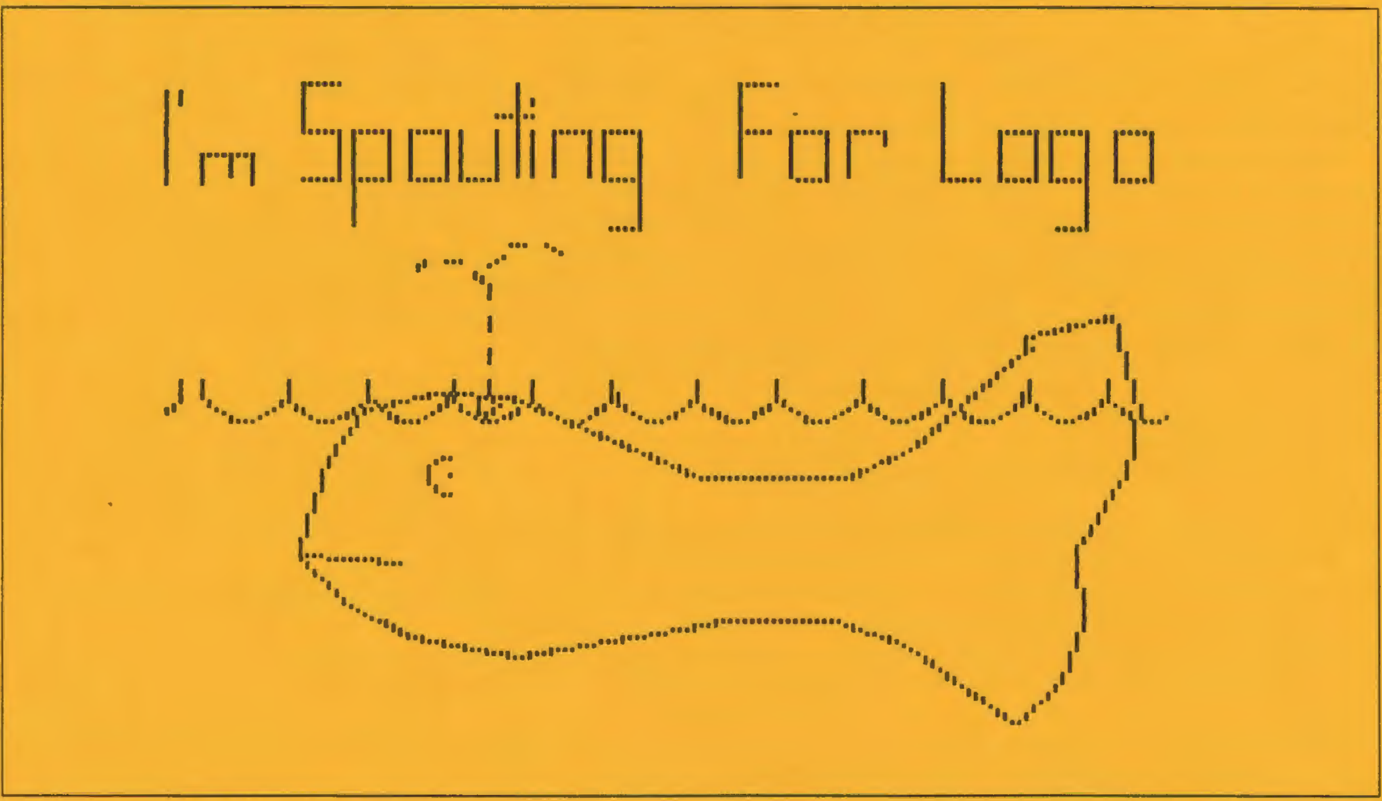
Journal of the ISTE Special Interest Group for Logo-Using Educators

LOGO EXCHANGE

November 1990

Volume 9 Number 3



International Society for Technology in Education

ISTE Publications

LOGO EXCHANGE

Volume 9 Number 3 Journal of the ISTE Special Interest Group for Logo-Using Educators November 1990

Logo Exchange is the journal of the International Society for Technology in Education Special Interest Group for Logo-using Educators (SIGLogo), published monthly September through May (including combined December/January issue) by ISTE, University of Oregon, 1787 Agate Street, Eugene, OR 97403-1923, USA; 503/346-4414. This publication was produced using Aldus *PageMaker*®.

POSTMASTER: Send address changes to Logo Exchange, U of O, 1787 Agate St., Eugene, OR 97403-1923. Second-class postage paid at Eugene OR. USPS #000-554.

Contents

From the Editor— Looking BACK *and* FORWARD
Sharon Yoder 2

Monthly Musings—Just a Little Bit
Tom Lough 3

Beginner's Corner—Quilt-making Revisited
Dorothy Fitch 4

Questions Please!
Frank Corley 6

Logo Ideas—Two New Ideas
Eadie Adamson 7

Logo LinX—DisSPELLing a Myth
Judi Harris 11

Logo Linquistics: What to do with those "Silly Sentences"
Mark L. Evans 14

Logo Connections—Imported Graphics
Glen L. Bull and Gina L. Bull 20

Object Logo is Back
Hazem I. Sayed 24

MathWorlds—Logo and Mathematics: Sprite Control with First Graders
João Filipe Matos
Sandy Dawson, editor 26

Extra for Experts—Fractals III: Monsters of Mathematics
Mark Horney 28

Logo: Search and Research—Strategies for Writing Procedures
Douglas Clements 32

Global Logo Comments—Report from Japan
Dennis Harper, editor 35

ISTE Membership

	U.S.	Non-U.S.
	36.00	43.00

SIGLogo Membership (includes *The Logo Exchange*)

	U.S.	Non-U.S.
ISTE Member Price	25.00	30.00
Non-ISTE Member Price	30.00	35.00

Send membership dues to ISTE. Add $2.50 for processing if payment does not accompany your dues. VISA, Mastercard, and Discover Card accepted. Add $18.00 for airmail shipping.

From the Editor

Looking BACK *and* FORWARD

History is in the air this year. Have you noticed? Quite a number of conferences have themes like "Ten years of educational computing." Educational computer magazines are doing special issues on the past. It looks as if we'll be doing a lot of looking back this year.

Just to keep up with the times, let's take a moment and look back at Logo. My *Apple Logo* master disk carries a copyright date of 1982. My *Terrapin Logo* master has an MIT copyright date of 1981. Even before I worked with those versions of Logo on the Apple II, I played with that delightful version of Logo on the TI 99-4A computer. Indeed, Logo has also been a part of the educational scene for nearly 10 years.

And how Logo *has* changed in the last 10 years. Less than 10 years ago, we were thrilled with a Logo that would not allow graphics and text on the same part of the screen. We were delighted with color and only moderately concerned that there was no FILL command. We were happy to use the klunky Logo editors as word processors of sorts.

Those early machines were quite different than those we use today. Some of the earliest machines had only 4K of internal memory. I thought my first machine was a wonder with 16K. And the day I replaced my tape player with a disk drive...well, I was in heaven! (Of course, I also remember wondering what on earth I would do with more than one floppy disk. After all, my tapes all fit onto *one* of them! How times change.)

Of course we can all reminisce and reflect endlessly. If you want to take a look back to see what others have to say about the past, find a copy of the "looking back" issue of *Electronic Learning* (May/June 1990) or *Technology Learning* (September, 1990). (Note that *Technology Learning* is the new name for a new decade of *Classroom Computer Learning*.) You might also enjoy attending one of the conferences with a focus on the last 10 years.

Dave and I did our own bit of looking back this summer as we took our vacation driving down the east coast. In addition to trips to the beach and visits with friends, we explored the Computer Museum in Boston and wandered through the technology exhibit in the Smithsonian Museum of American History. Both of these museums provide a look back to the technologies that preceded computing as well as a glimpse of the actual machines that started the revolution in which we now find ourselves. Perhaps even more startling is to find machines in these museums that we used only a few years ago as well as machines that are even now sitting on our desks. To be in a field where the museum pieces match the real world is absolutely amazing and certainly indicative of rapid change. No wonder it's so hard to "keep up."

Looking back is fascinating, but if we stand still too long to reflect, the future will pass us by. It seems that every few months there is an announcement of new computer equipment. Prices continue to fall. Capabilities continue to increase. Things we thought we would never see in schools are becoming affordable. Did we ever believe that laser printing would be feasible in a school setting? Did we ever think that we could create professional quality school newspapers and yearbooks "in house?"

And what about the Logo community? Where are we going? Over the past few years we have seen remarkable changes in Logo. First *LogoWriter* revolutionized our expectations of Logo. Next *Logo PLUS* offered us new capabilities based in a traditional model of Logo. And what next? This issue of *LX* is full of indications. LCSI has recently released *Logo Express*, a package that allows telecommunications using Logo. (Telecommunications with Logo? Yes!) And, for those of you who are interested in "high end" Logos, *Object Logo* is back after many, many months of languishing in an unavailable limbo. Then there is *Findout*. This new package being developed in Japan is discussed in the Global Logo Comments column. Talk about exciting new ideas!

There are yet other exciting ideas on the horizon. Research with LEGO/Logo is moving towards free standing robots with "intelligent bricks" at their center. Terrapin, Inc. is revising its Logo for the Macintosh. LCSI is completing work on *LogoWriter* for the Macintosh. Work continues on such projects and BOXER, Function Machines, and others. (*Technology Learning*, September 1990.)

It is clear that Logo still has a viable place in educational computing. No matter how the software itself evolves, the philosophy of education that drew many of us to Logo in the first place must always be a part of Logo products. We will insist on it! Just as we wouldn't cling to our 4K microcomputer, we must not cling to our early-1980s version Logo. Let's take the new Logos into the next decade with a sense of vision and purpose.

Sharon Yoder
ISTE/SIGLogo
1787 Agate Street
Eugene, Oregon 97403
503-346-4414 or 2190
BITNET: YODER@OREGON

Monthly Musing

Just a Little Bit
by Tom Lough

Just for fun, let's imagine that you see an article in a major computing magazine with the headline, "Logo—More Than a Language." What would you do?

Just for argument's sake, suppose that you are curious enough to take a look at the first sentence. Maybe it would say something like this. "Many of today's professional programmers cut their teeth on Logo." Would you honestly believe that if you saw it in print?

After reading this, maybe you are wiling to take a chance on the second sentence. "It has all the elements of a full-blown language and performs wonderfully." Hmmm. Not bad.

Would you go on to the third sentence? "Elementary and middle school teachers who've seen students progress over the years vouch for the way Logo sharpens students' cognitive processes for future programming endeavors." My goodness.

Ah, but this is just an exercise in wishing. Surely no self-respecting computing magazine would lower itself to print such drivel. Besides, right here in this column last month, Tom Lough said that Logo doesn't get any respect.

Well, folks, we have waited a long time, but such an article has happened, published on page 10 in the May 1990 issue of *Compute!* magazine. Written by no less a person than Richard Leinecker, programming manager for *Compute!* Publications, the article continues after its opening salvo to describe in detail how Logo helps students along the road to abstract thinking and creative writing. Although Leinecker writes from a programming point of view, he makes important connections with traditional curriculum areas and critical thinking skills.

Near the end of the article, he suggests that "Logo has been around for a good while, but it hasn't been exploited to its fullest." No kidding! And with the newer more powerful versions, the latest generation of related programs such as *Logo Express*, and the opportunity at last to connect Logo to a wide variety of other technological instruments, we haven't even started yet!

In the past, *Compute!* magazine has demonstrated a more than tolerant attitude toward Logo. Those of you who are already *Compute!* readers have seen David Thornburg's excellent earlier columns mention Logo frequently and positively, for example. If you presently are not a *Compute!* subscriber, maybe you should consider becoming one. (See below.)

I believe that magazines such as *Compute!* and articles such as Leinecker's will help computer users to realize the long term impact Logo is having on our educational system and our society. We have been patient for a number of years. Now it looks as if maybe, just maybe, Logo might be positioned to get a little respect. Just a little bit.

FD 100!

For subscription information, write to
Compute! Magazine
324 West Wendover Avenue
Suite 200
Greensboro, NC 27408.

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

About the Cover

This month's cover was submitted by Donna Rosenberg, a computer specialist in the Boston Public School. This project,created by Allyson Butler and Nicole O'Neill, won first place for their school district (District D) and received a trophy for fourth place in the Boston city-wide contest. Donna writes:

Every year, Boston holds a computer contest for elementary, middle and high school students. Elementary students submit Logo projects. I encouraged my fifth grade students at the Patrick J. Kennedy Elementary School in East Boston, to work on a Logo Project.

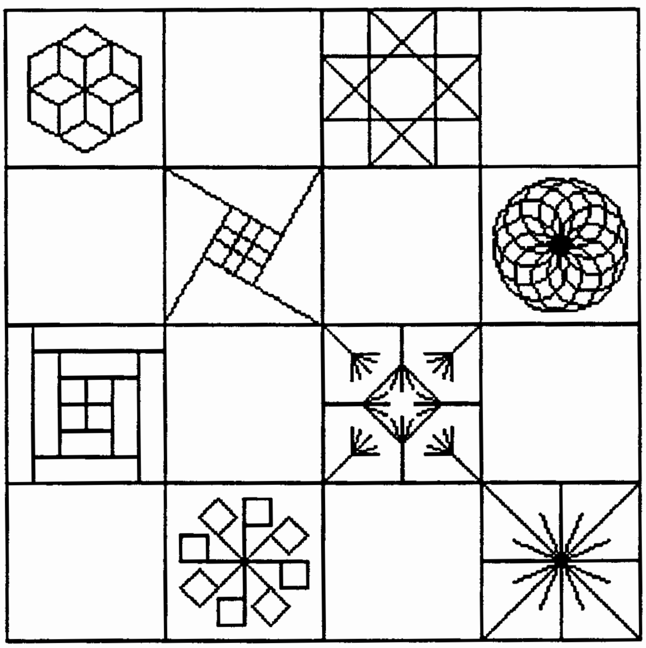
Allyson and Nicole decided to use a project Allyson had written for Science in their fifth grade room. Taking the 'cover' from her project, they worked to duplicate it using Logo. Once the whale was finished they decided to write a slogan. Again they wrote each letter using Logo primitives. Each time the whale spouted two musical notes would play, and a word would be added to the slogan.

Donna can be reached at P. J. Kennedy School, 343 Saratoga Street, E. Boston, MA 02128.

Beginner's Corner

Quilt-making Revisited
by Dorothy Fitch

Over the years, quilt-making has become a traditional Logo project. In this column we'll suggest some general guidelines for making a class quilt. Next month we'll explore symmetry and rotation in quilt designs.



A quilt project is both fun and educational for a number of reasons:

- it is an ideal way to conclude a turtle graphics unit.
- it is well-suited to group work.
- it is equally appropriate for Logo beginners and experts.
- it provides an opportunity for personal expression and creativity.
- it can be integrated into other areas of learning, such as history and art.
- it makes a "publishable" product of which everyone can be proud.
  (Examples of publishing a Logo picture include displaying it on a school bulletin board, printing it in a local or school newspaper, and sending it to the editor of *Logo Exchange* to use as cover art!)

A quilt with 16 squares will fit neatly on the computer screen and will give you squares that are a good size to work with. Each student or group of students will be responsible for one square. Then you can create a short program to combine all the squares into one quilt.

It will be easier to put the quilt squares together if each group is assigned a number. The group should then design a square according to the following guidelines:

1. Give every procedure you write a name that includes your group's number. Your main procedure should be called SQUARE followed by the number; for example: SQUARE14. (This is important when you load all the procedures into one computer; it prevents the possibility of having multiple procedures with the same name.)

2. Each square should be 60 turtle steps on a side.

3. Keep your design within the border of your square.

4. The turtle should begin and end at the lower left corner of the square, pointing up.

5. When your square design is complete, make the MOVE procedure (below) the last instruction in the main procedure. The turtle is left at the lower right corner of the square and ready to begin the next square in the quilt.

While your students are busily creating their individual squares, you or a student can put together the program that will serve as the template for the entire quilt.

```
TO QUILT
SETUP
ROWS
FINISH       ; this procedure is only needed for
END          Apple computers

TO SETUP
DRAW              ; your version may use
                  CLEARSCREEN or
HIDETURTLE        CLEARGRAPHICS
PENUP
SETXY -120 60     ; your version may use SETPOS
PENDOWN           [-120 60]
PENDOWN
END

TO ROWS
REPEAT 4 [ONEROW NEXTROW]
END

TO ONEROW
REPEAT 4 [SQUARE]
END
```

```
TO NEXTROW
PENUP
LEFT 90
FORWARD 240
RIGHT 90
BACK 60
PENDOWN
END

TO SQUARE  ; this procedure is included just for testing
REPEAT 4 [FORWARD 60 RIGHT 90]
MOVE
END

TO MOVE
RIGHT 90
FORWARD 60
LEFT 90
END

TO FINISH
PENUP
SETXY -120 (-119)   ; your version may require
RIGHT 90            SETPOS [-120 -119]
PENDOWN
FORWARD 240
PENUP
HOME
END
```

Here is a sample procedure for a quilt square.

```
TO SQUARE7
REPEAT 4 [FORWARD 60 RIGHT 90]
RIGHT 90
FORWARD 18
LEFT 90
REPEAT 8 [FORWARD 60 RIGHT 135]
LEFT 90
FORWARD 18
RIGHT 90
MOVE
END
```

Once all 16 squares are finished, modify the ROWS procedure to look like this

```
TO ROWS
SQUARE1  SQUARE2  SQUARE3  SQUARE4
   NEXTROW
SQUARE5  SQUARE6  SQUARE7  SQUARE8
   NEXTROW
SQUARE9  SQUARE10 SQUARE11 SQUARE12
   NEXTROW
SQUARE13 SQUARE14 SQUARE15 SQUARE16
   NEXTROW
END
```

Or, you might prefer

```
TO ROWS
ROW1
NEXTROW
ROW2
NEXTROW
ROW3
NEXTROW
ROW4
END

TO ROW1
SQUARE1
SQUARE2
SQUARE3
SQUARE4
END

TO ROW2
SQUARE5
SQUARE6
SQUARE7
SQUARE8
END

TO ROW3
SQUARE9
SQUARE10
SQUARE11
SQUARE12
END

TO ROW4
SQUARE13
SQUARE14
SQUARE15
SQUARE16
END
```

Have each group save its square (and any subprocedures that are used) on a data disk, *first making sure that the name of each procedure includes the group's number*. Then load all the files into one computer that already has the quilt procedures given above in its workspace. You can test each square independently by typing, or, if you are brave, type QUILT to see the entire masterpiece!

The squares don't have to be in numerical order. After looking at the quilt, students may wish to arrange them differently by modifying ROWS or ROWS and its subprocedures.

Beginner's Corner- continued

Save the entire quilt project on a disk. You will also want to save the screen design for the quilt using SAVEPICT (or your version's equivalent). That way, you can quickly load the finished design without having to wait for it to be drawn square by square. Print the quilt design and make enough copies for each student.

If you have several classes making quilts, create a slide show so that everyone can see all the projects. Here is a simple model for a slide show procedure. (If your version of Logo has a WAIT command, you can use it instead of the REPEAT 5000 [ ] instruction.)

```
TO SLIDESHOW
DRAW      ; your version may use CS  or CG
FULLSCREEN
HIDETURTLE
READPICT "QUILT1     ; the command for reading
                     pictures may be different
REPEAT 5000 []       ; this causes a delay
READPICT "QUILT2     between pictures

REPEAT 5000 []       ; change the number for a
READPICT "QUILT3     longer or shorter delay

REPEAT 5000 []
SLIDESHOW            ; include this line if you
END                  want a continuously
                     running show
```

You can show as many picture files as you can fit on one disk!

Suggested reading for next month's explorations is the September (1990) *Arithmetic Teacher* article "Symmetry in American Folk Art" by Claudia Zaslavsky. (*Arithemetic Teacher* is an official publication of the National Council of Teachers of Mathematics.)

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

Questions, Please!

by Frank J. Corley

Last's month's theme song was "We've Only Just Begun." I tried to pose small, easily answerable questions concerning specific situations likely to be encountered by a teacher or student new to Logo. This month, we go to the opposite extreme. This month I will try to pose "big" questions. These are questions that require extensive answers from experienced Logo users.

Some of these questions are "big" simply by virtue of the fact that they require long answers; they're questions that will require a lot of steps or a long list of resources to solve a problem. Other questions are quite thought-provoking. While many have been considered before in these pages, they call out for an article by one or several Logo experts in response.

While I do not want to incite a riot, I would love to see—either in this column or elsewhere in *LX*—point/counterpoint responses to some of these questions; they deserve discussion and lend themselves to differences of opinion.

Here They Are!

More than one person has asked me to include this first question. It may take one or several full articles to answer completely, but perhaps someone can provide a list of references that begin to answer the question.

1. Please give some ways that Logo can be used to enrich other areas of the curriculum, including literature, science, spelling, social studies, art and physical education. What about using Logo to teach a unit on native Americans?

I'm not sure whether this column is the place to ask the next question, but because it was asked of me, I'll pass it along. I hope that I have included enough detail to elicit the correct answer.

2. Bull, Bull, and Appert, in a *LX* Jan. '90 article called *Sight and Sound* left open exactly how to run a videodisc player from *LogoWriter*. Would they please share in print the procedures and primitives necessary to do this? If they know the answer, it would keep all of us from having to call LCSI, as they suggested.

While I *did* say that I did not want to incite any riots, these next two questions could provide some entertainment with the right mix of individuals debating. I did not make them up; they were actual questions proposed to me by real teachers.

3. What can be done—short of spending hours of exploration—to learn enough Logo to begin using it with students?

4. Why should my students learn Logo? Won't they just need to use the computer as a tool, anyway?

This summer, I begin reading Dennis Harper's *Logo Theory and Practice*. It may provide the best answers to this next question. Perhaps, though, someone can provide us with a briefer list of favorite answers to this question.

5. Is there a good list of Logo books accessible to beginners that would provide the educational and programming background, and a start on how to teach Logo?

   If you were stranded on a desert island with nothing but your computer and your Logo software what five (or six or 10) books would you bring?

I have several more of these questions that I called "big," but space is limited and I don't want to use all my good questions at one time. I hope that these inquiries have provided food for thought for the experienced and inexperienced Logo user, and perhaps food for a few articles in upcoming issues.

I should be receiving responses to some of the earliest questions by the time you read this, I hope in time for the next deadline. If so, next month this column will contain its first answers. I hope that all of your questions are answered, but if they aren't, send them to me!

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

Logo Ideas

Two New Ideas
by Eadie Adamson

Idea Number One
Required Reading:
Design Technology: Children's Engineering

> Children and teachers need to be actively engaged in the learning process. As designers of their learning, dynamic collaboration between adult and child produces thoughtful curriculum.
> Susan Dunn and Rob Larson
> *Design Technology: Children's Engineering*

Imagine a book filled with sage advice and ideas about the design process that goes on with any LEGO or Logo project. While at the National Educational Computing Conference in Nashville last June, Linda Polin of Pepperdine University shared such a book with me. I found it so relevant to working with Logo and with *LEGO TC Logo* that I decided to make it the required text for the LEGO-Logo Institute I taught at Long Island University's C.W. Post campus last August.

Written by Susan Dunn and Rob Larson, *Design Technology: Children's Engineering* does not mention LEGO or Logo, save in its list of resources, and yet the philosophy and the processes the authors discuss mesh beautifully with work in a Logo or LEGO-Logo environment. (There is one discussion of an activity that sounds much like the Soap Box Derby with LEGO-Logo, however.) The fact that LEGO and Logo are not mentioned specifically actually is an advantage. It allows Logo-using teachers to make their own associations of ideas as they apply to children's projects. In fact, the authors say explicitly that they do not intend to present a list of classroom ideas or techniques, but rather to add "to the repertoire of useful strategies for orchestrating design technology experiences." They challenge *us* to make the applications.

The Approach

Organized in a sequence that mirrors the design process, the authors deal sensitively with each topic. The initial chapter, "Spirit and Context," takes a Logo-like approach to children's design. Key phrases in this chapter include "active exploration," "cognitive constructivism," "participatory curriculum," "negotiated curriculum," and "child-centered approach." Sounds familiar, doesn't it?

The chapter on "Exploration" sets the stage for what follows. The authors divide the design technology process into four major categories that continue to recur as any project unfolds: investigation, invention, implementation, and evaluation. There is an excellent page outlining the process of reflection. This page includes key words and related questions. I suggested that my LEGO-Logo students at Long Island University use this page as a guide when writing their reflections on their own learning and design processes during the course. The most perceptive reflection turned out to be the one that used the outline on the reflection page as their guide.

Communication

Communication skills are taken as seriously as design and building. An excellent case is made for keeping journals. (What Logophile has not at least considered the use of journals in his or her classes?) Keeping a journal joins the engineering process with the development of language skills. In her journal the student can document the learning of new terms, organize and evaluate information gathered for the project, record the steps in the design process, document her project, and write descriptively about her design experience. The journal is, at the same time, building a fall-back collection of ideas to use if a problem seems insurmountable.

Problems of representation of the project in two-dimensional form are included in the discussion of the communication process. Some excellent drawings in the book detail a variety of methods for representing objects two-dimensionally: isometric drawing, orthographic projections, one-point perspective, oblique drawings, and use of a grid underlay. All these are contained in crisp graphic pages with wonderfully clear illustrations including brief definitions of terms—a wonderful source for a lesson or two that move in a slightly different direction. These tangential activities provide students with a variety of methods for making a visual presentation of their design.

There are sensitive discussions of all the pieces of the design process, from the initial exploration through the process of collaboration, the choreography (this is especially useful for teachers as they plan such projects), connections (setting contexts for design projects), and celebration (giving due attention to all aspects of the process and reflecting as the work is completed). One of the final chapters includes discussions of several design experiences. There are many italicized sequences of leading questions included within the text that relate to parts of the process. These questions are useful for us to consider and are equally useful to pose to children to prod them to think actively about their work.

Relevant Quotations for LEGO and Logo Users

As I read the book, many relevant comments stood out. I reflected on the work I have done with children and motion with Logo and read appreciatively:

> In children's engineering, then, understanding science comes from investigating related technology. Rather than requiring a complete scientific understanding prior to applying... a child learns...concepts in investigating their applications.

Or, in another chapter, how about this:

> A sensitively-timed suggestion, in the form of a question or demonstration, may nudge a child past a construction problem.

...or this:

> Timing is crucial. In identifying successful design resolutions prematurely, the resultant effect of praise is to stop the production of diverse ideas and limit the scope of solutions. Noticed after children have a chance to explore a range of ideas, sharing can provide opportunity for *exchange* of ideas and comparison or results.

This is a remarkable book, lavishly illustrated with color photographs and supplemented by Susan Dunn's delightfully clear graphic illustrations. The graphics are full of clever engineering ideas and would provide a wonderful resource for students. In addition, little side boxes appear occasionally throughout the text, filled with truly apt quotations that will probably, if you're like me, lead you to other reading! For example, this one is from Marilyn Ferguson's *The Aquarian Conspiracy*:

> The open teacher, like a good therapist, establishes rapport and resonance, sensing unspoken needs, conflicts, hopes, and fears. Respecting the learner's autonomy, the teacher spends more time helping to articulate the urgent *questions* than demanding right answers.

What an apt quotation indeed for a Logo environment!

Idea Number Two

LogoExpress Is Here!

For the past year I have had the privilege of testing *LogoExpress*, a telecommunications version of *LogoWriter* which is now available from Logo Computer Systems, Inc. This is not simply another telecommunication package, but rather a Logo-based telecommunication package with all the openness one expects from a Logo environment. With *LogoExpress* it is possible to exchange electronic mail, to initiate on-line "chats" or conversations, to set up your own electronic bulletin board, to transmit any kind of file, and even to exchange complete *LogoWriter* pages, both text and graphics. You can even send shapes pages.

What is *LogoExpress* like? It's like *LogoWriter*. The user interface looks familiar. Load the disk, and you see a *LogoExpress* screen. Press Return and you'll see the familiar Contents page. Although the graphics are missing (turtles, shapes, colors), most of the primitives are exactly the same as the *LogoWriter* primitives. A class of primitives designed especially for use with telecommunication has been added. The text is in 80 column format instead of *LogoWriter's* 40 columns.

The "**mailer**" page contains its own instructions on how to set up the page for the electronic bulletin board you want to call. You generally need only change the setup procedure: add your username, your password, and change the phone number to the correct number for the electronic bulletin board to which you connect. Once these changes are made, it's easy to name the page (just as in *LogoWriter*), clear the text, and lock the page.

I currently dial in to two bulletin boards provided by Logo Computer Systems. One is in New York and one is in Montreal. I have a **newyork** page and a **montreal** page set up on my disk. I simply adjusted the setup procedure on the **mailer** page for dialing New York, then named the page **newyork**. I cleared the text and then locked the page by typing **lock**. I repeated the process to create a **montreal** page. I simply press the Tool Keys on the page name of the board I want to call.

If all you wish to do is check for mail, a single word (it's a Logo procedure), **checkmail**, will dial the appropriate phone number, log you in, check for and collect your mail, and then hang up. If you want to mail a letter, first write it on the page and then type **mailthis "username** in the Command Center. The "username" is the name by which the person you are mailing to is identified in the system you wish to contact, called the host system. The **mailthis** command will dial, log you in, send your mail, and then hang up.

If you prefer to send an entire *LogoWriter* page or write a message on a separate page, create your page and message and then return to your mailing page. Then use these commands (use the name of your page and username of the person you are sending to):

```
mail "page.to.send "username
```

When you have several tasks to accomplish in a single call, you can use a few other terms. For example, to dial, log in, mail a letter, check for mail and hang up, type:

```
login mailthis "michaelt checkmail
   hangup
```

The system will be dialed, the letter written on your mailer page will be mailed to "**michaelt**", your mail will be collected, and the computer will hang up for you. If you want to mail just one part of a page, select the appropriate text on that page and the **mailthis** command will act only upon the selected text. This is useful if you want to write several messages that are to go to different people. You can select and mail each in sequence, but...

Since It's Logo....

Eventually you may find that there are processes you perform repeatedly. You can write your own procedures and add them to your **mailer** page. For instance, sometimes when I'm busy and the host is busy also, it's handy just to have a control key to **checkmail** or even to log in and mail a letter, as above. You can add these to your setup procedure on your mailer page (but don't forget to unlock and then lock the page to save your additions):

```
when "z [checkmail]
when "p [login checkmail checkpost
   hangup]
```

As you notice other operations you perform routinely, add procedures to do these also. If you send a lot of messages at once, all on different pages, or a lot of text files, you will want to write a procedure to send a list of pages for you. Add that to your **mailer** page as well.

There are some other nice features about *LogoExpress*. As with GS and MS DOS versions of *LogoWriter*, the Contents page alphabetizes itself so that finding the pages you want is a lot easier. When you receive mail from a host system bulletin board, your mail is automatically saved as a text file called "**mail.**" That page is updated each time you receive mail, so you need to decide whether to print or save your mail under a different name before you use the system again. Pages received on an Apple from an MS DOS computer (or on MS DOS from Apple) show up as "et" (meaning external type) files: you can read the text but the graphics will only be available on the *LogoWriter* version in which the file has been sent. The eighty-column text is a great feature if you're used to writing a lot, as I am.

What Can You Do with *LogoExpress*?

What use is *LogoExpress*? I have used it only minimally so far with students, but find that it is easy for anyone to understand how to use it.

As an electronic mail system, *LogoExpress* is extremely flexible and has allowed me to keep in easy contact with colleagues from all over with greater ease than by phone. A piece of electronic mail that I send sits in their electronic mailbox until they collect it. I don't need to dial over and over again. I find I am redeveloping the habit of writing letters now that I have *LogoExpress*, and that I am in more frequent contact with my far-flung colleagues.

LogoExpress also has a "terminal" mode so that you can use it as a standard telecommunications package if you aren't interested in customizing a special bulletin board.

LogoExpress comes with a quick reference card, a reference guide, an instruction booklet and a book of project ideas. I can imagine that during the year we will come up with many additional projects. I recall Jandy Bird's delightful article last year about a student exchange—by mail—and think what fun it would be to do the same electronically!

So What Services Are Available?

Logo Computer Systems maintains two hosts: one in New York and one in Montreal. Both are open to the public at absolutely no charge. These host systems have private "mail boxes" for individuals as well as one or more public "bulletin boards."

The New York host has a several bulletin boards, including one called "workshop," which was created last summer when there were workshops in St. Paul, Minnesota, and Hartford, Connecticut. The workshop board served as a general communication system between the two workshops. I was in St. Paul for one of the two workshops held there. We used the system several times a day to send communications to Hartford and to check on the postings on the bulletin board which we downloaded, printed and hung on the wall for more leisurely reading. At St. Paul we kept the computer on all of the time with the **mailer** page cleared. People added messages at their leisure, then sent them either individually or as a batch to be posted on the workshop board. During the workshop, a number of participants managed to plan "connections" for their classes for the school year. The "workshop" board will remain on the New York host and may be a good place to start communicating.

The New York host also has a "tech" board for technical support and a "sales" board for ordering information. All this costs is a phone call—your mailbox is free.

Getting Your Own Mailbox

If you want to have a mailbox on either system, follow the instructions on your mailer page to set up with the user name "guest" and the password "logo." The number for the New York board is 1-212-765-4924; the Montreal number is already on your disk. The New York-based LCSI host can be accessed via either 1-818-505-1511 (Los Angeles) or 1-212-765-4924 (New York). If you're dialing out through a switchboard, precede that number by a 9 and a comma (this causes a pause usually long enough for you to get a dial tone). And yes, you can leave out the dashes.

Now turn on your modem, type **login,** and press Return. When you're logged in—and you'll see messages in the Command Center as this is done—type **checkpost** to get the postings on the main board. You'll want to retrieve the first posting for instructions on how to get your own mailbox. Type

```
getpost 1
```

You will also want the user list for future reference. To contact someone on the board, you need to know their username. Follow the form above to get that posting. Asking for that posting will cause the host to send you a page with all of this information. You will see it on your disk as **ny.userlist**.

To get to the workshop board on the New York host, type

```
join "workshop checkpost
```

This will get the list of postings on the workshop board. They are listed by number, date, a short title, and name or place of sender. Ask for the postings you want, as above. If you want more than one posting, **getpost** will accept a list of numbers. For example,

```
getpost [1 6 10]
```

To get back to the main board, simply type

```
join "bbs
```

To hang up, type

```
hangup
```

Try sending someone a page, just for fun. Exchange project ideas with colleagues. Set up electronic penpals for your students. Try an electronic cultural exchange with *LogoWriter* illustrations of "Life and Times in....." As we think about sharing actively and writing and creating for "publication" think of the possibilities of electronic sharing of children's work. Consider using *LogoExpress* to broaden the world for you and your students.

Useful Information

LogoExpress is available for MS DOS and 128K Apple computers from Logo Computer Systems, Inc. It comes with a program disk, either 3.5" or 5.25" format, with several pages set up for you. There is also a quick reference, a reference guide, a *LogoExpress* project book, and a user guide. *LogoExpress* requires a Hayes or Hayes-compatible modem and can operate at 300, 1200, or 2400 baud. A single computer version is available for $99.00. A single cpu host version, with the software to allow you to set up your own board, is $134. A building site license, allowing you to use *LogoExpress* at several computers, is $329.00 with a 10% discount to LogoWriter building site license holders. There's even a District License for a series of schools in a district, a great deal if you're purchasing anything more than a single host and single access. This costs $599.00 plus $159.00 per participating site and comes with extra booklets and quick reference cards for each site. Add 5% shipping and handling to these costs. Logo Computer Systems, Inc. can be reached by phone at 1-800-321-LOGO.

Design Technology: Children's Engineering, by Susan Dunn and Rob Larson, was published in 1990 by The Falmer Press, a division of Taylor & Francis, Inc., 1900 Frost Road, Suite 101, Bristol, PA 19007. ISBN: 1-85000-590-7. Taylor & Francis can be reached at 1-800-821-8312. They will charge the $18.00 cost to a major credit card and ship via UPS. My copy arrived in just three days!

Eadie Adamson
The Dalton School
108 East 89th Street
New York, NY 10128

On both LCSI bulletin boards my username is EadieA.
I'd love to hear from you electronically!

Logo LinX

DisSPELLing a Myth

by Judi Harris

My students took spelling tests on words that they hadn't studied or memorized.

No, it wasn't cruel and unusual punishment. It was a different way to learn to spell. And, considering that they spelled most of the words (that they hadn't studied) correctly, I'd be willing to wager that either they learned to spell without memorization, or they were excellent spellers already.

If you had seen their writing in September, you would have known for sure that they did *not* begin the school year as national spelling champions. Yet they improved their spelling in a Logo-like way.

A Curricular Catalyst

"So what's so Logo-like about spelling?" you ask.

The answer is simple: *patterns*. Students can learn to spell better by recognizing and applying letter patterns, instead of memorizing 20 new words each week.

AIRS (Andover's Integrated Reading System), a mastery learning language arts program, contains a spelling module called "Structural Skills" that helps children to spell in this way. It is based upon the generalizations of eight structural skills: plurals, derived words, possessives, contractions, root words, hearing syllables, syllabication, and prefixes/suffixes.

With the AIRS program, students first learn a spelling *rule* by examining and discussing words that follow the same spelling *pattern*. For example, what common pattern do you see in the spellings of these root words?

START
BUZZ
CRISP
BLINK
LAUGH

All of them end with two consonant letters. If we wish to add a suffix, such as -s, -es, -est, -ed, or -ing to any of them, we can do so without altering their spellings.

STARTED
BUZZES
CRISPS
BLINKED
LAUGHING

The same is true for these words:

APPEAR
SEAM
COOL
TOOT
NAIL
BROAD

although they follow a different pattern. What is it?

Once students can recognize a spelling pattern and remember the rule that accompanies it, they can spell most words that they hear. There are a total of 39 structural skill rules in the AIRS spelling materials used in grades 3 - 8. That's fewer than two weeks' worth of things to remember, according to traditional spelling instructional methods.

DisSPELLing can be fun!

Paula Cochran and Glen Bull's "spelling fuzzies" ("SpecialTalk," *Logo Exchange*, October 1986) are a wonderful way to reinforce correct spelling using Logo. The AIRS materials inspired thoughts about some interesting ways to help insure correct spelling by attending to letter patterns.

Each of the words in the first sample (above) ended with two consonants. Let's see how to tell Logo to check a word for that pattern.

First, we need a procedure to tell if a letter is a consonant. One way to get the computer to determine this is to ask it to check to see if a certain letter is *not* a vowel.

```
TO CONSONANT? :LETTER
OUTPUT NOT VOWEL? :LETTER
END

TO VOWEL? :LETTER
OUTPUT MEMBER? :LETTER [A E I O U]
END
```

To see how the CONSONANT? procedure works, type, for example,

```
PRINT CONSONANT "L
```

The computer prints:

```
true
```

The following procedure uses CONSONANT? as a subprocedure to check for consonants as the last two letters of a word.

```
TO TWO.END.CONSONANTS? :WORD
OUTPUT AND CONSONANT? (LAST BUTLAST
   :WORD) CONSONANT? (LAST :WORD)
END
```

Now you can type:

```
PRINT TWO.END.CONSONANTS? "TURTLE
```

and the computer will return:

```
false
```

Impelled to Spell

Did you figure out the second sample letter pattern mentioned above? All of those root words have two *vowels* appearing together, forming a vowel blend. Here is a procedure that uses the VOWEL? subprocedure to check recursively a word of any length to see if it has two vowels appearing in succession.

```
TO VOWEL.BLEND? :WORD
IF (COUNT :WORD) < 2 [OUTPUT "FALSE]
IF AND (VOWEL? FIRST :WORD) (VOWEL?
   FIRST BUTFIRST :WORD) [OUTPUT
   "TRUE]
OUTPUT VOWEL.BLEND? BUTFIRST :WORD
END
```

Now let's make the computer generate some root word and ending combinations that follow the vowel blend pattern.

```
TO DERIVED.WORD
OUTPUT WORD (PICK TWO.VOWEL.WORDS)
   (PICK VERB.ENDINGS)
END

TO PICK :OBJECT
OUTPUT ITEM (1 + RANDOM (COUNT :OB-
   JECT)) :OBJECT
END

TO TWO.VOWEL.WORDS
OUTPUT [COOL MAIL NAIL WEED TOAST
   BREAD LEAP GROAN SEED LOAD]
END

TO VERB.ENDINGS
OUTPUT [S ED ER ING]
END
```

Type

```
PRINT DERIVED.WORD
```

and the computer might return NAILED or BREADING or GROANING. If you enter

```
REPEAT 5 [PRINT DERIVED.WORD]
```

what do you think will happen?

Pattern Power

Now, let's combine these procedures into a superprocedure that will help students to recognize and extend spelling patterns.

```
TO PATTERNS
HT
CT
PRINT [Here are some words:]
PRINT TWO.VOWEL.WORDS
PRINT.BLANK.LINE
PRINT [Please type a root word that
   follows the same pattern.]
PRINT [(If you need a hint, type
   HINT.)]
MAKE "NEW.WORD READWORD
IF :NEW.WORD = "HINT [PRINT [V V]
   MAKE "NEW.WORD READWORD]
IFELSE VOWEL.BLEND? :NEW.WORD [PRINT
   [Great!  Same pattern!]]
   [TRY.AGAIN]
END

TO PRINT.BLANK.LINE
PRINT CHAR 13
END

TO READWORD
OUTPUT FIRST
READLISTCC
END

TO TRY.AGAIN
PRINT.BLANK.LINE
PRINT [That's another spelling pat-
   tern.  Try typing a different root
   word.]
PRINT.BLANK.LINE PRINT [Press any
   key...]
IGNORE READCHAR PATTERNS
END

TO IGNORE :KEYPRESS
END
```

I encourage you to write a similar set of procedures that utilize the two-final-consonants pattern. And then, how about plurals? Words that end in silent e? Contractions?

Yes, 'dis spelling *can* be fun.

For more information about AIRS Reading and Language Arts materials, write to:

AIRS
Andover Public Schools
Bartlett Street
Andover, MA 01810

An earlier version of this article
was originally published in the
April 1987 issue of *Logo Exchange* magazine.

Judi Harris works in the Department
of Teacher Education at the
University of Nebraska, Omaha
as an assistant professor of educational technology.

BITNET: JHarris@UNOMA1
Internet: JHarris@Zeus.unomaha.edu
CompuServe: 75116,1207

Logo Linguistics: What to do with those "silly sentences"

by Mark L. Evans

Using computers to generate sentences from word lists is not a new idea, and both teachers and students have had fun examining the silly, or often absurd, results (e.g., "Seals wear wigs." "Horses play with bridges." "Mountains read books."). However, if we take a more serious approach to how and why we are using a computer to generate these kinds of sentences, we may discover a new tool in helping us teach grammar. (Though grammar consists of phonology, morphology, and syntax, I use it in this article to mean mainly syntax.)

Jim Sabol from Seattle Pacific University has proposed teaching writing to students by teaching certain kernal sentence types. He argues that teaching the underlying sentence structure enhances students ability to manipulate our language in producing their own written communications.

Sabol has argued that we should examine the syntax of sentences and break them down into kernal sentences for examination and manipulation. The kernal sentences he identifies are:

Type I (one noun and one verb)
Fish swim.
Birds fly.
Dogs bark.

Type II (a noun, a verb, and a noun)
Cats catch mice.
People write letters.
Dogs chew bones.

Type III (a noun, a "linking" verb, and either an adjective or a noun)
Houses are big.
Cars are vehicles.

In the type I sentence, the verb is such that the action "stops" after the verb. The subject noun is doing something, but not doing something to some other thing.

On the other hand, type II sentences have action verbs in which the subject is doing something to another thing, namely to the predicate noun. Occasionally we can get by with leaving out the predicate noun because it is understood. For example: Birds sing (a song).

Type III sentences have verbs, often called "state of being" verbs, that link the subject to the predicate. These sentences have the predicate noun or predicate adjective describing something about the subject.

Notice that all the example sentences are very basic with no determiners, and the subject nouns are all plural to avoid having to supply a determiner. Of course as understanding and sophistication are developed by students, these grammatical elements (and others) are added to generate gradually more complex sentences.

With these three basic kernal sentences, people can create many complex sentences through combinations. Other syntactic elements (parts of speech) can also be added to "dress up" the sentence. For instance, adjectives can be added to dress up the nouns, and adverbs can be added to dress up the verb. (See Note 1.)

Determiners are needed if we choose a singular subject: The bird flies. (A helpful rule Sabol taught us is that we have only one "s" to spend. Either we spend it on the subject to make it plural, or we spend it on the verb if the subject is singular.)

It takes students years to fully grasp the structure of our language, but in having these kernal sentences as the basis, they can begin to build their own understanding of how our language is put together. And for those of you who have attended Sabol's workshops, the writings of students he has taught attest to the usefulness of providing them with these basic tools. Progress is slow but steady in teaching these sentence types in the elementary grades. The product is well worth the effort.

Putting Logo to Work

Combining Sabol's approach with Logo can allow students to explore the grammar of our language and develop their own understanding of sentence structure. Borrowing from the Judy Harris article (*Logo Exchange*, Feb. '87) on sniglets, we can create our own lists of nouns, verbs, determiners, and other parts of speech.

```
TO STARTUP
MAKE "NOUNS [CATS DOGS BIRDS HORSES
   SEALS]
MAKE "VERBS [GO THINK JUMP RUN EAT
   SEE FLY]
END
```

Then, adding the procedures NOUN, VERB, and the PICK tool, we have the listing for the MAKE.SENTENCE program:

```
TO NOUN
INSERT (WORD PICK :NOUNS)
INSERT CHAR 32
END

TO VERB
INSERT (WORD PICK :VERBS)
INSERT CHAR 32
END

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

Using Logo with the MAKE.SENTENCE program, the computer can generate type I, II, and III sentences for us. (Three different MAKE.SENTENCE programs are listed at the end of this article, and each corresponds with the three sentence types. You should have the appropriate version loaded into Logo to produce the desired sentence type.) The structure of each sentence type needs to be specified in the Command Center. Typing NOUN VERB will create a Type I sentence without any end punctuation. NOUN VERB P.NOUN (see Note 2) will create a type II sentence without punctuation. Repeat the above commands or type REPEAT 10 [NOUN VERB] to have the computer put 10 sentences on the screen. Here are some sample sentences that may be generated.

Type I sentences

Cats think
Birds jump
Dogs go
Cats go
Horses think
Birds look
Cats run
Birds eat
Dogs jump
Horses go

Type II sentences

Birds hit bed
Cats reach wall
Seals hit food
Dogs hit wall
Cats see wall
Dogs see playground
Birds hit computer
Horses jump computer
Dogs reach wall
Cats reach playground

These sample sentences can serve as a basis for classroom discussions. (You may wish to refer to the objectives, below, before continuing.) A general structure for these discussions may be:

1. Generate the sentences.
2. Discuss the grammar or structure of the sentences.
3. Discuss the semantics of the sentences. (See Suggested Activities).

Are the sentences grammatically complete? Are the sentences semantically accurate? Why or why not? A distinction must be made here between grammatically complete (i.e. having all the necessary elements), and being semantically accurate (i.e. making sense or making realistic sense). Such discussions should lead students to deeper understanding of not only the structure of our language, but also the semantic limits placed on individual words or categories of words. This in turn will lead to the main objectives of this program: determining categories of nouns and verbs that will produce a minimal number of semantically faulty sentences; and writing procedures for these categories to be included in the program.

Notice that punctuation has not been included so far. Students will need to observe this missing element and suggest provisions for its inclusion. Also, the subject nouns are initially all plural. This avoids having to supply a determiner, for which students must again suggest provision. Such suggestions may look like this:

```
TO PUNC
MAKE "PUNCT [. ! ?]
INSERT (WORD PICK :PUNCT)
PRINT []
END

TO DET.
MAKE "DETERMINERS [A THE SOME]
INSERT (WORD PICK :DETERMINERS)
INSERT CHAR 32
END
```

Students will soon notice a need to include other syntactic elements. For example, adjectives need to be included, and this observation should be enough fuel to get students started writing lists and procedures for adjectives. ADVERBS is

Logo Linguistics - continued

another category to explore. A warning is in order here: These two categories are very complex, and you may wish to guide students' initial attempts toward single-word adjectives and single-word adverbs. Placement of these two categories inside sentences should prove revealing. Adjectives come before nouns (as students should discover through exploration) except in type III sentences where they are placed in the predicate. Adverbs can be placed in many locations throughout the sentence and still be grammatical. Why is this so? Why can we not move adjectives around as we do adverbs?

As additional lists of speech parts and their procedures are included, the computer will be able to generate as many sentences as needed for close examination and discussion. The process of refinement will continue as students modify their lists, keeping in mind the main goal of producing a minimal number of semantically faulty sentences. Of course, some students will find delight in making lists that produce a maximum number of faulty sentences, and some will experiment with jumbling the syntax to produce meaningless sentences. These are valid explorations if they can serve as a basis for developing a deeper understanding of the structure of our language.

Conclusion

It seems to me that this MAKE.SENTENCE program can be a delightful way to play with our language while learning its structure. Some students may wish to take their learning beyond the computer by using the computer generated sentences in their own writing assignments. Indeed, Sabol would encourage this activity, and so would I.

Modifying the lists to fit predetermined needs for sentence structures would be the main evidence that students are actively engaged in using and understanding English grammar. Our language textbooks have been used for this purpose with much difficulty (even torture!) in the past. Perhaps sentence building with Logo can help put some student interest back into grammar study.

Objectives for Logo Linguistics

Given the program(s) MAKE.SENTENCE running in Logo on an Apple IIe computer, or given a list of sentences created by the MAKE.SENTENCE program(s), students will:

1. Analyze the computer generated sentences to determine if each one is grammatically complete.
2. Analyze the computer generated sentences to determine if each one is semantically accurate.
3. Name elements necessary to make grammatically complete sentences (e.g., noun, verb, punctuation, determiner—depending on subject count).
4. State reasons why given sentences are semantically faulty (e.g., non-living objects do not think).
5. Create or modify noun and verb categories to produce more semantically accurate sentences (e.g., create ACTION.VERBS for type II sentences, or PREDICATE.NOUNS for type II sentences).
6. Create new categories for other elements of speech such as ADJECTIVES or ADVERBS.
7. Write procedures to add to the MAKE.SENTENCE program that will include the new or modified categories from objectives #5 and #6 above.
8. Choose sentences and include them in written compositions.
9. Determine the percentage of semantically faulty sentences produced by the computer with MAKE.SENTENCE.
10. Rate given sentences from 1 to 5 on an "absurdity scale."

Suggested Activities: MAKE.SENTENCE.1

These activities should probably be presented initially in a whole group setting with the teacher controlling the computer. A large monitor or PC viewer is suggested for reasonable viewing by the students.

1. Begin by discussing a type I sentence. It has a noun and a verb as its basic structure. Point out that the verb can stand by itself because the action "stops" after the verb.

Show the students some sample sentences that the computer can generate. Type this into the Command center:

```
REPEAT 10 [NOUN VERB]
```

What happened? (Output is a continuous string—one long sentence.)

What is needed in each sentence? (Punctuation.) Let's try again with punctuation:

```
REPEAT 10 [NOUN VERB PUNC]
```

Look at the sentences. Are they grammatically correct? Which ones are not? What's wrong with those?

Discuss grammar vs. semantics: grammatically correct means it has a subject and predicate, has correct punctuation, and has subject/verb agreement. Semantically accurate means is makes sense.

Add specific punctuation to individual sentences by typing NOUN VERB . or NOUN VERB ! in the command center. The structure of questions is beyond this microworld at this time, so you may have to eliminate questions for the time being. You may also wish to have students rewrite the sentences correctly on paper.

2. **What do you notice about all the nouns?** (Plural.) When we speak, are they all plural?

We need to make a new list of singular nouns. Flip the page and look at the procedure STARTUP to see the present list of plural nouns. Have the students decide what should be included in the singular nouns list.

What should we call our new list? (How about S.NOUN?)

Flip the page and generate new sentences:

```
REPEAT 10 [S.NOUN VERB .]
```

What's wrong with these sentences? (Ungrammatical— verb needs an "s," and determiners are needed.)

How should we fix them? (Make lists of singular verbs and a list of determiners.)

Flip the page and make the new lists.

Generate new sentences:

```
REPEAT 10 [DET. S.NOUN S.VERB .]
```

How do they look now? Review making sentences with plural subjects. Point out that plural or singular subjects can be generated now, but you must specify the verb to agree with the count of the subject. Discuss the Sabol rule about spending the "s" on the subject or spending it on the verb.

You may now wish to have students write for 10-15 minutes and have them include some of these sentences in their writing.

Suggested Activites: MAKE.SENTENCE.2
Begin by discussing the structure of a type II sentence— it has a noun, a verb, and a noun.

Type NOUN VERB NOUN in the command center and ask: Is this enough? (No, you need punctuation also. You might also need a determiner.)

Type:

```
REPEAT 10 [NOUN VERB NOUN .]
```

Are the sentences grammatical? Are they semantically accurate? Which ones are or are not? Why?

Discuss the nature of the verb in this type of sentence. It "carries the action" from the subject to the predicate noun. Since the predicate noun is being acted upon, you may need a new category of noun for that.

Flip the page and make a new category of predicate nouns (P.NOUNS?). You should guide the students toward including inanimate objects initially.

Generate new sentences:

```
REPEAT 10 [NOUN VERB P.NOUN .]
```

Now we have an animate subject doing something to an inanimate object. Are the sentences more grammatical? Are they more semantically accurate?

Add a determiner, if you wish, and you may wish to transfer the singular subjects from the MAKE.SENTENCE.1 listing. Be cautious of the verbs in MAKE.SENTENCE.1, because they "stop the action" and do not carry it to a predicate. You may wish to make a new S.VERB list for these type II sentences.

```
REPEAT 10 [DET. S.NOUN S.VERB
    P.NOUN .]
```

Have students correct ungrammatical or semantically faulty sentences. Have them state reasons why the sentences are ungrammatical or semantically faulty.

You may wish to have students write for 10-15 minutes and try to include some of the computer generated sentences in their paragraphs.

Suggested Activities: MAKE.SENTENCE.3
This is probably the hardest type of sentence to deal with

Logo Linguistics - continued

thus far because these sentences make assertions, give definitions, or describe something. Therefore, there needs to be more semantic accuracy or there will be more dispute. Also, these are the types of sentences that many people react to most strongly when the sentences are ungrammatical, because it is not a simple matter of adding or dropping an "s." Instead the verb form changes, so that lack of agreement is more apparent.

Begin by discussing the structure of type III sentences. These have a noun, a linking verb, and either adjective or another noun. Note that the verb is very different in these sentences. The verb "links" the subject and predicate together. Therefore the predicate describes or tells something more about the subject.

We will have to call the type III verbs "linking verbs" and will use L.VERB in our command center.

```
REPEAT 10 [NOUN L.VERB ADJ .]
```

What is wrong with these sentences? (Many have the wrong verb form, mainly due to lack of agreement with the subject.)

How can we fix it? (Specify the count of the subject and the verb, as we did in the other two types of sentences.)

Make list on the flip side of S.NOUNS and S.L.VERB — singular linking verbs. Now generate new sentences.

REPEAT 10 [S.NOUN S.L.VERB ADJ .]

Are these better sentences? (In general, yes.)

What's wrong? (The S.L.VERB "am" is only used with "I"; therefore you may have to get rid of it in the list.)

How about nouns in the predicate? Let's try.

```
REPEAT 10 [NOUN L.VERB P.NOUN .]
```

What's wrong? (Same as before, subject/verb agreement)

Let's try again.

```
REPEAT 10 [S.NOUN S.L.VERB P.NOUN .]
```

What's wrong? (Generally, they are semantically inaccurate, asserting something is something it couldn't be.)

How can we fix it? (Control the relationship between the subject noun and the predicate noun.)

This will amount to fairly narrowly defined lists that relate to given nouns. You could have students make these relational lists on paper by answering these questions:

How would you describe this thing (a given noun)?

What are some other names for this noun?

These list are the raw data (Sabol calls them "word caches") students could use to write with. You may want to plan a 10- to 15-minute writing activity using the lists and/or sentences.

Note 1: When adding adjectives or adverbs, we can add either one word (blue—adj., quickly—adv.) or we can add a phrase (completely blue—adj., in the morning—adv.). The analysis of phrasal adjectives and adverbs is quite complex because of the many kinds of adjectives (color, number, size, etc.) and many kinds of adverbs (manner, time, completeness, etc.). That analysis is beyond the scope of this article.

Note 2: "P.NOUN" stands for "predicate noun." This refinement is necessary to produce more semantically accurate sentences. Students should discover this as they develop their own lists. See "Suggested Activities for MAKE.SENTENCE.2" for a better understanding.

Program Listing: MAKE.SENTENCE.1

```
TO STARTUP
MAKE "NOUNS [CATS DOGS BIRDS HORSES SEALS]
MAKE "VERBS [GO LOOK THINK JUMP RUN EAT SEE]
END

TO NOUN
INSERT (PICK :NOUNS)
INSERT CHAR 32
END

TO VERB
INSERT (PICK :VERBS)
INSERT CHAR 32
END

TO PUNC
MAKE "PUNCT [. ! ?]
INSERT (PICK :PUNCT)
PRINT []
END

TO PICK :LIST
OUTPUT ITEM 1 + (RANDOM COUNT :LIST) :LIST
```

```
END

TO DET.
MAKE "DETERMINERS [A THE SOME]
INSERT (PICK :DETERMINERS) INSERT CHAR 32
END

TO .
PRINT [.]
END

TO !
PRINT [!]
END
```

Program Listing: MAKE.SENTENCE.2

```
TO STARTUP
MAKE "NOUNS [CATS DOGS BIRDS HORSES SEALS]
MAKE "VERBS [JUMP EAT SEE HIT REACH MISS]
END

TO NOUN
INSERT (PICK :NOUNS)
INSERT CHAR 32
END

TO VERB
INSERT (PICK :VERBS)
INSERT CHAR 32
END

TO P.NOUN
MAKE "PRED.NOUNS [WALL HOUSE PLAYGROUND FOOD
    BED FENCE COMPUTER]
INSERT (PICK :PRED.NOUNS) INSERT CHAR 32
END

TO PUNC
MAKE "PUNCT [. ! ?]
INSERT (PICK :PUNCT)
PRINT []
END

TO PICK :LIST
OUTPUT ITEM 1 + (RANDOM COUNT :LIST) :LIST
END

TO DET.
MAKE "DETERMINERS [A THE SOME]
INSERT (PICK :DETERMINERS) INSERT CHAR 32
END

TO .
PRINT [.]
END

TO !
PRINT [!]
END
```

Program Listing: MAKE.SENTENCE.3

```
TO STARTUP
MAKE "NOUNS [HOUSES CATS PEOPLE CARS COMPUT-
    ERS TOOLS ZEBRAS]
MAKE "LINK.VERBS [IS ARE WAS WERE AM]
MAKE "ADJ. [BLUE BIG FUN TERRIBLE ANCIENT
    STRIPED HAPPY WHITE CRAZY]
MAKE "PRED.NOUNS [ANIMALS HUMAN AUTOMOBILES
    INSTRUMENTS HOMES FELINES]
MAKE "DETERMINERS [A THE SOME]
MAKE "PUNCT [. ! ?]
END

TO NOUN
INSERT (PICK :NOUNS)
INSERT CHAR 32
END

TO VERB
INSERT (PICK :VERBS)
INSERT CHAR 32
END

TO P.NOUN
INSERT (PICK :PRED.NOUNS)
INSERT CHAR 32
END

TO ADJ.
INSERT (PICK :ADJ.)
INSERT CHAR 32
END

TO DET. I
INSERT (PICK :DETERMINERS)
INSERT CHAR 32
END

TO PUNC
INSERT (PICK :PUNCT)
PRINT []
END

TO PICK :LIST
OUTPUT ITEM 1 + (RANDOM COUNT :LIST) :LIST
END

TO .
PRINT [.]
END
```

Mark Evans in an elementary teacher in Springfield, Oregon. He has been teaching 3rd and 4th grade for thirteen years. Last year, during a sabbatical leave, he first became interested in Logo. Mark is supported by a wonderful family; his interests include research, learning, carpentry and current affairs. He can be reached at 297 Park Avenue, Eugene, OR 97404.

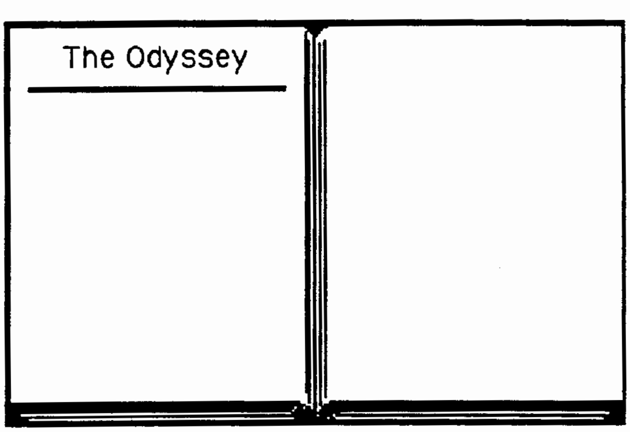
Logo Connections

Imported Graphics
by Glen L. Bull and Gina L. Bull

Logo Connections is about ways in which other types of hardware and software can be used in combination with Logo. In this year's columns we are discussing practical ways to create a multimedia environment with Logo. Before you purchase any hardware or peripherals, the most important enhancement of all consists of an upgrade to the latest version of Logo. Both *LogoWriter* and *Logo Plus* have excellent characteristics that lend themselves to a multimedia environment.

Acquisition of a mouse is another desirable step toward a multimedia environment. A mouse is a pointing device used by many graphics programs. Mice are standard equipment on the new IBM PS/2 computers, the latest Apple IIGS computers, and, of course, the Macintosh. If you have an Apple IIGS computer or a Macintosh, you already have a mouse. If you have an Apple IIe or IBM computer without a mouse, you can easily add one to your computer. A mouse will allow you to easily edit digitized pictures and clip art, and create original pictures that can be imported into Logo.

In last month's column, we discussed development of a hypermedia adventure story with *LogoWriter*. The pages of *LogoWriter* lend themselves to the metaphor of the pages of a book. In the hypermedia adventure described, a background graphic was used to reinforce the impression of pages turning in a book.



Background Graphic for Logo Hypermedia Adventure Story

This graphic could be created with turtle graphics, and some might find creation of the illustration an interesting exercise in Logo programming. However, there is no specific instructional advantage to creation of this particular illustration with turtle graphics. Since the illustration can be constructed in a fraction of the time with a paint program, and then imported into Logo, this alternative warrants consideration. As an added bonus, many paint programs offer a variety of fonts, such as the one used to create the title of "The Odyssey" shown above. There are a number of instances in which external graphics can be a useful complement to those provided with Logo.

Sources of Imported Graphics

A wealth of graphics is available for importation in Logo. There are three general sources of external graphics. Paint and illustration programs can be used to construct original graphics, which are then imported into Logo. Digitizing pads such as the *Koala Pad* or *Animation Station* combine a stylus with a graphics tablet. Many other graphics programs use a mouse as the drawing device.

Graphics digitizers such as *ComputerEyes* can be used to capture an image from a videotape or video camera. The saved graphic can then be imported into Logo. A digitized picture of your class or an image of your school can be incorporated into a Logo program. We will discuss video digitizers in more depth in a later column.

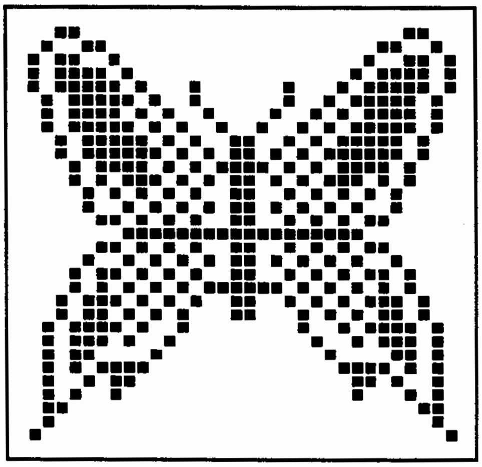
The third source of graphics is electronic "clip art." Clip art libraries consist of archives of illustrations that other people have created. Over time a surprising number of such libraries have been established. For example, dozens of disks of public domain *Print Shop* graphics have accumulated over time. You can obtain public domain collections of clip art through your local computer user's group, or through national user's groups such as the Boston Computer Society. Commercial clip art collections are also available. For example, commercial *Print Shop* clip art collections called "Minipix" are available from Beagle Brothers software for about $20.00 per disk.

Graphics Formats

Whether you create an original picture or acquire clip art developed by another artist, you will want to import the graphic into Logo. In order to import a graphic into Logo, it is helpful to know something about graphics formats.

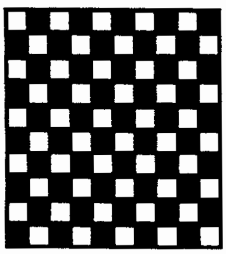
A Bit-Mapped Butterfly

An image on the computer screen is composed of dots called picture elements or *pixels*. If the butterfly shown above is enlarged, the individual pixels are visible.



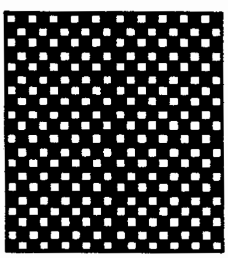
Individual Picture Elements (Pixels)

The resolution of the graphics mode depends upon the number of rows and columns of pixels available. The block shown below is composed of 10 rows and 10 columns of picture elements, for a total of 100 pixels.



A 10 x 10 Array (100 Pixels)

If the resolution is increased to 20 rows and 20 columns, a total of 400 pixels are required.



A 20 x 20 Array (400 Pixels)

The first Apple II computers had two graphic formats: low resolution (sometimes called *lowres*), with 40 rows and columns of pixels, and high resolution (sometimes called *hires*), with 140 rows and 192 columns of pixels. (Multiplying rows by columns demonstrates that there are 26,800 pixels in a high resolution screen.) Most versions of Logo for the Apple use this original high resolution graphics format. The illustration programs that accompany the Koala Pad and Animation Station graphics tablets, and the *Mouse Paint* program supplied with the Apple mouse also use this format.

When the Apple IIe, or *enhanced*, computer became available, a new graphics mode called double high resolution became available. A few of the earlier Apple IIe computers, know as *Revision A* computers, do not support this mode, but the majority of the machines, called *Revision B* computers, have it. (If you have an older *Revision A* Apple IIe, your Apple dealer can upgrade it to Revision B format for about $50.00) This graphics mode has 16 colors, in contrast to the six colors available in the standard high resolution mode. A paint program called *Dazzle Draw* and several other graphics programs were created that take advantage of this mode. The graphics possible with *Dazzle Draw* truly were dazzling, but since no version of Logo utilized the double high-resolution mode, *Dazzle Draw* graphics could not be imported directly into Logo.

When the Apple IIGS was introduced, even more graphics modes became available. A super high resolution mode of 320 rows and 200 columns of pixels was provided in this model. Paint programs such as *Deluxe Paint* that take advantage of this increased resolution can produce images that are truly stunning. These graphics modes can be summarized in the following table.

Graphics Mode	Pixels	Colors
Low Resolution	40 x 40	16
High Resolution	140 x 192	6
Double High Resolution	140 x 192	16
Super High Resolution	320 x 200	16

Some Apple II Graphics Modes

Several programs use their own proprietary graphics formats. For example, *Print Shop* uses a graphics format of 88 rows and 52 columns. *The Newsroom* desktop publishing program uses a graphics format of 245 rows and 192 columns. Needless to say, programs such as this make it more complicated to exchange graphics. However, in some cases conversion programs are available that make it possible to exchange graphics from one format to another.

Importing Graphics

Some of the earlier versions of Logo did not make any provision for exchanging graphics with other programs. As Logo matured, commands to import and export pictures were added. (This is one more reason to upgrade to the latest version of Logo, if you have not already.) For example, in *LogoWriter* this is accomplished with the **Loadpic** and **Savepic** commands.

If a graphics program uses the same format as Logo, the appropriate command can be used to import the picture into Logo. The following table lists some of the graphics programs available and the formats which they use. If your version of Logo is a current one, it will use the ProDOS disk operating system. Therefore, if other factors are equal, a ProDOS graphics program would be preferable. Apple provides a utility with ProDOS that will convert a DOS 3.3 file to ProDOS, but this adds an extra step to the otherwise simple process of importing a picture.

Program	Graphics Mode	DOS
Koala Pad	high resolution	DOS 3.3
Animation Station	high resolution	DOS 3.3
Mouse Paint	high resolution	ProDOS
Dazzle Draw	double high resolution	ProDOS
Deluxe Paint	super high resolution	ProDOS
Paint 8/16	high resolution *and* super high resolution	ProDOS

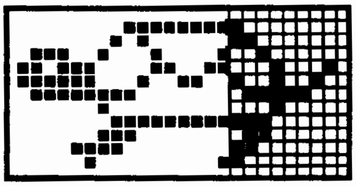
Some Graphics Programs for the Apple II

Reviewing the above table, two graphics programs utilize both the ProDOS format and the high resolution graphics mode compatible with most versions of Logo for the Apple: *Mouse Paint* and *Paint 8/16*. However, of the two, *Paint 8/16* has the wider range of features. As an added bonus, it supports both the high resolution and the super high resolution graphics modes. This means that you can use it in the high resolution mode to create graphics that will be imported into Logo. However, if you have an Apple IIGS, you can also use this program in the super high resolution mode to create images that take advantage of the superior graphics capabilities of that computer. *Paint 8/16* also supports the other graphics modes listed, in both ProDOS and DOS 3.3 formats, so it can be used with a wide range of systems.

We should quickly add that these are only graphics programs for the Apple which we have personally used. There are other good graphics programs for the Apple that are not in the table, and more programs may be developed in the future. By appending them to this table, you will be able to determine which ones have characteristics that are compatible with Logo.

File Formats

We should add a word about different formats in which graphics files may be saved. The simplest way to record the graphics on the screen and save the picture to disk is to divide the screen into a grid, one pixel per cell, and assign a number to each cell. For example, a 1 could be assigned to each black square and a 0 to each white square in the picture of the turtle below.



Recording the Value of Each Pixel

This would produce a tables of one's and zero's representing the picture that could be saved to disk. In actual practice, the high resolution mode has six colors rather than two—black, blue, green, purple, orange, and white—but the principle is the same.

The advantage of this method is its simplicity. It is very straightforward. The primary disadvantage is the amount of disk space required to save the picture, since a very large table is required to record all of the pixels. (You may recall that a high resolution screen has more than 25,000 pixels.) It is possible to save disk space by storing the graphic in a *packed* format. This method takes advantage of the fact that it is possible to store more than one pixel per byte. The primary disadvantage of this method that different software companies sometimes use different and incompatible methods for storing packed graphics files. Therefore, if you intend to import a graphic into Logo, and are given a choice of storage formats when you save the graphic to disk, you should choose the unpacked format.

Graphic Conversions

If you have a graphic that is not in the standard high resolution format and you would like to use it with Logo, it still may be possible to import the image into Logo. Before you can do this, you will need to covert the picture from its original format into the standard high resolution format.

Each Beagle Brothers "Minipix" collection of *Print Shop* art is accompanied by an editor that can be used to convert a *Print Shop* picture to the high resolution format that can be used by Logo. Judi Harris wrote a column in the *Logo Exchange* that provides detailed instructions on how to convert a *PrintShop* graphic to Logo format. See "Secular Conversions" on page 15 of the September 1988 issue of the Logo Exchange for instructions on how to import a *PrintShop* picture into Logo.

The *Graphic Exchange* is a program that was designed specifically for converting pictures from one graphics format to another. It can handle all of the graphics formats that we have mentioned above and more besides. For example, this program can also convert a Macintosh *MacPaint* image into a graphics format that can be imported into Logo on an Apple IIe. The *Graphics Exchange* requires an Apple IIGS to run, but if one is available in your school system, this program deserves consideration.

Graphics Cul de Sacs and Logo Heresy

There are some who view use of any graphics that were not created with the Logo turtle as heresy. These individuals oppose any use of paint programs, at least in combination with Logo.

Our view is more moderate. Turtle graphics are certainly a powerful feature of Logo. They can provide a useful tool for mathematical modeling, and for instruction on programming in a modular fashion. If the instructional goal is to teach a particular mathematical principle, or to teach programming, turtle graphics may be more appropriate. On the other hand, there are many circumstances in which these are not the primary instructional goals, and in which Logo can be used with other programs in an instructionally sound and appropriate manner.

In this respect, the recently released version of *LogoWriter* for the Apple IIGS is worth mention. We were unable to import any graphics into this version of *LogoWriter*. The technician at the Logo Computer Systems, Inc. (LCSI) technical support line informed us that this is because *LogoWriter* for the Apple IIGS does not use the super high resolution graphics format employed by other programs such as *Paint 8/16*, *Deluxe Paint*, etc. Instead, a proprietary format devised by LCSI is used that makes it incompatible with other graphics programs.

Those who feel that only turtle graphics should be used with Logo will applaud this step. We feel that it is a giant step backwards. Using a proprietary, incompatible graphics format creates the equivalent of a technologic moat. This means that video digitizers such as *ComputerEyes* cannot be used with *LogoWriter* for the Apple IIGS. It also means that it will not be possible, at least initially, to use graphics conversion programs such as *The Graphic Exchange* to convert pictures in the *LogoWriter* II+ / IIe format to the newer *LogoWriter* IIGS format. (The older version of *LogoWriter* will still run on the Apple IIGS, of course; it just will not take advantage of the additional colors and increased resolution.) It is possible that a conversion program may later be developed to convert graphics to the proprietary LCSI format, but the necessity of a conversion process adds another layer of complexity that will deter many.

Windows on the Future

Future computers will make it easy to transfer graphics from any program to any other program. For example, on the Macintosh, the key combination command-C can be used to copy a graphic from any program; the key combination command-V can be used to insert the graphic into any other program. This openness is not an accident. It is specified by Apple human interface guidelines.

Consequently, the user does not need to know nearly as much about different graphics modes and formats to transfer a picture from one document to another on the Macintosh. Users simply copy the graphic from one program and paste it into another.

This model has proven so powerful and attractive to users that it has been incorporated into future operating systems for the IBM-compatible computers. Both Microsoft *Windows* and the IBM OS/2 operating systems have been influenced by the Macintosh interface, and make it easy for the user to transfer information from one program to another. (These operating systems have been designated as the successors to the MS-DOS operating system used on the previous generation of IBM personal computers.)

As these newer generations of machines find their way into schools, movement of graphics from one program to another will be viewed as commonplace. In the meantime, we hope you will be able to use the information we have provided to add a mouse to the turtle that is already in your graphics menagerie.

References

Darooge, M. (1987). *8/16 Paint*. Grand Rapids, Michigan: Baudville.

Harris, J. (1988). Secular conversions. *Logo Exchange*, 7(1), 15-17.
MacLean, J. (1988). *The Graphic Exchange*. El Cajon, California: Robert Wagner Publishing, Inc.

Glen Bull is a faculty member in the Instructional Technology program of the Curry School of Education at the University of Virginia. He is currently serving as president of the Virginia Educational Computing Association, and is a member of the Virginia Department of Education planning commission for educational technology. Gina Bull is a system administrator for the University of Virginia Department of Computer Science. By day she works with UNIX; by night, with Logo.

Glen and Gina Bull
Curry School of Education
Ruffner Hall
University of Virginia
Charlottesville, VA 22903

BITNET addresses:
Glen: GBULL@ VIRGINIA. Gina: GINA@VIRGINIA.

Object Logo Is Back

by Hazem I. Sayed

Object Logo is available again as a result of a recent agreement announced at Macworld in Boston between Paradigm Software and Apple Computer. Paradigm Software has acquired *Object Logo* and is now responsible for its development and support.

Object Logo was originally developed and published in 1987 by Coral Software of Cambridge, Massachusetts, as an advanced implementation of the Logo language on the Macintosh. It became generally unavailable, however, along with Pearl Lisp, another Coral product, when Apple acquired Coral's assets in January 1989. Apple was interested in strengthening its tool set in the AI field and it has continued to support and develop Coral's high-end product, Allegro Common Lisp (now known as Macintosh Allegro Common Lisp).

The decision to bring *Object Logo* back is another indication of Apple's refocusing on the educational market. Steve Scheier, director of K-12 education marketing for Apple Computer, said,

> With more and more educators choosing the Macintosh for classroom use, it's good to know that an object-oriented version of the Logo language, which is so important to computer science literacy and problem solving curricula, will now be available on the Macintosh.

The renewed availability of *Object Logo* is also due in no little part to the sustained efforts and expressions of interest on the part of members of the Logo community in general and of *Object Logo* users in particular.

"The response from *Object Logo* users to the news has been extremely positive," said Paradigm's president. He added, "We were pleasantly surprised to learn of the number of ongoing research, development, teaching, and publication projects involving *Object Logo*. These and other users can now look to Paradigm for support, and we have ambitious plans for developing *Object Logo* and keeping it abreast of advances in Macintosh hardware and system software."

Object Logo can be described as a superset of traditional Logo. In addition to standard Logo features such as turtle graphics, editors, lists, and so on, *Object Logo* includes arrays as a data type, an extensive math package, an incremental compiler, a stand-alone application generator, and most important, an object-oriented programming system.

Significant math features include true fractional arithmetic, integer operations of unlimited size, and complex number arithmetic. In *Object Logo*, typing

```
Print (1 / 7) + (1 / 11)
```

yields

```
18/77
```

and not

```
0.233766
```

With Object Logo you can easily raise 2 to the 1024th power, compute the factorial of 300 or find the sine of a complex number.

Object Logo's incremental compiler compiles user-defined procedures on the fly. This means programs run more quickly, but without sacrificing the interactive environment commonly associated with interpreters. Fully debugged programs can also be turned into double-clickable stand-alone applications.

What makes *Object Logo* really different from other Logos is its object-oriented features. Object-oriented programming is now widely seen as the programming paradigm of the future, and *Object Logo* is clearly one of the most accessible entry points to this way of programming. In *Object Logo*, many traditional Logo features such as turtles and editors are implemented as built-in objects. In addition, many Macintosh interface features such as menus and windows are also implemented as objects.

In *Object Logo*, you can start out by using the default turtle in the same way you would use a turtle in other implementations of Logo. If or when you need more turtles or turtles with different properties, you can define them as subclasses or instances of the built-in turtle class. For example, the following instructions define a new class of turtle that beeps whenever it moves, at a frequency 10 times the distance it is moved.

```
Make "NoisyTurtle Kindof Turtle
Ask :NoisyTurtle [To Forward
    :Distance]
Usual.Forward :Distance
Toot :Distance * 10 200 .1
End
```

To create an actual beeping turtle and to move it you would type something like this:

```
Make "ANoisyTurtle Oneof :NoisyTurtle
Ask :ANoisyTurtle [Forward 30 Left
   45]
```

The same ideas apply to windows and menus. You can use the default windows, or you can design ones with the particular properties you want. It can be argued that turtles and turtle geometry are an ideal way to learn and explore object-oriented programming concepts such as classes, inheritance, and specialization, much in the same way that turtles and turtle geometry are an ideal way to learn general programming concepts such as program control, recursion, and so forth. In fact, one can view the elegant fit between Logo and object-oriented programming as further evidence of the general robustness of Logo as a learning environment with few limits.

Paradigm Software is currently shipping *Object Logo* 2.5, a version that adds a few enhancements to the latest version available from Coral Software. *Object Logo* 2.5 adds support for 32-bit color QuickDraw, hierarchic menus, compatibility with all Macintosh hardware, and comes with a single fully indexed manual. It is available directly from Paradigm for $149.00, plus shipping and handling. Paradigm is extending a limited time offer (through January 31, 1991) to earlier owners of *Object Logo* to convert to version 2.5 for $55.00, plus shipping and handling. *Object Logo* is also available in five-, 10-, and 20- license packs for lab settings.

Paradigm provides free technical support to its customers by telephone or via AppleLink at PARADIGM.TEC.

Paradigm Software
P.O.Box 2995
Cambridge, MA 02238
(617) 542-4245
AppleLink: PARADIGM

MathWorlds

edited by A. J. (Sandy) Dawson

The mathematical world of the professional mathematician and that of children in schools are both matters of social construction (or so many writers today are claiming), but in the article below, Matos argues that the latter world is much more restricted than the former. He writes about a small experiment he carried out that attempted to remove some of the restrictions from the school mathematics world so as to bring it more in line with the mathematician's world.

Logo and Mathematics: Sprite Control with First Graders

by João Filipe Matos

It is important to emphasize the distinction between mathematics—a vast domain of inquiry whose beauty is rarely suspected by most non-mathematicians—and something else we could call school mathematics, a social construction derived from a set of historical accidents that determined the mathematical knowledge that all adults should have. In fact, the difference between the type of mathematics greeting children in most schools and the sort of intellectual activity enjoyed by mathematicians is enormous. I will use the term mathematical investigation to refer to an activity that focuses on mathematics as something that people *do* rather than as a study of something people have done.

By their own nature, mathematical investigations are intimately linked to the mathematical content, but the focus of those activities is the process used to deal with the content. Processes that are relevant in terms of mathematics education include posing problems, generating examples, specializing and generalizing, devising symbols and notation, recording observations, exploring a question systematically, identifying patterns, making and testing conjectures, and communicating with an audience.

Sprite Control with First Graders

Ana's pupils are 20 first graders, six- to seven-year-olds. She wanted to have a computer in the classroom which she sets aside for the children to use during the whole day. The software available is an *MXS Logo* version with 30 turtles and the capability to set 60 different shapes and to define a constant speed for each turtle.

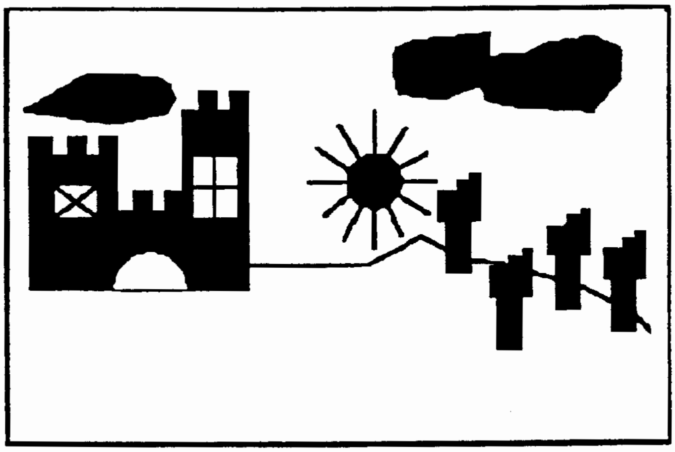
Thirty different sprites can be worked with and defined.

In addition to the *MXS Logo* version, children use the program LP (Logo Português) developed to allow children to construct procedures "in action" (Matos, 1986). When loaded with the program LP, the computer asks if the student wants to *teach* or to *do*. With the *teach* option, the child can teach the turtle a procedure. Entering the *do* option allows the learner to execute any procedure or command in direct mode. With this program

- the turtle executes each command at the moment it is entered in the computer.
- children can erase the last command entered.
- messages advise in case of invalid name for a procedure, space default, input default or invalid name for a command.
- children can write and use procedures with variables and tail-recursive procedures.

Children started using Logo with a project of their own.

Peter and Mary made the scenario below and had to determine how to get the king, the queen and princess to walk on into the picture



Ana (A), the teacher, watched Mary (M) and Peter (P) working on their project.

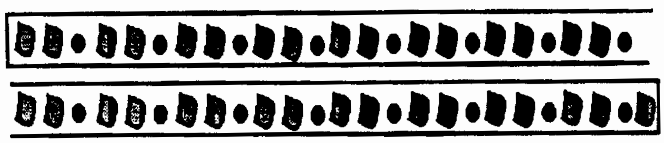
M: Ana, we want the king to walk until there.
P: How much is it?
M: It's 33!
M: Yes, but if you put FD 33 he goes too fast!
A: So you want the king to walk 33 steps...
M: We can walk a little and wait a little, walk and wait... It can be FD 1 WAIT 1 all the time...
P: How many times... we can write REPEAT 33 FD 1 WAIT 1.

They did it. But it was too slow.

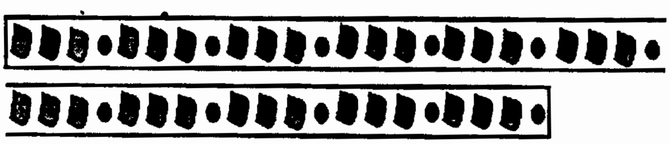
P: That's not the idea... The king is very lazy...
M: I think that's good.
A: What could you do to make the king not so lazy?
P: Well... we can put a smaller wait... less than one... wait zero...
M: So the king doesn't walk!
P: Yes... but less than one is zero... zero, one.. so we write wait one zero.
M: That's wait ten... It takes too much time...
P: We put a half of one and a half of zero...
M: Oh no... what's that? There is a way. We can write FD 2 or FD 3 and wait less times...
A: That's a good idea, Mary.

Mary and Peter started working on their conjecture. Using paper and glue they built a coded model of the sequence of orders FORWARD and WAIT. And soon they started handling the models and discussing.

M: Here we have 33 WAIT 1.



P: But this one has 16 WAIT 1s. It needs another step...



P: Hum... this one is better, we have only 11 WAIT 1s. Let's try it now.

A good solution was found. From this point on the children were willing to investigate the problem of decomposition of an integer as a product of two integers, plus another integer. This is a really important investigation for first graders, with very powerful consequences from a mathematical point of view.

All during their work together, a great determination to solve the problem was observed in the children. Moreover, a certain delight developed, and the children enjoyed the possibility of shaping the conclusions they were constructing. The initial rejection of the command FORWARD and the adoption of the strategy of a combination of the commands FORWARD and WAIT with REPEAT provided the opportunity for the children to engage in a problem situation with close links to mathematics. Based on models created by themselves with paper and glue, children developed different solutions for the problem of displacement of the figures and discussed the criteria they should use to make the right decision.

Conclusion

Even though the children lacked some of the usual language to verbalize and discuss mathematical ideas, in this situation they had a context upon which they could base their reasoning. Logo provided the opportunity to investigate a small but important and significant problem in the context of a project children really wanted to perform.

School mathematics—as it is in most schools—tend to transmit the conception that mathematics is something ordinary people cannot do. This can be one of the elements that leads to a negative attitude about mathematics. Logo offers a medium to explore and do mathematics. It involves a process of internalization that takes place through the dialectic relation of the student with the situation and the communication/justification activity that it tends to develop.

Reference

Matos, J. (1986). Constructing LOGO Procedures in Drive Mode. Proceedings of the Second International Conference for LOGO and Mathematics Education. London: University of London, Institute of Education.

João Filipe Matos is with the Departamento de Educação da Faculdade de Ciências Universidade de Lisboa

A. J. (Sandy) Dawson is a member of the Faculty of Education at Simon Fraser University in Vancouver, Canada. He can be reached through Bitnet as userDaws@SFU.BITNET

Extra For Experts

Fractals III: Monsters of Mathematics
by Mark Horney

Around the turn of the century a group of mathematicians, among them Giuseppe Peano (1858-1932), George Polya (1887- 1985), Georg Cantor (1845-1918), and Helge von Koch (1870-1924), collected a bestiary of mathematical monsters. It had curves weaving such intricate tapestries they actually filled the spaces inside squares and triangles. There were sets that remained infinitely large after being entirely thrown away. Finally, there were lines with so many kinks, no part of them is straight. These mathematical oddities were strange beyond belief and shredded intuitions to tatters.

In recent years these monsters have found a home in the theories of Benoit Mandelbrot, who calls them fractals, after a Latin verb meaning *to break* (Mandelbrot, p. 4). Today, fractals are commonplace: still strange and beautiful, but now quite approachable, especially to students of Logo. Deceptively simple Logo procedures draw all the classic fractals, and students can invent new examples almost at will. (See the two previous *LX* articles in this series, *Making Recursion Visible* and *Representations*, for information about constructing fractals). Fractals and their bizarre characteristics are also a gateway to several topics in mathematics. Three of these characteristics are self similarity, space filling, and fractional dimensionality.

Self-Similarity

Self-similarity is the most visible characteristic of fractals and is seen in the way fractal shapes repeat themselves over and over as a fractal is examined in smaller and smaller detail. Self-similarity is a result of the recursive procedures used to define fractals, where each new level is built from those proceeding. The level 10 Dragon curve below is made from two level 9 Dragons, each of which made from two level 8 Dragons, made from two level 7 Dragons, made from two level 6 Dragons...made from two level 1 Dragons, which finally are made from two level 0 Dragons, which are just line segments, but Dragons none the less. Any part of a completed self-similar curve hides the whole curve repeated in miniature.

Figure 1: Dragon Curve

Self-similarity is also shown in graphs used to represent fractals. The figure below shows the Square Sweep fractal and a graph representing the five procedures used to draw it (Mandelbrot attributes this fractal to Peano, p. 66. It uses the same left and right handed "Hat" generators that the C and Dragon Curves do). Imagine traveling around the graph starting at any one of the six vertices. Note how it's possible to travel to any vertex, from any vertex. A graph with this property is call *strongly connected.* Strongly connected graphs yield self-similar fractals.

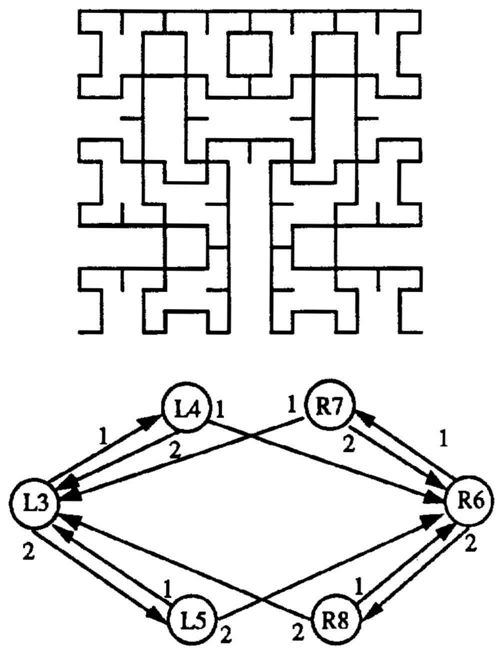


Figure 2: Square Sweep and Graph

Graphs that aren't strongly connected represent fractals only partially self-similar. The figure below shows the Super Sweep and its graph. The Super Sweep is made from four Square Sweep curves wired together with three other procedures. Each part of the Super Sweep is self-similar, but the curve as a whole is not, its overall shape is never repeated at lower levels. This same thing is true of von Koch's famous Snowflake. Each of its three sides is self-similar, but the curve as a whole is not.

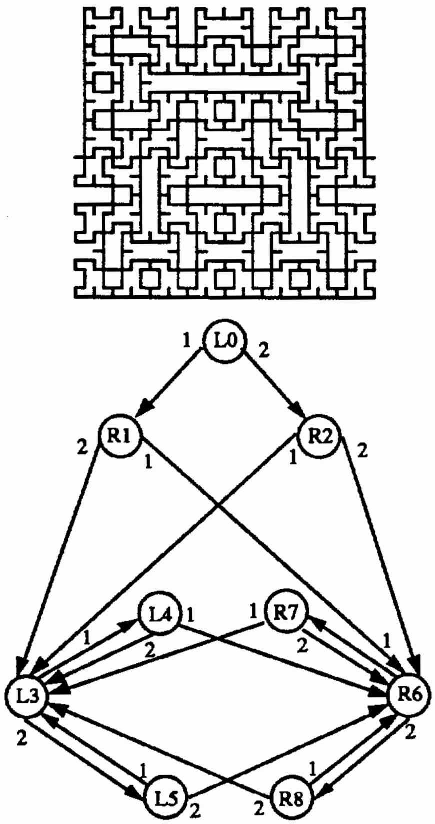


Figure 3: Super Sweep and Graph

Self-similarity is a frequent characteristic of fractals, but it is not mandatory. Here's a project for you to try: Design a fractal that is nowhere self-similar.

Space Filling

The Peano curve shown below fills a square. It is a one-dimensional line that twists back and forth so much that it actually passes—eventually—through every point inside a square, which is of course, a two dimensional object. Keep in mind that what seems to be a filled square shown in below, is an optical illusion produced by the limited resolution of the screen or printer. The square is actually only filled after an infinite number of steps. Peano was able to prove mathematically that his curve would follow such a path. To do this, he found a correspondence between the points on the curve, and the points in the square. This is the standard procedure, invented by Cantor, for showing sets to be equal, particularly infinite sets. Cantor used this technique to show there are as many even numbers as odd numbers, as many whole numbers as fractions, and more irrational numbers than rational ones (a fact many frustrated 7th graders will attest to).

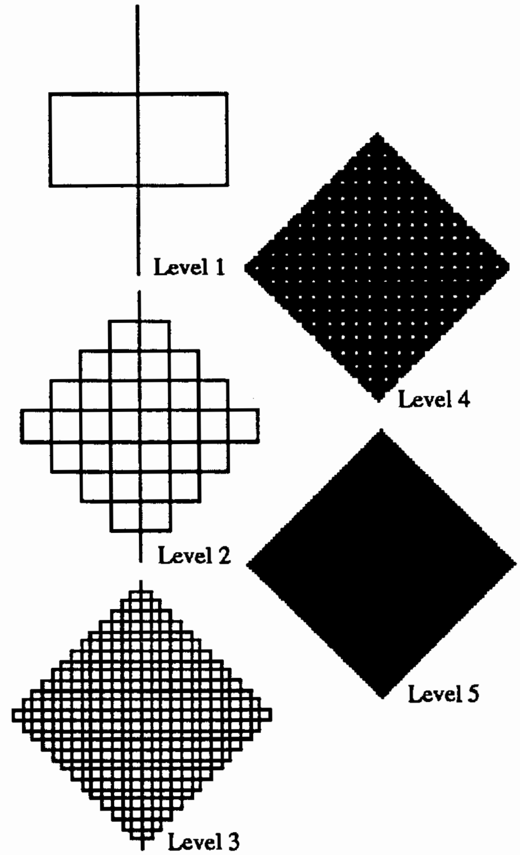


Figure 4: Peano Curve

Peano noted his curve draws a rectangular lattice inside the square and with each iteration, the distance between the vertical and horizontal lines of the lattice was reduced by a third. Capitalizing on this he defined positions on the curve and in the square using base three coordinates, and then used these coordinates as a recipe for drawing the curve, the digits telling when the curve would pass though any particular point in the square. This algorithm completes the proof since it matches up every point in the square with a point on the curve.

Here's another project: Design space filling curves that work in bases other than 3.

Fractal Dimensionality

Peano's curve presents a paradox about dimension: the shape of the curve is both a one-dimensional line and seemingly a two-dimensional square. This violates the the way dimension taught in geometry classes. There students learn about one-dimensional lines, objects with length but no width,

Extra for Experts - continued

two-dimensional surfaces having both length and width, and three-dimensional spaces with length, width, and depth. Dimensions are always given as whole numbers and nowhere are there objects with two different dimensions at the same time.

Mandelbrot solves the paradox with the mathematics of Felix Hausdorff (1868-1942). In 1919 Hausdorff developed a way to measure shapes that when applied to normal shapes like points, lines, squares and cubes, gave the expected "sizes" of 0,1,2,and 3, but which could also be applied to more exotic shapes. The Hausdorff measure of the Peano curve is 2, since it fills the two dimensional shape of a square. The "C" curve, the Dragon curve, the Square Sweep and the Super Sweep also have measure 2, although its less clear that they fill any shape. Mandelbrot claims that the Hausdorff measure of a shape should be considered its dimension. If this is accepted, then the Snowflake of von Koch has a fractional dimension since its Hausdorff measure is log 4 /log 3 (approximately 1.2618). The justification for Mandelbrot's claim is based on arguments of what is meant by "dimension." In one sense, dimension is a measure of the "wiggliness" of a shape, which is just what Hausdorff's measure calculates (this highly abstract mathematical term comes from Richard F. Voss, p. 29). The more corners a shape has, the more that it bends back and forth, the higher it's dimension. Peano's curve wiggles so much it becomes equivalent to a square. The Snowflake wiggles more than a line, but less than a square, so has dimension between one and two. In this way fractals are said to have fractional dimensions.

The Hausdorff measure can be very difficult to calculate, but for simple fractals, those with only one generator and where each segment of the generator is reduced by the same amount, the formula is (log N / log R), where N is the number of sides in the generator and R is the amount sides are reduced each iteration. The Peano curve has 9 sides, each reduced by a third, so its dimension is log 9 / log 3, which is 2. The Dragon curve has dimension log 2 / log $\sqrt{2}$, which is also 2. The Snowflake has 4 sides, each reduced by a one third and so its dimension is log 4/log 3.

Fractals can also be drawn with dimensions higher than two and less than one. The basic C curve is reduced at each level by the square root of 2. If this reduction factor is changed to the cube root of 2, a C curve of dimension three is produced. Reducing sides by the fourth root of 2 yields a C curve of dimension 4 and in general, the root of 2 by which sides of the C curve are reduced gives the curve's dimension, even for fractional roots of 2 (see below).

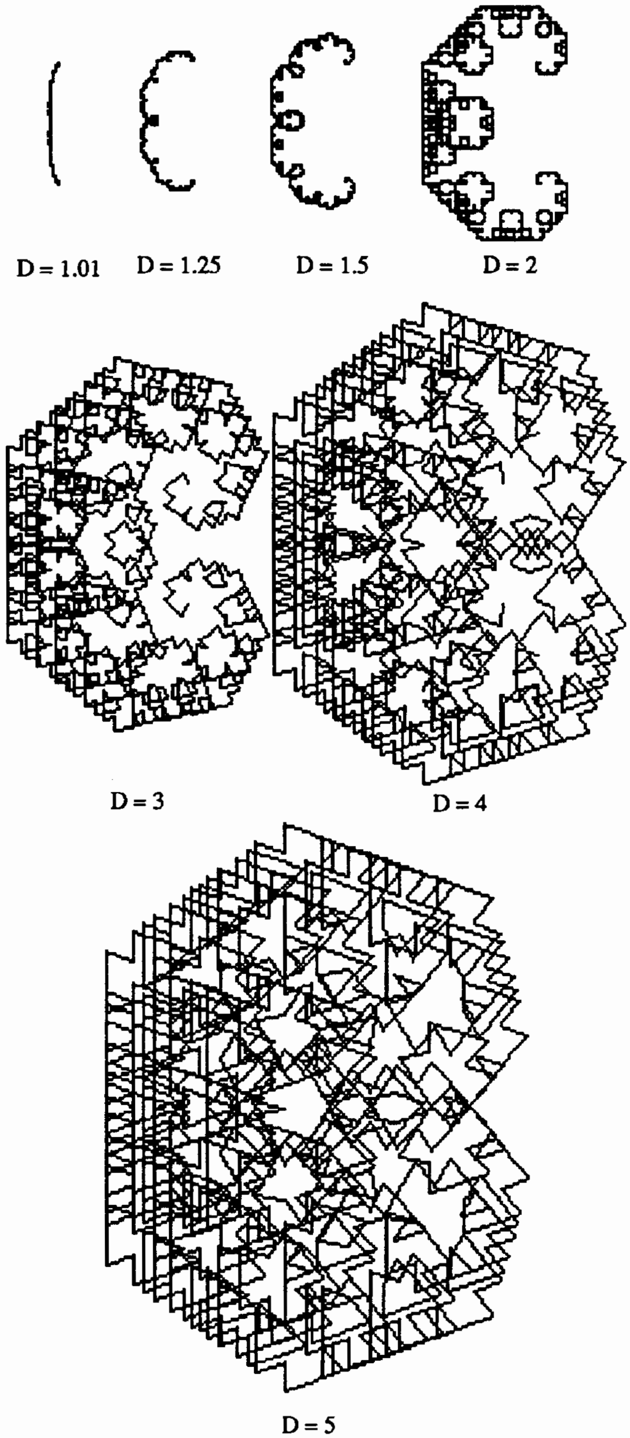


Figure 5: Level 10 C Curves

The Cantor set described above has dimension .6309, (log 2 / log 3). Since this set is constructed by throwing pieces away, it wiggles less than a line and so has dimension less than one (see Figure 6).

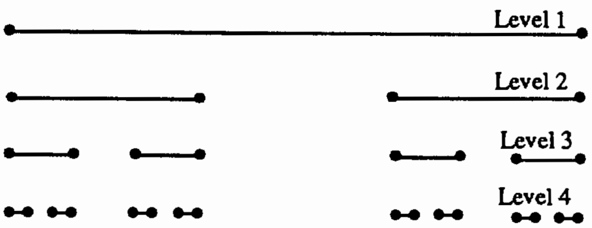


Figure 6: Cantor Set

Here's yet another project: The dimension three C curve shown above grows fast enough to fill the points of a cube but is still a flat two-dimensional object. Devise dimension three curves using new Logo commands for UP and DOWN in addition to LEFT and RIGHT that allow the turtle to escape into 3-space. Drawing such fractals will require procedures for projecting 3-D images onto the 2-D computer screen. These can be found in *Turtle Geometry* by Abelson and diSessa.

Fractals, with their counter intuitive characteristics, make a wonderful playground for Logo students. In it they can explore in open ended, self-directed fashions, and also delve into mathematical topics in geometry, analysis, topology, and graph theory. With environments such as this, students can learn about Logo and with Logo. Being able to do both is critical for students and for Logo itself.

References

Abelson, H. & diSessa A. (1980). *Turtle Geometry*. Cambridge, MA: The MIT Press.

Horney, M. (1990). Fractals I: Making recursion visible. *Logo Exchange*, 9 (1), 23-29.

Horney, M. (1990). Fractals II: Representations. *Logo Exchange*, 9 (2).

Mandelbrot, B.B. (1983). *The fractal geometry of nature*. New York: W. H. Freeman and Company.

Voss, R.F. (1988). Fractals in nature: From characterization to simulation. In H. Peitgen and D. Saupe (Eds.) *The science of fractal images*. New York: Springer-Verlag.

Mark Horney was a middle school math, science and reading teacher for 10 years and later spent three years as a high school computer science instructor. He is currently a doctoral student at the University of Oregon.

Mark Horney
ISTE
1787 Agate Street
Eugene, Oregon 97403
BITNET: mhorney@oregon

Statement of Ownership, Management and Circulation (Required by 39 U.S.C. 3685). 1A. Title of Publication, Logo **Exchange.** B. Publication No., 000-554. 2. Date of filing, Sept. 12, 1990. 3. Frequency of issue, monthly except June, July; August; combined December/January issue. 3A. No. of issues published annually, 8. B. Annual membership dues, $25 in U.S. 4. Complete mailing address of known office of publication (not printer), ISTE, University of Oregon, 1787 Agate St., Eugene, OR 97403-9905. 5. Complete mailing address of the headquarters of general business offices of the publisher (not printer), ISTE, University of Oregon, 1787 Agate St., Eugene, OR 97403-9905. 6. Full names and complete mailing addresses of the publisher, editor, and managing editor, Publisher, ISTE; Editor, Sharon Yoder; Managing Editor, David Moursund; University of Oregon, 1787 Agate St., Eugene, OR 97403-9905. 7. Owner, International Society for Technology in Education, University of Oregon, 1787 Agate St., Eugene, OR 97403-9905. 8. Known bondholders, mortgages, and other security holders owning or holding one percent or more of total amount, None. 9. The purpose, function, and nonprofit status of this organization and the exempt status for Federal income tax purposes has not changed during preceding 12 months. 10. Extent and Nature of Circulation. Average No. Copies Each Issue During Preceding 12 Months. A. Total no. copies (net press run), 1,444. B1. Paid Circulation, sales through dealers and carriers, street vendors and counter sales, 0. B2. Paid Circulation, mail subscriptions, 1,219. C. Total Paid Circulation (sum of 10B1 and 10B2), 1,219. D. Free distribution by mail, carrier or other means, samples, complimentary, and other free copies, 200. E. Total Distribution (Sum of C and D), 1,419. F1. Copies not distributed, office use, left over, unaccounted, spoiled after printing, 25. F2. Copies not distributed, returns from news agents, 0. G. Total (Sum of E, F1 and 2—should be equal to net press run shown in A), 1,444. Actual No. Copies of Single Issue Published Nearest to Filing Date. A. Total no. copies (net press run): 1,510. B1. Paid Circulation, sales through dealers and carriers, street vendors and counter sales, 0. B2. Paid Circulation, mail subscriptions, 1,134. C. Total Paid Circulation (sum of 10B1 and 10B2), 1,134. D. Free distribution by mail, carrier or other means, samples, complimentary, and other free copies, 200 E. Total Distribution (Sum of C and D), 1,334. F1. Copies not distributed, office use, left over, unaccounted, spoiled after printing, 176. F2. Copies not distributed, returns from news agents, 0. G. Total (Sum of E, F1 and 2—should be equal to net press run shown in A), 1,510. 11. I certify that the statements made by me above are correct and complete. Tasanee Fry, Bookkeeper, ISTE.

Logo: Search and Research

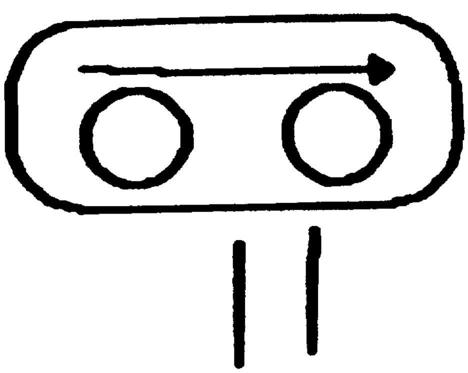
Strategies for Writing Procedures

by Douglas H. Clements

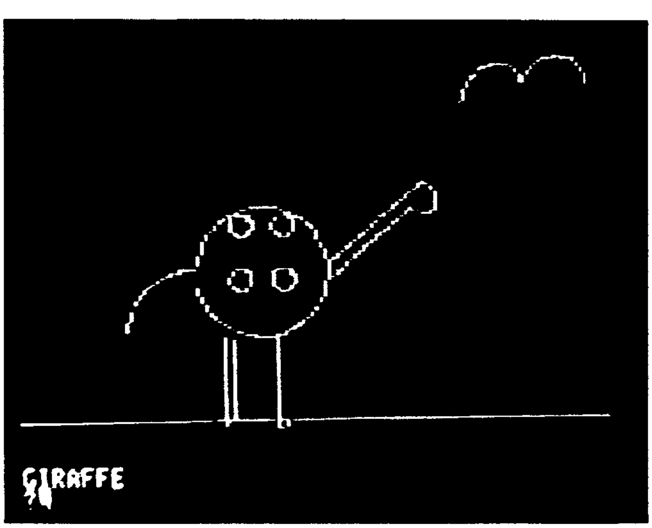
Three young students started their Logo project by writing the following procedure.

```
TO E.T.
THROAT
HADE
EYE
EYE
EYEBROWS
END
```

They then taught the computer how to draw each part, defining the throat, "hade" (head), eye, and eyebrows separately. Eventually, this produced The figure below.



Two others just "played" with a new ARC procedure they had written, adding to their picture until a scene emerged:



Many who teach Logo have observed that students employ different types of strategies, as did these first graders. In a previous column (March 1990), we saw that students develop through *stages* in learning to program. For the next several months we will take a closer look at specific tactics students use as they tackle Logo tasks.

Top-down or Bottom-up?

Observers often notice two types of programmers, top-down and bottom-up (Papert, Watt, diSessa, & Weir, 1979; Watt, 1979). Top-down programmers prefer to plan in advance, have a clear idea of the end result, and look at the "big picture." They often write the main procedure (e.g., E.T.) first in terms of a few, general parts and then break those down into smaller parts. They break down each of these smaller parts, and so on, until they have specified every step. One student described these parts as "mind-sized bites" (Papert, 1980). These students often use mathematical analyses.

Bottom-up programmers "plan as they go," using what they see happening to make decisions about what to do next. They build up a program piece by piece, discovering what works as they proceed. Often, especially in their early stages of learning, they do not use subprocedures at all. They frequently work on shorter tasks.

Students' products alone cannot always determine what strategy they used. It is the process that is important. In fact, students using these two strategies often will converge on similar projects. Additionally, many children use a combination of the two approaches (Papert et al., 1979).

Is One Strategy "Better"?

Top-down strategies seem more sophisticated. Is this just an adult bias? Possibly not. Students who used top-down strategies in a writing task were also more successful at Logo programming (Bradley, 1985). Perhaps success depends on the ability to formulate a broad idea about a task. This idea may provide a structure for thinking through the task. Lack of success may result from an inability to structure such an idea. This leads to the question: What if we attempt to teach top-down planning?

Teaching a Top-Down Strategy

A group of researchers in Belgium did just that (De Corte, Verschaffel, Hoedemaeders, Schrooten, & Indemans, 1988). They provided sixth-grade students with a balance of exploratory projects and systematic instruction. They taught the students a top-down strategy for programming. (Other successful Logo projects have used virtually the same method; see Clements, 1986; Clements & Gullo, 1984.)

Students plan their procedures away from the computer. They draw a tree diagram in which they subdivide a complex drawing into building blocks that are easy to program. Such a diagram is at the bottom of the page.

They construct separate drawings of the different parts. For each, they record the lengths and turn amounts, as well as the starting and ending position of the turtle. The also plan "connecting links" that move the turtle from the ending position of one part to the starting position of the next.

At the computer, the students start with the most global procedure, called the "mother procedure" (e.g., CASTLE). They then write each subprocedure according to a "left first, depth first" rule until they reach the lowest level of the tree diagram.

Students test each new procedure immediately after defining it by calling the "mother procedure." The result appears on the screen and can be checked and debugged. The error message "THERE IS NO PROCEDURE NAMED..." indicates the next procedure to write. For examples, students might write the following.

```
TO CASTLE
TOWER
CL-TB        (LC-TB stands for "Connecting Link
BRIDGE       Tower to Bridge")
CL-BT
TOWER
END
```

The students type CASTLE and read that THERE IS NO PROCEDURE NAMED TOWER IN LINE TOWER AT LEVEL 1 OF CASTLE. Next, they define TOWER procedure.

```
TO TOWER
WALL
CL-WW
WINDOW
CL-WR
ROOF
END
```

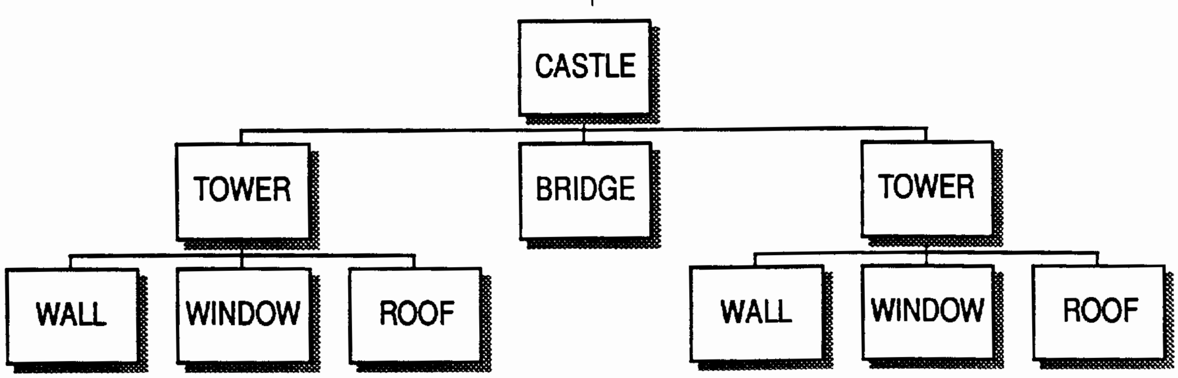
The students call the mother procedure again; the computer says there is no procedure called WALL. And so on.

Actually, the castle task was the end-of-the-year assessment for the research project. Most students got it right (21 out of 24 students), although about half needed help, mostly on inputs to turn commands.

Only five, however, spontaneously used a tree diagram (as they were taught). Several of these had shortcomings. This may not be as bad as it seems; 15 students thought a tree diagram was unnecessary for the simple castle drawing. They may have internalized the process. However, 11 students could not do it even when asked.

What about the actual Logo code they produced? Most students developed well-structured programs consisting of separate parts and the appropriate connecting links. They used the instructed top-down programming style and verified regularly the coded procedures. Half of the 24 students wrote a "model" solution. So, overall they did well. We need further research, especially classroom-based research, to see if modifications of this approach and strategies are more useful to all students. (See previous columns on planning for additional suggestions.)

In summary, it may be helpful to structure Logo work so that students predict and plan before programming. Many find it easier to think and reflect while using paper and pencil then while working on the computer (De Corte et al., 1988).



Search and Research - continued

Instruction in top-down strategies may be helpful, especially for students who use only bottom-up strategies. Remember that certain children may resist planning their Logo programs, enjoying instead working with mathematics in the intuitive style that is more natural to them (Papert, 1987). Appreciate and encourage such work. However, also lead these students to use top-down, procedural thinking where this is appropriate (Clements, 1986; Clements, 1989; De Corte et al., 1988; Watt, 1979). Most important, perhaps, is observing students' thinking strategies and using this understanding to guide instruction.

References

Bradley, C. A. (1985). The relationship between students' information-processing styles and LOGO programming. *Journal of Educational Computing Research, 1*, 427-433.

Clements, D. H. (1986). Effects of Logo and CAI environments on cognition and creativity. *Journal of Educational Psychology, 78*, 309-318.

Clements, D. H. (1989). *Computers in elementary mathematics education*. Englewood Cliffs, NJ: Prentice-Hall.

Clements, D. H., & Gullo, D. F. (1984). Effects of computer programming on young children's cognition. *Journal of Educational Psychology, 76*, 1051-1058.

De Corte, E., Verschaffel, L., Hoedemaeders, E., Schrooten, H., & Indemans, R. (1988, April). *Acquiring programming skills in Logo: An exploratory study with sixth graders*. Paper presented at the meeting of the International Association for Computing in Education, New Orleans.

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

Papert, S. (1987). Computer criticism vs. technocentric thinking. *Educational Researcher, 16*, 22-30.

Papert, S., Watt, D., diSessa, A., & Weir, S. (1979). *Final report of the Brookline Logo Project. Part II: Project summary and data analysis (Logo Memo No. 53)*. Cambridge, MA: Massachusetts Institute of Technology, Artificial Intelligence Laboratory.

Watt, D. (1979). *Final report of the Brookline Logo Project. Part III: Profiles of individual student's work (Logo Memo No. 54)*. Unpublished manuscript, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, Cambridg
e, MA.

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

Global Logo Comments

Edited by Dennis Harper
University of the Virgin Islands
St. Thomas, USVI 00802

Logo Exchange Continental Editors

Africa	Asia	Australia	Europe	Latin America
Fatimata Seye Sylla	Marie Tada	Jeff Richardson	Harry Pinxteren	Jose Valente
UNESCO/BREDA	St. Mary's Int. School	School of Education	Logo Centrum Nederland	NIED
BP 3311, Dakar	6-19, Seta 1-chome	GIAE	P.O. Box 1408	UNICAMP
Senegal, West Africa	Setagaya-ku	Switchback Road	BK Nijmegen 6501	13082 Campinas
	Tokyo 158, Japan	Churchill 3842	Netherlands	Sao Paulo, Brazil
		Australia		

Japanese educators are becoming more enthusiastic about the use of Logo in schools. Only recently have the number of computers in Japan's schools seen a dramatic increase. Our Asian correspondent, Marie Tada, files this report from Tokyo, Japan.

Findout
by Marie Tada

The intrinsic power of Logo to motivate and fascinate is witnessed in its changing "cloaks" throughout the years. My students were mesmerized with the *Terrapin Logo* that we used on the Commodore 64 and delighted in the FILL command that was added with *IBM Logo*. They have created magnificent projects using COLORUNDER, WHEN statements, and multiple turtles in *LogoWriter* and have been able to extend Logo use to more subject areas using *LogoWriter's* built-in word processor. They have also been totally absorbed in the creative process involved in *LEGO-Logo* projects.

I have often thought about what the next version of Logo should include and have considered that somehow the addition of integrated applications including a database, spreadsheet, graphing, and telecommunications functions would provide us with most of our software needs rolled into one. I have also felt that the whole package, like the Logo language itself, should somehow be "built" by the students in an ongoing Logo learning environment. In this way a student's disks would become more than ever a kind of history of ideas and projects that build up to the present.

With this in mind, I was delighted to discover here in Japan a company that is working on many of my "dream features" and incorporating these into a new version of Logo. The Fukutake Publishing Company has been a leader on the educational publishing scene for years. Mr. Hiroshi Goto, the director of the New Media Laboratory, has been a Logo enthusiast from way back and was the person who brought Logo to Japan. He has worked closely with Hillel Weintraub, and his company has close ties with the MIT Media Laboratory, to which they have sent many researchers to work on projects over the years.

From all of this enthusiasm and research has come the software package called *Findout*, which is designed to be a "personal work station" that provides an interactive environment for learning. Word processing, graphics, and database management are all rolled into one software package. The built-in functions can be combined or modified using Logo procedures to produce more personalized tools. *Findout* aims to be a truly integrated package in that the data management tools are available to the user at all times and can be used actively within the word processor or graphics functions.

The *Findout* word processor contains an outlining mode that allows students to jot down thoughts as they occur to them and later rearrange, edit, expand, or modify these in the creation of a final text. Students are able to design and build their own dictionaries that allow the inclusion of personal study helps such as parts of speech and grammatical rules. The Japanese students using this program have found this part of their "work station" very valuable in both Japanese and foreign language study.

The graphics mode includes all of the usual Logo commands, along with a number of special features. Up to 30 turtles can be manipulated simultaneously. Over 3,000 Kanji (Chinese characters used in writing) and all of the English characters, numbers, and punctuation are included in the shapes area. There is room for users to create 192 shapes of their own. Automatic scaling of the turtle's shape can be accomplished using the SETSIZE command with inputs for the desired length and width. A GRID command allows one to see a coordinate grid on the screen with the grid increments and color being determined by user input. Another primitive, LINK, allows "rubber bundling" to take place, much like moving rubber bands on a geoboard. A turtle can be commanded to move away from the others with the same shape; then changes occur to reflect the new turtle positions. In the hands of a creative teacher, use of the grid and rubber bundling can have excellent applications for the exploration of higher level mathematics such as plotting equations and watching the rotation of three-dimensional figures. A PAINT command allows mixing of colors to create an astounding number of color combinations. The function keys can also be programmed to make pop-up menus for directions or help.

The overall structure of the program is done using the concept of an "area map." Movement from the database, dictionary, shapes, word processing, and graphics screens is easy and fast. Procedures can be viewed in a window without leaving the contents list, and files can be easily interchanged.

I observed two Japanese junior high school classes that used *Findout* as a teacher/student learning tool. One was a science class that used a teacher-prepared graphics program to explore planetary paths. Another was a Japanese language class that used the database function to examine older forms of Japanese writing. In both of these classes the students were arranged in discussion groups using the computer as a learning resource for exploration and discovery.

Fukutake's New Media Laboratory has a bi-monthly publication with accompanying disk that gives ideas for use of *Findout* in a variety of subject areas. An electronic bulletin board—FindNET—is available for users to exchange ideas and applications. The laboratory also works on requests from teachers around the country for the development of lessons based on particular topics of interest. In the future, Fukutake plans to add an infrared remote control to be able to control a mechanical turtle without wires, LAN-like remote terminal typing, and a telecommunications module.

Findout is certainly an excellent package and many of you would probably be interested in taking a closer look. As of now it is available only in Japanese, but an English version will hopefully be on the way soon.

Eurologo 91

Call for papers

Where:
Area delle Scienze
Universit'a di Parma
Parma, Italy

When:
27 to 30 August 1991.

Themes:
1. Classroom experiences with Logo from primary to higher education, including special education.
2. Logo prospects and research. Logo and otheR languages. Logo anD artificial intelligence.
3. Programming environments. Microworlds. Logo and subjects in school curriculum.

Conference Programme
Invited speakers
Presentation of papers
Demonstration of programs and new applications
Poster session
Video tape exhibition
Scientific secretary: Eduardo Calabrese

E-mail concerning the conference
BITNET: CALABRESE@VAXPR.CINECA.IT

Postal address:
Dipartimento di Fisica
Viale delle Scienze
43100 PARMA, Italy

Phone:
(Italy) 0521 560268

Papers:
The official language of the conference is English. Papers must not be longer than 20 pages (1 page = 40 lines). Submit 3 copies or ASCII text by 31 March 1991. A decision on acceptance is promised by 15 May 1991.

Logo Exchange
ISTE/University of Oregon
1787 Agate Street
Eugene, OR 97403-9905