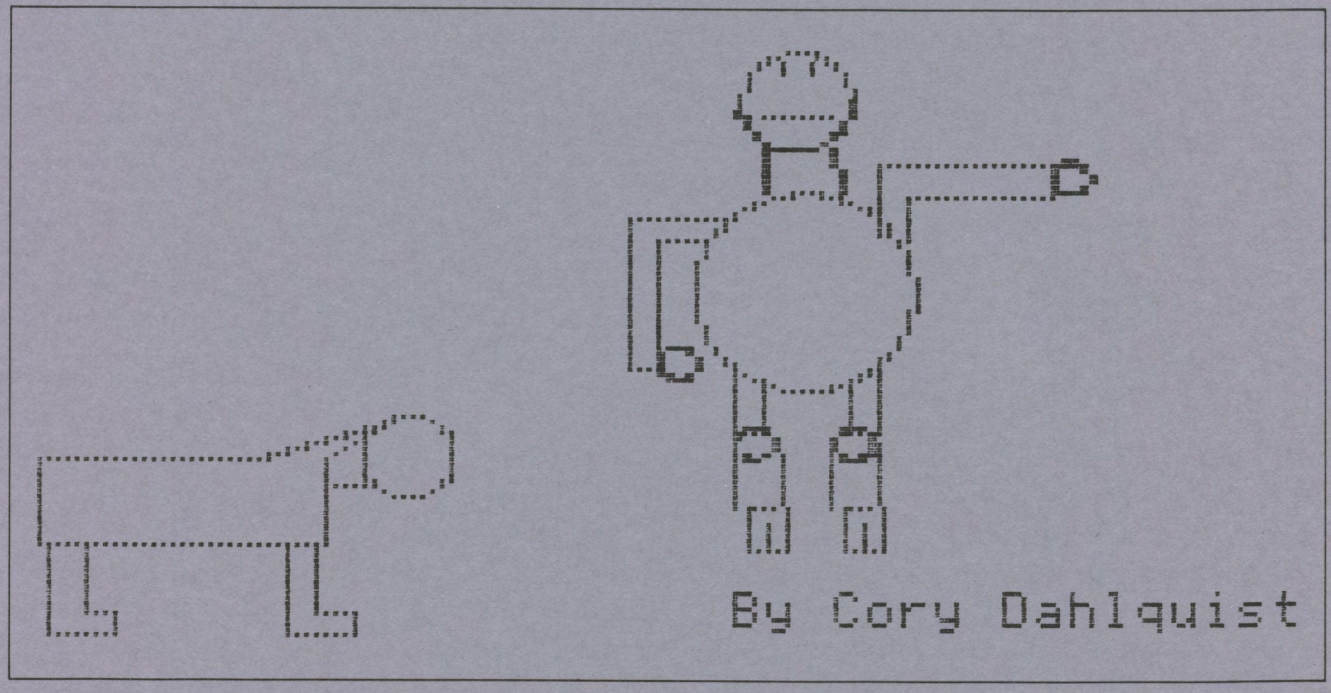
Journal of the ICCE Special Interest Group for Logo-Using Educators

LOGO EXCHANGE

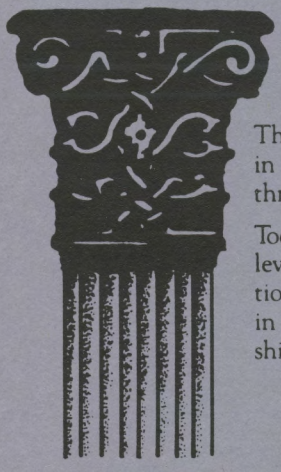
APRIL 1989 VOLUME 7 NUMBER 8



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 12,000 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!

INTERNATIONAL COUNCIL FOR COMPUTERS IN EDUCATION
ICCE

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

LOGO EXCHANGE

Volume 7 Number 8 Journal of the ICCE Special Interest Group for Logo-Using Educators April 1989

Publisher
International Council for Computers in Education

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

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

Contents

ICCE Membership (includes *The Computing Teacher*)

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

SIGLogo Membership (includes *The Logo Exchange*)

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

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

From the Editor

Reflections and Impressions

Have you ever gone to a computers in education conference? How about a Logo conference? If you answered "yes" to either of these questions, then you understand the sense of exiliration and exhaustion that I am feeling after two days at the West Coast Hands On Technology Conference, affectionately known as the West Coast Almost Logo Conference. Ahtough I left Eugene on Thursday afternoon and returned on Sunday morning, I felt as if I were gone for a week!

As I am slowly recovering from my brief stay in L. A., I have been asking myself, why do we go to these conferences? Why do we put ourselves through these exhausting experiences only to return to a desk full of mail to answer and phone calls to return? For each of us I'm sure the reasons are different, but I suspect there are three factors thay play a role in everyone's decision go a-conferencing.

People

We centainly go to conferences because of the people that are there. Perhaps we go to hear "famous" speakers; perhaps to see old friends, I certainly know that people are an important part of any conference for me.

This conference, like any Logo conference, was rich in opportunities to interact with old friends and make new ones. Come with me as I share some of my people impressions...

Even before I opened the door of the Los Angeles Airport Hilton (where this conference has been held before), I was flooded with warm memories from past conferences. I no more than set foot through the door when I heard a familar "Hello." Michael Tempel waved his greetings. As I stopped to chat, I realized that I was surrounded by Logophiles; my intended brief stop turned into many delightful interactions.

Eventually I managed to head towards the hotel registration desk, only to encounter Tom Lough. Tom kept me company while I stood in line. Since the exhibit area was closing down (at 10:00 at night, no less) weary souls began appearing from all over — there seemed no end to the familiar faces. Finally I made it to my room where I collapsed. After all, I had to be ready for a presentation early in the morning!

My memory of the next two days is a blur. It was nonstop talk and non- stop sharing. A particular pleasurefor me at these conferences is meeting people I have only known by phone or letter. For me, it was a special delight to get to know Judi Harris in person — we have only communicated via electronic mail before. I even got to introduce Judi to Eadie Adamson. What a special treat!

And of course Gary Stager was everywhere. (Is it a Logo conference without Gary?) And there was Robs Muir — remember *InteXual Challanges*? And I had a chance to chat with Dan and Molly Watt about something other than the next *LX* deadline. Of course it was great to see Ricky Carter and find out what Paul Goldenberg was doing and be on a panel with Maggie (Moore) Niess and talk to Dorothy Fitch and Jock McClees and hear about Steve Sesko's "kids" and find out what Brian Silverman was up to and — well, this list could go on and on and on.

Products

Of course, the vendors all hope the reason we come to conferences is to visit their booths. On the exhibit floor there is always an emphasis on products — and usually a sense of excitement over some new development. LogoPLUS, Terrapin's upgraded version of Logo attracted a lot of attention at this conference since it is so new and few people had seen it. A frequently heard question was "Which should I get for my school: LogoWriter or Logo PLUS?" Needless to say, the Terrapin and LCSI booths were certainly busy. While Lego/Logo is not a new product, many people have not seen it — or don't believe it until they do see it. Lego also had their hands full.

Ideas

Conferences always bring a richness of new ideas. This conference was no exception. The program was filled with more Logo sessions than you could possibly attend. Participants agonized over the schedule as they wished they could be in two places at once.

As is often the case, there were both formal and informal meetings after the sessions. A group of about 15 of us gathered to discuss the future of SIGLogo and *LX* as well as the future of Logo in general. The meeting was intense but most of us came away with an optimistic feeling that the Logo community is continuing to grow and that there are indeed ways that we can those new to Logo. A much larger group of LCSI consultants met to discuss the problems involved in teacher training. The conclusions were much as you might expect: we need more teacher training and we need more classroom oriented materials. We all agreed to do our part as best we could -- how about you?

There were, of course, keynote speeches. Of particular interest to me was the contrast between the keynote speech given by Tom Snyder (of Decisions Decisons, the Other Side and other similar social studies software) and the keynote speech given by Seymour Papert. Tom's talk focused on the computer as a vehicle to empower the teacher. He speaks to real classrooms where there are few computers and reality means limited budgets for hardware and software. Practicing teachers often respond to Tom's message with a sense of relief: Maybe they **don't** have to get **every** student on the computer **every** day after all.

The next day, it was Seymour's turn. He spoke of "instructionism" versus "constructionism." Visionary that he is, he asked us to look towards a future where the whole nature of education will change and we'll look back in amazement at the archaic way education is delivered today. As always, he challenged us to dream about the future and work towards that vision.

Are Tom and Seymour in conflict? No, I think not. Both care about kids. Both care about teachers. Both want the best in our educational system. We need the realists and we need the visionaries. Both give us to tools to go into the classroom and do our best for our students. They both gave us ideas to savor and relish in the months to come.

What Next?

So now it's back to work — with warm memories and neat new ideas. My classes need teaching and the April *Logo Exchange* needs to be readied for all of you. Even if you can't attend a Logo conference, there is little doubt in my mind that you will reap the benefits as Gary and Eadie and Judi and Dan and Tom pass on to you some of the ideas that they gleaned from their whirlwind visit to Los Angeles.

Perhaps we'll see you are the next conference. Perhaps in Boston at NECC this summer. Perhaps you'll join us in Cleveland in the spring of 1990 for the Logo conference to be held there. Perhaps it will be at the next (?) West Coast conference? Wherever or whenever, I hope you'll introduce yourself and join the community of people that we all look forward to seeing at the next conference.

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

Letters

[The other day, I logged on to BITNET only to find electronic mail from Ken Johnson of the British Logo User Group. That was the first of what has turned out to be many delightful long-distance communications. The letter below was in response to my questions about what was going on in Ken's part of the world. Ed.]

The British Logo User Group is putting the finishing touches to a fairly solid-looking book of things to do with Logo in the secondary school (in England this means 11-17 years, in Scotland 12-17.)

You may have heard that the Ministry of Education (whose writ runs in England and Wales, but not in Scotland) and the Scottish Education Department have been approving "The National Curriculum", a series of fairly heavy-going documents which have legal force and which specify what children will do in school for certain "core" subjects. This has been contentious, but it is now fact and we have got to get on and live with it.

Therefore a gang of us (Mike Doyle, chairman; me, vice-chairman; Ros Penny, secretary; Peter Smith, Viv Cayley, Peter Butt Dave Pratt and Chris West) met for a weekend and wrote a pile of materials relating to some National Curriculum topics: Ros and I chipped in a longish chapter suggesting practical activities on probability. Mike has been editing the result and we expect it to be distributed to BLUG members and to local education authorities in March. Later there will also be an Acorn Archimedes version. There is a lot of good material in the book so we are hoping it will attract lots of interest.

Ken's BITNET address is KEN@AIAI.ED.AC.UK

Monthly Musing

Assembly and the Salad Bar
by Tom Lough

I like salad bars. I enjoy the visual feast of the beautiful greens, reds, and whites. Aromas from the various selections give me pleasure. And, of course, nothing can beat crunching down on crisp fresh broccoli or carrots.

Recently, while constructing an all-you-can-eat masterpiece, I began musing about the salad bar on a different level. It all happened quite suddenly. I found myself thinking about the broccoli and the carrots and the croutons as procedures, and the salad bar itself as a superprocedure!

Then I thought a little about the history of each element of the salad bar. The alfalfa sprouts came from one place, the radishes from another, and the tomatos from yet another. The three-bean salad was another superprocedure in itself! And don't even ask me about the pasta salad! Each had its own fascinating developmental history, the result of the effort of countless people.

Finally, I thought about the act of assembling the entire combination. Someone had to figure out where each selection in the salad bar should go. Let's see, put the dressings here, the leafy greens here, and the sprinkles over there. Place the regular vegetables along in here, with the more expensive ones in the back row so people would have to reach furthest for them. It had to be just so.

It was the action of assembly that especially fascinated me. There was something about the idea of bringing together elements from so many different places. It stirred up a special memory from an earlier day.

It was in the spring of 1982. How could I ever forget? As a student in the very first Logo class taught by Steve Tipps and Glen Bull at the University of Virginia, I was interested in learning how this computer language might be applied to science teaching. It was "nice" that Logo was interpreted, so that I could see the results of my commands immediately.

But, when Glen and Steve showed that one could create a HOUSE procedure by assembling a SQUARE procedure and a TRIANGLE procedure together, the idea set my mind on fire! I could make up any new procedure I desired, just by bringing together other procedures and commands in any order I wanted! It was this excitement which fueled the establishment of the original newsletter, which later grew to become your *Logo Exchange* magazine.

Because of the new generations of Logo versions and Logo-related products, I continue to sense that excitement today. I am able to assemble elements of many different kinds in many different ways than I ever could have imagined previously. The sense of accomplishment this action gives me is difficult to describe, but it is extremely satisfactory.

I often think about all the different situations from which my students came. Each individual was a fascinating procedure; the class assembled was a magnificent superprocedure! And, as their guide and facilitator, I could suggest various subprocedures which produced interesting results. Heady stuff, this student salad bar!

This magazine you are holding in your hands is also a salad bar of sorts. It is made up of a variety of selections. You can pick and choose from among them in any order you want. I hope that it is a useful superprocedure to you. But, more than that, I hope that you will introduce several of your professional acquaintances to the *Logo Exchange*. Share with them the different articles and reports. Give them the opportunity to benefit from the contributions of their colleagues. Better yet, send me (or Sharon) their names and addresses and we will arrange for them to receive a free sample issue of the Logo Exchange. Let them experience for themselves this fascinating collection of application ideas.

FD 100!

Tom Lough
Founding Editor
Box 394
Simsbury, CT 06070

PS: Here is a small assembly challenge which you or your students might find amusing. Consider the classic HOUSE drawing.

Using only classic SQUARE and TRIANGLE procedures, can you draw such a house with the following procedure?

```
TO HOUSE
SQUARE
TRIANGLE
END
```

The classic SQUARE and TRIANGLE procedures consist only of a REPEAT command with a turtle move and a turtle turn inside the brackets. I'll send a special prize to the first five readers who send me workable solutions.

Logo Ideas

Part Two: Text and LogoWriter
More on Text:
Putting the Reader in Control
by Eadie Adamson

Last month we looked at the idea of programming a story, focusing on the display and pacing of text. This month we'll look at some more ideas that my students developed when working with text.

You can allow the reader to control the appearance of the next piece of text. First, use a procedure which asks the reader to press a key when ready:

```
TO MESSAGE
CC
TYPE [Press any key to go on...]
TYPE CHAR 13
END
```

Add a procedure which requires an input, but then does nothing:

```
TO IGNORE :KEY
END
```

Use these two procedures in a text procedure like this (if you name it STARTUP it starts automatically when you get the page):

```
TO START
MESSAGE
IGNORE READCHAR
PARAGRAPH1  <--This procedure
END              contains some text
```

The reporter, READCHAR, produces the input for IGNORE. READCHAR waits for a key to be pressed, passes the value of the key pressed to IGNORE and then Logo runs the next command in your START procedure. The next command can simply clear the text (CT) or it can insert another paragraph, add animation or sound, or go to another page.

Switching Screens at the Touch of a Key

The command NEXTSCREEN will move the next screen of text into view on a page with more than one screenful of text. TOP returns the cursor to the top of the page. Some of my students chose to put their entire story on the front of the page, and then move the text so that each screen contained only one or two paragraphs at a time. They wrote a program like the one below to control these screens of text.

```
TO START
MESSAGE
IGNORE READCHAR
NEXTSCREEN
IGNORE READCHAR
NEXTSCREEN
IGNORE READCHAR
TOP
END
```

Some students used multiple screens but programmed control keys to do the work:

```
TO KEYS
WHEN "Z [NEXTSCREEN]
WHEN "X [PRESCREEN]
WHEN "P [BOTTOM]
WHEN "Q [TOP]
END
```

The command NEXTSCREEN moves the next screen of text into view. The command TOP moves back to the top of the page. Similarly, BOTTOM moves the cursor to the bottom of the text, and PRESCREEN (previous screen) moves back one screen

Other students used the copy keys to put parts of their story onto other pages, leaving the STORY page as a kind of backup page of text. Then they programmed their stories using the GETPAGE command.

Giving the Reader a Cue

When using control keys to move from page to page or screen to screen, the reader needs to be told which keys to press. In the procedure below, the instructions appear in the Command Center, leaving the LogoWriter page for the text:

```
TO PRESSKEYS
CC
TYPE [PRESS CONTROL AND Z to read
   the next paragraph]
TYPE CHAR 13
TYPE [CONTROL AND Q to get back to
   the beginning..]
TYPE CHAR 13
END
```

TYPE, which makes text appear in the Command Center, works just as INSERT does on the LogoWriter page: the cursor remains at the end of the text. TYPE CHAR 13 adds a carriage return, moving the cursor to the next line. CC simply clears the lines below the page so that only the instructions appear in the Command Center.

Logo ideas -- continued

Other students used screen shifts combined with alternately changing background from black to white to signal new text for the reader:

```
TO STORY
MESSAGE
IGNORE READCHAR
SETBG 0
PARAGRAPH1
IGNORE READCHAR
SETBG 1
PARAGRAPH2
END
```

Some students even added a beep with each screen change.

The Title's the Thing

A number of my students felt that a title page for a story was very important. Although we were deliberately not using graphics in this exercise, some students wanted to design letters and stamp the shapes on the screen for a title. The title page, often created without the use of procedures, included a STARTUP procedure on the Flip side.

```
TO STARTUP
MESSAGE
IGNORE READCHAR
GETPAGE "PART.ONE
END
```

This STARTUP procedure uses the MESSAGE procedure described earlier in this article to print the message in the Command Center. The reader can then press a key to start the story, which begins on page PART.ONE.

Since I did not outlaw sound, some students added music to the title page. Sometimes pressing a key to stop the music was a great idea!

A Title with a Play on Words

Commands such as TAB, which indent the cursor five characters, can be used to aid in creative placement of text on the screen. Using various combinations PRINT [ ] (or PRINT "), TAB and a SP procedure (which prints a blank space), it is possible to program text to appear in any part of the page.

One of my students had written a story he named "Sky Jump" and decided to play with some of the words. Matthew wanted the JUMP in the title to appear to move down the page. Here's the procedure he developed as the title for his story:

```
TO TITLE
CT
PRINT []
PRINT []
PRINT []
PRINT []
PRINT []
TAB
TAB
SP
SP
PRINT [SKY]
WAIT 10
REPEAT 3 [TAB]
PRINT [J]
REPEAT 3 [TAB]
SP
PRINT [U]
REPEAT 3 [TAB]
SP
SP
PRINT [M]
REPEAT 3 [TAB]
SP
SP
SP
PRINT [P]
END
```

Although it is possible to use a quotation mark before a single letter or word and brackets for a list of words, many students seem to prefer using brackets, as Matthew did. As you can see, such procedures can get easily get quite long.

More Motion

I was also fascinated with the idea of moving the title letters about. I played with Matthew's TITLE for awhile and came up with a variation I liked. It uses three more cursor commands:

CU (cursor up) to move up to the printed line above where the cursor is
CB (cursor back) to move back one character
CD (cursor down) to move down a line.

Here is the procedure I wrote:

```
TO TITLE2
SKY
J
U
M
P
FINISH
END
```

It is made up of subprocedures that handle each word or letter in the title.

```
TO SKY
CT
REPEAT 5 [PRINT []]
TABS 2
SPS 2
PRINT [SKY]
WAIT 10
END

TO J
TABS 3
PRINT "J
END

TO U
CB
CB
PRINT []
TAB.W 3
CF.W 2
TAB.W 3
SP.W 1
PRINT "U
END

TO M
CU
CB
PRINT []
TAB.W E
CF.W 2
CD
TAB.W 3
SP.W 2
PRINT "M
END

TO P
CU
CB
PRINT []
TAB.W 3
CF.W 4
CD
TAB.W 3
SP.W 3
PRINT "P
END

TO FINISH
CU
CB
PRINT []
CD
CD
PRINT []
TAB.W 4
END
```

In order to save typing, I wrote a number of tool procedures when there were actions that were repeated frequently.

```
TO TABS :NUMBER
REPEAT :NUMBER [TAB]
END

TO SPS :NUMBER
REPEAT :NUMBER [SP]
END

TO TAB.W :NUMBER
REPEAT :NUMBER [TAB W]
END

TO SP.W :NUMBER
REPEAT :NUMBER [SP W]
END

TO CF.W :NUMBER
REPEAT :NUMBER [CF W]
END

TO W
WAIT 1
END
```

This gives results similar to Matthew's with one exception: The letters in JUMP get pushed down to the next line after they are printed on the screen.

An example like this will cause your students to think more actively about the title they use. It provides an interesting way to explore the various text commands that you can use with LogoWriter. Challenge your students dream up some good titles or phases which can be programmed interestingly. What kinds of plays on words can they invent which can be illustrated in some way by the display of text? Can they present their titles, all by means of procedures using text commands?

Some Additions to March Logo Ideas

Memory Problems

Some of my students liked the idea of using the INS procedure (together with IN) for inserting the text a letter at a time. They thought they had found a quick and easy way to program their stories by simply enclosing all their text in brackets preceded by the command, INS, which inserts each word a letter at a time. The effect they get is not what they expect. At first the letters appear rapidly, but as more text

Logo Ideas -- continued

accumulates on the page the process begins to slow down noticeably. In fact, if you wait long enough it may stop entirely!

How to Fix the Memory Problem

There is a way around this problem which preserves the letter at a time process but encourages more thinking about how to tell the story. If you tried this with a long paragraph you probably noticed that for the first sentence or two everything moved quite well and then gradually moved slower and slower. The solution is to clear the text periodically, before the slowing down of the cursor becomes apparent. My students elected to use a procedure like this:

```
TO CLEAR
CT                              This moves the
REPEAT 5 [PRINT []]   <-cursor down
END                             the page.
```

CLEAR not only clears the text (CT) but moves the cursor down so that the next line or two of text is displayed more towards the center of the screen. The text becomes more interesting to look at than if it continually appears at the top of the screen. Use CLEAR this way:

1. Insert the word CLEAR at the beginning of the story procedure.
2. On the next line enter INS and then enclose the first sentence or two in brackets.
3. Repeat the process as often as necessary for the story.

Using CLEAR to solve the memory problem also requires a little more thinking about how to pace the telling of a story. How many sentences or how much of a sentence should appear before invoking CLEAR again? The technique of clearing the text and then printing a bit more also allows one to insert pauses in the story (and even appropriate sounds). A line of dashes or dots could appear at a crucial time, or the screen could clear, background change to white, pause, and then return to black as more text appears. Alternating between rapid display of text and a very slow, measured pace can add considerable drama to the telling. In this case, a problem with page memory forces students to think more creatively about how to tell their story.

Coming up, still more on text....

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

Stager's Stuff

The Magic Circle Microworld
by Gary S. Stager

If one of my math teachers had told me that circumference was virtually the same as perimeter (I learned this secret at a Logo conference) I would have been a much better math student. Occasionally, something I learned years before finally makes sense. Recently, while driving down the interstate, the concept of binary arithmetic finally clicked after twelve years of computer use — I won't even discuss how long it took me to understand recursive operations! Whenever I really understand a powerful idea I try and find a way of teaching it to my students. This article provides some simple ways for students to understand the seemingly complex area of circle measurement: circumference, diameter, radius, and pi.

One of the powerful ideas inherent in measurement is that there are often several different ways of expressing the size of an object, in our case a circle, and that there is a relationship between these measurements. The power and flexibility of turtle graphics provides us with an environment for the playful exploration of circles. Not everybody will understand the interrelationships among circumference, diameter, radius, and pi through pencil and paper arithmetic definitions and exercises. Logo can provide a more visual and concrete method for exploring these powerful ideas.

I think some definitions are in order to create a framework for understanding this microworld.

Circumference : The distance around a circle.

Diameter: A line segment from any point on the circle, through the center, to another point on the circle. The diameter is a chord passing through the center of the circle and is the longest chord in a circle.

Radius: A line of segment from the center to any point on the circle. For a given radius and a given center there is only one possible circle.

Pi (π): Pi is an irrational number (never ending repeating decimal) and equals the circumference of a circle divided by the diameter of that circle.

The Microworld's Tool Procedures

```
TO CIRCLE1 :CIRCUMFERENCE
REPEAT 360 [FORWARD :CIRCUMFERENCE
   / 360 RIGHT 1]
END

TO CIRCLE2 :DIAMETER
REPEAT 360 [FORWARD (PI * :DIAMETER)
   / 360 RIGHT 1]
END
```

```
TO CIRCLE3 :RADIUS
REPEAT 360 [FORWARD (2 * PI *
   :RADIUS) / 360 RIGHT 1]
END

TO PI
OUTPUT 3.1415927
END
```

One explore the concepts of radius, diameter and circumference is by using turtle graphics. In this activity, students will use the CIRCLE1, CIRCLE2, and CIRCLE3 procedures to create congruent (same size and shape) circles by using different inputs. All three procedures draw circles, but each produces circles that require different values as input. Begin by typing:

```
CG          (CS in some Logo versions)
CIRCLE1 300
```

A circle with a circumference of 300 turtle steps will be drawn on the screen and the turtle returns to its original position and heading.

Now use the CIRCLE2 procedure and try to draw the same circle (it should overlap the original circle). Do this by trying different inputs to CIRCLE2. You may wish to change the turtle's pen color between trials so that the circles are distinguishable. When the turtle finally traces the original circle, the number you used as an input to CIRCLE2 is the diameter of the circle. CIRCLE2 95.5 should create the same circle as CIRCLE1 300. Pi, the ratio between the circumference of a circle and it's diameter, is a constant that never changes, regardless of the circle's size.

Check the accuracy of your diameter discovery by typing:

```
SHOW <value used for diameter> * PI
```

or, in the above example

```
SHOW 95.5 * PI
```

The result of your calculation should be approximately equal the original circumference of the circle (the input to CIRCLE1). The resolution of the screen is not perfect so there is bound to be some slight error in your measurement - using HT may improve your accuracy.

Try this same experiment again, but this time command the turtle to draw a circle with a circumference of 300 by using the CIRCLE3 procedure whose input represents the circle's radius. Hint: The radius of a circle is half of the diameter. Therefore, CIRCLE3 47.75 should draw the same circle as CIRCLE1 300.

Challenges:

Try the activity above but this time vary the size of the original circle or use the three procedures in a different order. What is the circumference of a circle drawn as a result of CIRCLE3 50 ?

Circle Magic

We can make this activity more challenging and game-like by asking Logo to draw us a circle of an unknown size. The MAGIC procedure makes a global variable, :CIRCUMFERENCE, containing a value between 200 and 400. This global variable can be used as an input to the CIRCLE1 procedure thereby creating a mystery circle.

```
TO MAGIC
MAKE "CIRCUMFERENCE 200 + RANDOM 401
END
```

Type:

```
MAGIC
CIRCLE1 :CIRCUMFERENCE
```

We can use the turtle to measure the diameter of our mystery circle (or any circle) by turning RIGHT 90 and commanding the turtle, by successive approximations, to walk across the circle until the turtle reaches the opposite side of the circle. The distance across the the circle traveled by the turtle is, of course, the diameter. The circumference of the mystery circle equals a little more than 3 times the diameter of the circle. Half of that distance is the circle's radius.

To check your answer, find out the real circumference of the circle by typing:

```
SHOW :CIRCUMFERENCE
```

Find the difference between your estimated diameter and the :CIRCUMFERENCE. The magic number should be very close to the estimated diameter measured by the turtle.

Let the Turtle Do the Walking

In LogoWriter, Logo II, and Terrapin Logo Plus we can even more dramatically illustrate the relationship between diameter and circumference by having the turtle actually do the measuring for us. The procedure, MEASURE, is a reporter that uses Logo's collision detection abilities to determine when the turtle hits the line that forms the circle and then reports the distance traveled across the circle. It is always fun to let the Logo turtle work for us.

1. Draw a circle with CIRCLE1, CIRCLE2, or CIRCLE3.
2. Turn the turtle towards the opposite side of the circle (generally RIGHT 90)
3. Type:

```
PU
PRINT MEASURE
```

when the turtle hits the opposite side of the circle it will display the approximate diameter of the circle.

In LogoWriter, use:

```
TO MEASURE
PU
OUTPUT MEASURE.DIAMETER 1
END

TO MEASURE.DIAMETER :DIAMETER
FORWARD 1
IF NOT (EQUAL? COLORUNDER BG)
   [OUTPUT :DIAMETER + 1]
OUTPUT MEASURE.DIAMETER :DIAMETER + 1
END
```

In Terrapin Logo PLUS, use this procedure:

```
TO MEASURE.DIAMETER :DIAMETER
FORWARD 1
IF SDOT? THEN OUTPUT :DIAMETER + 1
OUTPUT MEASURE.DIAMETER :DIAMETER + 1
END
```

In Logo II use this procedure instead:

```
TO MEASURE.DIAMETER :DIAMETER
FORWARD 1
IF NOT (DOTP POS)
   [OUTPUT :DIAMETER + 1]
OUTPUT MEASURE.DIAMETER :DIAMETER + 1
END
```

Different versions of Logo or different hardware have different rates of error in their collision detection abilities (the reasons are not worth explaining). My goal for these activities is not to find the empirical value of pi or the circle's diameter accurate to 6 decimal points. I want students to have practical and meaningful experience with these concepts in the hope that these ideas will stay with them for years to come.

I would like to leave you with a view on mathematics education I believe is worth pondering. An official of the National Science Foundation recently testified that, "America is the only Western industrialized nation that thinks that mathematics is an innate ability." The official went on to say, "The Japanese do not think they are particularly good at mathematics. They just work hard at it." How many times have you heard a parent, student, or colleague say, "I'm no good at math"?

Gary Stager
NAME, Fallon Education Center
51 Clifford Drive
Wayne, New Jersey 07470
CIS: 73306,2446 Applelink: K0331

"Mr. Logical" is Logo Winner

The 1988 International Computer Problem Solving Contest: Junior Logo Results

by Donald T. Piele

Matthew Frank, better know as "Mr. Logical" by his friends at Mirman School in Los Angeles, California, won the 1988 Junior Logo Division of the International Computer Problem Solving Contest. This was Matthew's fourth entry in ICPSC and this year he left nothing to chance. To reach his peak, he worked on solving similar Logo problems several weeks before the contest and attributes his success to regular practice.

Matt's first exposure to computer programming began at summer camp in Southern California. This led to sharing the family Apple IIe with his father and eventually taking it over completely. He now uses the same machine for word processing, BASIC and Logo programming, and occasionally shares it with his younger brother.

Matt has many interests including music, word games, logic puzzles, and mathematics. He has enrolled this year in a college calculus course at night while attending Culver City High School as a freshman. His athletic interests are cross-country running and tennis and he still has time for Spanish, writing, math, chess, and computer clubs.

This year there were four teams that solved all five problems in the Junior Logo Division out of the approximately 93 teams that tried.

1988 Junior Logo Division Rankings

First Place

Matthew Frank
Mirman School
Los Angeles, CA
Director/Advisor: A. Treacy

Second Place

Drew Narayan
Oswego Middle School
Oswego, NY
Director/Advisor: Harold Travis

Third Place

Niall Couse and Kevin Motherway
Colaiste an Spioraid; Naoimh
Bishopstown, Cork, Ireland
Director/Advisor: Michael D. Moynihan

Fourth Place

David Frink and Charles Powell
Ligon Middle School
Raleigh, NC
Director/Advisor: Cathay Smith

The Importance of Style

This was our second year of offering a Logo contest in the Junior Division. It could not have been done without the assistance of Sharon Burrowes Yoder, who teaches computer education courses at the University of Oregon as well as serving as editor of the *Logo Exchange*. Sharon was responsible for creating the Logo problems, solutions, and for ranking the best teams. She remarked that style was often the determining factor and that bad style consists of:

- lots of unrelated statements on a single line;
- extremely long procedures;
- modularity not used to its fullest;
- procedures without meaningful names;
- procedures shown which weren't part of the final program.

She also observed that some teams were ranked lower because they apparently did not take the first problem seriously enough and thus never polished their code.

1989 Contest

The 1989 ICPSC will be held on Saturday, April 29, 1989 with Friday, April 28 and Monday, May 1 as alternate contest dates. *Compute It!*, the newsletter of the ICPSC, includes more information on the event and how your school or school district can become a contest site. For a free copy, write to me at the address below.

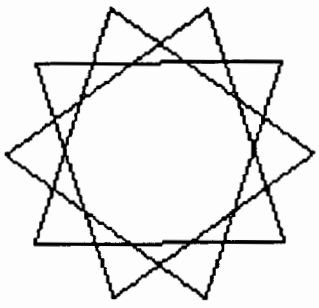
Donald T. Piele, ICPSC
P.O. Box 085664
Racine, WI 53408

Logo LinX

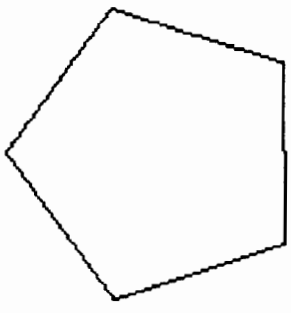
Multiplication Clouds Have Turtled Linings

by Judi Harris

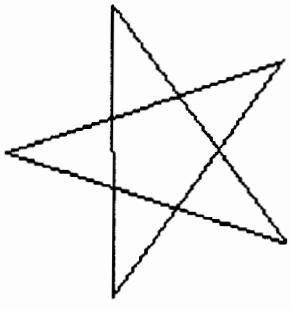
Does it seem that April showers may have made your students' multiplication facts a bit rusty? Believe it or not, here is a way for them to practice the sevens table:



In drawing this figure, they can review the eights facts:



And this design can help them to memorize the sixes table:



Aligned for Review

G. H. Hardy told us that "a mathematician, like a painter or a poet, is a maker of patterns." Turtled designs make mathematical patterns viewable in graphic form. Since it is easier to remember a series of numbers that fit an overall pattern than it is to memorize seemingly random numbers, it makes good pedagogic sense to help children to recognize multiplication table patterns by drawing them with the turtle.

Consider, for example, the nines table. It is often the most difficult for students to memorize.

1 x 9 = 9
2 x 9 = 18
3 x 9 = 27
4 x 9 = 36
5 x 9 = 45
6 x 9 = 54
7 x 9 = 63
8 x 9 = 72
9 x 9 = 81
10 x 9 = 90

You may have already asked your students to notice the order of the digits in the ones column:

9 8 7 6 5 4 3 2 1 0

The sequence of numerals in the tens place is similar:

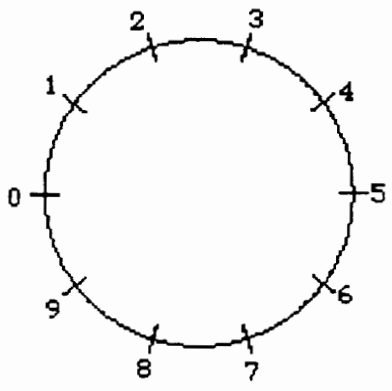
0 1 2 3 4 5 6 7 8 9

And if you add each pair of digits, what is the result in each case?

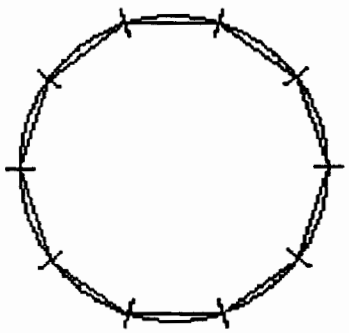
Attending to patterns such as these make drill more meaningful and (hopefully) more interesting. Let us now look at multiplication facts in more *graphic detail*.

Digital Lineage

If we arrange the digits 0 - 9 at equidistant intervals around the circumference of a circle,

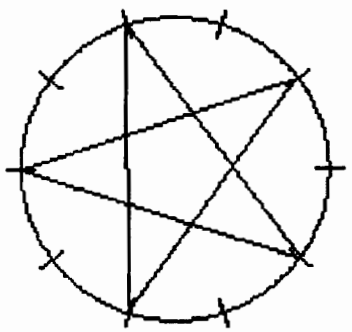


then draw a line connecting the ones' place digits for the nines facts, in order,



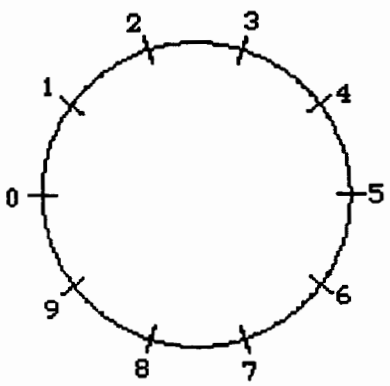
the result is a decagon.

But if we connect the ones' place digits for the fours table, 4 8 2 6 0 4 8 2 6 0



the resulting graphic is a bit more surprising.

Sketch the ones' digit design for the sixes facts here, and you will receive a surprise of a different sort.



Why do you think some of these patterns are the same?

Fall in Line

Turtling these patterns can begin by using Logo to measure 10 evenly-spaced positions around the circumference of a 360-step circle. We know that the circumference of any circle is approximately equal to pi (~3.14) multiplied by twice the circle's radius. Therefore, to center the circle on the screen, we can reset the turtle's position to one radius to the left of HOME. 360 / 3.14 / 2 ~57, so we tell the turtle to

```
PENUP
SETPOS [-57 0]
PENDOWN
```

A 360-step circle can be drawn by entering the command

```
REPEAT 360 [ FORWARD 1 RIGHT 1 ]
```

but we would like to stop at 10 equal intervals along the way, and ask the turtle what its screen position is each time.

```
REPEAT 10 [ PRINT POS REPEAT 36
          [ FORWARD 1 RIGHT 1 ] ]
```

Round these numbers to the nearest integers, and construct ten subprocedures that output the ten screen positions as lists.

```
TO PLACE0
OUTPUT LIST -57 0
END

TO PLACE1
OUTPUT LIST -46 34
END

TO PLACE2
OUTPUT LIST -18 55
END

TO PLACE3
OUTPUT LIST 18 55
END

TO PLACE4
OUTPUT LIST 46 35
END

TO PLACE5
OUTPUT LIST 58 1
END

TO PLACE6
OUTPUT LIST 47 -33
END

TO PLACE7
OUTPUT LIST 18 -54
END

TO PLACE8
OUTPUT LIST -17 -54
END

TO PLACE9
OUTPUT LIST -46 -34
END
```

You may want to make your circle demarcations on a larger (i.e., 720-step) or smaller circle.

Single File

The superprocedure should be constructed so that students can enter a list of digits that represent the ones' place of a multiplication table to make the computer draw the corresponding pattern. If a user's list begins with 3 6 9...., the turtle should move from PLACE3 to PLACE6 to PLACE9, leaving a turtled trail as it goes.

Logo LinX -- continued

Let us assume that the user's digit list is stored in a variable named DIGITS. The procedure MAKE.LINE tells the turtle to reset its screen position according to the first elements of :DIGITS.

```
TO MAKE.LINE :POINT
SETPOS RUN PLACE :POINT
END

MAKE.LINE (FIRST :DIGITS)
```

MAKE.LINE calls a subprocedure (PLACE) that translates the digit from the list into a PLACE0 - PLACE9 subprocedure name.

```
TO PLACE :NUMERAL
OUTPUT LIST WORD "PLACE :NUMERAL
END
```

The recursive procedure DRAW tells the turtle to proceed from point to point around the circle, according to the order specified by the user's digit list.

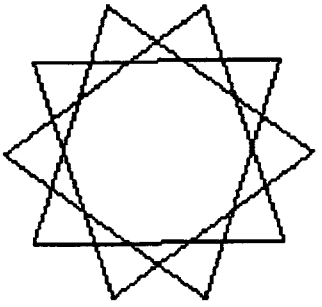
```
TO DRAW :DIGIT.LIST
IF EMPTYP :DIGIT.LIST [SETPOS
   RUN PLACE :FINAL.POINT STOP]
MAKE.LINE (FIRST :DIGITS)
DRAW BUTFIRST :DIGIT.LIST
END
```

DRAW is called by the superprocedure DESIGN.

```
TO DESIGN
CLEARSCREEN
PENUP
SETPOS [-57 0]
PENDOWN
CLEARTEXT
PRINT [PLEASE TYPE THE DIGIT LIST
    HERE:]
MAKE "DIGITS READLIST
MAKE "FINAL.POINT (FIRST :DIGITS)
PENUP MAKE.LINE (FIRST :DIGITS)
PENDOWN
DRAW :DIGITS
END

DESIGN PLEASE TYPE THE DIGIT LIST
HERE:
   3 6 9 2 5 8 1 4 7 0
```

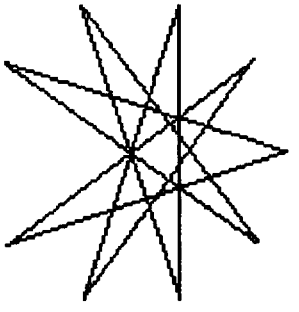
Drop a Line

Like rabbits in this spring season, patterns beget patterns.



What if these same tools were used to examine sequences of ones' place numerals when multiplication fact answer digits are summed?

```
 1 x 5 =  5  Sum: 5
 2 x 5 = 10  Sum: 1
 3 x 5 = 15  Sum: 6
 4 x 5 = 20  Sum: 2
 5 x 5 = 25  Sum: 7
 6 x 5 = 30  Sum: 3
 7 x 5 = 35  Sum: 8
 8 x 5 = 40  Sum: 4
 9 x 5 = 45  Sum: 9
10 x 5 = 50  Sum: 5
```



What patterns might emerge if answer digits were subtracted? Multiplied? Why are some patterns identical? Why are others asymmetrical? Would similar graphics emerge using addition facts?

Perhaps *this* spring's out-of-line(s) classroom behavior *should* be multiplied!

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

About the Cover

Cory Dahlquist drew this picture when he was a 6th grader in Karen Thimmeschs' class at Galtier Magnet School in St. Paul. Paul Krocheski, computer teacher at Galtier, writes that Cory's picture was the result of a project "to use squares, rectangles, circles and triangles in a master procedure to produce a human and/or animal representation." He notes that this project helps students become better at understanding the use of subprocedures in a master program

Logo Connections

A Text Instant

by Glen L. Bull and Gina L. Bull

A popular type of Logo program is called Single-Key Logo, or sometimes, Instant Logo. In these versions of Logo a single key press makes the turtle move. For example, pressing the letter "F" might make the turtle go forward, while pressing the letter "R" might make it turn right. Through LogoWriter it is now possible to create a Single-Key Logo for the text screen rather than the turtle.

Dr. Dolittle could talk to the animals, but it is not recorded whether he could talk to words ... or, if he did, whether they answered back. LogoWriter can talk to words on the text screen, and they will answer back. In this column we are going to demonstrate use of a simple Text Instant which allows a child to move the cursor around the screen and search for words which contain a target letter. If the child thinks the cursor is over a word containing the target letter, she can press the space bar. LogoWriter will beep if the word contains the target letter.

Moving the Cursor

This Instant Logo requires two components:

1. a procedure to move the cursor around the screen, and
2. a procedure to detect words which contain the target letter.

Here is a procedure that moves the cursor around the screen. It makes use of the LogoWriter commands CU (Cursor Up), CD (Cursor Down), CB (Cursor Back), and CF (Cursor Forward.

```
To Move.Cursor
MAKE "Input ASCII READCHAR
IF Up.Arrow? [CU STOP]
IF Down.Arrow? [CD STOP]
IF Left.Arrow? [CB STOP]
IF Right.Arrow? [CF STOP]
IF Equal? CHAR :Input "Q [STOPALL]
Move.Cursor
END
```

The **Move.Cursor** procedure has four subprocedures which detect which arrow keys have been pressed.

```
TO Up.Arrow?
OUTPUT :Input = 11
END

TO Down.Arrow?
OUTPUT :Input = 10
END

TO Left.Arrow?
OUTPUT :Input = 8
END

TO Right.Arrow?
OUTPUT :Input = 21
END
```

The numbers generated by the arrow keys on the IBM and Apple keyboards are different. Therefore, if you are using IBM LogoWriter, substitute the following values in the arrow key procedures above:

	IBM LogoWriter Version
Up Arrow	328
Down Arrow	336
Left Arrow	331
Right Arrow	333

To try out the **Move.Cursor** procedure, type some text on the front side of a LogoWriter page. Then go down to the Command Center at the bottom of the screen, and run the procedure by typing:

```
Move.Cursor
```

When the procedure has run, the arrow keys should move the cursor around the screen. After testing the procedure, type "Q" to halt it.

The Letter Under the Cursor

Next a procedure is needed to detect whether a word has the target letter in it. The following procedure, found in the WORDTOOLS page of your LogoWriter utilities, will show the character under the cursor:

```
To CursUnder
SELECT
CF
MAKE "Letter SELECTED
UNSELECT
CB
OUTPUT :Letter
END
```

After you define the **CursUnder** procedure, position the cursor over a letter in the text on the top half of the screen. Then return to the Command Center at the bottom of the screen, and try the following:

```
SHOW CursUnder
```

Did the **CursUnder** procedure show the letter under the cursor?

Logo Connections -- continued

Clipping Out a Word

The following procedure uses **CursUnder** to **Clip** out the letters from the cursor position to the end of the word. It does this by using the CF (Cursor Forward) command to move forward a letter at a time until the end of the word is encountered. The end of the word is identified by the presence of a word separator, such as a space (Character 32) or an end of line character (Character 13).

```
To Clip :Item
IF WORD.Separator?
   [SETTEXTPOS :Place OUTPUT :Item]
MAKE "Item WORD :Item :Letter
CF
Clip :Item
END

To Word.Separator?
MAKE "Letter CursUnder
OUTPUT (OR :Letter = CHAR 32 :Letter
    = Char 13 :Letter = Char  9)
END
```

Position the cursor at the beginning of a word. Then move the cursor back down to the Command Center, mark the current cursor position, and run the **Clip** procedure:

```
MAKE "Place TEXTPOS
SHOW Clip "
```

The Word Under the Cursor

The **Clip** procedure should clip out and print the word beneath the cursor. It is necessary to mark the cursor position so that the cursor can be returned to its original position. However, this process can be automated, so that a procedure simply prints the word beneath the cursor. This procedure, **CursorWord**, also moves the cursor to the start of the word so that it is not necessary to move the cursor to beginning of the word before running the procedure. (Note: If you are using Version 2.0 of IBM LogoWriter, you can speed up the procedure by substituting the primitive CURSORCHAR in place of the procedure **CursorWord**.)

```
To CursorWord
IF Word.Separator? [OUTPUT " ]
MAKE "Place TEXTPOS
Start.of.Word OUTPUT Clip "
END

To Start.of.Word
IF TEXTPOS = 0 [STOP]
IF Word.Separator? [CF STOP]
CB
Start.of.Word
END
```

Position the cursor in the middle of a word. Then move down to the Command Center and run the procedure by typing:

```
SHOW CursorWord
```

The procedure should show the word under the cursor.

Detecting the Target Letter

The procedure **Check.for.Letter** looks for a target letter in a word. If a target letter has not been specified when the procedure is run, **Set.Letter** requests a target letter. When a target letter is found, **Check.for.Letter** beeps, but you can substitute any consequence you wish.

```
To Check.for.Letter
IF NOT NAME? "Target.Letter
    [Set.Letter]
IF MEMBER? :Target.Letter CursorWord
    [Beep]
END

To Set.Letter
CC
TYPE [Type the new target letter: ]
MAKE "Target.Letter READCHAR
CC
Show :Target.Letter
END

To Beep
TONE 400 2
END
```

Place the cursor in the middle of a word containing the target letter. Then return to the LogoWriter Command Center at the bottom of the page, and run the **Check.for.Letter** procedure. LogoWriter should beep, and then return the cursor to its original position.

Creating a Text Instant

If everything worked so far you now have a control structure (Move.Cursor) and a detection procedure (Check.for.Letter). These two procedures can now be combined to form an instant souffle. The **Instant** procedure reads a character from the keyboard, and then calls the **Command.List** procedure. The **Command.List** procedure checks the character to see if it matches a command. **Command.List** is a slightly modified version of **Move.Cursor**

```
TO Instant
MAKE "Input ASCII READCHAR
Command.List
Instant
END
```

```
To Command.List
IF Space.Bar? [Check.For.Letter]
IF Up.Arrow? [CU STOP]
IF Down.Arrow? [CD STOP]
IF Left.Arrow? [CB STOP]
IF Right.Arrow? [CF STOP]
IF Equal? Char :Input "S [Set.Letter]
IF Equal? CHAR :Input "Q [STOPALL]
END

To Space.Bar?
OUTPUT :Input = CHAR 32
END
```

Summary of Commands

This Single-Key Logo has the following commands:

Arrow Keys: move the cursor
Space Bar: checks for target letter
S: sets the target letter
Q: quits procedure

The structure shown for "S" and "Q" in the **Command.List** procedure can be extended to add other commands to this Text Instant.

This version of an Instant Logo for the text screen utilizes a procedure which detects the presence of a letter in a word. The concept could be extended to create suffix detectors which respond to the presence of "ing" or "ed", or prefix detectors, or word detectors which respond to a specific class of words (nouns or verbs or adjectives).

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

Little Kids and Logo

by Leslie Thyberg

Off-computer activities have at least five overall benefits. First, they provide another kinesthetic connection for the child to experiment with that becomes an extension of the computer and the Logo microworld. Second, they provide an avenue for reinforcement and review. Third, they provide a mechanism for individualizing. Fourth, at the preschool and primary level, readiness skills such as prereading, prewriting, and precomputing can be further developed. Last, they serve as a good management tool for maximizing hands-on computer time, especially if the computer-to-student ratio is not as rich as one would ideally have.

Teaching Turtle

The following is an example of a typical precomputer or off-computer lesson. Each year I have had the students name our turtle mascot. Over the years my turtle puppet has been christened Yertle, Myrtle, Tomi, The Big T, and Seymour. Once it is named, then you can play the following directionality game as "Name-of-the-Turtle" Says (e.g. Myrtle says)

Directionality Game

Objectives:

1. Student will differentiate between left (LT) and right (RT) turns
2. Student will recognize LT and RT as commands to make turns
3. Student will differentiate between FD and BK commands
4. Student will be able to carry out the commands

Materials:

1. A turtle puppet (Myrtle, Yertle, etc. Let the children name it unless you want to reserve the honor for yourself.)
2. LT and RT stickers for each handor yellow yarn on one hand and red yarn on the other)
3. Flash cards with LT, RT, FD, and BK printed on them (or L, R, F, B for QUICKDRAW or INSTANT versions)
4. Chalkboard and chalk or some other means to record directions

Procedure:

1. Introduce the puppet and tell the children that the turtle will be giving them directions in the "Myrtle Says" game (played just like Simon Says)
2. Give each child the LT and RT stickers or other hand markers.
3. Show the children on the board the commands for LT and RT and FD and BK. Demonstrate action with each one.
4. Play "Myrtle Says" giving oral directions (and modelling movement)
5. Play "Myrtle Says" using flash cards
6. (Optional): Let several children take turns giving the directions

Little Kids and Logo -- continued

Evaluation:

1. Teacher observation.

This first lesson can then be developed into further lessons using actual rotation commands (LT 90, RT 90; LT 180 RT 180, etc.). When students become reliably proficient they can be divided into pairs to give and receive commands from their partners.

In the spirit of a scavenger hunt, children can be given and/or receive commands to find an object or shape hidden in the classroom.

Other activities can include having the children make a large keyboard poster or play a twister-like game with either the keyboard characters or specific Logo primitives marked on an oilcloth or checkered tablecloth (which serves as a grid).

Making Turtles

Clay turtles (even chocolate turtles) can be made as a tactile activity to make the cybernetic turtle a friendlier playmate. Turtles can be made by drawing a turtle base pattern which the child can cut out. Half a walnut shell or the cup from an egg carton can be used to glue to the base. Turtles can also be made from paper plates, tag board or simple construction paper that is then laminated for durability. Using the turtles they have made, children can then follow simple directions that are given (moving FD, turning RT, shapes, etc.). Actual turtle paths and mazes can also be made. Shimabukuro and Green and Jaeger have some additional suggestions for turtle play.

Playing Turtle

Using turtle puppets (purchased at the store or student made) or hats with turtle attachments, students can play turtle. Other children can give the turtle commands which not only encourages the following of directions, but also involves discrimination and estimation skills as well creating opportunities for constructive social interaction.

Talking Turtle

While LogoWriter is not necessary to carry out this activity, its programming features make it much easier than other versions of Logo to implement. As a part of reading readiness children can readily learn some of the basic principles of word processing to complement the graphics which they generate with Logo. Story telling and writing activities are effective language learning experiences for preschool and primary age children. Logo graphics and inventing adventures for the turtle are wonderful motivators around which to spin stories. An additional bonus is how quickly children learn to recognize words and even the error messages. Periodic publication of a class newsletter ("Turtle Times" or "Turtle Soup") can give the students an audience and purpose for their projects.

Monitoring Turtles

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

Tagging Turtles

A system of sign-up charts having a daily or weekly schedule and time slots is an easy management scheme for scheduling computer time. If the chart is laminated, student names can be entered with water-based markers and changed daily or from week to week. Placing a small clock or timer by the computer lets the students know when their time is up. A group of teachers that I am currently working with in Tarentum, PA. divided their students up according to types of turtles. Students are grouped in one of six categories for rotation to the computer: Snappers, Sliders, Mappers, Painters, Boxers, and Giants. Because the students work with partners, their respective turtle groups are listed on the chart on a rotating basis, rather than individual names. The turtle "classes" that students work through are in their designated turtle groups: Turtle Tot, Turtle Trainer, and Turtle Teacher. Or Logo Learner and Logo Explorer. Or still another metaphor, putting aside the turtle theme: apprentice, sorcerer, wizard. The children in Tarentum make and wear buttons for each of the levels. Buttons are sold to supplement printer paper and ribbon costs.

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

In my former classroom I held monthly Academy Awards where everyone was a winner. Students submitted their favorite graphics which was then put on a show disk and run continually for Fun Friday afternoon. Usually this was done in conjunction with an open house for parents. Students presented group and individual projects and offered tutoring sessions for their parents on a variety of computer related topics. The culminating activity was the construction of an off-computer turtle village (Turtleburgh) that was incorporated with the class study of our community. A wide variety

of construction materials were used ranging from clay to papier-mache. In the spring we hosted the production of a student authored operetta, "Adventures in Logoland."

The following procedures can be used to have your own Academy Awards. They are written in the Terrapin version of Logo.

```
TO ACADEMY.AWARDS :L
PRINT FIRST :L
RUN (SENTENCE [READPICT]
   WORD " " FIRST :L)
PRINT REQUEST
RUN (SENTENCE [SAVEPICT]
   WORD " " FIRST :L)
PRINT REQUEST
MAKE "L BUTFIRST :L
ACADEMY.AWARDS :L
END

TO R
ACADEMY.AWARDS :L
END

TO SHOWIT :L :N
IF :L = [] SHOW
HT
FULLSCREEN
RUN (SENTENCE [READPICT]
   WORD  " " FIRST :)
SPLITSCREEN
REPEAT 2 [PRINT []]
PWZ 500
MAKE "L BUTFIRST :L
MAKE "N BUTFIRST :N
SHOWIT :L :N
END

TO PWZ :TIME
IF :TIME < 1 THEN STOP
PWZ :TIME - 1
END

TO SHOW
MAKE "L :LIST
MAKE "N :NAMES
SHOWIT :L :N
END

MAKE "LIST [ <names of the pictures
    on disk go here>]
MAKE "NAMES [<corresponding names
   of student artists go here>]
```

Run R to put pictures on disk. The program pauses to switch disks so that you can read a picture from an individual student's disk and save it on a class disk. Use the procedure SHOW to show the completed class disk.

References

Green, Carolyn and Jaeger, Christi. (1984). Teacher, Kids, and Logo. Irvine, CA: Educomp.

Rifkin, B. (1983, Jan.). Turtle folders help third graders. *National Logo Exchange*, 1-2.

Shimabukuro, G. (1988). *Thinking in Logo: A Sourcebook for Teachers of Primary Students*. Menlo Park, CA: Addison-Wesley.

The activities were developed with the Apple II family of computers and the Apple Logo programming language disk (LSCI). However, the computer activities contained in this book are easily adaptable to other versions of Logo and other computer systems.

Watt, D. and Watt, M. (1986). *Teaching with Logo: Building blocks for learning*. Menlo Park: Addison-Wesley. Watt, M. (1986). *Welcome to Logo!* LCSI Primary Level. Lexington, MA: D.C. Heath and Co.

Dr. Leslie F. Thyberg
5637 Rippey Street
Pittsburgh, PA 15206
AppleLink ALS 038

Assessing Logo Learning In Classrooms

VII: Planning, Carrying Out and Completing a Logo Project

by Dan Watt

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

This month I will write about each of the subheadings included in the seventh cluster of critical aspects, Planning, Carrying out and Completing a Logo Project. For a fuller sense of what we mean by critical aspects of Logo learning, and our rationale for this approach to assessing Logo learning, please read the September '88 column in this series.

Supporting a Logo Programming Process

One of the fundamental principles of Logo's educational philosophy is that students will learn mathematics, computer science and problem-solving, by engaging in projects of their own choice, and working through them in their own ways — with support from peers and teachers. I find this last point crucial. I have come to believe that support for all phases of project development, from the initial choice of the idea, to final publication and sharing of the results, is an essential ingredient of an effective Logo learning environment.

Molly Watt and I have developed a working description of what we call a "Logo programming process." We have observed that the development of a typical Logo project goes through several distinct phases: brainstorming and exploration; choosing a project; planning and simplifying the project; creating a working draft; debugging, revising and elaborating; sharing through publication (Watt and Watt, 1987a, 1987b, 1989). Our description is influenced by descriptions of the writing process, articulated by Donald Graves and others (Graves, 1983), and used by thousands of writing teachers all over the world.

A schematic diagram of a Logo programming process is shown in Figure 1. Notice that the process one goes through in completing a project is not the linear one shown in Figure 1a. Rather, as shown in Figure 1b, the process is recursive, moving back and forth between different working modes as the project develops over time.

The Statue of Liberty Project

In this article I will show how two sixth graders, Heather and Joanie, progressed through several phases of project development to carry out one long-term project — drawing

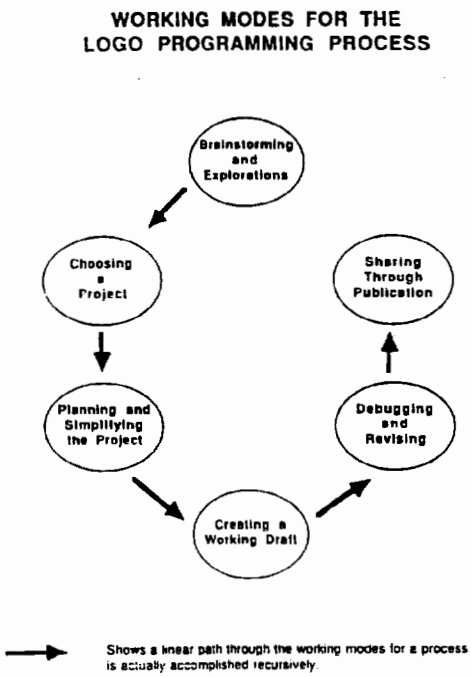


Figure 1a

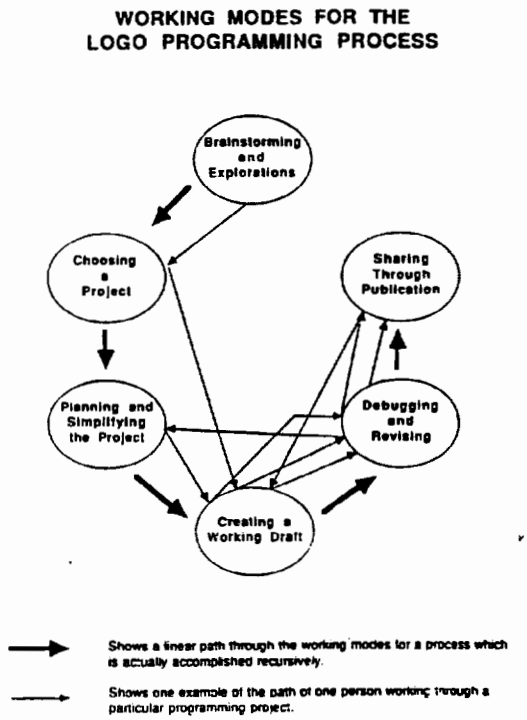


Figure 1b

the Statue of Liberty. By the spring of sixth grade, the girls had worked with Logo for almost three years. They were extremely comfortable with turtle graphics, with using variables to draw shapes such as rectangles, with dividing complex projects into small subprocedures, and with a structured planning approach that their teacher had emphasized. Our research data about their project includes their drawings; written plans for procedures; printouts of computer procedures, including some that were discarded; notes taken by an observer (a computer coordinator in their school district); and a copy of a published article they wrote about their project for a class newsletter.

One extremely interesting feature of this particular project is that half-way through it, the girls discarded their original plan, threw out all the programs they had written, and completed the project successfully using an entirely different approach than the one they had originally planned.

Developing and Completing a Project

A. Choosing a project and establishing goals

A1. Sketching the desired result

A2. Using a procedure hierarchy or procedure tree in planning

Heather and Joanie's teacher insisted that his students engage in a careful planning process, before starting to type their programs on the computer. Figure 2 shows their sketch of the Statue of Liberty, showing the elements that they planned to include in their program. Figure 3 shows their hierarchy chart (which I call a procedure tree) — their plan for a superprocedure and subprocedures which they anticipated writing to draw the entire figure.

Figure 2

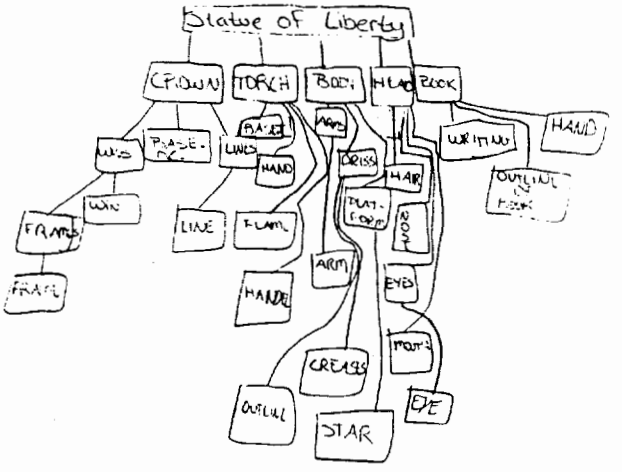


Figure 3

B. Establishing specific preliminary goals and plans

Heather and Joanie continued to work away from the computer. Figure 4 shows an elaboration of their hierarchy chart into specific procedures and subprocedures, and it also shows (at the top of the page) that they had changed their minds about the best way to actually construct the figure. Instead of starting with the crown, and working their way down to the base, they now decided that it would be easier to start from the bottom of the screen, and work their way up. At this point they identified some preliminary goals. First they

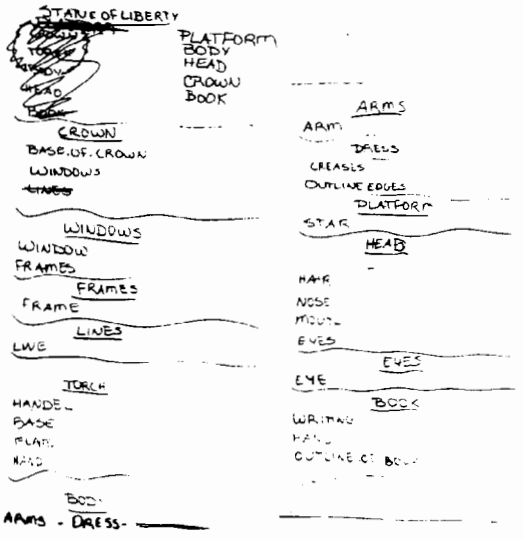


Figure 4

Assessing Logo Learning in Classrooms -- continued

planned to draw the outline of the entire figure, next the crown, then the torch and finally the book. Once they completed all these steps, they planned to elaborate the figure, putting in all of the details shown in their sketch.

C. Creating a working draft

C1. Working out the details of the plan

Heather and Joanie began working out and typing the procedures and subprocedures needed. They began with a superprocedure which they called SECRET, and they also wrote out all the main subprocedures as smaller superprocedures before actually filling in any details. Figure 5 shows their project at an intermediate stage of completion. All the major sub-procedures are written as shells, with the sub-subprocedures still undefined. Only the platform and body subprocedures have been substantially completed. And the girls expressed some concerns that the project was more difficult to complete than they had expected.

C2. Re-planning the entire project to incorporate a new idea

At this point, something unexpected happened. Heather and Joanie's teacher taught a lesson to the entire class, showing how to plot points on the screen using SETXY. He also taught his students to use a subprocedure, TO S, which allowed them to adjust the x and y coordinates of a completed figure separately:

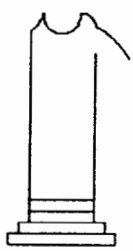
```
TO S :X :Y
SEXTXY (:X *17) (:Y * 14)
END
```

Students can change the numbers by which the X and Y inputs are multiplied to stretch their final drawings in the horizontal and vertical directions. The values shown above, 17 and 14, were chosen by Heather and Joanie for their Statue of Liberty project after a good deal of experimentation.

To Heather and Joanie, this new technique seemed made-to-order for the complex shape they were attempting to draw with the turtle. They literally abandoned all their existing procedures and subprocedures, and plunged into their new approach. Figure 6 shows how they started planning their drawing all over again, tracing a drawing of the statue of Liberty from a class magazine and transferring it to graph paper.

Then they worked out the coordinates for every element in great detail, interpolating between grid lines to estimate coordinates to the nearest tenth of a unit. Finally they wrote each procedure by hand before typing it at the computer.

```
TO SECRET
 PLATFORM
 BODY
 HEAD
 CROWN
 TORCH
 BOOK
END

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

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

TO BODY
 DRESS
 ARMS
END

TO DRESS
 HT RT 90 PU FD 20 PD FD 130 BK 130 LT 90 FD 60 RT 90 FD 150 RT 55 REPEAT 20 [FD 1 LT 2]
 SETH 90 SETH 180 REPEAT 19 [FD 3 LT 18] SETH 90 REPEAT 20 [FD 1 RT 4]
END

TO ARMS
 SETH 120
 REPEAT 15 [FD 3 RT 2]
END

TO HEAD
 HAIR
 NOSE
 MOUTH
 EYES
END
```



```
TO CROWN
 BASE.OF.CROWN
 WINDOWS
END

TO TORCH
 HANDEL
 BASE
 FLAME
 HAND
END

TO BOOK
 WRITING
 HAND
 OUTLINE
END
```

Figure 5

Figure 6

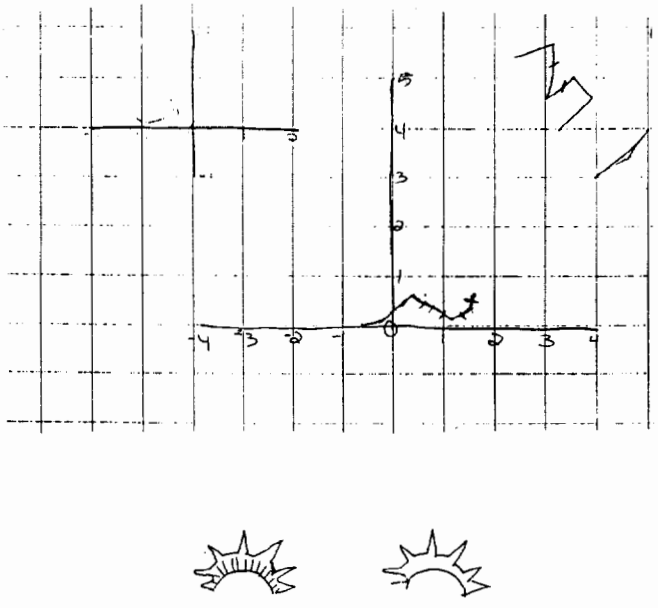
D. Completing, debugging and embellishing the project

As the girls later wrote in summarizing the process, "This was a difficult and very tedious task, but extremely rewarding. ... After we completed it, we did a lot of debugging." However, they were excited by the results, as each bit of the Statue emerged from procedures that were, in themselves, virtually unreadable.

Their final result (Figure 8), was still incomplete, but they were quite satisfied with what they had accomplished. An interesting feature of the particular approach their teacher had taught them — using the procedure, S :X :Y — was that they could not only draw the Statue in proportion, but could distort it and even reflect part of it about an axis, by varying the scale factor used to multiply the X and Y coordinates used in S (see Figure 9).

E. Publishing the results

Heather and Joanie's teacher required that each major Logo project be written up in a Newsletter format, as a way of publishing the results for other students and parents. This gave Heather and Joanie an opportunity to explain their approach in written form. The results, which were produced in printed form using the Newsroom software, are shown in Figure 9.



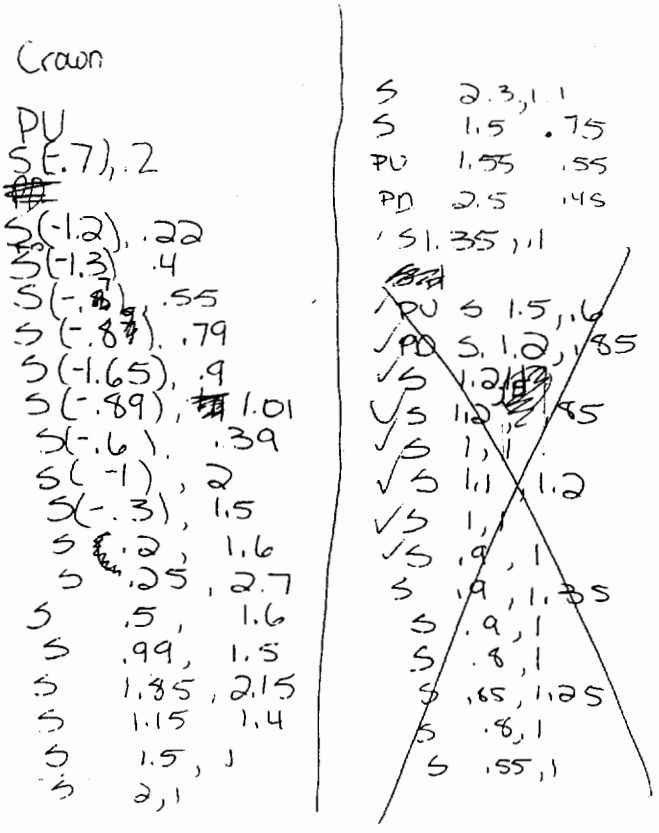


Figure 7

Assessing Logo Learning in Classrooms -- continued

```
TO TIP.TOP
 FULLSCREEN TORCH LHAND LARM CROWN EYES NOSE MOUTH HAIR NECK CHIN BOOK RHAND DRESS LOD
END
```

```
TO S :X :Y
 SETXY :X * 17 :Y * 14
END
```

```
TO TORCH
 HT
 PU S - 2.5 8.5
 PD S - 2.8 8
 S - 3 8
 S - 3.2 7.5
 S - 3 7
 S - 2.2 7
 S - 2 7.5
 S - 2.2 8
 S - 2.5 8.5
 PU S - 2.4 5.8
 PD S - 2 5.9
 S - 1.9 6
 S - 1.95 6.6
 S - 2 6.7
 S - 2.1 6.5
 S - 2.3 7
 S - 2.5 6.7
 S - 2.6 7
 S - 2.7 6.7
 S - 2.9 7
 S - 3 6.7
 S - 3.1 7
 S - 3.2 6.7
 S - 3.3 6.9
 S - 3.5 6.5
 S - 3.6 6.55
 S - 3.6 6.4
 S - 3.5 6
 S - 3.4 5.8
 S - 3.1 5.6
 S - 2.4 5.8
 S - 2.4 5.3
 PU
 S - 2.5 4.7 PD S - 2.5 3.7
 S - 3 3.6
 S - 3 4.5
 PU
 S - 3 5.2
 PD S - 3.05 5.7
END
```

Figure 8

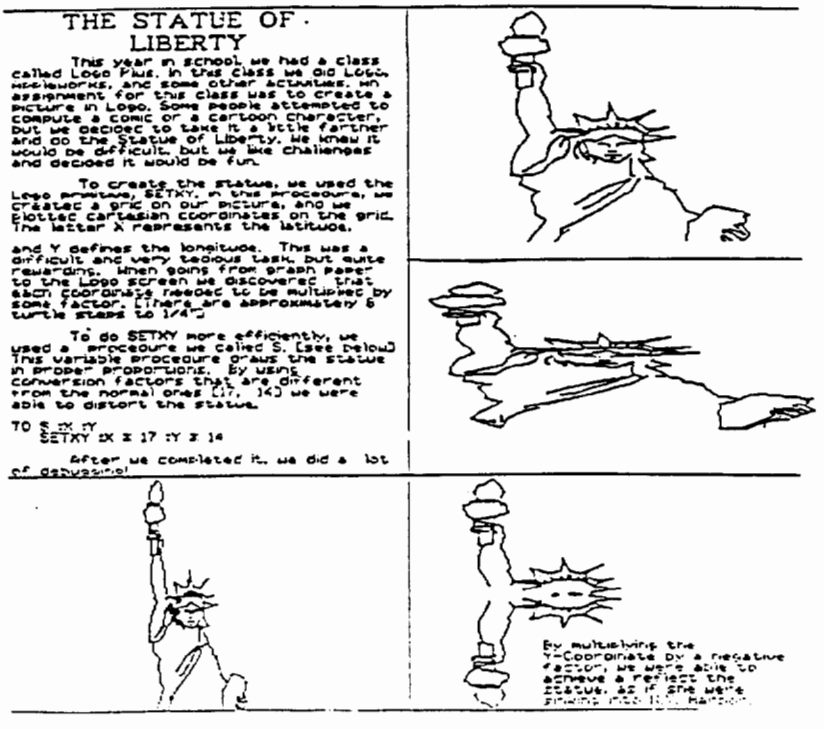


Figure 9

Some Concluding Remarks

I focussed on just one project because Heather and Joanie's opus illustrates almost all of the phases of the Logo programming process described above — and because such complete start-to-finish data was available. The highly structured approach encouraged by their teacher is certainly not the only way to support project development. And such an approach might be more appropriate for some students and some projects than for others. Nevertheless, I believe that by providing a framework which encouraged students to plan their work carefully, to keep records of all stages of the work, and to publish their results, Heather and Joanie's teacher enabled them to get as much educational value as possible out of their Logo experience.

Complete as it is, however, my story of Heather and Joanie's project development is missing an essential element. Projects don't typically come to life full blown. Our data does not tell us anything about how the girls came to choose the project they did, or about the exploration, brainstorming and messing around that preceded the work described here. Such preliminary thinking, comparable perhaps to the pre-writing phase of the writing process, is just as important as the phases we have described here, and should not be ignored in attempting to support students in project development.

Finally, I feel the need to say something about the particular approach Heather and Joanie used to draw their Statue. At first I was somewhat uncomfortable that conventional turtle graphics and readable Logo procedures were abandoned for the much less intuitive process of plotting points and connecting them. However, when I realized that the girls had been using standard turtle graphics effectively for almost three years, and I saw all the new types of mathematical thinking that this approach involved, and when I took into account the excitement that Heather and Joanie felt at using their new knowledge to create a result that they deemed superior to what they could have accomplished using **forward**, **back**, **right** and **left**, I was convinced. Even this ardent turtle-o-phile has to admit that there are times when coordinate systems can be just as powerful as turtles in producing excellent results and outstanding learning experiences.

References

Graves, D. (1983). *Writing: Teachers and Children at Work.* Heinemann, Portsmouth NH.

Watt, M. and Watt, D. (1987a). Polishing Programs for Publication: A Neglected Step in the Logo Programming Process. *National Logo Exchange*, May 1987.

Watt, M. and Watt, D. (1987b). Using the Process Approach to Teaching Writing as a Model for Teachers of Logo, in Hillel, *J. (Ed.) Proceedings, Third International Conference on Logo and Mathematics Education (LME3)*, Concordia University, Montreal Canada, July 1987.

Watt D. and Watt, M. (1989). Re-Thinking Logo Pedagogy Using a Process Approach, accepted for publication, *Proceedings, National Educational Computing Conference*, Boston Massachusetts, June 1989 (in press).

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

Dan Watt
Educational Alternatives
Gregg Lake Road
Antrim, New Hampshire 03440

Logo: Search and Research

Planning for Planning
by Douglas H. Clements

In our previous column, we saw that although students gradually come to plan their Logo programs, they often do not transfer this ability to other planning tasks. To develop planning skills it may be necessary to structure students' work with Logo so that they predict and plan before programming.

Studies have dealt with three specific problems students have with planning: Giving up on a plan, not having a strategy for planning Logo programs, and getting stuck.

Giving Up on a Plan

You have probably noticed, along with other researchers (Noss, 1984; Rampy and Swensson, 1984) that there are different types of Logo programmers. Some like to plan. They stick to their plan to produce the result they desire. Some give up and stray from their original plan (they "de-plan," according to Noss). They seem more interested in the process of doing turtle geometry and programming.

There may be several reasons students "give up" on a plan (Kull, 1986).

1. Students may not feel ownership of the problem.

It might not be important enough to them. If this is the case, certain instances of de-planning might have motivational origins. Or, it might be that children redefine a problem someone else gave them to be one that they wanted to solve. For example, an adult suggested to first grader Tommy that he draw an E. He spent the entire period drawing a right angle. To him, the problem was lining up the turtle to draw two perpendicular line segments. He verbalized his entire plan, including the corrections he had to make along the way.

This does not mean that children shouldn't be given offered suggestions, or that they must discover everything by themselves. However, they should be allowed to experiment with, modify, and incorporate an idea or procedure enough to "make it their own."

2. Students lack the tools to solve the problem.

This, of course, presents the teachable moment if the teacher is actively observing. For example, Jonathon had given up on drawing a series of orange bullets coming out of his gun (this problem was not posed by his teacher!). He was taught the REPEAT command and the gun quickly shot bullets (e.g., REPEAT 5 [PC 4 FD 5 PC 0 FD 5 yielded

— — — — — ▶

3. A more interesting problem presents itself.

This is especially likely when students work together. For example, Andrew wrote a procedure for drawing an angle. He ran it five times, producing a star. Then he reproduced the star 25 times. He then posed the problem of drawing a stem to add to these petals. This was difficult, but was accomplished with the help of a friend, who agreed with Andrew's interpretations along the way.

As with the other two reasons, this implies that we teachers need to be sensitive to students' planning activity. In addition, we need to strike a balance between encouraging students to:

persist on accomplishing a plan vs.
pursue interestings paths;

formulate their own problems vs.
solve presented problems;

make detailed plans in advance vs.
work in a more intuitive, "seat of the pants" approach.

A Strategy for Planning Logo Programs

Presented with the admittedly complex task of programming, students often find it difficult to formulate a basic planning strategy. The following approach has proven helpful to such students in several research projects (Clements, 1986, in press). From the beginning, the teacher helps students break their problems down into manageable pieces. Then the class develops a general planning strategy; for example:

1. Make a Creative Drawing (a free–hand picture). Remember to keep it simple and label its parts.

2. Make a Planning Drawing.
   - Use the planning form (paper turned "right 90" with the turtle at HOME).
   - Draw the turtle where it starts the procedure
   - Have turtle end in the same location and same heading at which it started ("state transparent"; see below).
   - Label each line, turn, or procedure.
   - Use the ruler and protractor to measure line segments and angles.
   - Show the "moves" with the pen up (or seams) as dotted lines.
   - For each new procedure that needs to be written, make a whole new planning form (i.e., start at the beginning of step 2 for each new procedure).

3. Have one partner read the instructions in order as the other records them at the right–hand side of the planning sheet.

4. Type them into the computer.

5. Debug each procedure separately, then the program as a whole (specific ideas for debugging will be described in an upcoming column on "monitoring solution processes").

Getting Stuck

When I teach, I confess I have the most difficult time with students who get stuck. I usually give them too much too soon. What should you say first? What next? A useful sequence of questions was successfully used in a research project by Perkins and Martin (1986). There are three types of questions that are asked in order, starting with the most general:

1. **Prompts.** Prompts are general strategy questions that one might ask oneself. "What do you have to do at first?" "What do you need to do next?" "Are there any other ways to...?"

2. **Hints.** If a couple of prompts are not enough, try hints. Hints are leading questions or tiny bits of information that suggest a strategy. "Doesn't that remind you of a problem we did yesterday?" "Do you think making a drawing would help?"

3. **Giveaways.** If several hints are not successful, try a giveaway. Here you provide a specific solution idea. (That's the one I too often start with!)

Perkins and Martin found that prompts led to a successful resolution of difficulties from 32% to 55% of the time. Hints added another 17%. Giveaways were required for the rest. This suggests that young students can benefit from general strategic questions—questions they could learn to ask themselves! It also suggests that jumping too soon to hints, much less giveaways, could be a grievous error. It may rob students of the opportunity to put the pieces together themselves.

To be more specific, here are some prompts, hints, and giveaways for Logo.

•Prompts:

What's the first thing you need to tell the computer to do?

What does this command (e.g., RT 90) do?

Are there any other ways to make the turtle...?

What instruction (or procedure) should go next?

What will this procedure really do?

Why did my procedure do this, instead of what I intended?

•Hints:

Can you think of a command to get the turtle to move without drawing a line?

Your problem is to repeat something 12 times. Do you know a command for that?

Could you use a variable in the procedure somehow?

•Giveaways:

Try PENUP then FD some amount.

Type REPEAT 12 [FD 10 RT 30].

One valuable way to present information as giveaways to students (in the context of their own work) is as a good trick or some good advice. For example, the use of variables, REPEAT, or recursion might be seen as good tricks; writing only state transparent drawing procedures as good advice.

Such research-based suggestions can help students plan their Logo projects. It's important to remember to help them generalize their new-found strategies (see the previous column for additional suggestions). That's another reason to stress the general prompts: These are the most likely to transfer to other situations. For any planning strategy, similarities and applications in other situations should be discussed.

References

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

Clements, D. H. (in press). Metacomponential development in a Logo programming environment. *Journal of Educational Psychology*.

Kull, J. A. (1986). Learning and Logo. In P. F. Campbell & G. G. Fein (Eds.), Young children and microcomputers (pp. 103-128). Englewood Cliffs, NJ: Prentice-Hall.

Noss, R. (1984). Children learning Logo programming. Interim report No. 2 of the Chiltern Logo Project. Hatfield, U.K.: Advisory Unit for Computer Based Education.

Perkins, D. N., & Martin, F. (1986). Fragile knowledge and neglected strategies in novice programmers. In E. Soloway & S. Iyengar (Eds.), *Empirical studies of programmers*. Norwood, NJ: Ablex.

Rampy, L. M., & Swensson, R. (1984, April). The problem-solving style of fifth graders using Logo. Paper presented at the annual meeting of the American Educational Research Association, New Orleans, LA.

Douglas H. Clements, SUNY at Buffalo
Department of Learning and Instruction
593 Baldy Hall, Buffalo, New York 14260
CIS: 76136,2027 BitNet: INSDH@UBVMSA

Object-Oriented Programming: Initial Experiences.

by Robs John Muir

During the past several years, there has been a heightening interest in object-oriented programming (OOP) languages. The principal reason for the increasing popularity of this programming paradigm is the availability of such languages for micro-computer implementations. This has been fueled by the ever increasing power of modern PCs and the concomitant complexity of these programming environments. This complexity demands a set of programming tools which allows a programmer to isolate and modularize data structures with greater specificity than simple procedures allow; modularity helps reduce the bug count during the development of complex programs.

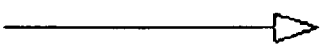
Serious programmers have been playing with object-oriented programming in other languages for years. SmallTalk, Lisp, C, and Pascal users have OOP extensions available in certain versions of these languages; Logophiles have been left out of the score. While Logo has made some overtures in this direction—LogoWriter's multiple turtles, for instance—there has been no concerted interest within the Logo community, until recently. With the introduction of Object Logo for the Macintosh, by Coral Software, Inc., Logo users now have a powerful implementation of OOP which rivals the best of these software tools.

What is OOP? How are these ideas integrated into traditional Logo programming, and what advantages of OOP for education? OOP is currently being used by Claremont High School in California for our introductory computer courses. This article is a reflection of our experiences thus far.

What Are Objects?

Assume a computer lab containing several objects. For the time being, let's call these objects *computers*. Each of these different computers is running Logo and each has its own workspace containing different Logo procedures with different variables. They even have different looking turtles. Some of the objects in this lab "know" how to draw some nifty looking designs.

One of the objects in this lab is your own personal computer. It has some neat looking designs of its own—after all...you defined them! When you type RIGHT 90 FORWARD 50 on your object's keyboard, a turtle, shaped like a triangle, draws this



When you "ask" another computer in the lab to RIGHT 90 FORWARD 50, a diamond-shaped turtle produces this result

On a different computer, RIGHT 90 FORWARD 50 produces a still different result

Each of the objects in this computer lab is behaving very differently even though you've given them identical commands. Each has its own procedures, variables, turtle-state, and "primitives." Some object's turtles may draw with different pen widths, colors, or patterns; this is entirely dependent on how the object was defined. Since each object in our fictitious lab is a separate computer running Logo, each computer contains different procedures. Notice that each object "knows" how to FORWARD and to RIGHT—though some versions of FORWARD may result in unusual behavior.

In the object-oriented world of our computer lab, if you see some interesting behavior or design, you merely say MAKE "MY.COMPUTER ONEOF THOSE. Suddenly, your computer can draws waves (or thick stripes, depending on which object you prefer). You need not retype all of the procedures which were required to get *that computer* to do *that* thing. Your computer is said to have *inherited* that skill when it became ONEOF those. Your computer still can do the things it originally could—it has now incorporated *that computer's* behaviors. In essence, your computer now contains two objects. It is now two computers in one box.

You can "talk" to these different computers in the same way you might communicate with friends—you send messages. When you type

```
ASK :WAVER [REPEAT 4
   [FORWARD  50 RIGHT 90]]
```

you will see a wavy-sided box. If you type

```
ASK :MY.COMPUTER
   [REPEAT 3 [FORWARD 50 RIGHT 120]]
```

your screen will display an equilateral triangle composed of straight lines. :MY.COMPUTER knows how to FORWARD; :WAVER'S FORWARD is different. While we are talking to :WAVER, this version of FORWARD is said to shadow (or take the place of) :MY.COMPUTER's. :WAVER is a modular *frame* or environment that contains a distinctly different set of procedures.

The OOP Environment

Object-oriented programming (OOP) provides an environment where, in a single computer, multiple computers can

coexist. In the parlance of OOP, an object is a modular entity composed of both **procedure** and **data**. Objects may have their own data and procedures, or they may inherit such "knowledge" from their parents. Objects may also inherit from more than one other object. An important point to grasp is that each object exists in its own right.

Proponents of object-oriented programming contend that OOP is a natural extension of the real world—a world populated with objects. Advocates argue that OOP encourage a more natural style of programming which encourages programmers to modularize and control complex programs by limiting the scope of procedures to a local area—the object.

There is some basis for such a belief. In the real world, we often use objects without needing to know how they actually function. My car has three pedals. A push causes each of them to operate. When I push the accelerator, my car performs in a certain way. It behaves very differently when I push the brake or the clutch. Yet, for each of them, the message is still the same—**PUSH!** Similarly, :WAVER draws wavy lines. Its behavior is different from :MY.COMPUTER's FORWARD. I might want to use :WAVER in a programming project; I need not know how :WAVER does it. It is simply enough to ASK :WAVER [FORWARD 50].

Early work on languages built around this object-oriented paradigm is credited to Alan Kay with his work on *Simula*. Interest in *SmallTalk*, another OOP language developed by researchers at the Xerox PARC in the 70's, encouraged others to experiment with this new style of programming. *Loops*, and *Flavors* (each a dialect of *Lisp*), further extended OOP into mainstream computing. OOP has been further popularized by Apple's HyperTalk language used in HyperCard. While not a true OOP language (one cannot create new classes of objects within the language), HyperTalk does encourage a modular programming style.

Coral Software currently markets a version of Logo for the Macintosh called *Object Logo*; this powerful extension of Logo is based on Q-Logo—developed at the Cambridge Atari Research Center in the early 80's. Object Logo is arguably the most powerful implementation of Logo available for personal computers; in addition, this language closely follows LCSI's Apple Logo yet offers OOP capabilities which exceed those of other more widely know OOPs.

Teaching Object Oriented Programming

Object-oriented programming is of interest to educators for several reasons. First, OOP may help reduce the errors made by naive programmers as they classify components of a program. Second, OOP supports the development of computer-based simulations. Additionally, object-oriented programming encourages beginning programmers to work in a style which has been empirically shown to be used by professional programmers. Finally, OOP languages are gaining wide acceptance in modern programming communities; educators need to participate in the development of appropriate software tools in order to evaluate their effectiveness and utility for new learners.

There is some question about how (or whether!) object-oriented programming should be integrated into a modern curriculum. Little formal research has been done in this area and the literature offers few examples of OOP pedagogy. Based on our experience with Object Logo at Claremont High School (Claremont Unified School District, CA), we would argue that OOP is learnable, teachable, and eminently useful in modern computing. The concepts are transferable to different environments and the results extend the expressive power of the student who uses them.

Our approach to introducing students to OOP is modeled on Gary Drescher's tentative outline of OOP pedagogy—Drescher was instrumental in formalizing the syntactic form of Object Logo. A feature of this approach the gradual way in which students can acquaint themselves with objects. The sequence is as follows.

1. Introduction to simple Logo expressions without objects.
   Object Logo is used initially to introduce students to procedural programming—since Object Logo offers strong syntactic similarities to Apple Logo™, this provides an (often) familiar link to earlier experiences.

2. Use predefined objects.
   Multiple turtles, each with its own behaviors, are incorporated into student projects. Students can freely choose from prefabricated classes of turtles. Some choices of classes of turtle objects include: noisy turtles, wavers, turtles with different line widths or pen patterns, turtles with different HOMEs, turtles with different shapes, "typing" turtles, broken turtles, 3-dimensional turtles, etc.

3. Make and use instances of an existing class.
   Students can create several copies of a particular type of turtle for appropriate projects. Several wavers can interact with each other.

4. Customize existing instances.
   Students can modify existing instances so that specialization becomes a distinguishing feature. FORWARD and RIGHT commands may cause an object to behave differently when they shadow a parent object's procedures.

Drescher goes on to suggest additional steps in a teaching sequence. Included in this sequence are such instantiation and multiple inheritance. However, at this time, we have not had time to incorporate these more advanced ideas into our curriculum.

Object Oriented Programming -- continued

In addition to learning about object-oriented programming using Object Logo, our students are also introduced to concepts of shadowing and modularity using HyperCard. While providing a rigid hierarchy of objects, HyperCard does a excellent job of concretizing the role of shadowing (sometimes referred to as masking). Buttons on different layers of a card can demonstrate in a very visual way how messages can be intercepted and interpreted differently by different objects.

The issue of preparation and environment is of crucial importance to the teaching/learning process. This is especially true in the the computer laboratory. There is no substitute for qualified, involved, and experienced professionals—professionals who have *direct* experience with a wide range of programming experiences. No amount of simplified computer experience or superficial literacy will supplant the time and discipline required for a student be become truly computer literate. Our role, in a fast-changing discipline, is to participate fully in the design, implementation, and evaluation of emergent technologies.

Object-oriented programming represents one of many possible alternate futures for educational computing. We are excited with what we see.

References.

Adelson, B. and Soloway, E. (1985). The Role of Domain Experience in Software Design. *IEEE transactions on Software Engineering*. SE-11(11).

Drescher, Gary L. 1987. *Object-oriented Logo*. Coral Software, Inc. 16-18.

Goldberg, A. 1978. SmallTalk in the Classroom. *Xerox Palo Alto Research Center Technical Report*.

Goldberg, A. andKay, A. (1977). Methods for Teaching the Programming Language SmallTalk. *Technical Report SSL 77-2*. Xerox Palo Alto Research Center.

Goldberg, A. and Robson, D. (1983). *SmallTalk-80: The Language and Its Implementation*. Addison-Wesley.

Guzdial, Mark (1988). *Use and Design of an Object-Oriented Logo*. Dissertation proposal. University of Michigan.

Jones, Jeremy A. (1985). An Object-Oriented Extension to Logo. *Logo '85 Pre-Proceedings*. July 22-25, 1988. Massachusetts Institute of Technology. Cambridge, MA. 127-128.

Kay, A.C. (1972). A Personal Computer for Children of All Ages. *Draft. Xerox Palo Alto Research Center*.

Muir, Robs (1987). ObjectLogo: A Review. *The Logo Exchange*. Vol. 6(9). 23-26.

Muir, Robs J. 1988. InLXual Challenges: Turtle Waves. *The Logo Exchange*. Vol. 6(9). 23-27.

Robs Muir, Claremont High School
1601 N. Indian Hill Blvd., Claremont, CA 91711

European Logo Conference

Place: Faculty fo Pedagogical and Psychological Sciences, The State University of Gent, Gent, Belgium
Dates and times: 1600H on 30 August to 1300h on 1 September 1989
Deadline for registration: 30 April 1989
Deadline for proposals: 30 April 1989
Deadline for full version: 30 May 1989
Outline of full programme will be mailed in June 1989
Language: English
Programme: The Eurolo 1989 themes reflect a wide variety of topics:

- Classroom experiences with Logo; Logo and the school curriculum: Primary and Secondary education, Higher education and special education.
- Teacher training in relation to Logo and Logo use
- Logo related languages: programming environments, microworlds, new developments.

Keynote: Dr. U. Leron (Israel)
Speakers: Dr. R. Noss and Dr. C. Hoyles (UK); Dr. M. Rosello (Spain)
Organisers: Prof. G. Schuyten and Dr. M. Valcke
Address: Eurologo 89, State University, Gent, Department of Education, EDIF, H. Dunantlaan 1, B-9000 GENT, Belgium
Phone: (91) 25 41 00 extension 342 or 354
Telex: 12754 rugent b
E-mail: EDIF@BGERUG51
Fees (in Belgian francs)

	Before 30 March	After 30 March
Residential	8000	9000
Non-residential	3500	4500
Students: residential	6500	7500
non-resitendial	2800	3800

If booking at student rate, send the number of your student identification card

Format for proposals:

To submit a poster session, send 2 copies of a 3-page normally spaced summary, 40 lines per page A4, and 2 copies of a 200 word Abstract.

To submit a paper presentation (40 minutes) send 2 copies of a 3-page normally spaced summary, 40 lines per page A4, and 2 copies of a 200-word abstract.

Any proposal should include

- the theme(above)
- the title of your contribution
- the author(s)
- the institution(s)
- your mailing address and e-mail address

Abstracts and proposals should be in English

You will be advised by the programme committee about acceptance within 3 weeks'time. The deadline for sending a more elaborated version is 30 May 1989.

Global News

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

Logo Exchange Continental Editors

Africa
Fatimata Seye Sylla
Lab Informatique et Ed
BP 5036 Dakar
Senegal, West Africa

Asia
Jun-ichi Yamanishi
Faculty of Education
Toyama University
3190 Gofuku
Toyama 930 Japan

Australia
Jeff Richardson
School of Education
GIAE
Switchback Road
Churchill 3842
Australia

Europe
Harry Pinxteren
Logo Centrum Nederland
P.O. Box 1408
BK Nijmegen 6501
Netherlands

Latin America
Jose Valente
NIED
UNICAMP
13082 Campinas
Sao Paulo, Brazil

It is the time of year for spring cleaning and this year my son and I attacked a cleaning job never before attempted in the Harper household — cleaning up floppy disks. The one-day job turned into three 16-hour days as we attacked over 1200 Apple II and Macintosh disks. [...and **you** think **you** have a lot of disks! Editor]

With the help of my 13-year-old son we had to throw away nearly 350 disks because of mildew. Having traveled around the world several times and living for years in places like Malaysia, Singapore and Finland, the elements finally took their toll on the media. Gone forever were VisiCalc, Apple PILOT, Apple Pascal, Programmable Turtle, Delta Drawing, Apple Writer, Scott Adams Adventures, YPLA MIT Logo Demo, DBASEII, and most of my Softswap disks. Age, brand of diskette, copy or original - there seemed to be no pattern as to which disks mildewed. Anyway, by reusing, consolidating and eliminating we are down to a manageable 700 disks all cataloged on my son's new HyperCard inventory system that says the name of the program when the card appears on the screen. He says his Logo training really has helped him learn HyperCard scripting.

* * * * * * * *

Our European correspondent sends us a transcript of a speech given by Hannu Korhonen at the Computers in Education conference held recently in Tallinn, Estonia, USSR and sponsored by the Soviet Academy of Science's Institute of Cybernetics. It was the only speech to address Logo and is interesting in many ways, including the use of Logo primitives and procedures in local languages. Mr. Korhonen works at the Institute of Educational Research in the University of Jyväskylä in Finland.

AI Languages in School Mathematics

by Hannu Korhonen

You learn mathematics by doing, not by reading. Today you need efficient tools for doing mathematics. Pocket calculators and computers do numerical calculations and in some years symbolic mathematics programs such as MuMath and Macsyma will do algebraic operations as well. How can you help yourself at school to do mathematics like a real mathematician?

Usually words, symbols and graphs are the objects by which you think. If you want to use a computer as a tool in mathematics, you need a language both you and the computer can understand. It is not very easy to decide which kind of language is useful at school. There is a sensitive balance between the efficiency and the easiness of the language. At school you need a language which is easy to begin with and which is close to your own language.

Later on you can perhaps use a concise language such as APL where you need to learn many specific symbols. You can also use PROLOG where you need a lot of logic in programming. But in the beginning the Soviet Union would be wise to use an easy-to-begin and easy-to-access language such as Logo.

The structural basis of Logo is the same as LISP's but the syntax is simpler. It is quite easy to create mathematical objects by naming them and by giving them the necessary properties. However, the Logo language is too difficult for the beginner if he or she wants to learn mathematics to solve problems — not learn to program.

Data Abstractions and the Mother Tongue

There is a great difference between using a variable PI which has a value 3.141 and creating a procedure Pi

```
TO PI
LOCAL "TERMS MAKE "TERMS 100
   ; THE NUMBER OF TERMS IN A
     POWER SERIES
PI1  1  2
OUTPUT SQRT 6 * :TERMS
END
```

Global News -- continued

```
TO PI1  :SUM  :IND
IF :IND = :TERMS
   MAKE :TERMS  :SUM  STOP
PI1  :SUM + 1 / (:IND *  :IND)
   :IND + 1
END
```

where you can see how the figure is arising and decide how precise the approximation will be. The former is learning by reading, the latter is learning by doing. In some Logo versions there is a ready-made procedure PI so you do not need to program it yourself. The self-made procedure is naturally much slower and thus technically less adequate, but it is pedagogically superior to the ready-made one.

There are also two other important aspects in naming objects. The first is data abstraction. You not only have a number or another object and a technical way to point at it, but you have a word for your object. This is analogous to having a concept in normal language. There you have a more or less obscure meaning of a concept and a name for it. In a programming language you have a procedure as a meaning and its name.

As an example of a technical way to point at an object, there is a command (ROUND 100 * PI) / 100 for a two-decimal approximation of PI. The data abstraction alternative is a procedure DECIMALS which you can call by DECIMALS 2 PI for the same purpose.

```
TO DECIMALS  :DECDIGITS  :NUMBER
OUTPUT FORM  :NUMBER  1  :DECIDIGITS
END
```

The second aspect is naming concepts or objects in your own mother tongue. It is very important that you can think in your own language. For example in Finnish you must have a procedure PI1 with two i's for English or Logo PI. DESIMAALIT for DECIMALS and KERTOMA for FACTORIAL. As object and function names you can use ordinary mother tongue words even if the primitive names are English-like Logo words.

Functional programming is a very useful tool. If your procedure is a function, you can use the procedure name as a name for an object instead of writing defining calculations formulae. For example the 20th Fibonacci number is 6765. If you wish to use it in an expression, you do not need to calculate it first and then write it down, but you can use its name as an object. The sum of the 10th and 20th Fibonacci numbers is F 10 + F20 = 6820. The operation F is not linear because F30 is 832040.

The difference between Logo versions can be very great in this aspect. The greatest whole number in Commodore Logo is about two billion or ten decimal places. Using IBM Logo you can use numbers up to one hundred decimal places. The restrictions are not technical any more, but logical and depend on human thinking capability.

Microworlds

It is a general misconception that Logo is merely something for drawing simple geometric figures. Turtle graphics is a very clever tool in geometry, but the LISP based Logo is also a very efficient tool in processing numeric and symbolic information. School mathematics is often so uncomplicated that you do not need a modern object-oriented Logo or LISP version and any version of Logo will do.

Logo is a possible tool for ambitious goals. You can use it as a programming language when creating learning environments. For example, in a geometric microworld you can define basic objects such as points and lines, get them on the screen, and then define new objects such as line segments and triangles using the existing ones. The objects in the number world are numbers. The procedures are functional definitions of different kinds of numbers. In addition to definitions, there can be procedures to study the properties of numbers.

The difference between computer environments and paper-and-pencil ones is that on paper the figure is the object and you must imagine everything else. In a computer environment you can have the figures of objects on the screen, but the objects really exist in the memory of the computer. You can clear the screen and then call a defined object back by name.

An environment does not need to be a very demanding project in the very beginning. You can start with small procedures which have been written for a special topic in the curriculum. When you and your students are using, planning, studying and altering procedures you will learn mathematics by doing.

This kind of an environment is different from a usual computer-aided educational program. It is quite open enabling the student to create new objects and relations using Logo commands after he or she has become acquainted with the ready-made ones and will have new tools for more advanced problems. Naturally there are no other restrictions for environment topics other than the imagination of the teacher and his students.