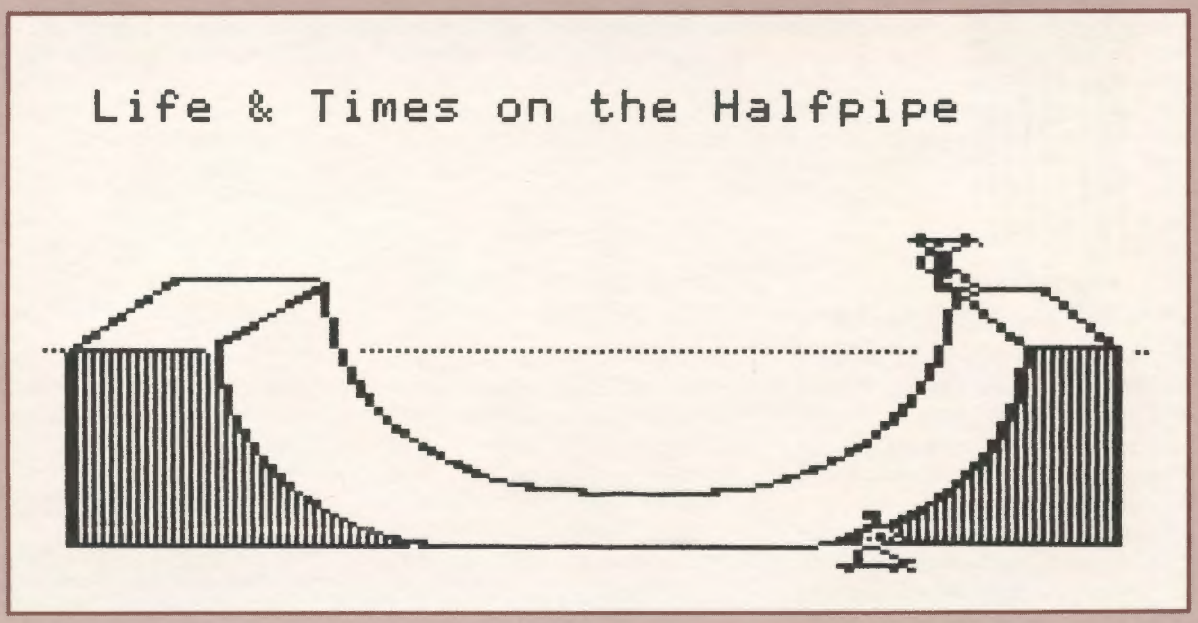
Journal of the ICCE Special Interest Group for Logo-Using Educators

LOGO EXCHANGE

APRIL 1988

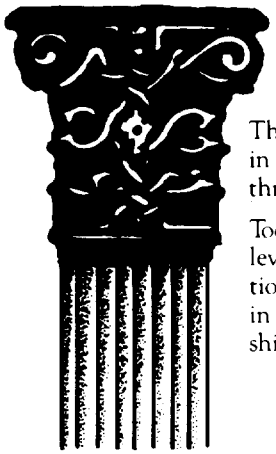
VOLUME 6 NUMBER 8



International Council for Computers in Education

ICCE Publications

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

LOGO EXCHANGE

VOLUME 6 NUMBER 8 Journal of the ICCE Special Interest Group for Logo-Using Educators APRIL 1988

Logo Exchange is the journal of the International Council for computers in Education Special Interest Goup 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.

CONTENTS

SIGLogo Membership *(includes The Logo Exchange)*

	U.S.	NON-U.S.
ICCE Member	24.95	29.95
Non-ICCE Member	29.95	34.95
ICCE Membership *(includes The Computing Teacher)*	28.50	31.50

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

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

(LOCAL "PAGE)

From the Editor

Who owns Logo?

by Sharon Burrowes Yoder

More correctly, who owns Logo ideas: those who developed Logo? those who sell Logo? Logo users? Papert? No one? Everyone? Such a seemingly innocent question doesn't seem to have a simple answer at all.

In the past few months, this simple-complex question has arisen numerous times as I have prepared issues of *LX* for publication. Let me give you some background.

Frequently authors and columnists "footnote" code contained in their articles, sometimes as their own and sometimes as belonging to others. It is not uncommon for me to recognize this code as the product of some of the very earliest work in Logo, often having its roots before Logo was even available for microcomputers. Many of the references given by these authors are not technically accurate because they don't give credit to the originators of the ideas.

Allow me to cite a few examples. The PICK procedure used to randomly select an item from a word or list is practically a Logo primitive for those working in traditional list processing. Often references in *LX* articles cite Glen's work...but I personally first saw the PICK procedure in Hal Abelson's books before *LX* even existed, and undoubtedly that procedure was part of even earlier work with Logo. Glen, who is well aware of Abelson's work, first used PICK in *LX* in a column in 1984. So, who "owns" PICK?

Another example is dynamic turtle microworlds. Dan Watt's books (*Learning With Logo*, McGraw Hill, 1984) are often cited here, and yet much of the work with dynamic turtles goes even further back to Andy diSessa's work in the early 1970's at MIT. Does someone "own" the dynamic turtle concept? I have seen references to language microworlds such as Bob Lawler's Beach World attributed to both Michael Tempel and Gary Stager, both of whom extended the ideas, but they did not originate them.

There seem to be three things going on here. First there is the "requirement" placed on those who publish articles to give appropriate references. From a scholarly point of view, accurate citation of sources is certainly important. Proliferation of misinformation is not something any of us wants to foster, and it is certainly not a good model to set for our students in any case.

Secondly there is the sharing that is an innate part of the Logo culture and enviroment. We encourage our students to work together, to help each other, to share ideas. We share our ideas among ourselves at professional meetings and over telecommunications networks. We seldom think about ownership in these situations.

Third, there is the natural tendency of human beings to express pride in their accomplishments. Thus, when an author or columnist develops or extends an idea, it is only natural for him or her to take credit for the work.

What is the place of each of these viewpoints in the Logo literature? As I see it, those who write about Logo fall into three categories. First is the scholar writing research articles or Ph.D. dissertations. Here accurate citation of original sources is clearly appropriate.

Second are those individuals with deep roots into the history and philosophy of Logo. These people have either been working with Logo for a great deal of time or have read much of the early Logo literature, and thus are aware of the historical roots of many Logo ideas. Those with such deep roots should share in knowledge with the rest of us, especially when such knowledge is anecdotal and unlikely to be cited elsewhere.

Finally, there is the newcomer to Logo who is extremely enthusiastic about his or her Logo experiences, but lacks the depth of the researcher or the historian. Such individuals should feel free to openly share their personal experiences. Their point of view is a valuable as that of the scholar or historian.

Which of these viewpoints belong in *LX*? To that my answer would be all three. We need the enthusiastic newcomer to tell us that Logo is still as exciting as it once was to those of us with experience. We need to hear from those with deep roots in the Logo culture to give us perspective. And, certainly, we need to hear from the scholars in order to know what is going on in the research community.

So, perhaps, my concern has really been unfounded. In keeping with the Logo philosophy, we should all be more accepting of a variety of points of view from all quarters. Only through sharing can we all grow and learn.

[Sharon Burrowes Yoder, ICCE/SIGLogo, University of Oregon, 1787 Agate St., Eugene, OR 97403-9905].

Cover: Mark Benda was in the 7th grade when he made this skateboard program. He is in the Walker Jr. High School Computer Club in La Palma, California. Mark thinks Logo is neat to make shapes, draw scenes and animate pictures.

Monthly Musings

Hyper Hype III

by Tom Lough

In the February and March musings, I introduced HyperCard and explored some specific comparisons between features of HyperCard and those of Logo. I'd like to wrap up this series of musings with a look at one of the most exciting capabilities of HyperCard. This capability is shared with Logo and other powerful computing languages, and brings HyperCard reasonably close to what some might term an artificial intelligence environment. In a word, I'm talking about self-modification.

The properties of various HyperCard objects make self-modification possible. For those of you familiar with the Logo property lists, the concept of properties is not totally foreign. However, the idea is applied a little differently in HyperCard.

HyperCard provides nearly 40 properties which can be applied to a variety of objects. The properties are gathered into global, window, stack, background, card, field, button, and painting property groups. Properties include such things as location, name, text font, and the like. The values of the properties are manipulable by HyperCard instructions within scripts.

The particular property which makes self-modification possible is the *script* property. It roughly corresponds to the TEXT operation of Logo, in that it provides access to the text of HyperCard scripts. This means that the text of a script (of a card, button, etc.) can be modified from within the same or another script. Thus, it is possible to devise stacks with scripts which can change themselves. Scripts might be able to "learn" from experience and change their performance accordingly. This powerful capability suggests that there may be some artificial intelligence related applications in HyperCard's future.

Experienced Logo users have been experimenting with self-modifying procedures for several years. There are many well known examples of changing values of variables as a means of modifying the behavior of the procedures as they run. For example, see the ANIMAL procedures in Abelson's BYTE/McGraw-Hill series of Logo books (e.g., *Apple Logo*, p. 162 ff.).

It is also possible to change the behavior by modifying the Logo procedures themselves. For LX readers who have not yet given this activity a try, here is a brief example to introduce you to the idea. Once, when I was using Logo as the basis of an algebra exploration, I wanted a procedure which looked like this:

```
TO  X
 OUTPUT 42
END
```

The actual number to be output by the X procedure would change from time to time. In fact, I wanted this number to be assigned by a random number generator at specific points. Since the class members had not had much computer experience, I wanted to stay away from the quotes-vs-dots syntax problems. I designed an additional procedure to redefine the X procedure with a random number as the output value.

```
TO REDO
 DEFINE "X LIST [ ] LIST "OUTPUT (1 +
   RANDOM 50)
END
```

Here, I used the fact that the text of a Logo procedure is just a list of two lists. The first list contains the names of any input variables. In the X procedure, there are none, as indicated by the empty brackets [ ] in the DEFINE line above. The second list consists of procedure instructions. Here, I wanted OUTPUT and a random number from 1 to 50.

To see the text of a Logo procedure, use the TEXT operation.

```
SHOW TEXT "X
[ [ ]  [ OUTPUT 42 ] ]
```

An IS procedure helped create a sort of "natural language" feel to the experience. [Note: The IS procedure will not work for Apple Logo I, Terrapin Logo, and IBM Logo. If you are using one of these versions, do not define the IS procedure. Use PRINT instead, as in PRINT X > 5.]

```
TO IS  :WHAT
 PRINT  :WHAT
 PRINT  [  ]
END
```

Now the students could do things such as the following.

```
REDO
IS  X  >  5
TRUE

IS  X  <  10
TRUE
```

Monthly Musings — CONTINUED

```
IS  X  <  7
FALSE

IS  X  <  9
TRUE

IS  X  =  8
TRUE

PRINT  X
8

REDO
X  =  8
FALSE
```

Although this example is nowhere near an artificial intelligence application, I hope that it illustrates how self modifying languages, such as Logo, can change the way procedures perform.

The same idea applies to the self-modification concept with HyperCard. In his widely-read book, *The Complete HyperCard Handbook*, Danny Goodman suggests that some HyperCard stacks might be designed so that they will "learn" the habits of their users and make appropriate adjustments in their performance. This has some significant implications where personalization of instruction or learning environments is an issue.

For example, in an article in the August 1987 issue of *Computer Currents*, George Por talks about the potentials of HyperCard-like programs for such educational applications. "Some of those potentials are a fully personalized educational system in which learners are more in charge of the pace, style, and direction of their learning; and the development of collaborative learning/teaching methods that provide both team rewards and individual accountability."

This certainly reminds one of some similar properties of Logo learning environments, doesn't it?

To be sure, HyperCard could be used for computer assisted instruction, drill and practice, online quizzes, and the like. This may happen, since HyperCard does not come with a suggested philosophy of application such as Papert provided Logo users in *Mindstorms*. But the potential is there for individualized learning environments with the learners in charge. The question is, "How to exploit it?"

In his article, Por holds up a challenge to educators. "The new [hyper] media can provide a supportive technological environment for educational innovation, but it won't fulfill its promises without teachers and learners dedicated to exploring its potentials and pushing its limits."

But, with all this, let's remember that we can also apply in our Logo environments what we learn from HyperCard. For example, if you were at the Great Lakes/East Coast Logo Conference in Cleveland May 5 - 7, 1988, you may have seen Andy David's Saturday session entitled, "Hypertext: A LogoWriter-Based Environment for Non-Linear Reading and Writing." That's right, folks. Andy has devised a way to create a hypertext environment from the LogoWriter's screen pages. His project is being field-tested in Chicago classrooms.

In the past three columns, I have introduced HyperCard and pointed out several similarities it shares with Logo. I am not suggesting that HyperCard is a replacement or substitute for Logo. But I believe that HyperCard's potential is worth exploring and learning from. We are in a position to influence constructively the direction of development of worthwhile HyperCard applications in education, based on Logo-related teaching philosophies, and, in turn, be influenced by HyperCard itself. Shall we give it a go?

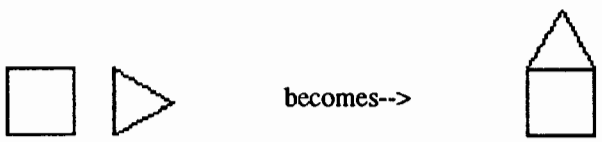
FD 100!

Tom Lough
PO Box 5341
Charlottesville, VA 22905

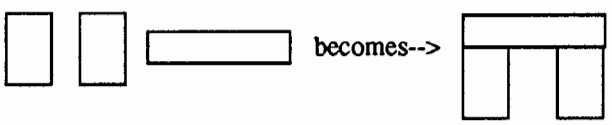
Bottom Up Programming

by David Moursund
and
Sharon Burrowes Yoder

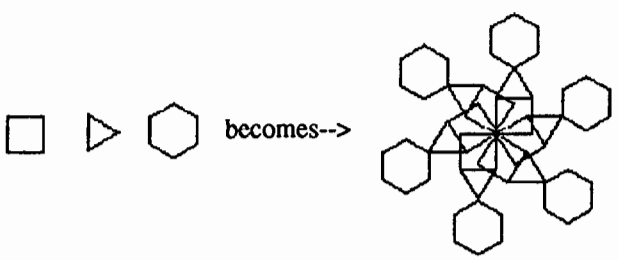
Much of the writing aimed at Logo teachers talks about discovery learning. These discussions are often based on ideas put forth by Papert in *Mindstorms*. Teachers are encouraged to let students explore on their own; not to impose particular assignments to be completed. At the heart of this idea is the concept of *Bottom Up Programming*. Programming in the bottom up style involves working from the parts to the whole. The classic "draw a house" example is often approached in a bottom up manner. A student first learns to draw a square and then later learns to draw a triangle (as well as other geometric shapes).



Throughout the learning process, the student is encouraged to mess around with shape-creating procedures, using one or more of of these shapes to produce interesting combinations. One student may report, "I made an arch using two squares and a rectangle as the top."



Another may report "I made an interesting design using a triangle, square and hexagon."



Note that the important concept here is that the *student* who drew the house sees that a square and a triangle go together to make a house rather than being told to draw a house. But not all students will spontaneously make such discoveries.

Teaching Bottom Up Programming

How then does the teacher encourage students to explore using bottom up techniques? Initially, the student may need models such as those shown above. Bulletin boards of samples of other student's work could serve as a guide for those just beginning such open ended work. If a computer system appropriate for demonstration is available, a teacher could demonstrate (with student input) a variety of explorations

Brainstorming is a technique used in many other contexts, and it is especially valuable here.. The teacher can present students with components, such as the shapes used in the examples above, and ask students what they can think of that could be made with these shapes. Here, working with a group of students is important. Ideas generated by one student will trigger ideas from others. It is important to value every student's contribution to the discussion. There should be no "right" answers, and little or no discussion of any student's ideas. With a variety of ideas in hand, students can then move to the computer to explore on their own.

Assignments to Mess About

The assignment to mess around and see what you can create can be given at any place in a student's Logo development. And, it can always be accompanied by a suggestion that students brainstorm by themselves before going to the computer. (This is particularly appropriate when computer access is quite limited.) For example, given mastery of the fundamental turtle graphics commands, the HOUSE procedure given below, and the REPEAT command, a student can easily produce a variety of products.

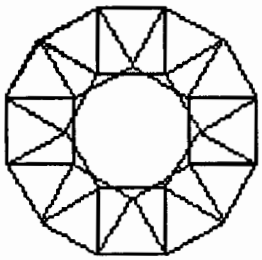
```
TO HOUSE
 SQUARE
 MOVE.TO.ROOF
 TRIANGLE
END

TO SQUARE
 REPEAT 4 [FORWARD 50 RIGHT 90]
END

TO MOVE.TO.ROOF
 FORWARD 50
 RIGHT 30
END
```

Bottom Up Programming — CONTINUED

```
TO TRIANGLE
 REPEAT 3 [FORWARD 50 RIGHT 120]
END
```

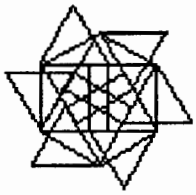
Below is the kind of exploration that you might observe in a Logo classroom where students have learned to use these procedures and have been encouraged to use them creatively.

```
REPEAT 12 [HOUSE]
```



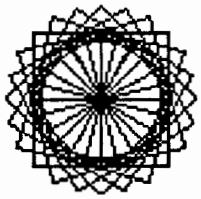
Hmmmm...this looks a bit like a wreath. Perhaps I could draw a holiday scene.

```
REPEAT 6 [HOUSE RIGHT 30]
```



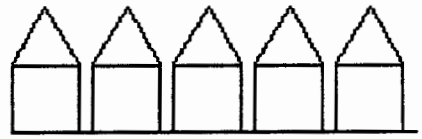
Now I have a pinwheel. Perhaps I could make a group of pinwheels of different colors.

```
REPEAT 24 [HOUSE LEFT 15 BACK 25]
```



What an attractive design! I could use it for the blossom of a flower in a flower garden.

```
REPEAT 5 [HOUSE LEFT 30 BACK 25 RIGHT 90
          FORWARD 30 LEFT 90]
```



Ah! A street full of houses...now I need trees and cars and maybe even some clouds in the sky.

Note that the kind of results described above can result from two categories of explorations:

1. The student is merely experimenting with combinations of commands, much as a composer might experiment with notes and phrases to produce a pleasing whole. The student has no specific goal in mind. Whenever a pleasing pattern is produced, the student takes note of the result and perhaps builds further on the result.

2. The student is trying to reach a particular goal ("I wanted to make a flower.") and initial progress or a bug produces something interesting that leads to a project different from the original plan. ("My attempts to make a flower produced something that reminded me of a Christmas wreath so I decided to make a Christmas wreath instead. See how nice it looks!")

Bottom up programming and Creativity

Bottom up programming often results in a lot of messing about — much as a child in art class might mix paints in an experimental manner to see what new colors emerge or try various pieces in a collage with no particular goal in mind at the outset. Students sometimes will experiment for a long time with a new idea. Just as the HOUSE procedure above was turned into a variety of designs leading to new ideas, so working with even the simplest building-block patterns may result in a myriad of interesting projects.

This kind of exploration can be viewed as encouraging creativity. Making new connections is encouraged and valued. Fluency (the number of different ideas produced) and flexibility (the variety of ideas produced) are both part of the bottom up process. In fact, bottom up programming can be *used* as a way to encourage and teach creativity.

What can be said about bottom up programming and problem solving? In the case of open ended exploration, the student

Bottom Up Programming — CONTINUED

needs to explicitly understand that the goal is to create an interesting whole made up of parts. In the case of an abandoned goal, the student should be helped to see that s/he explicitly stopped working towards the original goal in favor of a new goal. Both of these approaches encourage use of the creative process to put together pieces in unexpected ways.

A caution is in order here, however. Some students may always choose to abandon their stated goal when it is not easily reached. While there is a place for such decisions, especially when the student is particularly bothered by failure, there is also a place for establishing guidelines to be reached and achieving them.

Deciding when and how to honor the bottom up approach to Logo programming can be somewhat complex. Some students' learning style lends itself quite well to the bottom up approach. Others have structured styles of thinking that make the bottom up approach difficult for them to use. Still others may use the bottom up approach to avoid failure. Clearly dealing with these issues requires the attention of a caring and thoughful teacher. Making students explicitly aware of the advantages of building from the parts to the whole will help the students *themselves* make better decisions regarding the problem solving style that they should use for a particular project.

Transfer Examples

As indicated in the previous article in this series, learning to use a particular problem solving skill only in Logo is not enough. Explicit effort has to be made to help student to see how each new problem solving technique applies in other ares of their lives. Below is a list of examples that illustrate possible uses of a bottom up style problem solving in a variety of other areas.

1. You're hungry...you go to the refrigerator to see what is there. Your meal is based on what you find (rather than some pre-planned menu.)

2. In art class, a wide variety of materials are made available for a project. Some students will take a bottom up approach and let the materials they see dictate the content of their project.

3. A research project is assigned and a student picks a topic. Along the way, an interesting article leads towards a different topic entirely.

4. You want to listen to some music at a friend's house. Your choice is dictated by the tapes that the friend has in his/her library.

5. You are presented with a sentence in a foreign language that you know moderately well. You may begin the translation by examining the words you know and guessing at the rest from context.

6. You are in charge of planning the choreography for a dance. Your steps are limited by what the members of the group that will perform your dance can do.

7. You have a game plan in mind for winning a game. In the midst of your assault on your opponent, your opponent's key player is injured and can no longer play. You immediately devise a new strategy to take advantage of weaknesses of the substitute player.

8. You're planning to create a last-minute gift. Your final product is limited by the scraps of fabric or wood that are available.

9. You are shopping for a birthday present for a friend. You had in mind buying a sweater. As you walk down the isle toward the sweater section of the store you see that shirts are on a half price sale. You decide to buy a shirt.

10. There's a jigsaw puzzle on the table to put together, but the box is missing.

Note that many of the above examples could indeed be viewed in light of the creative process. Non-linear or unusual connections need to be made to solve many of these problems. Any place that you as a teacher are working towards the development of the creative process may indeed be a place where a connection with bottom up programming can be made.

[Dave Moursund and Sharon Burrowes Yoder, ICCE, University of Oregon, 1787 Agate St., Eugene, OR 97403-9905].

Logo LinX

to the Left
Sunny Side Up
Judi Harris

"April Comes like an idiot, babbling and strewing flowers."
——Edna St. Vincent Millay, 1921

In honor of All Fools' Day (albeit a few months late), I would like to suggest some Logo foolery. WHAT? No fools are allowed in your classroom? No foolishness is prescribed in your curriculum guide? Your lesson plans are foolproof? It may be a foolhardy move, but may I humbly suggest:

"To never see a fool you lock yourself in your room and smash the looking-glass."
——Carl Sandburg, 1936

Slanted View

What is the difference between these two electronic mail messages?

> Subject: Project Grading
> From: Glen
> Date: April 1, 1988
> To: Judi
>
> Judi,
>
> About the assignments to grade: Let's toss 'em down the stairs & the ones that get nearest the **bottom** get the highest grades. (You did say grading was subjective.)
>
> Glen

> Subject: Project Grading
> From: Glen
> Date: April 1, 1988
> To: Judi
>
> Judi,
>
> About the assignments to grade: Let's toss 'em down the stairs & the ones that get nearest the **bottom** get the highest grades. (You did say grading was subjective.) ;-)
>
> Glen

After reading the first message, you may have thought that this professor was somewhat unfair in his evaluation practices. But after reading the second version, if your view was properly slanted, you would probably conclude that he was temporarily under the influence of Poe's Imp of the Perverse. The discerning factor, of course, was the strange symbol at the end of his suggestion. Still don't see it? Just tilt your head to the left until you see a familiar face.

Face It

Funny faces, made of text characters, and seen when looking sideways, can punctuate sentences with humorous emotion. These symbol collections are also called "emoticons," or

"....[figures] created with the symbols on a keyboard that [are] read with the head tilted to the left. Used to convey the spirit in which a line of text was typed."
——CompuServe data library

Emoticons are typically used in informal typed correspondence, such as electronic mail. They can also add graphic dimensions to word-processed stories and poems. And, as you may have guessed, they provide a creative Logo list processing challenge.

We can define procedures that output the characters of emoticons as lists.

```
TO EEK!
 OUTPUT [<:-O]
END

TO BATMAN
 OUTPUT [B^)]
END
```

These procedures can then be used in Logo stories.

```
TO CLIMAX
 PRINT SENTENCE [IT WAS THE MONSTER!] EEK!
 PRINT (SENTENCE [BATMAN] BATMAN [TO THE
   RESCUE!]
END
```

```
CLIMAX
IT WAS THE MONSTER!  <:-O
BATMAN  B^)  TO THE RESCUE!
```

A Fool's Paradise

Emoticon variations can reflect many different humorous themes. Here are just a few that I've collected from electronic correspondence.

:-)	humorous	:/)	not humorous
:-D	smile!	:-P	bleah!
:-*	oops! (covering mouth with hand)		

:~i	smoking	:/i	no smoking
:-X	kiss	:-B	overbite
:-J	tongue-in-cheek	:*)	clowning around

Emoticons can also represent famous personalities.

P-)	Bluebeard	:-)	Count Dracula
=l:-)=	Uncle Sam	7:)	President Reagan

"The features of our face are hardly more than gestures which have become permanent."

——Marcel Proust, 1913

Sidekicks

Emoticons can be generated at random with a few simple Logo procedures. Their "translations," of course, are left for your students to discern.

Using Bull and Cochran's PICK procedure,

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

text character facial features can be chosen at random,

```
TO EYES
 OUTPUT PICK [ : ; & % | ]
END

TO NOSE
 OUTPUT PICK [ - * ^ " > @ ]
END

TO MOUTH
 OUTPUT PICK [ ) ( # P O X [)=] B E I |~ ]
END
```

and then concatenated into complete emoticons.

```
TO FACE
 PRINT (WORD EYES NOSE MOUTH )
END

REPEAT 3 [FACE]
|^#
%-B
:@(
```

Calling All Fools

Are you interested in helping your students to look sideways? If you will send us printouts of the text faces that you and your students create, we will compile and publish a collection of some of the most original submissions in a future issue of *Logo Exchange*. Please be sure to include each emoticon "translation," or label, and the fool's name who designed it.

Send your creations to:
Judi Harris
287 Ruffner Hall
University of Virginia
405 Emmett Street
Charlottesville, VA 22903

Here's hoping that these faces will launch a thousand smiles!

LXionary

by Bill Craig

"Computers: A Contrary View" by Howard Rubin, February, 1988.

The article is a short description of one school's instructional use of computers. The author, the superintendent of Franklin Square School District in New York, cites the inability of his teachers to use skills learned in computer literacy classes and the subsequent decision to devote computer use exclusively to word processing. In response to those who ask why Logo is not being taught, Rubin says

> *I strongly believe that with the limited computer time available, the improvement of writing skills is more important than learning Logo. There is little evidence that skills used in Logo are transferable to the elementary school curriculum.*

My initial reaction was to snicker at such an unenlightened statement. I stopped snickering when I realized that the statement was made by a person of significant responsibility and is probably representative of the opinions of other people with similar decision making power. The statement is indication that the issue of evaluation of Logo is not just a university ivory tower issue. It is indication that evaluation of Logo use is the most important issue the Logo community today faces.

Continued on page 22

Making the Switch to LogoWriter 2.0

by Eadie Adamson

LogoWriter 2.0 for the Apple is here! What's in it for you? Some of you may be wondering: What's the real advantage, if any? Is it worth it to switch? Should I bother now? How complicated is it to switch if my students have work they want to keep?

For the past few weeks (this is written in early February), I have been playing a bit with this new version, which requires 128K. I have also been testing a pre-release version of the Disk Manager for 2.0. I hope that my experiences might be helpful, regardless of which of the above questions applies to you.

Before beginning my discussion, a couple of comments are in order.

- *You can use version 2.0 as if the only change to LogoWriter were increased memory. You can* ***ignore*** *both ProDos and any added features discussed below.*

- *LogoWriter Version 2.0 is ProDos based. Don't panic. ProDos is just an operating system. Any program uses some kind of operating system. An operating system is simply a collection of programs that the computer uses, but are generally invisible to you. The "Dos" part of the name refers to Disk Operating System, thus ProDos gives you the capabilities of working with disks, among other things. LogoWriter 1.1 uses its own Dos. Much earlier educational software for the Apple uses Apple's DOS 3.3.*

- *Throughout this article I will be referring to ASCII files. An ASCII file is a standard text file, consisting only of the keyboard characters that you can type. Formatting and other special characters are removed. Most word processors allow you to save and load ASCII files (sometimes called text or text only files.)*

Is it worth your time to switch to LogoWriter 2.0? For my purposes it definitely is, even in the middle of the year, but I do not have a large number of students to deal with and that obviously makes the investment of time in converting files much less.

Let's look first at the new features, then at the problems, and finally the processes of switching. First of all, you **must** have 128K Apples to be able to use this version. If you are still using 64K, this means you need an extended 80-column card for each computer in order to run Version 2.0.

New Features of LogoWriter 2.0 for the Apple

There are many real advantages to the new version. Here are some I consider important:

If your students, like mine, are writing large programs and are beginning to have problems with the amount of memory on the page, the doubled page size of 8K instead of 4K is an excellent reason to make the shift.

There is another feature in this new version which can become very important, especially with more elaborate projects. Now pages with completed programs can be locked by typing LOCK in the command center. This allows someone else to play with the pages, but as long as the file is locked (you see LOCKED clearly on the upper left of the screen) no changes made on either side of the page will be saved when Esc is pressed. This eliminates the need for a "SAFETY" procedure to protect a page. As long as everyone follows a rule of *not* unlocking someone else's page, their pages remain protected.

Some new and helpful messages have been added to version 2.0. **Busy...** appears on the right side just above the command center when a disk is being accessed, or **Printing...** when you are printing.

A new command, SAVETEXT, permits saving files as ASCII files. These are not LogoWriter files and do not show on the Contents Page. These files can be read by other programs such as AppleWriter and AppleWorks and can be sent via modem to another computer. When you use SAVETEXT you can see if the file was saved by typing SHOW FILELIST.

I made use of the SAVETEXT feature to add the LogoWriter procedures to my "Creating a Kaleidoscope" column (*LX* Feb. 1988). Within LogoWriter I arranged the procedures as I wanted them, then saved the file as text (SAVETEXT "PROCEDURES). Next, I started my word processor and loaded the procedure file into my word processor document. This way I could be sure that I didn't make typing errors when transcribing my procedures.

You can also use LOADTEXT to load LCSI Logo II files (or any other standard ProDos text files) into LogoWriter 2.0, then give the page a name, and the files are now LogoWriter files! That's a boon for all of us who have done a lot of work with Logo II and have files we still want to use. Or, if you're like me, and have written a lot of mini-instruction sheets on another word processor which allows you to save standard ASCII files, it's now possible to put them on a LogoWriter disk if you wish.

SETX, SETY, XCOR, YCOR, SQRT, and TOWARDS are all now primitives, as are INT, ROUND, and ARCTAN: you don't need to write them any more. SAVEPAGE saves the page without leaving it (quite a time saver if you're transferring a single page to several disks!), while LEAVEPAGE allows you to leave a page without naming it. GETSHAPES loads the shapes from the current disk quickly without going to the shapes page itself; this is a useful command to put in a startup procedure if you're working with programs on several disks or moving from one subdirectory to another..

Making the Switch to LogoWriter 2.0 — CONTINUED

Undocumented primitives in the new version include RL (for READLIST) and RLCC (for READLISTCC) and DIR (for directory). Still missing are TEXT and DEFINE.

In addition to new primitives and the convenience they offer, all of the features of ProDos are also available. This means that subdirectories can be created, each with its own shapes page, so that there is no longer a limit of one shapes page per disk. You can now have thirty shapes for *each* subdirectory. The GETSHAPES command, when added to a STARTUP procedure or typed in the command center, will switch shape pages when you switch directories.

Getting Organized to Convert Files

Once you've decided to switch to 2.0, it pays to be organized about it. If you're converting a number of disks at once, it will speed the process if you have several computers with double disk drives to work with.

Be aware that conversion makes a scrapbook disk only. You will still need to set up a **program** disk, either by putting the LogoWriter program on one side and the files on the other, or by giving students **only** a **scrapbook** and requiring that they — or you — load LogoWriter into each computer before beginning. I prefer to have each student be completely independent, so I want each to have a program as well as a scrapbook. The LogoWriter program itself takes up a large part of one side of a disk, leaving only a little page space, a good reason to work with a scrapbook. Whether you give your students two disks or one is up to you. In the process outlined below, I placed the converted files on the back of each disk and then put the LogoWriter program on the front.

You also need to know that conversion converts all files on a disk. There is no option to select which files to convert. You can save time either by having your students clean up their disks first (not a bad idea from time to time) or be prepared to convert all their files. If you need to convert just a single file, save time by putting it on an empty Version 1.1 Scrapbook disk, then convert it.

The easiest way to work is probably the most expensive: give everyone a new disk. If you don't want to give everyone a brand new disk, you'll still need a few blank disks to work from. Then you can work as outlined below, transferring from one disk to another as you convert.

Here's how to set up to convert 5 1/4" disks:

1. Load the Disk Manager in each computer you plan to use for conversion.
   (If you are converting disks the Disk Manager does not have to stay in the drive.)

2. Select CONVERT LOGOWRITER DOS TO ProDOS. (Read the selections carefully. They are crowded together so that it is too easy to choose the wrong one.)

3. Put the **old** disk (the one **with** files to be converted) in Drive 1, the new (to be ProDos) disk in Drive 2. (I notched all my "new" recycled disks and put the converted files on the back. Each "new" disk was put in Drive 2 upside down at this point.)

4. When prompted, type a volume name for the destination disk (I use students' last names as a volume name and for the disk.)

   Occasional renaming will be required for nonlegal names in ProDos. If your students have used numbers for page names or put quotes at the end of a page name, for example, you will be asked to give a new name or press the space bar to skip the file. (**Be careful** not to press Esc to skip a file or you'll need to restart the conversion!) ProDos accepts only a letter as a first character, and accepts only letters, numbers, and periods in the remainder of the name.

5. Don't relabel a disk until it is converted! Pull out the old disk, check the name, put the correct label on the converted disk. The next old disk gets its label torn off and is then placed in Drive 2 to be used for the next "new" disk.

6. One important record keeping idea: **Mark** each disk you convert as well as each program disk you make. I put "LogoWriter 2.0" and number each disk, as well as adding a "Version 2.0" and the date of the Disk Manager (which I mark when I receive it) on the hard-backed notebook in which my students keep their disks. That way, should there be an upgrade, I can tell quickly which disks have this version.

One caveat which can't be repeated often enough: Be aware that you are only **converting files** to ProDos LogoWriter during this process. You may think, as I did when I first tried this, that you are also creating a program disk: you are **not**! If you try to boot a converted disk, your Apple will report UNABLE TO LOAD PRODOS. There's nothing wrong with your computer; you simply need a LogoWriter Program disk in order to use the new files.

Although conversion can be a slow process, just remember that once everyone begins using ProDos LogoWriter, this is really a one-time task!

Making the Switch to LogoWriter 2.0 — CONTINUED

Copying LogoWriter Disks

The "Copy a Disk" option is included on the Disk Manager. It is there to help people who have no other copy program. I found, however, that it took two minutes and fifteen seconds to copy a LogoWriter disk with this option, and just under fifty seconds with Copy II Plus. If you have another copy program, do use it to make your copies.

When you have finished converting files, set up several computers to copy LogoWriter on the front side of your student disks. Make one master disk configured for your printer (see below). Start your copy program and copy this LogoWriter disk onto the new ProDos disks. Be sure **not** to copy on the same side on which you converted: this will wipe out those files!

Printer Settings

Setting a printer and printer card can be a little confusing, although it is a quick and easy operation. When you create a LogoWriter disk it is preset for the ImageWriter and Super Serial card. To set your disk for any other combination of printer and card, you must choose your printer and printer card from lists that appear on the screen. If you have a non-standard printer or card, it may not show on the list. Call LogoWriter Technical Support for advice or try different combinations until you find what works. (If you do explore on your own, change only **one** setting at a time and write down what you did. Make a note on the disk label when you find the correct setting.)

When your selection is made the confusion begins. You will see a message: "ERASING DEFAULT PRINTER" followed by a message "WRITING DEFAULT PRINTER." What is actually happening is that the "default" printer is now the printer you just selected. The same thing happens for the printer card. Don't panic over the confusing message, but **do be sure** to check that your settings were correct by booting the disk in the computer you plan to use with the printer. Choose a new page and try PRINTSCREEN, PRINTTEXT, and PRINTTEXT80. If they work, your job is done. If you have a problem, try again before calling Technical Support. (Be sure to check your printer with other software also just to be sure it **is** working!)

LogoWriter for the GS 3.5 Disks

The Disk Manager will give you the impression that it makes a 3.5" disk for you. I tried to do this with an unformatted 3.5" disk and had no success. To make a 3.5" LogoWriter disk on the GS, use your system disk. First, format your disk in ProDos. Then copy the files from your 5 1/4" disk onto your 3.5" disk. Next, if you wish, copy the tool files onto your disk. This is a fairly rapid process. (I suspect the Disk Manager would have worked if I had used a formatted disk, but the System Disk is bound to be faster on the copying, and why load it twice?) From here, once your master disk is made, use the FastCopy option on the System Disk to make your other disks. (If you don't have the new System Disk, any Apple dealer can give you an upgrade. Just bring your original and a blank disk for the copy.)

Other Management Techniques

Although I managed to have some computers converting and others copying at the same time, I don't recommend it. It's too easy to get lost in the process if someone interrupts you. Stick to converting first (it's likely to be the lengthier process); then switch to copying the program onto each disk or making separate program disks, as you prefer.

Don't discard your old LogoWriter master disks or Disk Managers. You will still have occasional use for them. Store them away safely. Keep at least one master for each version in a handy place.

Keep some spare master LogoWriter disks and a few scrapbook disks on hand. Then when you need to make new disks, you will only need to load your copy program to do the work.

If you have a number of tool files for your students, make a scrapbook disk which contains all the tools. Label it TOOLS, and use the CATALOG.TOOL (from the "Try Me" disk) to print a catalog. Attach this to the disk jacket so you can find these files quickly when you need them. It's certainly worth making a backup copy of this disk if you have a number of tool files.

If you have several different printers, make a startup disk for each and store the startup disk near the computer(s) connected to that printer (hang it on the wall if there's room).

I haven't touched upon quite all the new features for ProDos LogoWriter, but I hope I've mentioned enough to convince you that it's worth the change, if not now, certainly for next year. And I hope the system I've outlined for doing conversions will make the task easier. If you're lucky, you might even find a few students who would love to help!

[Eadie Adamson, Allen, Stevenson School, 132 East 78th St., New York, NY 10021.]

Logo Connections: An Open Architecture for Logo

by Glen Bull, Gina Bull, Tom Lough, Judi Harris and Cheryl Wissick

In the early 1970's Seymour Papert developed a paper titled"Twenty Things You Can Do with a Computer," published both as an MIT Logo Memo (#3, June 1971) and in the April 1972 issue of Educational Technology. This paper outlined a vision of what a classroom computer might become. Many of Papert's prophecies have proven true. For example, at the beginning of the 1970's, Papert worked on applications predicated on the assumption of one or more computers per classroom, even though the personal computer did not exist at that time. Today there is scarcely a school in the country which does not have a microcomputer.

Surprisingly, many of the applications Papert listed in "Twenty Things You Can Do with a Computer" are not yet in common use. At least, they are not yet found in the typical classroom. For example, Papert suggests that a yo-yo might be tied to the end of an arm with a motor. Logo commands such as "UP" and "DOWN" could be used to operate the motor and control the yo-yo. A similar application involved placement of an upside-down broom in the middle of a motorized wagon. Logo commands and sensors could be used to balance the broom and keep it upright. The Logo program required to do this would be compared to kinesthetic and sensory feedback required to keep a broom balanced upright on the palm of a hand.

Even though some of Papert's visions are still in advance of present-day reality, they are useful because they point to possible futures. As the applications described suggest, Logo need not be an end in itself, but can be a starting point for connections to other applications.

Open Architectures

"Connectivity" is a characteristic which distinguishes two of the most successful microcomputer systems of all time. At one time there were dozens of microcomputer systems. Of these dozens, two which survived and flourished are the Apple II computer and the IBM personal computer. These two systems share a common characteristic: they both have an *open architecture.*

An open architecture chiefly means two things. First, both the Apple II and the IBM PC have a bus with expansion slots in which additional computer cards can be placed. Second, details of the inner workings of the computer are openly published so that the computer can be made to work with other vendors' hardware and other third-party programs. As a result, the Apple II is popular in schools today even though the basic technology was designed by Steve Wozniak over a decade ago. And the IBM PC became a standard architecture for business applications.

An open architecture implies that it is possible to use other people's hardware and other people's software with a computer. Today that seems self-evident, but it was not always so. Many microcomputers of the past were sold in the form of a sealed box, with no expansion slots. Details of the computer's architecture were held as proprietary secrets. As a result, thousands of hardware and software developers focused their efforts on the systems of manufacturers which welcomed their efforts.

It is worth pausing for a moment to consider a possible exception to the rule that the most successful computer systems have all had open architectures. The Macintosh was originally packaged as a sealed box without slots. However, Apple was careful to provide developers with detailed information about the workings of the computer, and provided a Small Computer Systems Interface (SCSI) port intended to overcome the lack of actual hardware slots. Interestingly, slots have been restored to the more recent Macintosh computers, such as the Macintosh SE and the Macintosh II computer systems.

Twenty Things You Can Do with a Computer

In this light, it is worth considering what an "open architecture for Logo" might mean. An open architecture for Logo implies connections between Logo and other programs and other hardware. This link between Logo and other hardware and software multiplies Logo's power many times, and opens new vistas for educational applications. With this in mind, here is an updated list of "Twenty Things You Can Do with a Computer (Using Logo)".

With two exceptions, few of these applications are in widespread use. However, they have all been used at one time or another by teachers in the local schools in Charlottesville, Virginia or by education students at the University of Virginia. Many of them have been discussed in past "Teaching Tools" columns in the *Logo Exchange* magazine.

1. Videodiscs
   Logo can be used to control a videodisc player, providing access to up to 50,000 pictures on a single disc. It can also be used to control animated video sequences.

2. Slide Projectors
   Logo can be used to control a random-access slide projector, giving new meaning to the phrase "computerized slide show."

Logo Connections: An Open Architecture for Logo — continued

3. Speech Synthesizers

Logo can be used to make the computer talk, using a speech synthesizer. The procedure SAY used in place of the PRINT command allows Logo to talk.

4. Digitized Sound

A sound digitizer uses a microphone connected to the computer to record sound. The Logo procedure "MOO" can be used to play back cattle mooing, while "BARK" produces the sound of a dog barking. A Logo microphone can also be used to examine other phenomena, such as heart beats and noise levels.

5. Koala Pad

Touch tablets such as the Koala Pad and its cousin, Animation Station, can be used to input coordinates to Logo. One Logo program allows students to generate Logo graphics procedures by touching points on the graphics tablet. Pictures produced by touch tablets and graphics programs such as the Micro- Illustrator can also be imported into Logo.

6. Muppet Keyboard

The Muppet keyboard is an oversized keyboard with keys arranged in alphabetic order. A number of elementary school programs have been written specifically for use with the Muppet keyboard, but it can also be used directly with Logo procedures. For example, the combination of the Muppet keyboard and a speech synthesizer produces a very nice talking alphabet. Pictures can be generated with Logo, or imported from a variety of external graphics programs. The addition of a random-access slide projector or videodisc player opens the door to even more possibilities.

7. Touch Windows

A touch window is a clear screen which fits over the monitor screen. When the window is touched, Logo can detect the area where contact is made. This permits children to point to the screen, rather than using the keyboard. If Logo is used to control a videodisc player, the touch window can be placed over the television screen rather than over the computer monitor.

8. Switch Inputs

External switches can be used in place of the keyboard for control of Logo procedures. This may be particularly helpfully for physically handicapped students, who can not operate a regular keyboard. Other students may use plywood and switches to construct the control panel for a spaceship.

9. Logo Robots

Logo floor turtles ranging from the Valiant turtle to the Turtle Tot are the original Logo robots. The infrared control used with the Valiant turtle raises the possibility of operating other infrared controls through Logo, such as videocassette recorders, televisions, and stereo systems. No one has yet done this, to our knowledge, but students in Lynchburg, Virginia have controlled an industrial robot via a serial interface using Logo.

10. Lego and Logo

A special version of Logo called *Lego-Logo* allows motorized Lego constructions to be operated through Logo procedures. Regular Lego bricks can be mixed with special blocks containing sensors. This permits many explorations in the areas of robotics and artificial intelligence, as well as practical experience with mechanical physics.

11. Light Sensors

With the proper Logo procedures, a light sensor can be used to count people entering and leaving the classroom, or record the amount of cloud cover over the course of a day.

12. Temperature Sensors

Temperature sensors can be used to record changes in liquids as they warm or cool during experiments. Logo can also be used to record and graph the change in outdoor temperature across an entire 24 hour period. Other meteorologic instruments measuring wind speed, wind direction, and humidity can also be connected to the computer and monitored by Logo.

13. Logo Pendulum

An inverted joystick can be used to construct a pendulum, permitting Logo to record and graph the sine wave produced by its back-and-forth movement. This sine wave can be compared with a mathematically-generated sine wave produced by Logo.

14. *AppleWorks*

Poetry generated in Logo can be embedded in an essay written with *AppleWorks*. The text screens for a Logo adventure story can be developed and formatted with the *AppleWorks* word processor.

15. Spreadsheets

The original electronic spreadsheet, *Visi-Calc*, was named for the "visible calculations" made possible by this tool. Data captured with Logo probes and sensors can be transferred to a spreadsheet for visible data manipulation and examination.

Logo Connections: An Open Architecture for Logo — continued

16. *Print Shop* Pictures

Print Shop pictures can form the background for a Logo drawing. Graphics from other picture banks and programs such as *Newsroom* and Beagle Brothers *Mini-Pix* can also be imported into Logo. Similarly, Logo drawings can be exported and used in these graphics programs.

17. Plotters

A plotter can be used to produce color transparencies. This capability can be used to project the beautiful Logo graphic designs, or to make signs for the class bulletin board.

18. Telecommunications

Logo procedures can be shared across schools and across the continent via electronic networks. Schools in Charlottesville will soon use the network to share data collected with scientific probes and sensors controlled by Logo programs.

19. Logo Library

A library of Logo procedures donated by contributors can be stored on an electronic bulletin board. Libraries of this kind are available through commercial services such as CompuServe. Logo libraries are also available on local bulletin boards, such as one available to area teachers and students at the University of Virginia.

20. Electronic Blackboard

Recent breakthroughs in liquid crystal display (LCD) devices have made it possible to project the computer image using an ordinary overhead projector and an LCD projection tablet such as the Kodak Datashow. If one of these devices is not available, an RF (radio-frequency) adapter available at Radio Shack can be used to display Logo on a 25-inch television set. This provides the teacher with an electronic blackboard which permits the entire class to participate in a Logo adventure story, or a Logo exploration in geometry.

This list of 20 activities which can be accomplished by mixing Logo with other technologies was intended to illustrate the range of possibilities. These are all examples of activities which have all been employed in local schools. The field could be widened to include other applications which we have not yet employed, but which could be accomplished with today's technology. For example, home control systems are available which are programmable via a serial interface. With such a system, Logo procedures could be written to control lights, turn on a radio, perk coffee, or operate other electrical devices.

Design Decisions in Constructing Logo

All of these applications are made possible through an open architecture which permits Logo to be used with other hardware and software. Some versions of Logo are more open than others. Some dialects of Logo only have a partially open architecture." For example, some versions of Logo can send output to a serial interface, but can not accept input. Other versions will permit neither input or output, except to a printer. Some kinds of Logo store Logo procedures in an encrypted format that is not accessible to other programs. Other versions of Logo store procedures in a form that makes it difficult to share them over a telecommunications network.

There are two reasons for versions of Logo which have partially open architecture. The first and most important reason consists of hardware constraints. Until recently it has been necessary to make Logo fit in as little as 64 K (kilobytes) of memory. This has made it necessary to leave something out in order to make Logo fit in a limited memory space. Often the part left out includes features that would make it easier for Logo to interact with other programs.

The second reason some versions of Logo do not work well with other software and hardware stems from the design philosophy of the developers. Logo was originally intended to be a language with a low threshold and a high ceiling. That is, it was to be a language which could be used by very young children, but would not be a baby language of little interest to adults. However, some Logo designers later felt that applications such as those listed above would only be of interest to five percent of Logo users — and that it was sufficient to develop a Logo which would only fulfill the needs of the other 95 percent. Of course, this is a self-fulfilling prophecy, since users will not explore these applications if limitations of the language discourage them. (It should be noted that the assumption that only five percent of the users want such features is undemonstrated.) Other Logo designers felt that every imaginable application - or at least every imaginable "worthwhile" application - could be built into the language. In these cases, the ability of Logo to interact with other programs received a lower priority in the design process.

A school's perception of "what a computer is" may also be a limiting factor. In many cases, computers are used just as they come from the manufacturer. In a business environment, a word processor and a copy of Lotus 1-2-3 may be all that is needed to be productive. However, almost every elementary classroom could benefit from speech synthesizers and programs that talk. Touch tablets like the Koala Pad allow young artists to give

Logo Connections: An Open Architecture for Logo — continued

scope to their creativity, and light sensors and temperature probes turn the computer into a scientific instrument. In some schools these tools are used effectively, but in others a printer is the most exotic peripheral to be found. Papert's original vision of "Twenty Things You Can Do with a Computer" needs to be recaptured to expand these limits.

Teachers also need information on ways of combining these technologies with Logo. The commands for turtle graphics (FORWARD, BACK, LEFT, RIGHT) are similar in almost every version of Logo. However, there is hardly a dialect of Logo that does not access the serial interface port in a different way (for those versions of Logo in which this is possible at all). This bewildering variety of differences makes it difficult to share ideas with teachers at another school who use a different dialect of Logo (or a different brand of computer). Each brand of peripheral is also different. For example, Logo commands which operate a Pioneer LDV-6000 videodisc player will not work with a Philips videodisc player.

Fortunately, a few general principles can be utilized in connecting Logo to many different kinds of peripherals. For example, once principles underlying serial interfaces are understood, they can be used to connect Logo to many kinds of serial devices: videodisc players, random access slide projectors, color plotters, and speech synthesizers. Similarly, once the principles underlying the analog-to-digital (A/D) interface found on every Apple are understood, they can be employed to connect Logo to joysticks, light sensors, temperature probes, touch windows, Muppet keyboards, and switch inputs. Once general formats in which graphics are stored are known, *Print Shop* pictures, *Min-Pix* from Beagle Brothers, clippings from *Newsroom*, *Mousepaint* pictures, and Koala Pad pictures can all be integrated with Logo drawings. And once the standard American format for storing text (ASCII) is understood (for versions of Logo which provide this option), Logo can be utilized with word processing programs, and Logo procedures can be shared via telecommunications and electronic bulletin boards.

Future Visions

Hardware and memory capabilities are expanding dramatically. The original IBM personal computer introduced in 1982 had 64 kilobytes of standard memory and cost $3000. Today a similar computer with ten times as much memory can be purchased for less than $1000. A megabyte or more of memory is standard on today's Macintosh computers. As a result of its open architecture and hardware slots, the Apple II computer can also be upgraded.

This increased hardware capacity makes it less necessary to omit features in modern versions of Logo which would result in an open architecture. Some Logo designers may be reluctant to include features that they feel will be employed by only five percent of the users. However, Logo must provide room for visionaries as well as for the other 95 percent of the users. At one time Logo itself was deemed an impractical vision.

Call for Papers
Second European Logo Conference

August 30-September 1, 1989

State University Gent - Belgium

The EUROLOGO '89 conference themes are:

- **Classroom experiences with Logo**
- **Logo and the school curriculum (Primary, Secondary and Special education)**
- **Logo projects**
- **Research projects**
- **Teacher Training in the use of Logo**
- **Technical developments**

If you wish to participate in paper presentations, demonstrations, workshops, or special interest groups, contact:

State University of Gent
Department of Education - EDIF
EUROLOGO '89
H. Dunantlaan 1
B9000 Gent Belgium

Testudinal Testimony

Research on Variables, Algebra, and Logo, Part IV: Logo Tools

by Douglas H. Clements

Nancy Roberts, Ricky Carter, Frank Davis, and Wally Feurzeig are constructing a Logo-based pre-algebra course for upper elementary and middle school students with his colleagues (Feurzeig, 1987; Roberts, Carter, Davis, & Feurzeig, 1987). Their initial question was: How much algebra can be taught given the creation of courseware using advanced technology suitable for the sixth grade classroom?

Logo Gossip

In the first component of their curriculum, students are introduced to algebra through Logo projects such as generating gossip, making and breaking secret codes, and writing quizzes. These activities are rich in algebraic ideas, yet their content is meaningful to students. They begin with English, because students have relatively more knowledge of English than mathematical structures and because the English-based activities hold great interest for students. For example, the initial gossip program randomly selects a name from a list of names, an action from a list of actions, and puts the two together into a sentence.

```
TO GOSSIP
 OUTPUT (SENTENCE WHO DOESWHAT)
END

TO WHO
 OUTPUT PICK [SAM JANE SALLY BILL CHRIS]
END

TO DOESWHAT
 OUTPUT PICK [ CHEATS [LOVES TO SING] GIGGLES
       [TALKS YOUR EAR OFF] [LIKES SMELLY
        FEET]]
END

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

If students were to type "repeat 4 [print gossip]" they might see:

jane talks your ears off
sally giggles
bill loves to sing
chris likes smelly feet

Students begin by running, then modifying and extending these simple procedures to create their own gossip. They first alter the name and action lists. Then they might extend gossip to be of the structure "who doeswhat towhom." Such activities illustrate mathematical notions of functions with inputs and outputs and pave the way for a deeper understanding of variables. This leads to simple algebra quiz programs that present problems such as:

5 * BOX + 4 = 49
What is the value of BOX?

First students learn to have the program choose three numbers (e.g, 5, 7, and 4), compute 5 * 7 + 4 to get 49, conceal the 7, and output the problem as above, with 7 known to be the answer. Later, they are led to see there there is a procedure they can use to solve linear equations, initially working with simple problems such as BOX + 3 = 10. Students often work out standard procedures on their own; one student incorporated his knowledge of it within his quiz program:

```
-78 * BOX + +97 = -3023
WHAT IS BOX?
35
IT TOOK YOU 33 SECONDS TO ANSWER ME YOU
KNUCKLEBRAIN   THAT IS SLOW!
WRONG
THE REAL ANSWER IS +40
AN EASY WAY TO GET THE ANSWER IS TO SUBTRACT
+97 FROM -3023 AND THEN TRY TO DIVIDE -78 INTO
-3120
```

Note that the Logo skills required are taught along the way, as an integral component of the project work, and students are only expected to write fairly simple programs.

Logo Function Machines

The second component introduces standard algebraic notation and the manipulation of algebraic expressions through pictorial representations of algebraic objects and manipulations. The first such environment is an icon Logo in which procedures and functions are represented as machines that may have inputs and an output. This model is introduced from the start to illustrate the functional relationships among procedures. For example, the icon Logo for one student's gossip program is shown in Figure 1. These "machines" can be constructed and "run."

Testudinal Testimony — CONTINUED

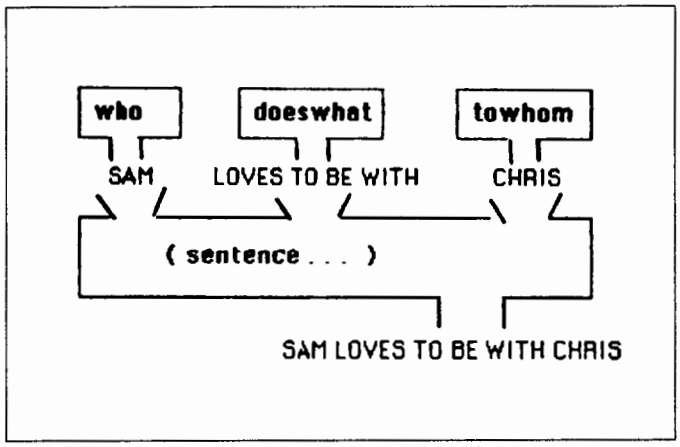


Figure 1

Marbles and Bags

The second environment is a marbles and bags microworld in which bags contain an unknown number of marbles. This is introduced through a well-known guessing game, that might be introduced by the teacher as follows.

Think of a number. Don't tell me what it is, just remember it.

Now double it. Add 3 to that. Subtract 2 from that. Double the result.

Now tell me what you have. 50? Then I say your original number was 12...right?

Students are shown how to do this type of number puzzle by using a notation called marbles and bags. A bag (variable) represents the student's original number. Figure 2 illustrates the Logo program. The top half shows the operations (symbolic, as in the top middle window, or in English—"pick a number," "double," etc.) and operands (numbers from 1 to 10 or "your original number") that can be selected. The illustration is of the problem presented above. The bottom windows show the history of the problem in three formats; from left to right: marble and bag, English, algebra. To solve the problem, one works backwards (see the final window on the bottom); that is, if 4 bags and 2 marbles equal 50 marbles, then two bags and one marble equal 25 marbles, and so on.

The next iconic metaphor is a balance. Students manipulate bags and marbles on the balance to solve problems (e.g., subtracting the same number of marbles from each side and then dividing the bags). Students practice telling and recording marble bag stories using standard notation or whatever combination they prefer (see Figure 3). Here we again see the power of having students experience multiple representations of mathematical ideas and learning to translate among these representation.

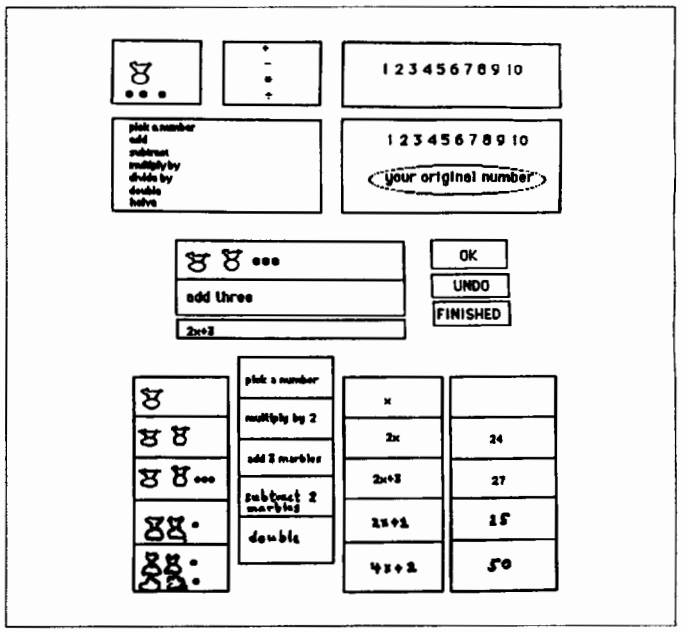


Figure 2

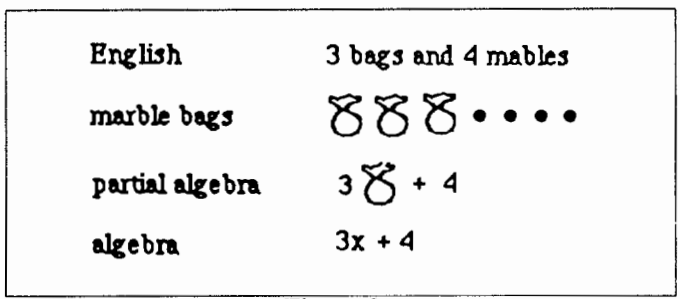


Figure 3

Undoing the Turtle

Other exercises include learning to solve inversion problems through a task with turtle geometry. Students are given a list of commands that direct the turtle along a path (only the first part of the path can be seen). The challenge: Invent a procedure that will bring the turtle back home (i.e., "undo" the series of commands). This, of course, involves constructing a "reverse path" in which each command is the inverse of a corresponding command in the given list. They are then led to see that the process of solving an equation is, similarly, the inverse of the process used in generating it. For example, to generate or "do" the equation 2 * X + 4 = 14, one:

Starts with X,
Multiplies that by 2,
Adds 4 to that, and
Ends with 14.

Therefore, to solve (or "undo") the equation, one should:

Testudinal Testimony — CONTINUED

Start ("unend") with 14,
Subtract ("unadd") 4 from that, to get 10,
Divide ("unmultiply") that by 2 to get 5, and
End ("unstart") with X, so X = 5.

Thus, the procedure for solving simple linear equations follows directly from that already constructed to reverse the turtle's path.

The Algebra Workbench

A final microworld is the Algebra Workbench, an intelligent instructional tool designed to provide a structured algebra exploration environment. It has two parts, a calculator and an expert equation solver. The calculator can be used by the student to solve equations by automatically performing operations on equations or algebraic expressions. The students can perform operations such as multiplication on both sides of an equation, or invoke functions (Do Math, Remove Parentheses, Collect Terms, Expand Terms) on parts of an equation. This allow the student to focus on strategies for solving problems without needing to worry about the arithmetic. It also allows the student to try more than one solution path through a problem, branching at any point along the way (Figure 4).

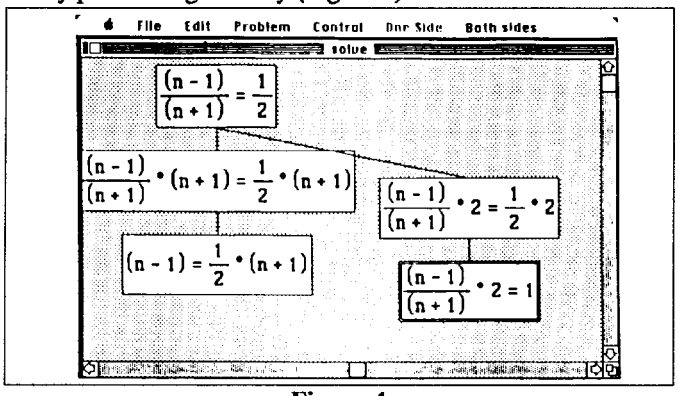


Figure 4

If the student is unable to figure out the next step, he or she can call upon the program's advisor. The advisor suggests one possible next step to the student. It states the goal, then how to reach the goal (Figure 5).

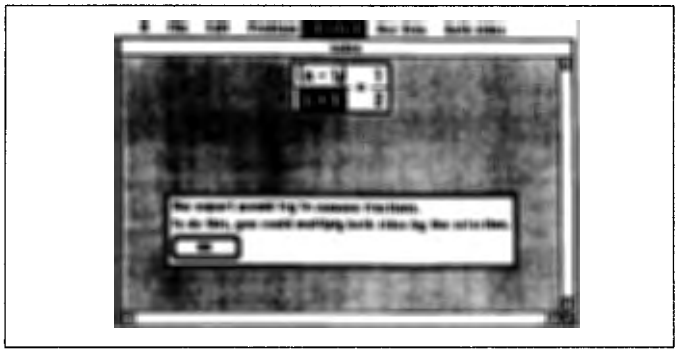


Figure 5

If the problem is entirely new, the student can call on the expert. The expert walks the student through the solution of an entire problem (Figure 6).

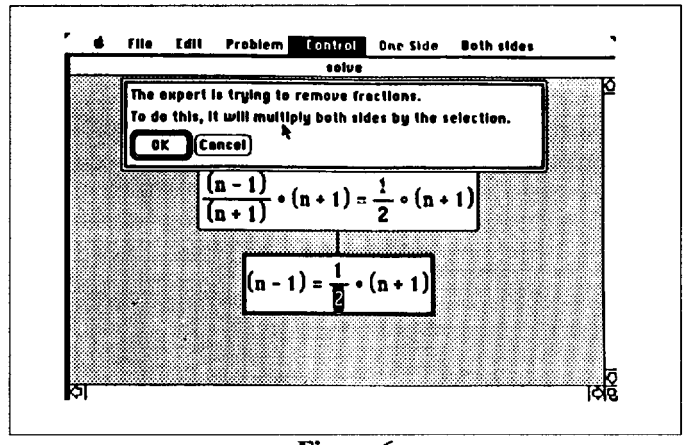


Figure 6

What happened when students used these programs? Six graders did learn algebra concepts in the context of these microworlds. However, they did not demonstrate transfer of the concepts to a more traditional algebra format (Roberts et al., 1987). In addition, it was difficult for the students to learn both Logo and algebra at the same time. When Logo was simplified through the creation of pseudo-primitives, programming became a useful tool for the learning of algebra. This point—that one must find the right level of representation—has been encountered in several previously-reviewed studies.

Such integration of Logo activities into the curriculum holds the potential to turn students away from the mechanical application of calculations toward the active construction of conceptual understandings and semantic connections (via frames) which underlie the ability to solve problems within a subject-matter domain.

[Doug Clements, 401 White Hall, Kent State University, Kent, OH 44242].

References

Feurzeig, W. (1987). Algebra slaves and agents in a Logo-based mathematics curriculum. In R. W. Lawler & M. Yazdani (Eds.), *Artificial Intelligence and Education: Volume One* (pp. 27-54). Norwood, NJ: Ablex.

Roberts, N., Carter, R., Davis, F., & Feurzeig, W. (1987). *Power tools for algebra problem solving: A Lesley College and Bolt Beranek and Newman Project Final Report* (NSF grant MDR—8400328). Cambridge, MA: Lesley College.

MathWorlds

Edited by
A. J. (Sandy) Dawson

In a column a few months ago, Ihor Charischak and Al Cuoco discussed the question of secondary students using Logo to explore mathematical ideas. Since that column appeared I have received a couple of items which illustrate what secondary students can do mathematically if given the opportunity. The first came to me in a traditional printed format and was submitted by a secondary teacher and his son. The second item arrived electronically late one night and was posted to the CLIME forum on our computer at SFU by a high school senior. Both give a flavour of what can be done when secondary students are challenged. Let's look at the electronic contribution first.

Sometime back Ihor had opened a section of the CLIME forum called Logo Problems. He introduced it as follows:

Stanley Burech (St. Clairsville, OH) asks: I would like a procedure comparable to the BASIC primitive SPEED = (No.) in order to control the scroll of the screen other than by Control W. Also I need a program that shall be able to list all the permutations from a set of inputs.

Ihor then challenged: Any one out there willing to tackle these?

About three weeks later the following reply appeared: I am a senior at Woburn High School, in Woburn, MA, USA. My name is Lisa Haverty, and I have been working on a project for some time that is related to the [second] problem you have. I have modified my Logo procedures to generate all the permutations of n consecutive numbers. Here are the procedures. PERMLISTS will take any number as input and output the list of permutations of the list of numbers between one and the input.

```
TO DISPLAY :L
 IF EMPTYP :L [STOP]
 SHOW FIRST :L
 DISPLAY BUTFIRST :L
END

TO PERMLISTS :N
 IF :N = 0 [OUTPUT [[]]]
 IF :N = 1 [OUTPUT [[1]]]
 OUTPUT NEXT :N PERMLISTS :N - 1
END

TO NEXT :N :L
 IF EMPTYP :L [OUTPUT []]
 OUTPUT SENTENCE FINISH :N FIRST :L NEXT :N
    BUTFIRST :L
END

TO FINISH :N :C
 OUTPUT ( SENTENCE ( EXTENDCOMBO :N :C )
          ( LIST LPUT :N :C )
END

TO EXTENDCOMBO :N :C
 OUTPUT EXTENDCOMBO.H :N :C 1
END

TO EXTENDCOMBO.H :N :C :K
 IF :K > COUNT :C [OUTPUT []]
 OUTPUT FPUT SETITEM :K :C
       ( LIST :N ITEM :K :C)
 EXTENDCOMBO.H :N :C :K + 1
END

TO SETITEM :K :L :I
 OUTPUT (SENTENCE (FRONT :K - 1 :L ) :I
    (TAIL ( COUNT :L ) - :K :L ) )
END

TO FRONT :N :S
 IF COUNT :S = :N [OUTPUT :S]
 OUTPUT FRONT :N BL :S
END

TO TAIL :N :S
 IF COUNT :S = :N [OUTPUT :S]
 OUTPUT TAIL :N BUTFIRST :S
END
```

The procedure SETITEM is just some list processing and it may be built into your Logo. (The same is true of FRONT and TAIL) DISPLAY PERMLISTS 3, will output:

```
[3 2 1]
[2 3 1]
[2 1 3]
[3 1 2]
[1 3 2]
[1 2 3]
```

This particular part of the problem I was working on dealt with generating all the possible combinations on an N-button Simplex Lock, a combination lock on which you can use anywhere from zero to N buttons, in any order, and you can push more than one button at a time. The company advertises that there are thousands of combinations on their 5-button lock, but there are actually only 1082.

Thanks for your response to the problem, Lisa!

Well, so much for secondary students not being able to tackle complex mathematical problems. The next submission

came from Robert Hogg, a supervisor of secondary instruction in a northern Alberta town—north of the Olympic site of Calgary, that is. Robert writes: [he called his piece]

Logowriter and Turning on High School Mathematicians

Toward the conclusion of a workshop on student learning styles I discussed with a high school mathematics teacher learning experiences for students. I suggested to him that it would be beneficial to many (if not all) students to actually do something substantial with mathematical content. He agreed, and further suggested that having students do projects would effectively bring this about. However, he went on to say there was no time for this because the content demands of the curriculum are so great.

Working with my son, who was then in grade eleven, we managed to generate the following project which not only "fit" the curriculum, but it was also engaging and challenging for my son. The project was the kind of activity a student might wish to develop in order to explore and demonstrate some of the elements of coordinate geometry.

As programmed in LogoWriter, the computer calls for coordinate inputs for two points A and B, draws the coordinate axes, joins the two points, and bisects the resulting segment, and in doing so gives the coordinates of the point of intersection and calculates the slope. The procedures follow:

```
TO PLOTTER
 CT
 CC
 CG
 PRINT [This procedure draws a segment by
    joining two points on the Cartesian
    coordinate system.]
 PRINT [ ]
 WAIT 60
 PRINT [A (x1 v1) and B (x2 v2) are the
coordinates of the two points.  A
third point C is plotted as the bisector
    of segment AB.]
 REPEAT 6 [PR [ ]]
 WAIT 200
 PRINT [The outputs of the SLOPE and the
    segment BISECTOR (point C) are stated.]
 WAIT 300
 PLOT
END

TO GRID
 MAKE "POS POS
 REPEAT 8 [FORWARD  10 LEFT 90 FORWARD  2
             BACK 4 FORWARD  2 RIGHT 90]
 PU
 SETPOS :POS
 PD
 RIGHT 180
 MAKE "POS POS
 REPEAT 8 [FORWARD  10 LEFT 90 FORWARD  2
             BACK 4 FORWARD 2 RIGHT 90]
 PU
 SETPOS :POS
 PD
 RIGHT 90
 MAKE "POS POS
 REPEAT 12
    [FORWARD 10 LEFT 90 FORWARD 2 BACK 4
     FORWARD 2 RIGHT 90]
 PU
 SETPOS : POS
 PD
 SETH 180
 MAKE "POS POS
 REPEAT 12
    [FORWARD 10 LEFT 90 FORWARD 2 BACK 4
     FORWARD  2 RIGHT 90]
 PU
 SETPOS :POS
 PD
END

TO PLOT
 CC
 CG
 CT
 PRINT [Input values for each of the coordinates]
 PRINT [ ]
 PRINT [ x1 =]
 MAKE "x1 READLIST
 PRINT [ v1 =]
 MAKE "v1 READLIST
 PRINT [ x2 =]
 MAKE "x2 READLIST
 PRINT [ v2 =]
 MAKE "v2 READLIST
 WAIT 60
 CT
 SETC  1
 HT
 PD
 GRID
 SETC 4
 PU
 HT
 SETPOS SENTENCE  :x1  :v1
 PRINT [ POINT A = ]
 PRINT POS
```

MathWorlds — CONTINUED

```
PU
FORWARD  2
PD
LABEL "A
SETPOS SENTENCE :x2  :v2
PRINT  [POINT B = ]
PRINT POS
PU
FORWARD 2
PD
LABEL "A
PRINT [ ]
MAKE "v2 FIRST  :v2
MAKE "v1 FIRST  :v1
MAKE "x2 FIRST  :x2
MAKE "x1 FIRST  :x1
PRINT [ SLOPE =]
PRINT ( :v2 - : v1) / ( :x2 - :x1)
REPEAT 3 [ PR [ ] ]
PU
SETPOS SENTENCE (:x1 + :x2) / (:v1 + :v2) / 2
PD
SETC 2
REPEAT 5 [FORWARD 2 BACK 4 FORWARD 2 RIGHT
  360/5]
PRINT [BISECTOR = ]
PRINT POS
PU
FORWARD 2
PD
LABEL "C
SHOW [to plot another segement with its
  bisector, type CTRL -P]
WHEN "P [PLOT]
END

TO STARTUP
 PLOTTER
END
```

A sample sequence of this program would look like this:

```
This procedure draws a segment by joining two
points on the Cartesian coordinate system.

A (x1,  v1) and B (x2,  v2) are the coordinates
of the two points.  A third point C is plotted as
the bisector of segment AB.

The outputs of the SLOPE and the segment BISECTOR
(point C) are stated.

Input values for each of the coordinates.

x1 = 67.34
v1 = 54
x2 = -45
v2 = -33.33

POINT A = 67.34,  54
POINT B = -45,  -33.33

SLOPE = 0.7774

BISECTOR = 11.17  10.335
```

My thanks to Robert and his son for contributing their work to MathWorlds.

My thanks also to Lisa, and her teacher Al Cuoco, for using the publishing capacities of electronic messaging to speed their contributions to me.

Robert Hogg is Supervisor of Secondary Education for the St. Alberts Protestant Board of Education, St. Albert, Alberta, Canada,T8N 0G4

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

LXionary -- CONTINUED FROM PAGE 9

"The Effects of Logo in the Elementary Classroom: An Analysis of Selected Recent Dissertation Research." by Mary Ann Robinson, Phil Fledman, George E. Uhlig, Education, Summer, 1987.

The authors provide summaries of dissertations completed in 1985 and 1986 which examined the effects of Logo use in elementary classrooms. The findings of 19 dissertations are examined. Among the conclusions,

After five hours of instruction over five weeks, Logo had no significant effect on the cognition of kindergarten students. Among fifth and sixth grade boys, students who completed instruction in problem solving and heuristics using Logo were more effective problem solvers than students who did not receive Logo instruction. No evidence was found that Logo had an impact on the problem solving ability of 36 third through fifth grade students in New Jersey. A group of upper elementary students who received Logo instruction performed significantly better in quantitative reasoning than students with no Logo instruction.

Other studies cited looked at the effect of Logo on attitudes towards mathematics, effects on problem solving skills of learning disabled students, and predictors of Logo success. A number of studies found no statistical evidence of Logo's instructional effectiveness. One study actually found that a group of fourth graders with no Logo experience outperformed an experienced group on a problem solving test.

Continued on page 30

PenPoints

Logo Book Reviews with ASTROLUG

by
Carol Gans, Gayle Lawrence, and Peter Rawitch

"LogoWriter is designed to allow children to learn by doing and exploring."

"Tell them 90 means corner, 180 means reverse or turn back, and 360 means turn all the way around."

Peter: Hi, and welcome back to PenPoints, a column written by ASTROLUG members to review Logo books and materials. ASTROLUG is the Albany-Schenectady-Troy Logo Users' Group in upstate New York. On my left is Carol Gans, an elementary school principal and on my right is Gayle Lawrence, an elementary computer specialist.

Carol: Next to me is Peter Rawitsch, a first grade teacher. Our opening quotations are from the LogoWriter Primary Classroom Materials Kit by Sharnee Chait and Susan Fischer. The kit includes 55 Activity cards. The cards are divided into four sections: Introducing LogoWriter, Turtle Fun, Projects and How To. There is a Scrapbook disk with a tools page and a few additional shapes on the Shape Page. A Teacher's Guide, Reference Guide, keyboard poster and stickers are also included. It is published by Logo Computer Systems, Inc (121 Mt. Vernon Street, Boston, MA 02108, telephone (800) 321-LOGO).

Guidance for Teachers

Gayle: The documentation is comprehensive and instructive. I was pleased to find that the authors had described some of the important tenets of the Logo philosophy in the Overview and that additional comments were included throughout the Teacher's Guide. This definitely sets a positive tone and provides an important foundation for the Activity Cards.

C: The Teacher's Guide and Activity Cards work together nicely. The materials in the Kit can be purchased separately, but I would not get one without the other. The Guide has been helpful to teachers and parent volunteers in my school who are new to LogoWriter. I especially appreciated the options that were suggested in Chapter 2 about utilizing computers in both labs and classrooms.

P: I was delighted to see the inclusion of dramatizations of procedures. The dynamic nature of commands and reporters lends itself to theatrical productions. The authors state that "acting it out makes the (LogoWriter) instruction concrete." More importantly, it makes it personal.

Curriculum Links

P: One of LogoWriter's strengths is its ability to be integrated with many interesting curriculum topics. Earlier versions of Logo were predominantly math-oriented. It is ironic that the kit identifies at least twice as many Math "links" as the ones for Language Arts or Social Studies/Science. This area deserves serious consideration and further development.

G: I don't feel as strongly about the seemingly disproportional number of math links as you do, but I think that some of the named links require me to stretch my imagination pretty far in order to see the connection. It is almost as if they thought up the activity from a Logo standpoint and then searched for a curriculum area to fit it into. I wouldn't mind a 3 to 1 ratio of math (or any other single subject) to the others if I felt the activities really contributed to the students' growth in that particular content area.

C: I basically agree with both of your points, but we have to remember what teachers will feel most comfortable with. I think the Language Arts "links" are adequate and the easiest to self-initiate. I can envision students and teachers making their own "links" to other curriculum areas as they begin to become familiar with the materials.

Primarily Primary or Not?

P: The Activity Cards were "specially designed for beginning readers." The large type, limited text and picture clues enabled my first graders to read most of them. However, primary students are also beginning mathematicians and beginning fine motor coordinators. The Introductory cards include 90 degree turns that are not developmentally appropriate. And while the authors do suggest having students scroll the command center to reduce typing, single key commands for controlling the turtle would be easier (i.e., f instead of f 50).

C: Although they started out as easy, I found the Activity Cards quickly advanced to the level of the Intermediate Kit in that they required a greater knowledge of LogoWriter. I also share your concern about math readiness. I wonder how well the primary students in my school understand the numbers they've used to draw a regular polygon. Sometimes it seems the product has become more important than the process.

G: Exactly! The "exploring" that is mentioned in our opening quotation from the Teacher's Guide is hardly that when the student is given (or selects) an Activity Card with the exact steps for creating a square or triangle listed on it. One of the beauties of Logo is that kids can develop their understandings and intuitions about mathematics without being bound by the formalisms. Opportunities for exploration and the subsequent internalization of some important mathematical notions are lost when a "recipe" is given.

Scrap the Disk

G: One of the first things I wanted as a teacher of first and second graders working in LogoWriter was a quick and easy way for children to name their pages. I remember several of us shared that frustration, and we developed that tool after only a few weeks of trying to work without it. Clearly, a simplified page naming tool should have been provided on the Primary Scrapbook Disk.

P: The Disk does not address some of the other needs of primary students. There is nothing wrong with the one-letter commands with input. However, additional tool pages with other command options should have been available. The commands could reflect the developing cognitive and motor skills of the primary years.

C: I also found this disk disappointing. I expected more from the "Target" game. It only stamps a "target" on the screen for a turtle to reach. I was surprised that it wasn't even interactive. The only things that might be helpful were the eleven new shapes on the Shapes page.

Mixed Review

G: Now, let's recap our reactions to the LogoWriter Primary Classroom Materials Kit. We give it one "penup" for the thoroughness of the Teacher's Guide and two "pendowns" for an underdeveloped Scrapbook Disk and Activity Cards that are not age-appropriate. We feel that calling this a "Primary Kit" is a misnomer.

InLXual Challenges

Recursive Spirals: Time and Again

by Robs Muir

Spirals that grow by fixed ratios are common in nature. Some examples of natural spirals can be found in the growth of conch and nautilus shells, the trajectories of satellites, whirlpools both in liquids and in the air, and in plant and animal growth. As an outgrowth of many early mathematicians' interest in understanding and describing natural phenomena, such spirals have been studied in some detail. Indeed, spirals have been an integral part of the Logo culture–due, in part, to their dynamic and engaging nature.

The geometric spirals that I wish to explore here are commonly called *equiangular spirals*—a name first coined by René Descartes in 1638. Bernoulli was so entranced with these spirals that he had them engraved on his tombstone 1705. In *his* study of these shapes, Bernoulli referred to them as *logarithmic spirals*—a fashionable topic of study for students of mathematics during his day. Because of the proportional increase in size, these spirals are also referred to as *proportional* or *geometric spirals*. We will simply refer to them as SPIs.

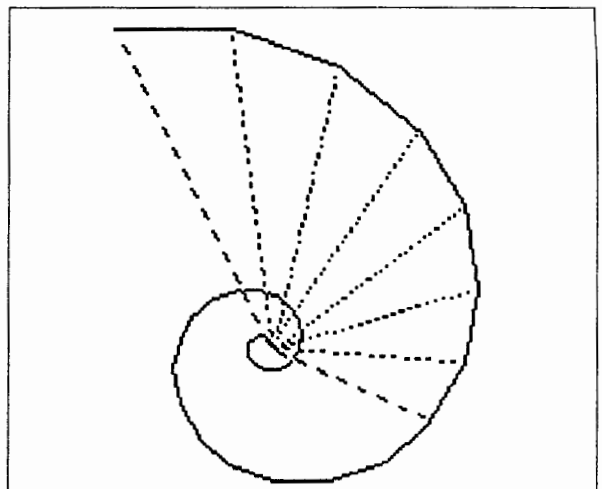
Regardless of the name, these spirals may be observed to be composed of central angles of equal measure. That is, the angle at which the radius cuts the tangent vector at any point is a constant. A simple program (written in Logo) can demonstrate an equiangular spiral. Using simple turtle commands and a recursive procedure, we can build a spiral in which each vertex of the spiral can be connected to the central point. Careful examination will reveal that this spiral is composed of successions of similar triangles, each built upon the previous.

The procedure which was used to generate an equiangular spiral is called EQSPI. It is a recursive procedure in that its final line is self-referential; however, each call to the procedure invokes an instance of EQSPI with a proportionally larger length. In an algorithmic pseudo-computer language, we might describe the steps in this way:

To EQSPI, use three values—a distance, an angle, and a constant
First go forward the distance value
Then turn right the angle value
Finally, EQSPI again using the constant times the distance as the new distance value, the same angle value, and the same constant as before.

In Logo, we would write an EQSPI procedure in this way:

```
TO EQSPI :DISTANCE :ANGLE :FACTOR
  FORWARD :DISTANCE
  RIGHT :ANGLE
  EQSPI (:DISTANCE * :FACTOR) :ANGLE :FACTOR
END
```



EQSPI 5 20 1.1

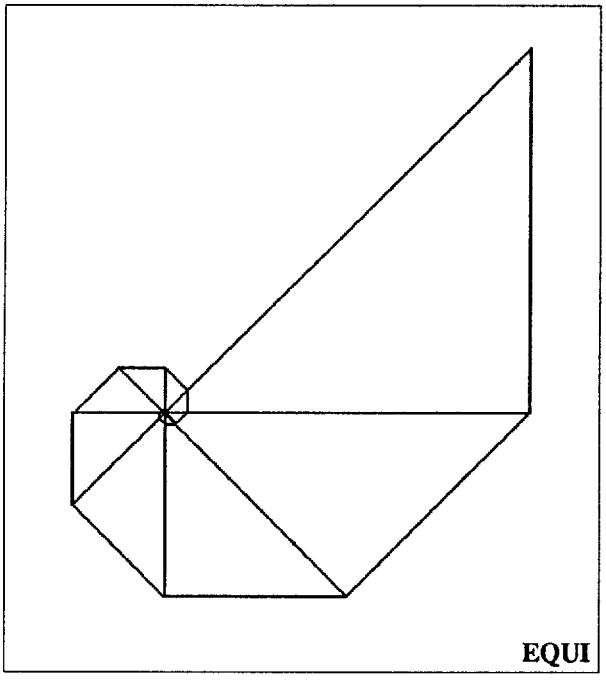
Spirals can be built of regular geometric forms—again, the key is to build successively on the previous section of the spiral using a fixed value which determines the next, proportionally larger shape.

For geometry students, SPIs are a rich area for study. Spirals based on isosceles triangles can be easily generated in Logo if adequate information is specified regarding the angles or sides. As a first challenge, generate a spiral based on 45-45-90 right triangles in which each subsequent triangle has legs equal in length to the previous hypotenuse. Although there are several different approaches a programmer might take in programming a computer to draw triangles, I chose to use the Pythagorean theorem as a starting point. (While this approach is computation-intensive and somewhat inefficient from a computer science standpoint, it does integrate elementary mathematics—always a wise approach for computer educators.)

Given an initial value for one side of a 45-45-90 right isosceles triangle, my procedures calculate the length of the hypotenuse, construct the triangle and then use the length of the hypotenuse as the length of the subsequent leg of the next triangle in the progression. The hypotenuse is computed by the following set of procedures. Notice that I used a hypotenuse procedure which calls explicitly for two distinct sides—clearly unnecessary for isosceles triangles. However, these procedures can later be incorporated into a spiral program for 30-60-90 right triangles.

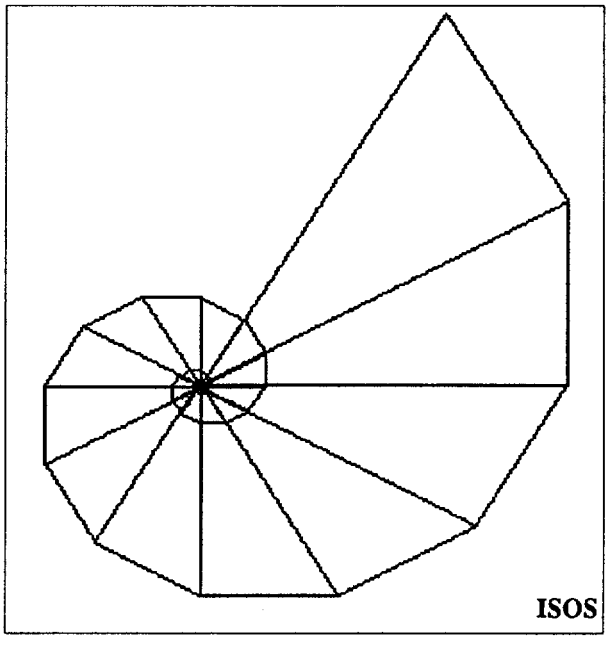
InLXual Challenges — CONTINUED

```
TO HYPOTENUSE :LEG1 :LEG2
 OUTPUT SQRT (SUM.OF.SQUARES :LEG1 :LEG2)
END

TO SUM.OF.SQUARES :X :Y
 OUTPUT SUM (SQUARE :X) (SQUARE :Y)
END

TO SQUARE :INPUT
 OUTPUT :INPUT * :INPUT
END
```



A Challenge:

A similar approach can be used to build a geometric spiral composed of 30-60-90 triangles. Here, the hypotenuse of the previous triangle is used to determine the side opposite the 60 degree angle of the next triangle of the series. Can you build any other solutions which do not use the Pythagorean Theorem (the square root of sum.of.squares)?



After building a successful ISOS using Logo, it is a simple matter to write a procedure, called ISOS.SERIES, which prints a number series that corresponds to the successive lengths of the side opposite one of the angles in these triangular spirals. In other words, this series represents the subsequent segments of an equiangular spiral. Here is the series printed to 16 significant digits. Does this series describe a 45-45-90, or a 30-60-90 triangular spiral?

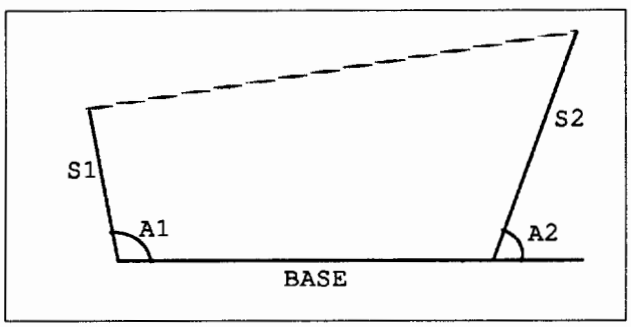
```
ISOS.SERIES 1
0.5773502691896258
0.6666666666666667
0.7698003589195012
0.8888888888888891
1.026400478559335
1.185185185185186
1.368533971412447
1.580246913580247
1.824711961883262
2.106995884773664
2.432949282511017
2.809327846364884
3.243932376681355
3.745770461819846
4.325243168908473
4.994360615759795
5.766990891877965
6.659147487679727
7.689321189170621
8.878863316906303
10.2524282522275
11.83848442254174
13.66990433630333
15.78464589672232
18.22653911507111
21.04619452896309
```

InLXual Challenges — CONTINUED

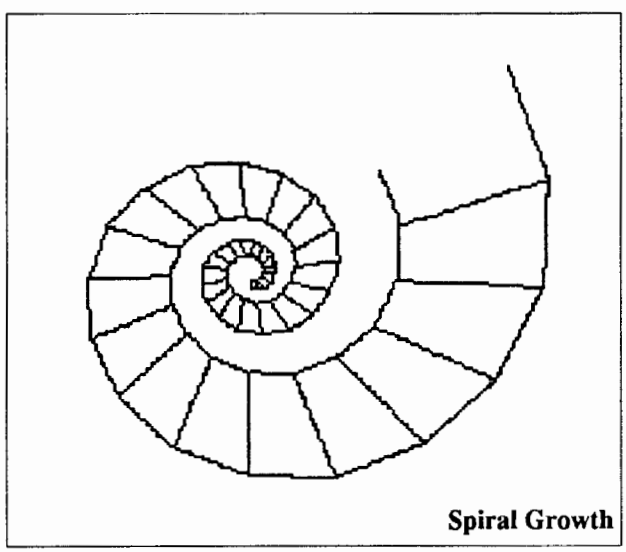
Compound Spirals

Spiral growth can be demonstrated with shapes other than triangles. Quadrilaterals, as well as other more complex shapes, can help illustrate growth patterns of organic forms such a the nautilus. Many such sea shells are formed because its occupant must build progressively larger chambers to house its progressively larger body. A classic treatment of this can be found in Abelson and diSessa's *Turtle Geometry* (MIT Press, 1981)

A multi-purpose procedure called SPIRAL.GROWTH can build quadrilateral spirals with any configuration of a initial quadrilateral with the following information specified.



How would you build a recursive procedure which requires five inputs—the BASE, A1, A2, S1, and S2—to generate spirals composed of quadrilaterals? Are there any quadrilaterals that will not make *good* spirals? What are the restrictions?



Spiral Growth

A Final (well...at least another) Challenge

Notice that the following Logo statement will cause the turtle to orbit around some point on the screen.

```
REPEAT A.ZILLION.TIMES [FORWARD 1 RIGHT 1]
```

(For you hard-core Logoists, write an operation called `A.ZILLION.TIMES` that will output the really huge value for the `REPEAT` command. For the rest of us, we will be content to use `REPEAT 9999 [FORWARD 1 RIGHT 1].`) As the turtle orbits, we notice that the turtle takes some finite time to complete a revolution, and that this amount of time (called the period) corresponds to the distance the turtle is from the center. The farther away, the longer the time required. Some students could measure this with a stop watch, or others could write a procedure to output a measure of the time required for the turtle to make one complete revolution, e.g., when `HEADING = 0`.

What does this mean for spirals? Unlike the circle, as the spiral continues to grow the turtle will require progressively longer and longer amounts of time to complete a revolution. How would the period increase as the spiral itself increased? Write a program in which the turtle spirals outward with a *constant speed* and compute the time required to each revolution. Why can't you use EQSPI as the starting point for this problem?

Is the number series that you determine for the increasing period of revolution recognizable? Is it a Fibonacci series, a logarithmic progression, multiples of 2π, or what? I leave this as an exercise for the reader, or a challenge for the doer.

[Robs Muir, 1688 Denver Ave., Claremont, CA 91711; CompuServe 70357, 3403; BITnet MUIRR@CLARGRAD].

Stager's Stuff

One Turtle — One Vote (or Papert for President?)

by Gary S. Stager

It's presidential election time again in the United States and the electoral system can be quite confusing even for many adults. In this and next month's article I will share ideas for using Logo to create an entire classroom curriculum unit on the elections.

No election would be complete without a voting booth for casting your ballot. This article features the code for a very flexible Logo voting booth. The need for flexibility and ease-of-use demanded slightly more memory than 64K. The program was written in LogoWriter 2.0 (the new LogoWriter ProDos upgrade for 128K machines) and runs beautifully under this configuration. You should be able to easily adapt this program for LCSI's Logo II, IBM Logo, or Terrapin Logo 128K with little difficulty. I've tried many approaches to make this program run in 64K versions of LogoWriter (Apple, IBM, Commodore) without success. I have, however, created a scaled-down, less flashy, voting booth program for 64K versions of LogoWriter and will mail you the code if you send me a self-addressed stamped envelope. (I'll also mail a disk containing the code for 128K versions if you don't want to enter the code contained in this article.)

As I write this column, the candidates in the 1988 Presidential Primary are changing daily. For this reason and others, I wanted to create a voting booth program which would be as flexible as possible. The result of this effort is a program which can be used for presidential or student council elections, for primaries and general elections. My voting booth program allows you to specify any number of different parties and candidates (as memory permits).

There is some rather sophisticated list processing and recursion in this program. Do not be dismayed by the complexity of the code. My goal is to provide teachers with a flexible tool for engaging children in the political process, not to teach recursive operations. If you can learn some new programming techniques by analyzing my procedures, more power to you. You or your students have to know very little Logo to use the program.

The voting booth program contains many procedures and takes up much of this column. Next month, I will provide you and your students with tools for graphing the results of your elections and a referendum ballot which will allow you to vote yes or no on questions entered by the teacher.

A Place to Start

The act of voting is only a small step towards understanding the political process. How about enhancing your election unit by using Logo to:

1) Create animated campaign commercials for each candidate;
2) Write reports on past elections, candidates, campaign issues, or past winners;
3) Create demographic bar graphs, pictographs, or pie charts;
4) Printout Logo campaign posters;
5) Draw maps; and
6) Print voter registration forms.

To setup the voting booth, the teacher or student in charge should type:

```
OPEN.BOOTH

Enter the name of each party in this
election. Type DONE to stop entering data.
1DEMOCRAT
2REPUBLICAN
3DONE
```

You may enter any number of different parties (memory permitting).

```
Enter the name of each candidate and press
RETURN. When you are finished entering
candidates, type: DONE
DEMOCRAT  JESSE JACKSON
DEMOCRAT  MICHAEL DUKAKIS
DEMOCRAT  ALBERT GORE
DEMOCRAT  PAUL SIMON
DEMOCRAT  DONE
REPUBLICAN  GEORGE BUSH
REPUBLICAN  ROBERT DOLE
REPUBLICAN  PAT ROBERTSON
REPUBLICAN  DONE

WHICH PARTY TICKET DO YOU WISH TO VOTE ON
 DEMOCRAT REPUBLICAN ?
DEMOCRAT
```

To choose a particular candidate, press any key other than RETURN. When you see the name of the candidate you wish to vote for, press RETURN to cast your vote. Don't worry if you

Stager's Stuff — CONTINUED

accidentally skip your choice - the candidates will wrap to simulate the interface in a real election booth (the names stay up there and don't disappear).

After you have set up the candidates and parties by executing the OPEN.BOOTH procedure you do not need to re-initialize the candidate and party variables. As a matter of fact, you do not want to type OPEN.BOOTH or any of its subprocedures (including SETUP and its subprocedures) *unless you deliberately wish to erase all of the election data.*

Once the voting machine is setup each voter only needs to type: VOTE to cast his/her ballot. You may wish to type:

REPEAT *(# of students who will be voting - 1)* [VOTE]

Even in a general election when there is only one party or one candidate per party, you must declare your party affiliation by typing the correct party name.

Type TOTALS to see the results of the election. TOTALS and it's subprocedures sort the candidates from the one with the highest number of votes to the one with the lowest number of votes in each party. TOTALS can be typed at any time as long as you don't wish the election results to be a secret.

TOTALS

```
                DEMOCRAT
- - - - - - - - - - - - - - - - - - - - -

JESSE JACKSON 4
MICHAEL DUKAKIS 2
PAUL SIMON 1
RICHARD GEPHARDT 0
ALBERT GORE 0
GARY HART 0

                REPUBLICAN
- - - - - - - - - - - - - - - - - - - -

ROBERT DOLE 4
PAT ROBERTSON 2
GEORGE BUSH 1
NONE OF THE ABOVE 73
```

The LogoWriter 2.0 Code

```
TO OPEN.BOOTH
 SETUP
 VOTE
END

TO VOTE
 SELECT.PARTY
END

TO SETUP
 CLEARNAMES
 SETUP.PARTIES
 INIT.PARTIES :PARTIES
 CANDIDATE.INFO
 SETUP.CANDIDATES :PARTIES
 MAKE.CANDIDATES :PARTIES
END

TO SETUP.PARTIES
 CC
 TYPE (SENTENCE [Enter the name of each party
      in this election. Press DONE to stop
      entering data.] char 13)
 MAKE "PARTIES GET.INFO [] 1
END

TO GET.INFO :LIST :NUMBER
 TYPE :NUMBER
 MAKE "IT READWORD
 IF :IT = "DONE [OUTPUT :LIST]
 OUTPUT GET.INFO (LPUT :IT :LIST)
      :NUMBER + 1
END

TO GET.INFO.2 :LIST :LABEL
 TYPE (SENTENCE :LABEL CHAR 32)
 MAKE "IT READLISTCC
 IF :IT = [DONE] [OUTPUT :LIST]
 OUTPUT GET.INFO.2 LPUT :IT :LIST :LABEL
END

TO INIT.PARTIES :PARTY.LIST
 IF EMPTY? :PARTY.LIST [STOP]
 MAKE FIRST :PARTY.LIST []
 INIT.PARTIES BUTFIRST :PARTY.LIST
END

TO CANDIDATE.INFO
```

Stager's Stuff — CONTINUED

```
 CC
 TYPE SENTENCE [Enter the name of each
      candidate and press RETURN.
      When you are finished entering candi
      dates, type: DONE] char 13
END

TO SETUP.CANDIDATES :PARTY.LIST
 IF EMPTY? :PARTY.LIST [STOP]
 MAKE FIRST :PARTY.LIST GET.INFO.2 [] FIRST
      :PARTY.LIST
 SETUP.CANDIDATES BUTFIRST :PARTY.LIST
END

TO MAKE.CANDIDATES :PARTY.LIST
 IF EMPTY? :PARTY.LIST [STOP]
 INIT.CANDIDATES THING FIRST :PARTY.LIST
 MAKE.CANDIDATES BUTFIRST :PARTY.LIST
END

TO INIT.CANDIDATES :PARTY
 IF EMPTY? :PARTY [STOP]
 MAKE UNPARSE FIRST :PARTY (LIST FIRST
      :PARTY 0)
 INIT.CANDIDATES BUTFIRST :PARTY
END

TO SELECT.PARTY
 CC
 TYPE (SENTENCE [WHICH PARTY TICKET DO YOU
      WISH TO VOTE ON]
      CHAR 13 :PARTIES "? CHAR 13)
 MAKE "IT READWORD
 IF MEMBER? :IT :PARTIES [VOTING.BOOTH THING
      :IT STOP]
 SELECT.PARTY
END

TO VOTING.BOOTH :CANDIDATES
 CC
 TYPE SENTENCE [PRESS ANY KEY TO SKIP TO THE
      NEXT CANDIDATE.
      WHEN THE CANDIDATE YOU WANT TO VOTE FOR
      IS ON THE SCREEN -
      PRESS RETURN] char 13
 HT
 CT
 MAKE "CHOICE UNPARSE VOTER :CANDIDATES
 IF EMPTY? :CHOICE [VOTING.BOOTH :CANDI
      DATES]
 ADD.VOTE :CHOICE
END

TO VOTER :CANDIDATES
 IF EMPTY? :CANDIDATES [OUTPUT "]
 PR SENTENCE FIRST :CANDIDATES "?
 IF READCHAR = CHAR 13  [OUTPUT FIRST
      :CANDIDATES]
 OUTPUT VOTER BUTFIRST :CANDIDATES
END

TO ADD.VOTE :CANDIDATE
 MAKE :CANDIDATE SENTENCE (BUTLAST THING
      :CANDIDATE)
      (LAST THING :CANDIDATE) + 1
END

TO SORT.CANDIDATES :PARTIES
 IF EMPTY? :PARTIES [STOP]
 MAKE (WORD FIRST :PARTIES ".S) SORT CRUNCH
      THING FIRST :PARTIES []
 SORT.CANDIDATES BUTFIRST :PARTIES
END

TO TOTALS
 HT
 CT
 SORT.CANDIDATES :PARTIES
 SHOW.TOTALS :PARTIES
END

TO SHOW.TOTALS :LIST
 IF EMPTY? :LIST [STOP]
 PR []
 TAB TAB TAB PR FIRST :LIST
 PR SENTENCE [----------] CHAR 13
 TALLY THING (WORD FIRST :LIST ".S)
 SHOW.TOTALS BUTFIRST :LIST
END

TO TALLY :LIST
 IF EMPTY? :LIST [STOP]
 PR (SENTENCE FIRST LAST :LIST LAST LAST
      :LIST)
 TALLY BUTLAST :LIST
END

TO UNPARSE :LIST
 OUTPUT U.P. :LIST "
END

TO U.P. :LIST :WORD
 IF EMPTY? :LIST [OUTPUT :WORD]
 IF EQUAL? LAST :LIST CHAR 32 [OUTPUT U.P.
      BUTLAST :LIST :WORD]
```

Stager's Stuff — CONTINUED

```
 OUTPUT U.P. BUTLAST :LIST WORD (LAST :LIST)
      :WORD
END

TO READWORD
 OUTPUT FIRST READLISTCC
END

TO CRUNCH :LIST :NEW.LIST
 IF EMPTY? :LIST [OUTPUT :NEW.LIST]
 OUTPUT CRUNCH BUTFIRST :LIST (LPUT THING
      UNPARSE FIRST :LIST :NEW.LIST)
END

TO SORT :LIST
 IF (COUNT :LIST) < 2 [OUTPUT :LIST]
 OUTPUT SORT1 (FIRST :LIST)
      (BUTFIRST :LIST) []
END

TO SORT1 :MIN :IN :OUT
 IF EMPTY? :IN [OUTPUT FPUT :MIN SORT :OUT]
 IF (LAST :MIN) < (LAST FIRST :IN)
      [OUTPUT SORT1 :MIN (BUTFIRST :IN)
      (FPUT FIRST :IN :OUT)]
 OUTPUT SORT1 (FIRST :IN)
      (BUTFIRST :IN) (FPUT :MIN :OUT)
END
```

Note: If you accidentally press the ESC key, you can return to your ELECTION page and continue running the VOTE or TOTALS procedures. This is because LogoWriter keeps global variables in memory until you CLEARNAMES or turn off the computer. If however, you wish to vote in the same election over several sessions, you will be required to store the variable names and associated values in a procedure to be run when you start your next session and before you run the VOTE procedure.

One way to store these variables is to flip the page, move the cursor to the bottom of the flip-side, go down into the command center (using the DOWN key) and type SHOWNAMES. Next use the SELECT keys to highlight the text created by SHOWNAMES. Use the CUT keys to put a copy of the text on the clipboard, press UP keys and use the PASTE keys to put it at the bottom of the flip side of the page. Then you need to create a procedure to hold your variables as shown below: You need to change lines like:

```
:DEMOCRATS is [[JESSE JACKSON]]     becomes

MAKE "DEMOCRATS [[JESSE JACKSON]]
```

Thus, you will have a procedure that looks like this:

```
TO VARIABLES
 MAKE "DEMOCRATS [[JESSE JACKSON]]
      .
      .
      .
END
```

Next month, the final column of this year, will feature the "referendum machine" and tools for graphing the results of the voting booth. Until then, Happy Ballot Stuffing!

Note the ideas on sorting data came from Brian Harvey's *Computer Science Logo Style: Advanced Projects - Volume 3.* Cambridge: MIT Press, 1987.

[Gary S. Stager is the Director of Training for the Network for Action in Microcomputer Education and can be reached at: 5 Eastside Avenue - 2F, Wanaque, NJ 07465, ph: 201/633-3121, CIS: 73306,2446, Applelink: K0331].

LXionary -- CONTINUED FROM PAGE 22

In summarizing the results, the authors do concede that

> ...these studies suggest that learning Logo is usually no more effective in effecting the dependent variable than non-Logo computer based instruction or non-computer instruction.

While they believe that the claims of Logo's value may be exaggerated, the difficulty in measuring gains in areas such as problem solving is discussed.

The good news here is that many people are interested in providing research-based evidence about the instructional effects of Logo. Those of us who have been using and encouraging the use of Logo hope that this trend continues and that the findings will begin to provide ammunition to our claims about Logo's benefits.

[Bill Craig, Hening Elementary School, 5230 Chicora Drive, Richmond, VA 23234].

LogoPals

by Barbara Randolph

It's fun! It's fascinating! Building with Lego blocks and gears, sensors and motors for the first time this year provided a unique combination of Logo and learning for me. Once I began constructing, it was spellbinding—first, a flashing stoplight; next, a car that had sensors on both ends so tapping either end set it moving in the opposite direction; and then, a merry-go-round with whirling riders. A turtle robot with a pen for drawing shapes really proved delightful. Imagine—a turtle built with one's very own hands! Granted they are only miniature versions of real-life objects but it felt as enriching an experience as if their lifesized counterparts were actually being created. Now I see why girls and boys (and teachers) become so involved with designing, probing, puzzling over and redesigning projects in the "Land of Lego and Logo."

LogoPals, as you may know, is also a wonderful opportunity for learning. It is a network of Logo student penpals from around the world. Here are some new boys and girls who would like to write to others learning Logo:

Kyle Stock and Jennifer Long (Madison, Connecticut, USA): We are ten and eleven years old. Our hobbies are hunting, sports and horseback riding. We would like penpal(s) from Tasmania, Japan, Nigeria or Alaska.

Devon Adams (Seattle, Washington, USA): Send me a penpal please. I am eight and a half years old. My favorite sports are football, baseball, volleyball, softball, basketball and on and on!

Michelle Meltzer (New York, New York, USA): I like to collect seaglass and seashells. My hobbies are swimming, tennis, sailing and biking. Please send me a penpal.

Amit Govil and Brandon Sprague (Madison, Connecticut, USA): My name is Amit and I am ten years old. I like soccer, reading, collecting coins and old artifacts. My partner is Brandon Sprague. Send us a penpal from Tasmania or Alaska.

Jordan Kean (New York, New York, USA): I live in Manhattan. My hobby is English style riding and golf. I would like a penpal.

Kelly Ward (Seattle, Washington, USA): I am eight years old. I am a third grader. My favorite hobby is playing the piano. I really like to play paddleball also. I would like a Logo penpal.

Nina Levenduski (Madison, Connecticut, USA): I like reading. I like to ride horses. I am 11 years old and in fifth grade. Please send me a penpal from Alaska.

Alexis Wolff (New York, New York, USA): My hobbies are tennis and horseback riding. I am ten years old. I would like a penpal from Ireland.

Shane Skriletz and Trevor Hanger (Madison, Connecticut, USA): Our hobbies are ice and floor hockey, soccer and drawing. We are both eleven years old and in the fifth grade. We would like to write to someone in Tasmania or Alaska (or somewhere above the Arctic Circle).

New LogoPal "ambassadors" to thank are Anne Beavers, a math and computer teacher, from the Country School from Madison, Connecticut, USA, and Mr. Harris, a computer teacher from the Lawton School in Seattle, Washington, USA. The support and encouragement which they give to their Logo students is tremendously appreciated. Thank you both!

How can Logo students become Logo penpals? Have them write to me, telling their age and grade, hobbies and interests, their favorite Logo activities and the name of their Logo teacher. Also it would help to know during which months of the year they attend school. We will match them with the students listed in this column or with others in the LogoPal network. Which ages can participate? Their ages can range from primary through high school levels. Their skill level can vary from beginner to advanced. (Requests for penpals from certain places will be accommodated when possible.)

Children in the USA need to send a self-addressed envelope with their letters. Those outside the USA should enclose international postal coupons (purchased at the post office) for a 1-ounce or 28-gram reply. Some teachers save on postage by mailing a group of their students' letters in one large envelope.

P.S. Thank you to the teachers from Finland, Singapore, New York and Maryland who have written to inquire about LogoPals—we hope your students will become part of our penpal network. As you can see, our boys and girls want to write to others all over the USA and the world!

[Barbara Randolph is a library and instructional media center teacher in the Chicago Public Schools].

Logo in the USSR

by Tom Lough

Is the Soviet Logo turtle coming out of its shell? Maybe.

Last summer, I worked for five weeks in Moscow as a member of the "Information USA" staff. As the educational computing specialist for the first exhibition of the current cultural exchange between the USA and the USSR, I was to help the Soviet visitors understand how Americans used computers and other technology in their homes, schools, and businesses as part of their everyday life.

Of the more than 7000 daily visitors to the exhibit, I was able to meet scores of teachers, programmers, teacher trainers, students, and other education professionals. I demonstrated a wide variety of software, including MacPaint, One on One, Blockers and Finders, The Other Side [an interesting experience!], and , of course, Logo.

Most of the teachers I worked with were from either the secondary or the college level. Many of them had heard of Logo, but dismissed it as a child's computer language. So I took it upon myself to help them understand what it was really all about.

Instead of starting with the "cherepaka" [Russian for "turtle"], I showed them some activities using words and lists. I usually set up a WHO and WHAT procedure similar to the ones below. They followed the explanation and development of the procedures with interest. [I was assisted by a young American guide who was fluent in Russian.]

```
TO WHO
OUTPUT  PICK PERSON
END

TO PERSON
OUTPUT [ MISHA  ALEX  BORIS  EKATERINA
         ANNA  IRINA ]
END

TO WHAT
OUTPUT PICK VERB
END

TO VERB
OUTPUT [ LIKES  LOVES  HELPS  SEES
            [TALKS TO]  ]
END
```

If they did not understand the PICK procedure, I asked them just to accept that it selected an item at random.

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

When I demonstrated how the procedures could work with PRINT and SENTENCE, they reacted strongly and excitedly. Even "hardened" systems programmers were intrigued. Naturally, certain of the sentences provokes some animated comments among the visitors!!

REPEAT 10 [PRINT (SENTENCE WHO WHAT WHO)]

```
ALEX TALKS TO IRINA
ANNA LIKES ANNA
BORIS LOVES MISHA
BORIS TALKS TO IRINA
EKATERINA SEES EKATERINA
EKATERINA HELPS ANNA
ALEX LOVES MISHA
EKATERINA LOVES ALEX
IRINA TALKS TO BORIS
MISHA SEES BORIS
ANNA LIKES MISHA
```

Then I showed some of the calculational features, and did a little "number crunching" with some of the trigonometric functions.

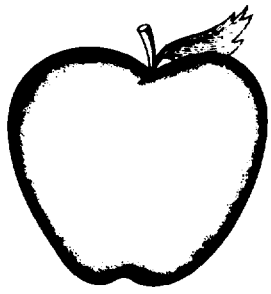
Finally, I informed them that you could also produce graphics. Enter the cherepaka.

After SQUARE, TRIANGLE, HOUSE, we had procedures called LEONID, LIDIA, and IGOR. Things really got interesting when I began reconfiguring the language with

```
TO NALEVO  :UGOL
LEFT  :UGOL
END

TO NAPRAVO   :UGOL
RIGHT  :UGOL
END
```

As you can imagine, by this point, the Soviet visitors were entranced. They began to talk about how their students could use a language like this. I was encouraged to hear this, because I did not find any indications of Logo use.

I hope that Soviet educators will be willing to investigate the potential for Logo in their system. I heard that Seymour Papert and Silvia Weir may have taken a trip to the USSR recently for this purpose. Then maybe the cherepaka will come out of its shell and join the worldwide Logo party!



AppleWorks for Educators-
A Beginner's Workbook

AppleWorks for Educators guides the beginning *AppleWorks* user through word processing, data base and spreadsheet management. Four complete sections provide step-by-step instructions, explanations and true-life examples for educators, and a data disk is included in each workbook. Over 200 pages of the most practical and useful information you'll find for learning *AppleWorks*. Excellent manual for workshops or individual learning.

Written by Linda Rathje. 8.5" x 11", laser-printed workbook and disk.

ClassWorks-
AppleWorks for the Classroom

This new, comprehensive package provides all a teacher needs to introduce *AppleWorks* to students. It includes detailed lesson plans, worksheets and quizzes, student and teacher data disks, and 54 overhead transparency masters with annotations for group instruction.

You need only one *ClassWorks* package for an entire class, or even an entire school!* Laser printed worksheets and student data disks are all ready to duplicate. All worksheets and quizzes are include on the teacher's *AppleWorks* data disk. Teachers are free to modify the materials and then duplicate them for student use.

ClassWorks was developed for middle school and high school. Use it as part of a Computer Literacy or Computer Applications course, or to introduce *AppleWorks* in content area classes. *ClassWorks* serves as an excellent supplement for other *Appleworks* materials!

* The purchase price includes a site license for one school building. It is not permissible to reproduce files or printed materials from *ClassWorks* for use outside of that school.

Written by Rick Thomas. 8.5" x 11", 180 pages plus disk.

Name _____

Address _____

City/State _____

Postal Code _____

Phone _____

AppleWorks

#copies	Price(US)	Shipping
1	$18.00	$2.50 (US)
2	18.00	3.50 (US)
3-4	18.00	5% of total
5-9	16.20	5% of total

Call for prices and shipping on large orders.

ClassWorks

#copies	Price(US)	Shipping
1	$39.95	$3.50 (US)
2-4	39.95	5% of total

Call for prices and shipping on large orders.

___ copies of AppleWorks
@ _____ ea = _____

___ copies of ClassWorks
@ _____ ea = _____

+ shipping rate + _____

+ 3% if mailed to P.O. Box, Alaska, Canada, Hawaii or outside U.S. + _____

+ $2.50 for billed orders + _____

TOTAL _____

☐ Payment enclosed. (US funds drawn on US bank)

☐ Bill me (add $2.50 for handling)

Mail to: **ICCE**
University of Oregon
1787 Agate St.
Eugene, OR 97403

NECC '89 Connections

9th National Educational Computing Conference
Invites you to attend

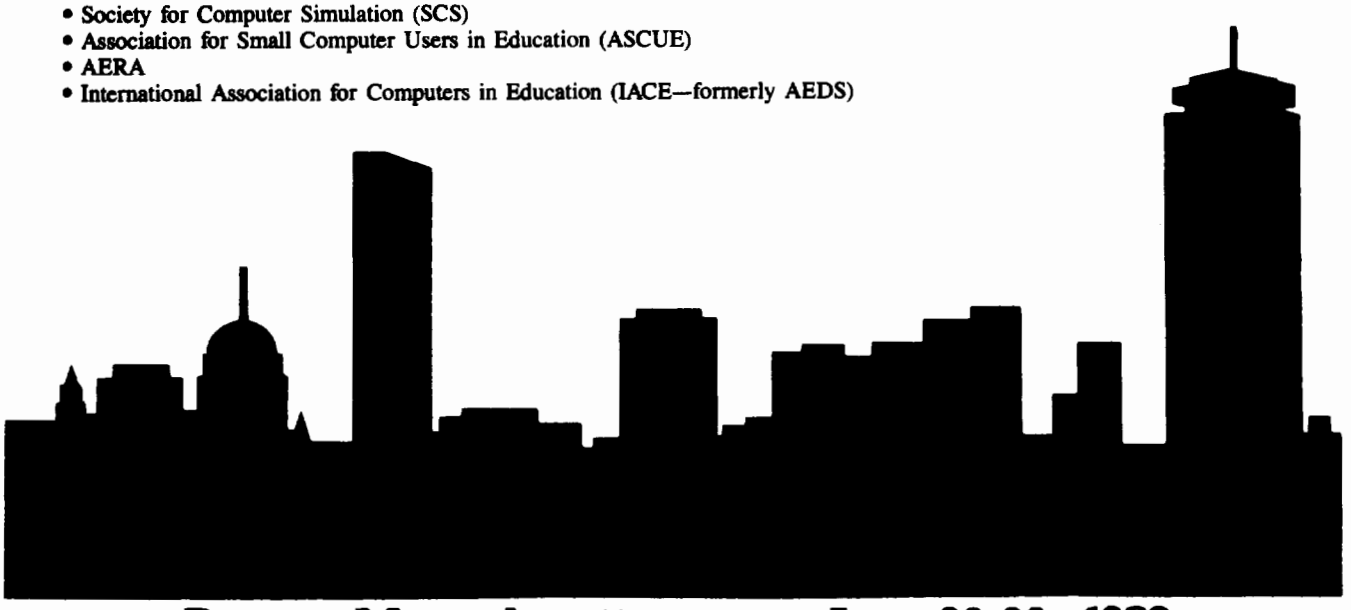
Papers, Panels, Projects, Exhibits, Full-Day Pre- Conference Workshops

For information contact:
Susan Friel and Nancy Roberts
Lesley College
29 Everett Street
Cambridge, MA 02138-2790

For information on Exhibits contact:
Paul Katz
NECC '89 Exhibits
University of Oregon
Continuation Center
1553 Moss Street
Eugene, OR 97403-9905

NECC '89 is sponsored by the International Council for Computers in Education (ICCE) in cooperation with member organizations of the NECC Steering Committee:

- Association for Computers and the Humanities (ACH)
- Association for Computing Machinery (ACM) Special Interest Groups on:
  - Computer Science Education (SIGCSE)
  - Computer Uses in Education (SIGCUE)
  - University and College Computing Services (SIGUCCS)
- Educational Computing at Minority Institutions (ECMI)
- IEEE Computer Society
- EDUCOM/EDUNET
- Society for Computer Simulation (SCS)
- Association for Small Computer Users in Education (ASCUE)
- AERA
- International Association for Computers in Education (IACE—formerly AEDS)



Boston, Massachusetts **June 20-22, 1989**