might anticipate and help them with problems they might have.

Three new bugs

Resnick identified three primary types of parallel programming bugs: problem-decomposition bugs, synchronization bugs, and object-oriented bugs. These categories can overlap.

Problem-decomposition bugs

Problem-decomposition bugs deal with some of FB's confusions. How do you break down problems into actions to be performed in parallel by several different processes?

Problem decomposition is a basic problem-solving strategy in programming and other fields. In parallel processing, you have an added burden: deciding how to divide the pieces of a problem among different processes. In all versions of Logo, you have to decide how to break down *what* needs to be done. In parallel processing, you also need to decide *who* should do it.

The two students quoted above, FB and NL, ran into problem-decomposition bugs right away in the walker/flasher task. FB used the same agent to perform both her "light" and "motor" procedures. She then recognized that she didn't know how to get them to run at the same time. "You could do repeat 200 [motor light] but then it will do motor and then it will do light. You have to combine them."

Eventually, all students solved that problem, but problem decomposition was still difficult for them. On a complicated touch-sensor task, one child, DB, used agents. When more parallelism was demanded, however, DB fell back on using recursion and doing one thing, then another, then the first one again quickly—a process called "time slicing." Eventually, DB used seven processes: two to check the sensors, one to run the walker, one to run the flasher, and three to get the program going.

What causes these bugs? Most students seemed stuck in sequential thinking. Possibly sequential programming experience contributed to this fixedness. The student who had the least

sequential Logo experience adapted to the parallel processing the easiest.

Synchronization bugs

Synchronization bugs deal with coordinating and orchestrating multiple parallel processes. These were the most common bugs. Here, students decomposed the problem, but couldn't coordinate the different processes. For example, for the walker/flasher project, three of four students wrote procedures such as the following:

```
to walk-and-flash
walk
ask 1 [flash]
end
```

Unfortunately, all the walking would stop before the flashing began. The two instructions have to be reversed.

We have discussed in previous columns that many novice programming bugs may stem from a too-literal interpretation of "talking to the turtle." For example, some novices think the turtle will "read between the lines," as a human often does, and do what they *meant* to say rather than what the code actually says.

In parallel programming, the conversation is more complex. Instead of talking to the turtle, you might talk to several processes, and they might communicate among themselves. In this sense, it is less appropriate to think of talking to a turtle, and more helpful to think of orchestrating a conversation among others.

But children have few experiences such as that. Instead, they use a typical conversational metaphor. FB explained walk-and-flash by saying, "...you start one, then you ask the other one to start."

Another relevant feature of children's thinking was excessive parallelism. That is, they thought individual processes could run more than one procedure at a time. For example, NL "acted out" walk-and-flash by walking and simultaneously "asking 2" to flash because she was busy walking (!). They may have been over-generalizing the new idea of parallelism. Also, the use of Lego devices might have convinced children that the processes needed to perform some continu-

ing action, for example, to *keep* a light on. But the most important reason seemed to be identification of processes with people—who *can* do several things at once.

Object-oriented bugs

Object-oriented bugs emerge from confusion about different types of "objects," such as processes (agents), procedures, motors, and ports. Students did not have a clear model of which commands influenced which objects or how objects interacted. The most common were confusions between processes and other objects.

For example, if they wanted to stop objects, they mixed up stopping a process, a procedure, a motor, and a turtle. FB had process 1 turn on a motor, then asked process 2 to halt process 1, but the motor kept running.

In their previous programming experience, students did not need to distinguish between different types of objects. "Stopping the procedure" and "stopping the turtle" amounted to the same thing.

Teaching and learning parallel programming

Studies such as this are valuable for many reasons. Perhaps most important in our role as teachers, they tell us how students think about parallel processes and how we might anticipate and help them with problems they might have.

In addition, Resnick found several teaching approaches effective. For example, when students seem stuck in sequential thinking, Resnick challenged them to think of programming real-life situations in which people must execute tasks simultaneously. Most children did "time slicing" but soon realized this was nonsensical. More use of such real-world processes would be helpful.

For synchronization bugs, it may help to build on new conversational arrangements that children understand. Both FB and NL found the task easier when Resnick suggested that they add a third agent to coordinate the activities of the other two.


```
to walk-and-flash
ask 1 [walk]
ask 2 [flash]
end
```

process 3

This helped because they saw each process as having “one job.” That made more sense to them. NL called process 3 “the teacher.” FB imagined processes 1 and 2 as herself and her brother; process 3 was her mom.

It would also be helpful to have students “play process” as we have them “play turtle”—acting out the role of a process or agent. They could represent messages with pieces of paper that they deliver to one another. Each process might hold a box with a queue of messages received. They can learn what orchestrating a conversation would mean, but also learn that they, as processes, must still do one thing at a time. Switching roles as different processes and re-running the program would help them see how everything coordinates.

Finally, teachers should carefully and explicitly introduce the various types of objects. Again, “acting out” a program may also be helpful.

In the next column, we’ll look at another parallel processing version of Logo by Resnick that allows you to control thousands of objects. This new version, *StarLogo*, is currently available for people to use (*MultiLogo* is not). ▲

Douglas H. Clements, Professor at the State University of New York at Buffalo, has studied the use of Logo environments in developing children’s creative, mathematics, meta-cognitive, problem-solving, and social abilities. Through a National Science Foundation (NSF) grant, he developed a K–6 elementary geometry curriculum, *Logo Geometry* (published by Silver Burdett, & Ginn, 1991). With colleagues, he is working on the above-mentioned NSF research grant and is finishing a second NSF-funded project, *Investigations in Number, Data, and Space*, to develop a full K–5 mathematics curriculum featuring Logo. With Sarama, he is coauthoring new versions of Logo for learning elementary mathematics. One, *Turtle Math*, is available from LCSi.

Julie Sarama, PhD, is a research codirector of the NSF project 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. She is coauthor of *Turtle Math* and is currently designing and programming new versions of Logo.

The authors may be reached at:
State University of New York at Buffalo
Department of Learning and Instruction
593 Baldy Hall, Buffalo, NY 14260
Internet: Clements@acsu.buffalo.edu
Sarama@acsu.buffalo.edu

Time to prepare this material was partially provided by a National Science Foundation Research Grant, “An Investigation of the Development of Elementary Children’s Geometric Thinking in Computer and Noncomputer Environments,” NSF MDR-8954664. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

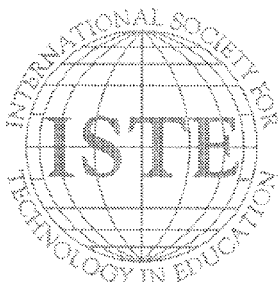
- Resnick, M. (1990). Multilogo: A study of children and concurrent programming. *Interactive Learning Environments*, 1(3), 153–170.



ISTE Books & Courseware Order Form

To order ISTE products advertised in this publication, find the product title in the following list and enter it on the form below.
To receive a free Resource Guide with a complete listing of ISTE products and services, please call our toll-free number, 800/336-5191.

Product (• Indicates an ISTE-published title.)	Member Price	Nonmember Price
101 Ideas for Logo	19.95	19.95
Children's Machine—Rethinking School in the Age of the Computer, The	12.00	12.00
• Exploring Recursion With LogoWriter		
Student Book (Specify Apple II series, Mac, or MS-DOS data disk)	10.75	11.95
Teacher Manual (Specify Apple II series, Mac, or MS-DOS data disk)	10.75	11.95
• Graphing With LogoWriter	9.85	10.95
• Introduction to MicroWorlds—A Logo-Based Hypermedia Environment	25.15	27.95
• Introduction to Programming in Logo Using Logo PLUS	18.85	20.95
• Introduction to Programming in Logo Using LogoWriter, 3rd Ed.	24.25	26.95
• Introduction to Programming Using Terrapin Logo for the Macintosh	19.80	22.00
• Logo Learning—Strategies for Assessing Content and Process	12.55	13.95
• Logo PLUS for Educators—A Problem Solving Approach	11.70	13.00
• LogoWriter for Educators—A Problem Solving Approach	11.70	13.00
• Math Activities Using LogoWriter (Specify Apple II series, Mac, or MS-DOS data disk)		
Patterns and Designs	18.85	20.95
Investigations	14.35	15.95
Numbers & Operations	16.15	17.95
High School Math	18.85	20.95
More Investigations	14.35	15.95
More Patterns and Designs	16.15	17.95
Probability and Statistics	12.55	13.95
• MicroWorlds—Hypermedia Project Development and Logo Scripting	31.45	34.95
• More Turtle Power—Text, Graphics, and Sound With LogoWriter (Specify Apple II series, Mac, or MS-DOS data disk)	18.85	20.95
• Turtle Power—Beginning Graphics With LogoWriter, Rev. Ed. (Specify Apple II series, Mac, or MS-DOS data disk)	20.65	22.95
• Turtle Time Designs—Working With Patterns in LogoWriter (Specify Apple II series, Mac, or MS-DOS data disk)	11.65	12.95
• Wacky Words for Traveling Tales—Logo Applications in Language Arts	15.25	16.95



Receive an additional 18% discount when ordering 10 or more of the same title of ISTE-published products.

Name _____ Membership # _____

School/Business _____

Address _____

City _____ State _____ Zip/Postal Code _____

Country _____ Phone _____

Shipping & Handling

\$0-\$15.99 (subtotal)	add \$4.50
\$16-\$45.99 (subtotal)	\$6.00
\$46-\$75.99 (subtotal)	\$7.00
\$76-\$100.99 (subtotal)	\$8.00
\$101 or more	8% of subtotal

GST Registration Number 128828431

Code LX3

ORDER

Quantity	Title	Member Unit Price	Nonmember Unit Price	Total Price

PAYMENT OPTIONS

- ☐ **Payment enclosed.** Make checks payable to ISTE—
International orders must be prepaid with U.S. funds or credit card.
☐ VISA ☐ MasterCard ☐ Discover Card Expiration Date _____
☐ **Purchase Order enclosed.** Please add \$4.00 for order processing—
P.O. not including \$4.00 fee will be returned.
☐ **C.O.D.** for U.S. book orders only. You will pay UPS the total upon delivery by check or cash—
ISTE will add \$4.50 order processing.
☐ **Airmail.** International orders for Books & Courseware are sent surface mail—
ISTE will bill you the additional shipping charge for airmail.
☐ Send me ISTE membership and subscription information.
☐ Send me a free ISTE catalog.

Deduct 18% on ISTE-published titles if ordering quantities of 10 or more of the same title

SUBTOTAL

SUBTOTAL

*Shipping and Handling (see box above)

*Add Additional 7% of SUBTOTAL if shipped to PO Box, AK, HI, or outside U.S.

Add 7% of SUBTOTAL for GST if shipped to Canada

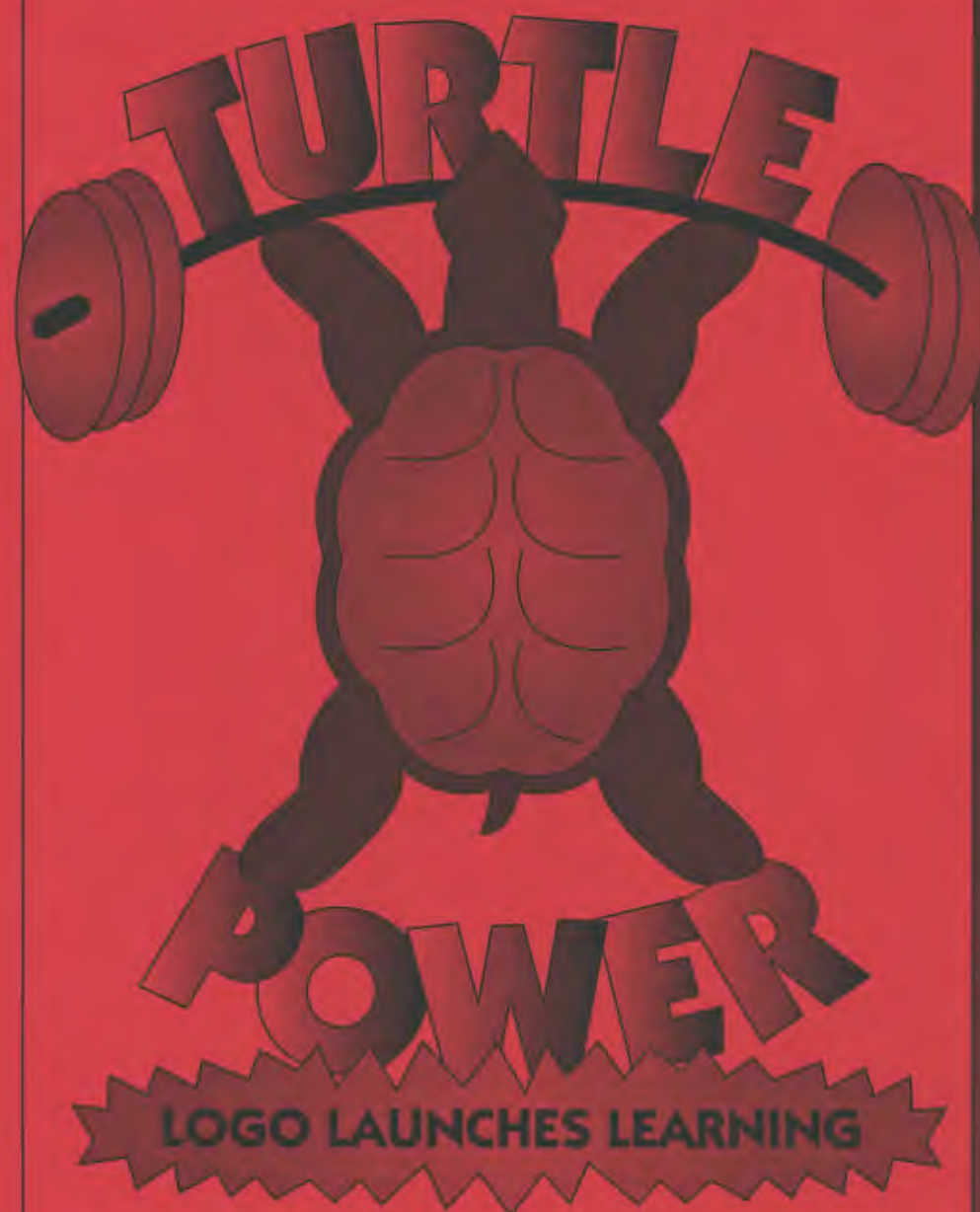
If billed with purchase order, add \$4.00; If COD, add \$4.50

* If actual shipping cost exceeds this amount,
we will bill you for the difference.

TOTAL

ISTE • 480 Charnelton Street, Eugene, OR 97401-2626 USA • Order Desk 800/336-5191 • Fax 541/302-3778

(Effective November 1, 1995, our area code changed to 541)



Many educators around the globe have been using turtle power in their classrooms with very positive results. As a beginning programming language, Logo is your logical choice.

ISTE's Logo in the Classroom series gives teachers fun, easy, and innovative methods for using Logo programming to improve the thinking and problem-solving skills of precollege students.

Logo concepts and tools can be used to teach a variety of subjects. The series offers interactive ways to use Logo for teaching mathematics, language arts, art, modeling and simulations, and computer science.



International Society for Technology in Education
Customer Service Office
480 Charnellton Street, Eugene, OR 97401-2626 USA
Phone: 800/336-5191 (US and Canada)
Phone: 541/302-5777 (International)
Fax: 541/302-5778 Fax-on-Demand: 800/991-0212
America Online: ISTE ComputerServe: 70014,2117
Internet: info@oregon.uoregon.edu
World Wide Web: <http://isteonline.uoregon.edu>

The ISTE Logo in the Classroom Series include the following:

101 Ideas for Logo

**The Children's Machine—
Rethinking School in the
Age of the Computer**

Exploring Recursion With LogoWriter

Graphing With LogoWriter

**Introduction to MicroWorlds—
A Logo-Based Hypermedia
Environment**

**Introduction to Programming in Logo
Using LogoWriter, Third Edition**

**Introduction to Programming in Logo
Using Logo PLUS**

**Introduction to Programming
Using Terrapin Logo for the
Macintosh**

**LogoWriter for Educators—
A Problem Solving Approach**

**Logo PLUS for Educators—
A Problem Solving Approach**

**Logo Learning—
Strategies for Assessing Content and
Process**

**Math Activities Using LogoWriter
Series:**

High School Math
Investigations
More Investigations
Numbers and Operations
Patterns and Designs
More Patterns and Designs
Probability and Statistics

**Turtle Power—Beginning Graphics
With LogoWriter, Revised Edition**

**More Turtle Power—Text, Graphics,
and Sound With LogoWriter**

**Turtle Time Designs—
Working With Patterns in LogoWriter**

**Wacky Words for Traveling Tales—
Logo Applications in Language Arts**

LOGO, LOGOWRITER, AND MICROWORLDS ARE REGISTERED TRADEMARKS OF LOGO COMPUTERSYSTEMS, INC., HILLSIDE SPRING, CT 06035
THE LOGO PLUS ARE REGISTERED TRADEMARKS OF TERRAPIN SOFTWARE, INC., CHAMPAIGN, ILL 61824

ISTE BRINGS THE WORLD OF TECHNOLOGY CLOSER TO YOU.

By drawing from the resources of committed professionals worldwide, ISTE provides support that helps educators like yourself prepare for the future of education.

ISTE members benefit from the wide variety of publications, specialized courseware, and professional organizations available to them.

They also enjoy exciting conferences, global peer networking, and mind-expanding in graduate level distance education courses.

So if you're interested in the education of tomorrow, call us today.

AL SOCIETY FOR



International Society for Technology in Education
Customer Service Office
480 Charnelton Street, Eugene, OR 97401-2626 USA
Phone: 800/336-5191 (US and Canada)
Phone: 541/302-3777 (International)
Fax: 541/302-3778
Internet: iste@oregon.uoregon.edu

WE'LL PUT YOU IN TOUCH WITH THE WORLD.