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

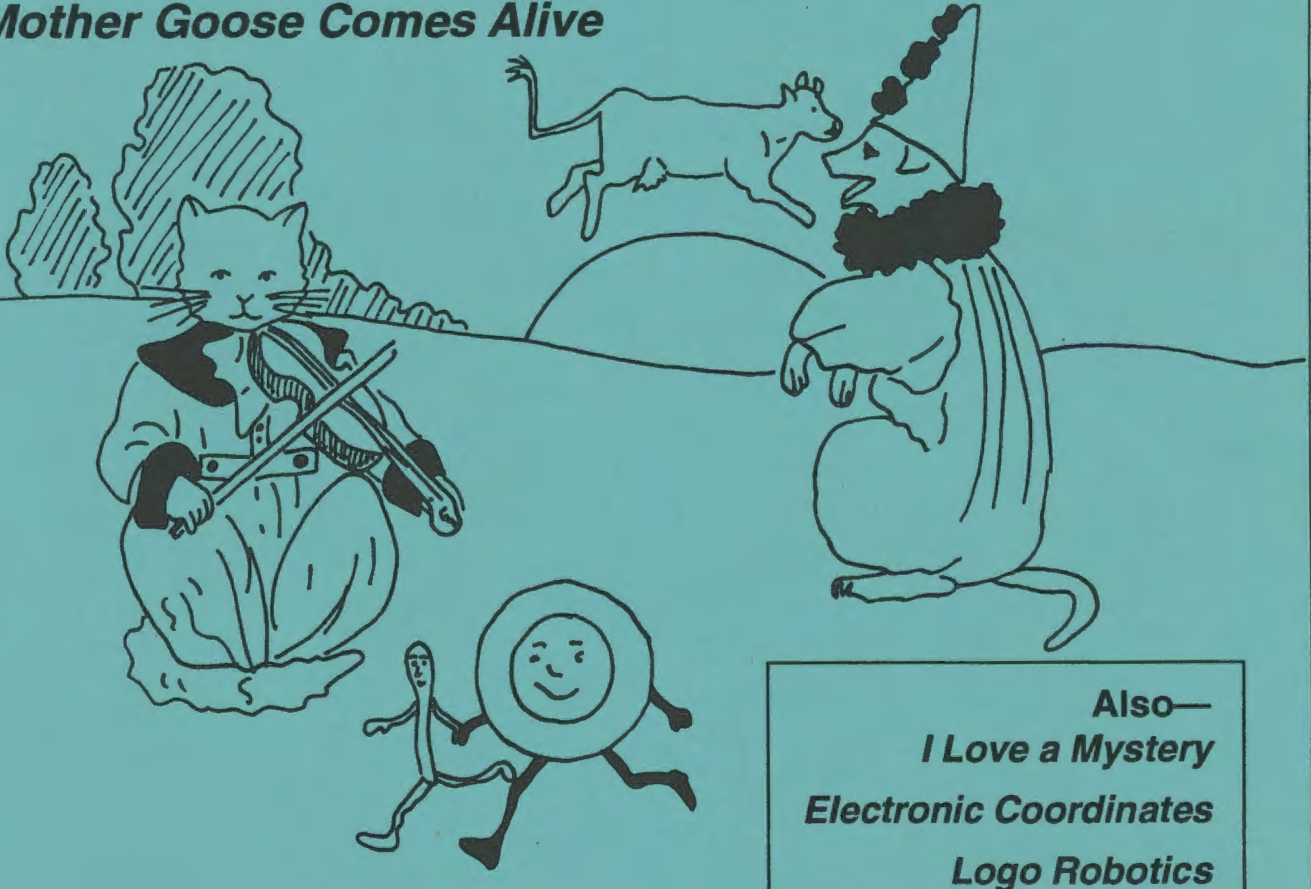
LOGO EXCHANGE

Winter 1993-94

Volume 12 Number 2

In this issue:

Mother Goose Comes Alive



Also—

I Love a Mystery

Electronic Coordinates

Logo Robotics

International Society for Technology in Education

LX Volume 12 Number 2

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. Articles appropriate to the International column should be submitted directly to Dennis Harper. Advanced articles should be submitted to Mark Horney, editor of the Extra for Experts column. Articles appropriate for the MathWorlds column should be sent directly to Sandy Dawson.

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

Send surface mail to:

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

Send electronic mail to:

Internet: YODER@oregon.uoregon.edu

Deadlines

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

Volume 13, Number 1	Feb. 1, 1994
Volume 13, Number 2	Apr. 1, 1994
Volume 13, Number 3	July 1, 1994
Volume 13, Number 4	Oct. 1, 1994

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

Individual ISTE Members may join SIG/Logo for $20.00. Dues include a subscription to *Logo Exchange*. Add $10 for mailing outside the USA. Send membership dues to ISTE. Add $2.50 for processing if payment does not accompany your dues. VISA, Mastercard, and Discover accepted.

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

Logo Exchange solicits articles on all topics of interest to Logo-using educators.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.

POSTMASTER: Send address changes to *Logo Exchange*, ISTE, 1787 Agate St., Eugene, OR 97403-1923. ISTE is a nonprofit organization with its main offices housed at the University of Oregon. ISSN# 0888-6970

LOGO EXCHANGE

Volume 12 Number 2 Journal of the ISTE Special Interest Group for Logo-Using Educators Winter 1993-94

Contents

From the Editor

Deep Learning and Problem Solving

by Sharon Yoder

Last summer I taught a sequence of five two-hour courses on desktop publishing. My students met with me for three to four hours every morning and did assignments in the afternoons or evenings, either in our lab or at home. I have often taught summer courses that immersed students in material for a week or so. However, this sequence of courses—one intense week after another—allowed students to build on deep knowledge from the previous week. We began with a course on graphics using *Aldus SuperPaint*. Next we learned *Microsoft Word*. The third week was devoted to advanced use of *Microsoft Word*. The fourth week was an introduction to *PageMaker*. The final week focused on design issues.

As I watched my students develop skills in using a variety of pieces of software at quite a high level, I was struck by the way in which their problem-solving skills grew. I generally began each class with a question-and-answer session in which students brought problems they had encountered in their work the previous day. At the beginning of the summer, I tended to be the "answerer" of nearly all of the questions. As the summer progressed I was less at the center of the problem solving. In fact, class often started late because groups of students "huddled," offering a variety of solutions to each other's problems.

The phenomenon that I saw in my summer desktop-publishing classes is what many of us want to happen in a Logo environment. We want our students to become better problem solvers and to see the amazing possibilities when using Logo, but somehow it is a rare environment in which that happens.

I'm quite certain that our frustration with the way in which some educators use Logo is not so different from the lack of understanding I see in people using applications programs. Let me cite an example on the use of style sheets from my summer class.

For those of you who do not know about style sheets, let me provide a brief description. The ability to define paragraph styles is built in to a number of powerful programs such as *Aldus PageMaker*, *Microsoft Word*, and *WordPerfect*. Style sheets allow the user to give a name to the formatting applied to a paragraph—typeface, type size, type style, margins, tabs, leading, etc.—and then apply that formatting with a mouse click or keystroke. A particular named style can be modified, causing all of the paragraphs with the named style to change. For example, if the headings are in Helvetica Bold Italic 24 point and you want them to be Avant Garde Bold 18 point, a simple change in the style definition causes the headings in the entire document to change—a tremendous time-saver.

The vast majority of users of software that includes style sheets have no idea that the feature is there. They methodically format every paragraph in their document time after time after time. Students in my class were amazed by the power style sheets gave them—but it took the better part of the week for them to be convinced that the initial effort in learning to create style sheets was worth the effort.

How often have you seen the same thing happen with Logo? You know that some task is incredibly easy and yet you see colleagues or students using the same time-consuming approach over and over in a manner that really doesn't contribute to their learning or problem-solving skills. Why? Most likely because these colleagues or students lack a depth of knowledge about Logo. Few teachers of Logo have had much training. Many have had only a day-long workshop. Yet those of us who have used Logo for years know that it takes more than a day—or even an entire course—to master

Logo. Such mastery takes time that few teachers have and training that few schools can afford.

Thus, it is not uncommon to find a "Logo expert" who knows only about turtle graphics, just as we see "applications experts" who have no idea how powerful many applications are. No doubt a lot of this occurs because the field of computing changes so fast. No sooner do users begin to understand a new application than a new version is introduced. We are moving so fast that we skim only the surface, never realizing the tremendous power that we have at our fingertips.

Perhaps we all need to catch our breaths, slow down a bit, and explore learning in depth. What can you do to help deep learning occur? You might take an intensive course on Logo or read a book that explores Logo at a deeper level. Look for ways to challenge yourself and your students. If you are involved in a restructuring effort in your school, think about ways in which you can provide deeper learning for yourself, your colleagues, and your students. Perhaps it is better to sacrifice learning a little bit about a lot of things for learning fewer things in some depth.

If you make an effort at deep learning, I think you will be as pleased with the result as my summer students and I were with their increased skills and problem-solving ability.

Sharon Yoder
Education 170C, DLIL
University of Oregon
Eugene, OR 97403
503/346-2190
yoder@oregon.uoregon.edu

Where Is the Spirit?

by Tom Lough

My flight got into Hartford late the night I found the Spirit. I was tired and sleepy. As I hopped aboard the parking-lot shuttle bus, I was totally unprepared for what was about to happen.

Completely unsuspecting as the bus pulled away from the terminal, I thought I saw the tail of a Delta airliner in the shadows just beyond the chain link fence. Even though I was about to fall asleep on the bus seat, my eyes automatically searched through the foggy darkness for the number on the tail, as I had done to what seemed like hundreds of Delta aircraft over the past several years. You see, I was in search of the Spirit.

Let me explain. For Father's Day back in 1985, Posy and Kyser gave me a book by Tom Peters and Nancy Austin called *A Passion for Excellence* (published by Random House in 1985). I really enjoyed the book. But when I got to page 204, I read a story that would change my life.

I learned that, in 1982, the 25,000 employees of Delta Air Lines bought a $30 million airplane and gave it to the company! I had to find out more. So, I began asking Delta employees about it whenever I had a little extra time in an airport. I discovered that, during the 1981-1983 recession, the employees had been searching for a way to express their gratitude to Delta when someone suggested that they give the company an airplane. Suddenly, no one was laughing. "Let's do it!"

Over the next several months, they organized a company-wide voluntary contributions program, complete with payroll deductions. Then, in December of 1982, they rolled a 236-passenger Boeing 767 airliner decorated with a giant red ribbon out onto the tarmac at the Atlanta airport and presented it to Delta management.

This was totally unprecedented—employees giving a gift of major proportions to their company as an expression of their gratitude! A determination began to grow within me. I wanted to find that plane!

Soon, I discovered that the plane was named The Spirit of Delta, the only airliner in Delta's fleet with a moniker. It was uniquely decorated with gold trim around the Delta logo, and a special plaque inside the cabin testified to its unique history. The tail number (by which all aircraft are identified) of The Spirit of Delta was 102.

I found that I could ask Delta employees at a computer terminal on any day where airliner 102 was going, and they would proudly look it up for me. Sometimes 102 was flying up and down the west coast. On other occasions it was flying cross country, but nowhere near my location.

Although I was disappointed that I never caught up with 102, I certainly enjoyed my interactions with the spirited Delta employees. Eventually, I came to realize that, even if 102 eluded me forever, I had really found what I was seeking; the stories told to me about that plane and what it meant came straight from the hearts of dozens of flight crew and ground crew members alike.

Meanwhile, back on the Hartford airport shuttle bus, I did a weary double take. Did I see correctly in the semidarkness? Was the number on the tail of that Delta airliner parked next to the fence 1 - 0 - something?

Suddenly, I was wide awake! I couldn't wait to get to my car and zip back to the short-term parking area near the terminal for a better look. Then I saw it. 102 had found me very quietly, late on that dark night.

I dashed inside the terminal, found Joe Aiello, a member of the Hartford Delta ground crew, and told him excitedly about my quest. Holding my breath, I asked if I could go aboard. Even though it was approaching midnight and the airport was shutting down, he said he would see what he could do.

As I waited, I took the opportunity to examine the aircraft in detail from the observation window in the waiting area. Yep, there was the gold trim around the "widget" (the triangular Delta logo). The Spirit of Delta name was painted in gold on her nose and on her midsection.

I heard someone approaching and turned to see a senior pilot coming down the hallway from a just-arriving late-night flight. Not even aware that I was watching him, he looked over at 102 as he passed, nodded in her direction, and acknowledged her aloud with, "The Spirit." His touchingly reverent personal testimony summarized my search.

Soon, Joe came back and proudly gave me a brief tour of 102. At the door, I paused to feel her metallic skin. I went inside and read the plaque mounted in front of seats 1A and 1B:

The Spirit of Delta
This aircraft was provided by Project 767, a voluntary program conducted by active and retired personnel of Delta Air Lines to express

the pride and gratitude felt for their company. Introduced into passenger service December 15, 1982.

All too soon, I had to go back into the terminal. But I had seen and felt enough. I had found The Spirit of Delta.

On the way home that night, I thought about another quest I had been pursuing for many years, a quest for the spirit of Logo. No, I knew I could never expect to find a turtle with "102" painted on its shell! But I have sensed a special spirit of excitement and wonder in the teachers and students I have seen working with this marvelous computer language. That's what I am looking for!

I expect to continue this rewarding quest for many more years.

FD 102!

Tom Lough
Founding Editor
PO Box 394
Simsbury, CT 06070

PS: Are you looking for the spirit of Logo, too? I'd love to hear your stories. Also, why not plan to come to the National Educational Computing Conference (NECC '94) in Boston, June 13-15. There will be plenty of spirited Logo-related activities going on!

There were several errors in the Fall 1993 Quarterly Quantum Column. The correct text follows:

Sooner or later, I began thinking about the FIRST command. (Incidentally, I also wondered if FIRST was the first Logo command developed. Any Logo historians out there?)

FIRST reports the initial element of a word or list. Here are a few examples.

```
SHOW FIRST "LOGO L
SHOW FIRST [LOGO IS DELIGHTFUL]
LOGO
```

The FIRST command can also be chained with other FIRST commands.

```
SHOW FIRST FIRST [LOGO IS DELIGHTFUL]
L
```

A First Course in Programming

in Terrapin Logo, LogoWriter, and PC Logo

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

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

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

Curriculum written BY teachers FOR teachers!

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

Logo Ideas

A Case of Curiosity

by Eadie Adamson

"...if man's intellectual excellence is the most his own among his perfections, it is also the case that the most personal of all that he knows is that which he has discovered for himself. How important is it, then, for us to encourage the young to learn by discovery?"
Jerome Bruner (1979)

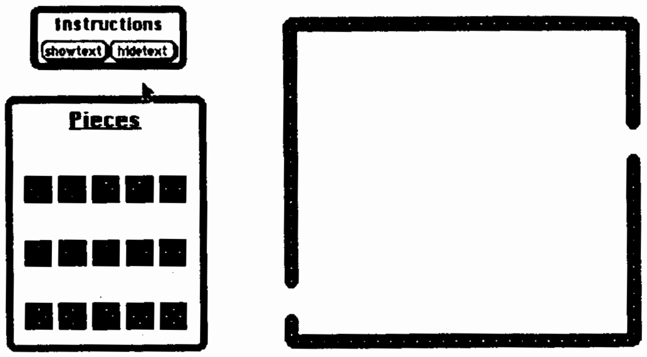
About Thinking and Being Curious

I remember when *LogoWriter* was new. I was showing some teachers a project that happened to put text in the Command Center instead of on the screen. One of them asked, "How do you know to do that? The books don't tell you about that." My answer was, "I looked at some of the samples. I saw that there was text in the Command Center and wondered how it was done, so I looked at the procedures."

Wanting to know how things work often takes a little work, but it's an excellent way to learn. I thought about that recently as I looked at some of the projects from *MicroWorlds Project Builder*.

Re-creating a Project in Another Size, or How Did I Get Myself Into This?

It all started when I looked at the MazeMaker project in the Going Further folder that comes with the new *MicroWorlds Project Builder*. Clicking on a button displays instructions; clicking another button hides the instructions. The project contains a set of maze-making pieces. To make a maze, you click on a shape and a copy of it pops up. You drag the shape to the place where you want it on the screen, click on it again, and it makes a minor adjustment into place. Continue this process and you can build a maze fairly easily.



I was working with a young girl who had limited vision. I thought the MazeMaker would be a nice project for her to use, but the screen size was much too small for her large monitor. The small pieces would also be a little hard for her to see. I obviously needed to enlarge the project! Since a page can't be duplicated if the new *MicroWorlds* project size is different, I knew I had to more or less start from scratch. I had to investigate how it all worked before I could decide how to enlarge and adapt the project for my special-needs student.

The experience seems worth sharing. Enlarging a project presents an interesting way to teach features of *MicroWorlds* to students who already have some Logo knowledge. Give them a project from the Going Further folder to explore. Their first task is to find out how it works. Their second task is to re-create the project for themselves, making changes if they wish. In the case of the Project Builder, samples are on smaller screen sizes. If all your computers have large screens, there are some nice projects just waiting for your students! Make all the projects the right size for your screens. This means that most projects will need to be re-created. Use the need to re-create them to set up an environment for exploring and sharing the discoveries. At the same time, ask students to add ideas of their own to the projects. Your students have a different way, with excellent motivation, for learning how to use *MicroWorlds*.

The Mystery of Very Few Procedures

After making part of a maze, I switched to the Procedures page to look at the procedures. I was curious about how it worked. Surprise! There were only two procedures—they will be included in the second part of this article. Neither revealed very much about the processes I had been using. What was going on here?

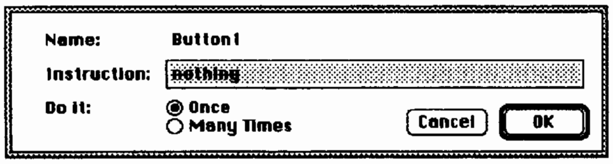
With other versions of Logo and also with other programming languages, it's possible, given a little knowledge of the language involved, to use a listing of a program and figure out what it does. With *MicroWorlds* Logo, this strategy doesn't work. Some projects do a lot, yet have no procedures at all. What can you do?

MicroWorlds Logo, because it is a programmable environment, has many facets that need to be explored in order to get the "full picture" on a project. Since the procedures alone often barely begin to tell the story, we need to know what other parts of the program to investigate in order to deduce how things work, or in the case of a buggy program, how they fail to work! This will be true not only for the problem of enlarging someone else's project but for debugging and evaluating student work or even our own projects.

Tools for Exploring

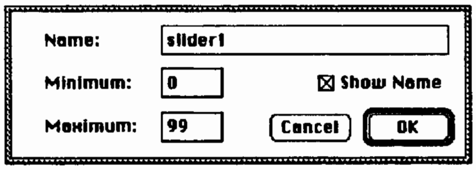
Here's a list of exploring tools you might use with *MicroWorlds* projects:

- Procedures page
  Look at this first. It will give some idea of what's going on. If there are no procedures at all, you'll *know* you need to look at other parts of the project.
- Buttons
  Buttons can do a lot. If there are buttons on a page, click on the button-maker on the tools palette, then click on a button. Its dialog box will come up and you can read the instructions. Notice also whether the instruction runs "once" or "many times." Which box is checked?



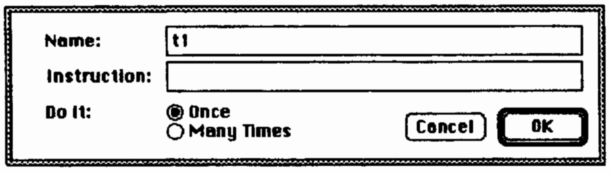
Name: Button1
Instruction: nothing
Do it: [✓ Once] Many Times

- Sliders
  Click on the slider-maker and then on a slider to read its instructions. Notice the maximum and minimum values that have been set. They could be important.



Name: slider1
Minimum: 0
Maximum: 99
☒ Show Name

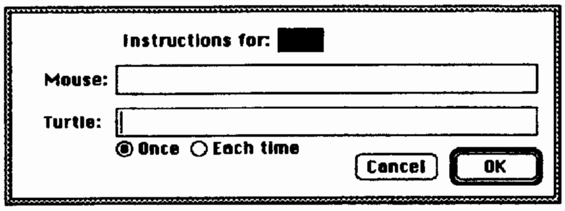
- Turtles
  See if a turtle has been programmed. Click on the turtle-hatcher in the tools palette and then click on a turtle. Look for a message in its dialog box. Notice whether "once" or "many times" is checked.



Name: t1
Instruction:
Do it: [✓ Once] Many Times

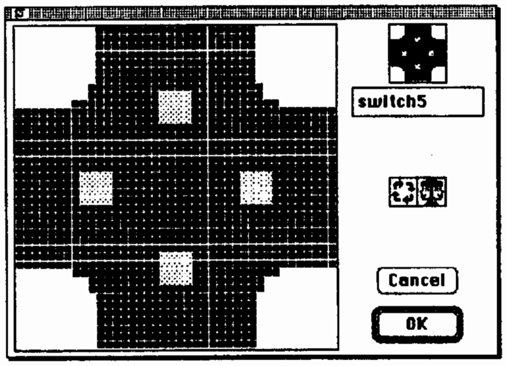
Also notice the turtle's name. It may have been changed from "t" followed by a number to a word. This is a feature not used too often in the projects, but you can give every turtle a unique name rather than the letter "t" and the turtle's number—just click on the name box and change it.

- Colors
  Get the drawing tools. Double-click on each color to see if it has been programmed. Has it been programmed for the mouse, the turtle, or both? For "once" or "each time"? Notice that you can only program a color once. The instructions affect the entire range of that color from light to dark.

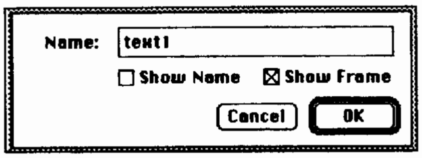


Instructions for: ■
Mouse:
Turtle:
[✓ Once] Each time

- Shapes
  Check the names of the shapes in the project by double-clicking on them to bring up the shape editor.



- Page names
  Because typing a page name is the same as giving the command to get the page, page names may give you a clue about a button or an obscure-sounding instruction line in a procedure or dialog box.
- Textbox
  Click on the textbox tool and then on a box to look at its dialog box. Notice the name of the textbox here. It may be addressed in a button or in a procedure or other instruction.



Name: text1
☐ Show Name ☒ Show Frame

- Help
  Pull down on the Help menu and choose Help, or just press Command-H. The mouse pointer turns into a question mark. Put the pointer on a spot you are curious about—a box, a turtle, a color. A balloon that pops up may give you more information. Add this idea to your list of things to do for a well-designed project. See the "How To" booklet for help on editing the balloons.
- Vocabulary
  Choose Vocabulary from the Help menu. Use it to look up any unfamiliar terms. If the word you need is not on the list, press Command-F to get the extended vocabulary list. Each term has an explanation and an example of its use. You can copy instructions and paste them in the Command Center or on the procedures page to try them out.

To be continued in the next issue... Meanwhile, happy exploring!!

Reference

Bruner, Jerome. (1979).The act of discovery. In *On knowing: Essays for the left hand*. Cambridge, MA: Harvard University Press.

Eadie Adamson
1199 Park Avenue, Apt. 3A
New York, NY 10128
AppleLink: EadieA

When You Are Really Serious About Logo...

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

New PC Logo 4.0 features include:

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

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

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

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

PC logo™
Version 4.0

Print Page: A *LogoWriter* Tool Procedure for Printing *LogoWriter* Programs

by Charles E. Crume

I have been working with *LogoWriter* for some time now. Recently, I have wished that *LogoWriter* would allow me to print the page name and the current date at the top of each printed page or *LogoWriter* code. Other word-processing programs have headers and footers as standard features of their programs. (The PRINTTEXT and PRINTTEXT80 primitives do not have this feature.) This article presents a *LogoWriter* tool procedure that prints the page name, the date, and a child's name (or other information) at the top of each printed page. The procedure is called PP (short for Print Page).

I believe this is important in classroom use because many teachers want to use *LogoWriter* as an alternative to other word-processing programs. Because schools are often limited in the amount of money they have allocated to computer software or because teachers may not want to invest class time in teaching a variety of programs, educators look for programs that best meet the limited funds and time they have. *LogoWriter* is capable of meeting these needs, but tools such as these must be added and adapted.

Unfortunately, *LogoWriter* does not include primitives for reading or writing individual lines of a text file (LOADTEXT and SAVETEXT load and save the entire file). Nor does *LogoWriter* provide any means for selectively printing text on the printer. (PRINTTEXT and PRINTTEXT80 print all text on the page.) PP, therefore, cannot simply print the page name, date, and child's name at the top of each printed page and then print lines of text from the entire *LogoWriter* page—repeating this process until the entire *LogoWriter* page is printed. Some other mechanism must be found.

The mechanism used (i.e., PP's theory of operation) is as follows:

a. Load the *LogoWriter* page with a program to be printed into memory. This implies that the current page is named in such a way that the GETTPAGE primitive will not return an error.
b. Flip to the "flip" side (the side of the page containing the *LogoWriter* program instructions).
c. Count the number of lines in the program.
d. Insert the page name, date, and child's name (or other information) every 42 lines. The PRINTTEXT and PRINTTEXT80 primitives print 50 lines of text on each printed page; the information contained in PP uses 8 of those lines.
e. Print the page on the printer using PRINTTEXT. PRINTTEXT80 does not work properly in all cases because of the way it processes carriage returns within the *LogoWriter* instructions.

Although PP inserts text into the *LogoWriter* program to be printed, *it does not make changes to the page on the disk drive!* After PP finishes printing the page, it executes a LEAVEPAGE primitive, which removes the changed page from memory and returns to the CONTENTS page (LCSI, 1988). This ensures that the original *LogoWriter* page is not altered. To ensure no changes could be made, the *LogoWriter* page to be printed could, of course, be made to be read only via the LOCK primitive (LCSI, 1988), but this is not necessary. PP uses the subprocedures PP.LINES and PP.PROCESS. The source code is as follows:

```
TO PP : PAGE : NAME
GETPAGE :PAGE
IF FRONT? [FLIP] ]
TOP
PP.PROCESS PP.LINES TEXTPOS
PRINTTEXT
LEAVEPAGE
END
```

```
TO PP.LINES : PP.CURRENT.POS
CD
IFELSE TEXTPOS > : PP.CURRENT . POS
   [OUTPUT 1 + PP.LINES TEXTPOS]
   [OUTPUT 1 ]
END
```

```
TO PP.PROCESS : LINES
TOP
REPEAT (INT : LINES / 4 2 ) + 1
   [(PRINT [PAGE -] WORD :PAGE ".
   LWR)
   PRINT [ ]
   (PRINT [DATE -] DATE)
   PRINT [ ]
   (PRINT [NAME -] : NAME)
   PRINT [ ]
   PRINT [
   ******************************************
   ]
   REPEAT 42 [CD]
   SOL]
END
```

PP requires two inputs. The first input is a word specifying the *LogoWriter* page containing the program to be printed. The second input, which can be either a word or a list, is usually the child's name, although any other information can be specified. The date is retrieved automatically from the computer system via the DATE primitive (Crume, 1990). Note that DATE is not available in all versions of *LogoWriter*. Omit the "date" lines if your version of *LogoWriter* cannot access the date.

Both PP and DATE must be loaded as tool procedures (Crume & Maddux, 1989; LCSI, 1988). The first thing PP does is to load the *LogoWriter* page to be printed into memory. If PP is loaded as a normal *LogoWriter* page, the GETPAGE primitive within PP will cause *LogoWriter* to save PP to disk and load the specified *LogoWriter* page, thereby replacing PP. This, of course, will prevent PP from finishing its task.

The following example shows how to use PP to print a *LogoWriter* page:

```
GETTOOLS "PP
GETTOOLS "DATE
PP "HOUSE "CHARLES
```

The GETTOOLS primitives load PP and DATE as tool procedures. The last command invokes PP and instructs it to print a page named HOUSE and to print CHARLES as the user name. The next example specifies a list for the second input to PP:

```
PP "CLOCK [LOGOWRITER PROGRAM THAT
   KEEPS ACCURATE TIME]
```

The GETTOOLS primitives do not need to be executed this time because after tool procedures are added, they remain in memory throughout a *LogoWriter* session (LCSI, 1988).

PP can be somewhat slow, especially for lengthy *LogoWriter* pages or if an original IBM PC running at 4.77 MHz is being used. It can, however, be interesting to watch as PP performs its task.

I hope that teachers and students will take advantage of this useful tool for *LogoWriter*. Learning to look for tools and to adapt them to individual needs demonstrates the kind of problem-solving skills Logo develops.

References

Crume, C E. (1990). Date, time, palette, and disk space: four new primitives for *LogoWriter* on the IBM/PC. *Logo Exchange, 9*(5), 28-9.

Crume, C. E., & Maddux C . D. (1989). Adding a set of alternative conditional primitives to *LogoWriter*. *Logo Exchange, 8*(1), 26-27.

LCSI (Logo Computer Systems, Inc.). (1988). *LogoWriter reference guide.* Vaudreuil, Quebec: Author.

Charles Crume is a frequent contributor to the *Logo Exchange*. He enjoys developing tools that make Logo useful and exciting.

Charles Crume
810 Matson Place, Apt. 504
Cincinnati, OH 45204

Mother Goose Comes Alive

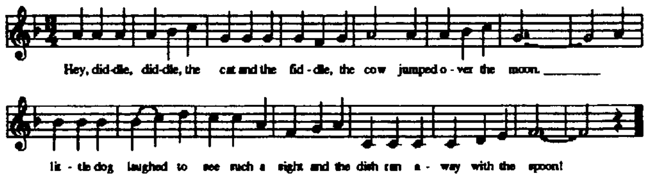
by Dorothy Fitch

Word-processing programs allow you to write text; some let you add graphics, too. Graphics packages help you draw pictures, but most don't let you animate them. But with Logo (most of the newer versions, anyway), you can write text, make shapes, create animations, and add music! In this column, we will combine all these fun features by animating a nursery rhyme.

Choosing a Project Idea

Pick something easy. (Watch out! Even a simple-sounding idea can get complicated in a hurry.) Find a poem or rhyme that uses nouns that you can create as shapes. If the rhyme has a tune, all the better—you can add music as well. Check out your old Mother Goose books. Her rhymes offer many possibilities!

For our sample project, we will use "Hey Diddle Diddle," which goes something like this:


Hey, did-dle, did-dle, the cat and the fid-dle, the cow jumped o-ver the moon.____
lit - tle dog laughed to see such a sight and the dish ran a - way with the spoon!

What does this rhyme offer? Some shape possibilities (cat, fiddle, cow, moon, dog, dish, and spoon), a tune (anonymous composer), some action (jumped, laughed, and ran away), and it's not too long!

I used Terrapin's Logo PLUS for the Macintosh for this project. Even if your version of Logo does not offer shapes, text on the graphics screen, and music, you can still combine the features you *do* have to create an impressive production.

The Music

This project has many different parts to work on. It doesn't matter where you start. I began with the tune. You need to know the command to play a note (NOTE in Terrapin Logos, TONE in other versions) and the numbers to produce the correct pitches and durations. It also helps to be able to read music, but I'll help you with the notes!

The tune is made up of two phrases, so it is logical to write a procedure for each of these musical sentences. Using note-name procedures makes composing easy—you don't have to type the pitch number for each note, just its name. Later, if you decide to start on a different note or change the octave, you have to modify only a few procedures. Check your Logo documentation for pitch values for other notes. Doubling a pitch number makes it sound an octave higher (compare C and HIGH.C); dividing the number in half makes it sound an octave lower.

The procedure Q provides the length of a quarter note, the basic beat of the tune. Every note uses Q to specify its duration. A half note is Q* 2; the long notes are Q* 5. To speed up the piece, make the number Q reports smaller. To slow it down, make the number bigger.

The following procedures play the tune. The main procedure is HEY.

```
TO HEY
PHRASE1
PHRASE2
END

TO PHRASE1
NOTE A Q
NOTE A Q
NOTE A Q
NOTE A Q
NOTE B.FLAT Q
NOTE C Q
NOTE G Q
NOTE G Q
NOTE G Q
NOTE G Q
NOTE F Q
NOTE G Q
NOTE A Q * 2
NOTE A Q
NOTE A Q
NOTE B.FLAT Q
NOTE C Q
NOTE G Q * 5
END

TO PHRASE2
NOTE A Q
NOTE B.FLAT Q
NOTE B.FLAT Q
NOTE B.FLAT Q
NOTE B.FLAT Q
NOTE HIGH.C Q
NOTE HIGH.D Q
NOTE HIGH.C Q
NOTE HIGH.C Q
NOTE A Q
NOTE F Q
NOTE G Q
NOTE A Q
NOTE C Q
NOTE C Q
NOTE C Q
NOTE C Q
NOTE D Q
NOTE E Q
NOTE F Q * 5
END

TO Q
OUTPUT 15
END

TO C
OUTPUT 262
END

TO D
OUTPUT 294
END

TO E
OUTPUT 330
END

TO F
OUTPUT 349
END

TO G
OUTPUT 392
END

TO A
OUTPUT 440
END

TO BFLAT
OUTPUT 466
END

TO HIGH.C
OUTPUT 523
END

TO HIGH.D
OUTPUT 587
END
```

The Shapes

Now for the shapes. The words of the rhyme tell me what I need. I used the shape editor to create my not-very-artistic-but-more-or-less-recognizable shapes.

SETXY or SETPOS instructions help to position the shapes on the screen (or you could use FORWARD, LEFT, and RIGHT commands). Multiple turtles are useful, too—one for each shape and one to print the text. The programming isn't complicated. It just takes some time to put your shapes where you want them and get them to move in just the right way, for after creating the shapes, the next trick is to animate them.

In my version, the fiddle shape changes its orientation, making it appear as if the cat is picking it up. Logo PLUS for the Mac's shapes rotate, so I just turned the fiddle 90° for the new shape. Of course, the cat doesn't have to pick up the fidde at all! Or you could create two different fiddle shapes.

The cow moves in a half-circle as it jumps over the moon. I locked the cow's heading so that when it moves and turns, it doesn't rotate. If your shapes don't rotate, then you are all set—your cow will stay pointed in the same direction. My original moon was just a circle, but my colleagues at Terrapin thought that one with a face would add character. So I made another shape. (Be prepared: people will always have suggestions for improving your program! Easy for them to say what would be nice, isn't it?)

To make the laughing dog, I used two shapes: one standing on four legs and the other raised up on its hind legs. I set the pen to draw from the hind legs so that when I change from one shape to the other the dog stays in the same location. (A shape both draws and turns from the pen's location.) When I switch from one shape to another, the dog appears to jump up and down. Since I have never heard a dog laugh, I can only assume that this is a reasonable action for a happy dog.

The dish and the spoon exit to the right off the screen as they run away together.

The shapes take some time to draw, but once they are done, you can use them in any project—if you ever have another need for a cow or a dish!

The Text

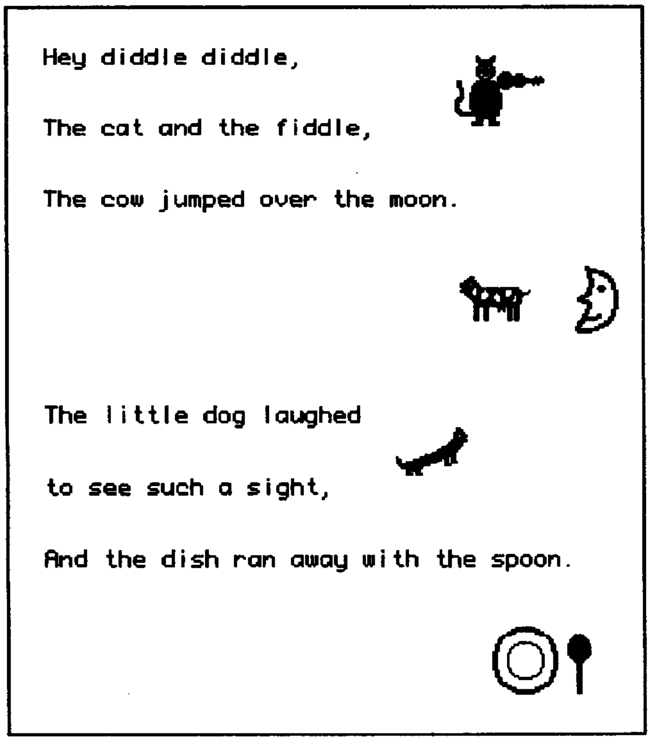
Adding the words for the rhyme can be very simple or very complicated, depending on how you present them. You can print text to the textscreen (easy); print the words to the graphics window, usually at the turtle's location (medium); or make each word—actually, each syllable—appear at the same time as its musical note is played (hard). See the following sections for more ideas.

Putting It All Together

Now that you have all the pieces—text, shapes, and music—how do you put it all together? It would be nice to have the words, music, and animation all happening at the same time, but the computer can only do one thing at a time. And Logo is not too speedy even on the best of machines. Therefore, you may not want to present everything at once.

You could present the words, play the tune, and then finally show the animation, one event after the other. Or you could present the words together with the animation, followed by the music. After you have all the pieces, you can experiment with the final presentation. My version prints the text and animates it one line at a time and then plays the song at the end.

Here is the final scene from my way-off Broadway Logo production of "Hey Diddle Diddle."



```
Hey diddle diddle,

The cat and the fiddle,

The cow jumped over the moon.



The little dog laughed

to see such a sight,

And the dish ran away with the spoon.
```

Publishing Your Project

When you finish your project, you'll want to publish it. I'm lucky! I can just send my projects off to *Logo Exchange* as part of my column. (*LX* might publish yours too if you send it to the editor!)

But there are other ways to "publish" a project. You can run the program for your friends to watch. You can print a screen that shows the words and your shapes. You can send a disk with your program to someone who uses the same version of Logo. You can share the instructions you wrote on a printout. You might even be able to videotape your production.

I can't share my finished project on this silent printed page. However, a Hey Diddle Diddle file comes with Logo PLUS for the Macintosh as a demonstration program. And if you want a printout of the program to help you work on your own version, send a self-addressed, stamped envelope to me at the address at the end of the column and I'll mail you a copy.

Now you are ready to animate your own rhyme. Here are some rhyme ideas that come to mind. Check out a children's song book or poetry book for further inspiration.

- Little Miss Muffet
- Jack and Jill
- Humpty Dumpty
- Mary Had a Little Lamb

I would imagine that any of these would make a great group project, because there are so many independent parts that can be combined near the end. Happy Logo adventures!

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

Dorothy Fitch
Terrapin Software, Inc.
400 Riverside Street
Portland, ME 04103-1068

CompuServe: 71760,366
Internet: 71760.366@compuserve.com

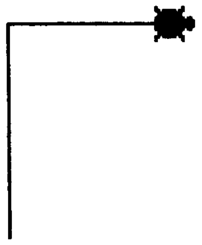
Pythagorean Theorem

by Valerie Rosenthal

The Pythagorean Theorem is one of the most famous and useful ideas in geometry. It expresses the relationship of the three sides of a right triangle. The Logo environment can be used to help students state what the theorem is and gain a better understanding of its purpose. This environment can also allow students to explore many different sizes of right triangles and how the variety of sizes affects the outcome. The following Logo activity can be used with any student who is studying geometric concepts.

First, the student should be introduced to the Pythagorean Theorem and how it has been stated for centuries: "The square of the hypotenuse is equal to the sum of the squares of the other two sides." Initial Logo exploration might have students visually construct the length of the legs of the right triangle. Through trial and error, students could then find the length of the hypotenuse and the angle the turtle needs to turn. For example, the student might first draw the following:

```
FORWARD 80
RIGHT 90
FORWARD 60
```



The student would then need to figure out the angle to turn next and the length of the hypotenuse. He or she would keep trying until the sides met to form a triangle.

Another way to state the Pythagorean Theorem is "Leg squared plus leg squared equals hypotenuse squared." This definition can assist the student in using Logo to help find the length of the hypotenuse.

```
TO HYPOTENUSE
FORWARD SQRT ((:LEG1 * :LEG1) +
   (:LEG2 * :LEG2))
END
```

LEG1 and LEG2 can be inputs determined by the user or by Logo through random choice. Procedures to randomly pick the legs of the right triangle might look like this:

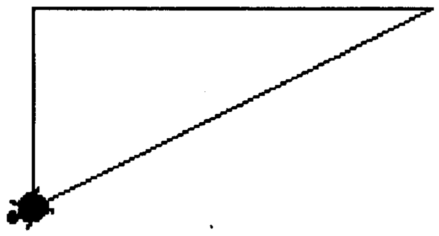
```
TO PICK.LEG
OUTPUT RANDOM 150
END

TO FIRST.LEG
MAKE "LEG1 PICK.LEG
END

TO SEC.LEG
MAKE "LEG2 PICK.LEG
END
```

Random selection allows the student to see a variety of different-sized right triangles and how the lengths of the legs directly affect the angle and the length of the hypotenuse. It is the angle that will baffle many students. This is the perfect time to introduce the tangent ratio, or, in Logo, ARCTAN, to figure out the angle. ARCTAN is the angle obtained from the ratio of the opposite leg divided by the adjacent leg of the acute angle. The turtle will have to turn 180 degrees minus the angle of the triangle that we found by using ARCTAN:

```
RIGHT 180 - ARCTAN (:LEG1 / :LEG2)
```

The following procedure puts together all of the parts described above.

```
TO RIGHT.TRI
FIRST.LEG
SEC.LEG
FORWARD :LEG1
RIGHT 90
FORWARD :LEG2
RIGHT 180 - ARCTAN (:LEG1 / :LEG2)
HYPOTENUSE
END
```



This procedure allows the student to see instantly the size of the triangle and how the lengths of the legs affect the angle and the hypotenuse. The lengths of the legs that the computer picked can be seen by typing

```
PRINT :LEG1
PRINT :LEG2
```

The procedure also gives the student immediate feedback on whether the angle or hypotenuse are correct. The student will be able to see immediately how the triangle looks rather than just seeing numbers stuck into a formula.

Later on, as the special right triangles of 45-45-90 and 30-60-90 are introduced, the relationship between the sides and the angles can be explained in greater depth. You might try this for a 45-45-90 triangle.

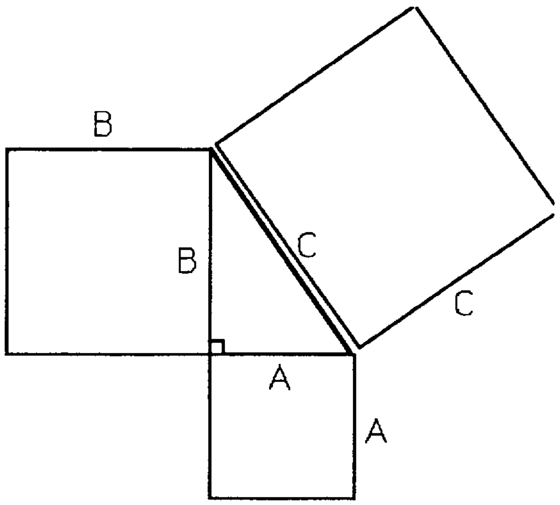
```
TO 45.45.TRI :LEG
FORWARD :LEG
RIGHT 90
FORWARD :LEG
RIGHT 135
FORWARD :LEG * SQRT 2
END
```

Note that RIGHT 135 is obtained from 180° - 45°.

Or you can create a procedure for a 30-60-90 triangle.

```
TO 30.60.TRI :SHORTLEG
FORWARD :SHORTLEG
RIGHT 90
FORWARD :SHORTLEG * SQRT 3
RIGHT 150
FORWARD :SHORTLEG * 2
END
```

Students can expand on this idea if they are given a different side of the triangle to start their drawing and asked to speculate on how their given information would affect the order of commands and the calculations of the angles. The students might also want to explore the proof of the Pythagorean Theorem by showing the following figure on the Logo screen:



Students of all ages can use this type of exploration of right triangles in Logo. They need only a little introductory information on Pythagoras and his theorem to get them started. The Logo environment allows them to see the relationship of the sides and how powerful the Pythagorean Theorem really is for right triangles. It is an interesting way to make a theoretical, formal representation quite concrete.

Valerie Rosenthal has taught high school mathematics for eight years. Presently she teaches geometry and general math at Elkhorn High School in Elkhorn, Nebraska. She has also taught in Marshalltown, Iowa, and Boys Town, Nebraska. Valerie earned her Bachelor of Science degree from Dana College in Blair, Nebraska in 1984 and her Master of Science from the University of Nebraska in Omaha in 1992. Her master's degree is in secondary education with an emphasis on educational technology. She can be reached at

13063 Patrick Circle
Omaha, NE 68164

I Love a Mystery

by Robert Macdonald

American classrooms are filled with diverse talents. Finding ways to stimulate as many children as possible with a minimum of teacher involvement is highly desirable. Perhaps this microworld may be as helpful to you as it is to me.

I have been fortunate in the past few years to have inherited a goodly number of fourth graders who have fine typing skills and who work very well independently. In addition, a computer lab in our school has been in operation for the past year, but some scheduled time was very much underutilized. How could reliable and capable students use these hours that no one else found convenient to use? The obvious answer was a guided and open-ended, yet independent, computer assignment.

One of my first independent projects—and surprisingly the most successful—was a language microworld called mystery. The main procedure describes a mysterious situation. Some of the main words in the message displayed on the screen are the names of procedures that contain clues to the mystery itself. One of those procedures will give a clue that will solve the mystery for you. If you type a word that is not the name of a procedure, Logo simply replies, "I don't know how to ...."

In most versions of Logo, the print statement can take a large number of words. For example:

```
print [Now is the time for all good
   computers to come to the aid of their
   confused owners.]
```

If the students have no programming experience, you'll need to spend some time teaching about procedures (Yoder, 1990). Setting up a simple procedure is an easy enough task to explain to 10-year-olds. Help them remember the two reserved words: the preposition TO and the word END. In addition, I find a "protecting" line helpful.

```
if not front? [flip]
```

Because we will be using a **clearpage** procedure, this line is a gift from the gods.

I've found that the easiest way to present this microworld to a class is through group instruction in which the class together writes a mystery while a few students enter the material at a computer. A brief guide sheet including the following template might prove helpful.

```
to mystery
if not front? [flip]
clearpage
print [ - - - - - - - - - - - - - - - - -
   - - - - - - - - - -]
end
to clearpage
rg
ct
ht
cc
end
```

Here's an example of a **mystery** procedure.

```
to mystery
if not front? [flip]
clearpage
print [It was a dark and stormy
   night. Boomer, my dog, started
   barking. He woke up the house. We
   heard a door slam and the sound
   of footsteps on the back porch. A
   heavy thud was followed by a
   blood-curdling scream. What had
   happened?]
end
```

The next step is for students to begin creating procedures named after keywords in the story to help solve the mystery. Here are some examples that might be used with the previous "mystery."

```
to Boomer
if not front? [flip]
clearpage
print [Boomer, as I told you, is my
   dog. He's a wonderfully big and
   shaggy mutt. He's as old as I am.
   He's my best friend.]
end

to dog
if not front? [flip]
clearpage
```

```
print [Gosh! A dog is a dog is a
   dog.]
end

to door
if not front? [flip]
clearpage
print [Yes, the door did slam. And
   it was loud.]
end

to footsteps
if not front? [flip]
clearpage
print [Undoubtedly those footsteps
   came from the back porch.]
end

to thud
if not front? [flip]
clearpage
print [I thought the thud was caused
   by a falling body.]
end

to scream
if not front? [flip]
clearpage
print [That scream sounded terribly
   familiar. It seemed to be that of
   a tone-deaf SOPRANO.]
end

to soprano
if not front? [flip]
clearpage
print [YOU'VE SOLVED THE MYSTERY.
   That soprano was my big sister
   BETH who had stumbled over a box
   because she was too lazy to
   switch on the back porch light.
   She wasn't hurt but Boomer did
   make her uncomfortable with all
   his loving licks.]
end
```

To review, the students simply begin by constructing a central procedure mystery, which outlines a plot. From this plot they select words like **boomer, dog, door, footsteps, thud,** and **scream** as names of procedures that give clues to solving the mystery. In the preceding example, it was the command **scream** that contained the word **soprano** that gave the solution.

There are many possibilities in developing a microworld like this. Students who take a great interest in writing will delight in it. Those who have unbridled imaginations will relish endless variations. The challenge, of course, is to solve the mystery as quickly as possible. And the sharing of results is the best of all. To do this, I generally have the students name the page with their own first names. Then as they finish the assignments on their own disks, I transfer the work over to a class disk. Thus, a number of programs are available for perusal.

It might be wise to have the students do much of this work off the computer. If all of the important writing is done before going to the computer, a lot of class computer time is saved. Computers are wonderful tools that may guide even the poorly organized student toward developing good work habits.

References

Yoder, Sharon. (1990). *Introduction to programming in Logo using LogoWriter* (rev. ed.). Eugene, OR: International Society for Technology in Education.

Robert Macdonald
Hawthorne Meadows
10225 Nancy's Blvd, Unit 63
Grosse Ile, MI 48138

All About Logo

A comprehensive new shareware text for K-12 teachers, pre-service college instructors, and secondary students

- 14 chapters (115 pps) explain Logo from "Procedures" to "Advanced List Processing."
- Chapters organized around teacher-lead discussions of sample problems, followed by suggestions for classroom tested hands-on explorations
- Appendices deal with error messages, programming style, vocabulary, and approaches to teaching and learning Logo, starting in elementary school
- Appropriate for a 15-hour college course, semester high school course, or to provide background for K-8 or home study teachers.

Inspection copies available without obligation in one of four ways:

- Request a spiral-bound hard copy version from the author (enclose $5 to reimburse the cost of duplication and mailing)
- Send a blank 3-1/2" DD or HD disk in a stamped, addressed return mailer for a Macintosh Word 4.0 formatted version (no cash, please.)
- Download via ftp. Email dkresse@ctp.org for details.
- Obtain a copy from a colleague

Send disk and hard copy orders to David P. Kressen, 3081 Oneida St., Pasadena CA 91107. Specify LogoWriter version: 1.2 for Macintosh or 2.2 for Apple IIc and IIe.

Maria Montessori, Meet Seymour Papert

by Elizabeth Early Feller

If ever I would have enjoyed being a little mouse in the corner of a room watching all that went on around me, it would have been at the now impossible meeting of Maria Montessori and Seymour Papert. Maria Montessori died before technology became a part of the classroom, but I know she would have made *LogoWriter* the chosen program for her classroom because of the child's interaction with it. In my mind I can visualize her seated at a computer with Seymour Papert guiding her. After initial instructions, I envision her taking off on her own, with Papert smiling as she took great joy in watching the turtle accept her commands, simple at first, but ultimately involving variables and recursion. With those concepts learned, I'm sure she would have wanted to construct her famous Pink Tower on the computer!

Since she is not here to do this, I felt compelled to have my students, most of whom grew up in the Montessori environment, do it for her. I have found no better means of demonstrating the combination of recursion and variables than to ask them to construct the Pink Tower of their preschool days. For those readers not familiar with Montessori teaching, the Pink Tower is a collection of pink cubes, the largest being about four inches on a side and the smallest being about one-half inch on a side. The students construct a tower using these cubes, making sure that the cube placed on the tower is always smaller than the preceding one.

Doing the project using *LogoWriter* is fascinating, and I have heard fourth graders squeal with as much joy at the completion of the Logo tower as four-year-olds squeal when they finish the actual tower. Both Montessori and Papert wisely put the rewards for learning in the pride of the completed task.

By the time students are in the fourth grade, they already know the easiest procedure to draw a square. With the introduction of variables, they soon learn that the size of that square can be changed very easily. The addition of recursion shows them that they can indeed get the turtle to make more than one square. They are also shown that the recursive call can ask for the next square to be smaller than or larger than the preceding one by adding something to or subtracting something from the variable. They seem fascinated when shown that the turtle can, in addition to everything else, follow directions to stop. With these concepts introduced, the students are reminded of the Pink Tower of their youth and presented with the challenge of re-creating it on the monitor, using both the concepts of recursion and variables.

The project has many pitfalls for those who don't totally understand the power of variables and recursion. Variables are used to simplify the task of making the tower as large as possible without going off the screen and to allow for the successive shrinking of the squares with recursion. Problems most frequently encountered are:

- setting of the position within the recursive procedure (which would produce a bird's-eye view of a tower pushed into a corner)
- problems getting the squares to fill (usually due to the student trying to fill the completed tower rather than filling each square as it was drawn)
- an out-of-control turtle (caused by restrictions on the recursion placed after the recursive call).

It is indeed interesting to watch as a student decides that for this project it is far easier if the command to draw a square is

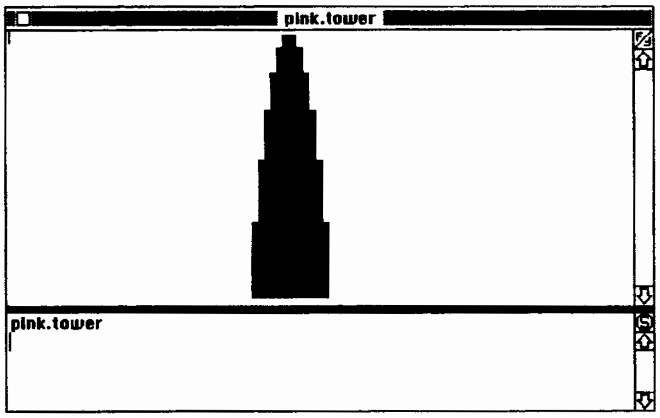
```
repeat 5 [forward :side right 90]
```

The following procedures illustrate one version of the completed Pink Tower.

```
to ready
pu
setpos [-56 -104]
pd
setc 6
ht
end
```

```
to tower :side
repeat 5 [forward :side right 90]
forward 5
left 90
pu
back 5
pd
fill
pu
forward 5
pd
if :side <10 [stop]
tower :side - 10
end
```

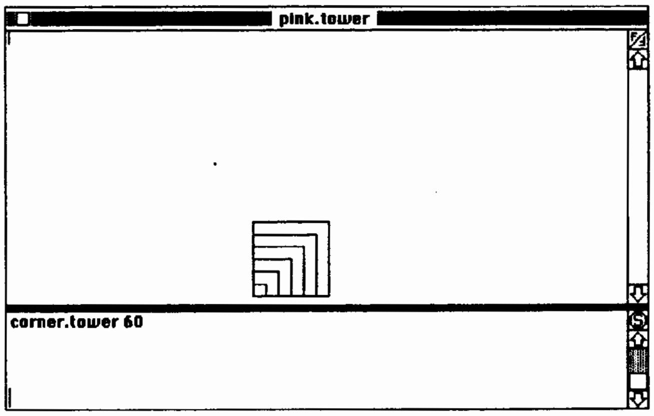
```
to pink.tower
ready
tower 60
end
```


pink.tower

```
pink.tower
```

These next two examples demonstrate two of the mistakes students frequently make.

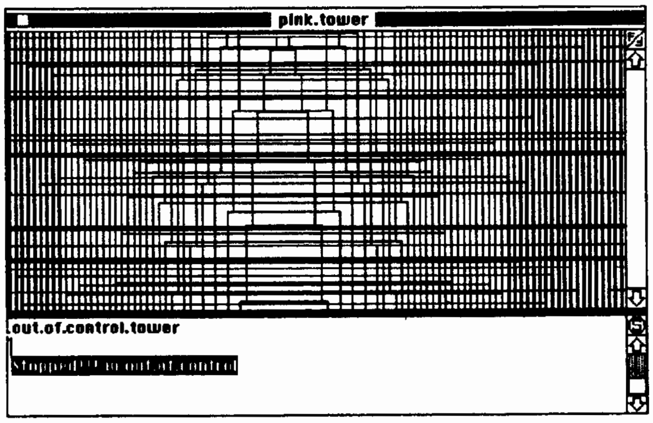
```
to corner.tower :side
pu
setpos [-50 -104]
pd
repeat 5 [forward :side right 90]
forward 5
left 90
if :side < 10 [stop]
corner.tower :side - 10
ht
end
```


pink.tower

```
corner.tower 60
```

```
to out.of.control :side
repeat 5 [forward :side right 90]
forward 5
left 90
out.of.control :side - 10
if :side <10 [stop]
end
```

```
to out.of.control.tower
ready
out.of.control 60
end
```


pink.tower

```
out.of.control.tower
Stopped!!! in out.of.control
```

The preceding procedures were written on a Macintosh version of *LogoWriter*. It should be noted that the position and size of possible squares vary if other versions of *LogoWriter* are used.

If Maria Montessori and Seymour Papert could be little mice in the corner of my computer lab, I'm convinced that they would be extremely impressed with what they saw as the children worked with enthusiasm and awe toward meeting the challenge presented to them.

Elizabeth Early Feller teaches of computer science at Visitation Academy in St. Louis, Missouri. She can be reached at
7140 Kingsbury Blvd.
St. Louis, MO 63130

Electronic Coordinates

by Glen L. Bull and Gina L. Bull

Turtle graphics was an early and fortuitous addition to Logo that contributed greatly to its popularity. Previous computing languages such as BASIC relied upon Cartesian coordinates for graphics. Turtle graphics provided a more accessible introduction to many mathematical concepts.

Ultimately, students need to know both mathematical systems—the intrinsic geometry of the turtle and the grid-based system of Cartesian coordinates. An understanding of Cartesian coordinates is important for graphing and other mathematical concepts. Logo and the turtle can provide a bridge for understanding Cartesian systems.

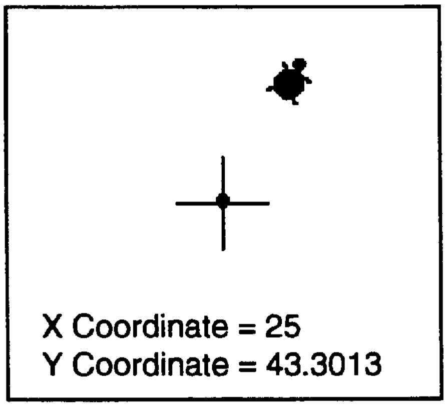
Cartesian coordinates define the turtle's location on the X-axis and Y-axis of an imaginary grid. To explore this system, try moving the turtle around the screen. Home is the center point (0,0) of the grid. Beginning at home, turn the turtle right 30 degrees and move forward 50 steps.

```
HOME
RIGHT 30
FORWARD 50
```

Then type "PRINT XCOR" and "PRINT YCOR" to see the turtle's coordinates on the X and Y axes.

```
PRINT XCOR
PRINT YCOR
```

You will find that the turtle is 25 steps to the right of the center point of the Cartesian grid and about 43 steps above the center point. Our version of *LogoWriter* gave the Y coordinate to four decimal places of accuracy. That's more accurate than needed for our purposes, but the ROUND command can be used to round the result off to the nearest whole number.

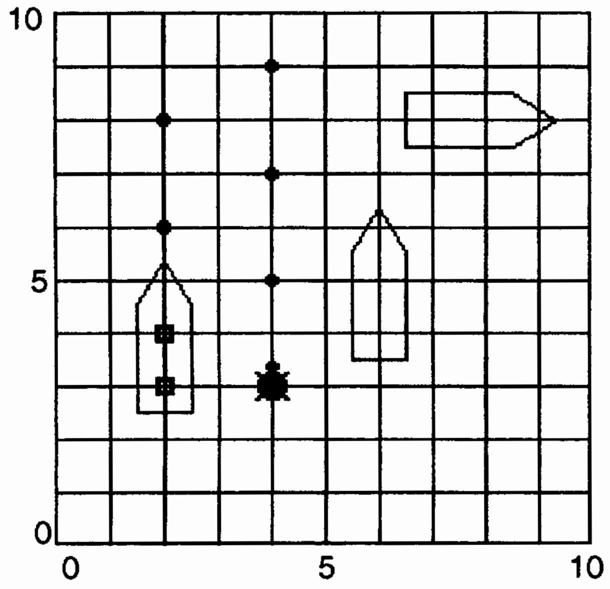


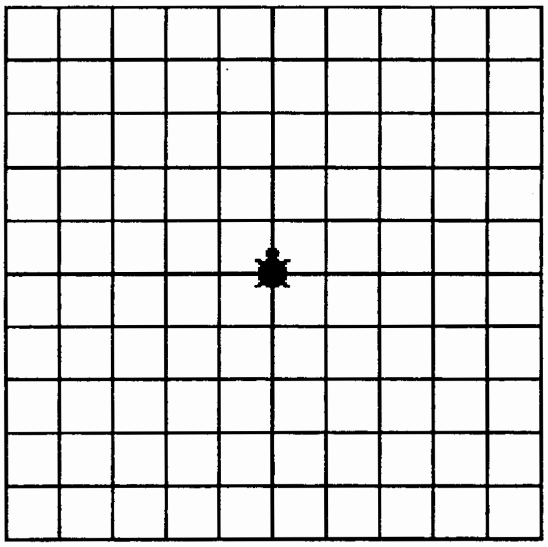
```
X Coordinate = 25
Y Coordinate = 43.3013
```

Electronic Battleship

The child's game of Battleship can provide an introduction to Cartesian coordinates. In this game, one player places several ships on a paper grid, while an opponent calls out coordinates at which the ships may be hidden. The opponent is given feedback as to whether the coordinates requested result in a hit or a miss. Battleship develops expertise with use of Cartesian coordinates in the context of a game.

An electronic version of Battleship is available for $50. However, readers with Logo can create their own electronic version of Battleship by using the turtle.





Building the Game Board

The first step in creating an electronic version of Battleship using Logo is to develop the basic grid that will serve as the playing field. The following GRID procedure uses two subprocedures, LINE and OVER. The SETX and SETY commands are used to set the turtle to the appropriate starting positions on the X-axis and Y-axis.

```
To Grid
PU
SETX -100 SETY -100
Repeat 11 [PD Line 200 PU Over 20]
SETX -100 SETY 100
RT 90
Repeat 11 [PD Line 200 PU Over 20]
PU
HOME
End

To Line :Distance
FORWARD :Distance
BACK :Distance
End

To Over :Distance
RIGHT 90
FORWARD :Distance
LEFT 90
End
```

We created a 10 x 10 grid in which each square is 20 turtle steps across. Depending on the size of your screen, you may need to adjust the dimensions of your screen to fit your computer (depending upon whether it is an Apple II, IBM, or Macintosh).

Moving the Turtle Across the Game Board

Once the basic grid that comprised the playing field for Battleship was created, we needed a way to move the turtle from square to square. The commands UP and DOWN seemed like logical ones to move the turtle up or down several squares. Because this is a nautical game, the commands PORT (left) and STARBOARD (right) seemed appropriate commands to move the turtle left and right.

```
To Up :Squares
SETY YCOR + (20 * :Squares)
End

To Down :Squares
SETY YCOR - (20 * :Squares)
End

To Starboard :Squares
SETX XCOR + (20 * :Squares)
End

To Port :Squares
SETX XCOR - (20 * :Squares)
End
```

These commands move the turtle from its current X or Y coordinate to a new location. For example, the command UP 3 would move the turtle up three squares on the battleship grid, while the command DOWN 4 would move the turtle down four squares. (*Programming tip*: If you have made the size of your squares on your grid larger or smaller than ours, you will also need to change the number 20 in the procedures UP and DOWN to reflect the size of your squares.)

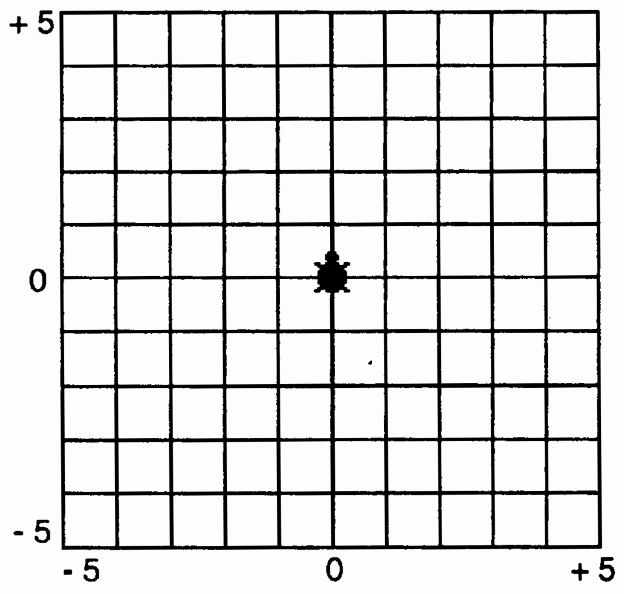
Determining the Turtle's Position on the Game Board

To make it easier to determine the position of the turtle on the Battleship grid, we created a POSITION command. To display the turtle's position, round the turtle's X and Y coordinates to the nearest whole number and divide by the size of the grid square used.

```
To Position
OUTPUT LIST ROUND XCOR/20 ROUND
   YCOR/20
End
```

This initial position command displayed the position of a turtle on a grid in which the center point is [0 0], and the bottom left point is [-5 -5]. Typing PRINT POSITION will display the position of the turtle on this coordinate system.

```
PRINT POSITION
0 0
```



Turtle Position = 0 0

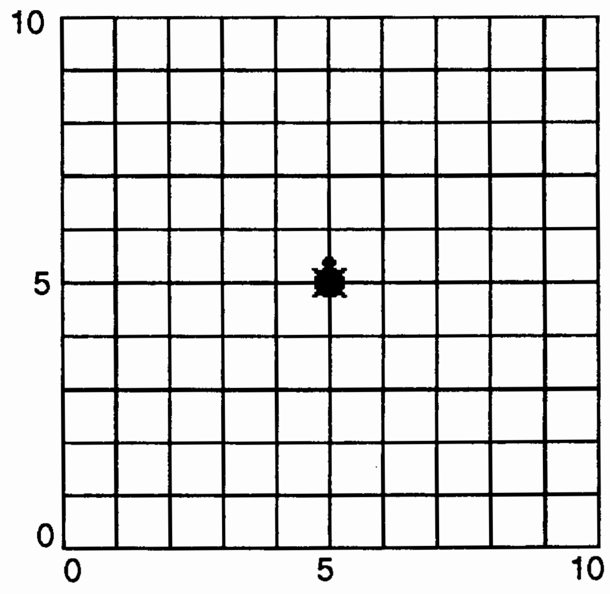
We wanted to work with a grid that did not have negative numbers, so we revised the POSITION command to eliminate negative coordinates (by adding 5 to the position displayed).

```
To Position
OUTPUT LIST ROUND XCOR/20 + 5 ROUND
   YCOR/20 + 5
End
```

To save time typing (we thought we might be using the POSITION command often), we created a procedure named P that would print the position of the turtle for us.

```
To P
PRINT SENTENCE [Position -] Position
End
```

The new POSITION command displayed the position of the turtle shown in the following illustration as 5 squares over and 5 squares up.



Position = 5 5

Playing the Game

After the game board was developed, we were ready to play the game. In the electronic Logo version of Battleship, Player 1 creates a list of coordinates at which ships are located. (Player 2 does not watch as the list of ship locations is entered). In the following example, Player 1 has indicated that a ship lies across the coordinates of [2 3], [2 4], and [2 5]. It is important to place each coordinate inside brackets, as shown below. The X coordinate is given first, followed by the Y coordinate. (If players are not certain of the coordinates of a location, they can move the turtle to that position and use the P command to print the coordinates.)

```
To ShipList
OUTPUT [ [2 3] [2 4] [2 5] ]
End
```

After Player 1 has entered the coordinates of ships in the SHIPLIST, Player 2 returns and moves the turtle to the possible location of a ship. The command FIRE is used to test whether a ship is found at that location. The FIRE command uses the function HIT? to determine whether a ship is at that location. The function HIT? uses the Logo command MEMBER? to determine whether the turtle's grid position is a member of the ship list.

If the turtle's position on the Battleship grid is on the SHIPLIST, a hit is marked by beeping and stamping a hollow square at that location. If a ship is not present, a miss is marked by buzzing and stamping a round dot at that location.

```
To Fire
If Hit? [MarkHit Beep]
If Not Hit? [MarkMiss Buzz]
End
```

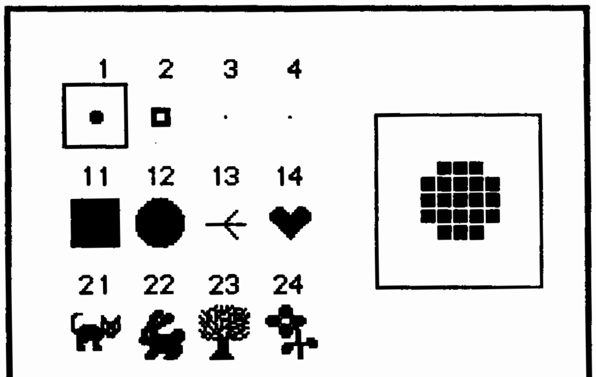
```
To Hit?
OUTPUT MEMBER? Position ShipList
End
```

The BEEP and BUZZ commands used by the FIRE procedure are straightforward.

```
To Buzz
TONE 100 10
End
```

```
To Beep
TONE 400 10
End
```

The MARKHIT procedure marks a hit by stamping a hollow square at the turtle's location, while the MARKMISS procedure marks a miss by stamping a round dot. We edited Shapes 1 and 2 in the *LogoWriter* shape table to create a dot and a hollow square. However, you could use the shapes you think are most appropriate to mark a hit and a miss, or use existing shapes in the shape table.



LogoWriter Shape Table

The MARKHIT and MARKMISS commands used by the FIRE procedure set the shape of the turtle to the desired shape, stamp the shape, and then set the shape back to the original turtle shape (Shape 0).

```
To MarkMiss
SETSH 1
PD STAMP
SETSH 0
PU
End
```

```
To MarkHit
SETSH 2
PD STAMP
SETSH 0
PU
End
```

Summary

The paper-and-pencil version of Battleship provides a useful way for children to become familiar with a coordinate graphing system in an enjoyable context. Our electronic version developed with Logo provides the opportunity for exploration of a classic game in a new context.

There are a number of enhancements that might be worthwhile. We quickly found that a MOVETO command that would move the turtle directly to any square was needed. Later we hope to add a Record function that will allow Player 1 to add ships to the ship list simply by moving the turtle to a given location and typing R. Both players have a grid in the classic pencil-and-paper version of the game and compete against one another to see who can sink the opponent's ships first. Provision of a second grid for Player 2 would be a useful enhancement of the Logo version, allowing players to compete directly with one another.

Tinkering with a game of this kind can often be as much fun for students as playing the game itself. We hope that this prototype game can serve as a springboard for further exploration by your students.

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

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

Make This Step Your Next Step.

You've got your new Mac... Now what?

You want to know what those cute icons mean. You need to find out how to use that mouse-thing-a-ma-jig.

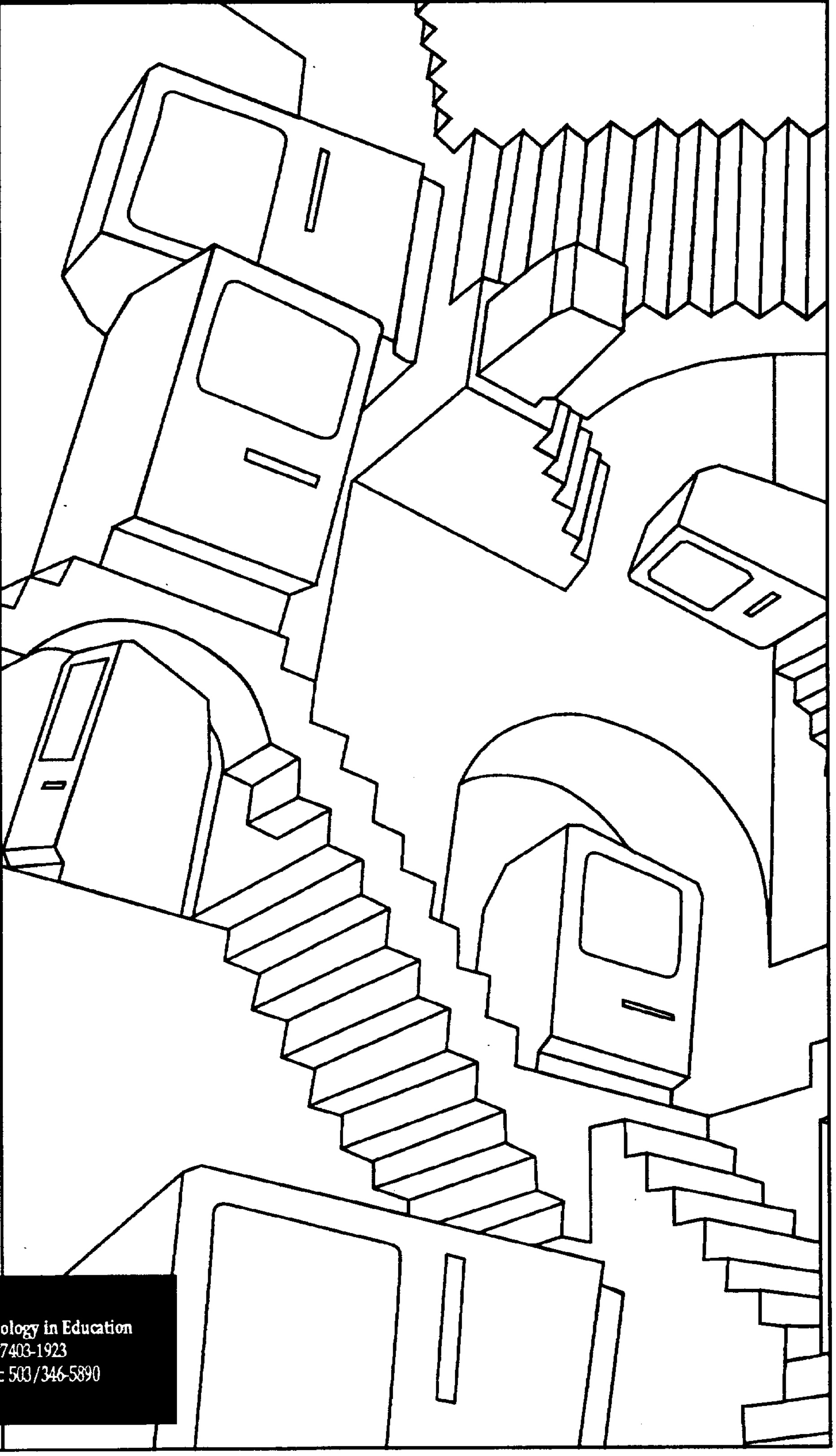
You'd like to know how to "empty the trash," and it'd be great to understand exactly what the "chooser" chooses.

By following the footsteps in ISTE's new book, *Macintosh Step by Step*, you'll pick up on all these basic Macintosh operating procedures, plus more—from finding the "power" button to demystifying the little Apple menu in the upper left corner.

And it also teaches you the theory behind Mac file storage and the importance of backing up your data on a regular basis.

In all, you'll find that this simple, illustrated guide will lead you on your way toward becoming a Mac user—not just a Mac owner.

Just call today and make your next computing step a step in the ISTE direction.



ISTE
International Society for Technology in Education
1787 Agate Street, Eugene, OR 97403-1923
Order Desk: 800/336-5191 Fax 503/346-5890

For pricing information or to order, see the order form in this publication.

Logo Robotics

by A. J. (Sandy) Dawson

The children we teach are growing up in a world filled with technology. Are they being prepared to use technology to their advantage to solve problems, to collaborate, to function in teams, to plan and adapt their plans as needed, and to understand how machines work? What happens when you combine the scientific notion of mechanical advantage with children's creativity? This column reports on a classroom-based robotics project that involved 50 children (9-11 years old), four teachers (one of them being Susan van Gelder), and a university researcher (Marion Barfurth) over a five-month period. In addition to learning about drawbridges and window washers, you get an intimate view of children's learning processes engaged in design and construction activities.

Robotics in an Integrated Curriculum: Children's Investigations of Mechanical Advantage With LEGO®TC logo

by Marion Barfurth and Susan van Gelder

Introduction

From a teaching perspective, the project described here provides a model of how to integrate LEGO®TC logo activities into curriculum. From the learning perspective, it provides a closer look at what the children did and how they went about doing it. You will encounter how within very different group dynamics, a variety of learning paths and an assortment of original inventions all coexisted at the same time in the same classroom.

An exploration of robotics served as a focal point for an intercurricular unit in Grades 4 and 5. Through team teaching (and learning), the robotics unit covered goals in science, math, language arts, and computers. The children explored notions of mechanics and robotics using LEGO®TC logo materials. The goals of the project were that the children:

- Understand the use of gears, pulleys, levers, and mechanical advantage.
- Work collaboratively in groups.
- Make a plan and follow it through, adapting as needed.
- Have more of an understanding of the technological world in which they live.
- Be able to reflect on their progress and difficulties.
- Understand the process involved in the development of a technological solution to problems.

The Robotics Project

During the project, math and science were taught during the same 45-minute period. Math was often used because it was a useful tool for trying to understand the physics concepts that were a foundation for the robotics work. As a result, math took on a variety of roles that were used by both the teachers and the children. These included using math as something with which to illustrate, communicate, reason, verify, and even argue.

Before the unit started, the children had studied simple machines in science. They had examined a variety of old objects (egg beaters, butter churns, etc.) in order to see how they worked and to find the simple machine that enabled them to help us do work. Thus, they were familiar with gears, pulleys, and levers. For the robotics unit they were introduced to the Lego® Technics materials. They had the use of two Apple IIe computers, one equipped with the LEGO®TC logo interface and one with an interface by Cameleon.

The project followed four steps:

1. The exploration of the materials, which was done by using the cards that came with the LEGO®TC logo kit as well as through free exploration.
2. The building of something that would be able to lift a weight.
3. The construction of something that would demonstrate mechanical advantage.

4. The design and construction of an automated machine that would use mechanical advantage to accomplish some kind of work.

Children worked in teams of four or five, mixed from both grades and whenever possible with an equal number of boys and girls. The teams stayed together throughout the project.

During the exploratory phase of the project, the children built a variety of objects. They often used gears and pulleys inappropriately (either using too many, not using them to advantage, or using them as decoration). They frequently operated under the notion that more is better. In phase two, the task was to build something that could lift a weight. At this point the children also began to focus on the problem of structure stability. Each team demonstrated its invention with a series of increasing weights. Their classmates were requested to offer constructive criticism. Some teacher intervention was necessary to keep the comments in a positive vein and to offer questions that would help steer the students toward finding solutions. It should be noted that the teachers themselves did not necessarily have solutions! It was when examining the lifting machines that the children discovered that some machines worked more efficiently than others. In comparing the various machines they noted differences in structure and lifting ability.

This led to the introduction of the notion of mechanical advantage. It was not only the number of gears or pulleys that improved an invention's efficiency but also the size and manner in which they functioned together. A demonstration of mechanical advantage was given using block and tackle, gear boxes, inclined planes, and a fulcrum and plank. This was done by using materials found in hardware stores or classrooms. In addition, a gear box constructed with Lego™ materials was demonstrated. The children then attempted to build something that would demonstrate an understanding of this concept. They were encouraged to refer to the machines around them.

The final phase of the project involved making written plans, building, evaluating progress, and revising. A few groups wanted to abandon their original ideas but were told that if they contracted to build a particular device (e.g., an elevator) their clients would not accept a windmill in its place! Thus, they had to focus on their problems and attempt to solve them. A log was kept, with a record of problems, triumphs, and design changes. At a couple of points in this phase the groups shared their inventions through formal presentations and got feedback from their peers. Informal exchanges occurred as children moved freely around the classroom, frequently stopping to see the progress of their friends' inventions. In this way each group benefited not only from the pool of knowledge in their own group but also from the class. In demonstrating their invention they often came up with problems that they presented to their peers. These sessions proved to be very beneficial, and the inventors went back to work with a new awareness of problems and possible solutions. This process led to a refinement of the inventions. When the machine functioned mechanically it was attached to the computer and programmed. Here again, a recursive process of planning, testing, and revising led to refinement of the programs. Children learned that the materials they used did not function in an ideal world (e.g., gears did not always meet perfectly, gravity affected their machines, and they had to allow for this in their programming). The inventions were displayed and explained at an open house for parents.

Within-Group Dynamics

To get a better idea what the children were doing and talking about as they worked, two groups of four children, randomly chosen, were closely followed by the researcher. Because the project combined children from a Grade 4 and Grade 5 classroom, each working group was composed of two children from each grade level. As it turned out each group had a Grade 4 and Grade 5 girl and a Grade 4 and Grade 5 boy. The research on small-group learning indicates that internal group dynamics play an important role in the learning process (Granott, 1991; Johnson & Johnson, 1985). The dynamics within each group of four children differed significantly.

Group A had a resident expert who played a very strong role in the group. This child, who had access to LEGO®TC logo at home, had a tendency to control the design and construction phases, apparently because his tolerance for failure was very low. Although the other children did insist on sharing, it seemed on several occasions that whenever the time was running out, it was handy to have the expert there to complete whatever task was being addressed at the time. The problems that Group A encountered with respect to LEGO®TC logo reflected their advanced abilities in design and construction. They tended to be resolved quickly, although often individually, perhaps shortcutting the potential exchanges of ideas and understandings within the group.

The dynamics within Group B were quite different from Group A. This group had no resident expert, but it did have a leader: the Grade 5 girl. The two Grade 5 children often held opposing views, especially with respect to the design, construction, and physical concepts related to their inventions. Early in the project, this opposition became an interesting foreground for the negotiation of understanding of concepts related to mechanical advantage during their investigation and interrogation phases. The two Grade 4 children, although participating quite actively throughout the project, had a tendency to make themselves invisible when strong confrontational exchanges about physical concepts were taking place.

The Children's Inventions

What LEGO/Logo products were invented by each of the two groups? Following is a description of the inventions designed and constructed in the last two of the project's four steps described previously. They cover approximately a 15-week period that includes holidays, school breaks, and pedagogical days.

Group A began their first mechanical-advantage task by successfully constructing a gear box. The gear box was then extended into a fan by adding a handle and a blade. The children called it the *Handy Fan* (making reference to its size). Because they were unable to obtain satisfactory performance from fan blades made with LEGO pieces, bristol board was introduced and used for the blades. The second project, which was planned on paper and then constructed, was a *Window Washer*. It consisted of a small scaffold holding two men that was able to go up and down along the side of their LEGO office tower. The mechanics of the *Window Washer* consisted of an adapted version of a gear box. In the case of this group's invention, little difficulty was encountered with using mechanical advantage, but they did encounter a design flaw in adapting their machine to this new invention. Because they chose to live with this flaw rather than improve it, this group spent more time on the automation aspects and on aesthetic detail, such as placing a lawyer—using Lego figurines— eating pizza with his clients on the second floor.

Group B had set as its first goal the building of a conveyor belt. They encountered many difficulties at a foundational level of the design. They wanted a *belt* (LEGO linked chain) to wrap around a string of gears that would use mechanical advantage. The task initiated many discussions on mechanical advantage and other physical notions. It concluded with them building, under the guidance of the computer teacher, a very simple gear box. As a final invention, this group designed and constructed a drawbridge. Several attempts were made before arriving at one that worked even though it had a design defect: the bridge would go up, but the children had trouble with the counter weight which would allow the bridge to go back down. The children worked on both the automation and aesthetic aspects of their drawbridge.

An Intimate View of the Learning Process

Among the observations in this study were those related to children's uses of mathematics. As the children grappled with their understanding of science concepts related to gears and mechanical advantage, they reached out for numbers, numerical relationships, and mathematical arguments for expressing and defending their understanding. Not only was math becoming something genuinely useful, it was used as a means of rendering their observations more concrete!

The following conversation is taken from the first session with Group B. The group was just beginning to assemble material (LEGO bricks, axles, gears, etc.) for building purposes. The excerpt provides a glimpse into the children's scientific and mathematical reasoning while grappling with the concept of mechanical advantage. It provides a very intimate view of a disagreement between two of the children (Pushpa and Jacques), with a third (Kayode) trying to help. Their discussion focuses on the potential that their choice of gears have for producing a mechanical advantage.

Pushpa: Count how many spikes are on here.

Jacques: [Counting the teeth on a large gear] Eleven times four.

Pushpa: How do you know times four?

Jacques: Cause I counted a quarter of it.

Pushpa: Ah, O.K. This one has 24. So that's not even double that. So this won't even turn two times with that one.

Jacques: Yes it will. It will turn two times.

Pushpa: 24, 44.

Jacques: So that's about two times.

Kayode: Two times 24? That's 48.

Pushpa: No. Forget it, you don't understand that that one is 44; this one is 24. This one won't even turn double every time.

Kayode: It will be less than double.

Pushpa: Yes, exactly. It won't even turn double.

Jacques: [Who has been counting the teeth on Pushpa's gear] This is 20.

Pushpa: That's 20? [Looking a little surprised.]

Jacques: Yup!

Although this is one small excerpt of conversation, it illustrates the children's use of mathematics in a physics world of gears and mechanical advantage. At its limit, one might say that in a way similar to mature scientists, the children were interweaving mathematical and scientific thinking and reasoning.

The Integrated Curriculum

It is obvious that a number of scientific concepts were introduced in this unit. How were other curriculum goals met? In language arts, the Grade 4 children researched inventions and inventors. Some of the construction groups had time to develop an advertisement to sell their product. The notion of mechanical advantage led us to explore the mathematical concept of ratio in reference to gear ratios. Because half the classes were conducted in French, the children expanded their vocabulary and their listening and speaking skills in a second language. The goals of the computer program—to encourage procedural thinking and to be aware of technology around us—were met in both on-computer and off-computer activities. An important component of the process was learning to function in groups. Group dynamics varied, and the problems presented were sometimes caused by inability of groups to reach consensus or the difficulty some individuals had in accepting other children's ideas. Teachers worked with groups to help them function better. Some of our class discussion focused on cooperation and group skills.

The teachers themselves used a process similar to the children to design the unit. An initial plan was prepared as the unit started. The teachers met weekly to discuss the process and the progress. They focused on problems and reflected on the best course to take for the following week, continually revising their plans based on the students' needs. It was an exciting process. Too often teachers work in isolation (although at St. George's, collaboration is not uncommon). The group was able to learn from each other, to utilize each individual's expertise to the best advantage, and to problem solve together. The ideas of a group proved to be much greater than the sum of individual contributions.

Concluding Remarks

The students came away from the project with a greater awareness of the process of planning and carrying a design through to completion. In addition, they experienced the scientific process of hypothesizing, experimentation, and revision. In presenting their machines to the parents, the children were able to demonstrate that they had acquired a good understanding of the scientific principles of simple machines. Many groups had difficulty functioning as a group, and they learned to work out some of their problems. It was through collaboration that they were able to bring their inventions to completion.

From the child's perspective, the *Robotics in an Integrated Curriculum Project* provided an environment in which science and creativity could coexist. What the children built for investigating, interrogating and integrating the notion of mechanical advantage goes beyond what is often found in science textbooks. From the teachers' perspective, the LEGO®TC environment integrated into the classroom provided a very rich situation in which to learn about the evolution of children's understanding of related scientific notions.

Bibliography

Barfurth, M. A., & van Gelder, S. (1993, March). *Drawbridges and window washers: Children's investigations of mechanical advantage with LEGO®TC logo.* In *Proceedings of the 10th International Conference on Technology and Education,* Boston, MA.

Granott, N. (April, 1991). *Play, puzzles, and a dilemma: Patterns of interaction in the co-construction of knowledge.* Paper presented at the annual meeting of the American Educational Research Association, Chicago.

Johnson, D. W., & Johnson, R. T. (1985). The internal dynamics of cooperative learning groups. In R. Slavin et al. (Eds.), *Learning to cooperate, cooperating to learn* (pp. 103-124). New York: Plenum Press.

van Gelder, S., & Barfurth, M. A. (1993, March). *Robotics in an integrated curriculum.* In *Proceedings of the 10th International Conference on Technology and Education,* Boston, MA.

Susan van Gelder is the computer teacher at St. George's School. She integrates computers across the curriculum and teaches children to use them in a variety of innovative ways. She can be contacted at

St. George's School
3685 The Boulevard
Montréal, Québec, CANADA H3W 2C1

Marion Barfurth is a professor in mathematics education at the Université du Québec à Hull. Her areas of research include innovative uses of technology that facilitate children's active learning of math and science concepts. Her teacher training courses for preservice elementary school teachers emphasize a constructivist approach to learning embedded in a problem-solving context. Marion can be reached at

Université du Québec à Hull
C.P. 1250 Succ. B, Hull
Québec, CANADA J8X 3X7
Barfurth@UQAH.UQUEBEC.CA

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

A. J. (Sandy) Dawson
Faculty of Education
Simon Fraser University
Vancouver, BC, CANADA V5A 1S6
Sandy_Dawson@sfu.ca

Deoxyribonucleic Acid

by Mark Horney

Almost every day now, the media is full of stories about parents found (or lost), crimes solved, cures affected, evolution revealed, humanity defined, businesses operated, organisms created, catastrophes predicted, and movies derived from Deoxyribose Nucleic Acid, or DNA[1] for short. The public career of DNA began in 1953 when James Watson and Francis Crick described, in a short paper in the journal *Nature*, a model for the structure of the DNA molecule and announced that the model suggested "a possible copying mechanism for the genetic[2] material." This proclamation was electrifying because in the 94 years since the publication of Darwin's *The Origin of Species*, no physical process had ever been identified showing just how evolution takes place or how organisms know how to grow. Once the structure of DNA was understood[3], and later the DNA code broken, a whole host of biological advances ensued that we see today in the form of genetically engineered organisms, the biological production of drugs, the early detection of genetic diseases, and in the identification of people from their DNA "fingerprints." Given all this, DNA is a very important subject these days with implications for science, medicine, government, commerce, the law, and theology. And of course, anything of such wide spread concern must also have some application in Logo.

1. DNA Refresher: DNA is a very long molecule found in every cell of every living creature on Earth. Its basic structure consists of two strands twisted together into a helix. Stretching between the strands, holding them together, and looking much like the steps on a ladder, are clusters of atoms called "bases." There are four bases: adenine, thymine, guanine, and cytosine. They are always found in pairs, an adenine base on one strand connecting with athymine base on the other, and a guanine always attaching to a cytosine. Thus, the base sequence on one strand is the reverse of the sequence on the other strand. The helix can be duplicated by detaching each base from its partner and then reconstituting each strand with new bases. The base sequence acts as a code giving directions for the production of proteins, and proteins are the molecules that make up and operate cells.
2. Genetic as in "genes." A gene is sequence of DNA bases that together code for some particular characteristic of an organism. Strands of DNA holding sets of genes form "chromosomes," of which every organism has a fixed number, 26 in the case of humans. The first genetic research was done by an Augustinian monk, Gregor Mendel (1822-1884), who in 1866 deduced the genetic structure of peas. Unfortunately, Mendel's work was lost until 1900. Mendel's life and importance can be read about in Bronowski's (1973) *The Ascent of Man*.
3. Understanding DNA took rather more than Watson and Crick's 1953 paper. Many researchers were investigating DNA during the early 1950s and engaged in what amounted to a race. Watson (1968) tells this story in his book *The Double Helix* and Crick (1989) has a somewhat different version in *What Mad Pursuit*. Another story, usually overlooked, is told in *Rosalind Franklin & DNA* (Sayre, 1975). Franklin was well on her way to discovering the double helix herself and was responsible for producing some of the key physical evidence used by all the various research teams. Only her untimely death in 1958 prevented her possible inclusion in the Nobel Prize given to Watson, Crick, and Maurice Wilkins in 1962.

Logo-Engineered Artificial Life Forms[4]

Meet Fred. Fred is a LEALF, a Logo-Engineered Artificial Life Form. While there are any number of strategies for making LEALFs, the one I have chosen for Fred is simple and straightforward:

Step 1: Decide an overall design for the LEALF and express this design in a set of parameters such as those shown in the following steps for FACEOIDS such as Fred. These parameters are analogous to the of DNA base pairs.

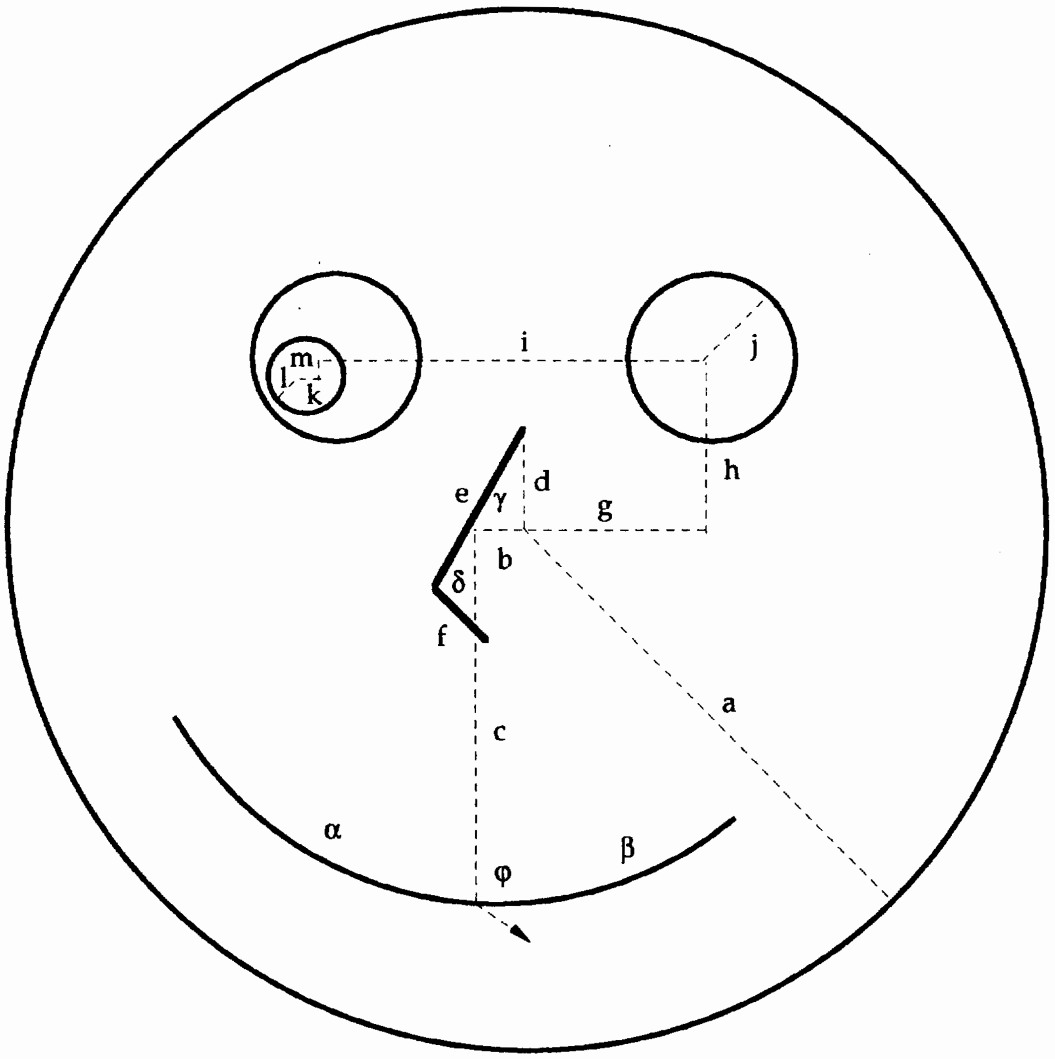
Step 2: Collect the parameters representing major LEALF parts into clusters (which are analogous to chromosomes) and pull the clusters together into a list (which is analogous to the whole DNA sequence[5]):

```
[[Head [a]] [Mouth [b c αβφ]] [Nose [d
   e F γδ]] [Eyes [g h i j k l M]]]
```

Step 3: Develop the primitive procedures needed for drawing LEALF parts such as the set of forward, back, right, left, and filled ARC procedures used by FACEOIDs.

Step 4: Build "chromosome" procedures, e.g., Head, Mouth, Nose, and Eyes, upon the primitive procedures.

Step 5: Actually draw LEALFs by using a superprocedure that accepts a DNA list as input, picks off "chromosomes" one at a time, and uses the RUN command to execute them.



4. For a general introduction, see *Artificial Life* by Steven Levy (1992). Another older, deeper, more philosophic approach is Herbert Simon's (1969) *The Sciences of the Artificial.*
5. Or in this case: Dynamically Numeric Affectations.

With all this Logo code in place, creating new LEALFs is as simple as writing out new parameter lists. It's also the place where this activity actually begins, since just making up a bunch of FACEOIDs isn't particularly productive. Here are a few ideas for what to do next.

Variation

While making FACEOID parameter lists is simple enough conceptually, it soon becomes tedious and quite apparent that Logo procedures for creating variations among creatures is an absolute necessity. Some possibilities:

- Vary the parameter list. Each parameter could be changed by some fixed amount (see Sybil). Or the DNA list could be expanded to include a "variation constant" for each parameter so that each can change independently, or even better, by some probability distribution.
- Change a creature's "handedness." Fred for instance is Left-Nosed and Smiling. By adding LEFT and RIGHT versions of the chromosome procedures, other FACEOIDs could be Right-Nosed and Frowning.
- Introduce new degrees of freedom. For instance, Fred is based almost entirely on circular arcs. These could be changed to ellipses or some other shape.[6]

Reproduction

I'm sorry, but there's no way around it, LEALFs just must reproduce. Cell Division is one option. This is the process used by single-cell creatures such as amoebas, which reproduce by splitting themselves into two fully functioning parts (with some VARIATION during the process no doubt). Another option is for LEALFs to exchange parameters or chromosome lists[7](see the Cromazians), or to merge parameters in some averaging process like morphing[8] (see Ma and Pa Morph). A more ambitious idea is to involve some sort of a transformation whereby the physiology of LEALFs change from one form to another.[9]

Evolution

If LEALFs can vary and reproduce, they can evolve. The trick is to establish a LEALF ecosystem and introduce into it some environmental factor whereby certain LEALF structures are reproductively preferred over others.

Ontogeny and Phylogeny

Evolving LEALFs as just contemplated here have a *phylogeny* but no *ontogeny*. *Phylogeny* is the evolutionary history of an species as it changes over many generations. *Ontogeny* refers to an individual's development over a single lifetime. One approach to LEALF ontogeny is to define rate genes rather than structural genes in the DNA. Rate genes control the speed of biologic processes rather than coding for some structure.[10] Take Peter Piper for example; rather than springing fully formed from the turtle's pen, suppose that for each of his parameters p there is an additional parameter p' that gives the value of p at some time T, which is specified at run time. Each depiction of Peter then would show him at some stage of his ontogenic development.[11]

6. See the article by Ken Large (1993) for a million ways to draw ellipses.
7. In other words—sex.
8. If you don't know what morphing is, you haven't been watching your MTV or Terminator movies. Morphing is the technique of smoothly transforming one image into another, e.g., Michael Jackson into a panther.
9. Edwin Abbot (1952) in his novel *Flatland* envisioned a biology for his polygonal creatures in which offspring have one more side than their parents. Status in *Flatland* naturally depended on having more angles than the neighbors.
10. A delightful look at rate genes is found in Stephen Jay Gould's (1980) article *A Biological Homage to Mickey Mouse.* Gould argues that Mickey's ontogeny is a prime example of "neoteny," the retention of juvenile features of ancestral forms. (Go rent some old Disney cartoons showing how Mickey was originally drawn as compared to now and you'll see what he means.) Neoteny works by slowing down the development of features and is a major factor in human evolution.
11. The topic of ontogeny and phylogeny is amply described in a book of that title by Gould (1977). If you remember anything about these concepts from high school biology, it is probably the doctrine of Haeckel (1834-1919) that "ontogeny recapitulates phylogeny." That is, as an organism develops from embryo to adult it passes through all of its previous evolutionary phases. This doctrine is no longer accepted.

Nongraphic LEALFs

FACEOIDs are graphical creatures; they exist as pictures. But there is no reason why LEALFs can't be creatures with textual or numeric manifestations, or for the DNA to code for the turtle's behavior.[12]

Reality

In reality, FACEOIDs, and all the mechanisms described here for creating and manipulating them, have very little basis in reality. Students need to be guided toward making LEALFs work more like real creatures.[13]

Implementation

The range of suggestions presented here surrounds the truly central idea: *DNA is an important topic with many connections into the lives of students. As such, it provides a rich environment for students to experiment with new programming techniques and to expand their understanding of concepts in several other domains.* The details of exactly what teachers and students do with the idea of DNA is almost immaterial. Some will need only the suggestion that DNA might be interesting; others will need more introduction and direction. More advanced students will be able to produce all of the necessary code themselves, others will need help, and some can be supplied a complete microworld of DNA resources and work productively without programming at all.

Resources

Abbott, E. A. (1952). *Flatland: A romance of many dimensions*. New York: Dover.

Axelrod, R. (1984). *The evolution of cooperation*. New York: Basic Books, Inc.

Bronowski, J. (1973). *The ascent of man*. Boston: Little, Brown, and Company.

Crick, F. (1989). *What mad pursuit*. London: Penguin Books.

Darwin, C. (1859). *The origin of species*. London: John Murry.

Gardner, M. (1983). *Wheels, life and other mathematical amusements*. New York: W. H. Freeman and Company.

Gould, S. J. (1977). *Ontogeny and phylogeny*. Cambridge, MA: Harvard University Press.

Gould, S. J. (1980). A biological homage to Mickey Mouse. In S. J. Gould (Ed.), *The panda's thumb* (pp. 95-107). New York: W. W. Norton.

Hofstadter, D. R. (1985). The genetic code: Arbitrary? In D. R. Hofstadter (Ed.), *Metamagical themas: Questing for the essence of mind and pattern*. New York: Basic Books.

Large, K. (1993). Ellipses. *Logo Exchange, 12*(1), 31-38.

Levy, S. (1992). *Artificial life*. New York: Pantheon Books.

Neumann, J. v. (1966). *Theory of self-replicating automata*. University of Ilinois Press.

Sayre, A. (1975). *Rosalind Franklin & DNA*. New York: W. W. Norton & Company.

Simon, H. A. (1969). *The sciences of the artificial*. Cambridge, MA: The M.I.T. Press.

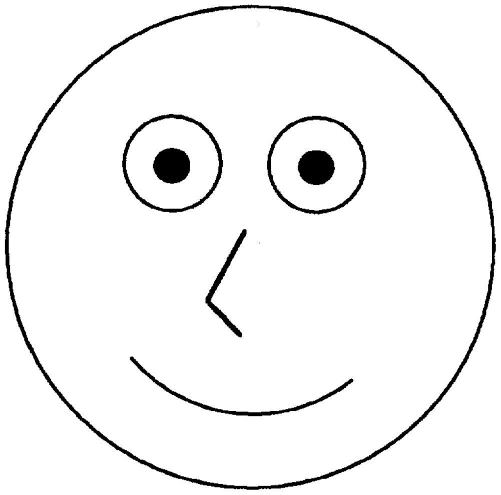
Watson, J. D. (1968). *The double helix*. New York: Atheneum.

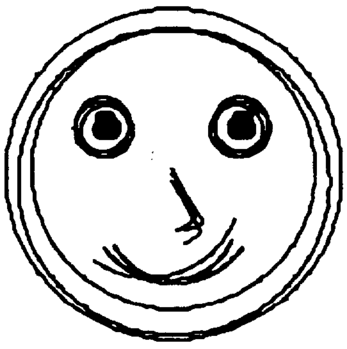
Watson, J. D., & Crick, F. H. C. (1953). A structure for deoxyribose nucleic acid. *Nature, 171*(4356), 737-738.

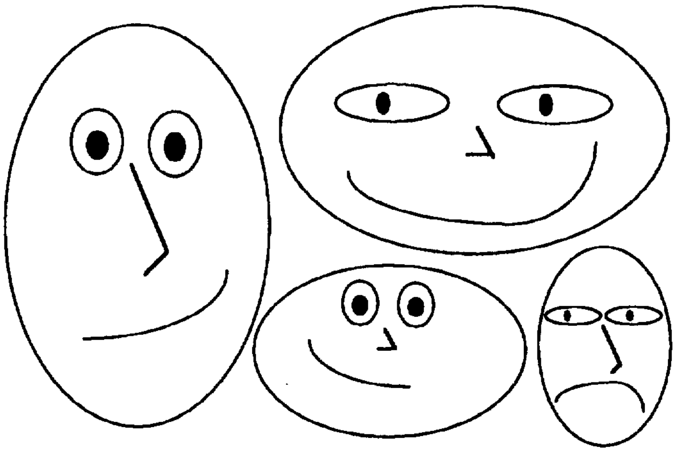
Mark Horney
College of Education
University of Oregon
Eugene, OR 97403
mhorney@oregon.uoregon.edu

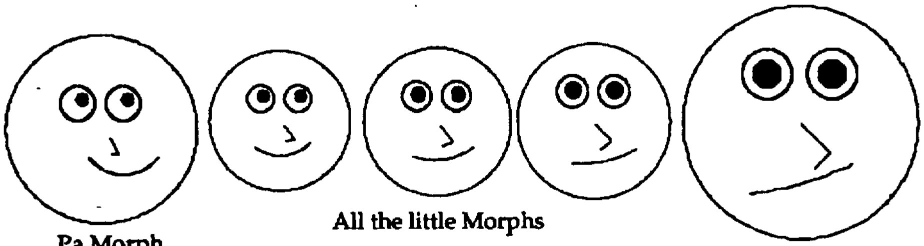
12. Two places to look for examples of such LEALFs are in the game of LIFE as described by Martin Gardner (1983) in *Wheels, Life and Other Mathematical Amusements*, and in *The Evolution of Cooperation* by Robert Axelrod (1984). LIFE is played on a checkerboard with each of the checkers being an individual creature. Once arranged in some pattern on the board, the creatures live and die according to fixed Rules of Life. (A more accurate term for "creature" is "cellular automata," which derives from the work of the mathematician John von Neumann, who sought to design machines capable of building exact duplicates of themselves.) The lives of Axelrod's creatures are governed by the Prisoner's Dilemma, a development of game theory, a branch of mathematics that seeks to explain, among other things, how to play poker.

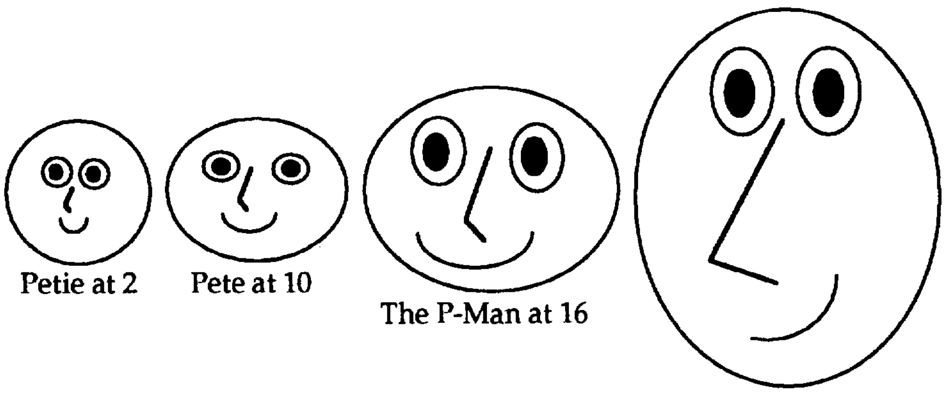
13. Here's a place to start. The DNA of FACEOIDs functions as an analog in that the parameters give actual physical dimensions. Real DNA is an abstract code. To see the difference read "The Genetic Code: Arbitrary?" by Douglas Hofstadter (1985).


Fred


Sybil


The Cromazians


Pa Morph All the little Morphs Ma Morph


Petie at 2 Pete at 10 The P-Man at 16 Peter at 30

A Research-Based Logo for Mathematics

by Douglas H. Clements and Julie Sarama Meredith

Can Logo help teach mathematics effectively? Readers of this column know that research is mixed—but also that it provides significant guidance in the teaching and learning of mathematics with Logo.

We are conducting a large-scale curriculum development project, funded by the National Science Foundation (NSF), that emphasizes meaningful mathematical problems and depth rather than exposure. We're developing the geometry and spatial sense strands and a modified Logo environment to accompany them. We based the environment on curricular considerations and a number of research implications.

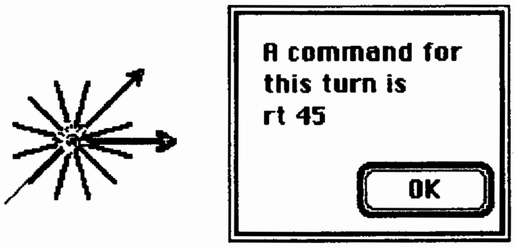
Research and the Design of *Geo-Logo*

Research provides directions for designing a Logo environment fine-tuned for the learning of geometry and other mathematical topics. We will briefly describe five principles we abstracted from several research reviews (Clements & Battista, 1992; Clements & Meredith, 1992; this column in previous years).

1. Encourage construction of the abstract from the visual and intuitive

Research supports Papert's hypothesis that Logo's turtle encourages students to construct geometric knowledge upon personal, intuitive, experiential knowledge (Clements & Battista, 1991). Often, though, teachers need to encourage this abstraction more explicitly in certain areas.

For example, there is a large literature on children's difficulty with turns and angles (Clements & Battista, 1991). Students may even confuse side length with rotation. *Geo-Logo* addresses this in two ways. First, *Geo-Logo* provides two commands, **rtf** and **ltf**, ("**rtf**" stands for "right face"), to help beginning students build on previous experience and appreciate the idea of turns before meeting measurement in degrees. Second, *Geo-Logo* provides measurement tools; for example, an on-screen protractor, placed at the turtle's position and heading, measures turns. One arrowhead shows the turtle's heading. The other follows the cursor, which students move with the mouse. When they click, this arrowhead "freezes" and a turn command is displayed.


A command for
this turn is
rt 45

Building robust ideas about rotation and angle measure, and connecting these two ideas, is often a long and difficult process, even when activities are specifically designed for the learning of these ideas (Clements & Battista, 1991). Furthermore, we should remember that even if students do not adopt our goals and use visual and empirical strategies, the environment should continue to support their activity. The visually-oriented measurement tools allow students to approach tasks in a wider variety of ways.

One of the main ways *Geo-Logo* supports the growth of the abstract from the visual lies in its overall structure, described in the following section.

2. Maintain close ties between representations

The nature of programming creates the need to make relationships between symbols (code) and drawings explicit. But students may lose the psychological connection between the two when involved in long projects. There are, then, two issues for a Logo programming environment.

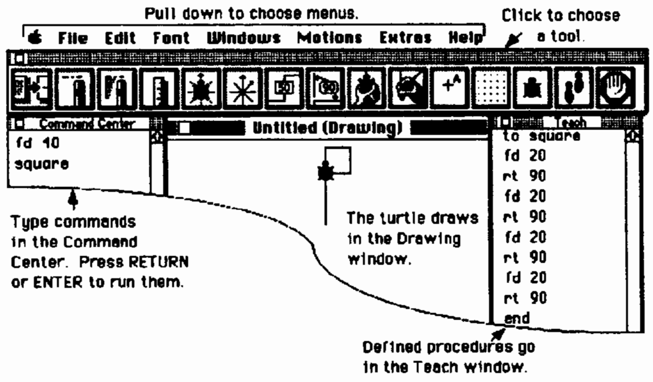
First is the issue of immediate-mode programming versus the use of procedures. Some have argued that students should use procedures created in the editor from the beginning to "force" them to see procedures as more than a way to "save their work." For certain goals and situations, this may constitute a viable approach. *Geo-Logo*, however, encourages immediate-mode programming—students type commands into the Command Center and see them immediately enacted. In our project, mathematical goals take priority over those related to computer science.

There are more important reasons for such a decision. Students (especially novices) often prefer this exploratory mode and find it easier to make sense of tasks. Furthermore, Hoyles and Noss (1987) found that having students record their commands while working in immediate mode encouraged them to take a more global perspective on the task and to look for structure within their program design. In contrast, use of procedures can lead to a separation between symbols and drawings (Hoyles & Noss, 1988).

Also, students working in immediate mode are more likely to abandon inappropriate solution strategies before they make incorrect generalizations, which keeps them on track. *Geo-Logo*'s Command Center displays all the commands, in sequence, even as they work in immediate mode.

These results lead to the following features of our Logo environment. Students enter commands in "immediate mode" in a Command Center or as procedures in a "teach" window. Any change to commands in either location, once accepted, are reflected automatically in the drawing. A tool (the first icon on the left) copies these into the teach window, applies a student-supplied name, and thus defines the procedure.

The dynamic link between the commands in the Command Center and the geometry of the figure is critical. Any change in the commands leads to a corresponding change in the figure, so that the commands in the Command Center precisely reflect the geometry in the figure. Thus, the Logo code in the Command Center stands half-way between traditional immediate-mode records and procedures created in an editor, helping link the symbols and drawing. The structure of the Command Center—long and narrow to the side of the graphics screen instead of the traditional short but wide placement below this screen—permits the immediate inspection of more commands, which facilitates connecting symbols and drawings as well as pattern searching. Furthermore, students can easily modify the code, encouraging experimentation and supporting later work with procedures.


Pull down to choose menus.
Click to choose a tool.
File Edit Font Windows Motions Extras Help
Command Center
Untitled (Drawing)
Teach
Type commands in the Command Center. Press RETURN or ENTER to run them.
The turtle draws in the Drawing window.
Defined procedures go in the Teach window.

```
fd 40
square
```

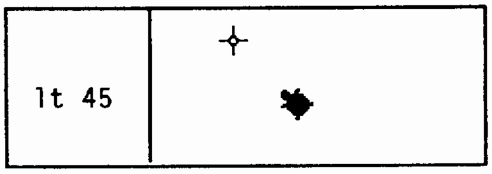
```
to square
fd 20
rt 90
fd 20
rt 90
fd 20
rt 90
fd 20
rt 90
end
```

For example, if you change the **fd 40** shown in the Teach window to **fd 50**, the drawing *automatically* shows that change when you press Return or Enter. Similarly, if you change a procedure in the teach window (e.g., change each **fd 20** to **fd 30** in the procedure **square**), the change is reflected in the drawing window as soon as you click out of that window.

The second basic issue is the direction of the symbol-drawing connection. One of Logo's main strengths has been its support of linkages between drawings and symbols. One of its limitations has been in the lack of two-way connection between these modes. That is, one creates or modifies symbolic code to produce visual drawings, but not the reverse. *Geo-Logo* provides a "draw commands" tool that allows the student to use the mouse to turn and move the turtle, with corresponding Logo commands created automatically.

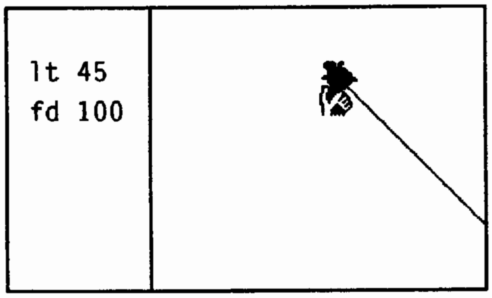
I

The cursor changes to a crosshair, the turtle continually turns to face it, and the corresponding turn command is dynamically updated in the Command Center.



```
lt 45
```

The turtle is then dragged forward or back and the corresponding **fd** or **bk** command is placed in the Command Center.



```
lt 45
fd 100
```

3. Facilitate examination and modification of code

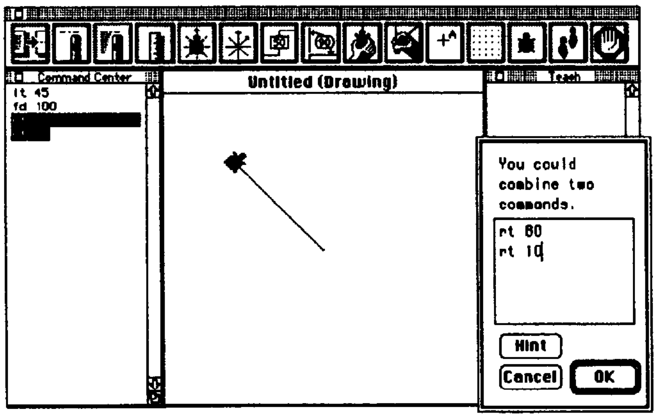
Research indicates that there is little reason for students to abandon visual approaches unless they are provided with support for such analysis. Several of *Geo-Logo*'s tools provide such support.

First, the Command Center interface just described helps students examine the sequence of commands reflected on the graphics screen, modify these commands, and immediately see the results of this modification.

In addition, a step function allows children to "walk through" any procedure or the commands in the Command Center. As they do so, they can edit any command in the sequence. The change is immediately reflected on the screen. Students can also step procedures with inputs. This is particularly important, given that otherwise they tend to lose sight of what the numbers specified as inputs mean and how they function within the procedure.

There are also two types of optional advice. With the "combine advice" option on, *Geo-Logo* gives students advice when they type two commands they could combine into one command, such as **rt 80 rt 10.** This helps students write a more "elegant" program. More importantly, it helps them reflect on geometry: If they don't see that **rt 80 rt 10** is equivalent to **rt 90,** they often do not see the two turns as creating a right angle. Similarly, when the "repeat advice" option is on, *Geo-Logo* gives students advice when they type a sequence of commands they could combine into one repeat statement. This helps them see the pattern of their commands and how that pattern is reflected in the geometric figure.

As an example of the combine advice feature, suppose a student typed **rt 80** and then **rt 10.** The advice dialogue box would appear.

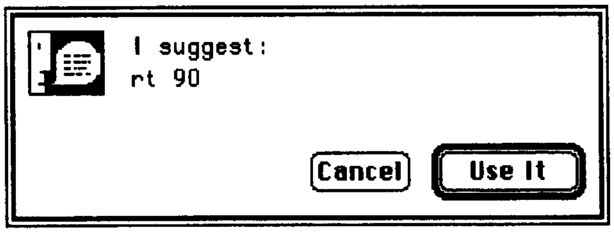


```
lt 45
fd 100
```

You could combine two commands.

```
rt 80
rt 10
```

They can change these two commands to "**rt 90**" (e.g., the student deletes the **rt 10** and the 80, and types in 90). Alternately, they might click on the Hint button and see the following:



I suggest:
rt 90

If they click on "Use It," **rt 90** is placed in the first dialogue box, replacing the two commands there. Then they can edit the command if they wish.

Finally, they can click on the OK button to have the single command replace the original two commands in the Command Center. They click the Cancel button to avoid making any changes.

4. Encourage procedural thinking

The benefits of using the Command Center and immediate-mode programming have already been described. *Geo-Logo*'s structure encourages use of procedures from the beginning. First, the teach tool walks students through the steps of defining procedures. Second, changes made to procedures within the teach window are immediately reflected on the graphics screen upon exiting that window.

5. Provide freedom within constraints

Our goal is to encourage problem solving and student-directed exploration within structured activities, both within the constraints of the *Geo-Logo* environment. We provide a set of activities, but they are not intended to be static. Teachers and students can invent and save their own activities.

In addition, within this structure students can solve problems at various levels. Students and teachers can enable or disable the tools through an options menu. Students can use the tools to analyze figures or to work in a visual, empirical manner through measurement. Thus, they solve problems at different levels of mathematical sophistication.

In the next issue, we'll tell you about what happened when we took *Geo-Logo* into several different classrooms.

References

Clements, D. H., & Battista, M. T. (1991). *The development of a Logo-based elementary school geometry curriculum* (Final Report: NSF Grant No.: MDR-8651668). Buffalo, NY/Kent, OH: State University of New York at Buffalo/Kent State University.

Clements, D. H., & Battista, M. T. (1992). Geometry and spatial reasoning. In D. A. Grouws (Ed.), *Handbook of research on mathematics teaching and learning* (pp. 420-464). New York: Macmillan.

Clements, D. H., & Meredith, J. S. (1992). *Research on Logo: Effects and efficacy.* Manuscript submitted for publication.

Hoyles, C., & Noss, R. (1987). Synthesizing mathematical conceptions and their formalization through the construction of a Logo-based school mathematics curriculum. *International Journal of Mathematics Education, Science, and Technology, 18,* 581-595.

Hoyles, C., & Noss, R. (1988). Formalising intuitive descriptions in a parallelogram Logo microworld. In *Proceedings of the Fifteenth Annual Conference of the International Group for the Psychology of Mathematics Education* (pp. 417-424). Veszprem, Hungary: International Group for the Psychology of Mathematics Education.

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, metacognitive, problem-solving, and social abilities. His book *Computers in Elementary Mathematics Education*, published by Prentice-Hall in 1989, emphasized Logo. Through a National Science Foundation (NSF) grant, he developed a K-6 elementary geometry curriculum, *Logo Geometry*, published by Silver Burdett & Ginn in 1991. He is currently working with several colleagues on a second NSF-funded project, Investigations in Number, Data, and Space, to develop a full K-6 mathematics curriculum featuring Logo.

Julie S. Meredith is a mathematics education doctoral student at the State University of New York at Buffalo. She has taught secondary mathematics and computer science, gifted math at the middle school level, and mathematics methods courses. Along with Clements, she is currently designing and programming a new version of Logo for the NSF-funded Investigations project.

Support for the preparation of this material was partially provided by the National Science Foundation under Grant Nos. MDR-8954664 and MDR-9050210. 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.

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

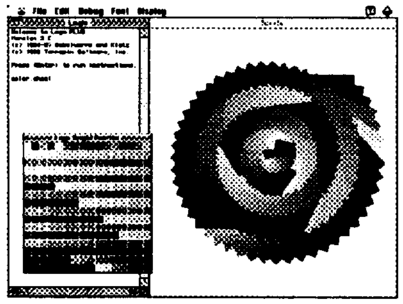
Logo PLUS™

for the Macintosh, v. 2.0

Logo PLUS for the Macintosh 2.0 is packed with exciting new features and 40 new commands.

Just imagine! Now you can...

- create pictures using 16 or 256 different colors
- create custom colors of your own
- preview your color selections in a Display window
- fill your shapes with different colored patterns
- pop to the Turtle window and type text
- turn your turtle shapes to point in four directions
- flip and rotate shapes in the enhanced shape editor
- lock and unlock your shape's heading
- create animations using 50 new ready-made shapes
- add music to your programs—Bach or the Beatles!
- select text and control the cursor with commands
- play 3 new Logo games: Solitaire, Jotto, and Darts



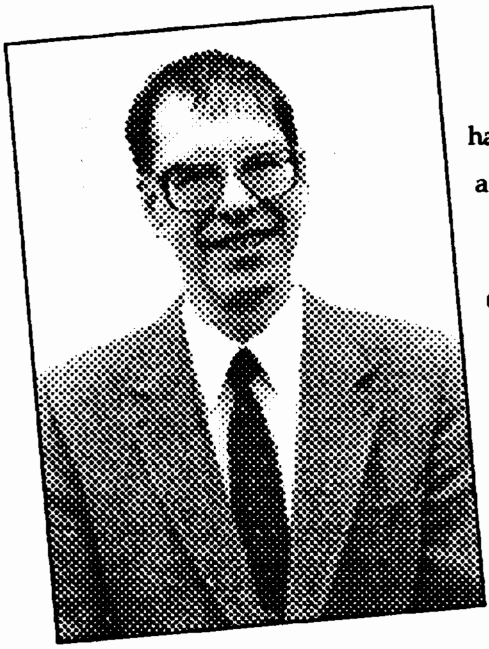
Requirements: Macintosh® computer running System 6.0.5 or higher and 1 MB of RAM or System 7 and 2 MB of RAM. Color optional. 32-bit compatible. Site License version is network aware.

Single version: **$99.95** Site License: **$450.00**
Lab Paks: 5-Pak, **$199.95**; 10-Pak, **$299.95**; 20-Pak, **$399.95**
Upgrade from Terrapin Logo/Mac Site License: **$150.00**
from Terrapin Logo/Mac: **$25.00**, **$7.50** for each additional disk
from other Terrapin Logo: **$50.00**, from any other Logo: **$75.00**
Complete set of all new documentation: **$15.00**
(when ordered with upgrade)
Please add 5% (minimum $3.50) for U.S. shipping and handling. Foreign charges as incurred.

Terrapin Software, Inc. 400 Riverside St., Portland, ME 04103 1-800-972-8200

LX122

Dave Moursund knows...



ISTE Executive Officer Dave Moursund is dedicated to improving education.

Since he first began working with computers in education in 1963, Dave has constantly sought to improve education at all levels through the use of appropriate computer-related technology.

As a continuing part of this effort, Dave's newest book—*The Technology Coordinator*—discusses the goals of education and how teachers and teachers of teachers can join in the enormous change process facing today's schools.

This book was written for educators who want to play a leadership role in the instructional use of technology. Dave uses interviews with technology coordinators to illustrate the needs, roles, and qualifications needed to accomplish the goals of educational technology.

In recent years, Dave has focused much of his writing efforts on problem solving and on staff development. Some of Dave's other books also available from ISTE include *Problem-Solving Models for Computer Literacy, LogoWriter for Educators: A Problem Solving Approach, Logo PLUS for Educators: A Problem Solving Approach* (co-authored with Sharon Yoder), and *Effective Inservice for Integrating Computer-as-Tool into the Curriculum.*

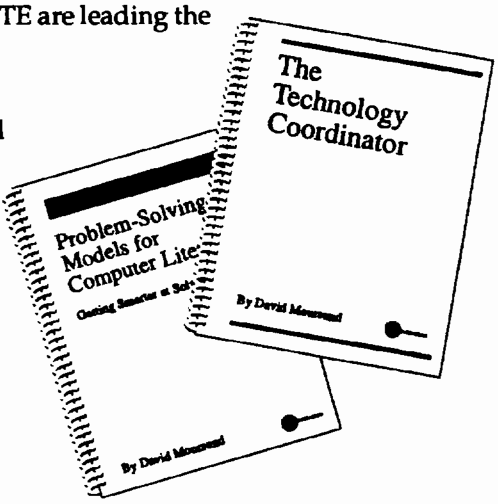
"Our educational system is not nearly as good as it could be," says Dave. "We know that computer-related technology can make substantial contributions to improving both the content of the curriculum and the instructional delivery system. Such technology can empower students and teachers."

ISTE believes that computer-related technology must become a tool that students and teachers use routinely if students are to be adequately prepared for adult citizenship in our Information Age society.

ISTE's wide range of periodicals, publications, projects, and other activities mirrors this vision and commitment.

Call us today to order Dave's latest books, and discover how he and ISTE are leading the way for the educators and school systems of tomorrow.

ISTE. Proudly putting you in touch with the world of educational technology.


The Technology Coordinator
By David Moursund
Problem-Solving Models for Computer Lite
By David Moursund

International Society for Technology in Education
1787 Agate Street, Eugene, OR 97403-1923
Order Desk: 800/336-5191 Fax: 503/346-5890

For pricing information or to order, see the order form in this publication.

Global Logo Comments

by Dennis Harper

Logo Exchange Continental Editors

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

Computers in Education: Shifting the Pedagogical Paradigm

From Instructionism to Constructionism

by José Armando Valente
NIED-UNICAMP

Introduction

This article is an attempt to reflect upon the work we have done regarding the use of computers in education in Brazil and other countries in Latin America. Two key questions addressed in this column are (1) Why do we want to use computers in education and (2) How could we be more effective educationally considering the economic investment developing countries have to make?

As I have mentioned in a previous issue of *Logo Exchange* (Valente, 1991a), in the majority of the Latin American countries the major projects concerning computers in education are based on the use of multipurpose educational software packages, such as Logo, text editors, and so forth. One of the reasons for this is that in Latin American countries there is very little educational software. Multipurpose software packages permit the development of computer activities in several domains and allow students from different grades and with different capabilities to participate in the process of educational computing. This is the trend of major computers-in-education projects in Latin America countries, and Logo fulfills this requirement.

However, from the pedagogical point of view, what do we gain or lose by using these multipurpose educational software packages? I will argue that they allow the pedagogical shift from instructionism to constructionism.

Different Uses of Computers in Education

Any attempt to classify the different uses of computers in education can lead to mistakes and be overly simplistic. However, it can be an interesting exercise and help us to understand the concepts of instructionism and constructionism.

Computers have been used to teach about computers, computer literacy, and practically any other subject. In computer literacy courses, the students use the computer to learn about programming, about computer principles, and about the implications of computer use in society. Although the majority of computers-in-education projects are about computer literacy, this is certainly not the kind of application we want to discuss here.

Teaching through the computer means that the student uses the computer to gain knowledge in a particular area. However, the pedagogical approach used varies greatly, oscillating between two poles. These two poles are characterized by the same elements: computers (hardware), software, and the student. However, what establishes the polarity is the way these elements are used. With the Instruction pole, the computer teaches the student through a particular software. With the Construction pole, however, the student "teaches" the computer through the software.

When the computer teaches the student, the computer assumes the role of a teaching machine. Pedagogically, this is the instructionism approach. Examples of software that implements this approach include tutorials, drill-and-practice software, and educational games.

When the student "teaches" the computer, the computer can be seen as a tool with which the student can solve problems or carry out other tasks, such as drawing, writing, and so forth. The computer as a tool constitutes one of the major sources of changing teaching practices and the process by which we manipulate information. On the other hand, the instructionism approach can be seen as an attempt to computerize the traditional teaching approach—it is more of the same, but now it is done with the help of the computer.

As a tool, the computer constitutes an environment in which learning takes place because the student is engaged in solving problems. This is not different from the learning-through-a-project approach, which is a well-known teaching methodology that can be done without the computer. However, the computer adds a new dimension to the traditional learning-through-a-project approach: the student has to express the solution to the problem through a computer language. This is an important twist that makes all the difference—it takes us from the traditional Piagetian constructivism to Papert's constructionism.

Although we are going to explain constructionism through Logo, as we will see later, constructionism transcends Logo and can be used to explain other ways of using the computer as a tool.

The Constructionism Approach

Papert (1986) first used the term *constructionism* in a proposal to the National Science Foundation. He introduced this term to explain the difference between Piaget's idea of the construction of knowledge (constructivism) and the construction of knowledge that can happen when the learner constructs a meaningful product, such as a work of art, a research report, or a computer program. In Papert's definition of constructionism, there is one major idea that makes this type of construction of knowledge different from Piaget's constructivism—learners build something that is meaningful to them. However, I believe what contributes to the difference between these two ways of constructing knowledge is the computer's presence—the fact that the learner is building something by using the computer as a tool. The computer usage requires certain actions that are very effective in the process of constructing knowledge.

When students interact with the computer in a Logo environment, they are actively building their own intellectual structure (Papert, 1980). Papert called this *Piagetian learning*. The students interact with computers and learning concepts in the same way they learn other concepts by interacting with objects of the world. However, after a decade of using Logo, we now know that in a Logo environment, more is going on than just "learning through interaction with the world" (with the computer), as Piaget observed.

To explain what happens when students interact with the computer in a Logo environment, I will concentrate on the graphics aspect of Logo. In the Logo graphics environment, as students solve a problem their interactions with computers are mediated by the Logo computer language or, more precisely, by Logo procedures. This interaction is an activity that begins with initial ideas the students develop about how to solve the problem. This idea is passed to the computer (or the turtle) in terms of Logo commands. Thus, students act upon the object "computer." However, this action is a description of the solution to the problem through the Logo computer language (Logo procedure). The computer then executes these procedures. The turtle "walks through" the procedure's commands and presents a result in terms of a picture on the screen. The students look at the picture being constructed and the final product, and they can then reflect upon them.

This reflective activity can produce several levels of abstraction, which, according to Piaget (1977), will affect the students' intellectual structure. The simplest level of abstraction is the empirical abstraction that allows learners to extract information from the object or from an action on the object, such as color, form, and weight. The pseudo-empirical abstraction allows learners to deduce a piece of knowledge from their actions or from the object. The reflexive abstraction allows the projection to a higher level of cognition, that which is extracted from a lower level and the reorganization of this knowledge in terms of previous knowledge (abstraction about the students' own ideas). The students' reflective activity can lead to one of two alternative actions: (1) doing nothing, when the students' original ideas correspond to the result presented by the computer, in which case the problem is solved; or (2) debugging, when the result is different from what the students intended. The debugging can be in terms of concepts in the subject area (the students do not know about angles), or in terms of some convention in the Logo language, or in terms of strategies (the students do not know how to apply a particular concept).

The debugging activity is facilitated by the existence of the computer program. This program is the students' descriptions of their ideas in terms of a formal, precise, and simple language. The Logo graphics language is easy to assimilate and the commands are similar to the terms we use in everyday life. This minimizes the arbitrariness of the conventions and the difficulty in translating ideas into code. These characteristics are not found in any other type of activity we do in the world. That is, as students act on the world they have, as a subproduct, the description of the ideas that support their actions. Also, there is a direct correspondence between each command and the computer's action. All these aspects available in the programming process facilitate the analysis of the computer program so students can find the bugs. The process of finding and correcting bugs constitutes a unique opportunity for students to learn about particular concepts involved in the solution to the problem or about problem-solving strategies. Students can use their programs to relate to their thinking at the metalevel. They can analyze their programs in terms of the effectiveness of their ideas, strategies, and problem-solving styles. In this way, students begin to think about their own thinking (reflexive abstraction).

However, this process does not happen just by placing students in front of a computer. The student-computer interaction needs to be mediated by a professional who knows about Logo ideas from a computer-related, pedagogical, psychological point of view. This is role of the students' parents, friends, or even their communities. They can use all these social elements as sources of ideas, knowledge, or problems to be solved through the use of the computer.

Constructivism vs. Constructionism

Why do we need another term to define what goes on in the Logo environment?

As was mentioned before, one reason is that the student-object interaction is mediated by a computer language. Through this language students can describe their thoughts and then execute and debug them. This adds another dimension to the already-known interaction with the world that Piaget observed and described as the source of construction of intellectual structure.

Second, the interaction that Piaget mentioned was not necessarily mediated by an expert. In Piaget's studies, the child interacts most of the time with objects, and the role of the experimenter is to use the clinical method to learn as much as possible about the child's mental structure. The experimenter's role is not to teach or to lead the child to learn. In the Logo environment, the Logo teacher has to learn about the student's ideas and to intervene appropriately in the situation so the teacher can be effective and contribute to the student's understanding of the problem in question. Thus, the Logo teacher's role involves much more than the use of the clinical method. The model that best describes the Logo teacher's role is the action that takes place within the Zone of Proximal Development (ZPD) as defined by Vygotsky (1978)—"the distance between the actual developmental level as determined by independent problem solving and the level of potential development as determined through problem solving under adult guidance or in collaboration with more capable peers" (p. 86).

Third, students are placed in a social context and are not isolated from their communities. This social context can be used as a source of intellectual support or contextual problems to be solved, as Paulo Freire (1970) has suggested. The student can learn from the community as well as help the community by identifying problems, solving them, and presenting the solution to the community. This is the approach used in the Genese Project, the computer-and-education project in the municipal educational system of São Paulo (Valente, 1992a).

Thus, the theoretical support for the activity that takes place in the Logo environment not only comes from Piaget. Other scholars have contributed by explaining other levels of interaction and activities in the Logo environment. These different levels of interaction and their contribution to the students' intellectual development are much more than what is captured by Piaget's constructivism. Papert's constructionism is an attempt to better characterize the construction of knowledge that takes place in the Logo learning environment.

Constructionism Transcends Logo

The activities that take place in the Logo graphics environment are ideal for explaining constructionism. However, other ways of using the computer as a tool allow construction of knowledge according to the constructionism approach.

As we mentioned before, the constructionism approach happens when we use certain aspects of Logo, such as the Logo graphics. The computer commands are simple to learn, the description of spatial problems in terms of Logo graphics commands is straightforward, and the computer feedback is a figure that facilitates interpretation, reflection, and debugging. In other Logo domains, such as list processing, the description, reflection, and debugging activities are not as simple as in the graphics domain. First, the description of recursive processes is not an everyday type of activity. Second, the execution of a recursive procedure in list processing is very opaque, making it impossible to see what the computer is doing. Third, the reflective action is not helped by the computer's action. Thus, it is not by chance that Logo is well known for its graphics.

When we think in terms of other computer tools, we have to analyze them as to whether they present the same facilities as Logo graphics in order to set up a learning environment according to the constructionism approach. For example, as a computer language, Pascal has the same kind of problems encountered in Logo list processing. Pascal is not very simple to learn. We have to understand about certain computer representation structures (list and arrays) or about algorithms before we are able to describe the solution of the problem to the computer. Most of the feedback is not graphic, and debugging can be very problematic—to find the bug is a major exercise. It is not impossible but it is very difficult to set up a constructionist Pascal learning environment. Other computer languages and tools have the same problems. For example, languages for database manipulation can present some of the same barriers encountered in Pascal that prevent learning through constructionism.

With text editors, the difficulties are of a different nature. If we see text editing as "teaching" the text to the computer, we can include these tools in the Construction-of-Knowledge pole, and we can analyze them in terms of the constructionism approach. First, it is very simple to use them and to describe our thoughts to the computer. However, the execution part is problematic. Text editors can execute only the formatting aspects of the text or some aspects of writing style, but they cannot yet execute the content of the text and present feedback in terms of whether the content represents what we mean. Thus, to get helpful feedback we need to print the text, give it to others to read, and tell us if they understand the content. Only with this information can we then debug our ideas and the text.

Therefore, to be able to construct knowledge through constructionism, the computer software needs to have certain characteristics to facilitate the description, reflection, and debugging activities. Programming languages seem to have most of these characteristics, even though (depending upon which language we use) we may have these activities more or less facilitated. However, as was pointed out to me, programming today does not need to be done in terms of a sequence of written commands (Ackermann, personal communication, 1993). We can have tools, such as Paintbrush, that help us solve a problem by using the mouse; however, it can leave a trace as the activities are done. Thus, as we select items in the menu or do things on the screen, the computer collects this information and constructs a procedure. The trace then becomes the description. *Mondrian*, a software product developed by Lieberman (1992), has this feature.

Conclusion

In the beginning of this article we set up the task of answering questions about why we want to use computers in education and about how we could be more effective educationally. The idea that we have tried to convey to people in charge of making decisions and implementing computers in education in Latin American countries (especially in Brazil) is that the computer should be the catalyst of a change in the educational approach currently used. The new approach would be one that promotes learning instead of teaching, that puts the control of the learning process in the hands of the students, and that helps the teacher to understand that education is not only the transference of knowledge but a slow process of constructing it. In other words, we need an approach that shifts the pedagogical paradigm from instructionism to constructionism. In Latin American countries, the objective of introducing computers in education should not be simply to be up to date with the "new technologies." If it were, the implementation of computers in education would be far too costly.

Fortunately, we have a few examples of using computers in education that show that if the right decision is made it is possible to implement computers in education in a massive way and still promote the constructionism approach. These examples include the projects in Costa Rica, Venezuela, and the city of São Paulo (Valente, 1991a), and more than 50 centers in Latin America using Logo in special education (Valente, 1991b). We hope we will see more of this approach rather than the instructionist computers-in-education approach that seems to be taking over the U.S. educational system.

References

Freire, P. (1970). *Pedagogy of the oppressed.* New York: The Seabury Press.

Lieberman, H. (1992). *Mondrian: A teachable graphical editor.* Unpublished manuscript, Massachusetts Institute of Technology, Media Laboratory, Visible Language Workshop, Cambridge, MA.

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

Papert, S. (1986). *Constructionism: A new opportunity for elementary science education.* Proposal submitted to the National Science Foundation, Massachusetts Institute of Technology, Media Laboratory, Epistemology and Learning Group, Cambridge, MA.

Piaget, J. (1977). Recherches sur l'abstraction réfléchissante. *Études d'épistemologie génétique.* PUF, Tome 2, Paris.

Valente, J.A. (1991a). Report from Latin America. *Logo Exchange, 10*(2), 43-48.

Valente, J.A. (1991b). *Liberando a mente: Computadores na educação especial. Gráfica da UNICAMP, Campinas, São Paulo.*

Valente, J.A. (1992a). Logo and Freire's educational paradigm. *Logo Exchange, 11*(1), 39-43.

Valente, J.A. (1992b). Logo and special education in Latin America. *Logo Exchange, 10*(4), 36-40.

Vygotsky, L.S. (1978). *Mind in society: The development of higher psychological processes.* Cambridge, MA: Harvard University Press.

Dennis Harper
1113 Legion Way SE
Olympia, WA 98501

ISTE Books & Courseware Order Form

How to order ISTE products advertised in this publication

To order ISTE products advertised in this publication, please find the product(s) in the following list and enter it on the attached form. To receive a free 32-page ISTE catalog with a complete list of ISTE products and services, please call our toll-free order number listed below. ISTE Members receive an additional 20% discount when ordering 10 or more of the same title of ISTE Published products. ISTE Published products are indicated by a • symbol. Sale price items are not eliglible for an additional discount.

product name	nonmember prices	member prices
	1-9 copies	1-9 copies
• Macintosh Step by Step	11.65	12.95
• Education, Technology, and Paradigms of Change for the 21st Century	12.00	12.00
• The Technology Coordinator	21.55	23.95

code LX 2

Name ____________ Membership # ____________

Address ____________

City ________ State ________ Zip/Postal Code ________

Country ________ Phone ________

Shipping & Handling

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

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

GST Registration Number 128828431

ISTE Books, Courseware, & Nonmember Subscriptions

Qnty.	Title	Member Unit Price	Nonmember Unit Price	Total Price

SUBTOTAL	
Add Shipping, based on SUBTOTAL	+
Add additional 5% of SUBTOTAL if shipped to P.O. Box, AK, HI, or outside U.S.	+
Add 7% of SUBTOTAL for GST if shipped to Canada	+
If billed with purchase order add $2.50; if COD, add $2.75	+
TOTAL PAYMENT	=

Payment Options

☐ **Payment enclosed.** (Make checks out to ISTE. US funds only.) Non-U.S. orders must be prepaid with U.S. funds or credit card.

☐ VISA ☐ Mastercard ☐ Discover Card

☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

__________ Exp. Date

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

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

☐ **Please send me a free 32-page ISTE catalog!**

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

Please send me membership/subscription information! ☐

ISTE, 1787 Agate St., Eugene, OR 97403-1923 • order desk 800/336-5191 • fax 503/346-5890

KING GEORGE WOULD HAVE HATED THIS CONFERENCE...

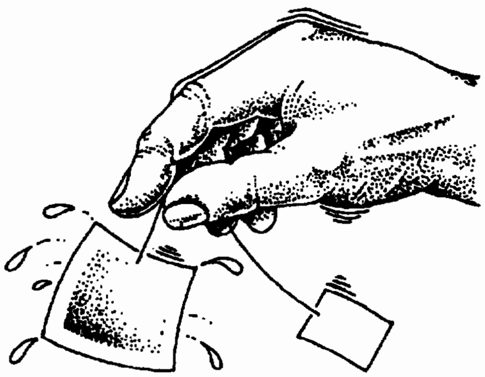
After all, it's going to be everything he was against...new ideas, new thinking, new challenges—*Revolution!*

Taking place in the historically non-traditional city of Boston, Massachusetts, NECC '94 will offer possibilities for exploring educational uses of technology that have never been more exciting.

The special focus for '94 is on educators who have been using technology to create *revolutionary* learning environments in their classrooms. We also hope to provide participants with more hands-on opportunities to experience groundbreaking technologies than you can shake a tea bag at!

Session topics include:

- multimedia and distance learning,
- uses of computer-based technologies at all academic levels and in all academic disciplines,
- integration of technology into the curriculum,
- technology and school restructuring,
- teacher education,
- technology and the arts,
- and new and emerging technologies for instruction and information management.



The conference also boasts one of the largest and most progressive trade shows of its kind. Nearly 500 exhibitors will be on hand to give participants the chance to explore the latest technologies up close and personal.

With NECC '94, we invite you to share in *Recreating the Revolution*—and to recognize that nothing short of an educational revolution will provide the new basic skills necessary for the Information Age.

Besides...we think Paul would have *loved it!*

For more information on attending workshops, registering, and finding accommodations, please call, write, fax, or send email to:

Donella Ingham, NECC '94
1787 Agate Street, Eugene, OR 97403
(PH) 503/346-2834
(FX) 503/346-5890
(IN) donella_ingham@ccmail.uoregon.edu

Recreating the Revolution
NECC '94 *Boston*

Hosted by Lesley College and Bolt, Beranek & Newman, Inc.

"Awesome"

"Way cool."

"Slammin'."

"Totally there."

"Swinging."

"Wicked good."

(And these are just the teachers' comments.)

The fact is, whenever we show our three new educational software products to teachers and curriculum coordinators, they get as excited as kids.

And for good reason.

You see, our line of learning software for Macintosh® computers gives teachers of grades 4-8 a unique way of motivating their students.

For one thing, **MicroWorlds**™ products are specifically designed for the classroom. Their flexibility lets students with all different learning styles use what they know to tackle new learning experiences.

What's more, the **MicroWorlds** packages were designed by LCSI, the company known for its award-winning educational products.

Take **MicroWorlds Math Links**™ - it doesn't camouflage math as some space game. Instead, it lets you link math to art, science, and social studies. Students don't just study math, they think mathematically, using math to develop projects ranging from kaleidoscopes to Navaho textile patterns.

With **MicroWorlds Language Art**™ you'll encourage students to explore words and images. Write text in any shape, color or direction. Add effects such as scrolling text, animation. Projects, including Visual Poetry, Ads, Haiku, help you assist students in developing writing skills.

MicroWorlds Project Builder™ gives you the tools to develop a problem-solving, creative-thinking, learning culture across the curriculum. And features like text, drawing tools, animation, and music give students the tools to create anything from simple ecosystems to dynamic maps.

Plus there's more: Each of these products is offered under LCSI's well-known site/network license - the most flexible policy available to schools today.

So for information or a **free** demo disk, call us today at **1-800-321-5646**.

We think, like, you'll be blown away.

MicroWorlds

LCSI®

International Society for Technology in Education
1787 Agate Street, Eugene, OR 97403-1923
Order Desk: 800/336-5191 Fax: 503/346-5890