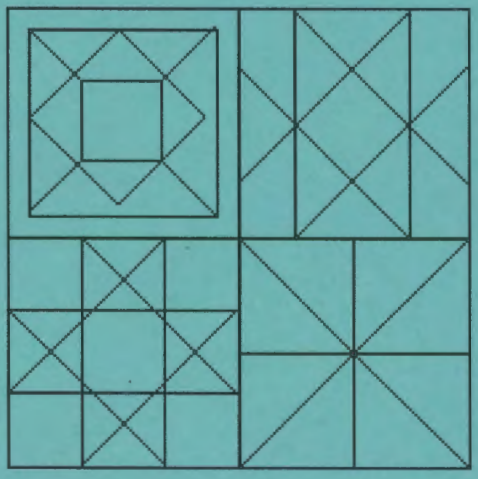
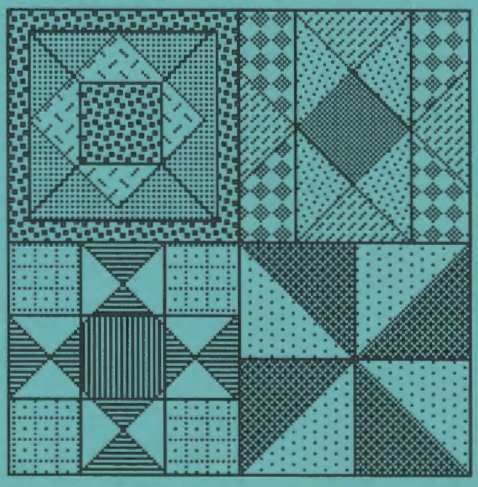
Journal
of the ISTE
Special Interest Group
for Logo-Using
Educators

LOGO EXCHANGE

Summer 1994 Volume 12 Number 4

In this issue: *Viewing the RIGHT Way: A New ANGLE on Quilting With Logo*





Also—

The Crystal Rain Forest

Color Displays and Hypermedia

King Arthur Had a Daughter

International Society for Technology in Education

LX Volume 12 Number 4

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 3	July 1, 1994
Volume 13, Number 4	Oct. 1, 1994
Volume 14, Number 1	Feb. 1, 1995
Volume 14, Number 2	Apr. 1, 1995

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. Second-class postage paid at Eugene OR. USPS# 660-130. ISTE is a nonprofit organization with its main offices housed at the University of Oregon. ISSN# 0888-6970

LOGO EXCHANGE

Volume 12 Number 4 Journal of the ISTE Special Interest Group for Logo-Using Educators Summer 1994

Contents

From the Editor

Discouraged? ...Don't Despair!

For more than 13 years now, I have worked with one version of Logo or another. Sometimes it was teaching elementary or high school students, sometimes it was teaching teachers-in-training, sometimes it was master's- and doctoral-level computer educators, and sometimes it was in the process of writing a book. It seems that there has always been a new version of Logo, a new challenge, a new group of students to reach.

During those years, whenever I encountered "kids" whom I could engage in conversation and who had not been among my students, I asked if they knew about Logo. I would then ask what they thought of Logo. I often got answers like "Logo? Is that the thing with the turtle? Oh, yeah, it's OK, I guess." End of conversation. My interviewee was off to other things. These encounters were universally discouraging.

Lots of Logo—Little Understanding

This year, for the first time since I came to the University of Oregon, I taught a group of freshmen. The class was a freshmen seminar. These special-topic seminars are always small classes made up of 20 or fewer students. They are designed to allow freshmen to get to know a faculty member well and to interact in a small-class environment. The topic I chose for my seminar was an exploration of using technology for communication in the modern world. The students worked at improving their personal productivity skills while at the same time exploring such topics as user interfaces, learning with technology, the information highway, and how we will live and work in the 21st century.

One of the readings for the class was an article that discussed *Children's Machines,* Papert's new book (Schwartz, 1993). The students were to reflect about the article in their journal. Almost all of my 20 students had been exposed to Logo, and nearly every one of them commented in their journals about their experience as they noted their reactions to the article. Their responses were universally depressing. They were not unlike the responses I had received from younger students in earlier years. In general, their experiences had been neutral to quite negative. They thought Logo was a bit silly. They couldn't see the point. Few even recognized that they had learned any programming or that there was any power in the language or the environment. After reading their journals, I was thoroughly depressed.

It is clear that Logo has infiltrated the elementary curriculum and that most elementary students see Logo at some time in their education. Unfortunately, it also seems clear that they simply don't "get it." It's unclear whether that is because their teachers don't "get it," because their exposure is so limited, or because Logo just doesn't work somehow in our current school environment.

Logo and Hypermedia—Again

Also disturbing is the number of teachers I have encountered in the last year or so who are switching from Logo to a hypermedia product, usually *HyperCard, HyperStudio,* or *LinkWay.* Recently, I presented a conference session on *MicroWorlds,* the new LCSI product. Several people came up to me afterwards, disturbed that they had just purchased a hypermedia product in addition to their version of Logo. They realized that *MicroWorlds* would serve the purpose of both their Logo product and their hypermedia product.

Of course those of us who have been working with Logo for years know that most versions of Logo allow you to connect "pages" to create nonsequential documents much like *HyperCard* stacks. The buttons in *MicroWorlds* simply make this capability more obvious. Somehow those of us using Logo and those marketing Logo just haven't made the point that Logo can be thought of as a "scripting" language and that Logo files or pages can be thought of as "cards."

Again I was discouraged. It has seemed of late that everywhere I turn, the news about Logo is depressing. Perhaps that's because I'm not teaching Logo this year and therefore don't see the excitement that Logo generates in a class.

Insight Makes the Difference

Recently I asked a colleague who knows a good bit about computers and about programming but nothing about Logo to review my latest Logo book. I wanted to see how a non-Logo person would react to the content. Her initial reaction to Logo was one of excitement. She continued to have a delightful time as she worked through the book. When she was finished, she wanted to know more and simply could not wait until the second book was written. She was hooked. In a recent E-mail message to me, she said that she had figured out

that Logo was a grown-up's programming language disguised as a kid's toy.

Wow! What insight! If Logo can still evoke such a response from a computer-savvy adult, then Logo is still in a position to "compete." How exciting to know that Logo can stir someone who has dabbled in high-end graphics software and sophisticated programming environments.

If you are teaching Logo, you know that your students are excited and that they learn. But are you an advocate for Logo? Do you let your colleagues who are excited about hypermedia know that modern versions of Logo can do nearly everything that hypermedia products can do? Perhaps if we in the Logo community were more vocal, the rest of the world would more often share our excitement.

Reference

Schwartz, E. (1993, July 26). Can computers re-create the classroom? *Business Week*, p. 12.

Sharon Yoder
DLIL College of Education, Room 170C
1215 University of Oregon
Eugene, OR 97403
503/346-2190
Internet: yoder@oregon.uoregon.edu

A Farewell to Some Columnists

Long-time readers of *Logo Exchange* will be sorry to hear that Sandy Dawson is not continuing as a columnist for *LX* as of next year. As many of you know, Sandy has been a strong voice advocating the use of Logo in mathematics. His many contributions to *LX* have provided us with insights into thinking and learning, problem solving, philosophy, and Logo activities worldwide. His contributions will be missed.

Mark Horney, editor of the Extra for Experts column, will also be leaving as a regular contributor. For the past several years, Mark has assumed the role of both editor and writer of articles aimed at "high-end" users of Logo. We will certainly miss his challenging and thoughtful contributions.

An Open Letter to the Logo Community

As this editorial year draws to a close, *LX* finds itself losing at least two columnists. Over the years, *LX* has been very dependent on the tremendous contributions of time and energy on the part of such loyal columnists. As we lose columnists, we lose important materials for *LX*.

In the past, *LX* has had more articles available for publication than it could publish. This also is no longer the case. Very few submissions have been received during this last editorial year. With the loss of columnists and of contributors, the future of *LX* may be in jeopardy.

Clearly, some decisions need to be made about *LX*. It seems that SIGLogo needs to assume the responsibility for making those decisions. Perhaps the meeting at NECC in Boston is the time for such a discussion.

In any case, without the addition of some columnists and/or submissions of articles, next year's issues of *LX* will not be as rich or as long as last year's were. Help us out by sharing with the Logo community your thoughts, your ideas, and your projects. Help keep *LX* vital!

Sharon Yoder, Editor

It's All Relative

by Tom Lough

It is so much fun to introduce Logo to young students. Once they learn the rudiments—ah-ah, note that I did *not* say the basics!—of moving the turtle around the screen, they usually want to explore on their own. The variety of styles, paths, patterns, and figures they produce is staggering.

Sooner or later, they stumble upon a surprise. One I especially enjoy happens when the turtle is facing the bottom of the screen—with a heading of 180—and the students want to turn the turtle to the left. When they type

```
LEFT 90
```

the turtle carries out the instruction. However, this action usually precipitates a most animated discussion! I can almost always tell this surprise is operant by the particular contortions of the students as they try to line up their heads with the former orientation of the turtle in an attempt to figure out why it turned "right" instead of left as they had instructed!

Sooner or later, the "Aha!" experience manifests itself; and I hear a comment such as, "Hey! I know! It turned to *its* left!" Once this point is realized and understood, the students' Logo learning can move to an important level: grasping the concept of relative motion.

As we adults already realize, the turtle follows the LEFT and RIGHT commands according to its own body coordinate system, regardless of which direction it is heading. This refers to the body syntonicity Seymour Papert mentioned in *Mindstorms*. To an outside observer, such as the students in the case just described, the movements are sometimes a surprise. There is a good lesson in this for us all.

In a limited way, students are like turtles. As teachers, we issue instructions to students and expect the instructions to be carried out consistently. When a student does something unexpected, we are surprised. But before we take any corrective or disciplinary action, it might be worthwhile to check the orientation of the student. How they are thinking about something and what they do about something is usually relative to their "heading" at the time.

Questions such as "What is your understanding of the instructions?" and "Would you help me to understand what you have done?" sometimes elicit amazing insights and penetrating perspectives that you might never have considered. (I have learned a few humbling lessons about ambiguous instructions myself!)

Like the turtle facing the bottom of the screen, perhaps the student carried out the instruction consistently relative to his or her frame of reference but apparently contrary to the expectations of an outside observer. Like the students observing the turning behavior of the upside-down turtle, we might benefit from a positively curious query and discussion instead of a hasty conclusion.

Finally, like the students trying to line up their heads with the orientation of the turtle, perhaps we could try to find new ways to align our thinking relative to that of our students. This would help us appreciate their unique perspectives, and, in the process, we just might become a bit more effective as teachers.

FD 100!

PS: What Logo surprises have you seen your students discover? I'd love to hear about them.

Tom Lough
Founding Editor
PO Box 394
Simsbury, CT 06070

An Adventure in Logo

by Dorothy Fitch

Do you want a new way to introduce kids to Logo that is easy to understand, requires little (if any) teacher guidance, allows for creativity and exploration, and is lots of fun?

Meet *The Crystal Rain Forest*, a new product from Terrapin Software. This adventure game is reviewed elsewhere in the issue, but I've been given an opportunity to show how Logo skills are taught in this tropical rain forest setting.

The Crystal Rain Forest was developed in England, where Logo is a required part of the curriculum. The adventure played out in the program solves the problem of how to introduce Logo to all students, regardless of their teacher's computer literacy or Logo experience. It is essentially a self-guided introduction to Logo for kids in grades 3-8. Teachers may find it useful as their own introduction to programming in Logo. The adventure does not cover words and lists, music, or shapes; but it does let kids experiment with basic turtle graphics commands, repeated patterns, variables, and procedure definition, as well as save and print their work. Not a bad start! And they get to save rain forests to boot!

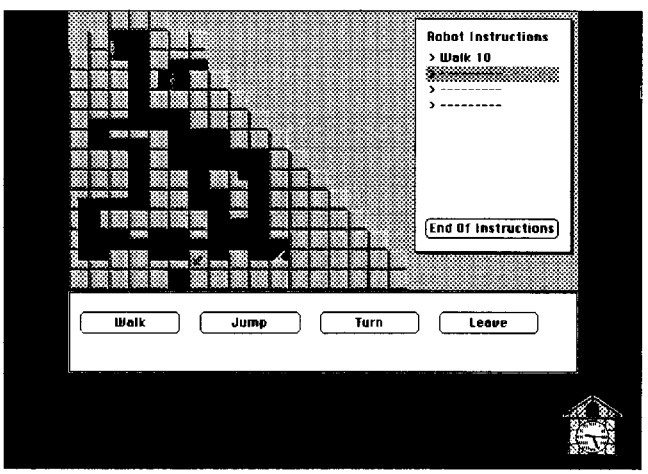
How does this adventure do all this? Through a carefully sequenced series of challenges and puzzles that fit seamlessly into the plot. Although the order of the activities is fixed so that Logo skills are taught in a reasonable order, the adventure does not give the impression of having just one path through it. If you reach the location of a challenge for which you are not prepared, your path is blocked but you are provided with a hint that something else needs to happen first. And there are a few plot twists and turns that I won't divulge. It all makes for an extremely captivating story that each student will experience in a slightly different way.

Here is the premise: The planet Oglo is in trouble. Its rain forest is vanishing quickly. The greedy Cut and Run gang is senselessly slicing through it and has poisoned the king. He can be cured only by the crystals that are hidden deep in the rain forest. Only you can save the rain forest, the king, and the planet. Search for the magic crystals, but be careful—the Cut and Run gang will try to stop you. As you hunt for clues, you will have to keep your wits about you.

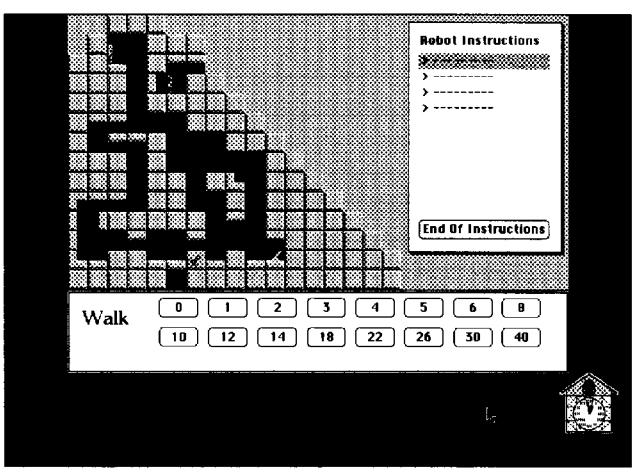
After an introduction that explains all this, you are charged with finding Professor Roberts, who can help you locate the magic crystals. As you explore the town and the rain forest, you face the following challenges, which teach you Logo one step at a time.

1 • Robot Temple

The passageway to Professor Roberts' workshop is blocked. Your job is to guide a robot to the door switch to open the way by clicking the mouse on the appropriate command. Select WALK and a number to tell the robot how far to go. Choose TURN to make it face in the opposite direction. Choose JUMP to make it hop up a step.



In the first maze, you give instructions one at a time to move the robot to the door switch. In the next maze,



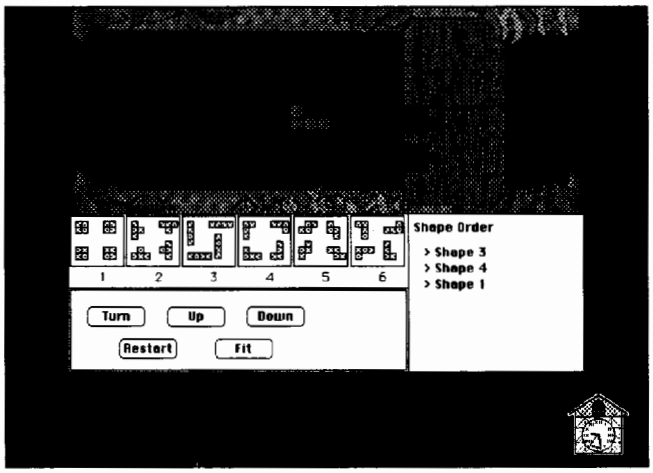
you are encouraged, though not required, to enter several instructions before running them. After you have opened the door and led Carlos, your native guide, through the maze, you can explore Bridgetown and hunt for more clues.

This activity introduces students to the idea of moving an object on the screen. They learn to give an input to a command and to create a series of instructions.

2 • Garden Bridges

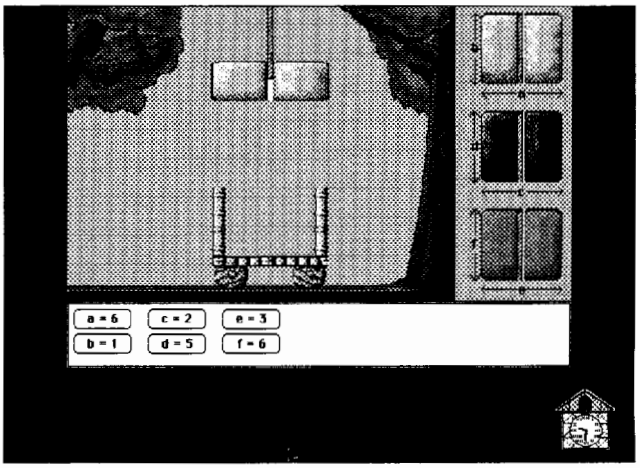
In this activity, you rotate shapes and move them up or down to float into holes in the bridge. Like Tetris, this is a lesson in the rotation and translation of shapes. You must plan ahead for the more complicated puzzle, and you must be sure to put the instructions in the right order.

Later, in Crystal Logo, students can put these ideas into use. They will know that they can move and turn the turtle before drawing a shape for different effects. The relative turtle commands will make sense. They will have learned to plan ahead.



3 • Monkey Puzzle

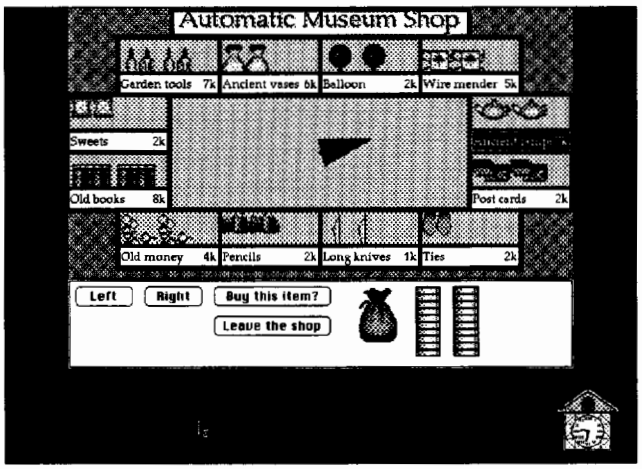
The next puzzle, at Herbert's tree house, is a lesson in the effect of changing inputs. You must help monkeys put leaf packages into shipping boxes, changing the dimensions of each package so that it fits the container exactly. Clicking on a box dimension labeled *a* or *b* (and later *c* or *d*) lets you change its value, and consequently you can change the length of that box side.



Students learn that changing a value alters the result of a command.

4 • Automatic Shop

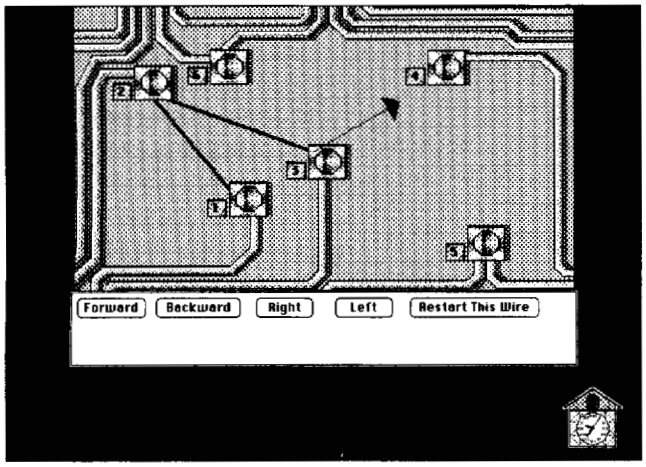
Here you can spend the money you earned at the tree house to buy items you will need. To buy an item, rotate the pointer right or left to aim at it and specify the number of degrees (5, 10, 15, ... 90) to turn. Should you buy the ancient lanterns, the garden tools, or some sweets? You can return later to buy different items.



As students spend their money, they learn about angles and relative turns.

5 • Wire Connections

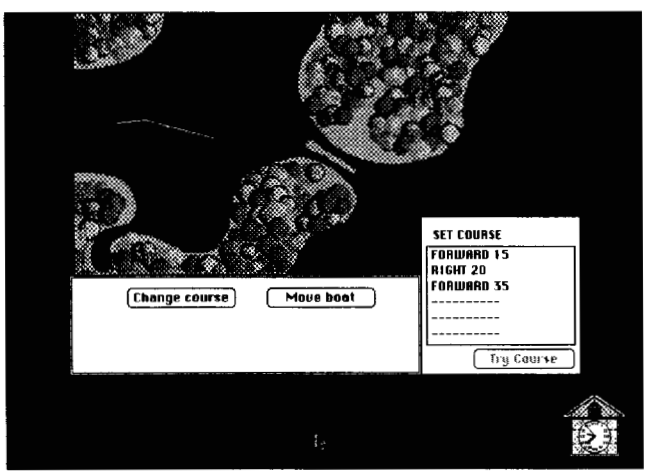
An important door control has been damaged. Can you fix it? You must draw lines to connect the terminals to make it work again. Give the turtle instructions to move forward or back and turn right or left. After you open the door, your quest for the magic crystals continues.



This connect-the-dots puzzle gives students good practice in moving the turtle.

6 • Navigating the River

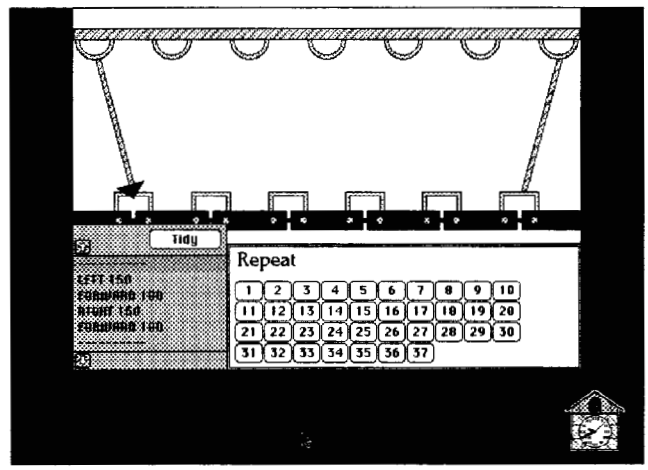
On the river, you must learn to maneuver the boat by using its computer. You can move it forward or back and turn it right and left. Initially, you give one instruction at a time. Later, you give a series of instructions, which you can test before moving the boat. You will negotiate challenging rocky passages and narrow twists and turns along the way. At each jetty, you hunt for more clues, meet more people (and rain forest animals), and solve more puzzles.



Controlling the boat gives students lots of practice in estimating distances and angles.

7 • Rope Bridges

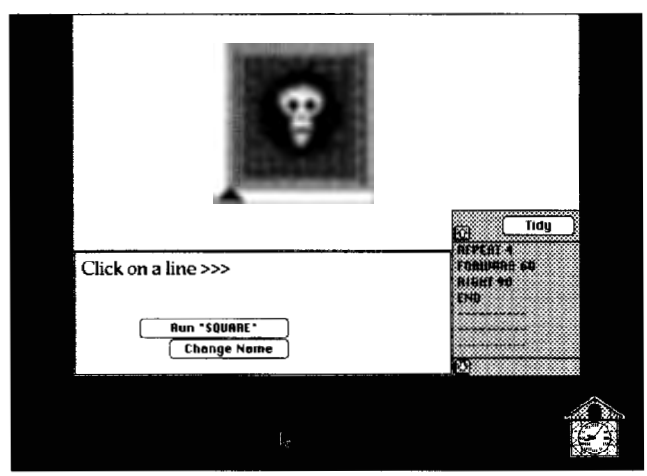
As you fix the rope bridges at one of the stops, you learn to use REPEAT to create a pattern. How many times should you repeat your instructions? How far should you stretch the rope? At what angle should you turn it? Using trial and error, you figure out the REPEAT instruction that will mend the bridge so that you can continue your adventure.



Students learn the power of the REPEAT instruction in this challenge.

8 • Safety Nets

In the final puzzle, you create nets to catch tins of poison before they pollute the river. In this puzzle, a triangular-shaped net gets ready to catch a tin of poison. The net is given a name and defined by using a REPEAT instruction. To catch all the tins, you must put a series of different-shaped nets in the right order.



To solve this challenge, students learn how to define a procedure and use it as part of a larger task.

• Crystal Logo

Finally, students use Crystal Logo to make their own crystals. They already know how to move and turn the turtle, repeat a pattern, define procedures, and combine them to make a larger program.

After students experiment with their own crystal designs, the story draws to a conclusion. Although it is part of the adventure, Crystal Logo can be used separately and the names of its commands can be modified. Matching its commands to your version of Logo will ease the transition to the complete version of the language.

The ease of use of the software, its carefully paced progression of skill-building activities, and the engaging plot help students of all abilities to learn Logo quickly and easily.

Happy *Crystal Rain Forest* 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
800/972-8200

The Crystal Rain Forest

by Irene Smith

OK, OK...so I'm running a bit late with this review. Your editor, Sharon Yoder, has been telling me that now is the time! But I was just trying to save the King's life and couldn't stop. In case you haven't read Dorothy Fitch's article about *The Crystal Rain Forest* that appears elsewhere in this issue, let me explain.

A Logo Adventure

Terrapin Software will soon release a new product for the Logo world to enjoy. This game-based program, which introduces students to basic Logo commands, has some fun features. The King? Well, the program establishes the traditional conflict between the good guys and the bad guys. To save the forest from total destruction, the King has decided that no more trees are to be harvested. The Cut and Run gang dislikes this decision and poisons the King. The challenge is in place. Can you overcome the obstacles placed in the way as you search for the magic crystals that can save the King's life? The tools to meet this challenge are found in the world of Logo commands.

The Crystal Rain Forest requires a Microsoft-compatible mouse and enough hard drive space to deal with large graphic and sound files. The graphics, while simple, are colorful and interesting. Some screens have active spots for the user to explore further. Clicking on these areas open new screens or pathways. Other scenes contain automatic animation. The story is explained in the text box displayed on the bottom of the screen. Directions to the user are clear and easy to follow. When the user is not able to interact with elements showing on the screen, the program restricts the cursor movement to the text area. The user navigates from one screen display to the next by clicking on the buttons that appear on the lower right of the screen.

A Look at the Program

As the program opens, the user has some options. Three buttons are visible that allow one to "start," "setup," or "quit" the program. The features on the setup button provide a great deal of the flexibility contained in this program. Unlike many other game products, the user is able to start this game at any part of the program simply by using the dialog box opened from the setup button. This means that students can redo sections to improve their skills or restart the game where they left off the previous day. Scoring is not an element of the program. The user succeeds if he or she creates the magic crystals to save the King. This activity is the culmination of learning the basic Logo commands, such as **forward, back, right, left, repeat,** and so forth. The program in no way forces the user to proceed at any particular speed. The user controls the progression of the game at all times.

Some Special Features

I particularly liked the sound feature in this program. After substitute teaching in the elementary classroom where computer games were allowed after school, I consider sound very carefully when I look at a program! The sound is used to emphasize actions. It is not the dominating feature on any screen. It has been carefully integrated into the presentation and is a natural part of the scene. At no time did I feel that the sound had become oppressive—a major breakthrough in a children's game-like environment.

The program allows you to save and print any screens. Yes, to do them justice you will need a color printer; but for some students, printing the picture and coloring it is an acceptable option. It is even possible to control this saving action and capture only a particular section of the screen display. Therefore, if a student creates a particularly effective crystal, he or she can easily save a copy of the graphic.

Two Levels of Content

In addition to the various Logo skill levels introduced as the student moves through the game, the program also provides a relevant environmental message. Students today do focus on environmental issues; and while this is not the primary focus of the game, it does communicate this message.

The intent of the program is, of course, to help students learn some of the basic commands in Logo. The events in the game are controlled with an increasingly difficult interaction with Logo until the final tasks require the student to write basic procedures. For some students, I think the introduction to Logo in this manner will be a welcome change. Students get to see the visual results of **forward, right,** and so on before they try to write code using those terms. Having the turtle pivot and move in response to the commands provides an understanding of the terms on a deeper level than merely being given verbal definitions.

Please don't think that this program will complete the study of Logo. While it does provide a good introduc-

tion to Logo, its limitations will soon become restrictive. Use it as an introductory tool, and have your students gain some expertise in a fun and nonthreatening environment. When they have mastered this program, move them on to something else—like Logo!

The Crystal Rain Forest varies activities and provides for different levels of challenge, but it still limits the user to the story line and tasks defined within that story. For older students, the time needed with this product will be minimal although many students will enjoy going back to it for a relaxing review session. Note, however, that there is little incentive in the program to encourage efficient code, so remember that good programming style will develop only if you encourage your students to demand this of themselves.

And ...

Now, if you'll excuse me, I left my boat somewhere upstream and I think if I just rewrite that last procedure correctly this time, I won't be taking the boat onshore again! I'll see you when I finish locating the crystals. And, yes, I'd better hurry. This review needs to go to press!

Irene Smith completed her doctorate in computers in education from the University of Oregon in 1993. She is currently working with the Lane County Educational Service District in Eugene, Oregon. In addition, she works closely with International Society for Technology in Education (ISTE) to develop curriculum, grade distance education courses, and provide technical support.

Irene Smith
ISTE
1787 Agate Street
Eugene, OR 97403
503/687-5813
Internet: smithire@oregon.uoregon.edu

Publisher's Note

By no means do we feel that *The Crystal Rain Forest* offers a complete Logo experience. Rather, we see it as an engaging introduction to turtle graphics and procedure building, leading to further exploration of Logo using a full version of the language. Although the Logo skill-building activities are tied directly to the plot and are therefore limited in scope, Crystal Logo, the version of Logo that is included, can be used independently from the rest of the adventure and it can be used to tackle any Logo graphics project. There are two sound options, both of which can be turned off for a silent game. By the time you read this, the PC version of *The Crystal Rain Forest* will be available. The Macintosh version, if not yet available, will be ready shortly.

—Terrapin Software

Logo Collaboration

by Mark Steinberger

Logo is a powerful microworld with a low threshold and very high ceiling. Young children easily get involved with moving the turtle around the screen. When it was first introduced on classroom computers more than 10 years ago, Logo was the only way children could put graphics on a computer screen. Its powerful programming environment empowered children to take control of their computers. Logo was by far the most flexible and powerful computer environment accessible to young students.

While Logo's abilities are virtually unlimited, the programming capabilities of young students are not. Frequently, the students' ideas of what they would like to do with their computers far outstrip their ability to translate those ideas into Logo code. A teacher can view this situation as a classic Logo teaching moment, or as an instance where the pedantic mastery of code conflicts with the children's desire to take control of their computers.

As a Logo teacher, I have seen many children in this situation. It is often possible to use the child's idea as a motivator to introduce new Logo concepts. The rationale is that, while working with Logo, the child struggles with powerful ideas that will develop his or her higher order thinking skills. The result is a scaling back of the child's ideas to the level of his or her programming ability. When pursuing this approach, the teacher has made a decision that it is more important for the student to work with Logo as a constructivist microworld than with Logo as a productivity tool. Developing critical-thinking skills becomes more important than developing creativity.

Limiting of Imagination

I would argue that there are times when it is more important to let the child's imagination, rather than Logo code, be the limiting factor in his or her Logo project. I feel there is an analogy to the writing process. When writing, we encourage students to get their ideas on paper without allowing spelling or punctuation to be an impediment. After the idea is committed to paper, it is shared and revised. The last step involves correcting spelling and grammar and preparing the work for publication. The process represents a triumph of creativity over mechanics.

Because of the limitations of currently available technology, mechanics frequently triumph over ideas. It is contradictory for children who use a process approach to writing to have their creativity limited by computer grammar and syntax. I would suspect that many Logo teachers are not comfortable with such limitations to creativity. Their students are taught that, when writing, it is most important to get ideas down on paper. After that, the writing can be edited, the spelling can be corrected, and the grammatical errors can be fixed. Similarly, we encourage children to be creative in other expressive media, such as drawing or drama. Currently available programming languages, including Logo, are painfully literal and unforgiving. Procedures don't run with spelling or syntactical errors. The creative process gets bogged down in arcane computer code.

Solving the Problem

I recently completed a project with a third- and fourth-grade class that offers a solution to this problem. A classroom teacher approached me and asked if I had any computer games that reinforced number-line skills. While I am opposed to CAI software, I tried to use this opportunity to expand my students' appreciation of the power of Logo, their ability to view software in an objective and critical manner, and their understanding of number lines.

I observed the children in the class working with number lines. Using Logo, I wrote a first draft for a game. Originally, the plan was to share the game with the teacher, make the necessary revisions, and then present it to the students as a finished product. I found myself very excited about the game and wanted to share it with the kids immediately. A new approach evolved.

I presented the game to the class as a work in progress. I asked them to try it out, explaining that it might contain bugs. I further suggested that it was not yet very interesting as a game and could be improved. The children spent about half an hour exploring and playing with the game and then wrote notes to me with comments and suggestions for improving it. I revised the game and presented them with *Numberline V. 2.0* the next time I saw them. This led to a discussion of what those "Vees" followed by numbers in commercial programming are all about. Version 2.0 faced the same scrutiny as Version 1.0. Eventually, seven revisions were made before we had a product with which we were satisfied.

Creating this game was a wonderful experience for the children and me. It created a true collaborative situation where we all used our strengths to create a joint product. I know more about programming in Logo than the students do. They know far more about what makes a computer game interesting for children. I did most of the code writing and allowed the children to concentrate on being creative. Because Logo is a modular language, I was able to let some volunteers create some of the subprocedures.

It is important for students to see their teachers as Logo programmers. Using Logo in this manner becomes something that adults do—it's not only a children's activity. They also had the opportunity to see me struggle with the code. The teacher is not a magical person. He really does make mistakes. I encourage the children to go to the flip side of the pages in my game and look at the code. The scene is analogous to a beginning reader looking at a book while an adult is reading to her. She doesn't understand all the words (code), but she recognizes some of them, perhaps can even make sense of a paragraph (subprocedure) or two, and knows that some day she will be a full member of the reading (programming, computer-literate) community.

The activity allowed children, as a group, to engage in software design without being constrained by their ability to write code. This, in fact, is how commercial software is developed. The programmers are not necessarily the "idea" people. The students were asked to explain in some detail how they wanted to see the program changed. I served as a mediator between the children and the computer and allowed them to speak to the computer in natural language, with invented spelling and all. We routinely take dictation from children because we believe it helps them become readers and writers. I took their "Logo dictation."

The technique I have described can be tried with other Logo projects. While not a replacement for traditional Logo learning, this process can be a valuable supplementary activity that allows children to experience some of Logo's more powerful possibilities that they have not yet mastered.

The students spent about 10 hours with this project. After it was completed, no one was interested in playing the game. I have a copy of all seven versions filed away some place. The fun and the learning was where it should have been—in the process. Don't send me a blank disk to get a copy of the game. That misses the point. All you'd get is some CAI software. Go out and create your own collaborative project with your students.

Mark Steinberger is the computer coordinator of Community School District Four in New York City. He has also taught Logo to first through sixth graders at River East Elementary School in New York. Mark is interested in new ways of using and looking at Logo in light of the increasingly powerful open-ended technologies that are becoming available to children and teachers.

Mark Steinberger
Community School District Four
319 East 117 Street
New York, NY 10035
212/860-5946
InterNet: msteinbe@nycenet.edu

Color Displays and Hypermedia

Glen L. Bull, Gina L. Bull, and Chris Appert

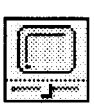
Recently *LogoWriter* seemed to fall prey to a mysterious malady on some Macintoshes at the Children's Rehabilitation Center (CRC) in Charlottesville, Virginia. Sometimes *LogoWriter* would flash [Error -15!] on the screen and quit when children attempted to start the program. The problem was puzzling because the same machines would sometimes work without any problems and at other times could not be coaxed into starting Logo at all. Generally if *LogoWriter* worked at all, it would work throughout an entire session.

Chris Appert, director of the CRC Computer Laboratory, found this experience increasingly frustrating because she could not depend upon specific machines for laboratory sessions with *LogoWriter*. The children using the machines found the experience frustrating as well.

Color Limits

Eventually the source of the malady was identified, and it turned out to have a relatively simple solution. The amount of video memory (video RAM) in some models of color Macintoshes affects the number of colors that can be displayed on the screen. With a limited amount of video memory, only 4, 16, or 256 colors can be displayed.

As more video memory is added to the computer, it becomes possible to display thousands or millions of colors. As the price of video memory decreased, it became possible to upgrade some of the computers in the laboratory. Even with added video memory, everything worked fine as long as the number of colors in the Monitor control panel



was set to 256 colors or less.

However, the moment the number of colors in the Monitor control panel was set to millions of colors (a setting that became possible with added video memory), *LogoWriter* would not run. Consequently *LogoWriter* would sometimes run on a particular machine when the control panel was set to 256 colors but would not run when the control panel was set to millions of colors.

There is a good-news/bad-news report to the aftermath of this finding. The good news is that *LogoWriter* will still run if the colors in the Monitor control panel of a color Macintosh is set to 256 colors. The bad news is that other users from time to time set the number of colors to thousands or millions in order to use other programs, and it is difficult to teach young children to check the Monitor control panel before launching *LogoWriter*.

Chris discussed this problem with a representative from Logo Computer Systems, Inc. (LCSI), and was informed that there are no plans to correct this deficiency in *LogoWriter*. The representative suggested that the Children's Rehabilitation Center discard *LogoWriter* and instead acquire a new site license for a different LCSI product, *MicroWorlds*. *MicroWorlds* is a Logo-like product that also has some hypermedia capabilities. (*MicroWorlds* has been reviewed in past issues of *Logo Exchange*.)

Shortly after *HyperCard* was first introduced, we authored an article with Judi Harris in which we suggested that hypermedia products (such as *HyperCard*) that contain extensible scripting languages might be used in support of Logo-like, learner-centered instructional activities. Such products also have the potential to be used in non-Logo-like ways, for example, in the development of computer-assisted instruction tools. The way in which a tool is used depends upon the wielder. In the aforementioned article, we suggested that broadening the scope of *Logo Exchange* to include discussion of Logo-like uses of hypermedia might be productive.

The Evolution to Color Hypermedia

After some discussion of this issue, both through subsequent articles by others in *Logo Exchange* and through discussion at educational computing conferences, the consensus appeared to be that broadening the focus of *Logo Exchange* to include articles on Logo-like uses of hypermedia was premature. Now it appears that elementary users of *LogoWriter* will face a choice regarding possible transition to a Logo-like product with some hypermedia capabilities.

Rapid change is the one constant of the microcomputing world. It is interesting to note that much of the original success of Logo stemmed from the introduction of turtle graphics as an alternative to the use of Cartesian coordinates to create computing graphics. In an ideal world, LCSI would have the resources to upgrade *LogoWriter* so that it will continue to work with new generations of computers. In the real world, finite resources require choices; and, at least as of the time Chris Appert discussed the matter with an LCSI representative, it appears that LCSI has determined that its most productive course of action will entail the direction of resources into development of *MicroWorlds*.

The title of our column, "Windows on Logo," was chosen to indicate an ongoing interest in new advances in Logo, particularly in a world in which the user interface for all computers will involve a (lowercase) windows-based operating system of some kind. Therefore, it is likely that in future columns we will address experimentation with products that combine Logo and hypermedia, such as *MicroWorlds* and Roger Wagner's *HyperStudio*, which is a hypermedia product that includes Logo as a scripting language. (It is perhaps an ironic coincidence that for many years the title of our column, inspired by Papert's vision, was "Microworlds.")

Charting Logo Colors

In the meantime, Chris Appert's original difficulty caused us to wonder about the current mapping of colors in Logo. As long as there are only six or eight colors, it is possible to assign an unambiguous name to each color: red, green, blue, purple, yellow, orange, and so forth. As the number of colors approaches 256, it becomes difficult to assign unambiguous, easily remembered names to each color. We decided to develop a set of tools that could be used to explore color mapping. These tools were developed for *LogoWriter* but could be adapted to any version of Logo.

The procedure **Spectrum** uses two subprocedures, **Line** and **Over**. The procedure draws a line, moves over one turtle step, changes to the next color, draws another line, and so on. The procedure begins with a specified beginning color and stops at a specified end color.

```
TO Spectrum :Color :EndColor
IF :Color > :EndColor [STOP]
SETC :Color
Line 10 Over 1
Spectrum :Color + 1 :EndColor
END
```

```
TO Over :Amount
PU
RIGHT 90
FORWARD :Amount
PD
LEFT 90
END
```

```
TO Line :Length
FORWARD :Length
BACK :Length
END
```

Initially we asked **Spectrum** to chart all of the colors in *LogoWriter* between 0 and 255 (that is, a total of 256 colors).

```
Spectrum 0 255
```

You may have to run the procedure yourself to see the full effect if you have a color computer. However, even in the shades of gray available in the reproduction in *Logo Exchange*, it may be possible to see that a pattern exists that repeats itself over and over.

The result you will obtain if you have another version of Logo may differ, but part of the charm and excitement of Logo is exploration of unexpected results; perhaps you may find a different pattern.

There appeared to be seven repeated blocks, with an initial block of primary colors at the beginning. Our first guess was that the first block consisted of 16 colors, followed by blocks of 32 colors. We chose 16 and 32 as initial starting points because they are powers of 2. (Powers of 2, such as 64, 128, and 256, are often encountered in binary computers.)

We thought it might be interesting to see what the repeat blocks of colors would look like if they were stacked on top of one another. We modified the **Spectrum** procedure, adding a **NextBlock** subprocedure to reposition the turtle to draw each color block beneath the one before.

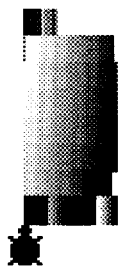
```
TO Spectrum :Color :EndColor
IF :Color > :EndColor [NextBlock
   STOP]
SETC :Color
Line 10
Over 1
Spectrum :Color + 1 :EndColor
END
```

```
TO NextBlock
PU
SETX 0
BACK 10
PD
END
```

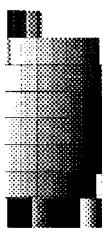
When we used the modified **Spectrum** procedure to draw a series of color blocks, we found it interesting that the repeating blocks of color did not exactly follow sequences of 16 and 32 colors.

```
TO ColorBlocks
CG
PU FD 75 PD
Spectrum 1 13
Spectrum 14 47
Spectrum 48 82
Spectrum 83 117
Spectrum 118 152
Spectrum 153 187
Spectrum 188 220
Spectrum 221 255
END
```

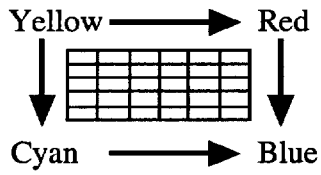
To get the striations (lighter colored lines) in the blocks of color to line up, we found in necessary to plot color sequences of 14, 34, 35, 35, 35, 35, 33, and 35. It would be necessary to ask a programmer at LCSI why the sequence follows this exact pattern, but we can make some intelligent guesses about the overall trend.



If a grid that follows the lines of striations is superimposed over the color blocks, left-to-right and top-to-bottom color trends are discernable. After excluding the blocks of colors at the top and bottom, a 6x6 grid of color blocks can be identified. (The partial block of colors at the top appears to be the *LogoWriter* equivalent of the basic Crayola colors.)



The effect, as you would imagine, is less pronounced in a gray-and-white representation, but you should be able to see the shading trend ... or, better yet, run the **Spectrum** program for your color version of Logo to see the actual effect in living color. We found that 6x6 blocks of *LogoWriter* colors shifted from one shade to another in the following way. Different implementations of color Logo may follow different sequences.



Everyone has probably had the experience of mixing yellow and blue water colors to make green. The color rules in this instance follow the *rules of light* rather than the *rules of pigment*. The three primary colors of light are magenta, green, and cyan (rather than red, blue, and yellow). However, the concept of mixing two colors to achieve a third still applies.

Millions of Colors

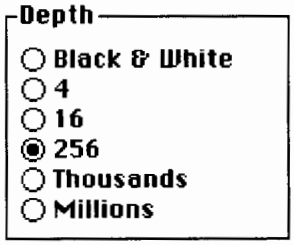
Each of the three color components can be represented by a byte of memory in video displays with 24-bit color. A total of 256 different values (0 through 255) can be represented within each byte (i.e., 256 is 2 to the eighth power). If each of the primary colors is systematically varied from its least to its greatest value, eight colors are generated. Shades between these colors can be achived by selecting intermediate values between 0 and 255. Because there are three positions with a total of 256 values in each position, the total number of possible colors is 16,777,216!

	Magenta	Green	Cyan
Black	0	0	0
Cyan	0	0	255
Green	0	255	0
Turquoise	0	255	255
Magenta	255	0	0
Red	255	0	255
Yellow	255	255	0
White	255	255	255

One disadvantage of all those colors is the amount of file space required to record them. If three bytes (24 bits) are required to record each dot (sometimes referred to as a *picture element*, or a *pixel*) in a picture, file size can quickly balloon. For example, a picture that is 8 dots across by 8 pixels down would require 64 bits (or 8 bytes) in black and white. The same picture would require 192 bytes in 24-bit color. Thus a modest picture

of 100 x 100 pixels could quickly balloon from 1,250 bytes to 30,000 bytes. If you have recently acquired a color paint program, this may explain why your files have grown so quickly as they are converted from black-and-white to color.

Some paint programs, such as *ClarisWorks*, allow the user to specify the number of bits that are used to represent the color of each pixel. For example, a one-page paint document would generate a 111K (kilobyte) file in black and white, an 888K file with 8-bit color (256 mode), and a 3,553K file in 24-bit color ("Millions" mode).



The current version of Macintosh *LogoWriter* is equipped to handle 8-bit color (with 256 different shades), and evidently will remain forever frozen in time in this state. We wish the program would die a bit more gracefully when it encounters a Macintosh set to display more than 256 colors. Perhaps the program could display an error message such as:

```
"The monitor control panel must be
   set to
256 or fewer colors to use LogoWriter."
```

rather than flashing the Macintosh "Error -15!" message as it dies. However, we hope that the foregoing provides an explanation that makes the underlying factors governing color representation less mysterious.

Glen Bull is an associate professor in the instructional technology program of the Curry School of Education at the University of Virginia. Gina Bull is a computer systems engineer in the information technology and communication organization at the University of Virginia. By day she works in a UNIX environment, by night in a Logo environment. Chris Appert is director of educational computing at the Kluge Children's Rehabilitation Center.

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

Using Computers in Summer Fun in the City

by Eileen Boyle Young

> An important theme of Logo research is the idea that individuals can—and in some cases must—follow very different learning paths.
>
> —Seymour Papert

Summer Fun in the City, a pilot inclusion project of the Council for Retarded Citizens of Jefferson County, Kentucky, is a two-week, all-day recreational educational program for students in special and regular education—children ages 7 to 14 with diverse abilities and disabilities.[1] This article discusses the use of Logo and computers as part of the program

This was a learning experience for both teachers and children; the teachers who worked with the campers using computers were at the same time enrolled in a three-credit, graduate-level education workshop at Spalding University in Louisville, Kentucky. Because this program activity involved both campers and their teachers in the use of computers, the working hypothesis was threefold:

- Using *LogoWriter* as their tool, all children can benefit from interaction with the computer.
- Working with learners in a Logo environment, teachers can increase their own understanding of how children think.
- As learning and social interaction tools, computers and *LogoWriter* can be integrated into any curriculum (even a recreational one).

The program created an opportunity for success for learners, especially for those who do not often experience success. The Logo environment provided participants with structured yet open motivation, a means of verbal and visual communication, a framework for writing and drawing, and a medium for playing and exploring.

As a result of their experience in this introductory-level project, campers should be able to use computers as tools to think, communicate, and learn in their own individual ways; and teachers should be able to use computers as tools to think, communicate, and learn with all students in the students' own individual ways.

Activities for Teachers

Each day teachers spent two hours with their instructor in a round-table setting during which they reviewed the preceding day's progress with students and prepared the next day's agenda. This "project" approach integrated computer work with the rest of the day's planned activities. In the process, the teachers learned how to work with students using computers as an integral part of the curriculum and how to use *LogoWriter* and Macintosh computers as learning tools.

Teachers spent additional time outside of class researching the topic "What the Research Shows About Logo and Special Needs Students." They read Sylvia Weir's (1987) *Cultivating Minds* and Seymour Papert's (1980) *Mindstorms: Children, Computers, and Powerful Ideas*, and thought about how children think.

In addition to these resources, teachers also had access to past issues of *Logo Exchange* (from 1983 to 1992) and *The Computing Teacher* (from 1982 to 1992) for ideas on teaching and learning.

The Research and Development Lab of the Council for Retarded Citizens, housed at Spalding University, provided the site where the computer-use portion of Summer Fun in the City took place. The lab had six computers (five Mac SEs and one Mac SE/30 linked to a LaserWriter II printer).

Activities for Campers

Before they came to work in the lab, students were given riddle questions, and time to think, discuss, and formulate answers. The riddle answers were mounted on the walls of the lab for students to find and match on their first visit to the lab. This activity served the dual functions of ice-breaker and assessment tool. The teachers were able to gauge students' levels of cognition and to adjust the first day's activities accordingly. These same or other riddles were used in later computer work; students used them as patterns in dialogue writing.

General Operating Procedures

Students were divided into small groups of three (two with cognitive impairments and one with no cognitive impairment) or four (two and two), based on age. Each group was assigned to a computer and a teacher. Each teacher followed the same lesson plan, making allowance for individual learner differences within the small group of students. Each day's activities included individual and small-group writing and drawing and an opportunity to share with the large group. Students began their Summer Fun in the City day with two hours of a computer-use activity in the lab.

Usually, students worked with teacher-made *LogoWriter* procedures. The teachers learned to create the procedures as part of their coursework. Working with students, the teachers used *LogoWriter's* word-processing capabilities for regular text; they also used the following Logo primitives: **label, shape, setsh, fill, shade, stamp, ht, st, slowturtle,** and **fastturtle.** They learned to clear the text, graphics, and Command Center. Some students worked with classic turtle graphics, others with sound. All students learned the concepts of getting new pages, naming pages, saving pages, and printing the screen.

Students were encouraged to use the printer to produce paper copies of their writings and drawings and to use the copy machine to make bulletin board exhibits. Other resources were used at the worktable to further enhance their off-computer activities. These resources included construction paper, plain paper, crayons, markers, hole punches, scissors, yarn, and clear tape. Selected children's books, including Bennett Cerf's *Riddle Books,* Shel Silverstein's *A Light in the Attic* and *Where the Sidewalk Ends,* Eric Carle's *Animals,* and Jack Perlusky's *Something Big Has Been Here,* were also made available to them.

Click, Point, and Drag

A Quick Tour of the Macintosh (the pointing, clicking, and dragging activities only) allowed students to practice these three necessary skills until they mastered them. *KidPix,* from Borderbund, reinforced mastery of pointing, clicking, and dragging. Even the more cognitively impaired students recognized the *KidPix* icon on the hard drive and understood how to open and use *KidPix.*

Ability to use the mouse was the only prerequisite skill asked of the students for successful participation in the lab Logo activities.

Integrating Computer Activities Into the Rest of the Day

Thank You Notes

Each day, students used *LogoWriter* to produce notes to send to the previous day's field trip host. In the process, small groups brainstormed about what each member enjoyed most and decided which items to include in their note. They used shapes and stamps to decorate their text. Using the same skills, the students created invitations for parents to attend the closing ceremony of the program.

Instant Logo, *LogoWriter,* and City Observations Scenes

Using teacher-made procedures, students explored with Logo primitives such as **forward, back, right, left, pu, pd, ct, cg,** and **stop.** This introductory exercise could be extended to focus on directionality using mazes.

Students used *LogoWriter* in the immediate mode to draw and label scenes from their experiences, such as riding TARC (the city bus system); visiting the Museum of Natural History and Science, the Kentucky Derby Museum, and Speed Art Museum; swimming; playing ball in the park; and eating lunch on the Ohio River Belvedere.

Working With Basic Geometric Shapes

Using the following set of teacher-made procedures—

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

to hexagon :size
repeat 6 [forward :size right 60]
end

to rhombus :size
repeat 2 [ forward :size right 60
   forward :size right 120]
end

to trapezoid :size
repeat 2 [forward :size right 60]
forward :size right 120 forward
   :size * 2 right 120
end

to triangle :size
repeat 3 [forward :size right 120]
end
to sun
repeat 6 [rhombus 30 right 60
   forward 30]
end

to star
sun
right 60 repeat 6 [forward 30 left
   60 rhombus 30 right 120]
end

to optical.illusion
seth 0
pu
setpos [-195 0]
pd
label [SUN]
sun
wait 200
seth 0
pu
setpos [100 0]
pd
label [STAR]
star
wait 200
end
```

—students worked with a range of geometric figures to form shapes and designs. This computer activity related to their visit to the craft shop,where they decorated tee-shirts with geometric designs. Also, using the fill and shade Logo primitives, they were able to simulate their tie-dying experience. This activity could be extended to include traditional Logo activities, such as making quilts or working with tangrams and pentominoes.

ID Shields

Students and teachers used teacher-made procedures to create individual name plates. They used available Logo stamps and the Logo primitives **shade** and **label** to personalize their shields. Students used the worktable resources to add finishing touches. They cut the shields out and wore them around their necks or used double-sided tape to attach them to their shirts.

Planetarium Sky

Using the following set of **starinsky** procedures—

```
to starinsky :size
setsh 16
repeat 5 [forward :size right 144]
end

to dipper
setsh 16
pu
forward 75
right 90
forward 100
pd
stamp
right 135
forward 50
stamp
right 25
forward 55
stamp
left 20
forward 60
stamp
right 30
forward 65
stamp
left 100
forward 45
stamp
left 85
forward 40
stamp
left 65
forward 40
ht
end
```

—students created a big dipper procedure and other starry skyscapes. Many of them also created a special ID shield of stars to wear on their field trip to the Rauch Planetarium.

Adventure Stories

Using teacher-made procedures, students and teachers created animated stories, such as "My Trip to the Swimming Pool." This activity could be extended to include other types of Logo adventures, such as "pick your own ending." Also, using music could be added for another dimension.

The Louisville Zoo

A trip to the Louisville Zoo gave the students the opportunity to learn about animals. Their task was to observe one animal so that they could use *LogoWriter* to describe that animal. The other students would then guess what animal they had described. Also, they wrote their response to the lead-in "If I were a(n) ... ." They shared what they knew about where the animal lived, its size, eating habits, and so forth. Lab use of a videodisc entitled *The National Zoo* was shown before the field trip to heighten the students' anticipation, and it was shown again on the day after the trip to refresh their memory of the animals they had seen.

Evaluation

To provide a summative evaluation, students were asked at the last session to complete the following tasks:

- Use *LogoWriter* to respond to each of these three topics—"What I liked most about Summer Fun in the City," "What I liked least about Summer Fun in the City," and "What I think about using the computer."
- Use *LogoWriter* to describe yourself in words.
- Use *KidPix* to draw a picture of yourself.
- Use the laser printer make hard copy of your work and then use the copy machine to make bulletin board copies.

The teachers completed formative evaluations daily in the round-table sessions with their instructor.

Recommendations

Next summer, we will make the following changes in our lessons:

- We will make sure that students know how to observe and how to mentally note their observations. Our experience indicated that students cannot make connections unless their skills of observation and memory are fine-tuned.

- We will adapt what teachers know about the writing process for use with students with diverse abilities and disabilities in an inclusion setting.
- We will reduce lab time to 90-minute sessions to maintain better focus.
- We will make computers available during other times of the day for a more integrated experience.
- We will create an ongoing workshop during the academic year to prepare teachers to work with students in this setting.

Project Replication Notes

Any form of Logo can be used. With appropriate modification, the procedures included in this article will work with any version of Logo. The use of the mouse and the Macintosh will allow teachers and students to begin as soon as they master pointing, clicking, and dragging. No special key combinations are needed to operate in this environment. *LogoWriter* used on the Macintosh gives users immediate on-screen access to help with primitives.

Footnote

1. A Summer Fun in the City Training Module is available from Council for Retarded Citizens of Jefferson County, 1146 S. Third, Louisville, KY 40203. The module includes a program description and information on the planning process, daily activities, learner and teacher outcomes, and the evaluation process.

References

Weir, S. (1987). *Cultivating Minds*. Harper & Row.
Paper, S. (1980). *Mindstorms*. Basic Books.

Eileen Boyle Young
Professor of Education
School of Education
Spalding University
851 South Fourth Avenue
Louisville, KY
502/585-9911

Teaching With Logo Means Being on the Cutting Edge

by A. J. (Sandy) Dawson

The first MathWorlds column appeared in September, 1986. Now, almost eight years later, it seems appropriate to move on, to pursue other developing interests. So this shall be my last column. One last time I am faced with the decision as to what to write about. I was mulling this over in the back of my mind as I sat down last night to read the latest issue (February, 1994) of *Kappan*. It contained three articles about the reform of mathematics teaching (see the References—one article was written by Michael Battista, a name very familiar in computer circles, especially Logo circles). The theme running through these articles was the essential role teachers play in making any modification in the ways children come to know mathematics. This will not be a new idea to those who have been working with Logo for many years.

In the years after Papert's *Mindstorms* appeared, many people alleged that he thought that children would magically learn to work with Logo without the assistance of their teachers, but he clarified that impression early on. Those working in classrooms with children (most likely all exploring Logo together because it was so new) quickly understood the critical aspects of a teacher's work. Battista (1994) reaffirms the centrality of the teacher's work if any major change is to be made to the manner in which mathematics is taught in schools. He also notes that if teachers do not believe in the change, their work will hinder rather than assist change. He writes:

> these teachers' beliefs not only caused them to implement an inappropriate curriculum but also blocked their understanding and acceptance of the philosophy of the reform movement, thereby precluding the possibility of substantive curricular change. (p. 467)

How often, in the early years, did one see Logo being taught in a rote, drill-and-practice, step-by-step manner? But even in those cases where teachers are sympathetic to the changes being suggested, they often do not have the necessary knowledge and skills to actualize their beliefs. And teachers of Logo, with nothing but the best of intentions, struggled when the kids got stuck, or they hunted for engaging and challenging tasks for the children to explore. From what Battista says, teachers of mathematics across the nation are now experiencing similar frustrations and anxieties as they attempt to reform school mathematics. Perhaps some of you mature, experienced Logo teachers could provide leadership for your troubled colleagues

Research in the past decade, both in computer and mathematics education, points dramatically to the fact that the situation in which learners meet and grapple with concepts plays a significant role in determining the meaning that those learners will attach to new ideas. Hoyles and Noss (1992), reporting on a wealth of research on Logo, came to the conclusion that

> the setting ... is central to understanding how children use the medium to express their ideas and how those ideas are themselves structured by the medium. (p. 465)

They further note that

> the best we can hope for is to provide situations and environments in which fragments of mathematical discourse make sense and are useful in solving the problems that structure, and are structured by, the setting, that become for pupils *situated abstractions*. (p. 463)

In her *Kappan* article, Marilyn Burns (1994) urges "mathematics teachers to stop teaching standard algorithms and to start having children invent their own methods" (p. 471). Hoyles and Noss, as well as Burns, seem to be focusing the teaching and learning of mathematics in the same direction, namely, in a direction toward having children experience situations that are potentially rich with mathematics and letting them explore, all the time being aware of and sensitive to the meanings that the children are putting onto the situations. Let the kids invent, they seem to be arguing. Teachers long experienced in using Logo and Logo-like environments would respond with joy to such suggestions. Moreover, the advances made in computer and calculator technology since Logo first appeared, or even since I started writing this column, are truly amazing. These advances bring with them all sorts of questions about the appropriateness of the current mathematics curriculum, particularly the great emphasis placed on learning and performing algorithms

that the technology can so readily and easily handle. Difficult and fundamental questions are being asked of the mathematics curriculum and of the teachers of that curriculum.

Nowhere are the questions as difficult as they are in dealing with the cultural and experiential diversity now found among children in North American classrooms. For example, in my home area, Vancouver, British Columbia, it is not unusual in a class of 35 elementary school children to have as many as 20 different native languages. This diversity of languages among children brings with it a richness of experience and meaning most teachers have difficulty comprehending, let alone responding to in a sensitive and caring fashion. Consequently, the social and political context of the children has to be taken into account, a new challenge for teachers of mathematics. As Tate (1994) comments:

> Until recently, embedding mathematics pedagogy within social and political contexts was not a serious consideration... [but] mathematics is embedded within the culture of their [the kids'] daily struggles. (p. 482)

So the challenges persist. Those of us experienced in the introduction of Logo to the schools of North America and Europe recognize that the issues raised by Battista, Burns, Tate, Hoyles, and Noss were foreshadowed by the work done by Logo teachers during the past decade and a half. Perhaps the experience we all gained as explorers (with children) of Logo environments can be used to assist with the transition of the mathematics curriculum now being undertaken by teachers and supported by organizations such as the NCTM. Perhaps the strategies and approaches developed for use in Logo environments can be exemplars for the teachers who are struggling with how to teach this new mathematics. Perhaps teachers of Logo can be on the leading edge of change in the creation of environments for the learning of mathematics.

If that were to happen, then I can happily pass over the writing of this column to those who have more recent experience of the classroom than do I, and who can therefore provide leadership for the remainder of the 20th century.

The last eight years have been great, and I thank Tom Lough (who first convinced me to take on this task) and Sharon Yoder for all their support and assistance over the years—and particularly to Sharon for understanding when my columns were not quite ready on time, or were too long, or whatever. Many thanks, Tom and Sharon. It's been a slice!

References

Battista, M. T. (1994). Teacher beliefs and the reform movement in mathematics education. *Kappan, 75*(6), 462-470.

Burns, M. (1994). Arithmetic: The last holdout. *Kappan, 75*(6), 471-476.

Hoyles, C., & Noss, R. (Eds.). (1992). *Learning mathematics and Logo.* Cambridge, MA: MIT Press.

Tate, W. F. (1994). Race, retrenchment, and the reform of school mathematics. *Kappan, 75*(6), 477-485.

Sandy Dawson is an associate professor of mathematics education at Simon Fraser University and is director of that institution's teacher education program. His developing research interests center on the areas of embodied learning and complex adaptive theory.

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

Viewing the RIGHT Way: A New ANGLE on Quilting With Logo

by Jody McQuillan

I am new to the magic of Logo but not to graphic design. I see a multiplicity of uses for this powerful tutee in the graphics area alone. For three years I have been teaching a basic design lesson to fourth-grade students, using colored, triangular-shaped pieces of paper cut from squares to create quilt designs. This activity usually follows a unit on Nebraska history. Although quilts are created by using a number of geometric shapes, I have limited these quilt designs to what we call "half squares." It has been a successful design lesson, and my students have enjoyed it.

One of the most successful ways in which we can teach children to draw is to teach them to see. A way to promote "seeing" is to ask children to trace an object with their eyes, allowing them to really see what is there, both with their minds and with their eyes. This activity requires referencing, concentration, and practice.

I have become very excited about Logo. Because it was new to me, I began to seek information on which to build my own knowledge base. As I looked through the readings about Logo, I found several variations of Logo quilting ideas. As I thought about the use of Logo, I realized how effectively the quilting lesson I already taught could be implemented through this platform. I could enrich, expand, and give new dimension to the basic visual design lesson. I realized how naturally I could incorporate math and visual imagery, as well as establish a quilting community through the use of small-group work.

I planned to use the initial paper pieces first, followed by re-creating the pieces on the computer screen with Logo. In doing this, the students would have to use their observation skills to place sides and angles to replicate the actual design. Of course, students would also need a basic background in Logo.

The Half-Square Lesson

Overview

1. Give the students a basic understanding of what a heritage quilt from the Plains area may have looked like. Talk about where artists ideas come from and why these particular people created the designs they did. Show examples, pointing out shape, line, color, and contrast.
2. Cut 2" x 2" colored-paper squares diagonally in half to form isosceles triangles. Place these half squares on 9" x 9" white or cream-colored paper and have students lay out their possible designs.
3. When the students are satisfied with their designs, place them in groups of four and have them use Logo to re-create their paper designs on the computer screen.

- The main objective is to incorporate contrast, color, and geometry into a quilt design.
- The main rationale for using Logo in this lesson is to encourage and exercise the students "right-brain" seeing by replicating their design to another form.

Objectives

Through this lesson, the student can gain:

1. The ability to recognize basic design elements in their own work and in the work of others.
2. The ability to use an advanced observation technique to copy their own designs into another form.
3. The ability to use color and contrast theory to create a pleasing design.

There are also several hidden curricular objectives that may be achieved by this lesson. The students can gain:

4. An understanding of community-building similar to actual quilting bees.
5. A deeper understanding of geometric forms and the ways they fit together mathematically.
6. The ability to exhibit a command of the Logo language while writing procedures and subprocedures.
7. An enhanced knowledge of Nebraska's pioneers and the things that may have motivated them to create quilt patterns.

Half-Square Quilt Designs Materials

- white construction-paper squares (9 inches)
- multicolored squares (2 inches) cut diagonally in half. (The students decide on how big to make their half squares.)
- glue
- rulers
- *LogoWriter*

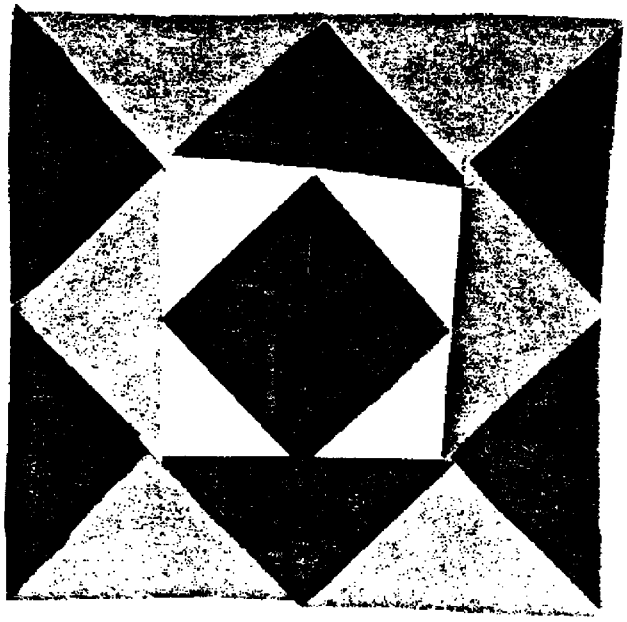
- *LogoWriter* data disks
- *Color Print Shop*
- sample quilt designs

Phase 1

The initial portion of this lesson teaches students about quilts and allows them to see and discuss actual traditional quilts and quilt patterns. When they look at the quilts, it is important to point out angles, negative space, symmetrical balance, and contrast of colors for emphasis.

Phase 2

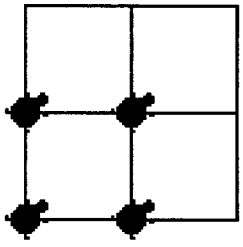
This portion of this lesson asks the students to design a quilt pattern using at least eight paper triangles. It is very important for students to experiment with paper pieces through this design process. Letting them have the glue too soon can cause them to miss some very special patterns. It is important to stress that traditional quilt designs like the ones they have seen are nice, but new quilt designs, no matter how simple, can be really special.



Phase 3

This phase consists of transferring each student's quilt design into Logo so that it can be drawn on the screen. Place students in groups of four to encourage cooperative learning and problem solving. You then can have conferences with the groups to discuss ways they might all be able to divide up the screen and come up with a size for their overall square and for their individual workspace squares. If the space is divided into four sections, each of *LogoWriter's* turtles has a quadrant. Next comes the need for alerting each turtle to come out and play and for placing each turtle in the same starting place on its own quadrant. With the teacher's help, the students might decide on something like this:

- 85 turtle steps by 85 turtles steps for the individual squares
- 170 turtle steps by 170 turtle steps for the overall square
- tell all
- show turtle
- pen up
- tell 0 set pos [0 0]
- tell 1 set pos [0 -85]
- tell 2 set pos [-85 -85]
- tell 3 set pos [-85 0]
- seth 45 to each



Phase 4

The next step is to have a conference with each small group of students to help them create a procedure that will help them pull their work together. By doing this, the teacher can start the group and model, review, and instruct them in the use of Logo tools. These tools may include flipping pages, writing basic procedures, using the Up and Down keys to copy what is in the Command Center, using SETPOS and SETH, hiding the turtle and other basic primitives, and learning how important it is to start and end the procedure with the turtle in the same place.

Here is an example of some possible teacher procedures:

```
TO STARTALL
FRAME
START
END

TO FRAME
RG
BACK 85
LEFT 90
FORWARD 85
RIGHT 90
REPEAT 3 [FORWARD 170 RIGHT 90]
FORWARD 85
RIGHT 90
FORWARD 170
BACK 85
```

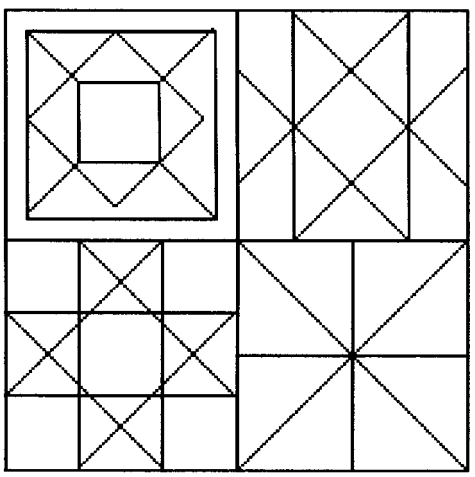
```
LEFT 90
FORWARD 85
BACK 170
PU
END

TO START
TELL ALL
ST
PU
TELL 0
SETPOS [0 0]
TELL 1
SETPOS [0 -85]
TELL 2
SETPOS [-85 -85]
TELL 3
SETPOS [-85 0]
TELL ALL
SETH 45
PD
END
```

Phase 5

The students work on their designs in small groups throughout the week. Make sure to give the students ample time to complete the assignment.

Phase 6

After the designs are completed and put into procedure form, the teacher should have conferences with the students and help them link their design together with the framework of the start procedure created in the first conference.



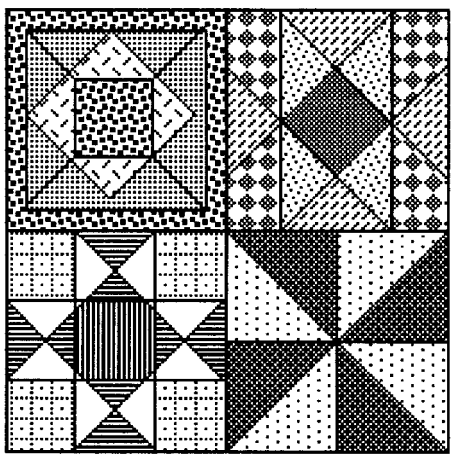
Phase 7. And Now for Some Color!

If students have not had prior experience with "fill," a demonstration may be needed. Depending on the ability of the student and/or group, the students can

1. Create procedures to fill designs. Advanced students can do this by using pen up/down, fill, and other commands in the procedure page.
2. Fill by using the Command Center and saving. Intermediate kids can do this in a manner similar to that in item 1, except that the filled image can be saved to disk without procedures.
3. Fill using the child's design yourself by saving on data disk or by using *Color Print Shop* and selecting Screen Magic to add the color. The less advanced students may need to have conferences with you and may need to have you fill in the color and spaces on their quilts using *LogoWriter* or *Color Print Shop.*

Phase 8

The final task of printing in color belongs to the teacher. (Of course each group may want to watch as their quilt-square images roll out in hard-copy form.) The designs may be patch-worked and mounted on a bulletin board just like a real quilt gets pieced together. To do this, save the color quilt designs to a data disk in PICT format and then read them from *Color Print Shop* to be printed in color with a color ribbon.



This lesson would naturally follow a social studies unit on pioneers or could be used in conjunction with such a unit. It also could be used as a culminating activity after reading books about how early Americans lived. Two examples of these types of books are *Prairie Songs* by Pam Conrad and *Sara Plain and Tall* by Patricia McLachlan. By weaving the theme throughout the total curriculum, you can reinforce all subject areas.

Jody McQuillan
6103 Hamilton
Omaha, NE 68132

King Arthur Had a Daughter

by Robert Macdonald

Not too long ago I attended the yearly Michigan Association for Computer Users in Learning (MACUL) Conference in Detroit. At a SigLogo luncheon on the final day I had an interesting conversation with Rudy Neufeld, a Canadian writer who has used Logo applications in presenting math concepts to students. I left the luncheon remembering one of his statements: "The positive atmosphere engendered by open Logo experimentation should be utilized by teachers in the area of mathematics." This tied in very clearly with some interesting presentations given by the Michigan SigLogo directors, Tim Meno and Mark Lindsay. Both of these gentlemen teach in inner-city environments, where alternative forms of teaching are essential to capturing the interests of students. They obviously are highly successful with what they do. I left the conference feeling that perhaps American public education is not nearly as bleak as critics have suggested. While driving back to my island home, I mused on what I had observed.

When I arrived home, I opened a letter that had arrived from Ihor Charischak, the president of Clime. In the letter, Ihor had detailed some interesting extensions that he and others had undertaken of a problem I had presented in *Clime Microworlds* several years ago (Macdonald, 1990). The microworld was based on an idea I had come across more than a decade ago in Marilyn Burns's (1982) *Math for Smarty Pants*. Entitled "King Arthur's Problem," it related the following problem:

> King Arthur had a beautifully gifted daughter Glissanda, who desired to marry not only a handsome and brave knight but also one who was as interested in mathematical puzzles as she was. A thousand years ago, this posed a problem for King Arthur. Today, Glissanda could have helped resolve her own dilemma by enrolling in math classes. But in those days, the knights of the Round Table were outdoor types: they enjoyed hunting and slaying dragons, helping damsels in distress, and combating evil. They spent little time on such things as math pursuits. How would King Arthur and his daughter provide a test that would not only prove the heroism of the selected knight but also his mathematical prowess?
>
> After much thought, Glissanda came up with the test. By knowing the number of knights who would be seated at the Round Table, the mathematically gifted knight would know which chair to occupy, or it would cost him his head. Suppose, Glissanda explained to her father, 24 knights appeared on the day of the test and seated themselves in chairs numbered in order from 1 through 24. To select the bridegroom, her father would draw his sword and point to the knight in the first chair and declare, "You Live." To the knight in the second chair, he would declare, "You Die," and chop off his head. To the third knight, he would say, "You Live"; to the fourth, "You Die"; and so on, going around the Round Table so that there would be only one surviving knight.
>
> Naturally, King Arthur was horrified by his daughter's suggestion. He just couldn't bring himself to kill off all but one of his knights. His chances of ever again drawing knights to his Round Table would appear slim. But Glissanda quickly consoled her father. He wouldn't really kill his knights off; they just had to believe he would. How else would she be able to prove that they were brave? It was a mathematical game that only an intelligent and brave knight would be able to solve. The winning knight would have to know where to sit among all of the knights who participated.

An Approach to the Problem

Alas, Ms. Burns does not supply a solution for the problem. But that only adds to the intrigue. Her article was not only of interest to me but also to my students, who found it intriguing as well. It was obviously a problem that demanded the collection of data. Therefore, I considered how the class might go about solving it. Obviously the casting out of knights seemed the most logical approach. Suppose that there were 11 knights seated around the Round Table. Which would be the safe chair? Write down the 11 numbers on a sheet of paper:

1 2 3 4 5 6 7 8 9 10 11

Now cast out knights—Stay, banish (cross out), stay, banish (cross out), and so forth. In the first trip around the table, all the even numbers are cast out, leaving the knights numbered 1 3 5 7 9 11.

The second round casts out the knights numbered 1, 5, and 9, leaving knights 3, 7, and 11.

The third and final round casts out 3 and 11, leaving 7 as the safe chair of the surviving knight (Charischak, 1993).

At the end of this article are two sheets, a worksheet and a data sheet, that will prove helpful in gathering data. Your students will have much more fun if you have them work in groups of two. Casting out knights with paper and pencil aided by a friend always makes the task more enjoyable.

Powers of 2

In gathering data, you will find that powers of 2 enter prominently into the solution to the encounter. It might be helpful to present this patterning to a class prior to their collection of data. It is a simple task to provide such patterning on a calculator. Enter 2 x 1, and then tap the equal sign. Keep tapping the equal sign. The following patterning appears:

2 4 8 16 32 64 128 256 512 1024 2048 etc.

In our technological age this patterning should be a vital part of any upper elementary student's patterning knowledge. Powers of 2 are fundamental to comprehending base 2—the base around which the mathematics of the computer is organized. A base 2 grid provides a meaningful organization of the only two digits (0, 1) used in that base. In the same sense, a grid organized around groups of 10s provides the necessary structure and meaning in base 10.

A Program for Producing Powers of 2

The following *LogoWriter* program will produce the patterning of powers of 2. It will be necessary to enter an input of 1 after the command of this program to get an output of the powers of 2. A **stop** command has been entered in the program. However, it might be interesting to delete the **stop** command to see how far the computer will carry the calculations. On the Apple IIe the calculations will go up to 549,755,813,888. It will then stop with the following remark in the Command Center

```
Number too big in powers.of.two
```

The output is much greater on the Macintosh because it resorts to scientific notation. This would be a good exercise for examining large numbers. Therefore, enter

```
powers.of.two 1

to powers.of.two :number
clearpage
print :number
continue :number
end

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

to continue :number
if :number = 65536 [stop]
print (2 * :number)
make "number :number * 2
continue :number
end
```

Base 2

After introducing the powers of 2, why not do a little work with base 2? The powers of 2 will provide a grid for outlining another system of counting. This can also be used as a form of reinforcement for providing skills for solving our problem. See the base 2 grids at the end of the article. Students might like to complete the grids in base 2 for the numbers 1 through 64 in base 10. Students should use only the digits 0 and 1, and remember that in base 10, the following 10 digits are used:

0 1 2 3 4 5 6 7 8 9

which we place under a grid we label

1 10 100 1000 etc.

Under the 1 in the Base 2 Grid, only a 0 or 1 may be placed; under 2 in the grid, only a 0 or 1 may be placed; and so on. Thus:

$1_{10} = 1_2$
$2_{10} = 10_2$
$3_{10} = 11_2$
$4_{10} = 100_2$
$5_{10} = 101_2$

In base 2, we use only two digits, 0 and 1; therefore, we get some extraordinarily large numbers that are more briefly expressed in base 10. After filling in the Base 2 Grid, we are ready to apply the collected data for use in a binary game.

A Binary Game

I came across the idea for this binary game some 20 years ago (Barr, 1971). A smaller database is used in this explanation. However, you may want to extend it up to any number you wish. You merely have to work out the equivalent numbers in base 2. Suppose we wish to have a number-guessing game for numbers 1 to 31. The student is to look at each card drawn from the special set used for the game and state if the guessed number is found on it. The solution is to add up the first number

in the upper left corner of the card on which the number is found. The sum of those upper left numbers will provide the number selected by the student and is to be guessed. Now let's construct our cards from the data listed in the Base 2 Grid.

On Card 1, place all of the base 10 numbers that have a digit 1 under the column 1 of base 2.

Card 1
1 3 5 7 9 11 13 15
17 19 21 23 25 27 29 31

On Card 2, place all of the base 10 numbers that have a digit 1 under the column 2 of base 2.

Card 2
2 3 6 7 10 11 14 15
18 19 22 23 26 27 30 31

On Card 3, place all of the base 10 numbers that have a digit 1 under the column 4 of base 2.

Card 3
4 5 6 7 12 13 14 15 20
21 22 23 28 29 30 31

On Card 4, place all of the base 10 numbers that have a digit 1 under the column 8 of base 2.

Card 4
8 9 10 11 12 13 14 15
24 25 26 27 28 29 30 31

And, finally, on Card 5, place all of the base 10 numbers that have a digit 1 under the column 16 of base 2.

Card 5
16 17 18 19 20 21 22 23 24
25 26 27 28 29 30 31

Now suppose you secretly selected the number to be guessed as 23. Your selector would find it on Card 1, which he puts aside; on Card 2, which he puts aside; on Card 3, which he puts aside; not on Card 4; and on Card 5, which he puts aside. The selector thus adds the numbers in the upper left corner on the selected cards: 1 + 2 + 4 + 16 = 23.

If you go beyond 31 (I like to go up through 63), you will need to add another card using the 32 column of base 2. Have the students prepare their own set of cards from their data sheets and start playing the game. I've found that children are more alert than adults in things of this sort. Many students find the patterning on the cards to be worth investigating.

The Solution

When students have completed the data sheet provided at the end of this article, a number of things become more and more evident. The powers of 2 enter the solution. Doubling is also an important element in arriving at a solution. In brief, double the number of knights, subtract the largest power of 2 that you can while maintaining a positive number, and then add 1. Let's experiment. Suppose that 11 knights are seated around the Round Table. Multiply 11 by 2, which equals 22. The largest power of 2 that you can subtract from 22 and still retain a positive number is 16. You now have 6 as a number. Simply add 1 to 6, and you have 7 as the knight who will be seated in the safe chair.

Let's try 24 knights:

24 x 2 = 48 (Double the number of knights)
48 - 32 = 16 (Subtract the largest power of 2)
16 + 1 = 17 (Add one)

Having introduced students to binary notation, it seems only logical to solve our problem in binary too. Consequently, let's try a solution in binary notation:

For 24_{10}

11000_2
$\underline{\times\ 10_2}$ (Double the number)
110000
$\underline{-100000}$ (Subtract the largest power of 2)
10000
$\underline{+\ 1}$ (Add one)
$1000_{12} = 17_{10}$

Several months ago, Ihor Charischak sent a brief message in which he very cleverly demonstrated a quick solution to the knight problem if the student employs binary notation. Here is a variation of the problem. Remember that the number of the chair in which the knight is seated around the Round Table is a vital bit of information. Let's write some numbers in base 10:

Number of Chair	Winning Chair
5	3
10	5
13	11
49	35

What if we were to write these numbers in binary notation? The preceding table is translated into:

1st Column Number of Chair	2nd Column Winning Chair
101	11
1010	101
1101	1011
110001	100011

Now if you compare the first column with the second, how could you arrive at an answer in the second column by making a slight adjustment in the first? It is here that Ihor shines. He suggests that you

merely transfer the last digit to the left (in the first column) to a position to the right of the last digit to the right (also in the first column). Hence 101 (in the first column) becomes 011. This is the solution for the winning chair (in the second column). 011 = 11. 1010 (in the first column) becomes 0101. 0101 = 101. Thus, if you are willing to translate the chair number into binary notation and make a slight switch, you can get an immediate answer. There is no need for casting out knights or making use of mathematical formulae. We can make use of Ihor's clever slight of hand. He again demonstrates his adroitness.

A Logo Program for Knights

One of the nice features of a *LogoWriter* program is that it is possible to write commentary on the flip side of a page, the same side of the page that contains the program. I frequently have wanted to enclose material within a program that I wanted students to have the opportunity to look over whenever they wished. The following program included is typical. In case a student wishes to reread Ms. Burns's article, I have provided the source. For those who want only a summary, I have included one here.

You will find that the program is user friendly. The program limits investigation from 1 through 60 knights. If you are interested in making a broader study, see the briefer program in the next section of this article.

```
**************************************************************
   The idea for this program was
   engendered by Marilyn Burns in her
   book Math for Smarty Pants (Boston:
   Little, Brown and Co., 1982, pp. 75-
   79). The article "King Arthur's
   Problem" should be read by the group
   of students undertaking this
   microworld.
**************************************************************
   A short summary:
   King Arthur had a daughter
   Glissanda, who was interested in
   mathematics and was of marriageable
   age. Her one requirement for her
   future husband was that he enjoy
   mathematics as much as she did. But
   King Arthur thought that Glissanda's
   future husband should be one of the
   brave and strong knights of the
   Round Table. How could he find out
   which one had in interest in
   mathematics? The king suggested that
   a test be given. But what should the
   test be? Glissanda came up with one.
   She proposed the following problem:
   Suppose that 24 knights came and
   seated themselves at the Round
   Table. And you, dear father, going
   in numerical order, point to the
   first knight and say, "You live."
   Then you turn to the second knight
   and say, "You die!" and proceed to
   cut off his head. You continue in
   turn, alternating between sparing a
   knight and killing a knight until
   there is only one knight left. Now,
   of course, you wouldn't really kill
   any knights. It's only a problem.
   The knight who knows where to sit,
   no matter how many knights there
   are, will have solved the problem.

   How would you go about solving the
      problem?

   Type BEGIN to start.
```

```
to startup
cc
type sentence [See Flip Side for a
   description or type BEGIN to
   start.]
char 13
end

to begin
clearpage
make "key 0
knight
end

to clearpage
rg
ct
ht
cc
end

to knight
cc
initialize
enter
check.powers
print.out
end

to initialize
make "chairs 0
make "powers [32 16 8 4 2 1]
make "knight 0
end
```

```
to enter
(insert [Type the number of chairs
   which will be found around the
   ROUND TABLE. Choose a number
   between 1 and 60—]char 32)
make "chairs readnumber
if not check [print [ ] enter]
end

to check
if or (:chairs < 1)(:chairs > 60)
   [output "false]
output "true
end

to readnumber
output first readlist
end

to check.powers
ifelse or (:chairs = first :powers)
   (:chairs > first :powers) [make
   "chairs :chairs - (first :powers)
   make "knight (:chairs * 2 + 1)]
   [continue]
end

to continue
ifelse or (:chairs = first butfirst
   :powers)(:chairs > first butfirst
   :powers) [make "chairs :chairs -
   (first butfirst :powers) make
   "knight (:chairs * 2 +
   1)][continue1]
end

to continue1
ifelse or (:chairs = first butfirst
   butfirst :powers)(:chairs > first
   butfirst butfirst :powers) [make
   "chairs :chairs - (first butfirst
   butfirst :powers) make "knight
   (:chairs * 2 + 1)] [continue2]
end

to continue2
ifelse or (:chairs = last butlast
   butlast :powers)(:chairs > last
   butlast butlast :powers) [make
   "chairs :chairs - (last butlast
   butlast :powers) make "knight
   (:chairs * 2 + 1)] [continue3]
end

to continue3
ifelse or (:chairs = last butlast
   :powers)(:chairs > last butlast
   :powers) [make "chairs :chairs -
   (last butlast :powers) make
   "knight (:chairs * 2 +
   1)][continue4]
end

to continue4
ifelse or (:chairs = last
   :powers)(:chairs > last :powers)
   [make "chairs :chairs - (last
   :powers) make "knight (:chairs *
   2 + 1)] [stop]
end

to print.out
print [ ]
(print [The surviving knight will be
   in chair number ] :knight )
print [ ]
cc
type [Press spacebar to repeat; Q to
   quit]
make "r readchar
ifelse (ascii :r) = 32 [knight] [cc]
end
```

The first computer program I wrote providing solutions for the knight microworld was in BASIC. It is presented here only to provide some insights into problem solving:

```
10 PRINT "TYPE THE NUMBER OF CHAIRS
   AT THE TABLE."
20 INPUT X
30 READ A
40 Y= X - A
50 IF Y > 0 THEN 70
60 IF Y = 0 THEN 70
70 GOTO 30
80 Z = 2 * Y + 1
90 PRINT "THE NUMBER OF THE LAST
   KNIGHT IS" ; Z
100 END
110 DATA 2048, 1024, 512, 256, 64, 32,
   16, 8, 4, 2
```

The command to activate the BASIC program is RUN. A quick glance through the program provides all of the essentials used to provide a solution for the microworld.

A Briefer *LogoWriter* Program for Knights

If the Logo program given previously appears a bit overdone, relax. Our friend Ihor Charischak has come to the rescue with a program in *LogoWriter* that is much more direct. Ihor suggested this solution to the knight encounter in a letter written on 14 November 1990. I added a conditional statement in the first procedure. If

you wish to discover why, omit it and see what happens. The command for the program is **knight** with the input of the number of chairs available to the knights, for example, **knight 24**. All of the work will be available in the Command Center. The response will be the chair in which the knight should seat himself in order to survive. It's always interesting to see how different individuals tackle the same problem. Perusing both Logo programs might be profitable to the reader.

```
to knight :number
if :number = 1 [show 1 stop]
prcc 2 * (:number - get.power
   :number) + 1
end

to get.power :input
output number 2 :input
end

to number :inp :input
make "number 2 * :inp
if :number > :input [output :number
   / 2]
output number :inp * 2 :input
end

to prcc :input
type sentence :input char 13
end
```

Final Commentary

This microworld has fascinated me for quite some time. It has had an equal effect on Ihor Charischak and Richard Moss at the Stevens Institute of Technology in Hoboken, New Jersey. Ihor writes that he has Richard Moss's lessons for this encounter on videotape. Ihor has continued his exploration in an article called "Finding a More 'Definitive' Rule, or King Arthur Discovers Logarithms." Surprisingly, my friend Michael Glover went into this microworld by-way when I introduced the problem to her more than eight years ago when her son was then a fourth-grade student. The by-ways of this little puzzle seem to have no end.

References

Macdonald, R. (1990). Knight. Clime Microworlds. *Clime, Vol. 2.* White Plains, NY I would like to thank Mr. Charischak for granting permission to use this encounter in a different format.

Burns, M. (1982). *Math for smarty pants.* Boston, MA: Little, Brown and Company. The entire article (pp.75-79) should be read to a class and if possible dramatized.

Charischak, I. (1993). Knight revisited. Clime Microworlds. *Clime, Vol. 3.* White Plains, NY. In his latest consideration of the knight problem, Ihor provides a printout of the surviving knights after each round.

Barr, G. (1971). *Entertaining with number tricks.* New York: McGraw Hill. Barr does not inform the reader that the data on his charts is derived from base 2. On page 33, the grid is presented in reverse order. A B C D E F represents 1 2 4 8 16 32. A mathematical grid demands 32 16 8 4 2 1.

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

Worksheet

Name _____ Partner _____

Date _____

Read "King Arthur's Problem" in *Math for Smarty Pants* by Marilyn Burns, pages 75-79. In the following problems, you are to work out manually the number of the chair in which the surviving knight must sit. You are given the total number of knights sitting around King Arthur's Round Table.

Use this manual method:

1 Write out the total number of knights.

2 Cast out numbers by alternating "You Live" (do not cast out) and "You Die" (cast out).

3 Only one knight will survive, and his chair number is the safe one for that number of given knights.

4 At the bottom of the page, work out the sample in which 24 knights are sitting around the Round Table. In which chair must the surviving knight sit? _____.

5 Use scrap paper for casting out numbers in order to find the safe chairs for knights 1 through 60.

6 Analyze your answers, which you are to record on the following Data Sheet.

7 Look closely and THINK! Can you think of a quick mathematical way of doing this so that all of the manual work might be avoided?

```
1 2 3 4 5 6 7 8 9 10 11 12 13
14 15 16 17 18 19 20 21 22 23 24
```

Data Sheet

Name _____ Date _____

Number of Knights	Winning Chair	Number of Knights	Winning Chair	Number of Knights	Winning Chair
1. =		21. =		41. =	
2. =		22. =		42. =	
3. =		23. =		43. =	
4. =		24. =		44. =	
5. =		25. =		45. =	
6. =		26. =		46. =	
7. =		27. =		47. =	
8. =		28. =		48. =	
9. =		29. =		49. =	
10. =		30. =		50. =	
11. =		31. =		51. =	
12. =		32. =		52. =	
13. =		33. =		53. =	
14. =		34. =		54. =	
15. =		35. =		55. =	
16. =		36. =		56. =	
17. =		37. =		57. =	
18. =		38. =		58. =	
19. =		39. =		59. =	
20. =		40. =		60. =	

Check your manual work against the Logo program. Analyze the data. Can you discover a pattern? The solution is rather clever. Glissanda was a very intelligent princess.

LX Copy-Me Page!

Base 2 Grid (1)

Name _____ Date _____

	64	32	16	8	4	2	1	
1.								1.
2.								2.
3.								3.
4.								4.
5.								5.
6.								6.
7.								7.
8.								8.
9.								9.
10.								10.
11.								11.
12.								12.
13.								13.
14.								14.
15.								15.
16.								16.
17.								17.
18.								18.
19.								19.
20.								20.
21.								21.
22.								22.
23.								23.
24.								24.
25.								25.
26.								26.
27.								27.
28.								28.
29.								29.
30.								30.

LX Copy-Me Page!

Base 2 Grid (2)

Name _____ Date _____

	64	32	16	8	4	2	1	
31.								31.
32.								32.
33.								33.
34.								34.
35.								35.
36.								36.
37.								37.
38.								38.
39.								39.
40.								40.
41.								41.
42.								42.
43.								43.
44.								44.
45.								45.
46.								46.
47.								47.
48.								48.
49.								49.
50.								50.
51.								51.
52.								52.
53.								53.
54.								54.
55.								55.
56.								56.
57.								57.
58.								58.
59.								59.
60.								60.

Extra for Experts

How About a Standard Logo Lexicon?

by David P. Kressen

Logo has been around for 25 years. I have been teaching children, young people, and adults how to use the language for half this period—since the first microcomputer version of Terrapin Logo came on the market in 1979. New and more powerful implementations of Logo are appearing all the time. Students growing up in Logo environments in their schools and homes have more and more choices among Logo versions to accommodate their changing programming interests and their improving hardware. Some will migrate from one version of Logo to another and will want to share their discoveries with others who use different Logo implementations on different hardware platforms.

Communication among Logo users is handicapped by lack of a universally agreed-upon lexicon of Logo terms. The manuals for each version often use different words to describe features of the language and programming principles that are common to all.

To cite just one example, here are three definitions of what is generally called an *instruction*—a command and its inputs. The *Apple Logo Reference Manual* defines this as

> a particular kind of expression. It starts with a procedure name, and that procedure must be a command. All other procedures in the expression must be operations. (p. vii)

The *LogoWriter Reference Guide* defines an *instruction* as

> a command followed by its input(s) or by reporters which provide the inputs. (p. 4-21)

The *Win-Logo Users Manual*, without giving the concept a name, notes that

> Many primitives ... need another piece of information or "data" ... before they can be executed. ... This extra piece of information [is] called a parameter entry or an argument and is a required piece of the command string. (pp. 4-6)

I believe the time is overripe for us, the members of SIGLogo, to adopt standard terminology to describe the universal Logo features we teach to our students and write about in *Logo Exchange*. I am convinced that both we and our students would benefit from such an initiative, and I am bold—or perhaps naive—enough to propose the following definitions as a starting point.

The computer scientists among us will recognize that this lexicon is proposed for use by teachers, not for use by computer scientists. My purpose is to implement communication among people, not to explain the way the Logo interpreter deals with our keyboard entries. With help from Brian Harvey, who was kind enough to review preliminary drafts of this article, I have tried to define terms that will be useful to us lay Logoists and at the same time usable by the pros.

Those familiar with *LogoWriter* will recognize the source of many terms in the proposed lexicon and the dialect used for the illustrative examples. The accompanying chart provides a graphic reference to the relationships among the programming concepts.

Program Element. A *procedure, expression* or *delimiter* that is meaningful to a Logo interpreter. (I use the phrase "meaningful to a Logo interpreter" in the same playful sense that I use the phrase "teach the turtle." If the interpreter does not return an error message, I assume that the program element is "meaningful" to it.) Program elements are represented by keyboard characters: a letter, a digit, a symbol, or a combination of letters, digits, and symbols. A Space or Return character signals the end of the keyboard entry that represents a single program element.

Delimiter. A program element (TO or END) that signals the beginning or end of a *defined procedure.*

Expression. A program element that provides an *input value* to a *procedure.*

Procedure. A program element that has a name and that either produces an effect or calculates an output value. Except for TO and END, the interpreter assumes that keyboard entries represent procedures (rather than expressions) unless the entry is a numeral or unless it begins with a quotation mark ("), a bracket ([), or a colon (:). Procedures may be classified by *origin* or by *function.* Classified by origin, procedures are either *primitive procedures* or *defined procedures.* Classified by function, procedures are either *commands* or *reporters.*

Command. A procedure that *produces an effect.* The effect may be visual (for example, PRINT, FORWARD, or FILL) or internal (for example, MAKE, TELL, or OUTPUT). Some commands require one or more inputs to produce their effect (for example, RIGHT, TONE, or IFELSE); others produce their effect without help from inputs (for example, HOME, STOP, or TAB.) An *instruction* is a command together with

any required inputs. The interpreter *executes* an instruction to produce the command's effect. Several instructions written in sequence without a Return character are called an *instruction line.*

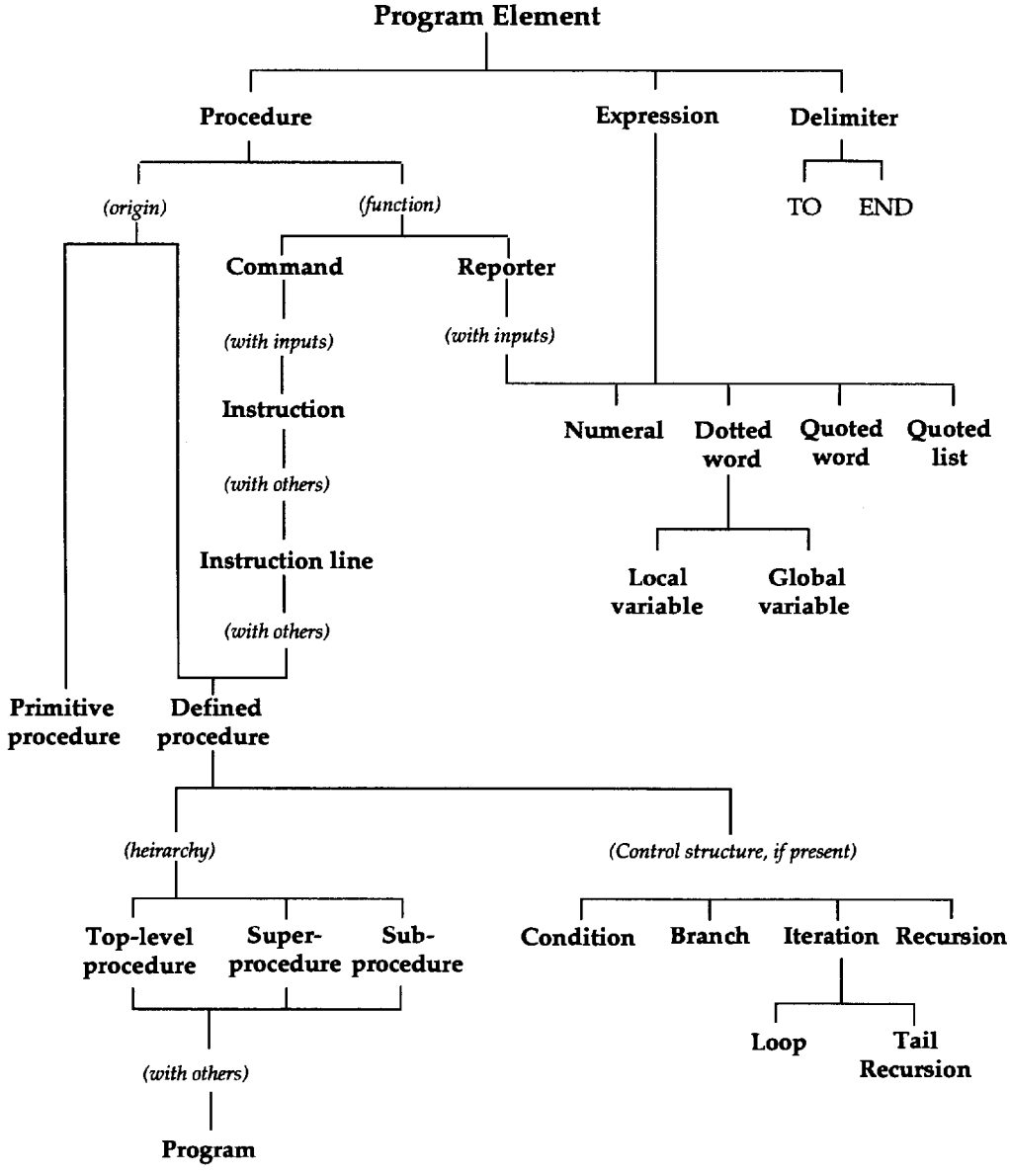
Reporter. A procedure that *calculates an output* that is then passed to a command or another reporter to use as an input. Some reporters require one or more inputs to calculate their output (for example, FIRST, =, or IFELSE); others calculate their output without help from inputs (for example, READLIST, XCOR, or COLORUNDER). Because a reporter provides an input value to a command, a reporter together with any required inputs is one type of expression.

Primitive procedure. A procedure defined by the person who wrote the Logo implementation. It is a program element that the interpreter recognizes as part of its "native language."

Defined procedure. A procedure defined by the user. It is represented by a keyboard entry chosen by the user and then "taught" to the interpreter. Defined procedures have three parts, each of which is written on a separate line:

The *title line* begins with the delimiter TO and contains the name of the procedure followed by the names of the *variables* that provide any required inputs.

One or more *instruction lines* produce the procedure's effect (if it is a command) or calculate its output (if it is a reporter.) If the procedure is a reporter, the last instruction line must contain the command OUTPUT, which provides the procedure with its output value.



The *end line* contains only the delimiter END.

Program. A group of related procedures that interact to solve a problem. The hierarchical relationships among the procedures are identified by the terms *top-level procedure, superprocedure,* and *subprocedure.*

Top-level procedure. The superprocedure whose name is typed by the user to initiate the run of a program.

Superprocedure. A defined procedure that calls another defined procedure, designated as its *subprocedure,* to assist it in producing an effect or calculating an output. The interpreter *suspends* execution of the superprocedure while it executes the subprocedure. When the interpreter reaches the end line of the subprocedure, it terminates the subprocedure and *resumes* execution of the superprocedure at the point where the execution was suspended.

Subprocedure a procedure called by a superprocedure. The call is made by writing an instruction line that includes the subprocedure's name, together with any required inputs.

Input. A value provided by an expression and passed to a procedure to help it produce its effect or calculate its output. There are three types of input values: *numbers, words,* and *lists.* Some procedures are particular about the type of inputs they receive; others are not particular in this regard. The keyboard entries representing expressions are typed immediately after the name of the procedure that requires them. There are five types of expressions, each of which has a unique representation. The interpreter *evaluates* each expression differently to obtain its value.

A *numeral* provides a procedure with a *number.* The interpreter evaluates a numeral by computing its decimal value. For example, the numeral 27 provides the input 2 tens plus 7 ones.

A *quoted word* provides a procedure with a *word.* The interpreter evaluates a quoted word by stripping off the quotation mark. For example, the quoted word "TRUE provides the input true.

A *quoted list* provides a command with a *list of elements.* The interpreter evaluates a quoted list by stripping off the outer brackets. For example, the quoted list [a b c] provides the input a b c; the quoted list [a [b c]] provides the input a [b c]; the quoted list ["true] provides the input "true.

A *reporter with any required inputs* provides a procedure with a *number,* a *word,* or a *list,* depending on the output of the reporter. The interpreter first evaluates the inputs to the reporter and then calculates the reporter's output. For example, the expression 6 + 3 provides the input 9; the expression first [the black cat] provides the input "the"; the expression butfirst [the black cat] provides the input black cat. Inputs to the reporter may be the outputs of another expression, in which case parentheses may be needed to indicate precedence or to specify the input, for example, (5 + 3) * 9 and (first [3 6]) + 5.

A *dotted word* provides a procedure with a *number,* a *word,* or a *list,* depending upon the value of a *variable.* The interpreter evaluates a dotted word by returning the value associated with the variable's name. For example, the dotted word :SIZE might provide the input 30; the dotted word :NAME might provide the input GEORGE; the dotted word :VERB.LIST might provide the input RUN JUMP SLIDE.

Variable. The two-part entity that provides a dotted-word input. A variable has a *name,* a *value* associated with this name, and a *scope,* which specifies the procedures in which it may be used.

A *global variable* is *created* and *assigned* its value by the commands MAKE and NAME. The variable is stored in memory until specifically deleted. Its scope is all procedures that do not contain a local variable with the same name.

A *local variable* is given its name in the title line of a defined procedure. The variable is created and assigned its value when the defined procedure is executed, and the variable ceases to exist when the procedure is terminated. The value may be supplied by the user (by typing the value after the procedure name) or by a superprocedure (by including the value after the procedure name in the calling instruction). Its scope is limited to the procedure in which it is named. Variables may be made local to a particular procedure with the command local.

Control Structures. The interpreter executes instructions in sequence from left to right on each line and from the first line to the last line of a procedure unless directed otherwise by a control structure. Four control structures are available:

A *conditional structure,* implemented by the command if, provides for the interpreter to skip a certain list of instructions. A *test reporter* outputs the word *true* or *false,* which determines whether or not the instructions should be skipped. If the input to if is true, the instruction list is executed; if the input is false, these instructions are skipped. See the stop rule example below.

A *branch structure,* implemented by the command ifelse, provides for the interpreter to execute either

of two possible instruction lists. If the test reporter outputs true, the first list of instructions is executed; if the test reporter outputs false, the second list is executed, for example,

```
IFELSE :ANSWER = 50 [print "Correct]
   [print "Wrong]
```

IFELSE may also be used as a reporter, in which case the interpreter evaluates either of two possible expression lists and outputs the result to a command, for example,

```
PRINT IFELSE :ANSWER = 50 ["Correct]
   ["Wrong]
```

An *iterative structure* provides for certain instructions in a procedure to be run more than once when the procedure is executed. There are two forms of iteration:

A *loop*, implemented by the command repeat, provides for the interpreter to repeat a list of instructions a designated number of times. The number of repetitions is known in advance and the same instructions are executed on each *pass* through the loop, for example,

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

A *tail recursive procedure* is implemented by writing the name of a defined procedure on the last line of a procedure that has the *same name*. If the procedure is a command, this *iterative call* may be part of a longer instruction line, but if the procedure is a reporter, the iterative call must provide an input directly to the output command rather than provide an input to another reporter. The effect of a tail recursive command is similar to that of a loop without a predetermined number of repetitions. Each pass may produce a slightly different effect by changing the input for the next iteration. A conditional structure, called a *stop rule*, is used to terminate the execution of the procedure, for example,

```
TO SPIRAL :SIZE
IF :SIZE > 100 [STOP]
FORWARD :SIZE RIGHT 90
SPIRAL :SIZE + 5
END

TO SPIRAL :SIZE :NUMBER
IF :NUMBER > 10 [SPIRAL :SIZE + 5
   :NUMBER - 1]
END

TO PICK :NUMBER :LIST
IF EQUAL? :NUMBER 1 [OUTPUT FIRST
   :LIST]
OUTPUT PICK :NUMBER - 1 BUTFIRST
   :LIST
END
```

A *recursive structure* is implemented by including the name of a defined procedure in an instruction line *before* the last line of a procedure that has the same name, or (if the procedure is a reporter) as an expression on the last line that does not provide an input directly to the output command. This is called a procedure with *embedded recursion*. The interpreter executes the procedure by *cloning* a new copy of the procedure each time it reaches the instruction containing the procedure's name (the *recursive call*). The original procedure becomes a superprocedure for its clone, suspending its own run while the clone is executed. This process continues until the stop rule terminates the run of one of the clones, at which point the interpreter resumes execution of each superprocedure in turn. Recursive procedures frequently appear as subprocedure reporters; each clone outputs a partial solution of the problem to its own superprocedure, and the original version outputs the final solution to the superprocedure that called it, for example,

```
TO ADD :ADDENDS
PRINT SENTENCE [The sum is] DO.ADD
   :ADDENDS
END

TO DO.ADD :ADDENDS
IF EMPTY? :ADDENDS OUTPUT [0]
OUTPUT FIRST :ADDENDS + DO.ADD
   BUTFIRST :ADDENDS
END
```

References

Apple Logo reference manual. (1982). LCSI.

Harvey, B. (1985). *Computer science Logo style, Volume 1: Intermediate programming.* MIT Press.

Kressen, D. (1994). *All about Logo.* Available from the author as shareware.

LogoWriter reference guide. (1987). LCSI.

Watt, D *Learning with Logo.* (1983). McGraw-Hill.

Win-Logo users manual. (1992). Concord, MA: Softeast Corp.

David P. Kressen
Pacific Oaks College
3081 Oneida Street
Pasadena, CA 91107
818/792-8546
dkresse@ctp.org

Learning by Design

by Douglas H. Clements and Julie S. Meredith

Computer games continue to attract and hold many children's interest. Would they be as interested in *designing* computer games in Logo? What could they learn from such experience?

Designing Software for Learning

To set the stage, we have to recall an earlier study. Harel (1991) had fourth-grade students design software to teach fractions to third graders. Students were given both the freedom and the responsibility to create their own designs and to teach themselves about fractions and representing fractions to other children. Students wrote in Designer's Notebooks before and after each working session.

Harel used both quantitative and qualitative research procedures to investigate children's learning. The children were divided into three groups: the instructional design (ID) group and two comparison groups. The first comparison group was given the same amount of exposure to Logo programming as the ID group. This exposure was integrated with various curriculum topics; however, the projects tended to be short and assigned by the teacher. The second comparison group received Logo once a week in a computer literacy course.

The ID group showed greater mastery of both Logo and fractions than the two comparison groups. Why?

The ID children's projects revealed a rich variety of ways to represent fractions. They divided a circle into four regions and flashed two on and off to show two-fourths with the written text "two-fourths"; they showed an animated clock; and they showed a one-dollar bill with four quarters underneath, two of which moved and stopped near the written words "two-fourths of one dollar."

As one girl, Debbie—who was initially not emotionally involved in the project—insightfully stated, "Fractions can be put on anything!" This statement is all the more remarkable if you compare it to her response, before the project began, to the question, "What is a fraction?" Debbie drew a circle divided into two, shaded the right half, and stated "That's a fraction." When the investigator asked her about the other half, she replied, "That's not a fraction; that's nothing." Debbie's fraction knowledge was limited and disconnected from her everyday world (Papert, 1993). But in designing her Logo software, she came to appreciate that there are many ways to represent fractions. It happened when she figured out that the animated decorations for poems she had written could be used to make representations of fractions more aesthetic. Other children asked her about the effects. She began to see fractions everywhere and to appreciate that people "impose" fractions on anything they wish.

There is another reason the instruction was so effective. Debbie and the other ID children were creating and programming these representations for other children. They had, in other words, a real audience.

Designing Software Games

Kafai (1993) built on this early work, but took a different tact. For six months, she worked with fourth graders in an inner-city school designing computer games to teach third graders about fractions.

What effects did this experience have? Kafai compared their learning to the learning in Harel's groups. (Harel's students will be called the ID group, and Kafai's the "game-design" group; together, these are the "design" groups). The game-design children's knowledge of Logo programming and fractions improved significantly from pretest to posttest.

Both the game-design and ID classes learned nearly twice as many Logo commands as the control group did. (One caveat: The students were somewhat better achievers at pretest.) The design classes also did a better job sorting Logo commands into meaningful categories. Finally, during the course of the study, the design classes kept on learning more commands. In comparison, the control classes reached a plateau and stopped progressing.

The design classes drew a picture from Logo code better. They did better on simplifying of Logo code, giving the number of inputs to Logo commands, and transforming Logo statements, for example, take a circle procedure and make a larger circle. The ID class did the best on this task, followed by game-design class, and then the control class.

How well did they learn fractions? Again, the ID class came out on top, the game-design class second, and then the control class. The game-design class performed a bit better than the ID class on a question about what fraction of a given group of balls were tennis balls.

To explain these results, Kafai notes that the game designers concentrated on making their games *fun* rather than concentrating on fractions. Also, to achieve

animation, the game designers used turtle shapes rather than shapes drawn with the turtle, which may have limited them in the representation of fractions.

For these reasons, they may not have created a more personal view of fractions or connected fractions to everyday usage to the degree the ID students had. In sum, Kafai and her children concentrated more on the game aspect and on learning programming. Harel and her children emphasized representing fractions.

The game-design children did successfully create complex software products designed for use by others. They learned a significant amount about programming, and more about fractions than the comparison children did.

Implications

Both studies are rich in detail. This is but the briefest of summaries. We can, however, still draw significant implications.

1. Designing in Logo appears to hold real potential for learning.
2. Designing software to teach concepts (e.g., as with the ID class) seems to be an effective means of learning mathematical content (such as fractions), of creating a personal perspective on that content, and on connecting the content to everyday life.
3. Designing Logo games appears to be an effective means of learning design and programming, and a different, if somewhat less effective, means of learning mathematical content.
4. A good Logo environment—one with features that support the designing—offers real advantages to this type of design activity. A powerful aspect of Logo appears to be giving children control over their own representations of mathematical ideas. Various pictures, animations, and texts can be composed, selected, and combined.
5. Children should keep notebooks of their work. The notebooks became a source of ideas, organization, and discussion.
6. Logo programming may contribute to general mathematical ability. These two projects have shown what an intensive four months of meaningful Logo programming can do. We have yet to see an investigation of several years' use of Logo (Clements & Meredith, 1993).

References

Clements, D. H., & Meredith, J. S. (1993). My turn: A talk with the Logo turtle. *Arithmetic Teacher, 41*, 189-191.

Harel, I. (1991). *Children designers.* Norwood, NJ: Ablex.

Kafai, Y. B. (1993). *Minds in play: Computer game design as a context for children's learning.* Unpublished doctoral dissertation, Harvard University. (Forthcoming as a book published by Lawrence Erlbaum Publishers, Hillsdale, NJ.)

Papert, S. (1993). *The children's machine. Rethinking school in the age of the computer.* New York: Basic Books.

Time to prepare this material was partially provided by the National Science Foundation under Grants No. 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, 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. Through a National Science Foundation (NSF) grant, he has 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. With Julie S. Meredith, he is coauthoring a new version of Logo, *TurtleMath*, for learning elementary geometry.

Julie Sarama 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 Douglas Clements, she is currently designing and programming a new version of Logo for the NSF-funded "Investigations" project.

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

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		

BK to Russia

BK in 1983, I visited the Soviet Union and wrote a column for *Logo Exchange* on the state of Logo and technology in the Soviet schools. The column was entitled BK to the USSR. This past summer, I was the guest of the Institute of New Technologies (INT) in Moscow. I spent five weeks criss-crossing Russia, giving workshops and speeches and talking to administrators and teachers. It was gratifying to see Logo use expanding into more Russian schools.

The person most responsible for Logo usage in Russia is Sergei Supronov, a former mathematician from Moscow State University who has been with INT for eight years. Dr. Supronov and his colleagues are at the forefront of changing Russian schools through technology. Acting as an international courier, I delivered the latest version of *MicroWorlds* to Dr. Supronov from Brian Silverman of LCSI. INT's band of brilliant programmers immediately began translating the disks into Russian.

INT earlier translated *LogoWriter*, including the reference manuals, into Russian. This Russian version was piloted in one school for two years, and then Dr. Supronov wrote a short Russian Logo workbook for teachers. Seymour Papert gave some workshops, and more schools started using Logo. Presently about 40 Moscow schools and 100 throughout Russia and the former Republics are using Logo, mostly in grades 2-8.

Two kinds of teacher-training workshops have been developed by INT. One is a three-day introductory workshop (six hours per day). Follow-up discussion sessions with each school are then held to discuss strategies and problems. Dr. Supronov personally teaches two second-grade Logo classes that he has had since the students were in first grade.

Logo work mostly takes place in a laboratory setting. A typical Russian school has a 15-station computer lab. These labs are typically PC computers, although education authorities throughout the country are becoming less enamored with PCs and are investigating other platforms—the Republic of Kariellia recently decided to place 15 Macintosh LC 475 computers into each of its 152 schools. Two students share one computer in the lab and work with Logo two hours per week.

During the first four years of Logo usage in Russia, Logo was used, according to Supronov, in a "freestyle way, with INT only advising schools and showing them some of Logo's potential." Logo integration is now becoming more systematic, with handbooks available on such topics as "How to Use Logo in Math and Language Classes." Children are now given tasks and projects to carry out.

Little Logo research has been done in Russia. A questionnaire was recently distributed to 20 Logo-using teachers. Only one negative comment to any of the questions surfaced, and it was about the old-fashioned, unreliable computers the teachers had to use.

Despite the turmoil and day-to-changes taking place in Russia, education reform is moving forward. Many Russian teachers never thought they could change the way schools operate because of the centralized curriculum planning of the Soviet era. It comes as somewhat of a surprise that Russia is finding the money for technology in the schools and that teachers are embracing computer use in the classroom as a way restructure their outdated school system.

Dr. Supronov visited LCSI in July and continues to develop a Russian version of *MicroWorlds*. You can contact him on the Internet at aseminov@glas.apc.org or on Apple Link at INT.SU.

Dennis Harper
1113 Legion Way SE
Olympia, WA 98501

ISTE Books & Courseware Order Form

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

Product (• Indicates an ISTE-published title.)	*Member Price*	*Nonmember Price*
• *HyperStudio in One Hour*	16.15	17.95
Internet Companion—A Beginner's Guide to Global Networking	12.95	12.95
• *Introduction to MicroWorlds—A Logo-Based Hypermdia Environment*	22.45	24.95
• *Way of the Ferret—Finding Educational Resources on the Internet*	22.45	24.95
The Whole Internet—User's Guide & Catalog	24.95	24.95

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

Name _____ Membership # _____

School/Business _____

Address _____

City _____ State _____ Zip/Postal Code _____

Country _____ Phone _____

Shipping & Handling

$0-$15.99 (subtotal)	add $3.50
$16-$45.99 (subtotal)	$5.00
$46-$75.99 (subtotal)	$6.00
$76-$99.99 (subtotal)	$7.00
$100 or more	8% of subtotal

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

GST Registration Number 128828431

LX4

ORDER

Quantity	Title	Member Unit Price	Nonmember Unit Price	Total Price

PAYMENT OPTIONS

☐ **Payment enclosed.** Make checks out to ISTE—
International orders must be prepaid with U.S. funds or credit card.
☐ VISA ☐ MasterCard ☐ Discover Card Expiration Date _____

☐ **Purchase Order enclosed.** Please add $4.00 for order processing—
P.O. not including $4.00 fee will be returned.

☐ **C.O.D.** for U.S. Book orders only. You will pay UPS the total upon delivery by check or cash—
ISTE will add $4.50 order processing.

☐ **Airmail.** International orders for Books & Courseware are sent surface mail—
ISTE will bill you the additional shipping charge for Airmail.

☐ Send me ISTE membership and subscription information.

☐ Send me a free ISTE catalog.

SUBTOTAL	
Subtract 18% on ISTE-published titles if ordering quantities of 10 or more	-
SUBTOTAL	
Shipping and Handling (see box above)	+
Add Additional 5% of SUBTOTAL if shipped to PO Box, AK, HI, or outside U.S.	+
Add 7% of SUBTOTAL for GST if shipped to Canada	+
If billed with purchase order, add $4.00; If COD, add $4.50	+
TOTAL	=

ISTE • 1787 Agate Street, Eugene, OR 97403-1923 • Order Desk 800/336-5191• Fax 503/346-5890

We Want You On Our Team!

"These are exciting times for educators involved with technology. As past-president of the International Society for Technology in Education, I am impressed with the abilities and resourcefulness of our professional educators who are bringing today's technology into the classroom.

"As educators, we must provide leadership for change. New technologies, funding requirements, and a myriad of hardware and software choices make our roles as classroom teachers, technology coordinators, and school administrators somewhat complex and challenging.

"The mission of ISTE is to improve education through the use of technology in learning., teaching, and administration. ISTE provides national and international leadership to promote that mission. We seek to nurture a professional community of individuals and organizations interested in technology in education. To meet this need, we provide timely information, materials, and services to help enable our members to be more productive and successful at their jobs.

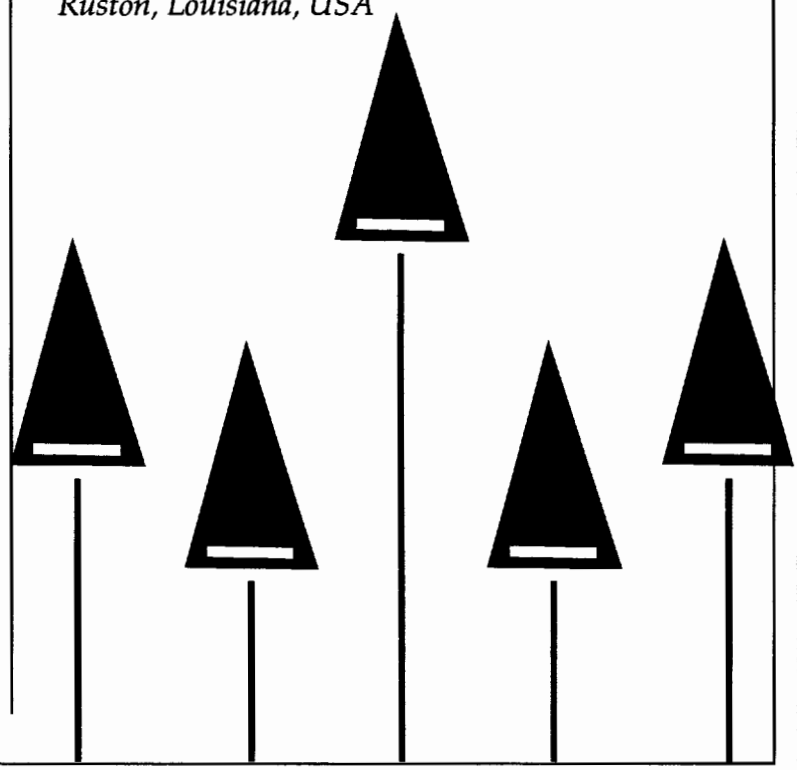
Contact ISTE for a free catalog with information on membership, subscriptions, books, and courseware.

International Society for Technology in Education
1787 Agate Street, Eugene, OR 97403-1923 USA
Phone: 503/346-4414 Fax: 503/346-5890
Order Desk: 800/336-5191
America Online: ISTE
AppleLink: ISTE
CompuServe: 70014,2117
Internet: ISTE@Oregon.uoregon.edu

"Of course, ISTE must rely on our network of Organization Affiliates—educational technology membership groups located throughout the United States and in other countries—to carry out the ISTE mission in schools and communities throughout the world. ISTE also 'partners' with other professional organizations with similar goals and values, with businesses and industries, with state, national, and international educational leaders, and with parents to provide today's students with the technological skills and knowledge needed for success in their future.

"I'm certainly proud of my association with ISTE and I encourage its growth and support. If you're not already an ISTE member, come join us. We want you on our team."

Lajeane Thomas
Past-President, ISTE
Louisiana Tech
Ruston, Louisiana, USA



"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