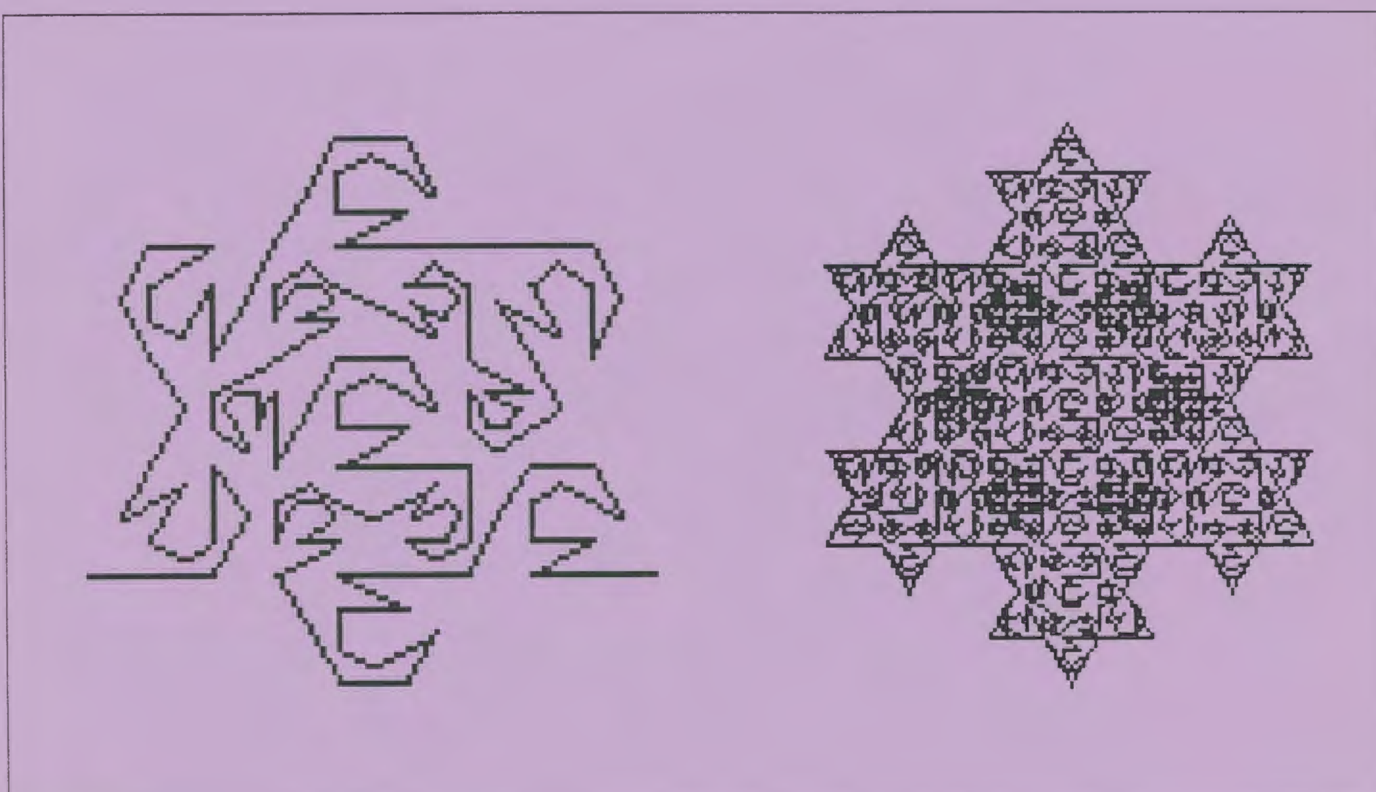
Journal of the ISTE Special Interest Group for Logo-Using Educators

LOGO EXCHANGE

March 1991 Volume 9 Number 6



International Society for Technology in Education

ISTE Publications

LOGO EXCHANGE

Volume 9 Number 6 Journal of the ISTE Special Interest Group for Logo-Using Educators March 1991

Advertising space in *Logo Exchange* is limited. Please contact the Advertising Coordinator for space availability and details.

Logo Exchange (ISSN 0888-6970) is published monthly September through May except for a combined issue for December January by the International Society for Technology in Education (ISTE), 1787 Agate Street, Eugene, OR 97403-1923, USA; 503/346-4414. Subscription rates per year: $29.00. ISTE members may join SIGLogo for $16.00 a year. $13 of this amount is for the subscription to *Logo Exchange* for one year. This publication was produced using Aldus *PageMaker*®.

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

ISTE is a nonprofit organization with its main offices housed at the University of Oregon.

Contents

From the Editor

Is Logo THE Answer?

This term I am teaching a graduate level course on the topic of integrating the computer into the curriculum. This year, instead of subdividing the course by subject area, I have chosen to subdivide the course by kind of software. Thus my syllabus is divided into sections on subject-based tools, generic tools, learner-based tools, and teacher tools. After I briefly explained these four categories to my class, one of my students asked whether a particular piece of software could fit into more than one of the categories I had just described. "Of course," I said, "a word processor easily fits into all four categories." I briefly explained my answer and continued with my course overview.

Later that same day, I was preparing material for this month's *LX* and for some reason, the question from that morning's class popped into my mind. "Why," I asked myself, "did I choose a word processor as my example?" All the students in that class had taken a Logo course in the fall, and haven't we in the Logo community been arguing for years that Logo is a multifaceted tool, useable in a variety of areas of the curriculum? My work on the articles and columns for *LX* ceased and this editorial was born.

Let us, then, look together at my four categories of software and explore whether Logo can, indeed, fill the variety of niches in the classroom that we have been claiming for so long.

We'll begin with subject-based tools. One key characteristic of a subject-based tool is that it is likely to be used by a professional in a particular discipline. Consider some obvious examples. A writer today most likely uses a word processor to do his day-to-day work. A layout artist would probably use a product like *PageMaker*. Architects use software that easily produces and manipulates three-dimensional shapes. A scientist would use software to capture and process data. I'm sure you can add many other examples to the list. Of course many of these professional tools are very high-level, sophisticated software packages that run on quite expensive computers. However, we often use "mini" versions of such software in our classrooms. Thus, while a mathematician might use *Mathematica* to solve complex problems in calculus, we might use a much simpler program to check answers to math problems or produce simple graphs. Similarly, while the professional draftsperson would use a high-level program that drives a sophisticated plotter, we might use a much less expensive, less complex program in a drafting class.

So where does Logo fit into this picture? One of the old arguments against Logo was that no one programmed in Logo in the "real world." How can we imagine that Logo might be a subject-based tool? In a previous editorial (February, 1988) I have argued that today's modern Logos serve as excellent low-level desktop publishing tools. This certainly fits the model of using in the classroom a simpler version of the kind of software professionals use. Those of us who teach math know that turtle geometry can provide some powerful insights into geometrical concepts. Logo can even be a great computational tool to check problems in arithmetic. In science, simple dynamic turtle simulations can allow the study of motion without friction and gravity or the examination of animal behavior. Logo indeed can be used in a variety of ways in a variety of subject areas.

Next in our list of types of software is generic tools. Generic tools include such programs as word processors, spreadsheets, databases, graphics packages, and the like. These are programs written with no particular field or application in mind. They are merely powerful aids in our day-to-day work. Can we think of Logo as a generic tool? If you are a long-time Logo user, you no doubt already *use* Logo as a generic tool. You or your students have at one time or another used the Logo editor as a word processor (whether you have *LogoWriter* or not.) You have used Logo to do minispreadsheet and database activities. And what Logo-using classroom has not seen all sorts of graphics produced for a variety of uses. It seems clear that Logo is often used in the classroom without being directed to a particular subject or speciality area.

Third is the category of learner-based tool. There is no question that Logo fits here. Logo is the ultimate example of a learner-based tool. Judi Harris and Glen and Gina Bull defined this category in the article "The Gears of Childhood" last May (1990) in *LX*. Learner-based tools encourage learner involvement and a three-way interaction among teacher, software, and learner. Here we clearly have no problem at all.

Finally, there is the category of teacher tool. This category includes gradebook programs, attendance programs, quiz and test generators—anything that makes the teacher's endless pile of paper work and record keeping easier. Logo as teacher tool? Why not? I recall that when I was teaching in the public schools I often used Logo in just that manner. If I had a list of grades to average or I needed to scale a test score, I would whip out a short Logo procedure to do the task rather than start a grade book program or a spreadsheet. If I had a quick memo to write, I'd use Logo rather than get out my word processor. It was a great piece of software to always have running on my machine.

"OK," you are thinking, "does she really believe that Logo is all that today's classroom really needs?" I'm sure there are those totally dedicated Logo users who would argue

Continued on page 3, second column

Monthly Musing

Surprise, Surprise!
by Tom Lough

Someone asked me recently, "Tom, you have been using Logo for a long time. Haven't you gotten bored with it yet?"

Not only have I not gotten bored with it, I find that my enthusiasm for this engaging computer language is still growing. It is difficult to get bored with something that is easy to learn (at first), that is both simply elegant and elegantly simple, that is so powerful and configurable, and that is full of surprises for the novice and experienced user alike.

Let me give you an example of a really nice surprise I encountered not long ago. One evening, I was just messing around with the turtle, having nothing particular in mind. You know, sort of doodling with Logo. And then it happened. I got the idea for a different combination of the FORWARD and RIGHT commands, but found myself unable to predict the outcome—well, at least not without some heavy thought on my part. And, even then, the results were quite unexpected!

I typed the commands, then repeated them a few times, and saw something interesting appear on the screen. No, it was more than just interesting; it was surprising. I repeated them a few more times, just to make sure of what I was seeing. It was nothing like I had predicted!

Then I typed the commands again, but made a small change in the turtle's heading. Once again, I was surprised. How could this be happening?

This initial encounter led me into a very stimulating series of personal explorations and discoveries. What fun!

My first inclination was to write a fully developed article explaining in detail how the two commands worked together and demonstrating all the different outcomes. Then I remembered something a student had once said to me: "Mr. Lough, thank you, but I wish you had not shown me that. I really wanted to discover it for myself." I had stolen from that student a precious thing: the opportunity to make a genuine and meaningful nonprefabricated discovery. (Since then, I have tried to be more careful about that sort of thing. Alas, I'm sorry to report that I do not always succeed.)

I would like for you and your students to have this special opportunity, so I'm going to provide you with only the combination of commands and one or two suggestions and then let you go. The combination is the following:

```
FORWARD 15 RIGHT HEADING
```

First, I suggest you try to imagine what drawing this would produce if repeated a few times. It may be easy (but not so interesting) to figure out the drawing if your imaginary turtle has an initial heading of 0. It is a little more challenging for other initial headings.

Next, you may wish to test your predictions. Set the initial heading of the turtle and repeat this combination a few times. Try several different initial headings. I hope you are as surprised as I was!

Later, you may wish to increase or decrease the turn input with something like the following:

```
FORWARD 15 RIGHT HEADING + 1
```

It is possible that many of you have discovered this combination of commands already. If so, good for you. If not, then I hope you and your students have a ball with it. Maybe you would like to write up something for the *LX* about it. In any event, please tell me about the results of your explorations. I'd love to see where your investigations took you.

As always, FD 100!

Tom Lough
Founding Editor
PO Box 394
Simsbury, CT 06070

PS: Would you believe it? This September, the LX begins its tenth year of publication, a milestone for any computer publication. What ideas do you have for a special celebration?

Continued from page 2, second column

that, indeed, Logo is all the software you need. However, I am not one of them. Logo is one piece of an increasingly rich selection of educational software. If we can easily argue (as I did) that a word processor can be used in such a variety of ways in the classroom, then certainly Logo can be used just as widely.

If you are already a long-time Logo user, then I certainly don't need to convince you of the breadth and depth of possibilities with Logo. If you are new to Logo, then you probably have many exciting discoveries ahead of you as you learn the variety of uses to which you can put Logo in your classroom.

Beginner's Corner

Stringing Beads
by Dorothy Fitch

Little children have fun stringing beads. They learn to recognize and create patterns, name colors, and identify sizes. Here's a set of procedures for stringing beads in Logo. The procedures offer three different shapes (squares, circles, and triangles), two different sizes (small and large) and as many different colors or pen patterns as your version of Logo provides. Children can create a string of beads with a pattern, or just compile a random selection of beads.

The procedures' names (which of course you can change to suit your students) are short and easy to remember. To add a bead, you just need to state a size and shape. (The color is optional, and you may not want to introduce it until the children are comfortable with the size and shape letters.)

Here is a sample string of beads with a pattern of beads using different sizes, shapes, and colors (in this case, textures).

```
LG W CIR   SM G TRI   SM E SQU   SM B
  CIR
```

The instruction for this pattern translates to:

Large white circle, small green triangle, small empty square, small blue circle.

Repeat this line twice more for a full string of beads. To do this on the Macintosh, simply press the up-arrow key and press <Enter>. With Terrapin's Apple versions, press Control-P <Return> to repeat the line again. (Other versions of Logo may have a different method for repeating a line.)

The one-letter codes for the color, two-letter codes for the size and three-letter codes for the shape were adopted so that the instructions would be easy to read. Think of how much harder it would be to interpret the instructions if all the codes were just one letter! Having codes with different numbers of letters also makes it easy to keep the attributes separate and less confusing.

The programming of these procedures isn't totally trivial, so you shouldn't expect your youngsters to create them from scratch. If you are new to Logo, simply try entering them and using them with your children. You don't have to understand them to play with them!

If you are interested in how the procedures work, the information included here will help you figure out what is going on. Spend some time looking at how they are written, and you'll see how they work. This would be a great project for experienced students to develop for their younger schoolmates!

The program mostly consists of procedures that are run at top level—at Logo's question mark prompt sign. (They are written using Terrapin's Logo PLUS, but will run, with slight modifications, with any version of Logo. See the end of the column for conversion notes.) Most procedures are simple, with only a few lines of instructions.

Here is an annotated listing of all the procedures and what they do.

START clears the graphics screen and places the turtle at the left side. It also sets up an initial small size and empty (black) color.

```
TO START
DRAW
PENUP
SETXY -130 30
PENDOWN
SM ; small
E  ; empty
END
```

The LG and SM procedures set up global variables for size that are used by the shape procedures.

```
TO LG
MAKE "SIZE 30
END

TO SM
MAKE "SIZE 15
END
```

The shape procedures (TRI, SQU, and CIR) first check to see if the shape is small (True or False is reported by the SMALL? procedure), and if so, uses the ADJUST procedure to move the bead up on the screen to center it along the string. Then it draws the shape according to the size variable and fills it with the current pen color. It then uses the MOVE procedure to move the turtle into position for the next bead.

You can figure out the inputs to MOVE in the TRI procedure by using the Pythagorean Theorem (thinking of the distance the turtle should move to the right as one of the sides of a smaller right triangle) or, perhaps more simply, by trial and error.

```
TO TRI
IF SMALL? THEN ADJUST :SIZE / 2
REPEAT 3 [FORWARD :SIZE RIGHT 120]
FILL.SHAPE
IF SMALL? THEN ADJUST -:SIZE / 2
IF SMALL? THEN MOVE 13 ELSE MOVE 26
END

TO SQU
IF SMALL? THEN ADJUST :SIZE / 2
REPEAT 4 [FORWARD :SIZE RIGHT 90]
FILL.SHAPE
IF SMALL? THEN ADJUST -:SIZE / 2
MOVE :SIZE
END

TO CIR
PENUP
HIDETURTLE
IF SMALL? THEN ADJUST :SIZE ELSE
   ADJUST :SIZE / 2
BACK ( PI * :SIZE ) / 30
PENDOWN
REPEAT 15 [FORWARD (PI * :SIZE) / 15
   RIGHT 24]
FILL.SHAPE
PENUP
FORWARD ( PI * :SIZE ) / 30
IF SMALL? THEN ADJUST -:SIZE ELSE
   ADJUST -:SIZE / 2
SHOWTURTLE
MOVE :SIZE
END
```

The PI reporter helps make the CIR procedure easier to read and understand.

```
TO PI
OUTPUT 3.14159
END
```

The SMALL? reporter tells whether or not the size is small. It is convenient to write a reporter for this job because it stores necessary information without adding another global variable to the workspace.

```
TO SMALL?  ;reports True or False
OUTPUT :SIZE = 15
END
```

FILL.SHAPE is a multipurpose fill procedure because it will fill any of the shapes in this program. The numbers for moving and turning were chosen to work for the square, circle, and triangle, in either size.

```
TO FILL.SHAPE
PENUP
RIGHT 30
FORWARD 5
PENDOWN
PENCOLOR :PC
FILL
PENUP
BACK 5
LEFT 30
PENCOLOR 1
END
```

The ADJUST procedure moves small shapes up a little to be centered on the string of beads.

```
TO ADJUST :SIZE
PENUP
FORWARD :SIZE
PENDOWN
END
```

The MOVE procedure places the turtle in position to draw the next bead. It is included in each shape procedure, which passes the :SIZE variable (or a number based on that value) to become the :DISTANCE variable in this procedure.

```
TO MOVE :DISTANCE
PENUP
RIGHT 90
FORWARD :DISTANCE
LEFT 90
PENDOWN
END
```

Here are one-letter procedures for color. On a black-and-white Macintosh screen, you could use the SETPPATTERN command for different effects.

```
TO E   ;empty (black)
MAKE "PC 0
END

TO W   ;white
MAKE "PC 1
END

TO G   ;green
MAKE "PC 2
END
```

```
TO P   ;purple
MAKE "PC 3
END

TO O   ;orange
MAKE "PC 4
END

TO B   ;blue
MAKE "PC 5
END
```

A Game

This last set of procedures challenges the children to recreate a string of beads. When you type BEADS, the computer draws a string of beads with a pattern and asks if you can make the same string. (Logo doesn't care if your string of beads matches the one it generated; the feedback is strictly visual.)

The BEADS procedure places the turtle, then chooses a random number (2, 3, or 4) to be the number of times the pattern will be repeated (PATTERNNUMBER). Then it computes how many different beads it needs to pick so that the total number of beads will be 12 (a string that will fit neatly on the screen). Then it hands the GETSHAPES procedure the number of beads it needs and an empty list in which to store them. The final list will be called SHAPELIST. Then Logo runs the list as many times as PATTERNNUMBER to draw the string of beads on the screen. Then it gives you a chance to copy it!

```
TO BEADS
START
MAKE "PATTERNNUMBER 2 + RANDOM 3
MAKE "SHAPENUMBER 12 / :PATTERNNUMBER
MAKE "SHAPELIST GETSHAPES :SHAPENUM-
    BER []
REPEAT :PATTERNNUMBER :SHAPELIST
YOURTURN
END
```

The GETSHAPES procedure creates a list of commands that will eventually be repeated to draw the string of beads. It chooses randomly from a list of sizes, colors, and shapes and adds each command to the end of the list (using LPUT). It does this as many times as there are to be shapes in the final SHAPELIST.

```
TO GETSHAPES :NUMBER :LIST
IF :NUMBER = 0 THEN OUTPUT :LIST
MAKE "LIST LPUT PICK [LG SM] :LIST
MAKE "LIST LPUT PICK [W G V O B E]
    :LIST
MAKE "LIST LPUT PICK [CIR SQU TRI]
    :LIST
OUTPUT GETSHAPES :NUMBER - 1 :LIST
END
```

The PICK procedure chooses an item at random from a list. It is used in the GETSHAPES procedure.

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

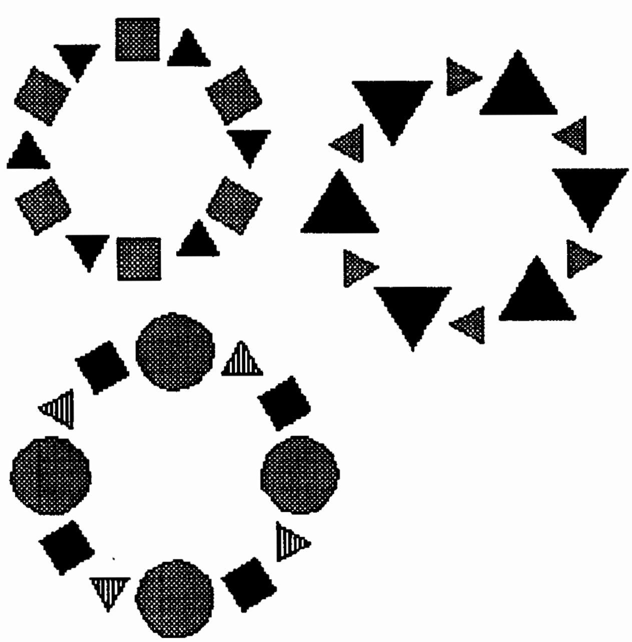
The YOURTURN procedure simply gives instructions and positions the turtle so you can recreate the same string of beads.

```
TO YOURTURN
PRINT [Can you make the same string
    of beads?]
PENUP
SETXY - 130 ( - 40 )
PENDOWN
END
```

These procedures do not limit you to copying the computer's string of beads. You can create a string of beads on your own, then type YOURTURN for a partner to copy it.

Modification

If you change the LEFT 90 command in the MOVE procedure to LEFT 60, you can use the same shape commands to create fascinating round designs like these:



Logo LinX

Conversions

If you are using other versions of Logo, follow these guidelines to get the procedures running:

Use ClearGraphics instead of DRAW.

Use square brackets around the resulting action(s) in an IF statement. Examples:

```
IF SMALL? [ADJUST :SIZE / 2]
IF SMALL? [MOVE 13] [MOVE 26]
```

If your version of Logo does not have a SETXY primitive, add this procedure:

```
TO SETXY :X :Y
SETPOS LIST :X :Y
END
```

Terrapin Logo for the Macintosh has both a PI and a FILLSH command, so you can eliminate the PI and FILLSHAPE procedures and change the line that draws each shape to this:

```
TRI: FILLSH [REPEAT 3
     [FORWARD :SIZE RIGHT 120]]
SQU:FILLSH [REPEAT 4
     [FORWARD :SIZE RIGHT 90]]
CIR: FILLSH [REPEAT 15
     [FORWARD (PI*:SIZE)/15 RIGHT 24]]
```

Happy stringing!

A former education and computer consultant, Dorothy Fitch has been the Director of Product Development at Terrapin since 1987. She can be reached at:

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

Rhyme Paradigm
by Judi Harris

What two rhyming words are described by this phrase?

"an ardent dam builder with a long flat tail"

Here's a hint: this is a HINK-Y PINK-Y, as opposed to a HINK PINK or a HINK-IT-Y PINK-IT-Y.

That's right; the dam builder is an "eager beaver." You may have used the expression many times without realizing that it belongs to an entertaining and pedagogically powerful phrase type.

Try this HINK PINK:

"a petty quarrel between two winged insects"

(gnat spat)

Or this HINK-IT-Y PINK-Y:

"someone who has farther to travel to get to the polls"

(remoter voter)

Enabling Labels

As you probably have guessed by now, the names for these delightful word puzzles indicate the number of syllables in each part of the correct response. For example, this HINK-Y PINK-IT-Y:

"an insect which gathers a substance used to make honey"

is correctly solved with two rhyming words, two and three syllables long, respectively.

(HINK-Y: nec-tar; PINK-IT-Y: col-lec-tor; nectar collector)

Now try this HINKY PINKY:

"a sailing vessel on a voyage to the earth's satellite"

(Try to solve this one without peeking at the answer!)

Pedagogic Logic

HINKY PINKY puzzles can be used to help students explore new vocabulary, synonyms, syllabication, and rhyming patterns. "Hinky Pinky," an excellently designed piece of educational software by Learning Well, Inc., gives users HINKY PINKY definitions in their choice of three difficulty levels. It will supply instructionally sound hints and prompts to assist the learner's deductive reasoning processes as s/he attempts to supply the two rhyming words that fit each definition. Although it is certainly possible to program the computer to do this in Logo (and that challenge might make for some interesting explorations with list manipulations), the "Hinky Pinky" program is so well done that I would recommend purchasing it for your students to use.

But, wait! Lest you unjustly accuse me of

"a chronic inability to use the right words when speaking or writing"

(diction affliction),

I should hasten to add that Logo can also be used to inspire some fascinating INDUCTIVE work with HINKY PINKIES.

Induction Production

Examine these puzzles and their solutions for word type patterns:

"a person who inspects sausages and removes the bad ones" (wiener screener)

"the sound heard when a marigold bomb detonates" (bloom boom)

"a horror-struck group of actors" (aghast cast)

"a physical education building with poor lighting" (dim gym)

Did you notice that the rhymed word solutions were either two nouns or an adjective followed by a noun? We can capitalize upon that regularity as we write Logo code that will generate new HINKY PINKIES.

Code Mode

A rhyming dictionary can assist your students' collection of groups of rhyming words. Separate the words into two groups: nouns and adjectives. Discard any of other word types. Using the PICK tool,

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

write two procedures that will supply the program with structured random choices of the word collection.

```
TO EEK.NOUN
OUTPUT PICK [ANTIQUE BATIK BEAK CREAK
    CREEK CRITIQUE FREAK GREEK LEAK
    LEEK PEAK PEEK PHYSIQUE SHEIK
    SHRIEK SNEAK SQUEAK WEEK]
END

TO EEK.ADJ
OUTPUT PICK [ANTIQUE BATIK BLEAK CHIC
    FREAK GREEK MEEK OBLIQUE PEAK
    SLEEK SNEAK TEAK UNIQUE WEAK]
END
```

Note that some of the words can serve both as adjectives and nouns, and therefore are placed in both groups.

Now we can tell the computer to select two words and concatenate them, according to the noun-noun or adjective-noun pattern identified earlier.

We can produce either a noun-noun (.NN) HINKY PINKY:

```
TO BEGET.NN
PRINT SENTENCE EEK.NOUN EEK.NOUN
END
```

or an adjective-noun (.AN) HINKY PINKY:

```
TO BEGET.AN
PRINT SENTENCE EEK.ADJ EEK.NOUN
END
```

If you type BEGET.NN, the computer may return

```
LEEK WEEK.
```

If you type BEGET.AN, it may respond with

```
GREEK PHYSIQUE.
```

Definition Renditions

Let's assume that the computer has just generated UNIQUE ANTIQUE. You can make it remember a user-supplied definition with a DESCRIBE tool.

```
TO RENAME :LIST
OUTPUT (WORD FIRST :LIST ". LAST
    :LIST )
END
```

```
TO DESCRIBE :LIST
PRINT (SENTENCE [TYPE THE CLUE FOR]
   :LIST ". )
MAKE RENAME :LIST READLIST
END
```

RENAME takes the rhyming two-word list as input and concatenates the two words into one unit by connecting them with a period. DESCRIBE then stores a user-supplied clue in memory under the newly formed name. In this instance, if the user types

```
DESCRIBE [UNIQUE ANTIQUE]
```

the computer will print

```
TYPE THE CLUE FOR UNIQUE ANTIQUE.
```

to which the user could respond

```
AN ARTIFACT OF WHICH NO DUPLICATE
   EXISTS.
```

This description is stored in memory as the value of the global variable UNIQUE.ANTIQUE . The CLUE tool,

```
TO CLUE :LIST
IF NAMEP RENAME :LIST
   [PRINT (SENTENCE :LIST "IS THING
   (RENAME :LIST)]
   [DESCRIBE :LIST]
END
```

first checks to see if the list is already defined. If it is, the computer prints the existing definition. If not, CLUE executes the DESCRIBE procedure, explained above.

Perusing Uses

HINKY PINKIES are welcome mid-winter language arts activities. There are basically three ways to present the puzzles to students. The clue (definition) can be supplied, and the children can be asked to deduce the corresponding two-word rhyme. This process encourages attention to syllabication patterns since the name of the puzzle (HINKY PINKY, HINKITY PINKITY, HINK PINK, etc.) is a valuable clue to the number of syllables in the solution. Clues can be written using new vocabulary words:

"a singing group with a full, rich timbre"

and solutions can encourage dictionary usage.

(sonorous chorus)

Students could, instead, be supplied with two-word rhymes, and challenged to write definitions, which could then be exchanged to be solved. This might very well make enthusiastic thesaurus users of even your most reluctant language artists!

Finally, HINKY PINKIES can be used inductively as students create their own rhyming puzzles, both with and without computer assistance. Clue-writing provides a meaningful purpose for synonym searches with dictionaries and thesauri. Solution-writing encourages attention to rhyming and syllabication patterns.

Addiction Restrictions?

I would be remiss, though, if I didn't warn you of the potential dangers of HINKY PINKITY use. Forewarned is forearmed. There may be

"explosive sounds of amusement forever more."

(laughter hereafter)

Is *that* within the boundaries of your (curricula dicta)?

Judi Harris works as an assistant professor of educational technology in the Department of Teacher Education at the University of Nebraska at Omaha. She has also coordinated the field components of a telecommunications network for teachers, student teachers, students, and professors, and served as an elementary school classroom teacher, mathematics specialist, and computer use facilitator.

Internet: JHarris@Zeus.unomaha.edu
BitNet: JHarris@unoma1
CompuServe: 75116,1207

A previous version of this article appeared in the February 1988 issue of Logo Exchange.

Questions Please!

Questions and Comments: Pleas and Thank Yous
by Frank Corley

This month's column comes directly from my school's department members and me. Like most Logo teachers, we labor in relative isolation, crazed voices in a wilderness of computer ignorance, phobia, and BASIC. We have some ideas, and we do try to execute them. But we are continually unsure if our ideas are the best ones and if we execute them well. This month's column will be full of questions in two senses. The first sense is that we have some specific questions about Logo theory and practice, if this topic can be addressed absolutely. The second sense is that we are in need of direction in our approach to Logo use, and would welcome the comments of other Logo users. I am certain that we are representative of many departments around the nation and the world using Logo, so the column is not at all entirely self-serving. I am also fairly certain that we may be doing some innovative, or at least positive, things with Logo, so this column may also serve to answer some questions of others.

We are a department of mathematics and computer science in an independent school that includes students in grades seven through twelve. As such, we are lucky enough to teach some fairly able students. We operate two Apple computer labs, which provide a computer for each student during our allotted two days per week, and some limited out-of-class time. Four of us teach computer science and math. Admittedly, we may be making a mistake trying to teach computer science and computer programming. But we believe that problem solving is an important skill and programming in Logo is an ideal context in which to teach problem and solution analysis. We also admit that we may be making a mistake trying to fit the traditional Algebra-Geometry-Algebra II-Trigonometry paradigm into the Logo educational philosophy. We do have constraints on our curriculum, but we want to enrich it as much as possible.

In the eighth grade, we teach an introduction to computer science, with an emphasis on programming. We use Brian Harvey's texts for this with the honors third of the class. *Half-rhetorical question: Is there a better book?* We love this book as teachers but have had trouble turning students on to his approach. They are stunted in their intellectual growth by years of deprivation from inquiry. *How do we get students fired up for such a free approach?* We have had to do a significant amount of outside work with this text, learning a lot along the way. *Are there sources of exercises associated with this text?* In the regular two-thirds track, made up of average-ability students, we use a new curriculum from Terrapin in association with a student text from West Publishing Company. *Not a rhetorical question: Can we do better? Are there textbooks out there written for teaching Logo that are sophisticated yet accessible?*

Our teachers have a number of questions about Logo. Here are a few of them.

- *I want to use local variables in my Logo procedures, but I do not see a way. Can I?*
- *Along the same lines, is there a way of defining a procedure within another procedure in Logo without using TO, which always sends me right into the editor upon execution of the top-level procedure?*
- *Computer programming as a high school course has largely gone out of vogue. Are we old-fashioned, or forward-thinking? Is there a place for this discipline in schools?*
- *Is there a version of Logo that implements a CASE statement in some form?*

In the ninth grade, the computer classes are associated with the geometry class, and we try to teach mathematical content through turtle graphics and other Logo activities. Again, the honors students, who are in a calculus-directed track, use a different text than the regular track. The honors students use Abelson and diSessa's *Turtle Geometry* book, and the regular track uses a series of Logo activities I am developing that investigate coordinate and transformational geometry through turtle graphics.

We have really struggled to use the Abelson and diSessa book, trying to get through the first four chapters meeting twice a week for thirty weeks with some very bright students. *Has anyone succeeded in teaching this course at this level and, if so, can you give us some assistance? Has anyone else who has tried and failed like to commiserate? Has anyone developed a series of activities that work through this book, maintaining the significant level of intellectual sophistication and yet making the material comprehensible to high school students?*

Two very simple characteristics of Logo make it particularly well suited for my use in geometry class. The first is that the graphics screen has the origin at the center. The second is that nearly everything can happen with FORWARD/BACK or RIGHT/LEFT. The former makes coordinate geometry, perhaps the most powerful topic in a geometry course, exceedingly easy to study using Logo. The latter indicates that every

Logo Connections

move of the turtle is a rotation or a translation, which makes Logo suitable for the elegant study of transformational geometry. Thus, I have developed a series of Logo programs, which I write in cooperation with the class, that explore many of the major topics in the typical high school geometry class. *Is anyone else involved in such activity? Surely there are other such "packages" existent, but is anyone interested in these?* We find ourselves, as math and computer science teachers, in an ironically uncomfortable position. We go to Logo conferences and because Logo has powers and applications throughout the curriculum, no one seems to want to talk about mathematics and programming. We go to math conferences and because no one programs anymore and "Logo is a grade-school language," no one wants to talk to us there either. We know about the Council for Logo in Mathematics Education but have not seen that much of it. *Is there a community of Logo users who are also high school mathematics and computer science teachers, and if so, what is the professional activity in this hybrid field?*

After the ninth grade, our formal computer science instruction ceases until the twelfth grade Advanced Placement course. The Pascal teacher tends to think of Logo as a toy language for drawing pretty pictures. *How does your school make the transition from programming in a functional language like Logo, taught to younger students, to a procedural language like Pascal taught at a higher level? Should an attempt even be made at transition, or should the AP teacher simply ignore earlier courses?* In the tenth and eleventh grades, we would like to do two things with computers and with Logo. The first is that we would like to teach a course in "quantitative reasoning." We would like to use computers and programming as the context for this course in "problem solving." *Does anyone have any ideas on or experience with this sort of course at the high school level?* The second thing we would like to do is continue to use Logo in the math classrooms. We have seen Albert Cuoco's and Philip Lewis' new texts from MIT. *Are those books for real; has anyone besides the authors really used them? Can AP Calculus follow Approaching Mathematics Discretely? Are there other such efforts underway?*

Thank you for indulging us in this inquiry. We are a new department with a lot of ideas but willing to try almost anything. We would love to see answers or pointers to answers personally or in this journal. Next month we'll be back to the old format. Until then, send your questions and responses to me.

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

Logo and Videodiscs

by Glen L. Bull and Gina L. Bull

In an essay written before microcomputers existed, Seymour Papert describes a hypothetical tour that might be given to visitors in the future to describe instructional uses of computers. At the end of the tour, a robot (possibly an ancestor of the Logo turtle?) rolls over to a film projector and presses the "On" switch to present a film about other educational uses of computers.

Today most film projectors have been replaced by videotape players in schools. The Logo turtle can now initiate a video presentation, and even pause it in places for questions if desired. This is achieved by sending electronic commands to a videodisc player rather than by physically pressing the "On" switch of a film projector. However, the end effect is the same, or possibly even better. It is not possible to go directly to a particular frame on a film and display it; the projector bulb will burn the film if it remains still at any one point. Any frame of a videodisc can be displayed continuously for as long as desired, and it is possible to go directly to a specific frame, bypassing all the others.

There are videodiscs which contain thousands of paintings from museums such as the National Gallery of Art and the Louvre. Other videodiscs contain images of thousands of plants, animals, and ecosystems for use in life science classes. There are videodiscs from the space program and videodiscs for the physical sciences. Formerly, teachers were limited to a few hundred illustrations in books and 35 mm slides that they might have accumulated for their classroom. Now a teacher can have thousands of images at her fingertips on a single videodisc.

Logo Multimedia

The term "multimedia" literally means use of "more than one medium." The use of computers and video together has received a lion's share of attention in the popular press. Videodisc players are well suited to use with computers because the computer can be used to access any of more than 50,000 frames on the videodisc. At one time, videodisc players were relatively expensive, but the price has been dropping in recent years. Currently the Virginia state contract price for a Pioneer 2200 videodisc player is about $650.

The decrease in price combined with the inherent instructional potential means that videodisc players are becoming more widely available in schools. Virginia's educational technology standards call for placement of a computer-controlled videodisc player in every elementary and secondary library by

1994, as well as a ratio of one player to every eight classrooms. If your school system is purchasing several computers with printers, you may wish to consider suggesting substitution of a videodisc player for one of the printers.

Questions Please!

Logo can be used for control of videodisc players. We have been using Logo with videodisc players since the mid-1980s. Recently, a reader sent a request to Frank Corley's "Questions Please!" column, asking us to describe how to write Logo procedures to control a videodisc player. We are pleased to comply forthwith. (Frank—we do read your column and we *are* listening!)

Videodisc players such as the Pioneer 2200 are connected to the computer through a cable plugged into the serial port. This particular videodisc player is widely used in schools because of its low cost, and because it can be controlled in three different ways:

1. through a remote control, like a videotape player,
2. through a bar code reader, and
3. through a computer program such as Logo.

Since the commands for the Pioneer 2200 and the Pioneer 4200 videodisc players are equivalent, the methods described below will work for either player.

Multimedia Approaches in Other Programs

Commands are sent to the videodisc player through the serial port by the computer. A list of these commands is provided in the videodisc manual. For example, to cause the Pioneer 2200 to beginning playing, the code "PL" is sent to the player. The videodisc player has a microprocessor, which in turn interprets the code and takes the appropriate action.

In order to see why Logo is so well suited for use with videodisc players, it is instructive to look at other hypermedia programs designed for use with videodisc players, such as *HyperScreen* or *TutorTech*. These are *HyperCard*-like programs for the Apple II computer. In *TutorTech* the following command would be entered to send the play command to the videodisc player:

```
•2 PL
```

A bullet (the round dot printed by holding down the Option key and typing the number "8" in *TutorTech*) is entered, followed by the number of the serial port, and the command "PL".

The PLAY Command

The *TutorTech* method seems to work. *TutorTech* is a successful hypermedia program that is widely used. However, there is an even easier way in Logo. In Logo, a PLAY procedure can be created so that it is only necessary to type the word PLAY.

```
TO PLAY
SEND "PL
END
```

This follows the Logo philosophy of substituting an English word for a more obscure computer command.

Sending the Command to the Videodisc Player

In the example above, SEND is not a built-in Logo command. Rather, it is a Logo procedure that must be written differently for each version and dialect of Logo. This is necessary because the command to send text to the serial port is different for each version of Logo. (Obviously, this incompatibility makes life much more difficult for Logo users, but it is a fact of Logo life.) In fact, the command to send text to the serial port is not even documented in the *LogoWriter* manual.

However, not to worry. Last week we were at a conference, and by coincidence Michael Tempel was scheduled to present in the same room. He said that Logo Computer Systems, Inc. (LCSI) will send users the *LogoWriter* procedures to control the Pioneer videodisc player if a floppy disk with a self-addressed return mailer is sent to LCSI technical support. Upon verifying this with LCSI technical support, we found that this offer applies only for the Apple version of *LogoWriter*, since the *LogoWriter* videodisc procedures for the IBM version are evidently not in the public domain at this time.

In any event, to be faithful to the spirit of the question asked in Frank Corley's column, we are going to describe all the procedures we use to access the Pioneer 2200 videodisc player with the Apple version *LogoWriter*. If you like, you can obtain the Apple *LogoWriter* videodisc procedures for the Pioneer on a disk from LCSI. Keep in mind, however, that we developed our procedures independently of LCSI, and so our procedures probably differ in some details from theirs.

The *LogoWriter* command to send a character to the serial port is the .OUT command. Since this is an undocumented command (not described in the manual), it means that LCSI is free to change the command in future versions of *LogoWriter* if they like. It also probably means that they thought the average user was unlikely to need this command. However, some means of sending a character to the serial port is essential to talk to the videodisc player.

The .OUT command has a couple of limitations in comparison with serial commands in other versions of Logo, such as *LCSI Logo II* for the Apple and *Terrapin Logo*. It can only send one character at a time to the serial port, and the ASCII equivalent of the character rather than the character itself must be sent. (ASCII stands for "American Standard Code for Information Interchange.") Fortunately, the flexibility of Logo means that these limitations are easily overcome through a procedure such as the following:

```
TO SEND :COMMAND
IF EMPTY? :COMMAND [.OUT 13 STOP]
.OUT ASCII (FIRST :COMMAND)
SEND BUTFIRST :COMMAND
END
```

For those interested in the details of the procedure, this three-line procedure is a traditional Logo list-processing program. The line

```
.OUT ASCII (FIRST :COMMAND)
```

sends the ASCII equivalent of the first character of the command out through the serial port. The next of line of the procedures calls SEND again, inputting everything *but* the first character of the command (which has already been sent). This continues until the command is empty and all the characters have been sent. When the last character has been sent, the procedure sends the ASCII equivalent of a carriage return (the number "13") to the serial port and stops.

If all this seems unduly technical, do not be concerned. The advantage of Logo is that it is possible to use complicated procedures just as though they were built-in commands. In a column on videodisc players last year we did not describe some of the lower-level details for fear that some readers would think videodisc players were complicated (they're not) and simply referred readers to LCSI for the procedures on the disc. This year we are including these lower-level details for the benefit of Logophiles, but want to be sure that we do not discourage others. If the code for the FORWARD procedure were printed, it might appear foreboding as well, but thousands of elementary children use the FD command itself without difficulty. In the same way, it is possible to type PLAY to start the videodisc playing without being concerned about details of lower-level procedures.

The *LogoWriter* serial command described above has one other limitation—by default it always sends the character to the serial card in slot 1 of the Apple. In many cases this limitation does not present a problem. However, in other instances a printer may already be attached to the serial card in the first slot, or this slot may be preempted to connect the computer to an Appletalk network. For those instances, LCSI has a multislot driver that can be used to direct the text to a serial card in any slot. For those who need it, this enhancement may also be obtained through LSCI technical support.

SEND in Other Logo Dialects

The SEND procedure is considerably easier to write for other versions of Logo. In Apple Logo II the procedure would be written in this way:

```
TO SEND :COMMAND
DRIBBLE 1
PRINT :COMMAND
NODRIBBLE
END
```

In Terrapin Logo the SEND procedure would be written in this way:

```
TO SEND :COMMAND
.OUTDEV 1
PRINT :COMMAND
.OUTDEV 0
END
```

The serial command is different in almost every version of Logo. This has hindered sharing of procedures that make use of peripheral devices such as videodisc players, and is one of the reasons we have not listed specific codes for these types of procedures more often. It is said that God created different human languages to disrupt cooperation in construction of the Tower of Babel, and the various dialects of the serial command have had the same effect in Logo. It is virtually impossible to account for every variation for every version of Logo on every brand of computer. However, once you develop a SEND command for your version of Logo, you will easily be able to share these kinds of procedures with friends who have developed a SEND procedure for their version.

The FIND and HALT Commands

With the SEND procedure defined, we can continue with exploration of additional videodisc player commands. The Pioneer command to stop the videodisc player is ST. Since STOP is already a Logo command, we used HALT to stop the videodisc player:

```
TO HALT
SEND "ST
END
```

The Pioneer 2200 command to search for a particular frame on the videodisc consists of the frame number followed by the letters SE. The Logo command to find a particular frame might look like this:

```
TO FIND :FRAME
SEND WORD :FRAME "SE
END
```

Once this procedure is defined, frame 23712 on the videodisc could be located by typing the following:

```
FIND 23712
```

With hundreds of slides on the videodisc, it might be difficult to remember the number of a specific slide. However, it is simple to write a Logo procedure that will remember for us. For example, if frame number 23712 is a picture of a robin, the following procedure could be entered:

```
TO ROBIN
OUTPUT 23712
END
```

This procedure would make it possible to enter the following Logo command:

```
FIND ROBIN
```

When this is entered, the videodisc would then find frame 23712 and display a picture of a robin.

Even the youngest elementary students will be able to type FIND ROBIN once the proper Logo tools are created. One version of a "Turtle Town" has even been set up by one teacher so that when the turtle enters a particular area of the town, a corresponding image on the videodisc player appears.

Playing a Video Segment

One other procedure frequently employed by teachers who use these videodisc tools is the SEGMENT command. This command asks to the videodisc player to play a video segment, beginning at one frame and ending at another. In true Logo fashion, this procedure builds upon previously defined procedures, such as FIND:

```
TO SEGMENT :START :END
FIND :START
SEND WORD :END "MF
END
```

The SEGMENT procedure can be used to play various segments on the videodisc. For example, to show the space shuttle lifting off on a NASA videodisc, the following Logo procedure could be written:

```
TO LIFTOFF
SEGMENT 1120 11840
END
```

Using the Videodisc Tools

When you first turn on the videodisc player and insert a videodisc, you will need to start the player. You can do this either by manually pressing the Play button on the front panel of the videodisc player or by creating the following procedure to initialize the videodisc player after you first insert a videodisc:

```
TO INIT.VIDEO
SEND "SA
END
```

After a moment some video should appear on the monitor attached to the videodisc player. If you encounter problems, you may want to refer to the Troubleshooting Tips which we have included at the end of the column.

Once you have developed your videodisc procedures, you will want to create an instruction sheet that describes the various commands for others. For example, the FIND and SEGMENT procedures are described in the following way in our user's guide:

FIND — The FIND procedure instructs the player to search to the frame number given.

Example: FIND 1325

It is also possible to write procedures that name videodisc frames.

Example:
```
TO ZEBRA
OUTPUT 1575
END
```

Once a frame is named, the FIND command can be used to locate that frame.

Example: FIND ZEBRA

SEGMENT — The SEGMENT procedure instructs the videodisc player to begin playing at a starting frame number and to stop playing at an ending frame number. It requires inputs to specify the starting and ending frames.

Example: SEGMENT 1000 2000

An instruction sheet or user's guide will allow others to use your videodisc procedures even if you are not around to explain how they work.

There are many other videodisc commands that can be developed. You can create commands to play slow motion, to play reverse video, to turn the audio tracks on and off, to display index numbers associated with each video frame, and to overlay text on the videodisc picture. Space does not permit inclusion of these procedures here. However, once you understand the process of creating Logo procedures to send commands to the videodisc player, you can create a procedure for any command described in the videodisc manual.

Logo: A Multimedia Language

Thousands of teachers are currently purchasing hypermedia programs, such as *HyperStudio*, *HyperScreen*, *TutorTech*, and others. We have used all of these and can attest to the fact that they are well-designed programs. However, teachers are often not aware that Logo can also be used to access videodisc players. Readers of the *Logo Exchange* know this, of course, but many other users have no way of discovering this.

Users of those other programs are aware that they are "videodisc ready" because this information is included in the manual. The manuals for programs such as *HyperScreen* and *TutorTech* simply provide a page listing the various videodisc commands and ask the users to enter the commands directly. Logo has the additional capability for creation of procedures to substitute English words such as "PLAY" in place of videodisc commands such as "PL".

Logo vendors such as Terrapin and LCSI could include similar information about the videodisc capabilities of their products directly in the manual, possibly including in a sample file videodisc procedures such as the ones described here . As you have seen from the description above, these procedures are short and would easily fit in a samples file.

The videodisc capabilities of a number of other hypermedia products are well known and receive prominent attention in their respective manuals. Provision of similar information in Logo manuals would heighten awareness of similar capabilities within Logo at very little added cost to the vendor. Logo *is* a true multimedia language.

Glen and Gina Bull
University of Virginia, Charlottesville, VA 22903
Internet/BITNET Addresses:
Glen: GBULL@Virginia Gina: GINA@Virginia

Troubleshooting Tips for Videodisc Players

Videodisc players such as the Pioneer 2200 are connected to the computer through a serial port. If you purchase your videodisc player through a dealer, he should set up the videodisc player so that it works properly. You can realize considerable savings by purchasing the videodisc player through mail order, but if you do it may be necessary to conduct trouble-shooting steps such as the ones below if a problem is encountered.

- *Be sure you have the proper cable.*

The cable should have the proper connectors at each end, or it will not be physically possible to plug it in. Even if the connectors physically match, it is possible that they will not be wired correctly internally. Serial cables come in two principle types: printer cables and modem cables. Sometimes they may look the same externally even though they are wired differently inside the cable. Your videodisc dealer should be able to supply the proper cable for your computer. If there is a problem, however, information about the proper cable can in most cases be obtained directly from the videodisc manufacturer .

- *Be sure the speed (baud rate) of computer's serial port matches the speed setting of the videodisc player.*

The speed determines how fast information is sent from the computer to the videodisc player. Usually this speed is set to either 4800 or 9600 bits per second. The important thing is that the computer and the peripheral device both be set to the same speed. On an Apple II computer, the speed is set through switches on the serial card. On the videodisc player, the speed is usually set through switches behind a panel at the front or back of the player.

- *Check to see if the videodisc player works in a stand-alone mode.*

A videodisc player can be operated through controls on the front panel, in much the same way that a videotape player is operated. If the videodisc player is not responding to computer commands, try operating it from the front panel controls to be sure that the player itself is operating correctly.

- *Try the player with another program.*

If you continue having difficulty controlling the player with your version of Logo, try borrowing another program, such as *HyperScreen* or *TutorTech*, to control the player. If the player operates with these programs, this indicates that you may need to look for a typographical error in your Logo program. If the player does not work with these programs, you may need to look for a hardware problem, such as an incorrect cable or a mismatched baud (speed) rate setting.

MathWorlds

edited by A. J. (Sandy) Dawson

The posthumous publication in 1976 of Imre Lakatos's classic work *Proofs and Refutations* heralded the beginning of a renewed interest in the foundations of mathematics education, which in turn led to lively discussions about the basis for mathematics instruction. Lakatos presented his arguments in the form of dialogues, composed of questions and answers. He showed how theorems can in fact be generated by proofs, and how testing of proofs is an essential part of the growth of mathematical knowledge. Others have extended Lakatos work and have suggested that the naive generation of hypotheses, and their testing, is very much a part of how children can be helped to acquire mathematical understanding (Dawson). However, this approach has not been applied to the teaching and learning of mathematics where Logo is used as a tool for the knowledge generation. At least not up until now, because in the material presented below, Rina Zazkis takes a very decidedly Lakatosean orientation to raising questions about and providing proof-generated answers for that very familiar Logo topic of star polygons. In doing so, Zazkis takes us beyond the usual *aha* experience that children have when seeing the lovely stars produced on the screen, into the world of mathematical proof and justification.

Star Polygons: More Questions and Answers

by Rina Zazkis

Let us start by considering this question: *How is it possible to create an 11-sided star polygon with turtle graphics?*

Working with Logo, teachers and learners frequently draw 5-sided, 10-sided or 12-sided star polygons but almost never the 11-sided one.

In these notes I will discuss the above question and then conclude with a general idea about the creation of star polygons.

I would argue that it is helpful for the learner and beneficial to the teacher to explore different aspects of the same geometrical topic and discover how they fit together. Star polygons serve as one of the bridges assisting students in relating the concepts of "standard" (Euclidean) geometry to those of the Turtle geometry. The ideas described here integrate the activities of a geometry class with a "hands on" approach to the computer as well as a "hands off" approach.

The usual way to create both simple and star polygons in Turtle geometry is by using the famous POLY procedure:

```
TO POLY :SIDE :ANGLE
FORWARD :SIDE
RIGHT :ANGLE
IF HEADING = 0 [STOP]
POLY :SIDE :ANGLE
END
```

The input to :ANGLE determines the shape of the given polygon. Angles of 90, 72, and 60 produce a square, a pentagon, or hexagon, respectively; while angles of 144, 108, and 150 produce, respectively, 5-sided, 10-sided, and 12-sided stars.

The usual approach to polygons using POLY is presented in Abelson and diSessa's *Turtle Geometry* (1981). The following problem is raised and discussed there: *Given the input to :ANGLE in POLY, determine the shape of the resulting polygon, i.e., how many vertices (sides, angles) will the polygon have?*

The answer is given by considering the total turning of the turtle from two different perspectives. On the one hand, the drawing of POLY is completed when the turtle has returned to the initial heading, i.e., has turned a multiple of 360 degrees. On the other hand, the drawing of an n-sided POLY is completed after the turtle has turned n * :ANGLE degrees.

Therefore, n * (:ANGLE) = 360 * R

We are looking for the smallest positive integer R to satisfy the equation, i.e., the first time the heading change reaches a multiple of 360.

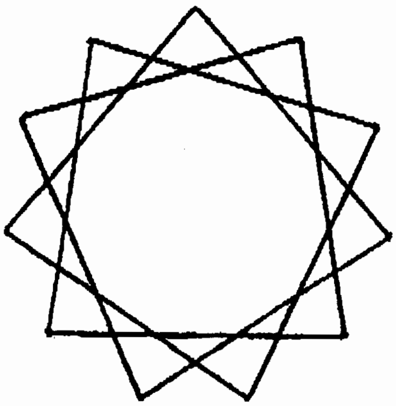
Hence, 360 * R is equal to the Least Common Multiple of :ANGLE and 360, which can be denoted as LCM (:ANGLE, 360) .

Substituting this in the above formula we see that n, the number of vertices of POLY :SIDE :ANGLE, is given by LCM (:ANGLE, 360)/ :ANGLE. This determines the shape of the polygon when the input to :ANGLE is given.

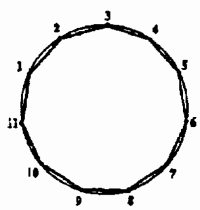
The problem raised at the beginning of this column is the reverse question: How is it possible to draw an 11-sided star polygon?

We can reword this question in terms of POLY: *What input should be given to POLY to create an 11-sided star?* In general, how can we find the :ANGLE when the shape is given?

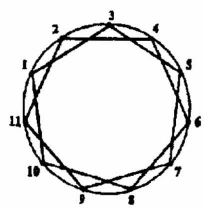
Before trying to answer this, let us see whether the question is well defined. *First, we need to know if an 11-sided star polygon exists!* (Your students probably know just from experience that there is no 6-sided star polygon). The answer is "yes," as shown below.



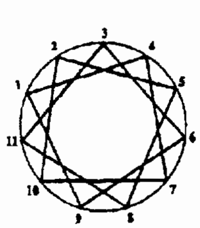
Second, we want to determine whether there is only one 11-sided star polygon. To answer this, start with a paper-and-pencil construction of star polygons. Take a circle and mark 11 points on it, such that all the arcs are equal. Choosing a direction (clockwise) and any point to start with (point #1), connect the points by line segments to get a simple 11-sided polygon, as shown below.



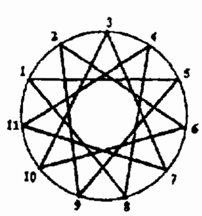
(1)



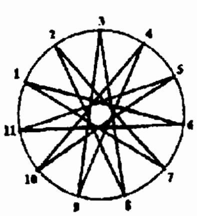
(2)



(3)



(4)



(5)

Now take another circle with 11 equally spaced points and connect the points, but this time skip one each time until you reach the initial one. The sequence of points would be 1,3,5,7,9,11,2,4,6,8,10,1. (Drawing 2 above.) Denote this polygon as <11,2>. Following the same method, <11,3> is drawn by connecting to the third point each time, i.e, going through the sequence 1,4,7,10,2,5,8,1,13,6,9,1. Other possibilities are shown in the table below. Clearly, there is more than one 11-sided star polygon. The question of how many there might be is left until later, because now I want to focus attention on how Logo can be used to create an 11-sided polygon.

Formal notation	Sequence of points	Picture of Polygon	Input for :ANGLE in POLY
<11,1>	1,2,3,4,5,6,7,8,9,10,11,1	1	360/11
<11,2>	1,3,5,7,9,11,2,4,6,8,10,1	2	(360*2)/11
<11,3>	1,4,7,10,2,5,8,11,3,6,9,1	3	(360*3)/11
<11,4>	1,5,9,2,6,10,3,7,11,4,8,1	4	(360*4)/11
<11,5>	1,6,11,5,10,4,9,3,8,2,7,1	5	(360*5)/11
<11,6>	1,7,2,8,3,9,4,10,5,11,6,1	5	(360*6)/11
<11,7>	1,8,4,11,7,3,10,6,2,9,5,1	4	(360*7)/11
<11,8>	1,9,6,3,11,8,5,2,10,7,4,1	3	(360*8)/11
<11,9>	1,10,8,6,4,2,11,9,7,5,3,1	2	(360*9)/11
<11,10>	1,11,10,9,8,7,6,5,4,3,2,1	1	(360*10)/11

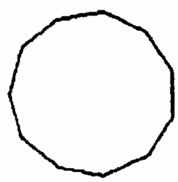
First we have to decide which one of the 11-sided star polygons we would like to draw. Once having done that, we are ready to deal with the original question.

Think of *how the turtle would create the drawings shown above*. Follow Papert's advice and play turtle—let the turtle follow your pencil when connecting the points. When drawing the usual convex polygon, the total turtle trip involves turning 360 degrees, a complete circle. Since it is done in 11 steps, the turning in each step is 360/11. In the second case, where we skipped a single vertex each time, the turtle turned twice as much before returning to the starting point. In this case the turtle completes two full circles, i.e., its total turning is 360*2 = 720 degrees. In the third instance, when connecting to every third point, the turtle completes 3 full circles, i.e., the total turning is 360*3 = 1080 degrees. The pattern seems clear. Knowing the total turning completed in 11 steps makes it easy to calculate the turtle turning needed in each step—we just divide the total turning by 11. The result determines our input to :ANGLE in POLY. To draw <11,2> the :ANGLE should be (360*2)/11, to draw <11,3> the :ANGLE should be (360*3)/11, and so on.

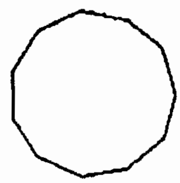
To summarize,

```
POLY <side> (360*k)/11
```

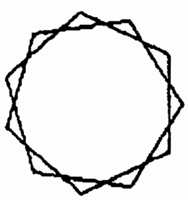
will draw the <11,k> star, where k = {2, ... , 9}. When k=1 or k=10, a simple 11-sided polygon will be drawn.



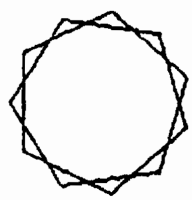
ANGLE = (360*10)/11



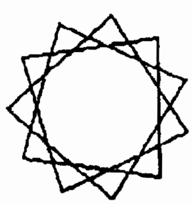
ANGLE = 360/11



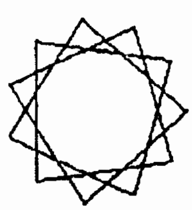
ANGLE = (360*9)/11



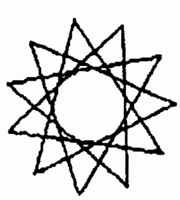
ANGLE = (360*2)/11



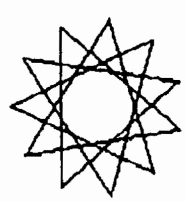
ANGLE = (360*8)/11



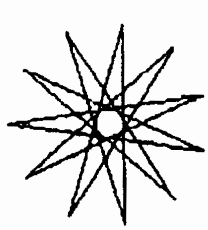
ANGLE = (360*3)/11



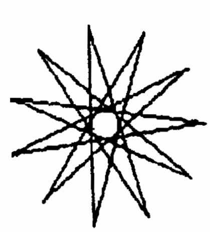
ANGLE = (360*7)/11



ANGLE = (360*4)/11



ANGLE = (360*6)/11



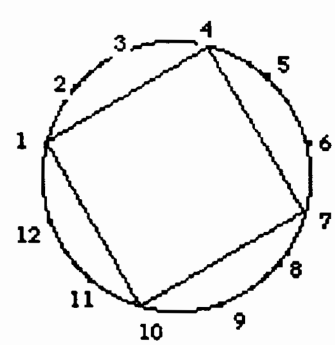
ANGLE = (360*5)/11

Now it is time to ask the question: *how many different 11-sided polygons are there?* Notice that <11, k> and <11, 11- k> are congruent. This can be explained in both turtle and pencil-and-paper terms. Think of the paper and pencil construction: to go to the point k clockwise is the same as to go to point (11-k) counterclockwise. Therefore we get the same picture. Notice also that the sequence of points for creating <11,k> is the reversed order of the sequence for creating <11,11-k>.

If we think in terms of the turtle construction, note that (360*k)/11 + (360*(11-k))/11 = 360. Therefore RIGHT (360*k)/11 has the same effect as LEFT (360*(11-k))/11, which explains the appearance on the screen of congruent star-polygons with opposite orientations. The drawings obtained are the mirror images of one another. So getting back to the original question of "how many"—the answer is 4, namely <11,2>, <11,3>, <11,4> and <11,5>.

What can be generalized from the above ideas? Well, it is now possible to say *how to create any n-sided star polygon with turtle graphics.* You might like to try it on your own before reading on.

In general, <n,k> denotes a polygon that is constructed when n equally spaced points on a circle are joined by connecting every k-th point, until the initial point is reached. A very legitimate question to ask at this time would be: *Do any two numbers n and k, where n > k, determine an n-sided polygon <n,k>?* The answer to this question is "no." Take <12,3>, for example. Marking 12 equally spaced points on the circle and connecting to the third point each time, we get the sequence (1,4,7,10,1)—a closed path is obtained before all the 12 points have been reached. As shown below, the shape we get is a square, not a 12-sided polygon.



Hence, we need to ask a further question: *When do two numbers n and k, where n > k, determine an n-sided star polygon <n,k>?*

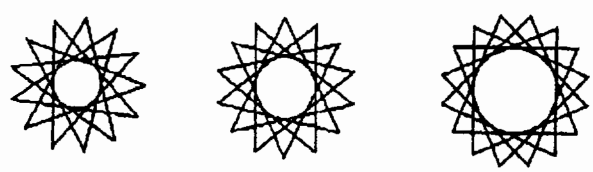
I invite you to move away from the computer for a moment—a "hands off" activity to complement the computer work—and do the following: Make n points on a circle, with n = 4,5,6,7,8,9,10,12, etc., and construct all possible n-sided polygons.

Did you conclude, as do other mathematicians, that it is only when n and k (n > k) are relatively prime (i.e., their greatest common divisor is 1, or, to say it yet another way, no integer greater than 1 divides both n and k), that there exists an n-sided polygon? When k = 1 or k = n - 1, the n-sided polygon is simple; otherwise it is a star polygon. (This also explains why there are no 4- or 6-sided star polygons, but I will leave you to ponder about that conclusion.)

We know now that we get an n-sided polygon only if n and k are relatively prime. *For a given n, how many n-sided polygons are there?* That is relatively simple to answer: just count the number of numbers smaller than and relatively prime to n. (In number theory this number is called Euler's j (phi) function of n.) This leads to the following question: *How many different <n,k> polygons are there?* The answer is arrived at by dividing the above result by 2 because of the congruency property (the mirror images of 11-sided polygons seen above) discussed above.

And finally we get to the end point we have been seeking, namely, how is it possible to draw an <n,k> polygon with Turtle graphics?

Extending to the general case the approach used with the 11-sided polygons leads one to the following conclusion: if there exists an n-sided <n,k> polygon it can be drawn by POLY <side> (360*k)/n. The following diagram shows examples of 13-, 14-, and 16-sided star polygons.



Remark: When trying to implement the approach described above, a small problem is encountered. In many cases it seems that the computer ignores the stopping rule and the procedure runs into an infinite loop. This inconvenience is caused by inexact decimal approximations of simple fractions, such as 720/11. Using the computer's approximations, the heading never returns to exactly 0. (Actually, Logo responds FALSE if we ask it to PRINT 11*(720/11) = 11). Being aware of this technical problem, it seems appropriate to ask POLY to stop when the heading is "almost" zero. We may define an ALMOST.EQUAL predicate and use it as the stopping condition or just replace the original stop rule with:

```
IF OR HEADING<0.01 HEADING > 359.99 [STOP]
```

Bibliography

Abelson, H., & diSessa, A. (1981). *Turtle geometry: The computer as a medium for exploring mathematics*. MIT Press: Cambridge, MA.

Dawson, A. J. (Sandy). (1969). *The implications of the work of Popper, Polya, and Lakatos for a model of mathematics instruction*. Unpublished doctoral dissertation, University of Alberta.

Dawson, A. J. (Sandy). A fallibilistic model for instruction. *Journal of Structural Learning, 3*(1), 1-19.

Lakatos, I. (1976). *Proofs and refutations: The logic of mathematical discovery*. Boston: Cambridge Press.

Newman, C., & Turkle, S. (1989). Integrating arithmetic and geometry with numbered points on a circle. *Arithmetic Teacher, 36,* 5.

O'Daffer, P., & Clements, S. (1977). *Geometry: An investigative approach*. Reading: Addison Wesley.

Rina Zazkis is an assistant professor in the Department of Mathematical Sciences at Northern Illinois University, DeKalb, IL 60115. She can be reached through BITNET as TF0RXZ1@NIU.BITNET.

A. J. (Sandy) Dawson is Director of the Professional Development Program and an Associate Professor in the Faculty of Education at Simon Fraser University in Vancouver, BC, Canada V5A 1S6.
He can be reached through BITNET: USERDAWS@SFU.BITNET

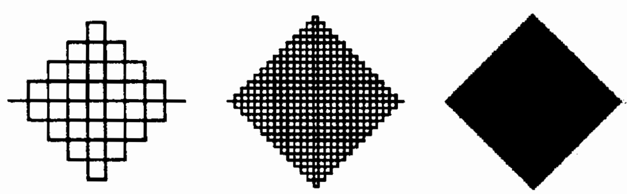
Extra for Experts

edited by Mark Horney

Mandelbrot's Sweep of the Koch Snowflake

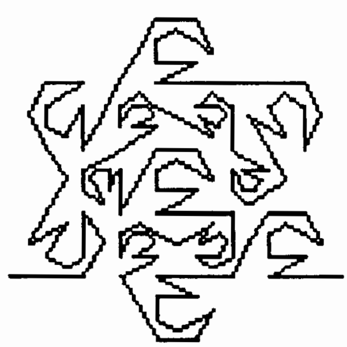
by Jane F. Kern

Peano curves, or plane-filling curves, represent the class of fractals that fill, or sweep, a piece of the plane. In 1890, Guiseppe Peano discovered the first plane-filling curve, and as a result this set of fractals is called Peano curves. The original Peano curve fills the interior of a square by using a recursive process that divides the interior of the square into smaller and smaller squares until the region is filled:



Peano's original plane-fill design

Plane filling curves can fill a piece of the plane bounded by a polygon, a Koch curve, or another fractal. Around 1960, Benoit Mandelbrot, the father of fractal geometry, designed a Peano curve that sweeps the interior of the Koch snowflake. Mandelbot's snowflake sweep is shown below.



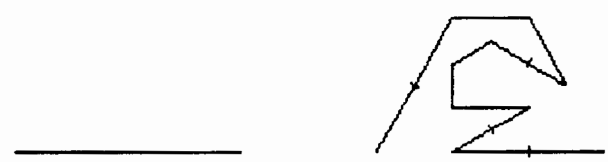
Mandelbrot's Snowflake Sweep

Mandelbrot's snowflake sweep presents an excellent opportunity, within the context of this article, to investigate the construction and generation of this plane-filling fractal and to apply this information to generating similar plane-filling curves. This investigation also provides the opportunity to review the properties of fractals in general. The two most important properties of these geometric shapes known as fractals are their self-similarity and their fractal dimension.

Generating Mandelbrot's Sweep

Generating Mandelbrot's sweep of the Koch snowflake presents a challenge. The challenge lies in constructing the generator itself and then in scaling it and moving it to its proper position to draw the next level of the curve. At the Level 2 drawing of this curve, each line segment of the generator must be replaced by a scaled-down version of the generator, and the scaled-down versions must be positioned so the curves do not self-intersect. Since the fractal, or self-similarity, dimension for Peano curves is 2, the scaled-down versions of the generator will eventually fill the plane at a higher level of recursion.

As with other fractals generated by using a recursive procedure, the construction of the Mandelbrot sweep begins with two geometric shapes, the initiator and the generator.



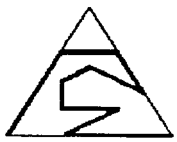
Level 0 - Initiator Level 1 - Generator

The initiator and the generator of Mandelbrot's sweep

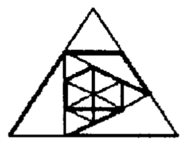
The initiator is a line segment of unit length. Notice that the generator is composed of 13 line segments of two different lengths. This means that this design has two scaling factors, r_1 and r_2. To construct the generator, it is necessary to determine the values of the scaling factors and the measures of the turning angles. In order to do this, a study must be made of the generator. First, scale an equilateral triangle, with side length equal to the unit length of the initiator, by a factor of 1/3. This divides the equilateral triangle into 9 congruent equilateral triangles (a below). Place the generator inside the scaled equilateral triangle (b below).

Equilateral triangle scaled by 1/3 Generator inside scaled triangle

Notice that seven segments of the generator are equal to one-third of the initiator, so the first scaling factor is $r_1 = 1/3$. Second, to find r_2, place the generator inside the equilateral triangle again (a below).This time, using the generator as a guide, draw in a middle equilateral triangle and divide this triangle into nine congruent triangles (b below).



a



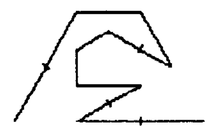
b

Since the value of r_1 is known, to calculate the value of r_2 choose one of the 30, 60 right triangles that are formed. (Exercise for class: How many different methods exist for finding the value of r_2? This value can be determined by using the Pythagorean Theorem, the relationship that exists between the sides of a 30, 60 right triangle, or the trignometric functions. How many 30, 60 right triangles are formed by the subdividing? Decide on which one you want to use and solve for the value of r_2.)

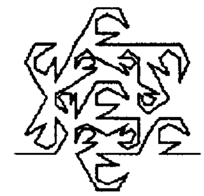
This scaling factor, r_2, is found to be $1/(3*\sqrt{3})$, and there are six segments of this length. Thus, the two scaling factors for the Mandelbrot Sweep are $r_1 = 1/3$ and $r_2 = 1/(3 *\sqrt{3})$. The number of components, N, is 13, since this is the number of segments used to construct the generator. It is necessary to know these values when checking the fractal dimension.

To determine the measures of the turning angles, walk the turtle around the generator for the Mandelbrot snowflake sweep. The generator should be turned to the vertical position so that the turtle will be in the home position to begin the walk. Using properties of the equilateral triangle, isosceles triangles, vertical angles, and parallel lines, the measures of the turning angles can be found. When the turtle ends its walk around this curve, the turtle should return to its original position and heading. It is important that the turtle is left with the same heading and position it had originally since the generator must be continually scaled and moved to its proper location at each level of recursion. The turning angles serve as the guide when writing the recursive procedure that generates this curve.

Once the lengths of the segments and the measures of the turning angles for Mandelbrot's sweep are known, attention can be given to writing the procedure MANDELBROT.SWEEP. Here, the Level 1 and Level 2 drawings of the curve, as shown in figures a and b below, will serve as guides.



a. Level 1



b. Level 2

Mandelbrot's Sweep

Observe, in b, that in order to obtain the Level 2 construction of this curve each line segment in Level 1 has been replaced by a scaled-down version of the generator, and some of these scaled-down versions are like the original while others are the flip version, or the reflection, of the generator. To generate the flip version, a variable for parity can be introduced in the procedure. When the procedure calls itself, the value of :PARITY must be positive 1 or negative 1. A positive :PARITY will produce the original version, and a negative :PARITY produces the flip version. Notice that each angle command must be multiplied by this variable. A negative :PARITY reverses all the angle commands, producing the flip version of the generator. The negative 1 was written to emphasize the flip version. A (- :PARITY) could also be used. The procedure reads as follows:

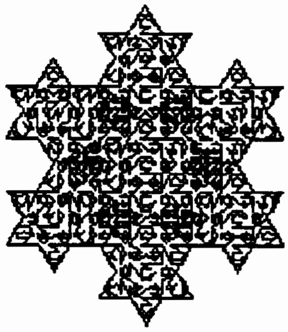
```
TO MANDELBROT.SWEEP :SIDE :LEVEL :PAR-
   ITY
IF :LEVEL = 0 FORWARD :SIDE STOP
LEFT 60 *:PARITY
MANDELBROT.SWEEP :SIDE/3 :LEVEL - 1
   :PARITY*(-1)
MANDELBROT.SWEEP :SIDE/3 :LEVEL - 1
   :PARITY
RIGHT 60*:PARITY
MANDELBROT.SWEEP :SIDE/3 :LEVEL - 1
   :PARITY
RIGHT 60*:PARITY
MANDELBROT.SWEEP :SIDE/3 :LEVEL - 1
   :PARITY
RIGHT 150*:PARITY
MANDELBROT.SWEEP :SIDE/
   (3*SQUAREROOT 3) :LEVEL - 1 :PARITY
MANDELBROT.SWEEP :SIDE/
   (3*SQUAREROOT 3) :LEVEL - 1 :PAR-
   ITY*(-1)
LEFT 60 *:PARITY
MANDELBROT.SWEEP :SIDE/
   (3*SQUAREROOT 3) :LEVEL - 1 :PAR-
   ITY*(-1)
```

```
LEFT  60  *:PARITY
MANDELBROT.SWEEP  :SIDE/
    (3*SQUAREROOT  3)  :LEVEL  - 1  :PAR-
    ITY*(-1)
LEFT  90*:PARITY
MANDELBROT.SWEEP  :SIDE/3  :LEVEL  - 1
    :PARITY
RIGHT  150*:PARITY
MANDELBROT.SWEEP  :SIDE/
    (3*SQUAREROOT  3)  :LEVEL  - 1  :PAR-
    ITY
MANDELBROT.SWEEP  :SIDE/
    (3*SQUAREROOT  3)  :LEVEL  - 1  :PAR-
    ITY*(-1)
LEFT  150*:PARITY
MANDELBROT.SWEEP  :SIDE/3  :LEVEL  - 1
    :PARITY*(-1)
MANDELBROT.SWEEP  :SIDE/3  :LEVEL  - 1
    :PARITY
END
```

(Exercise: This is not the only way the procedure to generate this curve can be structured. Since two different versions of the generator are used, the original version and flip version, another method is to write two separate subprocedures, one for each version of the generator. Using this method avoids the use of the extra variable for parity. Try writing the procedure to generate this curve using different generators.)

Sweeping the Koch Snowflake with Mandelbrot's Curve

After writing the procedure for Mandelbrots' Sweep, it is exciting to watch this fractal being drawn on the screen, but it is even more exciting to watch this curve sweep the snowflake. The Koch snowflake is a closed curve—it starts and stops at the same point—whereas Mandelbrot's fill of the snowflake is not a closed curve. When a curve fills a planar region it has an entry point to the region it is filling and an exit point from the region. Mandelbrot's sweep enters at the lower left vertex of the Koch snowflake, fills the snowflake, and exits at the lower right vertex.



Mandelbrot's Curve Sweep the Koch Snowflake

When Mandelbrot's curve sweeps the snowflake, the level used as input to generate the Mandelbrot sweep should be one higher than the level used to generate the Koch snowflake. For example, the interior of the triangle produced by the initiator of the snowflake, Level 0, would be filled by the Level 1 version, or the generator of MANDELBROT.SWEEP. Likewise, the curve generated by the Level 2 drawing of the snowflake would be filled by the Level 3 version of MANDELBROT.SWEEP. (These are the inputs for the drawing above.) Remember to position the turtle correctly before it starts its fill of the snowflake.

Self-Similarity of Mandelbrot's Fill

Fractals generate self-similar curves at each level of their construction. Self-similar means that part of the curve is similar to the whole curve. Mandelbrot (1983) says, "A fractal is a shape made of parts similar to the whole in some way." He further states, "When each piece of the shape is geometrically similar to the whole, both the shape and the cascade that generate it is called self-similar." Mandelbrot's fill or sweep of the snowflake can be used to illustrate this property. Look at the generator, Level 1, and the next level of construction for this curve in Figure 6. Now look at that part of curve in Level 2 that has replaced the first segment of the generator. Magnify, by a factor of three, this section and any section of this curve at Level 2 that has a scaling factor of one-third, and it gives the same curve as the generator. Magnify, by a factor of $3*\sqrt{3}$, the smaller sections of this curve at Level 2, and it also gives the same curve as the original. This property can be illustrated using any scaled-down version of the generator in the Level 2, or in a higher level, drawing of the curve. Each one of the scaled-down versions is similar to the whole curve.

Dimension of Mandelbrot's Fill

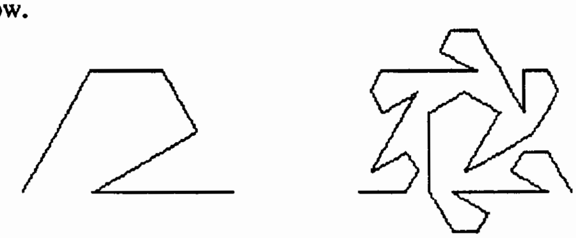
Since Mandelbrot originally defined a fractal in terms of its dimension, one cannot look at fractals without considering their dimension. The fractal dimension of a curve is determined by using the formula $D = \log N/ \log 1/r$. In the formula, N represents the number of components used to construct the generator and r represents the scaling factor. N and r are two necessary ingredients in the construction of fractal curves. For example, in generating the Koch snowflake, the scaling factor r is 1/3. The initiator, or unit line segment, has been scaled by 1/3 or divided into 3 congruent segments. Four of those segments are used to construct the generator for this curve. Hence, N = 4 and the fractal dimension for the Koch snowflake is $D = \log 4/\log 3 = 1.2618...$

It is interesting to note that the unit segment that was the initiator of the Koch snowflake has a dimension of 1. The dimension of the curve that replaced it has a fractal dimension of 1.2618..., and since four segments were used to construct this generator, using the $Nr^D = 1$ version of the dimension formula, it can be shown that $4 \times (1/3)^{1.218...} = 1$. Also, at the next level of the curve, $16 \times (1/9)^{1.218...} = 1$. This is why the dimension of these curves is referred to as the fractal dimension or the self-similarity dimension.

Since plane-filling fractals eventually fill a piece of the plane, the dimension of these curves is 2. In Mandelbrot's sweep, the unit line segment with dimension of 1 was replaced by a generator that has 13 components, 7 with a scaling factor of 1/3 and 6 with a scaling factor of $1/(3*\sqrt{3})$. Therefore, using the above analogy, it can be shown that $7(1/3)^2 + 6\,(1/(3*\sqrt{3})^2 = 1$. The formula for fractal dimension is useful not only when trying to determine the dimension but also when trying to find a scaling factor when the dimension and the number of components are known.

Mandelbrot's Other Snowflake Sweep

The construction and generation of Mandelbrot's sweep serve as a basis for investigating other fractals that are similar. Mandelbrot designed another sweep of the snowflake curve. The generator and Level 2 drawings of this fractal are shown below.



Level 1 - Generator Level 2

Another Mandelbrot Sweep

As with the previous fractal, the generator of this fractal must be scaled and moved to its proper place to produce the Level 2 drawing of the curve. However, to generate the Level 2 drawing of this fractal, the generator is not only flipped in some cases but also translated and rotated. One way to accomplish a translation and rotation is to have the turtle pick up its pen, move forward the necessary distance, rotate 180 degrees, put the pen down, and then draw either the original version of the generator or the flip version of the generator.

Various sweeps of the Koch snowflake may be obtained by sequencing the generators of these Mandelbrot sweeps and their flip versions, rotated versions and translated versions in different orders.

(Exercises: Using the Level 2 drawing of this curve as a guide, determine which transformation—a flip, a glide, a rotation, or is some cases a combination of two or more motions—is needed to produce the advanced generation of this fractal. Using the hints given in the previous paragraph, write a procedure to generate this curve. Write another procedure to generate this curve using the fact that four different generators are used to generate this curve. Can you design a different fill for the Koch snowflake?)

Steps to follow in investigating Mandelbrot's fills of the snowflake.

1. Scale a unit equilateral triangle by 1/3 or divide the equilateral triangle into nine congruent triangles.
2. Place the generator inside the equilateral triangle. If necessary divide the subtriangles into smaller equilateral triangles.
3. Determine the scaling factor or factors.
4. Determine the turning angles.
5. Study the Level 2 drawing of the curve. How many different versions of the generator have been used in the Level 2 drawing?
6. Write a recursive procedure using the turning angles and the scaling factors as guides.

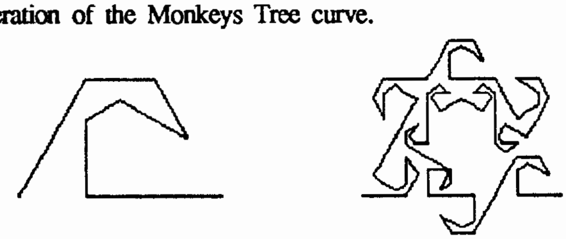
To design an original fill of the snowflake.

1. Begin with an equilateral triangle.
2. Scale the equilateral triangle by 1/3, 1/4,1/7, 1/n.
3. In designing the generator, determine the entry point to the interior of the triangle, the line segments the turtle will traverse, the scaling factors, and the turning angles.
4. Work with the Level 2 version of the curve when designing the generator.

Remember, the generator must be designed so that when it is scaled and moved at the next level the curves are self-avoiding. Replace each line segment of the generator with either the original curve, the flip version, the translated version, the rotated version, or a combination of any of these motions.

Generating Other Designs

Practice in generating these curves also helps in constructing various other Peano curves and provides clues to the generation of the Monkeys Tree curve.



Level 1- Generator^t Level 2

Monkeys Tree Curve

Logo: Search and Research

This curve, also known as Split Snowflake Halls, does not completely fill the snowflake; hence, its dimension is less than 2. This can be shown by finding the fractal dimension of this curve. The generator for the Monkeys Tree curve has 11 components, 6 with the scaling factors $r_1 = 1/3$ and 5 with the scaling factor $r_2 = 1/3 * \sqrt{3}$. Using this information and the formula for fractal dimension, the dimension of this curve can be determined by solving the following equation.

$$7(1/3)^D + 5\,(1/3*\sqrt{3})^D = 1.$$

(Exercise: Solve the above equation for D.)

Designing and generating fractals like the ones mentioned in this paper is not an easy task, but it does make one appreciate the work that Mandelbrot, Peano, Gosper, and others have done. Generating plane-filling fractals presents a challenge, one that is worth pursuing because of the insights gained in the new field of knowledge—fractal geometry. This pursuit is also valuable because it provides an opportunity to observe the integration of Euclidean geometry, turtle geometry, fractal geometry, and Logo, all working together, each one enhancing the understanding of the other.

References

Ableson, Harold, & de Sessa, Andrea A. (1981). *Turtle geometry: The computer as a medium for exploring mathematics*. Cambridge: MIT Press.

Barcellos, Anthony. (1984). "The fractal geometry of Mandelbrot," *The College Mathematics Journal*, *15*, 98-114.

Egsgard, John C. (1988). "An interesting introduction to sequences and series," *The Mathematics Teacher*, *19*, 108-111.

Kenney, Margaret J. (1987). "Logo adds a new dimension to geometry programs at the secondary level," in *Learning and Teaching Geometry, K-12*, 1987 Yearbook of the National Council of Teachers of Mathematics. Reston, VA: NCTM.

Kern, Jane F., & Mauk, Cherry C. (1990). "Exploring fractals—a problem-solving adventure using mathematics and Logo," *The Mathematics Teacher*, *83*, 179-185.

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

McWorter, Jr., William A. & Tazeleer, Jane Morrill. (1987) Creating fractals, *Byte*, *12*, 123-128.

Stevens, Roger T. (1989). *Fractal programming in C*, M & T Publishing, Inc., CA.

Jane F. Kern, Kutztown University
Pennsylvania State System of Higher Education
Kutztown, PA 19530

A New Vision of Logo in the Secondary School

by Douglas H. Clements

We have seen that—within limits—Logo can contribute to elementary students' construction of geometric knowledge. Unfortunately, researchers have not focused much on Logo in secondary schools. However, a team of educators from Bulgaria has done just that—and in quite an interesting way. Research with their materials has only begun, but in a recent meeting with the developers and others interested in the project I became excited enough by the work that I wanted to share it here.

Euclid: Should He Stay or Go?

Euclid's formulation of geometry continues to dominate secondary school instruction (Filimonov & Kreith, 1990). What is remarkable is that schools maintain this commitment despite overwhelming evidence regarding the approach's shortcomings. For over a century we have known that Euclid's formulation was pedagogically flawed. Worse, however, the deductive approach as presently employed is a failure for many. Only about 30% of high school geometry students enrolled in a course for which proof was a goal were able to write proofs or exhibit any understanding of the meaning of proof (Senk, 1985; Suydam, 1985). It is no wonder that doing proofs was the least liked mathematics topic by 17-year-olds on the 1982 National Assessment of Educational Progress and that less than 50% of the students rated the topic as important (Clements & Battista, in press).

A Logo-Based Approach to the Dilemma

Educators from the University of Sofia have developed a Logo-based software package to address these problems. They believe their approach maintains what is valuable about Euclid's geometry, with one major change. To better prepare for deductive proof, the process of *proof* is initially replaced by that of *algorithm*. Proof's "logical dependence" is replaced by a computer based "structural dependence." So, instead of citing a proven theorem in our proof of a later one, we call an established algorithm as a subroutine of a more elaborate one.

The teaching approach these educators propose fits nicely with recommendations from the National Council of Teachers of Mathematics (1989):

> Although the hypothetical deductive nature of geometry first developed by the Greeks should not be overlooked ... the organization of geometric facts

from a deductive perspective should receive less emphasis, whereas the interplay between inductive and deductive experiences should be strengthened. For example, students should first use an interactive computer software package that allows experimentation with figures and relations to observe across several trials that the length of the median to the hypotenuse of any right triangle appears to be equal to the lengths of the segments it cuts off on the hypotenuse. In the second phase, they would provide a deductive argument verifying their discovery. (p. 159)

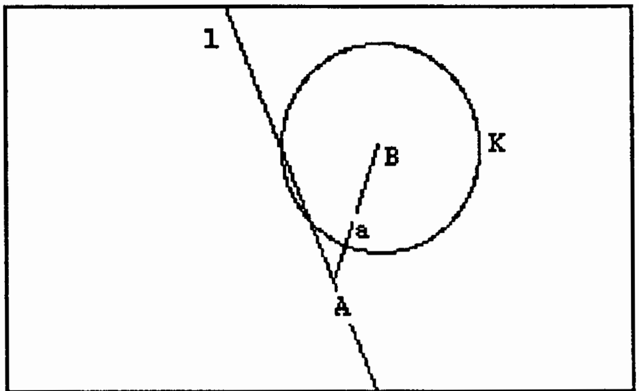
One could use manipulatives, compass, or especially software, such as the *Geometric Supposer* series, for such explorations. The Bulgarians' Logo-based software, however, embodies the ideas of algorithms and structural dependence in ways that may enrich such explorations in a unique fashion. Such an approach may help students appreciate the beauty and mathematical power in the geometry of the Greeks, including constructions.

The Plane Geometry System: An Example

The Plane Geometry System (PGS) is an extended version of Logo. Usual Logo text and graphics modes exist. A unique mode is a construction environment for representing geometric objects. New data types and corresponding operations on them are defined in typical Logo style. The basic primitives for constructing geometric objects bear familiar names, such as POINT, SEGMENT, LENGTH, LINE, RAY, CIRCLE, RADIUS, and VECTOR. OBJECT is a primitive that defines and modifies these geometric objects. It takes two inputs, name and a value. For example, a student programmer could type OBJECT "A POINT 10 -70 to place a point. (Students will also be allowed to locate points through synthetic techniques.) Then the following commands might be given:

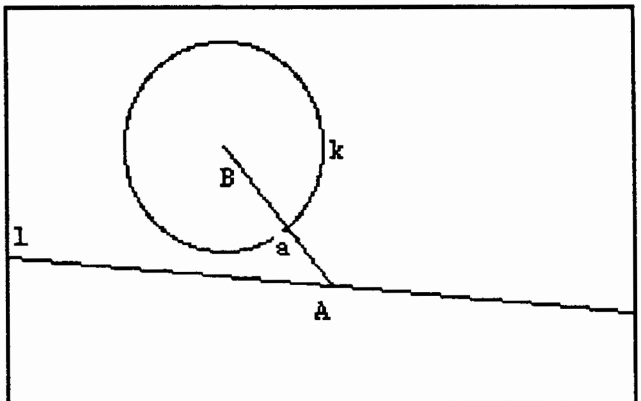
```
OBJECT "B POINT 60 50
OBJECT "a SEGMENT :A :B
OBJECT "l LINE :A 45 + HEADING :a
OBJECT "k CIRCLE :B 100
```

The screen would show




```
OBJECT "a SEGMENT :A :B
OBJECT "l LINE :A 45 + HEADING :a
OBJECT "k CIRCLE :B 100
```

The PGS, however, includes more than just new commands. Each object created, such as a circle, has a name, value, image, *and* relationships to other objects. So, in the previous example, suppose we type OBJECT "B POINT -95 60. The figure below is generated.

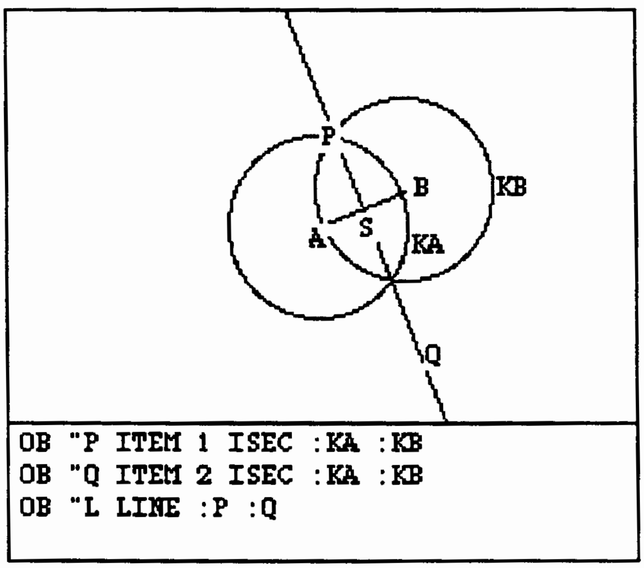



```
OBJECT "B POINT -95 60
```

Because circle k depends on point B for its definition, it changes automatically whenever point B is changed. The developers of PGS call this "automatically maintaining the definition (constructive) dependencies among objects." Such a feature allows students to create geometric structures and objects, experiment with them, transform them, measure them, and study their properties. For example, if students created another point C through construction, they could check colinearity by typing POINTON? :A LINE :B :C, which would return TRUE or FALSE.

The system keeps a log of all the commands given in the process of solving a problem or experimenting. Students can move any sequence of commands to an editor and change them into a procedure.

Assume that students have figured out how to construct a perpendicular bisector of a segment. The figure below shows their construction and the last three commands they issued.



The students might move their commands to the editor, add TO PERBIS and END, and thus define a procedure.

```
TO PERPBIS
OBJECT "A POINT 0 0
OBJECT "B POINT 80 30
OBJECT "S SEGMENT :A :B
OBJECT "KA CIRCLE :A :B
OBJECT "KB CIRCLE :B :A
OBJECT "P ITEM 1 ISEC :KA :KB
OBJECT "Q ITEM 2 ISEC :KA :KB
OBJECT "L LINE :P :Q
END
```

The real power, of course, emerges after students have written a procedure and then "generalize" it to create a tool. For example:

```
TO PERPBIS :a :b
OBJECT "S SEGMENT :A :B
OBJECT "KA CIRCLE :A :B
OBJECT "KB CIRCLE :B :A
OBJECT "P ITEM 1 ISEC :KA :KB
OBJECT "Q ITEM 2 ISEC :KA :KB
OBJECT "L LINE :P :Q
OUTPUT :L
END
```

Students can apply this to many segments to test its universality. Finally, to make the tool even more transparent—literally—students can instruct the computer not to show the intermediate steps of the construction. HIDE and NHIDE hide objects and their names, respectively.

```
TO PERPBIS :a :b
OBJECT "S SEGMENT :A :B
OBJECT "KA CIRCLE :A :B
OBJECT "KB CIRCLE :B :A
OBJECT "P ITEM 1 ISEC :KA :KB
OBJECT "Q ITEM 2 ISEC :KA :KB
OBJECT "L LINE :P :Q
HIDE ["KA "KB "P "Q]
NHIDE ["A "B "S]
OUTPUT :L
END
```

Now, the tool PERPBIS works just like a primitive in the solution of other problems. Here are two other student-constructed tools.

```
TO BISECTOR :A :B :C
; A function that outputs the bisec-
   tor of the angle ABC
OUTPUT LINE :B ((HEADING LINE :B :C)
   + HEADING LINE :B :A ) / 2
END
```

```
TO SYMLINE :A :B
; A function that outputs the symme-
   try line of the points A and B
LOCAL "a
MAKE "a SEGMENT :A :B
OUTPUT LINE POINTON :a .5 90 + HEAD-
   ING :a
END
```

The benefits of this approach are numerous. Consider the learning of geometric constructions. Students can examine a full record of their previous steps. They can reflect on, and edit, these steps and repeat the construction. In Papert's (1980) words, when "intuition is translated into a program it becomes more obtrusive and more accessible to reflection." Second, the student can then relegate the actual construction to the computer. At this point, the computer is to geometry what a calculator should be to arithmetic—a technological aid liberating the student from rote activity (Filimonov & Kreith, 1990). For example, students might use their PERPBIS procedure to solve the problem of constructing the intersection of the medians of a triangle.

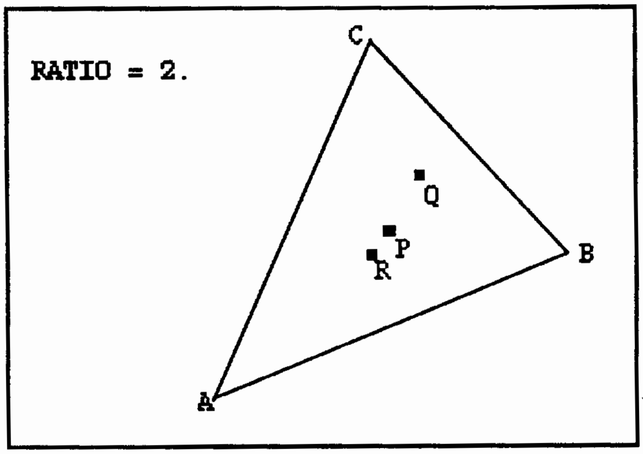
```
OBJECT "A POINT 120 60
OBJECT "B POINT 80 -50
OBJECT "C POINT -30 0
OBJECT "T POLYGON :A :B :C :A
OBJECT "MA SEGMENT :A PERPBIS :B :C
OBJECT "MB SEGMENT :B PERPBIS :C :A
OBJECT "MED ISEC :MA :MB
```

And so on. Through this sort of activity, students are likely to discover that the medians of the triangle intersect at a single point. But will it work for *any* triangle? The definition for points A, B, and C can be changed and the construction is automatically repeated. A "generalized" procedure can be constructed and tested. This type of activity has been shown to *motivate* proof (or a less formal way of understanding the construction). That is, students want to understand *why* this occurs (Clements & Battista, in press). Now it makes sense for students to use the algorithm in a generalized procedure (MEDIANS.TRI :A :B :C), possibly with the objects used in the construction hidden.

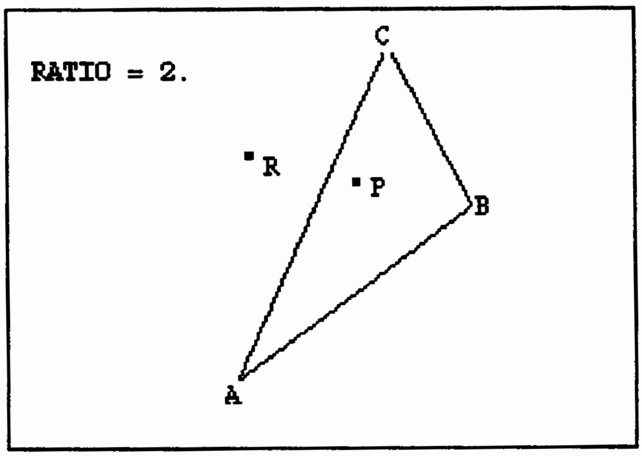
The computer now enables *exploratory* mathematics. Say that students have similarly defined procedures to construct the intersection of the altitudes of a triangle and the center of the circle passing through a triangle's vertices. They can use these procedures to discover what is interesting about these three points. Having the computer construct all three leads to the observation that they seem to lie on a line. Distances between them may lead students to add a statement that prints the ratio:

```
PRINT (LENGTH SEGMENT :ALT :MED) /
    (LENGTH SEGMENT :MED :CCIR).
```

The screen would appear as shown below.



To check if this is generalizable, students might change the definition of one or more of the points A, B, and C (see below; note that this pictures Euler's theorem).



The Plane Geometry System and Other Construction Software

In this way, software such as the PGS may offer the power of both geometric construction programs (e.g., Geometric Supposer) and Logo (including building functions and other tools). Conversely, some of the difficulties of each are ameliorated. For example, construction programs are not creative programming languages and are limited in the range of objects available at any one time. On the other hand, using Logo alone can make it difficult to model certain geometric objects easily within a traditional geometry course. In PGS, students dynamically model and manipulate geometric objects, but have all the functionality of Logo to capture, use, and explore algorithms.

Skeptics might point out that the Geometric Supposer has easy-to-use menus. It indeed might take more time to learn to use Logo or the PGS. But the result is potentially more powerful, extensible, and creative. The availability of Logo means that students can build primitives (e.g., for the area of any quadrilateral) that are often "black boxes" in construction programs. They can also ask questions (e.g., Will the same phenomenon be true of pentagons or octagons?) that they can't ask within more constrained (closed) construction programs. Students build a toolbox of algorithms and geometric knowledge simultaneously.

Finally, the creators of the PGS argue that the use of a language is central: "At the heart of this system is the philosophy that in order to do mathematics, students must have LANGUAGE to express their mathematical ideas and that the notion of DEFINITION is so central to mathematics that it cannot be ignored in mathematical education" (p. 1). They also state: "The most important and fundamental mathematical activity is dealing with notions—mainly composing and decomposing of notions—which definitely needs a language" (Filimonov & Sendov, 1990, p. 1). This is the reason they did not use menus. Geometric activity should, in their opinion, be explained explicitly in terms of a language. When geometric objects and processes are so described, they can be saved, studied, revised, generalized, utilized, and communicated.

Needless to say, PGS's combination of geometric objects and Logo also provides an opportunity to teach computer science concepts in the "concrete" terms of geometric ideas. This integrated teaching of mathematics and computer science is a main goal of the Bulgarians.

Of course, we have learned much about using tools such as the Geometric Supposer that is relevant to using the PGS and other similar tools. I shall review this research in a future column. Here we shall examine research conducted with the PGS.

Initial Research

The PGS developers believe that the most important feature of the Logo philosophy is the process of learning. Their goal is to extend Logo so that working in the traditional syllabus in Logo style would be more effective. Not much "hard" evidence has been collected to test that claim. Informal two-year experiments using the PGS in 30 Bulgarian schools with 14 and 15 year-old students, however, have reported success.

One study involved two classes of seventh graders at Vladimir Baschev School in Sofia, Bulgaria (Kolcheva & Sendova, 1990). This school is part of a system of experimental schools enrolling a cross-section of Bulgarian school children. Fifth graders begin a systematic study of Logo. The PGS was developed as an extension of this introduction. The data consists of discussions from those classrooms. Only initial reports are available at present, but they are suggestive.

Introducing the PGS, teachers engaged their students in a dialogue on the question, "What is the simplest geometric object?" Students made many suggestions, including the square, circle, and point. The rationale for the point was "Because we have to draw less. That is not the case with the triangle and the square—we have to draw a lot!" (p. 55). Some minutes later, everyone agreed that the point is the most "elementary." The teacher asked what they would need to type after the word POINT. "Two numbers ...we shall associate it with the coordinate system since each point has a certain location." The students learned they have to give a label, and the teachers provided the OBJECT command. Children placed many points on the screen and discuss the meanings of coordinates. They came to understand what characterizes a point and the relationship between the point's location and its coordinates. Few wanted to leave the classroom; one was overheard to say, "That's it! I've got it at last! I could hardly understand it before" (p. 57).

Later, students invent the command to create a line. Just through exploring the system, they also come to see that they need at least two points and that a line is determined by two points. Some invent the idea of determining a line with one point and a heading. Two boys are excited about creating a line with "invisible" (unlabeled) points (e.g., OBJECT "A LINE POINT 0 0 POINT 50 40). "You have just to imagine what points the line would pass through in order to construct it" concludes one. "What you mean is that it is not necessary to construct the points in advance" clarifies the second.

Another boy types:

```
OBJECT "A LINE :A :B
OBJECT "B LINE :A :B
OBJECT "C LINE :A :B
```

and concludes "If we are going to construct a line by points then there should be just two. And the line will be only one..." (a basic Euclidean postulate!).

The authors point out that while the students are quite active, the teacher "is not able to doze still. He or she has to think of situations which will

- stimulate the pupils' creative spirit;
- develop their mathematical intuition;
- make pupils ask questions whose answers they would otherwise have accepted on the basis of dogma. ... The teacher plays a role like that of a conductor of an orchestra comprised of members each of whom have the right to play a solo part and to give [their] own interpretation of the theme. The "conductor" will only have to follow their interpretation and ensure harmony. Only(!) (pp. 61-2)

Informal trials at the University of California at Davis with high school students and teachers have also been promising. In one workshop, over half of the participating teachers discovered Euler's theorem (Filimonov & Kreith, 1990).

As more research is produced, it will be reported here. In addition, the Plane Geometry System is but *one* of the extensions of Logo being produced by a project called the Logo-based Mathematical Laboratory. This project aims to develop a unified Logo-based environment covering the sphere of secondary mathematics (Filimonov & Sendov, 1990). That also should be interesting!

For more information on the Plane Geometry System, contact

Dr. Bojidar Sendov
Educational Computer Systems Laboratory
Sofia University
5 Anton Ivanov Blvd.
1126 Sofia
BULGARIA

(Note: A Bulgarian enterprise should be distributing a commercial version, possibly within the next year.)

In the U.S., write to:

Kurt Kreith
Department of Mathematics
University of California
Davis, CA 95616-8633

References

Clements, D. H., & Battista, M. T. (in press). Geometry and spatial reasoning. In D. A. Grouws (Ed.), *Handbook of research on mathematics teaching*. Reston, VA: National Council of Teachers of Mathematics/Macmillan.

Filimonov, R., & Kreith, K. (1990). *Euclid turns to programming*. University of California, Davis.

Filimonov, R., & Sendov, B. (1990). *A brief design rationale for the plane geometry system.* Sofia University, Bulgaria.

Kolcheva, M., & Sendova, E. (1990). *Reinventing the "Elements" in a Logo-based environment.* Sofia University, Bulgaria.

National Council of Teachers of Mathematics. (1989). *Curriculum and evaluation standards for school mathematics*. Reston, VA: Author.

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

Senk, S. L. (1985). How well do students write geometry proofs? *Mathematics Teacher*, *78*, 448-456.

Suydam, M. N. (1985). The shape of instruction in geometry: Some highlights from research. *Mathematics Teacher*, *78*, 481-86.

Douglas H. Clements is an associate professor in the Department of Learning and Instruction at the State University of New York at Buffalo . He has studied the use of Logo environments in developing children's creative, mathematics, metacognitive, problem-solving, and social abilities. Through a National Science Foundation grant, he has codeveloped a K-6 elementary geometry curriculum, *Logo Geometry* (published by Silver, Burdett, & Ginn). His most recent book, *Computers in Elementary Mathematics Education*, emphasizing Logo, was published by Prentice-Hall in 1989.

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		

Readers of this column probably know of the excellent Logo research and courses taking place in Bulgaria. Now Bulgaria's northern neighbor, Rumania, is beginning to make Logo ripples in their schools. A series of Logo modules called SSIRIUS-C has been tested in the Children's Palace in Bucharest. Plans are for the modules to be used in other forms of extrascholastic education (e.g., the Children's Houses). Preliminary results have lead Romanian educators to believe that Logo can spread throughout the schools and into such subject areas as physics, algebra, chemistry, and biology. Ambitious teacher training efforts are taking place. For more information please contact Mr. Ion Diamandi, Research Institute for Computers, Calea Floreasca Nr. 167, Sector 2, Bucharest, Romania.

Our major story this month from our Latin American correspondent, Jose Armando Valente.

Expanding Teacher's Views About Logo

Very frequently we hear that Logo is for children, that Logo is good for establishing the first contact with the computer, or that in order to do something serious and worthwhile with the computer students have to get beyond the squares and triangles and learn a *real* computer language like BASIC or PASCAL (at least they can get a job later).

These comments are made because people do not know Logo. However, if they try to learn about it, the information they find in the literature is, in general, about Logo graphics. If we give them a Logo demonstration, we ask them to do the squares and triangles. That is how Logo is being described and that is what the general public knows about it. To be able to get to the other side of Logo takes awhile. We have done very little to demystify this other side or to understand how one crosses to the other side.

In order to address these issues we recently ran a three-week workshop. The objectives of the workshop were

1. to change Logo teachers' views about Logo,
2. to provide them with the experience of seeing Logo ideas being used in different domains and seeing that Logo ideas can be applied even with another computer language; and
3. to develop the theme The Role of Description, Reflection and Debugging in Learning.

The workshop took place at NIED-UNICAMP, and its participants were 20 teachers, from different states in Brazil, who knew Logo and had some experience in using Logo with their students. The workshop was structured as several miniworkshops covering different topics, such as Recursive Logo Graphics, 3D-Logo, Logo-Music, LEGO-Logo, Lists, Logo Without Computers (ikebana and paper folding), and Introduction to Prolog. Our intention was to show the participants that even though the domains were different, problems could be solved in all of them by engaging in the process of describing the solution of the problem through a computer language, running this description, getting a result from the computer, reflecting upon it, and debugging the description if the result did not correspond to the initial solution. For example, the development of the recursive graphics activities was not done by talking about the recursive leap of faith but by dissecting the figure and describing it through a recursive process that would then be converted to Logo. In addition to the programming activity, the teachers had to reflect upon their own performance and their own learning in order to find out whether this way of developing the activities was something they could use in their own classroom.

The results of the workshop were very illuminating. First, it showed that no one is good at everything. For example, some teachers were wonderful at describing 3D figures but were quite incapable of doing things with their hands. Others did very well in ikebana but were a failure in LEGO-Logo. This tells us something about knowledge transference: it is one thing to say that a strategy can be used in several domains; it is another thing to show that this is true. Second, the same point made about strategy could be made about working style. If the person identifies with a particular domain, the working style used in this domain is much different from the style used in other domains that are less interesting to that person.

Both these observations indicate that Logo would better suit a much larger population if we all allow our students to use it in different domains. Initially the student must find the domain in which he feels most at home. Once this is done, then Logo's powerful ideas can be acquired and explored by working in this domain. This would make things less painful for everybody, and we would be able to see much more creative results. Please write to me at the following address for more details about the workshops.

Jose Armando Valente
NIED - UNICAMP
Predio V da Reitoria
13081 Campinas, SP
BRAZIL

Another large scale, longitudinal Brazilian study has been taking place at the Universidade Catolica de Petropolis, a private university with strong undergraduate programs in the sciences and humanities, and some graduate programs, particularly in the humanities. The University is located in Petropolis, a city founded during the imperial era and located 40 miles outside of Rio de Janeiro.

The Logo Project at the Universidade Catolica de Petropolis began in September 1984. It was created as part of the Education Department research plan to provide undergraduate and graduate students with the experience of using the computer as a tool to foster learning.

About 250 students per year participate in the project. Approximately 100 of them are undergraduate and graduate students, and about 150 of them are from elementary, secondary and high schools, where the computer facilities are used to develop extra-curricular activities. A large number of these students attend the city's public schools. Since 1988 two classes of public school students joined the project, and one of them is a special education class.

The elementary, secondary and high school students use the computer for about two hours per week during three months in the first semester and three months in the second. The main focus of the activities involves working with Logo and associating it with body syntonicity in order to help the students exercise their thinking capabilities according to their developmental level. The Logo activities are complemented with activities using concrete materials with which the students construct models of objects or with the development of activities such as music and plays. These activities are then implemented in the computer using Logo. For example, students develop scripts for plays by using Logo to create the dialogue between the characters and to illustrate their actions with drawings.

The result of each student's work is documented with the purpose of capturing the different levels of development of his or her activities and the cognitive, emotional, and social changes that can be observed in the student. This information is passed to the students' regular class teacher or to the head of the pedagogical sector of the school.

With respect to the undergraduate and graduate students of the Education Department, the objective of the work is the acquisition of the Logo methodology. Through Logo students can experience the use of the computer as a tool for thinking about thinking. Their Logo activities are used to discuss thinking and working styles, to question the teaching method of providing ready-made knowledge to the students, to understand the function of bugs in the process of construction of knowledge, and to experience the construction of knowledge as a process involving the appropriation of information. For the students who are learning to become teachers, this experience helps them to change their own ways of learning and to shift their view of the education process from teaching to learning. Thus, they can understand that their function as a teacher is not to pass information but to help the students to learn.

For more information please contact:

Silvia Branco V. Bustamante
Centro de Informatica Educativa
Universidade Catolica de Petropolis
Rua Benjamim Constant 213
25610 Petropolis, RJ
BRAZIL

The Logo Project: 1990 ICPSC Results

by Donald T. Piele

Once again students from Ireland have dominated the top rankings in the 1990 International Computer Problem Solving Contest (ICPSC). Irish students were ranked in eight of the top 12 spots in the Elementary Logo Division and in eight of the top 13 positions in the Junior Logo Division. And to finish it off, they came away with first place in both the Senior Logo Division and the Elementary Logo Division. The success of students from Ireland is probably related to the Logo Project.

The Logo Project began in 1985 at St. Patrick's College, in Dublin, Ireland. From an initial group of 18 children at one center, it grew to approximately 300 students in 15 centers. Each student spends approximately 75 hours in courses spread over a three-year period.

Project Aims:

The aims of the Logo Project are:

1. to enhance and extend the mathematical knowledge of young students and raise the level of their problem-solving skills,
2. to develop and test course materials and teaching strategies for using Logo as a tool for learning, and
3. to develop self-motivation and self-reliance in young students.

Students are selected for this extra-curricular activity on the basis of their performance on the Raven's Progressive Matrices and the Drumcondra Mathematics Tests. The courses are provided in 25-hour blocks on Saturdays, evenings, and summer vacations at ten schools and colleges around the country. The tutors are primary and secondary teachers—about 20 in all—who are interested in using Logo as a tool for learning. Local, national, and international ICPSC contests and exhibitions of project work form part of the experiences.

Irish Logophiles

Dr. Sean Close, one of the founders of the Logo Project and director of the ICPSC at St. Patrick's College, writes about two of his best logophiles:

> Andrew Farrell, winner of the Senior Logo Division, is 15 years of age. He is our best logophile and has been with us in the Logo Project for nearly five years. Andrew was well into the World Book and Childcraft Encyclopedia by age three. By age six he had been removed from three schools for apparent disruptiveness and continued his education at home for the next seven years. He scored 760/800 on the SAT at age 11 and shortly thereafter achieved joint first on the Irish National Math Olympiad. He entered Trinity College Dublin in 1988 to study for a degree in pure mathematics and is currently in his third (junior) year there. Each day he travels by bus for an hour and a half to get from the family home in County Meath to Trinity College in Dublin.
>
> Andrew is becoming increasingly interested in computers and is planning to take a degree in computer science when he is finished his mathematics degree. He enjoys reading (J.R. Tolkien's works at present) and doing mathematics puzzles. He also plays tennis and football. Last summer he acted as a Logo tutor in our summer school for mathematically able boys and girls and received an ovation from them at the presentations at the end of the course.
>
> Eoin Curran, winner of the Elementary Logo Division, is ten years of age. He joined the Logo Project when he was almost seven years of age. Eoin made the ICPSC rankings in 1988 and 1989. He is from the south side of Dublin city, and both his parents are teachers. His father runs a Logo Project Centre in a local school on Saturday mornings. Eoin's hobbies are swimming, science, hiking, and cooking. He has a BBC Master computer at home, which he uses almost exclusively for Logo. He attends his local primary school, but his parents are planning to start him a year early in secondary school.

ICPSC Objective

The primary goal of the ICPSC is to keep the tradition of creative computing alive by supporting teachers who want to challenge their students who enjoy this fundamental computer activity. We provide all the necessary materials for running the contest, including the problems and sample solutions in Logo, BASIC, Pascal, and C.

The contest has a very convenient format. Students enter as teams having one to three members. The teams are given two hours to write as many solutions as possible to a set of five problems, which involve turtle graphics, simulation, and words. Local winners are decided by local judges using the sample solutions we provide as a guide. Those who solve all five problems are eligible to have their solutions regraded and ranked internationally. Awards are presented to any team that solves all five problems and a plaque goes to each member of the best team and to the team's school.

1990 Rankings

1990 Elementary Logo Division

Rank	Team	School	City, State/Country	Director	Advisor
1	Eoin Curran	Bishop Galvin	Dublin, Ireland	Dr. Sean Close	Dr. Sean Close
2	Timothy Deegan	North Dublin School Project	Dublin, Ireland	Dr. Sean Close	Dr. Sean Close
3	Travis Kopp	Governor's Ranch Elementary	Littleton, Colorado	Forrest Smith	Jim George
4	Fionn O'Leary	St. Joseph's	Bishopstown, Ireland	Michael D. Moynihan	
4	Niall Douglas	Clogheen/Kerry Pike NS	Bishopstown, Ireland	Michael D. Moynihan	
4	Chelsea Jones Patricia Mazuch Nicole Pannekoek	St. Michaels University School	Victoria, B.C.	Dr. Alexander McMaster	Dr. Alexander McMaster
5	Ben White Steven Lobb	St. Michaels University School	Victoria, B.C.	Dr. Alexander McMaster	Dr. Alexander McMaster
6	Sean Boland	Athlonde	Dublin, Ireland	Dr. Sean Close	Mr. Tom Nohilly
7	Stephen Fuller	Scoilan Spioraid Naoimh	Bishopstown, Ireland	Michael D. Moynihan	Michael D. Moynihan
8	Gavin Hurley	Scoilan Spioraid Naoimh	Bishopstown, Ireland	Michael D. Moynihan	Michael D. Moynihan
9	Tony Kelleher		Dublin, Ireland	Dr. Sean Close	
10	Jeff Michaelson Troy Ablan Jeremy Damsgard	Galtier Magnet School	St. Paul, MN	Mike Amidon	Paul Krocheski
11	Sean Deschene Adam Johnson Mark Wegner	Oxbow Creek El.	Champlin, MN	Mike Amidon	Diane Hewitt
12	Michael Tamayo Michael Cole Lewis Blessing	Kendale Lakes Elemenary	Miami, Florida	Mrs. Sharon Freedman	Mrs. Sharon Freedman

1990 Junior Logo Division

Rank	Team	School	City, State/Country	Director	Advisor
1	Drew Narayan	Oswego	Oswego, NY	Rob Fredericks	
2	Anne Chazarreta	Bishopstown Community	Bishopstown, Ireland	Michael D. Moynihan	
3	Geraldine Hurley	Bishopstown Community	Bishopstown, Ireland	Michael D. Moynihan	
4	Matt Schneiderman Bo Thompson Brett McLean	Ligon Middle	Raleigh, NC	Cathay Smith	Donna Frink
5	Matthew Scully	St. Aidan's CBS	Dublin, Ireland	Dr. Sean Close	Dr. Sean Close
6	Damien Coady	St. Gerald's	Castlebar, Ireland	Dr. Sean Close	Mrs. Laura Kelly
7	Seamus Hoban	St. Gerald's	Castlebar, Ireland	Dr. Sean Close	Mrs. Laura Kelly
8	David Frink Charles Powell Kyle Rehder	Ligon Middle	Raleigh, NC	Cathay Smith	Donna Frink
9	John Paul Corkery	Presentation College	Bishopstown, Ireland	Michael D. Moynihan	
10	Gavin Hurley	Scoil An Spioraid Naoimh	Bishopstown, Ireland	Michael D. Moynihan	
11	Jon Rogness Joe Ditto	Ramsey Jr. Hi.	St. Paul, MN	Mike Amidon	Jim Steiner
12	John Sullivan	Colaiste An Spioraid Naoimh	Bishopstown, Ireland	Michael D. Moynihan	
13	Andrew Hryckowian Matthew Urhammer	Royal West Academy	Montreal West, Canada	Tom Murray	F. Parnell

1990 Senior Logo Division

Rank	Team	School	City, State/Country	Director	Advisor
1	Andrew Farrell	Trinity College	Dublin, Ireland	Dr. Sean Close	Dr. Sean Close
2	David Frink Charles Powell Kyle Rehder	Ligon Middle	Raleigh, NC	Catherine Smith	Donna Frink
3	Matt Schneiderman Bo Thompson Brett McLean	Ligon Middle	Raleigh, NC	Catherine Smith	Donna Frink
4	Mike Fiala Dwight Koga Mark Riehl	Live Oak H.S.	Morgan Hill, CA	David Foster	David Foster
5	Travis Cobbs Trent Olivera Scott Oalasek	Live Oak H.S.	Morgan Hill, CA	David Foster	David Foster

The 1990 Logo contest problems were written by David Green, then a master's student in computer education at the University of Oregon. He has now returned to his home in Australia to teach.

The Contest Divisions

Senior Division	Grades 10-12
Junior Division	Grades 7-9
Elementary Division	Grades 4-6

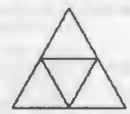
(L-R) Eoin Curran, Sean Close, Aileen Eglington, Andrew Farrell

Sample Problems

Elementary Problems

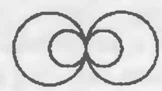
EQUILATERIAL TRIANGLES

Write a program which creates the following design. (NOTE: Each of the 4 small triangles are the same size and all sides are equal length.



KISSING CIRCLES

Write a program which produces the following design.



WORDS WORTH

Each letter in a word is worth a certain value depending on its position in the alphabet. The A has a value of 1, B has value 2, C has value 3, and so on, up until Z which has a value of 26. Write a program which takes a word as input and calculates its value by summing the values of each letter in that word. (You can assume that all letters are entered in upper-case). The output should look like the following.

```
?CALCULATE "HELLO
THE VALUE OF HELLO IS 52
```

You may use the following operation (reporter)

```
TO VALUE.CHARACTER :CHARACTER
OUTPUT (ASCII :CHARACTER)-64
END
```

Test your program with the words HELLO and GOODBYE.

Junior Logo Problems

SENTENCE WORTH

Each letter in a word is worth a certain value depending on its position in the alphabet. The A has a value of 1, B is of value 2, C is of value 3, and so on, up until Z which has a value of 26. Write a program which prompts the user for a sentence and calculates its value by summing the values of each letter in that sentence. (You can assume that all letters are entered in uppercase). The output should look like the following.

```
INPUT YOUR SENTENCE  THIS IS A SEN-
   TENCE
THE VALUE OF YOUR SENTENCE IS 170
```

You may use the following operation

```
INPUT YOUR SENTENCE  THIS IS A SEN-
    TENCE
THE VALUE OF YOUR SENTENCE IS 170
```

You may use the following operation

```
TO VALUE.CHARACTER :CHARACTER
OUTPUT (ASCII :CHARACTER)-64
END
```

Test your program with the sentences: THIS IS A SENTENCE and WHAT AM I WORTH

PICKING MARBLES

A bag contains 2 blue marbles and 5 red marbles. An experiment consists of

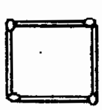
1. remove a marble from the bag
2. note its color
3. put the marble back
4. remove another marble
5. note its color
6. put the marble back
7. Compare the colors.

Write a program which simulates this experiment a given number of times and displays the number of times the same color was drawn. Test your program by repeating the experiment 5 times The output should look like the following.

```
HOW MANY TIMES? 5

BLUE
RED

RED
BLUE

BLUE
RED

RED
RED

RED
RED

THE SAME COLOR OCCURRED TWICE.
```

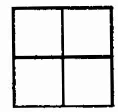
BUILDING SQUARES

Match sticks can be arranged to make squares e.g.

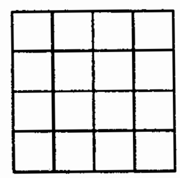


Write a program which draws a square grid (connected as shown) whose size is given by the user, and also *gives the number of matches that would be needed to make the grid.* NOTE: The number which is input refers to the size of the grid and not the number of squares.

```
BUILD 2
```



```
THERE ARE 12 MATCHES REQUIRED

BUILD 4
```



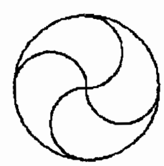
```
THERE ARE 40 MATCHES REQUIRED
```

Test your program for 4 and 5.

Senior Logo Problems

SEMI-CIRCLES

Write a program which produces the following design.



AVERAGE WORTH

Each letter in a word is worth a certain value depending on its position in the alphabet. The A has a value of 1, B is of value 2, C is of value 3, and so on, up until Z which has a value of 26. Write a program which prompts the user for a sentence and calculates its value by summing the values of each letter in that sentence. It should also calculate the AVERAGE VALUE PER word. (You can assume that all letters are entered in uppercase). The output should look like the following.

```
ENTER A LIST: THIS IS A SENTENCE
THE VALUE OF THE SENTENCE IS 170
THE AVERAGE VALUE PER WORD IS 42.5
```

Write a program which simulates this experiment a given number of times with the number of blue and red socks entered by the user. The program should also give the number of times a pair of the same color is drawn. The output should look like the following.

```
HOW MANY TIMES?  5
HOW MANY BLUE? 3
HOW MANY RED? 1

BLUE
BLUE

BLUE
BLUE

BLUE
BLUE

RED
BLUE

BLUE
BLUE

YOU GOT A PAIR 4 TIMES.
```

Test your program by running it 5 times with BLUE = 2, RED = 4.

1991 Contest

The 11th Annual ICPSC will be held on Saturday, April 27, 1991, with Friday April 26, and Monday, April 29 as the alternate dates. To receive a complete packet of information on the 1991 ICPSC, send your request to the address below. You will also receive a free copy of *Compute It!*

Donald T. Piele, ICPSC
P.O. Box 085664, Racine, WI 53408

National Educational Computing Conference
NECC '91
June 16-20, 1991 Phoenix, Arizona

Pre-Conference Research Workshop: Monday, June 17, 1991
Logo Research and Classroom Teachers

Although a decade of Logo use in the classrooms has not led to conclusive answers about Logo students' learning, it is now generally accepted that the knowledge and teaching strategies employed in a Logo classroom can make a significant difference in what and how well students learn. This session will focus on classroom research related to Logo teaching and learning. The purpose is to create a dialogue among university-based researchers, representatives from the National Science Foundation (and other funding agencies), and teacher researchers conducting research in their own classrooms. The workshop is intended to help bridge a well-documented gap between academic educational researchers and classroom practitioners and hopes to establish new partnerships and agendas for educational investigations.

In this workshop we invite classroom-based and academic researchers to share their questions, methods, and findings, in order to generate a more significant set of questions that have resonance for the classroom and that will allow the community of Logo-using educators to use Logo more effectively as a vehicle to support learning. We are particularly interested in descriptive, qualitative research focusing on how teachers teach and what students learn in Logo classrooms.

Proposal Solicitation Deadline: April 15, 1991

We request a one-page abstract that summarizes a presentation devoted to one or more of the following:

1. A synthesis of studies of Logo teaching and learning which may be useful for classroom teachers.
2. What types of questions about Logo teaching and learning are relevant for classroom teachers to consider? How can these questions be investigated in the classroom?
3. Logo was originally developed to foster student learning in areas such as problem-solving, mathematical reasoning and creativity, and elementary computer science. What do Logo students learn in these and related areas?
4. Case study reports of the methods and findings of particular research projects. Reports of research by teachers and qualitative research are especially welcome.
5. Issues related to research methodology and collaboration: What types of methods are best suited to answer particular questions? How important is "objectivity?" Can or should a classroom teacher ever be truly "objective?" Can an outside researcher ever understand the dynamic and educational milieu of a particular classroom? How can teacher researchers and university-based researchers collaborate to provide triangulation of perspective?

Workshop:

The authors of 12 abstracts will be selected to participate in the workshop. Each author will be asked to prepare a 5-10 page paper to be presented as a 10 minute talk during the workshop. The papers and the recommendations of the participants will be edited by the workshop chairs into a special monograph for dissemination to the field. Invitations to participate will be sent out by May 1, 1991.

Workshop Coordinators: Daniel Lynn Watt and Molly Lynn Watt, Logo Action Research Collaborative, Education Development Center, 55 Chapel Street, Newton, Massachusetts 02160; FAX (617) 244-3436. Send or Fax proposal abstracts to the workshop coordinators by **April 15, 1991.** Invitations to participate will be sent out by **May 1, 1991.** Papers are due to the workshop coordinators by **May 28, 1991.**

Educational Use of Computer Based Media in the Information Society

September 13-19, 1991
Tokyo, Japan

Co-sponsored by:

Japan Association for Educational Technology
&
International Society for Technology in Education

The use of multimedia in education and training will be the focus of this conference. The conference will include both an English and Japanese strand.

To receive more information, contact:

ISTE, Multimedia Conference (Japan)
1787 Agate St.
Eugene, OR 97403-1923
ph. 503/346-4414 FAX: 503/346-5890

Second International Seminar

Educational Computing in Latin America

Strategies for the Use of Computers in Education

April 23-26, 1991
Mexico City, Mexico

sponsored by
The Ministry of Education of Mexico (SEP) and
The International Society for Technology
in Education (ISTE)

For more nformation, write:
ISTE, 1787 Agate Street, Eugene, Oregon 97403-1923

Logo Exchange
ISTE
1787 Agate Street
Eugene, OR 97403-1923