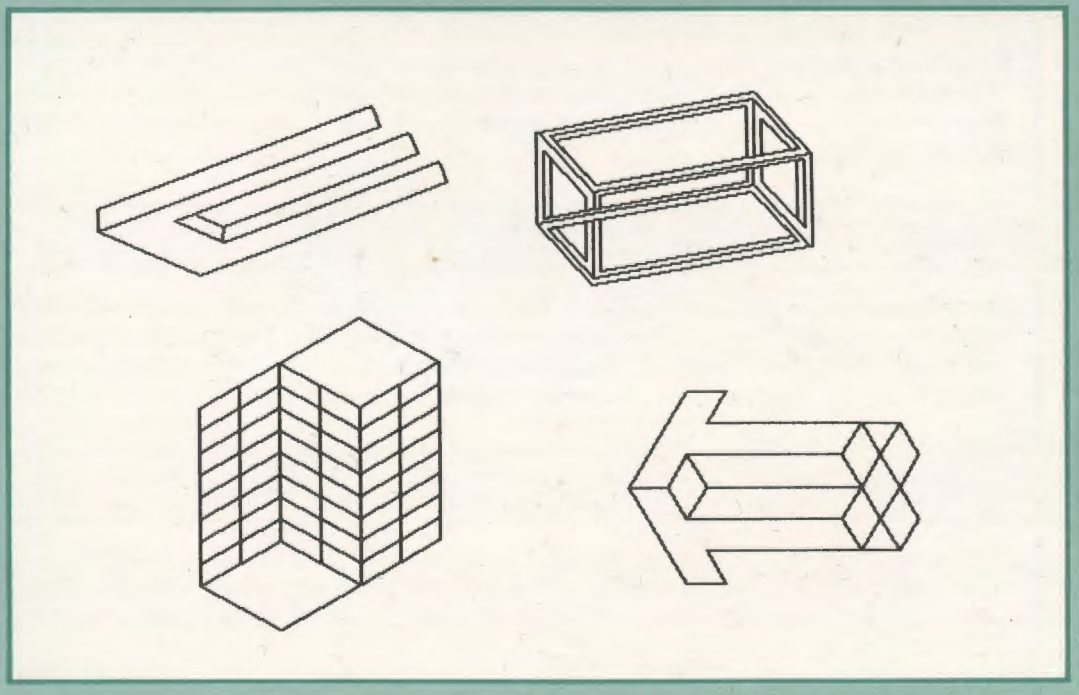
Journal of the ICCE Special Interest Group for Logo-Using Educators

# LOGO EXCHANGE

NOVEMBER 1988 VOLUME 7 NUMBER 3



International Council for Computers in Education

ICCE Publications

# LONG DISTANCE LOGO

Educators—You don't have to go to classes to earn graduate credit—let the classes come to you! Introduction to Logo Using LogoWriter, a graduate level independent study course, allows you to learn at your own pace while corresponding with your instructor by mail.

## WORK INDIVIDUALLY OR WITH A GROUP OF COLLEAGUES

Take Introduction to Logo Using LogoWriter at home, or study with a group of colleagues at school. The course uses a combination of video tapes (ON LOGO) featuring MIT's Seymour Papert, printed materials, textbooks, and diskettes. You view the tapes, read and report on course materials, do projects, design LogoWriter lessons for your students, and correspond with your instructor by mail.

## SAVE $$$

If your district supports group training, *you* save money! Districts enrolling six or more educators in this course will receive a reduction in the fees of each person enrolled. To qualify, your district must provide lab facilities and a resource person with experience in computing to help answer questions.

## NOT JUST ANOTHER CLASS

Dr. Sharon Burrowes Yoder, editor of the *Logo Exchange* journal, designed Introduction to Logo to provide staff development and leadership training. The four quarter-hour course meets the standards of the College of Education at the University of Oregon and carries graduate credit from the Oregon State System of Higher Education.

## ON LOGO VIDEOTAPES

School districts may acquire a license for use of the ON LOGO package of eight half-hour videotapes and 240 pages of supporting print for $599. For a one-time fee of $1,295, the package may be obtained with both tape and print duplicating rights, enabling the district to build libraries at multiple sites.

*Group Enrollment.* A tuition of $260 per participant is available to institutions that enroll a group of six or more educators. This special price does not include the ON LOGO videotapes. Your group must acquire the tapes or have access to them. Once acquired, the library of tapes and materials may be used with new groups enrolling for the same reduced fee.

*Individual Enrollment.* Educators with access to the tapes may enroll individually for $290. Tuition including tape rental is $320. A materials fee of $60 per enrollee is charged for texts and a packet of articles. Enrollees who already have the texts do not need to order them.

Additional information and order blanks can be obtained from: LONG DISTANCE LEARNING, ICCE, University of Oregon, 1787 Agate St., Eugene, OR 97403-9905. Ph: 503/686-4414.

# LOGO EXCHANGE

Volume 7 Number 3 | Journal of the ICCE Special Interest Group for Logo-Using Educators | November 1988





Logo Exchange is the journal of the International Council for Computers in Education Special Interest Group for Logo-using Educators (SIGLogo), published monthly September through May by ICCE, University of Oregon, 1787 Agate Street, Eugene, OR 97403-9905, USA.

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


Contents




ICCE Membership (includes *The Computing Teacher*)

| | U.S. | Non-U.S. |
|---|---|---|
| | 28.50 | 31.50 |

SIGLogo Membership (includes *The Logo Exchange*)

| | U.S. | Non-U.S. |
|---|---|---|
| ICCE Member Price | 24.95 | 29.95 |
| Non-ICCE Member Price | 29.95 | 34.95 |

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





The *Logo Exchange* is produced on a Macintosh SE and Laser Writer donated by Apple Computer, Inc.

# From the Editor

## Logo: It's Not Just for Little Kids

For some time now I have advocated the use of Logo at the high school level. Only a few years ago my ideas were frequently met with skepticism — I seemed to be the proverbial "voice crying in the wilderness." In the past year or so I have encountered more and more fellow educators (other than Brian Harvey) who are interested in exploring the use of Logo at the junior high and high school level.

My own experience with Logo began in the early 1980's. I don't remember when or where I first learned about Logo, but I do remember going to the local computer dealer with my copy of *Creative Computing* clutched in my hand, to ask them to order a copy of Terrapin Logo for me. (They had never heard of such a thing!)

I had been teaching a Pascal programming class at the level of today's Advanced Placement Exam at our high school for some years. With the sudden availability of microcomputers, I found that computer programming was in and that every parent decided that their child had to have a programming course or they would flunk life. My Pascal course clearly did not meet the needs of all of these students. With some reservations, I developed an Introduction to Computers and Computing course which had Logo as a major component. To my delight (and relief) it was a highly successful course, providing all students with positive experiences with Logo, and teaching them enough about programming so that they could make a reasonable decision about whether to pursue programming and computer science.

The last year (1986-87) that I taught the "Intro" course, I became aware of two factors that I believe will affect the use of Logo in the secondary schools in the future. For the first time, I found myself with students who had been exposed to Logo for some years at the elementary level. These students were weeks ahead of their classmates requiring that initial assignments be made flexible enough to challenge the young experts while easy enough to get the novices started. Fortunately, developing such open ended assignments is both natural and relatively easy in Logo.

The second new factor was the availability of LogoWriter as a replacement for Apple Logo II. Students had always enjoyed their Logo experience, but with LogoWriter, they appropriated Logo as a tool for use in other subjects much more than ever before. Students used Logo to do math assignments, write English papers, and even create mini-spreadsheets to do physics labs.

I think it is fair to assume that this trend towards use of Logo at the secondary level will continue. Whether incoming secondary students learned Logo as a programming language or used it as more of a productivity tool, they will carry with them the expectation that Logo will be a part of their computer repertoire. New versions of Logo such as LogoWriter and Terrapin's new Logo Plus as well as versions of Logo for the Macintosh will provide capabilities that make Logo easier to use in a variety of subject areas. Computer using teachers need to be prepared for the students who say: "You mean that with BASIC you can't fill the screen with color by typing FILL?" "Why can't I create my own shape to use in this graphics design?" "What do you mean I can't combine text and graphics on the same screen with this piece of software?" "Why don't you have Logo in the lab?"

These observations have broad ranging implications for the Logo community. First of all, developers of Logo software must continue to provide versions of Logo that are powerful enough to encourage older students to use them on today's increasingly sophisticated machines. Second, there is going to be a need for Logo materials appropriate for secondary students. This implies that materials that aren't cutsey and that get beyond how to draw a square need to be produced. Some of this needs to be done by software developers; other by existing publishers.

What part can you play in this growing trend? How about encouraging colleagues who are at the secondary level to explore the use of Logo? If you are a secondary teacher, share what you are doing with *LX* readers. Do you have a manuscript that you want to publish? Send it to me and we at ICCE will take a look at it. Let's all work together to get Logo spread to a wider audience.

Sharon Burrowes Yoder
ICCE, 1787 Agate Street, Eugene, Oregon 97403
CIS: 73007,1645 BitNet: ICCE@Oregon

### Lost and (hopefully) Found

During the past year, we have lost a lot of our former subscribers. Are you one of those missing persons? Do you know someone who used to subscribe to *LX* but no longer does?

As you can see, *LX* is alive and well and back on schedule. We have a lot to offer to you and your colleagues during this academic year.

Won't you please help us to find our lost friends as well as welcome new ones? *LX* will continue to thrive only with the support of SIGLogo memberships/*LX* subscriptions.

# Monthly Musings

## Tortoisensitivity

**by Tom Lough**

Have you ever had your thoughts lead you on a wild and free romp back and forth through time and remembrances? That happened to me just a few moments ago. I have rushed to the keyboard in an attempt to capture part of the experience before it escapes me. Here goes!

The trigger to this onset was an innocent looking catalog called *Whale Gifts*. I had noticed the catalog among the pieces of today's mail, picked it up, and flipped idly through a few pages. BINGO!! There on page 9 was a set of ceramic sculptures of baby turtles hatching from their eggs! Beautiful! And then on pages 22 and 23 were turtle mugs, turtle pendants, turtle sport shirts, and even a turtle necktie! Incredible! I had never seen such an assortment of turtle items in a single publication.

My mind raced back to a time two or three years ago when I had the idea to collect as many notices about turtle paraphernalia as I could find and then publish a little mail order turtle product catalog for Logo users. I placed a notice in the *NLX* newsletter asking readers to send me product information. No one did; the idea fizzled and died

Next, I remembered reading an article about the endangered Kemps ridley sea turtle. I contacted Carole Allen of Help Endangered Animals Ridley Turtles (HEART), and began a correspondence which continues to this day. She sent me information about her work to establish an additional nesting beach for the ridley, and a videotape of the program activities. I wrote an editorial in the *NLX* about her program and made an impromptu announcement about it in the middle of the Logo 86 conference at MIT. Later, I was pleased to read that Peter Rawitsch of Guilderland Center, NY, and his first graders had begun a project to raise money to help Carole and HEART.

Next, my mind took me to an incident I had nearly forgotten. I was driving to school one morning and saw a box turtle in the middle of the interstate highway. I screeched to a halt and backed up. (Fortunately there was no traffic at the moment. No state police either!) I stopped my car, opened the door, raced over to the creature, scooped it up, returned to the car, and deposited it on the floor by the front seat. Instead of going straight to school, I detoured to a wooded area nearby and released the turtle next to a peaceful pond. I hummed happily to myself that entire day.

FLASH! Next stop, Hawaii!! In the summer of 1985, I had the good fortune to teach for a short time with Glen Bull and Steve Tipps, the two pillars of the *NLX*. One night, I took them to a celebratory supper at a rather fancy restaurant. While we were waiting for our table, we strolled out to a side terrace. There, in the twilight, with the cool Hawaiian evening wind beginning to rise, we heard a gentle splashing coming from a nearby pool. When we investigated, we discovered a pair of huge sea turtles in a beautiful open air aquarium. The moment was magical! Needless to say, the maitre d' had to call us several times before we could tear ourselves away from our serendipitous colleagues.

Then I remembered reading a tiny item in a newspaper not long ago about a turtle species in Greece which was being threatened by beach front development. The item mentioned a group which was trying to save the turtle, but gave no specifics. I sent the item to a Grecian friend whom I had met at one of the MIT conferences. She put me in touch with a group coordinator who just happened to be scheduled to speak in London. I dashed off notes to Logo colleagues in the United Kingdom, who replied that they would plan to attend the turtle lectures and probably would not have heard about them except through my note.

ZIP! Nearer the present, about a month ago we hosted a dinner for the three winners of the UVA Summer Logo Fellowships. Michael Muir of Winslow, Maine, Mike Charles of Phoenix, Arizona, and Dee Miller of Texarkana, Texas, were coming up the walk when Kyser, our five-year-old, shouted, "Dad, the turtles! Let's put the turtles on the table!" Of course! Why didn't I think of that? We scrambled downstairs, grabbed handfuls of ceramic, wooden, and plastic turtles from our collection, raced back up to the dining room, and placed them in an impromptu arrangement just as our guests came in the front door!

POOF! Back to the present. Why all the fuss over these turtle stories, anyhow? This mental turtle trip [is there a theorem in here somewhere?] helped me realize a secondary effect Logo has had on me. My experiences with the turtle as a cybernetic object in the computer environment have heightened my appreciation of the real thing. If there is a tortoisensitivity index (TSI), I expect that my scores have been progressing steadily upward since 1981. [It might be fun to construct the TSI instrument. Any ideas for items?]

No matter how wonderful the world of the computer is, it is the real world outside which matters most. The world of turtles, tots, teachers We are all connected, and we all count.

Do you have any turtle stories? I'd love to hear them. Thanks.

**FD 100!**

Tom Lough
Box 5341
Charlottesville, VA 22905

(For more information on the organizations mentioned in Tom's column, see page 5.)

# Little Kids and Logo

## Simple Gifts

**by Leslie F. Thyberg**

The words to an old Shaker hymn tell us
'tis a gift to be simple
'tis a gift to be free
'tis a gift to come round where we ought to be....

This month's column features the first of a two part series, devoted to a few "simple ideas for coming round where we ought to be." In last month's column I wrote of a developmental approach that can be employed for implementing Logo. At this point in the school year, we ought to be making the transition from the introductory phase to a genuine implementation of process-oriented learning. If you recall, in the first phase of implementation it is the teacher's responsibility to provide mental models and concrete opportunities for "learning how to learn."

Providing realistic, context-based opportunities for students to internalize strategies and increase independence require active involvement by the teacher. It is

> ...not enough to teach students that they should plan, revise, and edit their work. Those are strategies that students [may] believe they are using already. Teachers need to help students understand these processes more fully and manage them more effectively. [It is a] different and perhaps harder task to extend and elaborate on approaches that may be relatively firmly ingrained, to help students understand that there may be more than one approach to this task, and to teach them how to choose among the alternatives (Applebee, 1986, p. 73).

Applebee's research on writing as a process is very similar to the work of Swiss psychologist Jean Piaget. His research with children suggests that learning is a constructive activity. Children are natural and gifted learners. Their progress in learning develops over time along a continuum. It is appropriate, therefore, that Logo and its corollary (computer related curricula) be designed to help children progress developmentally. The computer is not a magical medium in and of itself, and can never replace the teacher; nor should it replace the use of manipulatives or physical activity. Do you find yourself saying, "Yes, but how?"

Much like the reading readiness or math readiness activities are planned for young students, so too, can similar kinds of activities be devised for Logo learning. For the sake of simplicity, I have divided these into two categories: off-computer (pre-computer activities) and on-computer (hands-on activities) readiness activities. These labels are representative of some of the things you can do as a teacher (without being a master programmer) to make Logo more workable for your students.

Off-computer activities have at least five overall benefits.

(1) They provide another kinesthetic connection for the child to experiment with that becomes an extension of the computer and perhaps the Logo microworld.
(2) They provide an avenue for reinforcement and review.
(3) They provide a mechanism for individualizing.
(4) At the preschool and primary level, readiness skills -- such as prereading, prewriting, and precomputing can be further developed.
(5) They serve as a good management tool for maximizing hands-on computer time -- especially if the computer-to-student ratio is not as rich as one would ideally have.

Playing turtle has become a classic activity for Logo learners. Clay turtles (even chocolate turtles) can be made as a tactile activity to make the cybernetic turtle a friendlier playmate. Turtles can be made by drawing a turtle base pattern which the child can cut out. Half a walnut shell or the cup from an egg carton can be used to glue to the base. Turtles can also be made from paper plates, tag board or simple construction paper that is then laminated for durability. Using the turtles they have made, children can then follow simple directions that are given (moving FORWARD, turning RIGHT, making shapes, etc.). Actual turtle paths and mazes can also be made. Shimabukuro, Green and Jaeger have some additional suggestions for turtle play. Using turtle puppets (store bought or student made) or hats with turtle attachments, students can "play turtle". Other children can give the turtle commands. This not only encourages the following of directions, but also involves discrimination and estimation skills as well creating opportunities for constructive social interaction.

Students can then write stores or give dictation on the characters that they have invented. While LogoWriter is not necessary to carry out this activity, its word processing features make it much easier than other versions of Logo, unless you are using Logo in conjunction with a class word-processing program. Certainly, as a part of reading readiness children can learn some of the basic principles of word processing to complement the graphics which they generate with Logo. Story telling and writing activities are effective language learning experiences for preschool and primary age children. Logo graphics and inventing adventures for the turtle are wonderful motivators around which to spin stories. An additional bonus is how quickly children learn to

recognize words and even the error messages. Periodic publication of a class newsletter ("Turtle Times" or "Turtle Soup") can give the students an audience and purpose for their writing.

Another useful off- and on-computer activity is the use of Logo Logs. These can serve as on and off computer guides. I have always had my students keep journals. Each student has a Logo Log or a Turtle Book (Rifkin. 1983) that contains things such as a list of what the students can do, task sheets, student records of "Commands I Know," operating licenses, and prints of pictures and/or dribble files. Notebook checklists can include open-ended discovery questions such as finding the dimensions of the monitor screen, as well as more prescriptive tasks from a series such as Molly Watt's *Welcome to Logo* workbook sheets published by Heath. Use of a checklist or task cards and a daily show and tell time can give the teacher an adequate sense of what learning is actually taking place as well as providing a form of structure and guidance for the child. The journal or Logo Log can also be used to file off-computer worksheets such as those Green, Jaeger and Shimabukuro provide.

Using some form of a licensing system has proven effective as a management scheme for monitoring computer use and creating a sense of responsibility on the part of the user. Licensing can begin with Learner's Permits and progress through actual Driver's Licenses up to a Chauffeur's class. To accompany this licensing scheme, my management rule: "Ask three before me" is a useful strategy to continue enforcing. Beyond helping the teacher manage time, "Ask three before me" has proven to be an easy, but effective scheme for causing the wildfire phenomenon, in which one student catches on to a new idea or discovers the cause of some bedeviling bug -- and can't wait to pass it along to other peers. This helps promote social interaction and the creation of a healthy learning environment.

Using weekly class meetings as a forum for children to present and discuss their favorite turtle discovery is a fertile bed for developing communication skills and building vocabulary, stimulating further growth and interest, and creating an environment that encourages open exchange of ideas. Dan and Molly Watt stress the importance of this kind of activity in their text, T*eaching with Logo*.

Next month's column will feature some tailor-made gifts for you -- some specific examples of off and pre-computer activates and lessons (after all, it will be Christmas soon!).

**Bibliography:**

Green, C. and Jaeger, C. (1984). *Teacher, kids, and Logo.* Irvine, CA: EduComp Publications.

Rifkin, B.. (1983, Jan.). Turtle folders help third graders. *National Logo Exchange.*

Shimabukuro, G. (1988). *Thinking in Logo: A sourcebook for teachers of primary students.* Menlo Park: Addison-Wesley.

Watt, D. and Watt, M. (1986). *Teaching with Logo: Building blocks for learning.* Menlo Park: Addison-Wesley.

Dr. Leslie F. Thyberg
5637 Rippey Street
Pittsburgh, PA 15206
or
c/o Chatam College
Woodland Road
Pittsburgh, PA 15208
AppleLink ALS 038

Leslie is a former classroom teacher. Ten of her thirteen years of experience were with children in grades three and under. She is a reading specialist and recently completed her doctorate in teacher development at the University of Pittsburgh. She is currently an assistant professor of education at Chatam College and travels as an independent computer education consultant.

---

**Information on the Organizations Mentioned in Monthly Musings**

To get a copy of the Whale Gifts catalog, call 800-227-1929 between the hours of 9 a.m. and 5 p.m. Eastern Standard Time. Connecticut residents call 388-4436. Or, write to Center for Environmental Education (CEE), Whale Gifts Catalog, 167-2 Elm Street, PO Box 810, Old Saybrook, CT 05475. A portion of the proceeds from the sale of Whale Gifts is set aside for the Sea Turtle Rescue Fund.

For information on the ridley turtle program, write to Carole Allen, HEART, PO Box 681231, Houston, TX 77268-1231.

The New York Turtle and Tortoise Society (NYTTS) is dedicated to the preservation and propagation of turtles and tortoises. They hold monthly meetings, publish a newsletter, and host turtle shows, seminars, and field trips. For more information, write to NYTTS, Suzanne Dohm, 365 Pacific Street, Brooklyn, NY 11217.

CEE, HEART, and NYTTS are non profit organizations.

# Logo Ideas

## A Logo Concert

**by Eadie Adamson**

Just before the winter break last year, I decided to introduce LogoWriter music to a group of fourth grade students. Looking back on the experience, I am amazed at how many important elements of learning Logo were absorbed by my students in a short period of time. In addition, they learned to think about music in ways new for many of them.

### Exploring TONE

Rather than immediately writing music, it makes sense to explore what the LogoWriter primitive TONE actually does. TONE takes two inputs, one for the frequency (how high or how low the sound is) and the second for the duration (how long). For example,

```
TONE 273 40
```

As you might expect, the first questions which came up were "What is the highest frequency and the longest duration you can use?" After allowing some initial experiments just to be sure everyone understood how TONE worked, we attacked the two questions about frequency and duration directly. This was a good excuse for a lesson in problem-solving.

I divided the class in half: one group was to answer the frequency question, the other the duration question. Before beginning, I asked them to think of themselves as working in a team. The whole team needed to participate and to double-check their answer. As a group we discussed strategies for finding the answer to either question. We agreed that changing only one thing at a time was the surest way to an answer. The frequency group would therefore experiment only with changing the frequency; the duration group would change only the duration. How could we tell when we had it? A little thinking and a little of the prior experimenting led the students to conclude that they would get an error message when the limit was exceeded. From this point, they reasoned, they could keep decreasing the input by one until they got no error message. That number would be the limit. They decided they should also do several retests to be certain they were correct about the maximum number.

### Procedures for the Notes

Once we successfully determined the limits (you can experiment yourself to find out what they are), I showed them how to write a procedure for a given note, using a procedure input, TIME, for the duration.

```
TO C :TIME
TONE 273 :TIME
END
```

We wrote a few procedures as a group using frequencies I wrote on the board. This process quickly becomes a repetitive task. An easy solution is to use LogoWriter's word processing capabilities to copy and paste:

Put the cursor on the T in TO and press Select.
Highlight the procedure by pressing the Down Arrow keys.
Press Copy to put a copy onto the clipboard.
Press Paste to put the procedure at the current cursor position
Change the letter and the frequency.

### MUSICTOOLS

Since the students now had a good understanding of how the music procedures were written, I decided to cut short this tedious process For the next class, I put MUSICTOOLS, a page of procedures for several octaves, on one computer and taught everyone how to save these tool procedures onto their own disk. Next we chose a new page and gave that page a STARTUP procedure which looked like this:

```
TO STARTUP
GETTOOLS "MUSICTOOLS
END
```

Using MUSICTOOLS as a tool page means that the procedures are loaded invisibly onto the page and work as if they are Logo primitives. This made it possible to focus on programming the music itself without the distraction of all the note procedures.

### A Little Math Before the Music

Before we began to program, we needed to standardize the durations for each note so that we would all be using the same quarter note value as we programmed and listened to our music. (Later we could change this.) We made a chart which everyone copied into their notebooks to keep as reference for eighth, quarter, half and whole notes.

We added pictures of the notes and then decided to make a quarter note have duration 20 (equal to one second), working out the related values of the other notes. In addition we took a brief excursion into the question of dotted quarter and half notes as well as what to do if we found a sixteenth note. We added these and their values to our charts.

Since so many of our students are musically oriented, I also made a representation of the keyboard with the frequencies (see the chart at the end of this article.). This could sit

neatly above the computer keyboard and was a valuable tool for many students.

**What about the music?**

I decided to begin writing music by working as a group to ensure that the steps were clear. I chose a familiar tune, since much depended on everyone's being able to hear whether or not this first try at programming was correct. Since we were near the holiday season, I found a copy of "We Wish You a Merry Christmas."

First we looked at the pattern of the music (each of us had our own Xerox copy), marking off the phrases. The commas in the text of the song provided good clues to phrase endings. We also tried singing and noticing where you catch a breath. After marking the phrases, we looked for a pattern-match with the first phrase of the music. We gave each phrase a letter plus a number, in this case MC1, MC2 and so on. A matching phrase got the same letter-number combination.

We needed to watch the notes carefully: the C above middle C was called HC, the C below middle C was LC. I wrote the letters above the notes for the few students who had difficulty reading the music, leaving them the task of assigning durations. Some students worked everything out using the music first, then began to program. Others who were more comfortable with reading music moved directly to the programming.

**Superprocedure and Subprocedures for the Music**

We wrote a superprocedure which included all the little phrase letters in the order that we had marked on our music sheets:

```
TO MERRY
MC1
MC2
MC3
      and so on...
END
```

Trying out MERRY before writing any of the subprocedures causes an error message: "I DON'T KNOW HOW TO MC1." Task one: write MC1. Then try it out by typing MERRY in the Command Center. It should play, giving you a chance to check the notes, and then give an error message which prompts you to write the next subprocedure. When all subprocedures have been written, the tune will play. If your music pauses, adding RECYCLE as the first line of the song may help. RECYCLE forces LogoWriter to collect unused memory (called garbage collection by computer scientists) before playing the tune and may eliminate that pause in the middle.

We found that starting with a superprocedure rather than subprocedures helped us work in a more orderly way, made it possible to catch errors easily before moving on, and also kept us informed of missing procedures. It was easier to try out the music by typing MERRY than to type a list of procedures (MC1 MC2 MC3) to hear how we were doing.

Incidentally, while working with music it is not necessary to flip to the front of the page to hear your sound. Press Down to get to the Command Center, Up to get back up to your procedures. Since you are not creating graphics, the page will not flip when you do this. You can hear your music and follow the procedure at the same time. This helps when debugging is necessary.

Making computer music in this fashion teaches several things about looking at music: watching for phrases, learning about the values of notes, and learning about the notes themselves and their relationships. Programming the music by analyzing it first and then working from a superprocedure helps to drive home the importance of the superprocedure. It also encourages an orderly working style and the writing of small procedures. Small procedures for phrases are much easier to debug.

When my class finished programming the first piece as a group, I allowed each student to choose a piece of his own to program. I brought in a collection of song books and made a copies of the music for students who needed them. We decided we'd have a concert and invite the students' classroom teachers when we had finished Before we got to concert time, some of the students managed to include the lyrics of their songs by using PRINT statements to make the words for each phrase appear before the phrase played. Be careful if you do this: you must go to the front of the page or the lyrics will be printed within your procedures. Given more time, the students could have expanded by adding graphics or animation as well.

A rest was needed for some of the music: we used WAIT and a number. In the newer versions of LogoWriter, you can also use TONE 0 followed by an appropriate duration. One boy needed a staccato sound which he produced by shortening the note and inserting a brief WAIT. Some began experimenting with composing their own music and as they did they carried over the concepts of subprocedures into their compositions. Others began connecting all of the pieces they had programmed into their own individual concert.

Concert day was just what one might expect: everyone in a frenzy to polish their pieces before their turn to play. Much to my delight, a number of the students were busy explaining to their teachers not just how music worked, but

## Logo Ideas -- continued

also about the importance of super- and sub-procedures and how to use them. A little music had driven home a number of important Logo concepts. The students had also had an opportunity to write procedures with variable inputs and to realize the value this held for making sounds. In addition, they had learned about tool procedures.

**If You Try This:**

- Spend time exploring TONE first.
- Use MUSICTOOLS (a file of notes already programmed) after your students have had a chance to experiment with writing a few procedures for different notes. (You could divide up the work and let the class make their own file of notes if you wished.)
- Write a STARTUP procedure to load the procedures on the MUSICTOOLS page as tool procedures so that the flip side is clear for music programming.
- Use a familiar tune at first.
- Watch the key signatures for sharps and flats.
- Use WAIT and a number or TONE 0 and a number for a rest.
- If you are using a Apple II GS, set the speed to normal using the Control Panel before working with music.

We did not use the music frequencies that are listed in the LCSI Reference manual. Jim Wingate of Packer-Collegiate in Brooklyn provided them for us. Jim is a musician and was unhappy with the LogoWriter frequencies. He worked these out with an oscilloscope until they were satisfactory to him.

**Jim's Frequency list**

These frequencies are for notes from the C below middle C through the C above high C. Only sharps are included. If you need G flat, use F# instead. If your music does not require all of these notes, you might want to write procedures for only the notes you need.

| | | | | | |
|---|---|---|---|---|---|
| HHC | 1175 | HC | 559 | C | 273 |
| HB | 1101 | B | 525 | LB | 258 |
| HA# | 1034 | A# | 495 | LA# | 243 |
| HA | 974 | A | 466 | LA | 229 |
| HG# | 13 | G# | 439 | LG# | 216 |
| HG | 859 | G | 419 | LG | 203 |
| HF# | 806 | F# | 391 | LF# | 197 |
| HF | 785 | F | 368 | LF | 181 |
| HE | 715 | E | 346 | LE | 171 |
| HD# | 669 | D# | 326 | LD# | 161 |
| HD | 631 | D | 303 | LD | 152 |
| HD# | 593 | C# | 290 | LC# | 143 |
| | | | | LLC | 135 |

Eadie Adamson
Allen Stevenson School
132 East 78th Street
New York, New York 10021

# Stager's Stuff

## Physics for Kindergartners

**by Gary S. Stager**

This month we will look at ways in which you and your students can use Logo to explore elementary physics concepts such as velocity, acceleration, force, and friction. The first part of the article will describe ways in which Logo can be used to simulate different aspects of motion and the second part of the article will focus on Logo as a tool for collecting, recording, and calculating experimental data. Dreaded word problems concerning rate, distance, and time can be brought to life using these ideas.

**Turtle Racer**

One of the first questions I ask students and teachers when introducing them to Logo is, "How would we make the turtle appear to be driving?" Someone usually speculates that the following solution will do the trick:

```
FORWARD 50
```

We quickly observe that the turtle did move, however it seemed to fly or jump rather than drive. I then re-state my question by asking, "OK, how would we make the turtle move with constant speed so that it seems to drive instead of jump?" A student usually mentions the word repeat in their proposed solution, independently of whether the Logo command, REPEAT, has been introduced yet. "What do we want to repeat?", I then ask. Eventually we arrive at a Logo expression like the following:

```
REPEAT 100 [FORWARD 1]
```

Voila! The turtle drives! I then ask, "How would we make the turtle drive faster?" The answers I generally receive involve increasing one of the numbers, but which one? A good illustration would be to increase the 100 after REPEAT (incidentally, 100 was chosen arbitrarily) and run the expression again.

```
REPEAT 200 [FORWARD 1]
```

The audience usually observes that although the turtle went twice as far, the turtle's speed was the same as in the first experiment. By process of elimination we discover that we should increase the input to FORWARD.

```
REPEAT 100 [FORWARD 2]
```

The turtle went faster — Twice as fast! REPEAT 100 [FORWARD 5] would make the turtle drive five times as fast as the original. How do we know that the turtle went twice as fast? The answer lies in the fact that the turtle went twice the original distance in the same amount of moves (time). To state this in traditional scientific language we

would say, "The turtle traveled twice the distance in the same amount of time, therefore it's rate of speed is double that of the original." Speed is a quantity dependent on the relationship between the distance traveled and the amount of time the trip took, i.e , 55 miles per hour.

Rate of SPEED = DISTANCE / TIME

Physicists give speed the fancy name velocity, therefore:

VELOCITY = DISTANCE / TIME

**Friction Rubs Me the Wrong Way**

What happens when you type the following?

```
REPEAT 100 [FORWARD 5 WAIT 1]
```

Wait causes Logo to pause. In LogoWriter, Wait 20 = 1 second. In most other versions, WAIT 60 = 1 second.

The turtle now moves forward a little and then briefly pauses. The WAIT 1 command is being used to simulate the force of friction. Frictional force is a force which opposes the relative motion of an object in contact with another. Here we are dealing with kinetic friction. We are concerned with the force necessary to maintain a constant velocity once an object is in motion. (Kinetic friction is the product of the kinetic coefficient of friction and the normal force on the object.) What happens if you increase the input to WAIT? The more frictional force present, the more force that is necessary for the object (turtle) to maintain the same velocity. If you increase the frictional coefficient by five times, you must increase the force (the input to FORWARD) by 5 times to maintain a constant speed.

**Acceleration**

Another phenomena we can easily simulate in Logo is acceleration. Acceleration is the rate at which velocity changes, increases or decreases, over a specific distance. When we step on the gas pedal of our car we accelerate and when we hit the brake we decelerate or experience negative acceleration. The formal definition of acceleration is:

ACCELERATION =
DISTANCE TRAVELED /
CHANGE IN VELOCITY

In LogoWriter we can easily simulate the gas pedal/brake analogy and a similar effect can be achieved in other versions of Logo with a bit more programming.

LogoWriter has a powerful feature, called events, built into the language. Events are linked to control keys which interrupt whatever Logo is doing to run a specific list of commands. Events are very useful as word processing macros, for control in video games, and for debugging. (You should read the LogoWriter materials to learn more about Events). The format for LogoWriter events is:

```
WHEN "Q [FORWARD 50 RT 90]
```

Whenever Control-Q is pressed the turtle will now do FORWARD 50 RT 90. To erase events you must type, CLEAREVENTS. Try it!

We will make one event key the brake pedal (decelerator) and one the gas pedal (accelerator) and command the turtle to move with constant velocity unless we alter it's motion with either event.

```
TO STARTUP
MAKE "SPEED 1
WHEN "Q [MAKE "SPEED :SPEED - .1 ]
WHEN "W [MAKE "SPEED :SPEED + .1 ]
END

TO GO
FORWARD :SPEED
GO
END
```

STARTUP initializes the global variable, SPEED, and the two LogoWriter events. GO is a recursive procedure that instructs the turtle to move with the velocity specified by :SPEED. Whenever Control-W is pressed, the turtle will accelerate and whenever Control-Q is pressed it will decelerate. You may wish to change the turtle's shape and head it in the right direction.

To start the simulation, type the following:

```
STARTUP
PU
SETSH 27
SETH 90
GO
```

Hold down Control-W until the turtle is moving very quickly and then hold down the brake, Control-Q until the the turtle is moving very slowly. Now press Control-Q until the turtle actually pauses in one place. When an object reaches this state, as in when you throw a ball up in the air, it pauses for an instant before falling back to earth. At this point the object has zero velocity but is still accelerating.

This concept is confusing to understand so try holding down Control-Q. The turtle (or car) is moving backwards because it now has negative velocity. If you continue to hold down Control-Q the turtle will speed up in reverse.

## Stager's Stuff -- continued

To make the concepts surrounding acceleration even clearer you may wish to add the following line to your STARTUP procedure for monitoring the velocity of your turtle:

```
WHEN "Z [SHOW :SPEED]
```

Now whenever Control-Z is pressed and the GO procedure is running, the turtle's velocity will be reported to you. Use this new feature to demonstrate the turtle's velocity at the point of that it stops completely.

**Acceleration Simulation for Other Versions of Logo**

```
TO STARTUP
MAKE "SPEED 1
END

TO GO
IF KEY? [PEDALS RC] <- In LCSI
FORWARD :SPEED        dialects,
GO                    replace KEY?
END                   with KEYP.

TO PEDALS :KEY
IF :KEY = "Q [MAKE "SPEED
   :SPEED - .1]
IF :KEY = "W [MAKE "SPEED
   :SPEED + .1]
END
```

Then type

```
STARTUP
PU
SETH 90
GO
```

to begin. Press Q or W instead of Control-Q or Control-W

**Tools for Collecting and Interpreting Data on Velocity**

The rest of this article will provide you with the tools necessary for calculating velocity and average velocities of an object. An important component of any science curriculum is the scientific method and the following tools encourage students to conduct multiple trials and record their experimental data before drawing conclusions. By allowing Logo the pleasure of doing all of the calculations students can pay attention to experimental conditions and accuracy of their data.

The experiment merely involves timing an object traveling a specific distance. A nice motivation for the activity is to establish a race for student built cars or calculating the velocity of a thrown ball. Students should record multiple trials for the velocity of each object in order to achieve reliability of the experimental data.

If you have LEGO TC logo, you can use the procedures in this article to enhance the Soapbox Derby activity in the *Getting Started* student guide. Have teams of students build a car which they think will be faster than everybody else's car and refine the car based on its performance. You may wish to add the simple timer idea explained on page 38-39 of the *Getting Started* guide. Connect two optosensors to sensor ports 6 and 7 and put light bricks opposite the sensors. Be sure to write down how far apart the two sensors are! Your calculations will be more accurate the farther apart the two sensors are. If you are using LEGO TC logo, you should type the following procedure into the procedure center (flipside) of your page:

```
TO STOPWATCH
LTO 6
WAITUNTIL [SENSOR?]
RESETT
LTO 7
WAITUNTIL [SENSOR?]
SHOW TIMER / 10
STOPWATCH
END
```

Type STOPWATCH and roll your object past the two sensors or down a ramp towards the sensors. Be sure to pass the optosensor plugged into port 6 first.

If you don't own LEGO TC logo you can conduct the experiment the old-fashioned way (the way last year's class did) with a stopwatch and matchbox cards. Just substitute your eyes and a stopwatch for the optosensors and the LEGO logo STOPWATCH procedure.

It is important to note that the velocities calculated in this type of experiment are average velocities. When the car hits the end of the ramp some momentum is lost and various forms of friction act upon the car. These events and others may cause the velocity of the object to vary.

Be sure to keep your unit for measuring distance consistent. In other words, if one person is measuring the distance traveled by their object in inches, everybody should record their measurements in inches.

The procedure, SPEED, is a reporter which takes two inputs, the distance traveled by an object and the time it took to travel that distance, and outputs the rate of speed attained by the object.

```
TO SPEED :DISTANCE :TIME
OUTPUT :DISTANCE / :TIME
END

SHOW SPEED 25 5
5
```

If the unit of measurement for distance was inches and the elapsed time was measured in seconds, then the speed of this object was five inches per second.

The VELOCITY procedure differs slightly from SPEED in that it's input must be a two element list containing the distance traveled and the time it took for the object to travel that distance. VELOCITY is used as a subprocedure in later procedures.

```
TO VELOCITY :LIST
OUTPUT (FIRST :LIST) / (LAST :LIST)
END

SHOW VELOCITY [49 7]
7
```

AVG is a reporter procedure which takes a list of numbers as input and by using the subprocedure, ADD, outputs the average of any list of numbers. Both procedures are used in the AVG.VELOCITY procedure listed below. AVG and ADD are a useful addition to your Logo toolbox.

```
TO AVG :LIST
OUTPUT (ADD :LIST) / COUNT :LIST
END

TO ADD :LIST
IF EMPTY? :LIST [OUTPUT 0]
OUTPUT (FIRST :LIST) + ADD BF :LIST
END

SHOW AVG [57 43 22 3 15]
28
```

The reporter, AVG.VELOCITY, takes a list of distance and time lists as input and outputs the average of the entire collection of lists. This is an extremely useful way to find an average velocity for an object which had many trials or to find a class average. Each list in the input list to AVG.VELOCITY is in the same format as used by the VELOCITY procedure. The code in some of these procedures is fairly complex so don't worry if you don't fully understand them. My purpose is to provide you and your students with useful tools.

```
TO AVG.VELOCITY :LIST
OUTPUT AVG AV :LIST
END

TO AV :LIST
IF NOT LISTP FIRST :LIST [OUTPUT
   :LIST ]
OUTPUT AV (LPUT VELOCITY FIRST :LIST
   BF :LIST)
END

SHOW AVG.VELOCITY [[20 2][30 5]
   [45 9]]
7
```

All of the tool procedures shown above can be used in other situations. Since they are all reporters their outputs can be passed to procedures which would graph them or convert them to different units. How would we write a procedure to convert a velocity measured in inches per second to miles per hour? Since 1 Mile = 5280 Feet = 63,360 Inches and 1 Hour = 60 Minutes = 3600 Seconds.

```
TO VELOCITY.MPH :VELOCITY.LIST
OUTPUT ((FIRST :VELOCITY.LIST) /
   63360) / ((LAST :VELOCITY.LIST) /
   3600)
END
```

Note: the input list to VELOCITY.MPH should be in inches per second

```
SHOW VELOCITY.MPH [40 .4]
5.7
```

This tells us 40 inches per 4/10 seconds is equal to 5.7 miles per hour

The MPH procedure takes a velocity expressed in inches per second and outputs the same velocity expressed in miles per hour. The following example illustrates how you can combine the reporters (operations) to report a result recorded in inches per second in miles per hour.

```
TO MPH :VELOCITY
OUTPUT (:VELOCITY * 3600) / 63360
END

SHOW MPH AVG.VELOCITY [[20 2][30 5]
   [45 9]]
4
```

If you wish, you can create any number of conversion reporters, including ones which will take an input in the U.S. system of measurement and output the result in metric units. CPS converts from inches per second to centimeters per

## Stager's Stuff -- continued

second. KPH converts from miles per hour to kilometers per hour. The following are some ways in which you may combine several of the tool procedures we've created:

```
TO CPS :INCHES.PER.SECOND
OUTPUT :INCHES.PER.SECOND * 2.54
END

SHOW CPS VELOCITY [20 2]
25.4
   (centimeters per second)

TO KPH :MILE.PER.HOUR
OUTPUT :MILES.PER.HOUR * 1.609
END

SHOW KPH MPH AVG.VELOCITY [[20 2]
   [30 5][45 9]]
64
   (kilometers per hour)
```

**Want to Know More?**

I've only provided a tiny glimpse of the numerous ways Logo can be used to explore physics. If you wish to learn more, I suggest the following sources of ideas and information.

Abelson, Harold and diSessa, Andrea (1981) *Turtle Geometry: The Computer as a Tool for Exploring Mathematics.* Cambridge: MIT Press.

Hurley, James. (1985) *Logo Physics..* New York: Holt, Rinehart, and Winston.

LEGO TC logo (1987). *Student Guides.* Enfield, Connecticut: LEGO Systems, Inc.

*LogoWriter Reference Guide, Teacher's Guide, Project Booklets, and Activity Cards.* (1986). Vaudreuil, Quebec, Canada: Logo Computer Systems, Inc.

Our old friend, Tom Lough, has served as an inspiration to myself and many others and is an excellent source of ingenious ideas and Logo physics activities. In particular, see his column (coauthored with Steve Tipps) in *The Journal of Computers in Mathematics and Science Teaching.*

Gary S. Stager is the Vice President of SIGLogo and Director of Training for the Network for Action in Microcomputer Education in New Jersey. He can be reached at:

Gary S. Stager
5 Eastside Avenue - 2F
Wanaque, NJ 07465
CIS: 73306,2446
Applelink: K0331

# The SIGLogo Song

## What Do We Stand For?

**(c) 1988 by Peter Rawitsch**

(Editor's note: Since many of you were not able to attend the SIGLogo meeting in Dallas last June, I want to share with you the song that our president, Peter Rawitsch, wrote and performed there. Unfortunately we can't provide you with a musical chip so that you can hear it. You'll just have to imagine Peter and his guitar!)

**CHORUS:**

What do we stand for and what do we mean?
Are we the "cig" in some cigarette machine?
What do we stand for? Does anyone know
what we mean when we say that we're using Logo?

I surveyed some members as they walked through the door
and I asked them to tell me what SIGLogo stood for.
They said we're a special interest community
but on just what is Logo they could not agree.

Chorus

One classroom teacher said, "Logo is great"
and that her math and science lessons now integrate.
Her kids type in the data they collected in school
to make charts and graphs with their own Logo tool.

Chorus

One sales rep brought some new books to sell.
He claimed a good Logo book was so easy to tell.
It should have a green cover and cute turtles galore
and show kids RIGHT 90 before they explore.

Chorus

One administrator said, "Logo is fine."
Even he'd received inservice training one time.
But it was so long ago, he couldn't remember it all.
Still he thinks kids should use turtles to learn how to draw.

Chorus

A college professor said, "Logo's a word.
It's sometimes an adjective and sometimes a verb.
It can't make kids smarter; that's a job we must do.
It gets all its power from me and from you!"

Chorus

There was something recursive in what everyone said.
Maybe working with children is our common thread.
I think the kid in us all loves to learn something new
and there's so much to be gained from each point of view.

Chorus

# Logo LinX

## A Warped Idea

**by Judi Harris**

Logo LinX articles are about weaving Logo into traditional subject curricula; using Logo to teach, rather than teaching Logo. This month, I'd like to show you how weaving with Logo can be woven into Social Studies.

Does this look familiar?

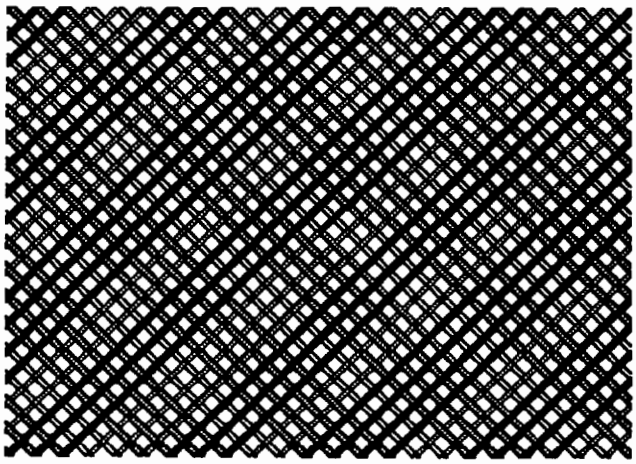


Figure 1

In every class, there seems to be at least one student who, given the opportunity, repeatedly chooses to explore these Logo plaids, often to the chagrin of his teacher, who would much rather that he explore more sophisticated procedural structures than this:

```
TO FIG1
RIGHT 45
REPEAT 100 [SETPC RANDOM 4
   FORWARD 9876 RIGHT 90]
END
```

I have observed several methods for discouraging persevering with plaids. Some teachers outlaw wrap-arounds; others limit all inputs to 140 or below; still others patiently wait until the child tires of entering multi-digit REPEAT values. Instead, why not make the plaids a subject of historical inquiry?

### Scottish Fabrications

A *tartan* is a plaid pattern that is now primarily associated with Scottish clans, although such heraldic devices appeared in many countries throughout history, from Japan to Italy. Scottish tartans were originally worn by both men and women as a large rectangular cloth draped over the shoulder, belted at the waist, falling into a pleated skirt. The earliest references to tartans date back to the 13th century. Later, the tartan was separated into a kilt (skirt) and a plaid (shoulder-draped shawl).

Scottish tartans (in Gaelic, *breacans*) were usually woven from wool. All known tartans had colored stripes in two directions, crossing at right angles to form a regular pattern. The cloth could be of any size, and the colors of any intensity, but the pattern had to be proportionally identical across garments woven for a particular group.

### A Different Mind Sett

The design of a Scottish tartan is called a *sett*. Different setts first represented different districts of Scotland; later they became symbols of large families or clans. After 1782, tartans were used to identify different military regiments. Early sett patterns were recorded with popsicle-sized *pattern sticks*. The exact number of threads of each color in the sett were wound around the stick in the correct color sequence, then fastened. Color orders were most often symmetrical in mirror- image form, i.e.,

blue-blue-green-red-red-white-white-red-red-green-blue-blue

The sett was the same for weft and warp, the two directions of woven cloth. Colors were often quite bright. Blues and purples were favorites at the end of the sixteenth century, yellows and blues 100 years later, and today, most tartans are primarily red or green.

### Plotting a Weave (Instead of Weaving a Plot)

Why not encourage your students to do some tartan exploration with Logo? First, they could plan the color sequence with simple PATTERNSTICK procedures, such as these, each of which outputs an ordered list of values that will be used by the SETPC primitive.

```
TO PATTERNSTICK1
OUTPUT [2 2 2 3 3 3 3 3 1 1 0 0 1 1
   3 3 2 2 1 1 1 1 2 2 3 3 1 1 0 0 1
   1 3 3 3 3 3 2 2 2]
END

TO PATTERNSTICK2
OUTPUT [1 1 2 2 2 3 3 3 0 0 0 1 1 2
   2 2 2 2 3 3 0 0 0 0 3 3 2 2 2 2 2
   1 1 0 0 0 3 3 3 2 2 2 1 1]
END
```

One of the best ways to decipher and translate the act of weaving into procedural form is first to do it without a computer. A strong piece of cardboard, scissors, a ruler, masking tape, and scrap yarn are all that are needed for hand weaving. Help the students to cut evenly spaced 1/2-inch

## Logo LinX -- continued

slits across two opposite sides of the cardboard, then use them to stretch parallel warp threads lengthwise on the miniature loom, securing them with masking tape on the back of the cardboard.

**'Twill be Bonny**

Weft threads can be shuttled across the fabric width in any regular pattern, according to the color sequences specified in PATTERNSTICK procedures. Scottish tartans were woven in a twill pattern, or, over two, under one, over two, under one warp threads, with the topside weft threads arranged in diagonals:

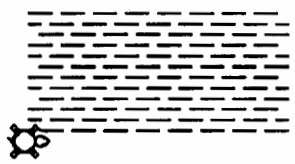


Figure 2

Warp (vertical) and weft (horizontal) shuttle passes can be simplified and executed in Logo like this:

```
TO WARP :THREADLENGTH
SETH 0
BACK :THREADLENGTH
FORWARD :THREADLENGTH
END

TO WEFT :THREADLENGTH
SETH 90
FORWARD :THREADLENGTH
BACK :THREADLENGTH
END
```

Weaving could then be done recursively, alternating shuttle passes between warp and weft in each recursion, and shifting X and Y starting positions from one recursion to the next.

```
TO WEAVE :COLORLIST :THREADLENGTH
              :XPOSITION :YPOSITION
IF EMPTYP :COLORLIST [STOP]
SETPC FIRST :COLORLIST
PU
SETPOS LIST :XPOSITION 0
PD
WARP :THREADLENGTH
PU
SETPOS LIST 0 :YPOSITION
PD
WEFT :THREADLENGTH
WEAVE :THREADLENGTH (BUTFIRST
   :COLORLIST) (:XPOSITION + 2)
   (:YPOSITION - 2)
END
```

An interesting set of Logo weaving explorations could emerge as students investigate other weaving methods. For example, dotted lines similar to those pictured above could be used instead of solid lines in WEFT and WARP procedures.

**Interwoven Objectives**

Weaving by hand or computer can be almost meditation-like. Perhaps this is what that persistent Logo plaid maker seeks when he gleefully enters multi-digit values for REPEAT and FORWARD, then enjoys the resulting graphic effects.

Indeed, weaving is a powerful metaphor for making unified wholes out of disparate parts by way of a repeated pattern of action. The shuttle is the new idea that restructures our perceptions. The resulting fabric often appears to be of quite a different form, even though its elemental components are quite familiar.

Weave, weave, weave me the sunshine
out of the falling rain.
Weave me the hope of a new tomorrow;
Fill my cup again!

—Peter Yarrow, 1972
(performed by Peter, Paul and Mary)

Judi Harris taught students in Philadelphia-area elementary through graduate schools to use computer in teaching and learning for six years. She now does similar work at the University of Virginia, where she is completing her doctoral work in Instructional Technology. She can be reached at

Judi Harris
621F Madison Avenue
Charlottesville, VA 22903
CIS: 75116,1207
BitNet: jbh7c@Virginia

# Logo Connections

## LogoWriter Connections

**by Glen L. Bull and Gina L. Bull**

LogoWriter has many powerful connections to the writing process, as its name emphasizes. It provides a useful metaphor for thinking about files (pages in a scrapbook). It automatically saves work when a student leaves a file, reducing the possibility that work will be lost. LogoWriter is available in equivalent versions for Apple, IBM, and Commodore computers. It also has an innovative site licensing plan with a home use option, making it possible for children in schools to take LogoWriter home.

At the time LogoWriter was introduced, most of the Apple computers in schools only had 64K (kilobytes) of memory. It was not possible to both use the Apple ProDOS operating system, and fit the LogoWriter language into 64K of memory. Logo Computer Systems found it necessary to write a smaller operating system which left more room for LogoWriter.

If LogoWriter had not been developed for 64 kilobyte machines, many schools would not have been able to use it. However, the necessity of using a non-standard operating system had several undesirable side effects. A proprietary operating system made it necessary for students to store LogoWriter work on a separate disk from all their other work. It also made it necessary to retype older programs from other dialects of Logo, since they could not be loaded directly into LogoWriter. In spite of the fact that identical versions of LogoWriter were available for three different computer systems, it was not possible to transfer LogoWriter programs from one machine to another without retyping them. This meant that students who used Apple's at school and IBM's at home could not easily carry their work back and forth. It also meant that students could not have access to libraries of Logo pictures which teachers had developed in other versions of Logo, and could not use other sources of clip art and pictures from sources such as Print Shop archives.

Now an update of LogoWriter which addresses these limitations has been released for the ProDOS operating system. In this version, LogoWriter work can be saved on the same disk with Appleworks files, Mousepaint pictures, and other work. Version 2.0 of LogoWriter for the Apple does require 128K of memory, but this configuration is becoming more common.

### 1. Loading Other Logo Programs into LogoWriter

The LOADTEXT command in Version 2.0 of LogoWriter makes it possible to load programs from other versions of Logo into LogoWriter. It may be necessary to edit some of the procedures so that they conform to LogoWriter syntax before they will run properly. However, that is much better than completely retyping them.

The WORDTOOLS provided with LogoWriter make it possible to search for every occurrence of a particular word and change it to a different word. For example, the command EMPTYP in Apple Logo and Apple Logo II is EMPTY? in LogoWriter. The word tools could be used to search for every occurrence of "EMPTYP" and change it to "EMPTY?".

If an earlier Logo program is stored on a DOS 3.3 disk, it will be necessary to use the ProDOS User's disk to transfer the file from DOS 3.3 format to ProDOS format before LogoWriter can access it. However, the conversion program on the ProDOS User's disk is easy to use, providing menus of choices at each step of the way.

We have also found that for some reason it is also necessary to load Terrapin programs into FRED Writer and save them as FRED Writer files before they can be loaded into LogoWriter. FRED Writer ("FRee EDucational Writer") is a public domain word processing program used in many schools, and should be available from your local Apple users group. There probably is a more direct way to accomplish this step, but since resaving the Terrapin file as a FRED Writer file works, we have not searched very hard for a different way. (If anyone knows of a more direct way, we would be happy to hear from you.)

When you load programs into LogoWriter, be sure that you are on the flip side of the LogoWriter page where programs are stored as you use the LOADTEXT command.

### 2. Loading Text into LogoWriter

Since a Terrapin Logo file resaved as a FRED Writer file can be loaded into LogoWriter, it is obvious that FRED Writer text files can also be loaded directly into Logo. A story or poem written in FRED Writer can be loaded into LogoWriter, and illustrated.

Appleworks files can also be loaded into LogoWriter. In the case of Appleworks files, it will be necessary to first save the file as an ASCII (text) file. To accomplish this, use the print command, OpenApple-P. When Appleworks asks:

Where do you want to print the file?

you should select Option 3 on the menu:

3. A text (ASCII) file on disk

This will save the file to disk as a standard text file, without the Appleworks control codes embedded.

## Logo Connections -- continued

**3. Loading Pictures into LogoWriter**

The new ProDOS version of LogoWriter also has a LOADPIC command which permits pictures as well as text to be loaded into LogoWriter. This means that pictures from other dialects of Logo can be loaded into LogoWriter. Pictures created with graphics tablets such as the Koala Pad and Animation Station can also be used. (If the pictures have been saved on a DOS 3.3 disk, it will be necessary to first transfer them to ProDOS format by using the ProDOS User's disk.)

Both commercial and public domain libraries of "clip art" pictures are now emerging. Many Apple users groups have 30 or 40 disks of public domain pictures available. In some cases the pictures are stored in a more compact format on the disk to save room. In those cases it will be necessary to "unpack" the pictures to restore them to standard format before they can be used. In the September 1988 issure of the *Logo Exchange*, Judi Harris explains how Print Shop picture libraries can also be converted for use with Logo (and Version 2.0 of LogoWriter).

**4. Transferring Programs from Apple to IBM**

It is often the case that children who use Apples in the classroom have access to an IBM at home. A LogoWriter site license makes it possible for a student to take home a copy of the LogoWriter program if the home use option is selected. Although this solves the legal issue, the mechanical issue of transferring a file from an Apple disk to an IBM disk remains.

There are a number of ways in which this can be accomplished. For example, there are cards such as the Microsolutions Matchpoint card which permit an IBM disk drive to read an Apple ProDOS disk. An IBM in the principal's office could be equipped with a card of this type and then used to transfer files from an Apple disk to an IBM disk. A card of this type is handy for other purposes as well, since it is surprising how often it is necessary to transfer an Appleworks file to IBM Word Perfect format, or vice versa. Secretaries who often wind up retyping files of this kind will be very grateful if a more effective method is provided.

**5. Telecommunications**

Schools are increasingly using telecommunications to exchange messages, access databases, and send files from one site to another. A number of good, public domain telecommunications programs are now available as well. These include programs such as Talk Is Cheap (TIC) and FRED Sender. To transmit LogoWriter procedures via electronic mail, first use the SAVETEXT command to save the procedures as a text file. Then send the file using the telecommunications program (commercial or public domain) of your choice. After the file has been captured at the other end, load it into LogoWriter by using the LOADTEXT command. It is also possible to transmit the entire LogoWriter file, but sending the complete file will take longer than just sending the text of the procedures. The telecommunications software used must also be capable of sending binary as well as text files.

Telecommunications not only provides a means for schools to exchange LogoWriter programs, it also provides a means to transfer LogoWriter files from an Apple disk to an IBM disk. Follow the same procedure as before, but send the file with an Apple computer and capture it with an IBM computer. Voila! The file is transferred.

In order for tranferred procedures to be useful, it will be necessary for IBM LogoWriter to have LOADTEXT and SAVETEXT commands similar to the equivalent commands in Apple LogoWriter. As this is written, the 2.0 update of IBM LogoWriter is scheduled for release. We have not actually seen it yet, but it should be available by the time you read this.

**6. Speech Synthesizers and Videodiscs**

Since LogoWriter is especially targeted for ease of use by younger children, a talking version of LogoWriter is desirable. Other versions of Logo can access speech synthesizers such as the Echo GP and the Votrax by means of the serial port; unfortunately the LogoWriter manual does not list any direct way of accessing this port. However, the command .OUT, which is undocumented (in our manual, at least), makes it possible to send a character out the serial port.

To send the letter "A" out the serial port, the .OUT (%OUT% in Version 1.1) command would be used in this way:

```
.OUT ASCII "A
```

To send a different letter, any other character could be substituted for "A." A short procedure, SEND.SERIAL, can be constructed to send an entire word or sentence using the .OUT command.

Some videodisc players can also be controlled via commands sent through the serial port. Sue Anderson has been using a videodisc player controlled by Logo with preschool handicapped children at Jackson Via School in Charlottesville, Virginia. The .OUT command permits the videodisc player to be used with LogoWriter as well as other versions of Logo.

As we write this, the .OUT command only works with a serial card in Slot 1 of the Apple. However, Chris Thamm, an intern at Logo Computer Systems Incorporated (LCSI) has written a utility called "Multi-Slot" which will permit the .OUT command to be used with a serial card in any slot in the

Apple. Contact Alain Tougas, Technical Support Coordinator at LCSI, if you would like to obtain a copy of this utility.

Many schools still have substantial numbers of Apple IIe computers which only have 64K memory. Obviously it would be better to have every computer equipped with a full 128K of memory, especially now that memory upgrades are available for less than $100. However, even if most of the computers in a school have only 64K, it still may be possible to take advantage of some of the new "connectivity" features of the upgraded version of LogoWriter. The ProDOS version of LogoWriter has a conversion menu which makes it possible to convert a ProDOS LogoWriter file to the 64K version of LogoWriter.

If a school has a single Apple IIe equipped with 128K, files from Apple Logo, Terrapin Logo, Appleworks, FREDWriter, MousePaint, Print Shop, etc., can be read into the ProDOS version of LogoWriter and saved as LogoWriter pages. Then the convert feature in ProDOS LogoWriter can be used to change these LogoWriter pages to the 64K LogoWriter format for use on all the other computers in the school.

The enhanced 2.0 version of LogoWriter offers many new possibilities for using LogoWriter with other kinds of software and hardware. These capabilities have been available in other versions of Logo such as LCSI Logo II for the Apple. Their inclusion in Version 2.0 of LogoWriter is evidence of maturation of the product. LogoWriter, an innovative way of thinking about educational computing, is now even more powerful.

**Additional Extensions**

In addition to the Multi-Slot extension to LogoWriter mentioned in this month's column, other utilities are now available for Version 2.0 of LogoWriter. These include:

a. A driver which will allow LogoWriter to print in color, using the ImageWriter II color printer
b. A driver which will allow LogoWriter to be used with the Valiant turtle

New drivers under development which have not been completely tested will allow LogoWriter to be used with the Pioneer 4200 and Pioneer 2000 videodisc players. In order to obtain copies of these utilities for Version 2.0 of LogoWriter, send a blank disk to Alain Tougas of Logo Computer Systems, Inc. Be sure to indicate which utility or utilities you are requesting.

Glen and Gina Bull, Curry School of Education
Ruffner Hall, University of Virginia
Charlottesville, VA 22903
Glen's BITNET address: GLB2B@Virginia
Gina's BITNET address: RLBOP@Virginia.

## Mixing Procedures and Data in Logo

# A Mad-Libs Implementation

**by Michael Katz**

Mad-Libs programs are often used as an introduction to Logo's list processing capabilities. In these programs, the computer picks words randomly from lists of nouns, verbs, etc. and constructs silly sounding sentences from them. There are several ways to implement this idea. I want to share an elegant solution which relys on a little-used but very powerful Logo primitive.

First let's consider a traditional implementation of the program. This approach is to create our word lists as global variables using the MAKE primitive. We can write procedures such as TYPE.NOUN and TYPE.VERB which randomly choose and display a word from their corresponding list. Then we can write Mad-Libs procedures which are a collection of TYPE and PRINT statements that use the procedures TYPE.NOUN, TYPE.VERB etc. at points where we want the computer to pick and insert words. Having to explicitly use TYPE and PRINT this way becomes tedious especially with long Mad-Libs. Also, each time we want to add to our word lists we have to use MAKE again and if we want to add another part of speech such as adjectives we have to add a TYPE.ADJECTIVE procedure. All of this repetition should be inspiration to look for a more efficient way to write this kind of program.

I have found an elegant implementation which uses Logo's amazing TEXT primitive as a way of converting procedures into data. (TEXT is available in most common versions of Logo except LogoWriter.)

The TEXT command takes the name of a procedure as its one input and gives as output a list of sublists. Each of the sublists is a list of the text found on one line of that procedure. (Actually the first sublist is the input (parameter) list of the named procedure. Thus, to get at the list of lines we must use BUTFIRST on the output from TEXT.)

Being able to think of data as procedures and procedures as data is one of the most powerful features of Logo and Lisp, the artificial intelligence language upon which Logo is based. In the Mad-Libs program, the TEXT command is really just used to make data entry more convenient, but consider as you play with the program the power of having a primitive command that will convert a procedure into a list of the text of that procedure and another primitive command, RUN, that will take a list of text and interpret it as a list of commands.

The main advantage the TEXT command offers our Mad-Libs program is that we don't need to worry about Logo

## Mad Libs -- continued

list structures when defining our word lists and stories. For instance, instead of putting our noun list in brackets we just make a procedure called NOUN where our nouns are listed one per line. Now BUTFIRST TEXT "NOUN produces the noun list. Note that there is no use of the MAKE statement, no brackets to worry about and no ugly exclamation points breaking up long lists in strange places (although this is not a problem for some Logo systems). Better yet, we can use this same technique in making our Mad-Libs stories. For instance, we can make a story called STORY1 by just making a procedure called STORY1 where the text of the story is just the text of the procedure. This is much more convenient than having to precede everything with a TYPE or PRINT command.

A second advantage of using the TEXT command is that adding new parts of speech to the program becomes very easy. We use a special procedure called GRAMMAR whose lines are each the name of a part of speech that the program recognizes. This way you can use (or not use) parts of speech as you desire. For instance, to include adverbs in your Mad-Libs, you would just add a word like ADV on its own line in the GRAMMAR procedure and create a procedure called ADV whose lines were each a single adverb. Then, any time ADV appeared in one of your stories, it would be replaced by one of the lines from your ADV procedure when the story was told. Get it?

But wait, there's more. The program also does neat things like check for periods (not commas, though — you can add this on your own as a challenge!) so as not to confuse "CHICKEN." with "CHICKEN" There is also a delay to give you time to read the story as it appears. And all this for the ridiculously low price of not ten, not seven, but four small procedures (plus the data procedures you add).

Students using the program get a good sense of how the different parts of speech can and cannot be used, particularly when they try to make up their own stories or try to add new parts of speech to the GRAMMAR procedure. The program is good for beginners because it lets them create and change words lists and stories easily without having to use MAKE or understand list structures. It might even be used as a fun introduction to the use of the Logo editor.

**How To Use It**

Enter the procedures TELL, TELL1, TELL2 and PICK given below. You also need to enter a GRAMMAR procedure, some corresponding parts-of-speech procedures and a story procedure. If your story procedure is called MY.STORY, then all you have to do is type

```
TELL "MY.STORY
```

and Logo will print the text of your MY.STORY procedure replacing the parts-of-speech words with random selections from the corresponding parts-of-speech procedures.

The four central procedures of the program are listed below along with a sample GRAMMAR procedure defining six parts of speech, the six corresponding parts-of-speech procedures and a sample story procedure. With all of this entered Logo, type

```
TELL "STORY1
```

There are any number of changes you (or your students) can make in this program. Change the story around. Change the adjectives, nouns and verbs. Make your own stories. Make new parts of speech. Modify the program to handle all sorts of punctuation. Have Fun!

**How It Works**

The TELL1 procedure recursively sends each line of the story to TELL2, pausing between each line so that the user can read it. TELL2 recursively considers each word on the line, checking if it is a part of speech (i.e. listed in the grammar procedure) and replacing it by a random selection from the corresponding part-of-speech procedure if it is. TELL2 also does the check for periods. These are very typical examples of using recursion to consider each element of a list. This type of thing is done so often in Lisp, it has a special name. It is called CDRing (pronounced could-er-ing) down a list because Lisp's BUTFIRST procedure is named CDR (could-er) for reasons too boring (and historical) to go into here.

This program was written in Apple Logo II but should work on most Logo systems. You might have to change predicates like MEMBERP to MEMBER?. If there are still problems, make sure your TEXT command acts as described above.

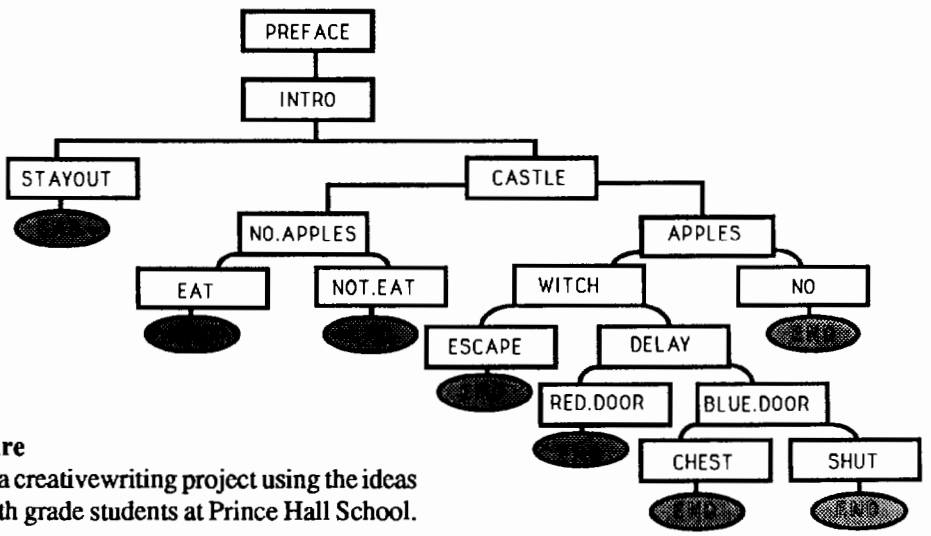
**Program Listing**

```
TO TELL :STORY
TELL1 BUTFIRST TEXT :STORY
END

TO TELL1 :LIST
IF EMPTYP :LIST [STOP]
TELL2 FIRST :LIST
WAIT 120
TELL1 BUTFIRST :LIST
END

TO TELL2 :LIST
IF EMPTYP :LIST [PRINT [] STOP]
MAKE "ITEM FIRST :LIST
MAKE "P " 
TEST = LAST :ITEM ".
IFTRUE [MAKE "ITEM BUTLAST :ITEM MAKE "P ".]
```

```
TEST MEMBERP (LIST :ITEM) TEXT "GRAMMAR
IFTRUE [TYPE PICK BUTFIRST TEXT :ITEM]
IFFALSE [TYPE :ITEM]
TYPE :P
TYPE CHAR 32
TELL2 BUTFIRST :LIST
END

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

TO GRAMMAR
NOUN
NOUNS
VERB
VERBED
ADJ
PLACE
END

TO NOUNS
FALL
COMPUTERS
ELEPHANTS
HOUSES
FEET
END

TO ADJ
HAIRY
SCARY
LOUD
WEIRD
FUNNY
END

TO NOUN
COMPUTER
CAR
STOVE
IGUANA
END

TO VERB
RUN
SLIP
DRIP
LISTEN
END

TO VERBED
WALKED
SANG
FLEW
DREAMED
END

TO PLACE
THE STORE
THE MUSEUM
CANADA
SCHOOL
PRISON
END

TO STORY1
SUMMER CAMP: ONE SUMMER MY PARENTS FORCED ME TO
GO TO CAMP. WHEN PACKING, I REMEMBERED THINGS LIKE
MY NOUN BUT I FORGOT MY ADJ NOUN. WHEN I GOT THERE
I THOUGHT IT LOOKED LIKE PLACE. OF COURSE THE
FIRST THING I DID WAS TO VERB AND UNPACK MY NOUN.
ALL OF THE PEOPLE IN MY CABIN WERE ADJ AND ONE OF
THEM EVEN HAD A ADJ NOUN. WE WENT CAMPING THE NEXT
NIGHT. FIRST WE SET UP OUR NOUNS AND THEN WE BUILT
A NOUN. AT NIGHT IT WAS SO ADJ THAT ALL I COULD
THINK ABOUT WAS GETTING BACK TO PLACE SO I COULD
VERB WITH MY NOUN. WHEN I GOT BACK HOME I TOLD
MY ADJ PARENTS ALL ABOUT CAMP. THEY WERE SHOCKED
THAT I HAD VERBED AND THAT I HAD BECOME SO ADJ AND
THEY PROMISED NEVER TO SEND ME TO PLACE AGAIN.
END
```

Michael Katz is a student at Brandeis University and has been interested in Logo and Lisp for many years. He is interested in any comments or suggestions you have (BOX 1282, Brandeis University, Waltham, MA 02254). This article has also appeared in the October 1987 issue of *Polyspiral*, a Logo newsletter of the Boston Computer Society.

**TREE DIAGRAM of THE ATTIC ADVENTURE**




**The Attic Adventure**
A procedure tree of a creativewriting project using the ideas in this article by sixth grade students at Prince Hall School.

# MathWorlds

## Lego/Logo

**by A. J. (Sandy) Dawson**

Lego is no more than a colored brick
And Logo is no more than a computer program
But if you put them together,
You need more than imagination
More than themes and creations
You need something inside you
To get you through the frustration, the
excitement, the happieness,
And the sadness
When you make something great
And you have to dismantle it.

CW

This poem, and the others contained here, were written the last day of the school term, June 1988. The authors, all in grade five from a school in suburban Vancouver, had just finished a five month exploratory study on the use of Lego/Logo. As I prepare this column, we are still in the process of analyzing the data from this pilot study even as we prepare to begin the main study in September, just a week away.

"Is the poem typical of the student reactions to the study?" you ask. Well...yes and no. First, let me say that we used three classrooms for the study: a grade four class, a grade five class, and a grade six/seven split class. The poems were written by the students in the grade five class. They were the group which went the furthest (whatever that means) in the period covered by the study. More on that a bit later.

We started in February amidst much excitement on the part of the teachers and the students involved. The teachers had come to the university for a crash day-long training session after having volunteered themselves and their students to be part of the study. They left this session enthusiastic and anxious to "get at it." Although there was a slight delay as we quickly arranged a meeting in order to explain to the parents what it was we wanted to do with Lego/Logo, the teachers began almost immediately. The paramount concern of the parents, and the teachers, was how does this fit into the curriculum. Good question. I wish we had better answers than we gave that night.

Each classroom was given two of the Lego/Logo kits and a computer. All the teachers started off with the soapbox derby activity which is described in the manuals which accompany the kits. David Bell, my co-investigator, and I had experience with this activity when we visited Hennigan School in Boston during Logo 86. Our research assistant was on hand making a video and still picture record of the first day, and David and I wondered about the rooms taking field notes like true non-participant observers. Thus begin five months of intensive observation as teachers and students attempted to come to grips with Lego/Logo.

That visit in Boston plus a brief description by Mitchel Resnick and Steve Ocko of their experiences with the children at Hennigan had motivated David and me to see if we could duplicate their results. Mitch and Steve reported that students at Hennigan, a high density computer site, displayed science and inventor like behavior, and developed in the area of cooperation, particularly cross gender cooperation. We wanted to know if children in a more typical classroom would display these behaviors as well.

We did not know, however, how to advise the teachers as how they should begin. We didn't know how much help we should give the teachers and the children. We didn't know how the Lego/Logo materials should be introduced. We didn't know how much of those materials we should have in every classroom in order to give our experiment a fair test. Hence, we decided the best thing we could do was to use the expertise and the experience of the teachers with their children as a guide to what to do, to hold weekly or at least bi-weekly meetings with them, and to devise tactics as we went along. And that's what we did.

Gradually, over the five months, our research assistant, Kevin Akins, took on more and more the role of participant observer; that is, he became a guide and advisor and Logo instructor to both children and the teachers. Of the five teachers involved only one had any experience with Logo. He was the school's designated computer expert, a fact we initially thought would greatly assist the other teachers. As it turned out, this guy was so busy that he couldn't be of much on the spot help to the other teachers, so our research assistant picked up this role. He had training in Logo, the language, though he had not taught in a Logo-like environment before.

There was an enourmous amount of data collected during our study: Kevin's field notes, two sets of interviews with all the teachers, two sets of interviews with the children, video records of both day to day events and the grande finale, and the parent presentation during mid-June We are still looking at Kevin's data and hence can't give definitive statements on the issues we were particularly interested in. However, we can make some observations which form the basis for some preliminary thoughts.

First, the grade four class did not get involved to the extent that the older children did. Indeed, their reaction to Lego/Logo has raised serious questions in our minds as to whether or not these children were just too young to engage

in Lego/Logo activities. We will be following these children in the coming year in grade five, so we should be able to gain some clarification on this issue as we observe them. In any case, the grade fours only made use of the materials for about three months, and then their interest and that of the teacher dropped off.

Second, the grade six/seven class experienced a change of teacher—maternity leave—half way through the experiment. The relieving teacher, though introduced to the materials during a special session we had for her, was not one of the original, volunteer teachers. She felt under a good deal of pressure to deal with the standard curriculum, and as a consequence the Lego/Logo experiment was not central to her teaching plans. Nonetheless, she did try and her children used the materials for upwards of four months.

It was the grade five children and their teacher who displayed the greatest degree of involvement with the project. During the final three weeks of the school term all the Lego/Logo materials were moved to this class in order to allow them to prepare a school presentation for their parents. It was the highlight of their school year, according to their teacher. During the preparation for this grande finale the children worked on their Lego/Logo projects for an hour to an hour and a half per day. Normally, all of the children got about three quarters of an hour twice a week for working with Lego/Logo. (This was also true of the other grades in the study.)

As you can tell from the poem at the beginning of this column, these grade five children found working with Lego/Logo to be both exciting and frustrating. The frustration arose not so much from not being able to build what they wanted to build, but from the fact that normally they had to destroy their creations so that other groups in the class could have their turn with the materials. Hence, it is clear that just the amount of materials we had available for the children, the shear quantity of Lego bricks and in particular the motors, gears and wheels, was a cause of concern for us and the children. By and large, the teachers subdivided their classes into working groups, and then designed a schedule to give each group a turn with the materials. This meant that projects imagined, designed and built during one session had to be destroyed so that the next group could have its turn. Not a very satisfactory situation, and one which we hope to reactify before we start again this fall.

But when that grade five class got all the materials into their room, and they were given the challenge to create whatever they wished, and it wouldn't have to be dismantled, and it could be shown to their parents at a special evening showing—well Wow!—that did cause excitement!

Imagination fills my brain
I can see it working
Bright lights flashing
Creativity is what you need
Structures of models you've made
You get better every day
By now you're proud of yourself before you've even made it
You slide in your disk
You press return
The rides begin to move
Suddenly your teacher says 5 minutes for you
You feel sad because you just got it working and you have to dismantle it
Tomorrow will be a new day

C

That big day came and the children made their presentation. Forty to fifty parents showed up that night. Five groups presented their working models. There was a logging operation, a tractor-pull, a conveyor belt, a ferris wheel and other rides. And they all worked—well, sort of.

The children did not learn a lot of Logo during the five months. They did explore with Lego an great deal, though their projects seemed to have some predictable patterns—mainly houses and cars. And they didn't use the computer much except as a hot port. And the groups were pretty well restricted to all girls or all boys, but that was gradually changing. But the children did feel like inventors and scientists.

On that final day, when all the Lego had been packed off to the university, the grade fives were asked to write one final time in their journals, and as stimulation for that their teacher shared with them some poems written by other children from around the world. So the grade fives from a suburb of Vancouver wrote their poems, and shared their thoughts about Lego/Logo.

Lego Logo is like a new creation
An extreme thought
Nothing else is in relation
It gets your mind working
Its not relaxation
You're on the edge of your seat
Frustration it is
But not enough to set you off
You need a great imagination
Great ideas you need
You made an invention
You don't want to break it
Cause it is a great creation indeed!

E

## MathWorlds -- continued

When we get the analysis done, I will write more about our Lego/Logo project. In the meantime, I must go search for that answer to that question as to how Lego/Logo fits into the curriculum. Anyone out there have any ideas?

A. J. (Sandy) Dawson is a member of the Faculty of Education at Simon Fraser University in Vancouver, Canada. His Compuserve number is 76475,1315. He can also be reached electronically through Bitnet as Daws@SFU.BITNET

# Lego®TC logo

## Some First Impressions

**by Theodore M. Norton and Howard A. Peelle**

Lego® TC logo is an exciting educational system, the product of a collaboration between Lego Systems, Inc. and Logo Computer Systems, Inc. Our first impressions here are just that: they are meant to initiate, not foreclose, an exchange of views on Lego® TC logo.

We have been able to examine the basic Lego® TC logo package, i.e., Technic Control 0. There are two others: Technic Control I and II. Additionally, while we greatly profited from the contributions of Eleanor, age 8, we have been necessarily influenced by teachers' responses to the product, and we look forward to reports based on extensive classroom testing of Lego® TC logo.[1] (See MathWorlds, above, in this issue of *LX*.)

At an very early phase of its developmental history, children used Logo to instruct a floor Turtle. Now, with Lego® TC logo, the physical Turtle returns — together with numerous relatives perhaps not envisioned by the early language designers. These include several model machines suggested by Lego Logo's creators (e.g., a washing machine, automobile, traffic light, conveyer belt), as well as those imaginary machines that LegoLogo may prompt its users to design. Now you can build the turtle as well as program it (in a specially enhanced version of LogoWriter). Accordingly, Lego® TC logo does not merely return to some abandoned phase of Logo's evolution.

Who likes Lego® TC logo? At least initially, everyone we observed. It was very warmly received by teachers at both sites; and our assistant, a third-grader (with no previous experience of Technic-level Logo), very rapidly and enthusiastically constructed several example machines from the specs provided by the manufacturer.

## Some First Impressions -- continued

Why do these people like Lego® TC logo? Primarily, it would seem, because the system may further the Logo ideal of helping to make what seems abstract palpable, manipulable, and concrete. Many teachers were excited by the new levels of complexity that the Lego component adds to Logo-based learning environments. A lot of practical programming comes from problems posed by attempts to control real rather than simulated machines. And, once the actual machines have been introduced into a learning environment, there is no theoretical limit to the kinds of construction and control projects students may wish to undertake.

Eleanor did not quite put her responses in these terms. She liked Lego® TC logo because you could (1) build a merry-go-round; (2) put little people in it; (3) have Logo make them all go round and round, to musical accompaniment. Eleanor enjoys output . And, one thing that can be affirmed prior to all evaluation is that Lego® TC logo adds a whole new output dimension to the repertoire of computer-based instructional systems. We are no longer limited to hard copy or screen displays.

We must leave it to a wider community of classroom users, however, to determine how many students will go how far, for how long, in their Lego Logo-inspired attempts to add physics, engineering, robotics, and (associated) mathematical notions to their conceptual repertoires. We raise this issue not because Lego® TC logo is crude or dull, but because having made the abstract concrete it is bound to lead us right back into that realm of higher-level abstractions and (often) quite intractable questions that are today bunched together under such rubrics as programming, problem-solving, and knowledge engineering. We wonder if there is a smooth conceptual transition from Lego to Logo: does automation prompt traditional engineers to add a twist to old techniques? Or does it displace the imagination of design into a quite different dimension?

We're not sure—but experience with programmable construction sets may throw some light on these questions. In the meantime we will not be shocked if at least some Legoites falter before the mysteries of Logo, whereas some Logoites can't begin to assemble those clever machines.

Certainly (as Eleanor observed), when the little plastic people whirl around, their feet fly upward. That kind of observation may prompt some rewarding discussions about the way the world works. More complex constructions using sensors will enrich these discussions; but we suspect that programming generates perplexities that additional fiddling with blocks will not resolve.

We are not suggesting that Lego® TC logo will fail in any cut-and-dried sense. On the contrary, at least a small

minority of determined students will profit greatly from working with it; whereas the renewable pool of first-time users (both students and teachers) will continue to be enthralled. The point of our cautions, however, may be moot: see the prices listed in the Lego® TC logo advertisement bound into the December 1987, issue of Logo Exchange. There are all too many school districts who lack the funds to refresh their Terrapin Logo V1.0, or original Apple Logo software holdings. But at these prices we doubt that many better-off districts will want to invest in even the least costly Lego® TC logo solution: they have to make choices and they have other priorities to meet. So we may never find out if Lego® TC logo was more nostalgia for the gears of our childhood, or if it truly had the force to set turning the wheels of our minds.

[1] We (one or both of us) observed teachers' reactions and responses to Lego® TC logo at the Logo Institute, Union College, 1987; and during a seminar-demonstration held recently at the Graduate School of Education, University of Massachusetts-Amherst.

Theodore M. Norton is currently pursuing studies toward a (second) doctorate in Cognitive Studies of Computers in Learning, Graduate School of Education, University of Massachusetts-Amherst, and is Visiting Associate Professor of Social Theory and Political Economy, at the University.

Howard A. Peelle is Professor of Education at the University of Massachusetts-Amherst, where he heads the graduate program in Cognitive Studies of Computers in Learning.

**About the Cover**

This month's designs were created by James MacPhee, a senior mathematics education major at Boston College. Margaret Kenny, his teacher, writes "A personal liking for optical illusions led us to consider them as a project for students to follow in a middle school or low level high school geometry class using Logo as [a] tool. Our philosophy is built around the premise that with knowledge of just a few primitive commands students can reinforce a great many geometric ideas effectively." She goes on to say that while there are more direct ways to do computer graphics, she likes Logo because the student applies geometric concepts in the process of creating the graphics.

# Assessing Logo Learning

## II. Using Variables

**by Dan Watt**

This is the third of nine columns based on a research project which Molly Watt and I have been carrying out with support from the National Science Foundation, "Exploratory Research on Critical Aspects of Logo Learning." In this project, we collaborated with a group of experienced Logo teachers to identify critical aspects of Logo learning and group them under eight headings which were listed in September's column.

This month I will talk about what we mean by each of the subheadings included in the second cluster of critical aspects, Using Variables, and illustrate these ideas with examples of student work. For a fuller sense of what we mean by critical aspects of Logo learning, and our rationale for this approach to assessing Logo learning, please read the September '88 column in this series.

**Variables — Using Names to Manipulate Information**

Understanding the use of Logo variables is a major step in harnessing the power of the computer. Logo variables provide experiences with some of the most important ideas in mathematics. For example, using Logo variables to draw shapes of different sizes provides an intuitive introduction to the idea of a mathematical function, a set of mathematical rules or operations whose result (output) is dependent on one or more inputs. And the use of Logo variables with mathematical operations, paves the way for understanding the algebraic concept of a unknown. The notion that we can give a name to a piece of information, and then manipulate that information symbolically, without knowing exactly what it is, substituting a specific value later, is one of the prerequisites for understanding almost any branch of mathematics.

At the same time, using variables unlocks the power of the computer to carry out algorithms: repetitive tasks in which the same operations are applied to information that changes as the task repeats. This is one of the core ideas underlying all computer science. Another important computer science and problem-solving concept that is supported by using Logo variables is the idea of modularity: the use of one procedure or tool for more than one purpose.

A Logo variable has one word of any length as a name, and can have a number, word or list as its value. Logo variables come in two flavors, local variables, which are defined only within a particular procedure and all its subprocedures, and global variables which can be used within any procedure, or at direct command level. Elementary Logo programming is commonly taught as a way of causing the turtle to draw shapes on the screen. Local variables are

## Assessing Logo Learning in Classrooms --- continued

introduced as a way to generalize a procedure, that is, to give one procedure the capability of drawing many different shapes. Therefore, most of the grade 4 - 6 students whose teachers collaborated in our research, used variables only to represent numbers and used only local variables, defined within particular procedures.

Logo variables can be given arbitrary names just like Logo procedures. As with procedures, students may need encouragement from a teacher to move from idiosyncratic, or ritualistic naming strategies, to use meaningful names that can communicate the purpose of each variable to a reader of their procedures.

Many students in grades 4 through 6 work with Logo without ever using variables. Some teachers may be reluctant or feel unprepared to teach their students to use variables. However, we found a number of students using variables on their own. They may have learned to use variables from other Logo users outside of class, from books, or by analyzing tool procedures. We have identified several different levels of sophistication involved in using ideas about variables, but we want to emphasize that students do not necessarily learn to use variables in this order.

**A. Pre-variables: using the idea of variables, without using inputs or variable names.**

For example, making separate procedures for squares of different sizes, in which the procedures are identical in every respect except the input to FORWARD. Or, using REPEAT with different angle inputs to create different shapes. Andy's and Sam's work (Examples 1 and 2), offer clear examples of students who are using pre-variables without using Logo variables directly.

**B. Procedures with simple inputs**

The simplest procedures with variables are ones that produce fixed shapes of different sizes. These procedures are usually constructed by starting with fixed-size procedures, and substituting a variable :SIZE in place of fixed sizes. For example,

```
TO SQUARE :SIZE
REPEAT 4 [FORWARD :SIZE RIGHT 90]
END
```

Such procedures are often given to students as models, or as tools to use in creating more elaborate designs than they could make with fixed-size figures only.

**C. Procedures with multiple inputs**

This type of procedure may include shapes for which more than one size is varied (such as a rectangle):

```
TO RECT :SIZE1 :SIZE2
REPEAT 2 [FORWARD :SIZE1 RIGHT 90
   FORWARD :SIZE2 RIGHT 90]
END
```

or shapes for which both size and angle, or for which the size and number of sides are varied (such as a general procedure to draw a regular polygon):

```
TO POLYGON :N :SIZE
REPEAT :N [FORWARD :SIZE
   RIGHT 360/:N]
END
```

**D. Using inputs within subprocedures**

These procedures can take on a variety of forms: complex drawings in which all elements vary together (using TRI :SIZE and SQ :SIZE to build HOUSE :SIZE); procedures which use different sizes of the same shape (varying squares to produce a tower or a tunnel). Students typically proceed from using variable procedures with fixed inputs, as in Sam's and Heather's projects (Examples 2 and 5), to using variable names within subprocedures (See Tony's, Aaron's and Diane's projects, Examples 3, 4 and 8). As these procedures become more complex, it becomes important to understand how local variables work in Logo.

**E. Procedures with proportional variables**

These are procedures with variables that are proportional to the original values used in a fixed-value procedure. For a simple example, suppose we have a fixed rectangle procedure,

```
TO RECT
REPEAT 2 [FORWARD 20 RIGHT 90
   FORWARD 60 RIGHT 90]
END
```

We might generalize this proportionally as follows,

```
TO RECT :SIZE
REPEAT 2 [FORWARD :SIZE RIGHT 90
   FORWARD 3 * :SIZE RIGHT 90]
END
```

or, alternatively, we might add a scale factor:

```
TO RECT :SCALE
REPEAT 2 [FORWARD 20 * :SCALE RIGHT
   90 FORWARD 60 * :SCALE RIGHT 90]
END
```

This type of procedure may help students learn to understand ratios, proportion and similarity as well as demonstrating how those mathematical ideas are used for practical purposes. Diane's project (Example 8), illustrates this.

**F. Procedures that involve recursion, incrementing variables and conditionals.**

These procedures are used when a variable changes in a repeated process. For example,

```
TO COUNTDOWN :NUMBER
IF :NUMBER = 0 STOP
PRINT :NUMBER
COUNDOWN :NUMBER - 1
END
```

Many students get into this type of situation with recursive designs that grow indefinitely. A conditional statement or stop rule is needed to terminate the growth. Tony's project

(Example 3) illustrates a common variation of this. Jerry's project (Example 7) shows an unusually creative use of this programming technique.

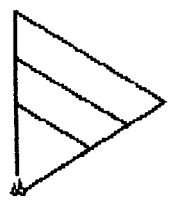
**G. Using global variables (MAKE); Understanding the relation between local and global variables.**

Global variables were used by only a few students in our research project. Most of these students were working independently on advanced projects. A discussion of how students develop their understanding of global variables would take an entire article. Therefore, I won't say any more about it here.

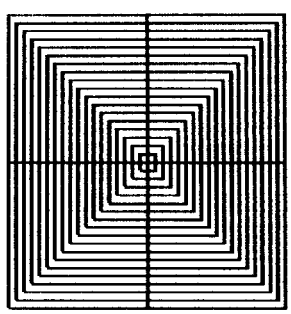
**Examples of Student Work With Variables**

(1) Andy's exploratory work (Grade 5) with his SPIRALS and TRIANGLES procedures show that he is using the idea of variables without actually knowing the Logo syntax for writing procedures with inputs. This is a good example of what we mean by pre-variables.

```
TO TRIANGLES
 REPEAT 3 [FD 50 RT 120]
 REPEAT 3 [FD 75 RT 120]
 REPEAT 3 [FD 100 RT 120]
END

TO SPIRALS
 REPEAT 36 [FD 1 RT 10]
 REPEAT 36 [FD 2 RT 10]
 REPEAT 36 [FD 3 RT 10]
 REPEAT 36 [FD 4 RT 10]
 REPEAT 36 [FD 5 RT 10]
 REPEAT 36 [FD 6 RT 10]
END
```



(2) Sam's DESIGN (Grade 6) also illustrates the idea of pre-variable thinking. Sam is using a variable square procedure as a subprocedure of DESIGN, but he is using it very much the way Andy used FORWARD in example one.

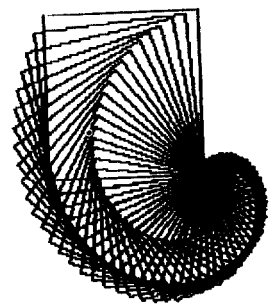


```
TO DESIGN
 SQUARE 90
 SQUARE 85
 SQUARE 80
 SQUARE 75
 SQUARE 70
 SQUARE 65
 SQUARE 60
 SQUARE 55
 SQUARE 50
 SQUARE 45
 SQUARE 40
 SQUARE 35
 SQUARE 30
 SQUARE 25
 SQUARE 20
 SQUARE 15
 SQUARE 10
 SQUARE 5
END
```

```
TO SUPERDESIGN
 DESIGN
 LT 90
 DESIGN
 LT 90
 DESIGN
 LT 90
 DESIGN
END
```

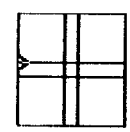
```
TO SQUARE :LENGTH
 REPEAT 4 [FD :LENGTH RT 90]
END
```

(3) Tony's WENDSDAY design (Grade 6) takes this approach to its logical conclusion. He's using the square procedure as a variable subprocedure within a recursive procedure which is programmed to stop when the size variable grows larger than a given amount.



```
TO WENDSDAY :LENGTH
 HT
 FULLSCREEN
 IF :LENGTH > 110 STOP
 SQUARE :LENGTH
 RT 5
 WENDSDAY :LENGTH + 2
END

TO SQUARE :LENGTH
 REPEAT 4 [FD :LENGTH RT 90]
END
```

(4) Aaron's project (grade 5) shows an incomplete understanding of the use of subprocedures with variables, and of procedures with more than one variable. His teacher taught him a convention for naming procedures with variable inputs: add ".ANY" to the name of the procedure to show that it can make a shape of any size. But Aaron's superprocedure, PRESENT, combines fixed FORWARD steps and subprocedures with variable inputs in the same procedure. It only draws the design Aaron wanted when given the inputs 80 10. Aaron has not taken proportionality into account. His naming also shows that he has not clearly distinguished the different purposes of the two different variables he is using.



```
TO PRESENT :SIZE :SIDE
 SQ.ANY :SIZE
 RT 90
 FD 35
 LT 90
 REC.ANY :SIZE :SIDE
 LT 90
 FD 35
 RT 90
 FD 45
 RT 90
 REC.ANY :SIZE :SIDE
END
```

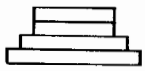
```
TO SQ.ANY :SIZE
 REPEAT 4 [FD :SIZE RT 90]
END
```

```
TO REC.ANY :SIZE :SIDE
 REPEAT 2 [FD :SIZE RT 90 FD :SIDE RT 90]
END
```

(5) Heather's Statue of Liberty project (Grade 6) used a variable rectangle procedure VRECT, as part of the PLATFORM on which the Statue was to stand. Heather correctly used VRECT as a tool, with fixed inputs, to create the drawing she wanted. Heather's variable names, :W (width) and :L (length) indicate a better understanding of what her rectangle procedure and its variables are supposed to do than Aaron's variable names, :SIZE and :SIDE.

## Assessing Logo Learning in Classrooms -- continued

```
TO PLATFORM
 HT PU HOME BK 119 RT 90 FD 50 LT 180 PD
 VRECT 100 10
 PU FD 10 RT 90 FD 10 LT 90 PD
 VRECT 80 10
 RT 90 FD 10 LT 90 FD 10 PD VRECT 60 20
 VRECT 60 10
END

TO VRECT :W :L
 REPEAT 2 [FD :W RT 90 FD :L RT 90]
END
```



(6) Jerry's work (Grade 4), illustrates a more sophisticated use of procedures with more than one variable. He started out by making a Logo circle:

```
REPEAT 360 [FD 1 RT 1]
```

He generalized this to a procedure with two variables to control the size of the forward step and the amount of turn:

```
TO CIRCLE :SIZE :TURN
REPEAT 360 [FD :SIZE RT :TURN]
END
```

Then he realized that with turns larger than RIGHT 1, he did not need to repeat his instruction 360 times, so he created another procedure with a variable to control the number of repeats:

```
TO S.M. :TIMES :SIZE :ANGLE
REPEAT :TIMES [FD :SIZE RT :ANGLE]
END
```

Jerry explained that he knew how to find the angle for a particular shape by dividing 360 by the number of sides. (But he had not yet figured out that he could incorporate this directly in his procedure.)

(7) Jerry's work with variables also shows that he has a working knowledge of recursion and conditionals. His teacher gave him a tool procedure to make the computer "waste time:"

```
TO WAIT :TIME
IF :TIME = 0 STOP
WAIT :TIME - 1
END
```

Jerry modified this on his own in the following way:

```
TO DECIDE :NUMBER
IF :NUMBER = 1 PR [GOOD!]
IF :NUMBER = 2 PR [AWESOME!]
DECIDE :NUMBER - 1
END
```

He noticed that with an input greater than 1, this procedure would print both GOOD! and AWESOME!, but would never stop. He then modified it:

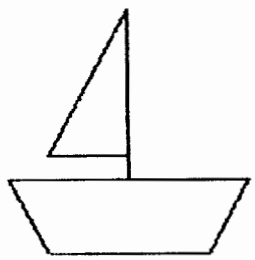
```
TO DECIDE :NUMBER
IF :NUMBER = 1 PR [GOOD!] STOP
IF :NUMBER = 2 PR [AWESOME!] STOP
DECIDE :NUMBER - 2
END
```

Jerry then explained that DECIDE would print GOOD! if it started with an odd number and AWESOME! if it started with an even number:

```
DECIDE 19
GOOD!
DECIDE 24
AWESOME!
```

(8) Diane's SAILBOAT (Grade 6) uses variables to create a shape that is proportional to the fixed-size design she started with. This project was suggested by her teacher to give students experience with variables and proportionality, topics that were part of the sixth-grade math curriculum:

```
TO SAILBOAT :L
 HT
 HULL :L
 MOVE :L
 SAIL :L
END

TO HULL :L
 LT 30
 FD :L
 RT 120
 FD :L * 3
 RT 120
 FD :L
 RT 60
 FD :L * 2
END

TO MOVE :L
 PU
 RT 90
 FD :L * 0.87
 RT 90
 FD :L
 PD
 LT 90
END
```



```
TO SAIL :L
 FD :L * 2
 LT 150
 FD :L * 2
 LT 120
 FD :L
END
```

## Assessing Logo Learning -- continued

**Some Closing Thoughts**

Most students in grades 4 - 6 who use variables at all, have a very incomplete understanding of the topic. In some cases they pick up ideas about variables outside of class, from other students, from books, or by examining tool procedures. Some fourth grade students (for example, Jerry) are capable of using variables quite flexibly, although their teachers do not teach variables in any formal way, and may have incomplete knowledge about Logo variables themselves.

All of this makes variables a tricky topic to handle at these grade levels. There are many ways to approach it. An ideal Logo teacher would have a working mastery of many different ways of using variables in Logo, and should be able to help each student learn to use variables in a way that best supports what that student is trying to accomplish. A more practical approach for many teachers would be to have some sample variable procedures handy, and give them to students who are ready for them, to experiment with and use as models for their own work. This is what Jerry's teacher did with the WAIT procedure. A third approach is to create specific lessons or activities designed to teach the use of variables in a particular way. This is what Aaron's, Heather's and Diane's teachers did. Whichever approach works best for you is the one to use. But be sure to look carefully at what your students are actually doing with variables, so that you can support them in consolidating their incomplete knowledge and using it effectively to carry out specific projects.

**References**

Abelson, H. (1982). *Logo for the Apple II, Apple Logo*, Byte Books/ McGraw-Hill, New York NY (Sections 2.1 and 5.5 deal with local and global variables.)

Clements, D. (1988). "Research on Variables, Algebra and Logo," *The Logo Exchange*, January, February, March and April, 1988 (A four-part series, which offers an excellent review of Logo research on variables and how students learn — and don't learn — to use them.)

Hillel, J. and Samurcay, R. (1985). "The Definition and Use of General Procedures by 9 Year Olds," *Research Report #2*, Department of Mathematics, Concordia University, Montreal, Quebec.

The word described here was conducted at Education Development Center (EDC), 55 Chapel Street, Newton Massachusetts, and supported in part by the National Science foundation under grant #MDR 8651600, Exploratory Research on Critical Aspects of Logo Learning. The ideas and opinions expressed are those of the author and do not necessarily reflect the views of EDC or of the National Science Fouocation.

Daniel H. Watt, Educational Alternatives
Gregg Lake Road, Antrim, NH 03440

# Search and Research

## Problem-Solving Processes: The Mental Company

**by Douglas H. Clements**

In the previous column we defined problem solving as "knowing what to do when you (temporarily) don't know what to do." We also saw that in Sternberg's theory, the mind is viewed as a kind of company in which a group of problem-solving processes (components) work with each other. The metacomponents—the "executives" of this mental company—plan and evaluate all one's information processing. Therefore, they are critical in the activity of problem solving.

**The First Metacomponent:**
**Deciding of the Nature of the Problem**

What exactly does it mean when you "understand the problem?" You must connect features of the problem with what you already know. That is, a relevant mental structure, or *scheme*, must be found. Understanding the notion of scheme is important enough to justify a side trip (adapted from Clements, in press). To begin our first excursion, read this passage:

> On Saturday, John told his date he was hungry and walked into the first building on the left. They sat for a while, then John left money on the table, paid the cashier, and left. They barely made it in time for the first feature.

What's interesting about this? Not the story itself (clearly), but what you can do after you read the story. You can, for instance, answer several questions: Did John and his date go to a restaurant? Did John order food? Why did he leave money on the table? Did he go to a movie that night?

What's so interesting about that? *None* of the answers are explicitly provided in the passage. Why do most people like yourself know the answers? If asked about the story an hour later, why would most believe that the passage had *said* that John went to a restaurant? On the slightly more absurd side, how do they know that John's "date" was not a dried fruit? (Don't be mean!)

What people do is use mental *schemes* in understanding the story and answering the questions. A scheme is a large mental knowledge structure, or framework, which includes a considerable body of information. Most of us have detailed schemes for "dates" and "restaurants." We know that a Saturday night date is a person, and that two people on a date often eat and go to movie "features." We know that when people are hungry, they often go into restaurants to buy food. At the restaurant they usually order food, the food is brought by a waiter, they leave a tip on the table in addition to paying

## Search and Research -- continued

for their food, and so on. These schemes provide us with the hooks upon which we hang what we read, hear, or see.

Schemes even ask us mental questions. For instance, they tell us to look for certain information: Where did they go on the date? What was the date's name? What did they do afterward? At what kind of restaurant did they eat? More than this, they often provide the answers to these questions if the story doesn't: They went to a restaurant first, then a movie. Yes, they ate. It's not certain what kind of restaurant they ate at, but John left a tip, so it probably wasn't a fast–food restaurant.

Let's try one more experiment with schemes. Read the following passage, trying to understand and remember it.

> With hocked gems financing him, our hero bravely defied all scornful laughter that tried to prevent his scheme. Your eyes deceive, he had said, an egg not a table correctly typifies this unexplored planet. Now three sturdy sisters sought proof forging along sometimes through calm vastness yet more often over turbulent peaks and valleys. Days became weeks as many doubters spread fearful rumors about the edge. At last from nowhere welcome winged creatures appeared signifying momentous success. (adapted from Shuell and Lee, 1976)

Test yourself. Try to retell the passage. Done? With two words I'm going to help you activate a relevant scheme in your mind that provides a context and connections for the story, and that automatically answers questions (i.e., fills in the gaps). Please read the two words (at this end of the column), then re–read the passage, and try to retell it again.

How much of the passage could you remember after reading the two words? How would you compare your first and second experiences with the passage in terms of "meaningfulness"? In terms of your feelings about yourself? If you're like most people, during the second reading you remembered more words, could "make sense" of the story, and felt comfortable and in control, rather than "lost."

When we teach children to solve problems by spoon-feeding them "correct" procedures, we deprive them of such schemes. We force them to memorize numerous unconnected bits of information. We lead them to believe that the subject matter doesn't make sense, and that their job is to passively receive and later recall rather than to understand. We undermine their self–confidence and motivation to learn by creating feelings of confusion and helplessness. When students solve problems based on schemes, the story is different. Consider Robert Davis' account of Alex, a five–year–old girl whose brother, Paul, was age three.

> Alex: When Paul is six, I'll be eight; when Paul is nine, I'll be eleven; when Paul is twelve, I'll be fourteen [she continues until Paul is 18 and she is 20].
>
> Interviewer: My word! How on earth did you figure all that out?
>
> Alex: It's easy. You just go "three-FOUR-five" [saying the "four" very loudly, and clapping hands at the same time, so that the result was very strongly rhythmical, and had a soft-LOUD-soft pattern], you go "six-SEVEN [clap]-eight," you go "nine-TEN [clap!]-eleven".... (Davis, 1984, p. 154)

Alex had learned to count at home and school. But she had never learned how to add, and certainly hadn't learned addition facts such as 12 + 2 = 14. How had she accomplished such remarkable mathematical thinking? Well, she *had e*xperienced addition in her life, through gathering stones and blocks—she had built her own intuitive addition scheme. She also had schemes for counting and for rhythmic patterns (from games, rhymes, and songs). What she did is put these schemes together in a way she had never done before—*and in a way no one had ever taught her to do*—to solve a mathematical problem. This is the way new schemes are built. This illustrates, too, that schemes are built through the creative problem solving of children—Piaget's "re–invention of knowledge."

Therefore, helping children learn to build and use such schemes is basic to quality instruction in problem solving. Students need to analyze relationships between parts of the problem or search for patterns among these relationships to build up a useful mental representation of the problem. Then this partially formed representation must be compared to existing schemes, because the student must locate a scheme relevant to solving the problem (e.g., "That's just like the 'How many outfits can you make from 7 shirts and 4 skirts' problem"). Once found, such a scheme contains information about the typical problem goal and helpful solution strategies. If the scheme is is not helpful, the process is repeated, as often as necessary.

**Deciding on a Strategy and on a Representation: Extending Understanding of the Problem**

Virtually at the same time that you are searching for relevant schemes, you use two additional metacomponents, selecting a mental representation and selecting a solution strategy for the problem. If the scheme contains specific information, this is often easy—the solution is already known (e.g., you "just know" how to represent and chose a strategy to solve "One toy costs $4; how much does a carton of 12 cost?"). If this information is not available, a mental

search begins. Usually unconsciously, you search for answers to questions such as: Is this problem like other problems about whose solution I know more? Might I plan a solution by breaking it down into parts, or subproblems, that I can solve in order? What representations have I used successfully (e.g., drawings or tables) that might help with this problem as well? Could I gather more information somehow?

**Good Strategy Users**

What you are searching for is a useful *strategy*. Good problem solvers know numerous strategies (see Pressley, 1986). Strategies include routine procedures such as counting or arithmetic operations (addition or division) and problem–solving procedures such as drawing a diagram. These latter processes are often called heuristics, which means "serving to guide, discover, or reveal," because they may (but may not!) help in solving a problem.

Thus, you should help students learn useful strategies. For example, in beginning arithmetic, strategies such as counting forward and back, and doubles + 1 are useful. This is a goal–specific strategy, applicable to one type of problem. Other strategies, such as the heuristic of pattern searching, are more general. They can be used in many situations and can be tried when problem solving becomes "what we do when we don't know what to do." With this generality comes the disadvantage that it may be more difficult to ascertain which one to choose and how exactly to use it to get an answer. In addition to drawing a diagram or picture and looking for a pattern, such heuristics include acting it out, making an organized list or table, systematic guessing and testing, breaking a problem into parts, writing a number sentence, and working backwards.

Good strategy users also know when, why, and how to use each different strategy. They also know how to modify a strategy to fit a certain situation and how to combine strategies. For example, they might recognize that a problem involving the perimeter of a rectangle could be efficiently solved with a formula. A slightly more complex problem, however, (e.g., the perimeter of a 4 meter walkway boardering a 20 by 10 m pool) might best be first approached by drawing and labeling a diagram as an aid to visualization (i.e., building a representation) as well as an aid to problem definition. This type of knowledge is used by the metacomponent of selecting a strategy. To help students gain this knowledge, you should provide practice in strategy use in a variety of situations, always striving to help students become explicitly aware of where, when, and how to apply the strategies.

Finally, good strategy users possess knowledge about the domain or topic at hand. Sometimes, this obviates the need to use a strategy at all—as we said previously, the answer, or an algorithm that yields an answer—is "just known." (This is fine from the perspective of solving that problem, as it effectively reduces the problem to an application. Of course, you should make sure that both applications and problems are presented to all students.) In other cases, knowledge allows the use of many strategies. For instance, general problem–solving strategies (e.g., break a problem into parts), can be used only when the student has a lot of information that can be plugged into the problem (e.g., what are meaningful "parts" of a certain problem?).

**Solution Monitoring: A Neglected Metacomponent**

Students often do not monitor their own problem solving. They believe ideas just "come to them" or not, without any willfulness on their part. Once they start working on an idea, they rarely pause to see if it will help them solve the problem, or if it makes any sense at all. They do not generally examine their ideas and work for errors. They believe that they fail because they are "stupid," and that errors are absolute; that little can be learned from them. There are ways—with and without computers—to help them develop these abilities, as well as the other metacomponents. Specific methods that research has identified as particularly successful will be discused in the next column. Until then, monitor yourself: The next time you solve a problem, stop yourself and try to identify what metacomponents you were using.

**References**

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

Davis, R. B. (1984). *Learning mathematics: The cognitive science approach to mathematics education.* Norwood, NJ: Ablex.

Pressley, M. (1986). The relevance of the good strategy user model to the teaching of mathematics. *Educational Psychologist, 21*, 139-161.Shuell, T. J., & Lee, C. Z. (1976). *Learning and instruction.* Monterey, CA: Brooks/Cole.

Sternberg, R. (1985). *Beyond IQ.* Cambridge: Cambridge University Press.

(The two words are "Christopher Columbus")

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

# Global News

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



It has been some time since we heard from the "wonder down under," but Australia continues to expand the use of Logo in the schools in some very unique ways as Jeff Richardson reports.

### Logo Freak Wanted

An advertisement like this would have to grab the attention of the readers of the *Logo Exchange*. And just such an advertisement appeared in a Melbourne newspaper late last year. Don't rush to apply. As you might have guessed the position was quickly filled. It's one of many Logo curriculum projects that are currently underway in Australia.

The Logo freak appointed is Sandra Wills, one of the world's original Logo pioneers. Sandra will be well known to many readers of the *Logo Exchange* in North America as well as Australia.

Sandra is at Mowbray College, a large school at Melton, Victoria. The school is well equipped with IBM-JX machines (two in every classroom and two class sized laboratories). More importantly the school has a full site license for LogoWriter, and a principal who is committed to fully exploiting this potential across the curriculum and from K-12. Hence his desire to appoint a Logo freak to facilitate the curriculum innovation process.

Sandra remarked to me recently that experience in the classroom use of Logo takes a very definite second place in her work at Mowbray. First comes the skill and patience required for substantive curriculum innovation. This is people centered, not machine centered work. The changes to any existing curriculum implied by the Logo philosophy are massive. Sandra is learning and relearning on the job, along with her colleagues at Mowbray College and of course, the children.

Anyone interested in further information should contact Sandra Wills, Melbourne College of Advanced Education, Swanston Street, Carlton 3053, Australia.

### POALL.

A familiar world to logophiles, its also the name of Australia's Logo journal. Published quarterly, POALL is stimulating, surprising, amusing and very wide ranging. Subscriptions are $10 Australian currency (airmail is extra) and back issues are available. Any *Logo Exchange* readers interested in POALL should write to the editor, Peter Carter, at Plympton High School, Errington Street, Plympton 5038, Australia.

Peter is also the author of a very challenging little book called *Thinking Logo: An Introduction to [the Universe of through] Programming*. If you enjoyed John Allen, Ruth Davis and John Johnson's *Thinking about [TLC] Logo*, if you balk at the cost of all three volumes of Brian Harvey's *Computer Science Logo Style*, and if you are still getting around to tackling *Turtle Geometry* by Hal Abelson and Andrea di Sessa, then this might just be the book for you. Send all enquiries directly to Peter at the same address above.

### Sunrise School

A computer curriculum project called The Sunrise School was formally opened in Melbourne in June. Housed amidst the vast collection of the Museum of Victoria, The Sunrise School is a joint project involving the staff and students of Princes Hill Secondary College, the Museum itself, and the Australian Council for Educational Research. The Sunrise School aims to combine new electronic technologies with traditional human and information resources.

Liddy Nevile, the director of the project, wants the school to empower the students as independent learners and creators, using electronic artefacts to give students access to the resources in their environment. A parallel purpose for the Sunrise School is to provide an observatory for new educational practices to be interpreted and evaluated ahead of institutionalization. The school is also intended to function as an exploratory research forum, developing and dis-

seminating a body of technological and pedagogical criticism. Liddy is a long term member of the Logo community and while The Sunrise School is not purely a Logo project, it is based on the emancipatory and progressive ideas that are familiar in the Logo culture.

The Sunrise School has gotten underway with the students and teachers working intensively with Lego/Logo. Expect a progress report in this column soon, but if you want to contact the project directly, write to Liddy Nevile, ACER, P.O. Box 210, Hawthorn 3122, Australia.

## Europe

Our European editor, Harry Pinxteren has sent a list of features of the new version of the LCN Logo incorporating some features of SCHEME. Experienced *Logo Exchange* readers will drool over this power implementation of Logo being used through the Dutch schools as well as elsewhere in Europe. Features include:

- Lambda calculus, lexical scope and functional arguments, anonymous functions, higher order functions such as combinators, currying, streams, continuations and function generators.
- Expression oriented rather than statement oriented syntax.
- One syntax rule, no exceptions; prefix notation.
- Both Logo and LISP syntax, expressions with and without parentheses, are available to the user along with a special "scheme-mode."
- Structure Editor (SED) which displays syntax visually; programs are formatted so that elements at the same level are aligned vertically.
- Expressions are displayed as colored planes.
- This display follows the principle of WYSIWYG and direct manipulation.
- Expressions (planes) can be run like options in a menu.
- Planes can be manipulated and edited following the cut and paste metaphor.
- Planes can be printed out as hard copy, thus extending the metaphor.
- Cutting planes supports the principle of hiding information from the user, literally.
- Special editor to edit list structures and colored planes.
- Free abbreviations of primitives with automatic completion of names.
- Inbuilt stepper and debug tools.
- Menu controlled search and change facilities.
- Horizontal and vertical scroll.
- Combination of edit facilities and graphics screen, thus promoting a control panel model.
- Integration of text and expressions amounts to a use of the editor as an electronic work sheet.
- Multiple turtles.
- 3D graphics including option for different modes of projection..
- Logo interface to save data to ASCII files, e.g., to control plotter, the floor turtle and other devices.
- Extra drawing facilities, including turtle shapes.
- Editor for graphics character table.
- Extra screen primitives to define views and windows.
- Integration of text and graphics
- Easy to program textwindows and pull down menu's.
- Function names can be used as option names in menus.
- Menu controlled I/O, including MS-DOS commands.
- Formattable Floats.
- Object oriented I/O
- File chaining.
  Run-time workspace management integrated with list processing.
- Programs and modules can be hidden from the user using protection option.
- Programs can be saved and loaded as textfiles.
- User definable status and error messages.
- Redefinable primitives.
- Use of EDLIN to write additional help menus.
- Screendump of graphics and textscreen.
- Tile-editor to redefine ASCII characters.
- Run-time interface to all MS-DOS commands.
- Assembly and machine language interface.
- Interface to other programming languages, e.g., PASCAL, C and dBASEIII.

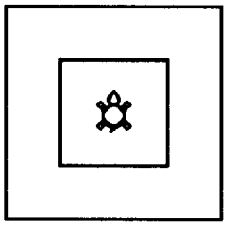
The LCN Logo also claims to have a lower threshold and by allowing for the design of several language layers, takes the no ceiling philosophy to new heights. For a more detailed explanation of the above and other features please write Harry Pinxteren who will be happy to send more information (all in English, of course)

# ICPSC Sample Problems: Elementary Logo Division

**by Don Piele and Sharon Yoder**

**Problem #1**

Draw two squares as shown in this drawing. Be sure that the inner square is one-half as big as the outer square. Put the turtle (whatever shape your turtle is) in the center of the inner square.



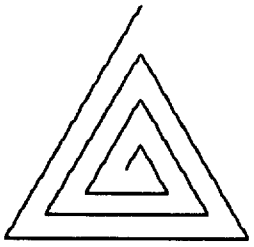
**Solution:**

```
TO BOTH.SQUARES
SET.UP
SQUARE 80
MOVE
SQUARE 40
ST
END

TO SET.UP
CG
HT
END
```

```
TO SQUARE :SIDE
REPEAT 4 [FORWARD :SIDE
    RIGHT 90]
END

TO MOVE
PU
RIGHT 90
FORWARD 20
LEFT 90
FORWARD 20
PD
END
```

**Problem #3**

Write a program to draw a triangular pattern as shown below. Be sure that your diagram is oriented exactly the way this one is. Be sure that you have exactly the same number of lines.



**Solution:**

```
TO GROW
SET.UP
GROW.TRIANGLE 10
END

TO SET.UP
CG
HT
RIGHT 30
END
```

```
TO GROW.TRIANGLE :INPUT
IF :INPUT > 100 [STOP]
FORWARD :INPUT
RIGHT 120
GROW.TRIANGLE :INPUT + 10
END
```

**Problem #4**

Write a program that will pick a name at RANDOM from a list of names. Your program should ask for a list of names and out THE WINNER IS one of the names picked at random. HINT: Use the following to pick out a name at random from a list.

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

Sample run:

```
GET.NAMES
TYPE SEVERAL NAMES
JOHN JANE JOE MARY MIKE
THE WINNER IS
JOE
```

**Solution:**

```
TO PICK.WINNER
PRINT [TYPE SEVERAL NAMES]
( PRINT [THE WINNER IS] PICK READLIST )
END
```

**Problem #5**

Write a program that reverses the words in a sentence. Test the program with the sentence: WHAT YOU SEE IS WHAT YOU GET

Sample run:

```
REVERSE.WORDS
TYPE A SENTENCE:
WHAT YOU SEE IS WHAT YOU GET
REVERSED IT READS:
GET YOU WHAT IS SEE YOU WHAT
```

**Solution:**

```
TO REVERSE.LIST1
PRINT [ENTER A SENTENCE]
MAKE "THE.SENTENCE READLIST
MAKE "BACKWARD []
REVERSE1
PRINT :BACKWARD
END

TO REVERSE1
IF EMPTYP :THE.SENTENCE [STOP]
MAKE "BACKWARD SENTENCE :BACKWARD
    LAST :THE.SENTENCE
MAKE "THE.SENTENCE BUTLAST :THE.SENTENCE
REVERSE1
END
```

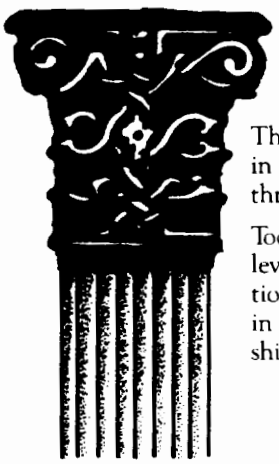
or

```
TO REVERSE.LIST2
PRINT [ENTER A SENTENCE]
PRINT REVERSE2 READLIST []
END

TO REVERSE2 :SE.IN :SE.OUT
IF EMPTYP :SE.IN [OUTPUT :SE.OUT]
OUTPUT REVERSE2 BUTLAST :SE.IN LPUT
    LAST :SE.IN :SE.OUT
END
```

See the October 1988 *LX* to learn about last year's elementary International Computer Problem Solving Contest winners. For moreinformation about ICPSC, contact

Don Piele, ICPSC
P. O. Box 085664
Racine, WI 53408

# ICCE



## About ICCE

The International Council for Computers in Education was founded by Dr. David Moursund in 1979 as an organization that would foster appropriate instructional use of computers throughout the world.

Today ICCE is the largest professional organization for computer educators at the precollege level. It is nonprofit, supported by 14,000 individual members and more than 50 organizations of computer-using educators worldwide. These organizations are statewide or regionwide in scope, averaging 500 members each. Approximately 84% of ICCE's individual membership is in the United States, 12% is in Canada, and the remainder is scattered around the globe.

## About *The Computing Teacher*

ICCE publishes *The Computing Teacher* journal. *The Computing Teacher* provides accurate, responsible, and innovative information for educators, administrators, computer coordinators, and teacher educators. The journal addresses both beginning and advanced computer educators through feature articles, columns, software reviews, and new product and conference listings. Contributors to *The Computing Teacher* are leaders in their fields, consistently providing the latest information in computer education and applications.

## Publications, Special Interest Groups

In addition to *The Computing Teacher*, ICCE provides a number of publications to computer-using educators. ICCE's Special Interest Groups provide in-depth information for computer coordinators, teacher educators, computer science educators, and Logo-using educators. *C.A.L.L. Digest* is published nine times per year for ESL teachers. ICCE committees address a variety of ethical and practical issues important to the computer-educating community.

## Independent Study Courses

ICCE offers graduate-level independent study courses, designed to provide staff development and leadership. These courses have been approved by the College of Education at the University of Oregon and carry graduate credit from the Oregon State System of Higher Education. Participants correspond with instructors by mail.

**Write for information and a free catalog today!**

**ICCE • University of Oregon • 1787 Agate St. • Eugene, OR 97403 • Ph: 503/686-4414**

# How to increase your Logo Power

Whether you're a Logo teacher, trainer, or enthusiast, you know that this powerful computer language has the potential to have a significant impact on how teachers teach and how students learn. ICCE's Special Interest Group for Logo Educators (SIGLogo) offers you a forum for the exchange of ideas, concepts and techniques.

**What is SIGLogo?** SIGLogo is a professional organization that helps Logo Educators get ahead. We sponsor meetings, conferences and workshops, providing a support community for Logo-using educators. Novice or expert, you will find helpful information in each issue of our journal, *Logo Exchange.*

**Satisfaction Guaranteed.** Whether you teach Logo or use Logo to teach, SIGLogo and *Logo Exchange* bring you a wealth of ideas from top Logo educators throughout the world, providing you with current information on Logo research, resources and methods. We're your personal window for professional Logo activities.

**Join SIGLogo Today!** As a member of SIGLogo, you will receive the *Logo Exchange* journal nine times per year. SIGLogo members are invited to participate in local, regional and national meetings and to contribute to the flow of ideas through *Logo Exchange. Logo Exchange* is published monthly except for June, July and August. SIGLogo membership is $24.95 for ICCE members, $29.95 for non-members, add additional $5 for non-U.S. SIGLogo membership.

The International Council for Computers in Education (ICCE) is the leading U.S. and international professional organization for computer educators. It is non-profit, supported by more than 50 organizations of computer-using educators worldwide.

INTERNATIONAL COUNCIL FOR COMPUTERS IN EDUCATION
ICCE

**Send For A Free Catalog Today!**

**ICCE/SIGLogo, University of Oregon, 1787 Agate St.**
**Eugene, OR 97403-9905. Ph: 503/686-4414.**