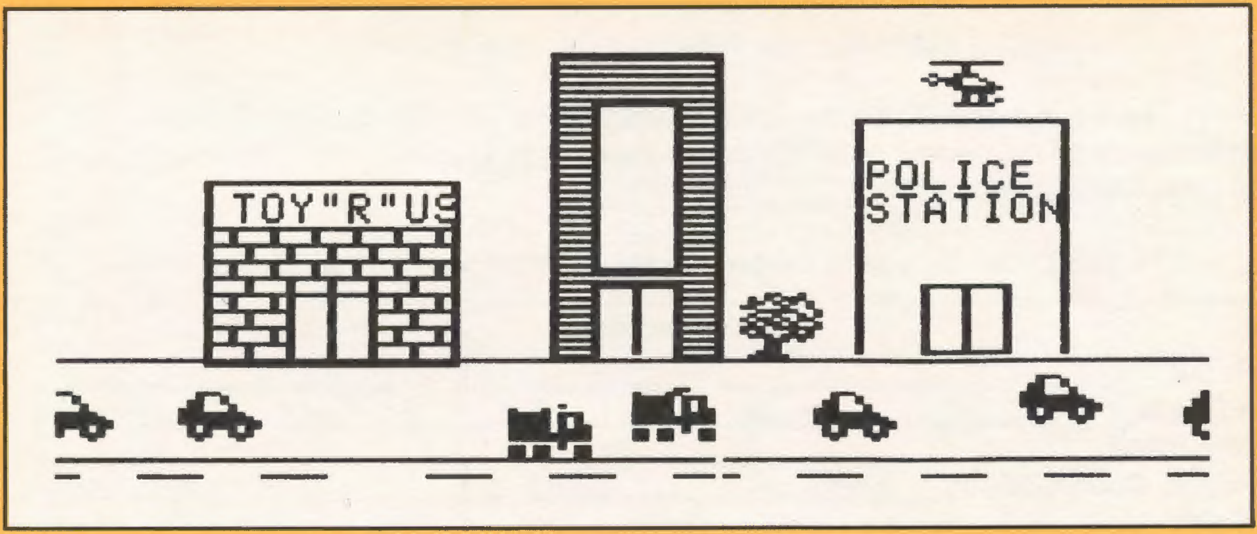
Journal of the ICCE Special Interest Group for Logo-Using Educators

LOGO EXCHANGE

FEBRUARY 1988 VOLUME 6 NUMBER 6



International Council for Computers in Education

INTERNATIONAL COUNCIL FOR COMPUTERS IN EDUCATION ICCE Publications

Non-Profit
Organization
U.S. POSTAGE
PAID
Permit No. 63
Eugene, OR

LOGO EXCHANGE

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

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

CONTENTS

SIGLogo Membership (includes *The Logo Exchange*)

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

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

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

From the Editor

by Sharon Burrowes Yoder

In this issue of *LX* you will find the bylaws of the SIGLogo. (Remember, long time *LX* subscribers that your *LX* subscription now carries membership in the ICCE SIGLogo!) Take a moment to look over these rules that govern this organization, and think about how you might like to contribute: by being an officer, by submitting articles, by sending letters to the editor...

Interwoven into this issue are a couple of, what I consider to be important threads in the ongoing Logo culture. First, we see a broadening of Logo beyond typical activities such as drawing polygons. Some of these articles barely mention Logo, at least at the outset, and yet are clearly about Logo-like thinking. Glen and Gina Bull use Logo to explore sound (largely without TONE), Tom Lough takes a look at HyperCard, Doug Clements examines algebraic concepts in some depth, while Sandy Dawson discusses the work of Gattegno, a mathematics educator influenced by Piaget.

A second thread is examination of Logo in more depth. Robs Muir takes a look at some specifics of Logo grammer, while Gary Stager explores the idea of predicates. If you are a relative novice, you may find these articles a bit overwhelming. Don't be discouraged, however. Put them away for a few months; then come back and see how much more you understand!

If Logo is going to continue to grow, we need both depth and breath. We need more microworlds like Glen and Gina give us; we need more research such as Doug reports; and we need discussions of underlying principles such as Robs has given us. I hope this issue, as well as others this year, have and will help you to grow in your understanding of Logo. I certainly never cease to learn more about Logo and to be amazed as it's flexibility. Hopefully the same is true of you!

Sharon Burrowes Yoder, ICCE / SIGLogo, University of Oregon, 1787 Agate Street, Eugene, OR 97405-9905

From the President

Why Did The Chicken...
by Peter Rawitsch
Acting-President of SIGLogo

At least once a month, the cafeteria at my elementary school offers a menu that includes chicken noodle soup and an egg salad sandwich. It always reminds me of the age old question about chickens and eggs. My role in the "birth" of SIGLogo raises an equally puzzling question. Last December I met with the SIGLogo Board of Directors to create the SIGLogo Bylaws that create the SIGLogo Board of Directors. Which came first? In this timeless tradition of recursive thinking, I welcome you to the Special Interest Group for Logo Using Educators.

This has been a challenging and rewarding year for SIGLogo. The challenge has been to let the international Logo-using community know we exist. The *Logo Exchange's* publication schedule for the fall and winter was very disappointing for all of us. As it begins to get on track, memberships, in the form of *LX* subscriptions, continue to grow. The reward has been the support we have received in the form of articles from faithful columnists and from the dedicated staff at ICCE. I am especially grateful for the advice and cooperation I have been given by Sharon Burrowes Yoder, the *LX* editor, and Keith Wetzel, the Special Projects Coordinator.

The next step for us will take place at NECC '88 in Dallas, Texas. SIGLogo will hold its first general business meeting on June 15th from 7 - 9 P.M. Look for announcements as to location on general bulletin boards and at the ICCE booth in the exhibit hall. The meeting will be led by Gary Stager, Acting-Vice President, Ted Norton, Acting-Communications Officer, and myself. At that time, members will have an opportunity to discuss and ratify the Bylaws. (A majority of the members who are present at the meeting will be needed for ratification.) We will also consider the future of SIGLogo. The establishment of a SIGLogo bulletin board and the sponsorship of a Logo conference are among the possible topics.

In addition to your ideas, we will be looking for volunteers. We need someone to serve as Acting-Treasurer for the coming year. The Logo Exchange is always in need of samples of students' work and articles about interesting ideas that can be used in the classroom. A Nominations Committee will also be appointed to recommend candidates for the April, 1989 election of Vice President, Communications Officer, and Treasurer. As prescribed in the Bylaws, Gary Stager will serve as SIGLogo President beginning at next year's NECC conference.

Gary Stager, Ted Norton and I look forward to meeting you at NECC '88 and to working with you in the coming year. SIGLogo is at the crossroads. Come and help us get it to the other side.

Peter Rawitsch, P.O. Box 254, Guilderland, NY 12085.

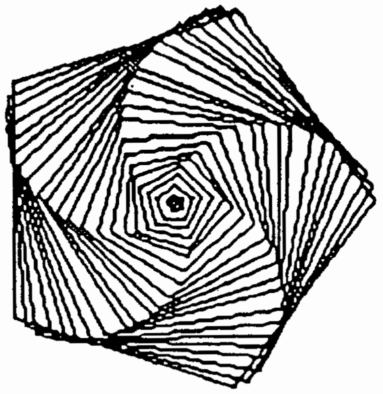
Cover: Freddy Victoria, The Allen-Stevenson School, New York, N.Y. Freddy did this sample in grade 4 upon first encountering LogoWriter. The assignment was to make a street scene, making use of stamped shapes, filled or shaded shapes, and label text. Later the class used their street scenes and programmed moving vehicles with keys to provide control over the speed.

Monthly Musings

HyperHype
by Tom Lough, Founding Editor

Recently, the prefix *hyper* has taken on a couple of new meanings for me. For example, our four and one-half year old son graciously shared some of his *hyper* behavior on several occasions over the holidays. The dizzying frenzy of his antics certainly created a memorable definition, needless to say.

Another meaning is embedded in the familiar looking graphic below.



Almost instinctively, one is tempted to think:

```
TO POLYSPI :SIDE :ANGLE
etc.
```

If you examine the figure carefully, however, you might hesitate. The figure was not drawn by a turtle going forward and turning left in a changing manner. Rather, the figure consists of a number of regular pentagons nested inside each other and slightly rotated with respect to each other.

The figure was drawn with the paint tools in a new application for the Macintosh called *HyperCard*. Many of you have probably seen or heard of it by the time you read this. As in the figure above, I believe we can observe something from more than just a quick examination of *HyperCard*.

During my first encounter with *HyperCard*, I explored the main ideas. The screen became a *card* with could contain text or graphic information and buttons. The cards could be grouped in *stacks*. In fact, the stack seemed to be the main metaphor.

I enjoyed browsing through address books, to-do lists, and pages of artwork. It was fun to click the various buttons and discover what action followed.

But it was not until I acquired *The Complete HyperCard Handbook* by Danny Goodman (Bantam 1987, ISBN 0-553-34391-2, $29.95) that I really began to understand what *HyperCard* was all about. Just as the second look at the "spiral" pentagon above may have given you second thoughts, Goodman's book gave me a glimpse of some exciting future possibilities.

In his book, Goodman quotes Bill Atkinson, the developer of *HyperCard*, suggesting the application is an authoring tool, and comparing it with Logo and other computer languages. (p. xxviii) On the same page, Atkinson also predicts that "education will pick it up right away," and suggests several interesting sample projects.

But more than Atkinson's comments, the thing which excited me about *HyperCard* was the HyperTalk language which accompanied it. Although definitely ***not*** Logo, this language is modular and extensible. Moreover, it was designed to enable ordinary users to create their own specialized applications, giving rise to the term "stackware." Suddenly, it is possible for you and me to write reasonably sophisticated programs for reasonably sophisticated applications for the Macintosh.

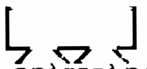
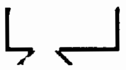
Sure, there are some (present) limitations. The minimum memory, for example, is one megabyte, with a hard drive recommended. You cannot create a card larger than the Macintosh Plus screen. You cannot look at more than one card at a time. But what *HyperCard* and HyperTalk can do within these limitation is both impressive and exciting.

As you might suspect, I sense many parallels between *HyperCard* and Logo. I'll not go into them here, however. Instead, let me invite you to explore HyperCard for yourself. As for me, I'd like to take up the challenge extended by Bill Atkinson to the education community. Let's "pick it up" and see what possibilities we can discover with the new tool. As always, I would be interested in your ideas.

FD 100!

Tom Lough, P. O. Box 5341, Charlottesville, VA 22905

Answers to this month's *InLXual Challenges* article, see page 18.
(Read upside–down and in a mirror.)

1. PADDLE, FIRST, LAST.
2. IF, REPEAT, IFELSE.
3. CS, HT, ST, PU.
4. 
5. POS, PENSTATE, XPOS.
6. 

Bylaws for SIGLogo

Special Interest Group for Logo-Using Educators (SIGLogo) International Council for Computers in Education

Article I - NAME

This organization will be known as the Special Interest Group for Logo-Using Educators of the International Council for Computers in Education. It is hereafter referred to as SIGLogo. The International Council for Computers in Education is hereafter referred to as ICCE.

Article II - PURPOSE

SIGLogo will support and promote the use of Logo as a tool for learning and as a language for computer science. Specifically, SIGLogo will:

a. Collect and disseminate information through publications and electronic communication networks.
b. Sponsor all or part of meetings, conferences, and workshops.
c. Organize working groups for research, study, and writing activities to meet the needs of its membership.
d. Develop and distribute position papers on Logo implementation and philosophy.
e. Serve as the Logo-using educators' liaison to the ICCE and its other Special Interest Groups.

Article III - MEMBERSHIP, DUES, and PRIVILEGES

Section 1: Membership

a. Individual members of ICCE are eligible for membership in SIGLogo upon application and payment of ICCE Individual Member SIGLogo dues.
b. Non-individual members of ICCE are eligible for membership in SIGLogo upon application and payment of non-ICCE Individual Member SIGLogo dues.

Section 2: Dues

The SIGLogo Board of Directors will determine the annual rate of membership dues for Individual Members and non-Members of ICCE. Upon approval of the ICCE Board, the dues rates will become effective on July 1 of each fiscal year.

Section 3: Rights and Privileges

All members of ICCE have the right to attend and to speak at SIGLogo meetings and at meetings of its Board of Directors. ICCE members also have the right to serve on committees of SIGLogo. The rights to hold office and to vote are reserved to SIGLogo members.

Section 4: Termination of Membership

A member may resign from SIGLogo by submitting a written resignatiom to the Communications Officer. Membership may be automatically terminated for failure by a member to pay dues. A member may also be expelled or suspended for reasonable cause after a hearing before the SIGLogo Board of Directors and by a majority vote of the SIGLogo Board.

Section 5: Membership Roster

The collection of dues and the maintenance of a membership roster will be administered by the ICCE. A copy of the membership will be provided to the ICCE Board upon request. The membership roster will be used for the ordinary business of the SIGLogo, such as mailing of meeting notices and ballots.

Article IV - OFFICERS

Section 1: Election of Officers

The Vice President, Communications Officer and Treasurer of SIGLogo are selected by a vote of the membership. The Vice President becomes the President at the end of the term.

a. The SIGLogo President will appoint a nominations committee to nominate candidates for the offices of Vice President, Communications Officer and Treasurer. Every two years, nominations for the ballot will be submitted prior to February 1 of the election year.
b. ICCE will oversee the election of SIGLogo officers. Vitae of the candidates, election ballots and voting instructions will be mailed to all members of SIGLogo or printed in the Logo Exchange no later than March 1 of the election year. Members will cast their votes no later than April 30 of the election year. Ballots postmarked later than April 30 will not be valid.
c. The candidates receiving a simple majority of votes from the total votes cast will assume duties of their respective offices at the NECC conference, or by July 1, whichever comes first.
d. All terms of office are for two years and expire upon resignation of an officer or at the NECC conference, or July 1, whichever comes first.

Section 2: Board of Directors

The four voting members of the SIGLogo Board of Directors will be the President, Vice President, Communications Officer, and the Treasurer. The Editor(s) will serve as ex-officio members of the board.

Section 3: President

The President is the principal officer. The duties of the President include:

a. Calling and presiding at meetings of the Board of Directors and of SIGLogo.
b. Appointing all standing and ad hoc committee chairpersons.
c. Appointing members to fill offices that may become vacant through resignation, incapacity or ineligibility of an incumbent officer.
d. Acting as the SIGLogo liaison to the ICCE Board and its other Special Interest Groups.

Section 4: Vice-President

The duties of the Vice-President include:

a. Assuming the duties of the President in the event of the President's resignation or incapacity.
b. Coordinating SIGLogo standing committees.
c. Communicating with regional contacts and setting up conferences and workshops.
d. Assuming the office of President after serving as Vice President.

Bylaws for SIGLogo — continued

Section 5: Communications Officer

The duties of the Communications Officer include:

a. Keeping minutes of business meetings of SIGLogo and of the Board of Directors.
b. Maintaining records and correspondence of SIGLogo.
c. Notifying members of the Board of Directors of the time, place and agenda of the Board of Directors meetings.
d. Notifying the general membership of SIGLogo of the time, place and topic of the general meetings.
e. Sending official notifications to the ICCE Board of Directors of changes of the officers of SIGLogo.

Section 6: Treasurer

The duties of the Treasurer include:

a. Overseeing finances for SIGLogo.
b. Filing financial reports as are required by ICCE.

Section 7: Editor(s)

The Editor(s) will oversee all SIGLogo publications. The SIGLogo Editor(s) are appointed by the SIGLogo President and the ICCE editorial staff upon approval of the SIGLogo Board of Directors.

Article V - BOARD OF DIRECTORS

SIGLogo will be governed by the Board of Directors, which comprises the four officers. All decisions made by the Board of Directors must be approved by a majority of the board.

Article VI - BOARD OF DIRECTORS MEETINGS

The SIGLogo Board will conduct regularly scheduled business meetings each year. At least one meeting will be held in a place that is open to all members.

Article VII - AMENDMENTS

Section 1:

a. A resolution by the majority of the Board of Directors present at a meeting that has a quorum will be sufficient to cause an amendment to the bylaws to be voted upon by SIGLogo members. An amendment can be proposed to the Board of Directors by any SIGLogo member.
b. A petition by two percent of the SIGLogo members will be sufficient to cause an amendment to the bylaws to be voted upon by SIGLogo members. The right to petition will be independent of any decisions taken in accordance with Section 1, Paragraph a, of the article.

Section 2:

The proposed amendment will be presented to the ICCE Board of Directors and SIGLogo Board of Directors prior to distribution to the membership.

Section 3:

The proposed amendment will be voted on by the following ballot procedure:

a. The ballots will be mailed out or printed in the Logo Exchange and returned to such address as will be specified by the ICCE Board. The ballot will include (1) a copy of the proposed amendment, including specification of the date on which it will become effective if approved, and (2) a copy of the article(s) in the existing bylaws that is (are) being proposed for amendment.
b. Only ballots reaching the return address designed by ICCE within forty days after the last ballot was mailed out and postmarked within thirty days after the last ballot was mailed out will be valid.

Section 4:

An amendment will become effective if approved by a majority of the valid ballots cast by voting members of SIGLogo, on the effective date specified on the ballots or three months after the last ballot was mailed, whichever is later, unless specifically disapproved by action of the SIGLogo or the ICCE Board of Directors.

Article VIII - DISSOLUTION

SIGLogo can be dissolved by the consent of its members or by joint action taken by the SIGLogo and ICCE Board of Directors. Consent of the members of SIGLogo to dissolution will consist of unanimous agreement of the officers of SIGLogo together with a majority vote at a meeting which has been announced in advance to all members of SIGLogo. Upon dissolution of SIGLogo, its assets, if any, will be used to repay any outstanding debts incurred on behalf of SIGLogo. Any remaining debts will be paid by the ICCE. Any assets remaining after payment of SIGLogo debts remain the property of ICCE.

Article IX - REGIONAL CHAPTERS

Section 1: Purpose

Regional Chapters will be allowed under the operating structure of SIGLogo. Regional Chapters will participate in a national network of SIGLogo Regional Chapters.

Section 2: Formation

A group wishing to become affliated with SIGLogo as a Regional Chapter will petition for Regional Chapter status by submitting its membership guidelines, bylaws or operating objectives to the SIGLogo Board of Directors.

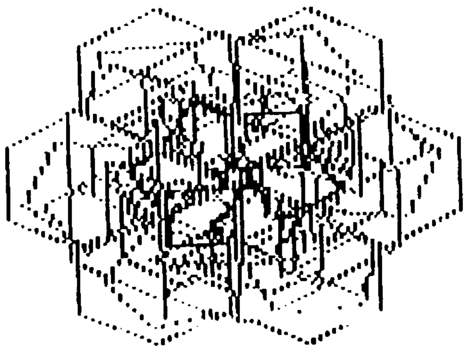
Section 3: Eligibility

Regional Logo groups will become eligible for recognition as Regional Chapters upon submission of a membership roster and verification that a majority of their members are current SIGLogo members.

Section 4: Regional Chapter Support

a. Each Regional Chapter will receive one copy of SIGLogo publications and notice of SIGLogo meetings, conferences and workshops. Additional correspondence involving issues to be addressed by SIGLogo, committee appointments and policy statements will also be sent to Regional Chapters.
b. Regional Chapters and SIGLogo will work together to establish groups for research, study and writing activities.
c. Regional Chapters may include the names of ICCE and SIGLogo in communications and advertising for non-profit activities.

Creating a Kaleidoscope



by Eadie Adamson

Kaleidoscopes are fascinating objects for children of all ages. A Kaleidoscope project is a nice follow-up for a random lesson and polygon explorations (see LX December, 1987 and January, 1988). I first used this idea a number of years ago when working with Commodore Logo. I had picked up David Thornburg's *Computer Art and Animation* (Addison-Wesley, 1984)

Incidentally, this book is full of *many* wonderful ideas about working with polygons as well as some ideas on planning animation — an excellent supplement if your interests lie in this area. You might want to refer to this text for a slightly different explanation of the kaleidoscope idea, for which credit and a deep bow from me goes to David Thornburg.

My students and I had been working with polygons, had used them in geometric patterns and had done a little study of Islamic design. David's chapter about making a kaleidoscope sounded like fun for us to try. I presented the idea to my students in a series of steps which built up to a final Kaleidoscope generator. (The version below is for LogoWriter, but can be adapted for other versions of Logo.)

Before you present the project to your students, make your own `KALEIDOSCOPE` program so that you can use it as an illustration of what you're aiming for. It's interesting to show it to students and ask them to analyze what is happening (without, of course, showing the procedures themselves).

The first requirement of the project is to choose at least six procedures for regular polygons, all written with a variable size. Since my students had already worked on the polygon project, this task was refined to:

Choose your six favorite shapes.
PASTE them on a new page.
Name the page `KALEIDOSCOPE` (or `IMAGE`.)

I like to make this activity a cooperative problem solving session. In this case, I really do have a completed program in mind and I strongly oppose simply giving students a page of procedures to type in and run (a friend of mine calls this Fascist Logo). Here's how to turn this project into a participatory one:

Step 1: A Problem to Pose

How can you start with the turtle at `HOME`, make the turtle jump out a given distance, draw a polygon, and jump back to the starting position?

You might simply assign this as a problem for students to solve on their own, and then have them share solutions. Or you might prefer make it a joint problem-solving session: ask the students to help you work out the necessary commands to produce a single shape which the turtle draws by moving out from `HOME`, drawing the shape, and moving back to `HOME`. Use a question mark for *some distance* out from the center and for a shape of *some size*. If they haven't used procedure inputs much yet, this leads nicely into substituting values for the names in WINDMILL (below). Together, write a procedure to do this with a single shape.

Step 2: Drawing Shapes Around a Center

The next task will sound more difficult, although it is really simple: How can you change this procedure so that it draws six shapes spaced equally around the center?

Again, this can be an individual problem to solve and then students can share strategies and solutions. As a group project, approach it this way: Ask how to get six of these shapes (at this point you might show one generation of the completed IMAGE to illustrate what you're asking them to work out) and have them suggest how to change the procedure to do this. See if they can relate it to their HEXAGON to find the necessary turn.

Step 3: Writing a Windmill Procedure

Next, having worked out the mechanics of the beginning process, suggest choosing a favorite shape and write a WINDMILL procedure which looks like this, using any shape desired where TRI is included below:

```
TO WINDMILL :DIST :SIZE
 REPEAT 6 [PU FORWARD :DIST PD TRI :SIZE
    PU BACK :DIST]
END
```

Now the students are ready to experiment with sizes, distances, and colors. Some may add color as another input for WINDMILL; others may simply elect the strategy of changing pen color before drawing. Encourage them to create procedures (with new names) of the designs they like. (This gets at the idea that a procedure can be used within another procedure, something that is sometimes hard to make clear.) If your class has been studying Islamic design they will find some nice relationships here. Allow plenty of time, one full class period at the very least, for experimentation with this part of the process. Be sure to save and print often. You'll find you will have quite a collection of beautiful images just from this simple beginning.

Step 4: The PICK Tool

Give this tool procedure to your students before continuing the

Creating a Kaleidoscope — CONTINUED

project:

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

Explain just enough to satisfy your students about how PICK works. A good explanation is picking numbers or notes out of a hat. You might want to use the PICK procedure as an opportunity to explain how OUTPUT works.

```
PICK [JIM MARY JACK JANE]
```

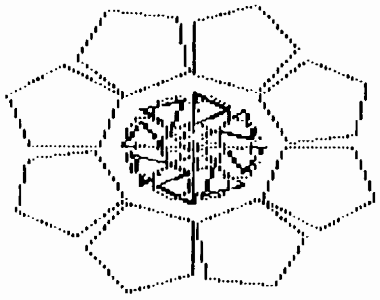
will generate an error message: I DON'T KNOW WHAT TO DO WITH JIM IN PICK. Here, if you want to digress, you can talk about the various ways you can display text in LogoWriter: PRINT, TYPE, INSERT, and SHOW. Encourage some experimenting with PICK and lists of names or numbers. Be sure to point out that PICK is a procedure used often in Logo when one needs the computer to make a random choice of items from a list. It's a useful procedure which might well have been included as a Logo primitive.

Step 5: Using the PICK tool

Now make a slight adjustment to WINDMILL to allow WINDMILL to use a shape chosen randomly by PICK (changes are in lower case):

```
TO WINDMILL :DIST :NAME
 REPEAT 6 [FORWARD :DIST PD RUN :NAME PU
                BACK :DIST RIGHT 60]
END
```

Note: Some students may want to preserve the "old" WINDMILL. They may do this by copying the original version first, then renaming the old unchanged version. The new WINDMILL will not work until IMAGE (below) has been written. WINDMILL is incorporated into IMAGE and fed information chosen randomly by IMAGE before it is generated.



Step 6: Creating the IMAGE procedure

The final step in creating IMAGE needs some explanation and does, unfortunately, require giving the students something to type in. I prefer to do this one line at a time, giving the explanation as we go along (see below). I've added line numbers here to connect with the explanations. Don't include these in your procedures!

```
TO IMAGE
 MAKE "LIST [TRI SQ PENT HEX OCT DEC]    (1)
 MAKE "NAME (SENTENCE
       (PICK :LIST) (5 + RANDOM 30))     (2)
 MAKE "DIST RANDOM 45                    (3)
 SETC 1 + RANDOM 5                       (4)
 PU
 WINDMILL :DIST :NAME                    (5)
 MAKE "TEMP READCHAR                     (6)
 IMAGE                                   (7)
END
```

(1) This line makes a LIST of the names of the shapes procedures that you plan to use (you'd change that line if you added new shapes).

(2) Next, PICK is used to choose a shape name randomly from the list and assign the shape a random size that is at least 5 (5 + RANDOM 30). It makes a sentence of the two items: the shape and its size. It might choose TRI 30, for example, then later choose OCT 20. (Later you could experiment with changing the minimum size.)

(3) Next, DIST is given a value generated by RANDOM 45 (0 to 44, remember?). You might experiment with changing these values to see how a different distance affects the design.

(4) Next, a random pen color is picked, but black is omitted (how?)

(5) Finally, WINDMILL is run using this information.

(6) Notice something in the line MAKE "TEMP READCHAR. READCHAR is the short way of saying Read a Character. READCHAR tells the computer to wait for a key to be pressed before continuing. Once a key is pressed, it goes to the next command, IMAGE. This starts a new version of IMAGE, randomly choosing a shape, a size, a distance, and a pen color. Point out that this is how you can make many things interactive by asking a player to press a key when ready, for instance.

There are a couple of programming issues worth mentioning here. In both cases, you make the code cleaner, but then they are probably harder for beginners to understand and may just add confusion.

- You could substitute IGNORE READCHAR for MAKE "TEMP READCHAR, but then you'd need to write another short procedure:

```
TO IGNORE :KEY
END
```

- You can "collapse" the code of IMAGE, causing the MAKE statements to be unnecessary.

```
TO IMAGE
 SETC 1 + RANDOM 5
 PU
```

Creating a Kaleidoscope — CONTINUED

```
    WINDMILL
      RANDOM 45
      (SENTENCE PICK [TRI SQ PENT HEX OCT DEC]
                    5 + RANDOM 30)
    IGNORE READCHAR
    IMAGE
   END
```

(7) Recursion is one of the most powerful processes one can use in Logo. Can you see that the recursive call to IMAGE creates an indefinite loop? If it were a FOR loop such as found in the BASIC language, it would repeat the process a definite number of times. This example of tail-end recursion shows one way that the power of recursion can be used to build programming structures to meet your needs.

Here's IMAGE again, with the additions to make Logo give you the information (in lower case):

```
TO IMAGE
 MAKE "LIST [your shapes]
 MAKE "NAME SENTENCE :LIST 5 + RANDOM 30
 TYPE SENTENCE "SHAPE :NAME
 TYPE CHAR 13
 MAKE "DIST RANDOM 45
 TYPE SENTENCE "DISTANCE :DIST
 TYPE CHAR 13
 SETC 1 + RANDOM 5
 TYPE SENTENCE "SETC COLOR
 TYPE CHAR 13
 PU
 WINDMILL :DIST :NAME
 MAKE "TEMP RC
 IMAGE
END
```

TYPE makes the message appear in LogoWriter's command center. Since it does not give a carriage return, generate one by TYPE CHAR 13 (CHAR 13 is ASCII) for the RETURN key. You could also write (TYPE SENTENCE "SETC COLOR CHAR 13). Note the parentheses at each end of this statement.

The output from your IMAGE procedure should read something like this:

```
SHAPE TRI 30
DISTANCE 20
SETC 5
```

Now if you want to save a design by making a procedure, you need just a small change to the WINDMILL procedure to allow you to use it with whatever shape you wish, BUT be sure to change the procedure name also, so you can have this procedure to use at the same time as WINDMILL:

```
TO MILL :DIST :NAME :COLOR
 SETC :COLOR
 REPEAT 6 [PU FORWARD :DIST PD RUN :NAME PU
                       BACK :DIST RIGHT 60]
END
```

Here's how to use MILL:

```
MILL 40 [TRI 25] 4
```

In this example 40 is the value for DIST and TRI 25 is the NAME; (the shape with its input must be enclosed in a list of its own). The COLOR selected here is 4. You can use MILL in a procedure now to reproduce a design produced by IMAGE. Give your design a special name, then incorporate the information for each generation of IMAGE into a single line for MILL, like this:

```
TO PIC
 MILL 40 [TRI 25] 4
 MILL 30 [OCT 15] 3
 MILL 45 [PENT 20] 2
END
```

Other Ideas for Image

If you're using LogoWriter, try creating the design with three or four turtles instead of one. A SETUP procedure will be necessary to get all the turtles to start in the same place. SETUP will also need to set each turtle to a different heading to begin. Three turtles work well if you generate six shapes, four turtles if you use eight. Each will draw a shape, then turn and draw the shape again. (See solution below.)

One idea that is only marginally satisfactory with LogoWriter is to FILL the shapes as they are drawn. This works at first, until shapes overlap. Then sometimes the results are not pleasing. Still, it's interesting to try!

Another idea might be to go to the SHAPES page and design a series of geometric shapes. Then change IMAGE so that it selects the shape, now no longer a procedure but a shape number, and STAMPS rather than draws the shape. You might even adjust the minimum distance from the center, change the turn, and allow the turtles to draw colored lines from the center each time (try something like a turn of 15) before STAMPing the SHAPE.

You might put a collection of printouts together to form a quilt or cut them out, color them, and paste them together to make a stained glass window or add sound as the turtle(s) draw, or -- Any other brilliant ideas out there?

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

Some Polygons to use with IMAGE:

```
TO TRI :SIZE
 REPEAT 3 [FORWARD :SIZE RIGHT 120]
```

Creating a Kaleidoscope — CONTINUED

```
END

TO SQ :SIZE
 REPEAT 4 [FORWARD :SIZE RIGHT 90]
END

TO PENT :SIZE
 REPEAT 5 [FORWARD :SIZE RIGHT 72]
END

TO HEX :SIZE
 REPEAT 6 [FORWARD :SIZE RIGHT 60]
END

TO OCT :SIZE
 REPEAT 6 [FORWARD :SIZE RIGHT 60]
END
```

The PICK procedure:

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

The IMAGE procedure:

```
TO IMAGE
 MAKE "LIST [TRI SQ HEX PENT OCT]
 MAKE "NAME SE PICK :LIST 5 + RANDOM 30
 MAKE "DIST RANDOM 45
 SETC 1 + RANDOM 5
 PU
 WINDMILL :DIST :NAME
 MAKE "NAME READCHAR
 IMAGE
END

TO WINDMILL :DIST :NAME
 REPEAT 8 [FORWARD :DIST PD RUN :NAME PU
                    BACK :DIST RIGHT 45]
END
```

For four turtles, eight shapes each time:

```
TO SETUP
 TELL [1 2 3]
 PU SETPOS [0 0]
 TELL 1
 SETH 90
 TELL 2
 SETH 180
 TELL 3
 SETH 270
END

TO I
 MAKE "LIST [TRI SQ HEX PENT OCT]
 MAKE "NAME SENTENCE PICK :LIST 5 + RANDOM 30
 MAKE "DIST RANDOM 45
 SETC 1 + RANDOM 5
 PU
 WINDMILL2 :DIST :NAME
 MAKE "NAME READCHAR
 I
END

TO WINDMILL2 :DIST :NAME
 TELL [0 1 2 3]
 HT
 PU
 REPEAT 2 [FORWARD :DIST PD RUN :NAME PU
          BACK :DIST RIGHT 45]
END
```

Expanding Kaleidoscope: Reporting on the Process

Suppose you'd like to be able to recreate some of the designs your IMAGE program generated. To do this, you can add a few lines to IMAGE which ask the computer to report its random choices as it runs.

What would you need to know? The shape—in this case, the NAME—chosen by IMAGE, and its size. You'd also need the distance and pencolor. It's not hard to make the computer give you this information at the bottom of the screen while the procedure runs. Since IMAGE waits for you to press a key before it continues, you have all the time you want to write down what's going on before generating another shape. With LogoWriter you can also scroll back up to read what's in the command center if you prefer to generate a number of shapes first.

Correction to Judi Harris' Picture Tools article:

For Terrapin Logo:

```
TO POS
OUTPUT SENTENCE XCOR YCOR
END

TO LOAD.PIC : NAME
READ :NAME
READPICT :NAME
PU
SETX FIRST THING WORD : NAME "POS
SETY LAST THING WORD : NAME "POS
PD
SETHEADING THING WORD : NAME "HEADING
ST
CLEARTEXT
SPLITSCREEN
END
```

With thanks to Mel Levin.

Natural Language Microworlds: An answer to the challenge

by Michael Tempel

In the October issue of *LX*, Robs Muir invites readers to explore the development of natural language microworlds. In that column, he raises a number of interesting ideas which this article will address and extend.

Initially, he draws what I think is a false dichotomy by separating list processing from turtle graphics. How about this pair of procedures, written in LogoWriter:

```
TO GO.AWAY
 TELL [0 1 2 3]
 ST
 PU
 CRAWL [0 1 2 3]
END

TO CRAWL :WHO
 IF EMPTY? :WHO [STOP]
 TELL FIRST :WHO
 RIGHT 45
 REPEAT 100 [FORWARD 1]
 HT
 CRAWL BUTFIRST :WHO
END
```

The structure of crawl might look familiar to you if you've seen:

```
TO TRIANGLE :LIST
 IF EMPTY? :LIST [STOP]
 PRINT :LIST
 TRIANGLE BUTFIRST :LIST
END

TRIANGLE [Welcome to Logo]
Welcome to Logo
to Logo
Logo
```

Another approach which integrates graphics and list processing has been created by Peter Rawitch. Peter created a Logo Zoo which puts together pieces of animals to form new creatures by processing lists of turtles to create the animals and lists of words to create their names. For example, each syllable of croc-o-dile is represented by a shape. By creating a number of these "subdivided" animals, "hybrid" animals can be created. For example, a croc-o-ake would be two parts crocodile and one part snake. (Can you expand on this idea as Peter has?)

It's also true that working with natural language in Logo does not necessarily require list processing. This is most true in LogoWriter where text may be manipulated through word processing. The perceived dichotomy between turtle graphics and list processing has never been as great as implied in most of the Logo literature. However, LogoWriter provides more things needing lists (such as multiple turtles) and more ways to process language. In fact, the natural language microworlds that Robs refers to require no more list processing knowledge than drawing a square does.

But, as Arlo Guthrie once said, "that's not what I came here to tell you about." I came to talk about the classroom applications that, according to Robs, the natural language microworlds are crying for. They do, in fact exist, and were an integral part of Bob Lawler's "Beach World" back in 1980.

The term *application* may be misleading when referring to this kind of natural language microworld. What is being built is a learning environment and a set of activities that go with it. In the case of Beach World the goal was to create a "play pen" in which beginning readers could attach written words to familiar objects and actions. Bob wrote the program for his three year old daughter, Peggy, using the familiar setting of the Connecticut shore near their home. Objects such as boy, girl, boat, car, sun and dog were created with the shape editor that was part of the versions of Logo he used (TI Logo and Sprite Logo). Logo procedures made the objects appear and other procedures moved them around. For example

```
TO GIRL
 NEXT
 SETSHAPE :GIRL
END

TO DOG
 NEXT
 SETSHAPE :DOG
END

TO NEXT
 TELL WHO + 1
 ST
 PU
END

TO UP
 SETY YCOR + 5
END

TO DOWN
 SETY YCOR - 5
END

TO FAST
 SETSPEED 50
END

TO SLOWLY
 SETSPEED 10
END
```

The versions of Logo used for the Beach World had 30 or 32 turtles. The procedure NEXT was used to get the attention of the next turtle and made it visible on the screen. Each time a noun like GIRL or CAR was used, a new turtle appeared with that shape. It could then be

Natural Language Microworlds: An answer to the challenge — CONTINUED

moved around and placed anywhere on the background which consisted of a sky, beach, and water with some buildings and trees.

The vocabulary of this microworld was written on index cards. Children using the program could refer to the cards to remind themselves of the available words. Even if they did not know a word, typing it would most likely produce a result that would reveal its meaning.

An important aspect of the Beach World was that its content was familiar and personally meaningful to the user. It was also easily modifiable so that Bob or his older children could make changes for his youngest daughter, and they, in turn, used parts of the Beach World for their own Logo projects. In a school setting, older students, as well as adults, could create or modify these language microworlds for use by younger students.

The Beach World worked best when single words were typed and entered. Each word was a Logo command that produced a result on the screen. Having Logo understand an English phrase or sentence is a bit more tricky. Some people say that Logo is like English. This is mostly false. (That's why Robs' challenge is challenging!) What *is* true is that a Logo instruction is made up of words, usually separated by spaces. This is like English. Also, Logo words may be English words and these words may be the names of procedures. Thus, you can have Logo appear to understand English with things like:

```
THE BLUE TURTLE GOES SLOWLY using the following procedures:

TO THE
END

TO BLUE
 SETC 3
END

TO TURTLE
 SETSH 0
END

TO GOES
END

TO SLOWLY
 REPEAT 100 [FORWARD 1]
END
```

This sleight of hand is great for demonstrations, but inquisitive second graders will quickly discover that

```
TURTLE THE BLUE SLOWLY GOES
```

works just as well! Logo syntax is not like English. For example,

```
IF IT'S RAINING AND I'M GOING OUT, I'LL TAKE
AN UMBRELLA
```

would look like this with Logo syntax:

```
IF AND IT'S RAINING, I'M GOING OUT, I'LL TAKE
AN UMBRELLA.
```

Natural language also allows for much ambiguity that programming languages do not. "Turtle the blue slowly goes" may be peculiar English, but we still get the drift. It is as interesting to see what gibberish a program accepts as it is to see what proper sentences it "understands." How can we create a microworld that accepts reasonable English phrases and rejects nonsense?

When all the words in the language microworld are Logo commands that take no inputs, they may be strung together in any order. The result may make little sense in English. English phrases are made up of words linked together in specific ways. By using reporters along with commands that take inputs we can impose some structure which limits the phrases that are acceptable.

Harry Nelson uses this approach in the "Logo Grammar" section of the documentation accompanying the *On Logo* videotapes. The sentence,

```
DRAW A RED SQUARE
```

requires the procedures

```
TO DRAW :INPUT
 RUN :INPUT
END

TO A :INPUT
 OUTPUT :INPUT
END

TO SQUARE
 OUTPUT [REPEAT 4 [FORWARD 50 RIGHT 90]]
END

TO RED :INPUT
 OUTPUT SENTENCE [SETC 5] :INPUT
END
```

DRAW is the only command in this line. The other words are the names of reporters that feed information on down the line. The procedure A just passes its input on. RED adds the command SETC 5 to its input and sends the longer input on. Finally, DRAW runs the accumulated list.

DRAW must be the first word of the line and RED must precede SQUARE. In contrast, one could write a commands only version like this:

```
TO DRAW
END
```

Natural Language Microworlds: An answer to the challenge — CONTINUED

```
TO A
END

TO RED
 SETC 5
END

TO SQUARE
 REPEAT 4 [FORWARD 50 RIGHT 90]
END
```

Using these procedures the line

```
DRAW A RED SQUARE,
```

would produce the same result as with Harry's procedures. But lines like

```
RED SQUARE
```

and

```
A RED SQUARE
```

would also work.

A language microworld based on commands without inputs works well for single words, but not so well for phrases. On the other hand, linking words by using reporters and commands that require inputs may impose a structure that is more restrictive than we might want it to be. The challenge is to create a reasonable mesh between Logo and English.

A different approach to creating natural language microworlds is to write an interpreter. This puts a layer between the user and Logo where a Logo program reads lines that are typed and acts on them. Here's an example:

```
TO LISTEN
 TYPE "?
 INTERPRET READLISTCC
 LISTEN
END

TO INTERPRET :THISLINE
 IF EMPTY? :THISLINE [STOP]
 IF MEMBER? FIRST :THISLINE
    [A THE AN DRAW MAKE DO]
           [INTERPRET BF :THISLINE STOP]
 IF MEMBER? FIRST :THISLINE
    [RED BLUE WHITE GREEN PURPLE]
           [SETC RUN (LIST FIRST :THISLINE)
            INTERPRET BF :THISLINE
            STOP]
 IF MEMBER? FIRST :THISLINE
    [SQUARE TRIANGLE CIRCLE]
           [RUN (LIST FIRST :THISLINE)
           INTERPRET BF :THISLINE STOP]
 (TYPE [I DON'T KNOW THE WORD]
           FIRST :THISLINE CHAR 13)
END

TO RED
 OUTPUT 2
END

TO WHITE
 OUTPUT 1
END
```

etc.

```
TO SQUARE
 REPEAT 4 [FORWARD 50 RIGHT 90]
END

TO TRIANGLE
 REPEAT 3 [FORWARD 50 RIGHT 120]
END

TO CIRCLE
 REPEAT 36 [FORWARD 10 RIGHT 10]
END
```

Writing your own interpreter overrides Logo. You can have your program accept words like MAKE which already have a meaning to Logo and would not be useable in the top level approach. LISTEN will print an "I don't know the word ..." message when an unknown word is encountered. The rest of the line will not be evaluated. In the top level approach, the regular Logo error messages will be printed and could be misleading to beginning readers.

On the other hand, the interpreter excludes normal Logo vocabulary. The top level approach does not prohibit using a FORWARD 100 whenever one wants. The interpreter doesn't know Logo unless you build in such knowledge.

The interpreter approach puts the programmer in touch with an area of computer science that is fundamental to Logo, writing an interpreter, a program that reads lines that you type and interprets their meaning. Logo itself is an interpreter. In fact, can you write Logo in Logo? Can you write BASIC in Logo? (Would you want to?)

Natural language microworlds provide a valuable learning environment both for the beginning reader using them and the programmer creating them. They have educational value at many levels: word and phrase recognition for the beginner, more sophisticated programming for the older student, and exploration of the nature of natural language for the student of any age. Why not present some aspect of this idea to your students of whatever age, and see what results!

Michael Tempel, Logo Computer Systems, Inc., 330 West 58th Street, Suite 5M, New York, NY 10019

Teaching Tools

Sound Concepts by Glen Bull and Gina Bull

1. A group of children sit around a collection of bottles, creating different musical tones with an impromptu jug band.

2. Several children on the playground dash ten yards, touch an oak tree, and then run back to the starting line. A second group repeats the process with a starting line that is twenty yards away from the tree.

3. Children gather around a computer monitor, using Logo tools to interpret their first two experiences.

At first glance it might appear as though events (1) and (2) have nothing to do with one another, or with Logo. Yet there is a common thread that links all three. It is desirable to bring a variety of both computer-related and non-computer-related experiences to bear on a concept. In this column, a specific illustration of how this might be done will be examined.

Teaching begins with a concept or objective to be learned. In Virginia these goals are listed as Standard of Learning objectives. Here is a Standard of Learning objective for the fifth grade:

"The student will demonstrate the ability of sound to travel through solids, liquids, and gases and describe the wave nature of sound."

By the eighth grade the Standard of Learning objective has metamorphosed into the following:

"The student will investigate the basic characteristics and technological applications of mechanical waves."

Characteristics listed include sound wave motion, water wave motion, reflection, refraction, and interference, while technologic applications suggested for this objective include music, communication, and hearing.

Both the jug band and the foot race are related to the topic of sound and motion, and the objectives listed above. The reason for inclusion of the jug band may be evident, while the foot race may require some explanation.

Sound Waves

As a starting point, consider why some bottles produce a low tone while others produce a high note. Generally tall bottles will produce a deep tone, while short bottles will produce a high pitch. A tube provides a good laboratory for experimenting with some of the variables underlying this physical phenomenon. Sound is a back and forth motion. It takes a longer time for a sound to ripple down to the end of a long tube and back again. Therefore fewer back and forth cycles occur in a given period of time, in comparison with a short tube.

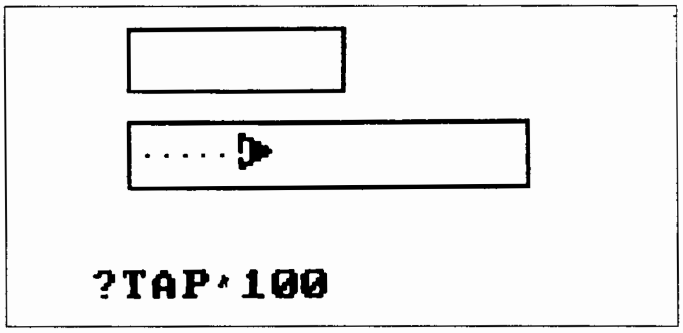
Several Logo tools will be used as building blocks to construct tools for exploration of sound and motion. The Logo TUBE procedure draws a rectangle which will serve as a tube, and places the turtle in the tube.

```
TO TUBE :LENGTH
 SETH 90
 PD
 RECTANGLE :LENGTH + 8 20
 OVER 10
 PU
END

TO RECTANGLE :LENGTH :WIDTH
 REPEAT 2 [FORWARD :LENGTH RIGHT 90
          FORWARD :WIDTH RIGHT 90]
END

TO OVER :DISTANCE
 PU
 RIGHT 90
 FORWARD :DISTANCE
 PD
 LEFT 90
END
```

The tube drawn is the length specified, plus the length of the turtle (about 8 steps). Thus, the command TUBE 50 draws a tube 58 turtle steps long, while the command TUBE 100 draws a tube 108 turtle steps long.



```
?TAP 100
```

Now that the turtle is in the tube, the procedure TAP can be used to tap the end of the tube, and send the turtle shuttling back and forth. The subprocedures STEP.FD and STEP.BK are used to ensure the turtle travels at a constant speed. In other words, the time required to travel 100 steps should be twice as long as the time required to travel 50 steps.

```
TO TAP :LENGTH
 STEP.FD :LENGTH
 STEP.BK :LENGTH
END

TO STEP.FD :TIMES
 REPEAT :TIMES [FORWARD 1]
END

TO STEP.BK :TIMES
 REPEAT :TIMES [BACK 1]
END
```

When the procedure TAP is run, it produces a tap on the end of the tube, causing the turtle to slide down to one end of the tube, and then bounce back to the other end. The TAP procedure must include the length of the tube as an input.

```
TUBE 100
```

```
TAP 100
```

Repeat taps allow students to determine how many round trips take place in 60 seconds. This makes it possible to explore the number of round trips that can be completed for tubes of different lengths.

```
TUBE 50
REPEAT 20 [TAP 50]

TUBE 100
REPEAT 20 [TAP 100]
```

For example, we found that 12 round trips occurred in 60 seconds when the tube was 50 turtle steps long. However, when the tube was 100 turtle steps long, there was only time for six round trips. (Your results may differ from ours, depending upon the type of computer and version of Logo you are using.)

Length of Tube	Round Trips in 60 Seconds
50	6
100	12

The behavior of the turtle in the tube makes it possible to understand why the sound made by blowing across the lip of a short bottle has a high pitch. The sound wave can travel to the end of the bottle and back more times in a short bottle than in a long bottle in a given period of time. We counted the number of times per minute that the turtle moved back and forth in the Logo tube, while the back and forth trips of sound waves in a bottle are measured in trips per second. The speed of a sound wave is faster than a Logo turtle. Otherwise the events are very similar.

At the beginning of the column we described a group of children dashing ten yards to an oak tree and then back again. The group that dashes ten yards and back will be able to make more round trips in a minute than a second group that dashes 20 yards. Both the turtle and the children are simulating the events in a bottle. The turtle will move at a more constant speed (some children are apt to be faster than others), but children are more likely to remember something that they do with the movement of their own bodies. Combining both experiences will facilitate interpretation of the phenomena underlying sounds produced by bottles of different sizes.

How often an event occurs is referred to as the *frequency*. For most weeks, if someone asked how frequently we go to school, the answer would be "five trips per week." (Exceptions, of course, could occur if there is a snow day, or if we forgot something and have to make an extra trip.) The turtle, as we have seen, has a frequency of five or ten round trips per minute, depending on the length of the tube. The frequency of school trips is about five per week, while the frequency of turtle trips is about five per minute. Sound waves have a frequency of about 20 to 20,000 back and forth trips per second.

Each complete round trip is called a *cycle*, so a sound wave could be said to have a frequency of *50 cycles per second*. Since the number of round trips which can take place in a second is greater in a shorter bottle, the frequency, or cycles per second, is higher. In recent years the measure *cycles per second* has been replaced by the term *Hertz* (abbreviated Hz) in honor of the scientist Heinrich Hertz. Giving the measure a name which has nothing to do with its derivation has made it harder to explain the concept to school children.

We also developed more sophisticated procedure which counts the number of trips the turtle has completed, and prints them when the space bar is pressed at the end of 60 seconds. You could develop a similar procedure as a programming challenge for yourself; or if you send us a stamped, self-addressed envelope, we would be happy to send you a printout of the procedure. However, it is important not to let the Logo programs get in the way of the concepts to be learned. The simpler procedure in which the students count the back and forth movements of the turtle themselves will work just as well. In either case, a watch with a second hand can be used to time when the 60 second period is complete.

A Closer Look At a Sound Wave

In this example, the passage of the turtle represents the movement of the sound wave. If a sound wave is examined in more detail, it is found to be the ripple produced by the jostling of molecules as the wave passes through air. In that respect it is very similar to the ripple produced by the movement of the coils of a slinky, or the movement of dominoes as they topple against one another.

We can simulate the movement of air molecules with ping pong balls in a tube. When the tube is tapped on the end, the first ball bumps into the next, and so on until the ripple reaches the end of the tube. At that point, the end ball rebounds on the opposite end of the tube, causing the ripple to reverse itself and flow back in the opposite direction.

The movement of the ping pong balls in the tube can be simulated with a Logo procedure. This type of procedure is easier to create in a version of Logo which permits multiple turtles. For example, LogoWriter provides up to four turtles. In the procedure below, each turtle will represent a different ping pong ball. The DRAW.TUBE procedure below does the following in LogoWriter:

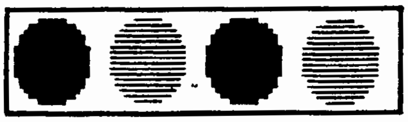
1. It draws a tube.
2. Each turtle is changed into a ball. (SETSH 12)
3. The turtles / balls are placed in the tube.

Here is one way of writing the DRAW.TUBE procedure.

```
TO DRAW.TUBE
 TELL 0
 PU
 SETH 90
 SETPOS [-10 13]
 PD
 RECTANGLE 82 25
 TELL ALL
 PU
 SETH 90
 EACH [SETPOS LIST (WHO * 20) 0]
 SETSH 12
END
```

Teaching Tools — CONTINUED

When the DRAW.TUBE procedure is run in LogoWriter, the results should look like this:



Now that the tube has been drawn, and populated with ping pong balls (turtles in disguise), it is necessary to write a procedure to cause them to bump into one another. There are four balls, labeled in computer fashion BALL 0, BALL 1, BALL 2, and BALL 3.

```
TO BUMP :BALL
 IF :BALL > 3 [STOP]
 TELL :BALL
 FORWARD 5
 WAIT 1
 BUMP :BALL + 1
 TELL :BALL
 BACK 5
 WAIT 1
END
```

The BUMP procedure makes use of embedded recursion. To run it, you must tell BALL 0 to bump into the ball next to it.

```
BUMP 0
```

The collision process continues until the last ball (BALL 3) is hit. At that point it collides with the end of the tube and rebounds, reversing the direction of the flow. The length of the wait in WAIT 1 can be increased to make it easier to observe the process, or the WAIT command can be taken out of the procedure altogether to speed up the movement. Alternatively, the procedure could be changed so that a key press is used to make each ball bump into its neighbor.

Just as the children on the playground replicated the movement of the turtle in tubes of differing lengths, the movement produced by ping pong balls in a tube can also be emulated. Line up students between two rows of desks. The first student can bump (gently) into the next, and so on, until the end student bumps into the wall. As the end student rebounds, the direction of the flow will reverse itself. This sort of activity can be timed for different numbers of students, representing tubes of differing lengths. In a short tube (with fewer numbers of students) the ripple should run back and forth more times in a minute. On a grander scale, it might be fun to time waves of the kind that have recently become popular in baseball parks. The people in each section stand up and then sit down again as the wave ripples around the stadium. This raises the question of whether a wave would complete its cycle faster in a small stadium than in a large one.

In these examples, both on-computer and off-computer activities have been combined to provide similar experiences. It is desirable for the computer to become one of several teaching tools rather than an end in itself. Papert describes the process of relating the movement of a child's body to the movement of the turtle as "body syntonic" learning. In this process, something with which the child is familiar (bodily movement) is related to a new expression in a different medium (*Mindstorms*, p. 63).

More Sound Concepts

The computer can also be used in other ways to reinforce experiments with sound. For example, some versions of Logo can be used to produce a tone of any frequency with the computer. Children can try experimenting with the difference between:

```
TOOT 500 10
TOOT 2000 10
```

In these dialects of Logo, a command such as TOOT or TONE is followed by the frequency and the duration of the tone. Typing the precise frequency in cycles per second reinforces the concept that frequency is an expression of how frequently the vibration is occurring. (In this instance, it represents how many times a second the speaker cone inside the computer is moving back and forth.)

Conclusion

A variety of computer-related and non-computer-related activities can be woven together to support existing curriculum goals. In science class, the computer can be used in several ways to provide teaching tools:

1. Modeling — It can be used to model external phenomena, as in the case of the Logo tools which were outlined in this column.

2. Stimulus Generation and Control — The computer can be used to control stimuli and external processes in science class, just as it does in real science experiments. A decade ago a digital tone generator capable of producing different frequencies in 1 Hz increments would have cost more than the computer does now.

3. Data Acquisition and Analysis — The computer can also be used to acquire and analyze data, as in the case of the microphone attached to the computer. Two columns related to this type of application were discussed in previous Teaching Tools columns on "Science and Sensors."

As a result of breakthroughs in computing power, even elementary grades can now have access to teaching tools which were once reserved for college or high school.

Glen Bull and Gina Bull, Curry School of Education, Ruffner Hall, University of Virginia, Charlottesville, VA 22903; CompuServe: 72477,1637

Logo Linx

Rhyme Paradigm
by Judi Harris

What two rhyming words are described by this phrase?
"....an ardent dam builder with a long flat tail"

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

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

Try this *HINK PINK*:

"....a petty quarrel between two winged insects"
(gnat spat)

Or this *HINK-IT-Y PINK-Y*:

"....someone who has farther to travel to get to the polls"
(remoter voter)

Enabling Labels

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

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

is correctly solved with two rhyming words, respectively two and three syllables long. (*HINK-Y*: nec-tar; *PINK-IT-Y*: col-lec-tor; nectar collector)

Now try this *HINKY PINKY*:

"....a sailing vessel on a voyage to the earth's satellite"
(Try to solve this one!)

Pedagogic Logic

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

But, wait! Lest you unjustly accuse me of

"....a chronic inability to use the right words when speaking or writing"
(diction affliction),

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

Induction Production

Examine these puzzles and their solutions for word type patterns:

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

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

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

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

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

Code Mode

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

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

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

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

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

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

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

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

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

Here's video-training for teachers for only $599!

Eight half-hour videotapes created by MIT's Seymour Papert show you how to use LogoWriter or traditional Logo to teach problem solving and thinking skills, analyze learning styles, and understand how knowledge is best appropriated. Now you can invite the "Father of Logo" into your own school, as your personal instructor, with the first videotape Logo learning system for teachers.

And the series is also designed to help you overcome technical "hurdles" that often stand in the way of project development and curriculum integration... opportunities often not found in schools because of teacher limitations.

Now available! Graduate credit courses from ICCE and University of Oregon

The International Council for Computers in Education and Media Microworlds Inc. present a graduate-level independent study course on Logo, available to educators employed by institutions purchasing the ON LOGO video tapes. Four quarter-hours of graduate credit are awarded upon satisfactory completion.

The course, *Introduction to Logo Programming with LogoWriter*, is for educators who wish to increase their knowledge about Logo and who wish to acquire graduate credit hours. Educators view the ON LOGO videotapes, do Logo programming, read and report on course materials, design lessons for their students, and correspond with their instructor by mail.

Additional course materials include a LogoWriter Single User disk, a guided Logo programming manual, a lesson guide, and an article packet. Texts include *Mindstorms: Children, Computers, and Powerful Ideas* by Seymour Papert and *Cultivating Minds* by Sylvia Weir.

Registration forms may be obtained from ICCE, University of Oregon, 1787 Agate St., Eugene, OR 97403-9905.

Duplicate tapes for all schools

For **only $1,295** you may purchase the video series with **duplicating rights.** Duplicate copies allow you to build check-out libraries at multiple sites, thus providing ongoing staff development opportunities for teachers at their convenience. Many of the videos are appropriate for administrators, board members, parents, and other groups interested in learning more about computers in education.

The duplicating rights include a **200-page loose-leaf notebook and five copyable data diskettes** which allow you to study, modify, or extend all demonstrations shown on the screen.

(Duplication offer made only to school districts and independent schools and restricted to use within one institution. Duplicating license includes in-house, closed-circuit television rights, but does not include broadcasting rights, use of material for video credit-courses broadcast outside of the institution, nor permission to lease or sell dubs. Rights to the material for video-credit courses are negotiated on an individual basis. Consortia and organizations servicing several districts should write for licensing agreements based on size of population served.)

Are you simply "teaching your students <u>how to use</u> computers?" Here's how you can do much more!

☐ Send me **Seymour Papert's eight videotapes for $599**.

Please circle one: **VHS** or **Beta**

☐ I wish to obtain a duplicating license with my institution's purchase of eight video tapes for $1,295. I understand that the fee also includes a 200-page notebook and five copyable data diskettes.

Please *circle one*: **VHS** or **BETA** (Three-quarter format available to licensees for additional $100.)

Diskettes available-please circle one: Apple LogoWriter, IBM LogoWriter, PC Jr. LogowWriter, Apple Logo, Apple Logo II, Terrapin, IBM Logo.

☐ I would like to receive additional information on graduate credit courses available through ICCE, University of Oregon.

Please add 5% for shipping/handling.
Allow 2-3 weeks for delivery.
All prices are subject to change without notice.
Materials must be in saleable condition and returned within one week after delivery for full refund.

Media MicroWorlds, Inc. -12 Clayton Terrace - St. Louis, MO 63131-(314)567-0150

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

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

If you type BEGET.NN, the computer may return

```
LEEK WEEK.
```

If you type BEGET.AN, it may respond with

```
GREEK PHYSIQUE.
```

Definition Renditions

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

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

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

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

```
DESCRIBE [UNIQUE ANTIQUE],
```

the computer will print

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

To which the user could respond

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

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

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

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

Perusing Uses

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

"...a singing group with a full, rich timbre"

and solutions can encourage dictionary usage.

(sonorous chorus)

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

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

Addiction Restrictions?

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

"....explosive sounds of amusement forever more."
(laughter hereafter)

Is THAT within the boundaries of your (curricula dicta)?

Judy Harris, 621-F Madison Ave., Charlottesville, VA 22903; CompuServe: 75116,1207; BITnet: jbh7c@Virginia.

Stager's Stuff

The Subject is Predicates
by Gary S. Stager

Predicates are one of the most often overlooked, and simultaneously, handiest features of the Logo language. They do not relate directly to turtle graphics and are therefore not often taught or discussed adequately in Logo texts. In this column I hope to illustrate what powerful programming tools predicates are and how they can be used simply to create interesting explorations into mathematics, language arts, logic, and programming.

What are Predicates?

Predicates are a special form of Logo operation (known in LogoWriter as reporters, see InLxual Challenges elsewhere in this issue) which evaluate a condition and *always* report (i.e., output) TRUE or FALSE. I like to think of them as independent agents; as checking machines or truth police who do part of my work for me (or my program) and report back the result of their investigation. Predicates reduce problems down to the simplest structure of logic gates or zeros and ones. A condition is either true or false and based on that determination some action takes place.

Conventions

Logo predicates are usually identified by a *P* or a *?* at the end of a procedure's name. LCSI dialects of Logo, with the exception of LogoWriter, have used a *P* to signify that a primitive is a *P*redicate operation. Versions of MIT Logo and LogoWriter use a *?* at the end of a predicate's procedure name to signify that the procedure will be in effect answering a question. I think that the *?* convention is easier for beginners to understand so I will use it throughout this column. If you are using a version of Logo which uses the *P* convention, simply substitute a *?* for *P*.

Some Examples of Primitive Predicates

When I say primitive predicates, I mean predicates that are built into Logo. We can also create our own predicate procedures as you will see shortly.

EQUAL? *expression1 expression2*
  Is expression1 equal to expression2?
MEMBER? *thing (word or list)*
  Is the first input an element of the second input?
NUMBER? *thing*
  Is the input a number?
NAME? *thing*
  Is the input the name of a global variable?
WORD? *thing*
  Is the input a Logo word?
LIST? *thing*
  Is the input a Logo list?
PROCEDURE? *thing*
  Is the input the name of a defined procedure?
PRIMITIVE? *thing*
  Is the input the name of a primitive?

INFIX PREDICATES
  =
  <
  >

Seeing Red

Since predicates are Logo operations/reporters, they must have a "spout" for outputting TRUE or FALSE and may or may not have "hoppers" for inputs (graphic below). (Again, see IntLXual Challenges in this issue.) Most predicates require one or more inputs, but don't have to as you can see in the following example.

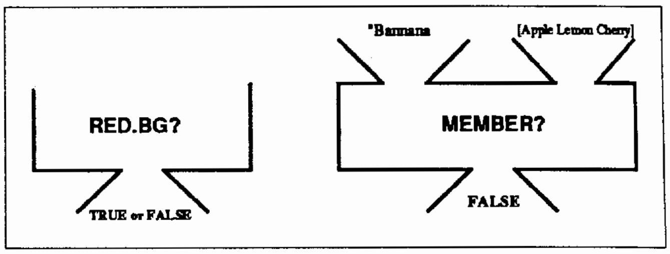
If I wanted to create a predicate that would tell me if my screen's background color is red, I might write it like this:

```
TO RED.BG?
 IFELSE BG = 4   [OUTPUT "TRUE]
                 [OUTPUT "FALSE]
END
```

or

```
TO RED.BG?
 IF BG = 4 [OUTPUT "TRUE]
 OUTPUT "FALSE
END
```

Note: The words TRUE and FALSE must always be outputted as Logo words with quotes before them. When a procedure uses OUTPUT (OP is the abbreviation) the procedure always terminates at that point. This is why IFELSE wasn't necessary above, even though it makes the intent of the procedure clearer..



We can shorten the above predicate by eliminating the IFELSE or IF conditional. The purpose of a conditional is to evaluate if a condition is true or false and then act accordingly. Because predicates (like the equal sign) only report true or false, we can just output the result of the condition, for example:

```
TO RED.BG?              TO RED.BG?
 OUTPUT BG = 4           OUTPUT EQUAL? BG 4
END                     END
```

Try these procedures if you don't believe me.

This RED.BG? procedure works, but is not too useful. It only checks if the BG is 4. The key to elegant Logo programming is to create procedures which can be generalized to other situations. We can make this predicate more useful and flexible by adding an input so Logo can check for any background color.

A Classroom Example

A fifth grade class I was teaching was studying mathematical factors. I quickly thought of a Logo activity for them to do which would allow them to experience the concept. The first part of the activity was to create a predicate "machine" which would tell if one number was a factor of another. From this problem we know that our machine must have two inputs (hoppers) and one output (spout) for TRUE or FALSE. Creating a factor predicate is an excellent problem because there several ways of solving it.

(The BASIC mode of thinking)

```
TO FACTOR? :FACTOR :NUMBER
 OUTPUT
 INT (:NUMBER / :FACTOR) = :NUMBER / :FACTOR
END
```

Note: If you are working in LogoWriter, copy the INT procedure from the MATH.TOOLS page.

(A More Logo -like mode of thinking)

```
TO FACTOR? :FACTOR :NUMBER
 OUTPUT (REMAINDER :NUMBER :FACTOR) = 0
END
```

REMAINDER is a Logo primitive which outputs the remainder of the quotient of two numbers. If the remainder is 0, the :FACTOR is a factor of :NUMBER. REMAINDER is useful for correcting the student's division homework problems which (for whatever reason) require a remainder.

Can you think of any other ways of building a FACTOR? machine?

Now that FACTOR? is written, the students can create a procedure which will give them a table of factors.

```
TO FACTORS :NUMBER :FACTOR
 IF :NUMBER < 1 [STOP]
 IF FACTOR? :FACTOR :NUMBER
       [(PRINT :FACTOR [ is a factor of ]
                            :NUMBER)]
 FACTORS (:NUMBER - 1) :FACTOR
END
```

```
FACTORS 100 7
```

prints a table of all of the numbers less than 101 which have 7 as a factor

The students were quite excited by this project and had the sense that the computer was cheating or doing their homework for them. In fact, they had solved the problem from the inside and probably have a greater understanding of factors than they would through drill and practice. This predicate and table combination can be adapted to a number of concepts.

Sex Machine

Now that I have your attention I would like to illustrate how predicates can evolve into interesting programming or curriculum relevant projects.

How might we write a predicate for determining if a number is even?

```
TO EVEN? :NUMBER
 OUTPUT (REMAINDER :NUMBER 2) = 0
END
```

```
TO EVEN? :NUMBER
 OUTPUT INT (:NUMBER / 2) = (:NUMBER / 2)
END
```

Here are some ways to write ODD?

```
TO ODD? :NUMBER
 OUTPUT NOT (REMAINDER :NUMBER 2) = 0
END
```

```
TO ODD? :NUMBER
 OUTPUT NOT EVEN? :NUMBER
END
```

As you can see in the second example of ODD?, the ODD? predicate can be evaluated in terms of an already existing EVEN? predicate.

```
TO LESSTHAN? :NUMBER1 :NUMBER2
 OUTPUT :NUMBER1 < :NUMBER2
END
```

```
TO GREATERTHAN? :NUMBER1 :NUMBER2
 OUTPUT :NUMBER1 > :NUMBER2
END
```

Why didn't I write GREATERTHAN like this?

```
TO GREATERTHAN? :NUMBER1 :NUMBER2
 OUTPUT NOT LESSTHAN? :NUMBER1 :NUMBER2
END
```

The VOWEL? procedure determines if a letter is a vowel. Instead of five separate IF conditionals, the procedure uses the predicate, MEMBER?, to determine if the input is contained in the list of vowels.

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

How might you write CONSONANT? ?

MEMBER? can be used in numerous situations, for example, the set theory concepts of subset, intersection, union, null set (EMPTY?) and for Venn diagrams.

How would we create a predicate which would tell us if somebody was a boy or a girl? First we must ask, "How do we know if a name belongs is a girl's name or a boy's name?" The simplest way of answering this question is that we have a list of names memorized, which are often added to, and we check the list in our head. This tells me that the predicates we are about to create needs to have a list of names in it. We will use MEMBER?, another predicate, to help us solve this problem.

```
TO MALE? :NAME
  OUTPUT MEMBER? :NAME [GARY MICHAEL BRIAN BILL
        MIKE PETE ROBERT ROB BOB ETHAN JASON STEVE
        STEVEN HENRY JIM JAMES DAN DANNY TOM THOMAS
        DANIEL]
END
```

```
SHOW MALE? "BRIAN
TRUE

SHOW MALE? "MARY
FALSE

TO FEMALE? :NAME
  OUTPUT MEMBER? :NAME [ CATHY TERRY MIRA KAREN
        SHARON MARY JULIE MARGARET GAIL GINA EMILY
        MARLENE MOLLY ARLENE ]
END

SHOW FEMALE? "TERRY
TRUE
SHOW FEMALE? "GWEN
FALSE
```

In the case of Gwen we know that this name is a girl's name and therefore the "sex machine" should get smarter by us adding her name to the list in FEMALE?.

Let's Build a Spelling Checker

By building simple "checking" machines we can expose some of the fundamental concepts of grammar, computer science, and knowledge itself. By modeling human or computer thought processes students develop their own personal "mind machine" which becomes better at solving problems by clearly identifying the nature of the problem and the steps involved in solving it. I have decided to build a rudimentary spelling checker to illustrate an actual application of Logo predicates.

The spelling checker project begins by asking your students (or yourself), "how do I know if a word is spelled correctly?" One answer could be, "by looking the word up in the dictionary," and another might be, "by thinking about how I learned to spell it or have seen it in print." In either case the solution to the problem involves "looking up" the word either in print or in one's head. This revelation hints that the CHECK? procedure may be very similar in construction to the VOWEL? procedure.

```
TO CHECK? :WORD
  OUTPUT MEMBER? :WORD [ CAT DOG BIKE BUS SCHOOL
        HOUSE IS A ARE THE AND IN UP UNDER OUT ]
END

PRINT CHECK "GARY
FALSE
PRINT CHECK "DOG
TRUE
```

We've stumbled across a bug. My Mother would be very disappointed to learn that *Gary* is not a word. We need a way to be prompted and asked to add new words to the CHECK? procedure's dictionary. The following is a LogoWriter example which illustrates this idea.

In this version you need to SELECT a word before typing CHECK.WORD. That is, you need to use LogoWriter's wordprocessing capabilities to highlight the word in question.

```
TO CHECK.WORD
 IF CHECK? SELECTED [ STOP ]
 WHAT.TO.DO SELECTED
END

TO CHECK? :WORD
 OUTPUT MEMBER? :WORD :VOCABULARY
END

TO WHAT.TO.DO :WORD
  CC
  (TYPE  "Is :WORD [ spelled correctly? ])
  IF READCHAR = "Y [ ADD.WORD :WORD STOP ]
  (TYPE CHAR 13 [ What word would you like to
       replace ] :WORD "with?)
  CUT
  INSERT READLIST
END

TO ADD.WORD :WORD
 MAKE "VOCABULARY LPUT :WORD :VOCABULARY
END

TO REMOVE.WORD :WORD
  MAKE "VOCABULARY (REMOVER :WORD :VOCABULARY)
END

TO REMOVER :WORD
  IF NOT MEMBER? :WORD :VOCABULARY [ OUTPUT
       :VOCABULARY ]
  IF EQUAL? :WORD FIRST :VOCABULARY
       [ OUTPUT REMOVER :WORD BF :VOCABULARY ]
  OUTPUT FPUT FIRST :VOCABULARY REMOVER :WORD BF
       :VOCABULARY
END

TO INITIALIZE
  IF OR NAMEP :VOCABULARY EMPTY? :VOCABULARY
       [ MAKE "VOCABULARY [] ]
END
```

Note: LogoWriter does not save global variables, such as :VOCABULARY, with the page so you must type

```
PRINT :VOCABULARY
```

either on the front or flip-side before leaving the page so that you can re-use your vocabulary list next session.

Of course, all kinds of complex embellishments could be added to this spelling checker, however the process of building and understanding just this small and accessible program is of the highest importance.

This spelling checker is intended to be more educational than practical. I have written a full-blown spelling checker in LogoWriter (Apple) which is enormously amusing (it knows no words and gets slower as it learns new ones). It automatically reads a passage of text and prompts the user to make corrections or additions to the dictionary. I'd be glad to share it with anyone who writes me.

Gary S. Stager, 12 Locust Place, Wayne, NJ, 07470; CompuServe: 73306,2446; AppleLink: K0331.

InLXual Challenges

Structuring the Structure
by Robs Muir

Certainly the most difficult part of learning a new language is mastering unfamiliar syntax. The learning of new vocabulary is trivial when compared with the enormous difficulty of putting those words together in meaningful patterns. As evidence, we can note that children learn substantial quantities of words long before they can put them together in order to express complex ideas.

Learning computer languages is no different. Communicating successfully with a computing device is a careful balance of both *syntax* and *semantics* - word order and usage, and meaning. Learning new commands usually is not too difficult for students; learning *how* to use them often requires real suffering. Fortunately, most computer languages provide some feedback when your meaning is not conveyed according to the rules. Usually, these rebuffs are instructive, or at least, non-judgmental. (This is more than can be said for the average French speaker, when correcting my broken French!) The trick is to learn to interpret the error statements so that you can improve your "communication" skills.

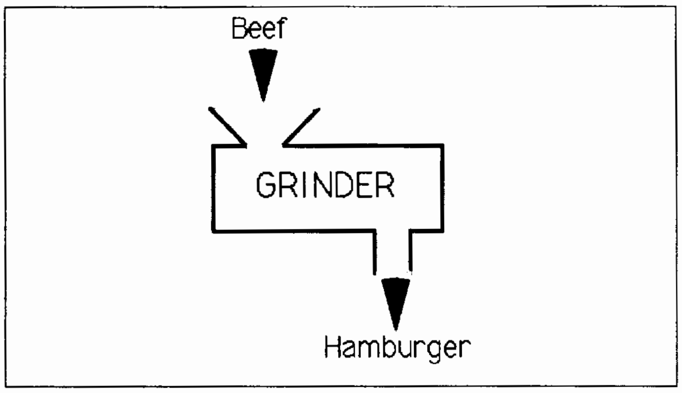
One of the significant contributions of Logo to the programming community is its careful attention to informative error statements. I DONT KNOW HOW TO FROWARD sure beats SYNTAX ERROR. Careful study of Logo's error messages can have a real impact in a programming class; students won't always need your expert advice when they are able to understand *why* their Logo statements don't work.

One fundamental rule of Logo grammar is the distinction between COMMANDS and OPERATORS. (In LogoWriter, OPERATORS are called REPORTORS) This distinction is as important as the distinction between nouns and verbs in English. The rules for these Logo "parts of speech" are, fortunately, much simpler than English.

- With the exception of a few "magic" words such as TO, END, TRUE and FALSE, *all* Logo primitives are either COMMANDS or OPERATORS.
- The first word of a Logo statement must be a COMMAND.
- COMMANDS *produce* results. OPERATORS *pass on* objects to other COMMANDS or OPERATORS.

Major Operators

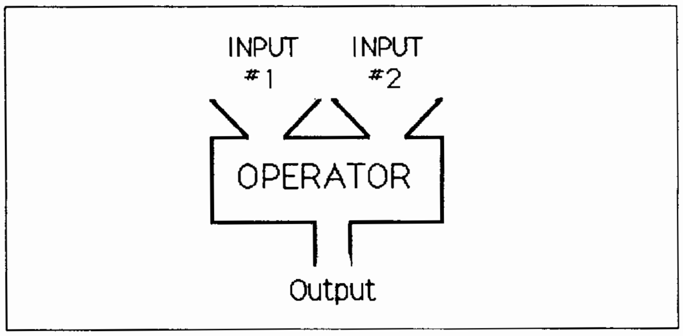
OPERATORS always generate an object; they can be thought of



as meat-grinders, that is, you drop something in and something else comes out.

(Operators could also be envisioned as mathematical *functions* such as $f(x) = x^2 - 3$ or $y = x^2 - 3$. If you put in an x, you get back an $f(x)$ or y. However, meat grinders are less intimidating.)

Let's diagram a Logo OPERATOR



Although Logo OPERATORS may have one or more inputs, most OPERATORS have two. Typical Logo OPERATORS that have two inputs are primitives such as WORD, SENTENCE, + (or SUM), LIST, QUOTIENT, etc. Such as

SUM 45 112	produces–>157
SENTENCE [WHO] [DONE IT?]	produces–>[WHO DONE IT?]
WORD "NOW "HERE	produces–>"NOWHERE
45 + 112	produces–>157

The usual word order for Logo would be to have the OPERATOR first, followed by the various inputs. This strict prefix notation has been relaxed for some familiar arithmetic operators like +, –, ., and /. However, these infix operators are the exception, rather than the rule. Incidentally, did you know that Apple Logo allows arithmetic operators to function as both prefix <u>or</u> infix operators? Try this in Apple Logo:

+ 3 4 produces 7, or
/ 8 2 produces 4.

InLXual Challenges — CONTINUED

Challenge #1
Name several examples of Logo primitives which are OPERATORS that require only one input.

Command Performances

Notice that if you type the above Logo statements, Logo will still produce an error statement.

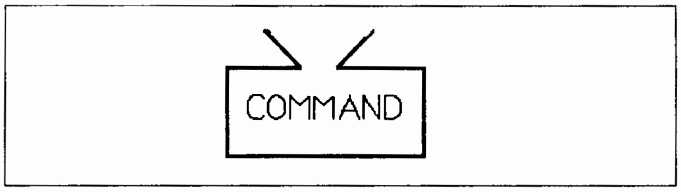
```
SUM 45 112
I DON'T KNOW WHAT TO DO WITH 157
```

If you add a new class of Logo primitive—the COMMAND—before the SUM OPERATOR, Logo will cease to complain.

```
PRINT SUM 45 112
157
```

PRINT is an example of a COMMAND. COMMANDS require zero, one, or more inputs, yet they do not produce any output—they cause Logo to do things by "side effect." FORWARD, RIGHT, and SHOW are all COMMANDS. All Logo statements must have a COMMAND as the first object.

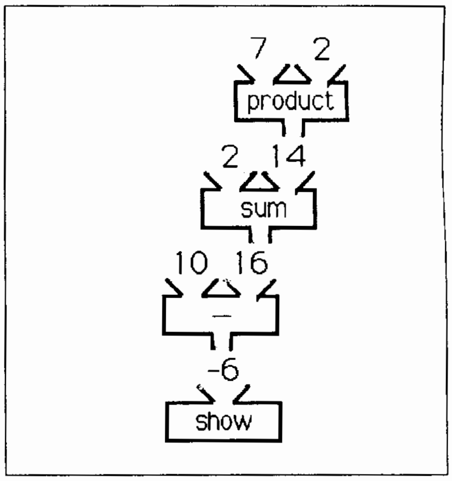
Here is a visual aid



All of these diagrams can be useful in understanding more complex Logo statements. For example, what will Logo produce from the following statement?

SHOW 10 – SUM 2 PRODUCT 2 7

Did you predict a minus six (–6)? Does this help?



Challenge #2
What are some examples of COMMANDS that require more than one input?

This notational scheme is a favorite of E. Paul Goldenberg (who along with Wallace Feurzeig, has written an excellent new book, *Exploring Language with Logo* [MIT Press, 1987. ISBN 0-262-57065-3]. Although this long-awaited book is intended to focus on natural languages, it relies on artificial languages like Logo to assist in language explorations.

Challenge #3
Identify some Logo COMMANDS that require no inputs.

Challenge #4
How would you diagram MAKE or IF?

Challenge #5
Are there any OPERATORS that require no inputs, yet produce some output?

Challenge #6
What might OUTPUT look like as a diagram?

For a major project, produce a program that takes a list containing a valid Logo statement as input and builds a visual diagram of the statement. For example:

DIAGRAM.IT [PRINT SUM 4 5]

What should the turtle draw? How will you represent the input [PRINT SUM 4 5]? This is no simple task. Think about it as you continue to use Logo. Think about it the next time you get a Logo error statement.

Robs Muir, 1688 Denver Avenue, Claremont, CA 91711; CompuServe: 70357,3403; BITnet: MUIRR@CLARGRAD.

Answers to this month's *InLXual Challenges* article may be found on page 3.

Testudinal Testimony

Research on Variables, Algebra, and Logo. *Part II: Misconceptions and Suggestions*

by Douglas H. Clements

In the previous column, we reviewed evidence that programming has the potential to:

- help students take a more active view of equations;
- enhance the understanding of variables for students from the primary grades to high school; and
- provide an "entry" to the use of the powerful tool of algebra.

Research also has uncovered many difficulties and misconceptions experienced by students. Fortunately, these studies have also produced teaching suggestions that may ameliorate some of these problems.

Variables in Mathematics

Let's begin by reviewing the types of misconceptions students hold about variables outside the domain of programming. Wagner (1983) tells of a teacher preparing students for the x, $x + 1$, ... literal symbolism by starting with a numerical example. Students easily answered that the next consecutive integer after 17 was 18 and that one added 1 to 17 to get 18. The teacher continued: "'Now suppose we use x to represent an unknown integer. How can we write the next consecutive integer after x?' Without hesitation, the response was, 'y.'" (Wagner, 1983, p. 474). Such confusion between the linear order of the alphabet and the whole numbers is common for novice students. But here, more experienced students also were seduced by the similarity.

Wagner gives two reasons that literal symbols are fairly easy to use, but hard to understand:

- Literal symbols are like numerals, only they are different.
- Literal symbols are like words, only they are different.

Although it is helpful to point out some of the ways literal symbols are similar to numerals and words, students often overgeneralize. Therefore, it is important to identify both similarities and differences.

Variables are like numerals in that:

- Some letters, such as π and e, represent actual numerals.
- Both often appear together in mathematical statements such as $n + 8 = 14$. Thus, they often look as though they behave like numerals. Students may believe they are temporary numerals—symbols you write until you know the real (single) numeral.

Variables are different from numerals in that:

- Letters can represent simultaneously many different numbers (e.g., $0 < x < 50$).
- Letters often represent random, variable, elements (e.g., "a point P").
- Letters placed next to other letters or numerals indicate multiplication (e.g., $6mn$), whereas numerals placed next to each other indicate place value assignments (e.g., 634).
- Unlike numerals, the signs attached to literal symbols do not always match their values. That is, the value of x can be negative (e.g., $x = -3$) and, similarly, the value of $-x$ can be positive. One can sympathize with the student who, not recognizing this, tries in vein to understand the definition for absolute value:

$$|x| = x \text{ if } x \geq 0$$
$$|x| = -x \text{ if } x \leq 0$$

Variables are like words in that:

- Both can act as placeholders (e.g., pronouns).
- Letters often are chosen to represent abbreviations for words (e.g., n for number). While this is a helpful teaching technique, recall the students and professors problem from the previous column. Students may come to believe that S represents students, instead of the *number* of students.
- Both have different meanings in different contexts.

Variables are different from words in that:

- Although letters and words assume different meanings in different contexts, they differ in their consistency throughout a context. Variables must hold a single value (e.g., in $2x + 15 = x^2$). This is not true for words, whether used in natural language (a tear came to her eye when she saw the tear in her dress) or mathematics (the sum of an even number and an odd number is always an odd number).
- Letters are not associated with fixed sets of meanings. "That is, we could never compile a dictionary of the meanings of literal symbols as they are used in mathematics" (Wagner, 1983, p. 477). We are free to delimit the meaning and the domain in any way. This provides variables with their generality. We can also choose any symbol we like for any referent. This gives mathematical language flexibility (e.g., in substitutions; if $y = 2x + 1$ and $z = 3y$, then $z = 3(2x + 1)$).

Teachers who recognize how variables are similar to and different from numerals and words are better prepared to gain insight into common misconceptions and to address comparisons among variables, numerals and words explicitly. They are also better prepared to understand difficulties students have with variables in the context of computer programming.

Variables in Logo

Hillel and Samurçay (1985) set out to find whether the variable notion in Logo can be given meaning by 9-10 year olds. They introduced the variable concept through procedures that take inputs (which they call "generalized procedures"). In this way, the variable concept also encompassed the concept of procedure. Neither of these concepts, according to the authors, arises spontaneously out of children's activities. So, they both require a high degree of elaboration fostered by organized instruction.

Hillel and Samurçay found that students may:

Testudinal Testimony — continued

- ignore the opportunity to use variables, instead writing multiple fixed procedures;
- declare a variable in a procedure, but then not use it within the body of the procedure;
- attempt to give inputs to a fixed procedure;
- believe that a variable might have different values within a procedures (this problem may arise because children fail to conceptualize how an input is passed on to the commands within the procedure); and
- confuse what the variable stands for.

In addition, students using generalized procedures often rely on the perception of global changes ("It makes a bigger [or smaller] square"). This approach does not lead to greater understanding, especially when more complex procedures are involved. For example, students might use a circle procedure and consistently relate the variable to the size of the circumference, although it actually represents the diameter. Use of names such as SQUARE :SIZE exacerbate this problem. Providing, and encouraging children to create, variable names such as :LENGTH.SIDE may be helpful. In all cases, children might be led to analyze the procedure itself to ascertain what is varying.

In summary, to *use* generalized procedures well, students should:

1. Accurately identify what is varying.
2. Assign appropriate values.
3. Construct an interface which is often itself dependent on the value given as an input to the procedure.

Similarly, when *defining* a generalized procedure, they should:

1. Identify what is varying and assigning it an accurate name. Children have to find all the commands within the procedure to which they need to assign a variable input. This is, of course, not always obvious. For instance, the length of an arc may be control by a variable input to REPEAT, when the list repeated is [FORWARD 1 RIGHT 1].
2. Name and declare the variable.
3. Operate on the variable when appropriate. This involves passing the variable to each command as needed. It also includes passing the variable to commands and subprocedures and modifying it (e.g., changing $:x$ to $:x + 5$).

Teachers might model these steps as they introduce students to generalized procedures.

In investigating major types of programming misconceptions, du Boulay (1986) found that students frequently misapplied analogies. For example, students are often taught to think of variables as a box. However, many students then believe that—like a box—a variable might hold more than one value!

As a second example, computer programming has done a service in encouraging the use of meaningful names for variables. Recall, however, that students often mistakenly believe that mathematics literal symbols, like words, are associated with fixed sets of meanings. Using the analogy of "Logo variables as meaningful names" engenders similar misconceptions. Similarly, using meaningful names may lead students to believe the names are meaningful to the computer! For these reasons, du Boulay suggests the occasional use of nonsense names as an exercise. This suggestion was echoed by Sutherland (1987), who warned that children may place undue importance to variable names. She proposed the following sequence: Start with meaningful names, use nonsense names as an exercise, then return to meaningful or abstract naming schemes.

Sutherland's main purpose was to investigate whether certain Logo experiences would provide pupils with a conceptual basis of variable. It was hoped that this would enhance their work with traditional paper and pencil algebra. As other researchers, she found that pupils rarely choose projects which need the concept of variable. They also resist using the variable concept even when it is suggested. When they did use the concept, they evinced misconceptions. For example, they brought their often limited concept of number to the Logo situation, and thus restricted the idea of a range of numbers to the positive integers.

Therefore, Sutherland introduced structured tasks in an attempt to show students the potential of the variable concept. First, students wrote a procedure to draw the letter "L" of a fixed size. They then were shown how to change their fixed L procedure to a general L procedure. They ran the procedure with a variety of inputs, including decimal and negative numbers. For example, they had to use decimal inputs to create the biggest and smallest letter on the screen. This provided motivation and purpose for using such numbers, and also developed the understanding of variable as representing a range of numbers. Later tasks included writing other variable letter procedures and creating designs that combined these procedures.

Sutherland used the idea of function to forge links between Logo and traditional algebra. Students constructed mystery function machines. Their partners had to guess the function and write a similar procedure themselves. This helped them see that changing a literal symbol does not change what the symbol refers to. Also, letting students choose any variable and function name was quite motivating for them. Finally, students completed paper and pencil exercises that presented input/output tables and asked students to write corresponding Logo procedures.

Were the tasks successful? Interviews revealed that more Logo than control students accepted the idea of using a variable to solve problems and accepted a lack of closure in an algebraic expression. Students were able to develop their intuitive understanding of pattern and structure to the point where they could make a generalization and formalize this generalization in Logo. Sutherland claimed that it is not easy to provide pupils with this type of experience in traditional algebra. She suggested that teachers use Logo as a context for generalizing and fomalizing, rather than attempting to contrive problems in beginning algebra. Then, links between the use of variable in the two contexts can be made.

As always, it is also important to plan for transfer. Researchers from Bank Street reported that after a year of programming experience, high school students had only rudimentary understanding of variables

Testudinal Testimony — CONTINUED

(Kurland, Pea, Clement, & Mawby, 1986). They did not improve on using letters to represent variables and on translating prose descriptions into symbolic expressions. The researcher suggested that to facilitate transfer, teachers would need to provide:

- multiple examples of the application of a skill or concept;
- links to real–world problem–solving situations;
- content area instruction; and
- abstract descriptions of the skills and concepts to be applied.

Finally, Soloway, Lochhead, and Clement (1982), with whom we opened last month's column, made four increasingly provocative suggestions:

- Recognize that the mathematics courses students take is not the same as the mathematics they possess.
- Integrate computer programming into the mathematics curriculum.
- Redefine much of the mathematics curriculum to be programming based.
- Teach algebra as an integral part of programming.

The challenge is to make the next step, building computer programs that link Logo and mathematical contexts for variables. Several projects attempting to do just this will be featured next month.

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

References

du Boulay, B. (1986). Part II: Logo confessions. In R. Lawler, B. du Boulay, M. Hughes, & H. Macleod (Eds.), *Cognition and computers: Studies in Learning* (pp. 81-178). Chichester, England: Ellis Horwood Limited.

Hillel, J., & Samurçay, R. (1985). *Analysis of a Logo Environment for Learning the Concept of Procedures with Variable.* Unpublished manuscript, Concordia University, Montreal.

Kurland, D. M., Pea, R. D., Clement, C., & Mawby, R. (1986). A study of the development of programming ability and thinking skills in high school students. *Journal of Educational Computing Research, 2*, 429-458.

Soloway, E., Lochhead, J., & Clement, J. (1982). Does computer programming enhance problem solving ability? Some positive evidence on algebra word problems. In R. J. Seidel, R. E. Anderson, & B. Hunter (Eds.), *Computer Literacy* (pp. 171-185). New York: Academic Press.

Sutherland, R. (1987?). *What are the Links Between Variable in Logo and Variable in Algebra*? Unpublished manuscript, University of London Institute of Education, London, England.

Wagner, S. (1983, October). What are these things called variables? *Mathematics Teacher*, pp. 474-479.

University of Virginia; Charlottesville, VA

Logo Connections:
A Two-week Retreat will be offered at the Univeristy of Virginia in Charlottesville, VA August 1-12. The intent of the retreat is to bring together teachers across the country who are interested in connecting with each other and learning how to create "Logo Connections." From its beginning, Logo was designed as a language which could be connected to objects in the outside world, such as switches, motors, lights, and sensors. In addition, Logo work has always emphasized the importance of interpersonal learning through connections with colleagues and friends. Both of these aspects of connections will be emphasized at the retreat in an informal, hands-on environment.

The workshop faculty of Glen Bull, Gina Bull, Judi Harris, Tom Lough and Cheryl Wissock have a wide range of interests and expertise, ranging from pendula in physics to switch inputs for handicapped users. Among them, they have written numerous books and several hundred articles on Logo.

Registration for the two-week retreat will be limited to a maximum of twenty participants. Air conditioned dormitory rooms will be available at Summer rates for those who require housing. Three hours of graduate credit will also be available through the University, on an optional basis. A wide variety of social experiences throughout the two-week period will emphasize the importance of personal relationships in teaching with Logo.

Participants must have intermediate Logo programming skills prior to the retreat. For an application, call Peggy Marshal at 804/924-7471 or write to:

Logo Connections
c/o Tom Lough
Curry School of Education
University of Virginia
405 Emmet Street
Charlottesville, VA 22903

PenPoints

Logo Book Reviews with ASTROLUG by Gayle Lawrence

Thinking in Logo by Gini Shimabukuro is aptly sub-titled "A Sourcebook for Teachers of Primary Students." The author has developed a comprehensive resource for the classroom teacher who wishes to begin to use Logo with his/her students, but has no previous computer or Logo experience. This sourcebook contains ten lessons, or mini-units, each devoted to one or more Logo turtle graphics commands. For example, Lesson Four covers BACK and LEFT; Lesson Ten deals with REPEAT. Several of these lessons emphasize exploration of geometric shapes: squares, triangles, rectangles and circles. Each lesson follows the same format and includes: a statement of objectives; a reminder to "center" (more on this later); playing turtle activities; one or more computer hands-on activities and associated reference pages; teacher helps; one or more Think Sheets (reproducible activities for the students to do independently or with the teacher); and a summary chart for content learned up to that lesson. *Thinking in Logo* is published by Addison-Wesley (2725 Sand Hill Road, Menlo Park, CA 94025, telephone (415) 854-0300).

In addition to the complete lesson plans and related support materials, there is a useful set of appendixes, which includes: a teacher tutorial; keyboarding activities; seven Logo programs; a collection of classroom aids; a resource guide; 63 Think Sheets; and an answer key. This is clearly one of the most complete appendixes to be found in a book for primary teachers using Logo.

One's overall impression of this Addison-Wesley publication is that it is a well-planned product. It has been designed to be taken apart and put into a three ring binder for easy use. Numerous illustrations delight the visual learner in this author and complement the text nicely. The layout of the Think Sheets is very attractive. They will make crisp copies and have definitely been prepared with the "age of photocopying" in mind.

The author's philosophy of learning has been woven into the clearly described and carefully thought out lessons. One example of this is her presentation of the concept of *centering* as a means of "clearing students' screens" or removing distractions or extraneous thoughts before beginning each lesson. Soft music and guided imagery are used to aid students in *centering* themselves before learning begins. I am intrigued by Ms. Shimabukuro's ideas and am persuaded to read more about creative visualization as it relates to cognition. Fortunately, she must have hoped or expected that would happen because she has provided several references in one of the appendixes for just that purpose.

The TEACHERS HELPER sections within each lesson offer practical tips for working on the topic of that particular lesson and also suggest opportunities for challenging the students to visualize, use alternate thinking, work on group projects and so on. These tips would be quite helpful to a teacher using Logo and the computer for the first time.

The cute turtles and coloring activities included on many of the Think Sheets may be quite appealing to the young learners initially, but I was struck a number of times as I flipped from the lessons back to check the Think Sheets that were associated with them. There were too many activities and they were too similar. At some point, these sheets will become just another worksheet. The author states in the introduction that the pacing of these lessons depends upon the group and teacher. She emphasizes the importance of the teacher's role in adapting *Thinking in Logo* to the group and even cautions against allowing the students to become dulled. There is a very important issue here. Without careful monitoring and selective use of the Think Sheets and lessons, the very exciting opportunity for fostering thinking skills in students could be lost in the monotonous cut, color, and paste of the Think Sheets.

Although I do have some concerns about use of the Think Sheets, two of them were of special interest. The author suggests a "Surprise Program," which is created by selecting cards bearing known commands at random from an envelope and then typing them into the computer. (Think Sheets are provided for the commands and for creating the envelope). I have seen this idea before and have used it with students in the computer lab. It is successful and meaningful because children readily see the significance of the order of the commands to the end product. This skill can be very useful to students when debugging. The Think Sheets for presenting the Logo `REPEAT` command offers a good visual model for it. The child is to count the number of repeated objects displayed, write that number after the word repeat, and then draw that object in the box, which is beside the number.

Despite the author's urging "to be creative" and "to explore," there were not as many opportunities for free exploration as I would desire. The author recommends using alternate approaches for reaching the same goal and provides opportunities to "make your own path," but the total approach is really quite structured and very task oriented. I sense that even the exploring that is suggested is really quite self-limiting. For example, in Lesson Three, in which students were exploring squares, the teacher taught them to use `RIGHT 90`.

Another concern I have about this book is that it begins with the use of regular Logo commands and numerical inputs rather than the use of a single-keystroke program such as `INSTANT`, *Kinder-Logo* or *EZ Logo*.

This emphasis on syntax, lists of commands and numerical inputs is inappropriate for young students. It dictates attention to details that prove frustrating to this age group and places greater emphasis on keyboarding knowledge than is necessary for young computer students. It makes the mastery of syntax an end rather than a means to an end. I would like to see the students carrying out the activities and creating squares and other geometric shapes without needing to be taught to use particular angles at the outset. The specifics of syntax and the understanding of angles can develop later as the natural extension of having first used simpler commands to explore the same powerful ideas.

Because the appendixes in *Thinking in Logo* contribute substantially to this book's value as a resource for a classroom teacher, it is worth noting some of the information contained within them. Some-

PenPoints — CONTINUED

times it is difficult to evaluate the completeness of a tutorial for a beginner, when you are no longer a beginner, but I think Ms. Shimabukuro has presented a very thorough and succinct introduction to Logo turtle graphics in the "Teacher Tutorial." The Logo programs, which the teacher can put on a files disk, are simple but quite appealing to children of this age. For example, there are programs for three mazes of increasing difficulty which provide students with practice in maneuvering the turtle. In the appendix on classroom aides, tips on equipment and printing are provided. A long list of books about turtles and several craft ideas for turtles complete this section. For each of Logo books referenced in the "Logo Resource Guide", the author gives a brief descriptive comment. She also offers lists of general readings, music for centering, and Logo teaching aids and publications. The 63 Think Sheets; a collection of reproducible task cards, keyboards and turtle images; as well as a short answer key bring the reader to page 259 feeling that there really is a lot of material here!

Summary

As an elementary computer specialist working in a lab program, I would not use the lessons in *Thinking in Logo.* I might have use for some of the Think Sheets and would certainly use other materials included in the appendixes. The classroom teacher, just beginning to use Logo with primary students, would find a number of worthwhile ideas and resources in *Thinking with Logo* . It could be used as a reference and combined with other material now available in developing a program which uses Logo to foster thinking skills in students. With the recommendation that a teacher using this reference focus on thinking and use Logo and other means to nurture their students' ability to think, I give this book a penup. (An ASTROLUG *penup* is equivalent to a *thumbs up*.)

Gayle Lawrence has been an elementary computer resource specialist in the Brittonkill Central School District in Troy, NY for four years.

LogoPals

by Barbara Randolph

Hobbies are fun, aren't they? Learning string figures, or cat's cradles as some people call them, is a new hobby of mine. Using a loop of string you weave designs on your fingers. What's really wonderful is that these step-by-step patterns originate from cultures all over the world — from Japan, the Philippines, and Hawaii to Native American Indian cultures, from south America to Africa and Australia. Do you have any string games you would like to share with me or your LogoPal? (If you would like a copy of directions for a few I have learned, send your request along with a self-addressed stamped envelope.)

Here are some new LogoPals who enjoy their own fascinating hobbies:

- Carley Tallman (Seattle, Washington USA): My favorite hobby is playing piano. My favorite composer is Mozart. February 9th is my birthday and I am in the third grade.

LogoPals — CONTINUED

- Sisi Kapp (Seattle, Washington USA): My hobbies are coin collecting and origami [paper folding]. My favorite Logo activity is the flip side. I am 8 years old and in second grade.
- Ayantu Bedada (Seattle, Washington USA): One of my favorite hobbies is horseback riding. I like gymnastics too. I am a third grader. I am eight years old.
- Jean Domalis (Seattle, Washington USA): One of my favorite hobbies is horseback riding. I like gymnastics too. I am a third grader. I am eight years old.
- Jean Domalis (Seattle, Washington USA): My hobbies are astronomy and my paper route. Paddleball is my best sport. I am nine and in third grade.
- Brett Sigles (Seattle, Washington USA): Drawing is my favorite hobby. I enjoy basketball and baseball. I also like planes. I am a second grader. I am 7 years old.
- Joel Murry (Seattle, Washington USA): Science and reading are two of my favorite hobbies. Juggling, football and unicycling are the sports I enjoy best. I am 8 years old and a second grader.
- Ben Driscoll (Seattle, Washington USA): My first favorite hobby is fishing. My second is Logo! Some of my other interests are books, games and clubs. I am nine years old and in third grade.
- Ann Ng (Seattle, Washington USA): Riding a bike is my favorite hobby. Baseball is my favorite sport. I am a second grader and I am seven years old.

Each of these boys and girls would like to correspond with another Logo student anywhere in the world. Teachers, your students are welcome to request any of these students for penpals or any of the others already available in our LogoPal network.

Just have them write a letter to me telling about themselves, their interests and hobbies, their age and grade, what they enjoy about Logo. They can also tell the name of their Logo teacher and the months during which their school is in session.

Students in the USA need to send a self-address stamped envelope. Those outside the USA should enclose international postal coupons (purchased at the post office) for a 1-ounce or 28-gram reply.

Write to: LogoPals, c/o Barbara Randolph, 1455 East 56th Street, Chicago, Illinois 60637, USA.

A Special Request

All LogoPals who began writing to each other during the last school year (1986-87) and who have continued during the school year are requested to send a letter to me indicating that you are still LogoPals. Secondly, if you and your LogoPal have developed any special projects together or shared special Logo activities, write and tell me about that too. This request is VERY IMPORTANT and I would appreciate hearing from you as soon as possible! Thanks!

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

LXIONARY

By Bill Craig

As I have done in previous columns, I offer something old and something new. The old comes from Uri Leron's contribution to the Logo 85 Theoretical Papers and the new comes from an active group of Logo practitioners in New Zealand.

"Some Thoughts on Logo 85" by Uri Leron, *Logo 85 Theoretical Papers*, Massachusetts Institute of Technology, 1985.

The paper serves as Leron's commentary on the state of the Logo community in 1985. Among these comments is a belief that Logo practitioners had not encouraged the same free and open exchange of ideas among themselves that they were claiming to encourage in their students. Also, Leron stated that while there was an impressive growth in volume of Logo use between 1980-1984, "it has nearly ceased to grow in depth." He points to Logo literature published during these years to illustrate the point.

These impressions are accompanied by a list of unresolved issues. Included are the following,

1. The Logo teacher's role. How much intervention and what kind is appropriate?
2. When is and what kind of Logo curriculum and written materials are appropriate?
3. Should students work with Logo individually, in pairs, or in groups?
4. How can Logo learning be researched and reported?
5. Teacher training and Logo.

The importance of this article is that these impressions are still relevant and the 1985 issues are still hotly discussed and unresolved in 1988. While I found it interesting that the article was still relevant in 1988, it is about time we generated answers to some of these questions.

"Logo, Teacher Intervention, and the Development of Thinking Skills" by Wing Kee Au, Jane Horton, and Ken Ryba, *The Computing Teacher*, November, 1987.

Uri Leron will be happy to know that these authors have addressed the issue of appropriate teacher intervention. Recent LXionary columns have highlighted the debate among Seymour Papert and other academics about the value of Logo instruction and methods for verifying this value. Given the passion and importance of that debate, this article is important reading for Logo users. The authors have attempted to define the type of learning that comes with Logo instruction and very specifically define the conditions under which this learning occurs.

The basis of the article is that "cognitive development gains are likely to occur in response to effective teacher intervention." . That statement is not news. What is new is the extent to which the statement is supported with examples. A process oriented, as opposed to content oriented, approach is advocated with three types of teacher interventions detailed: worksheets, questioning techniques, and the provision of a "socially reflective and interactive environment." An outline of a typical lesson is provided as well as a "Logo Environment Checklist" which provides criteria for evaluating effective worksheets, questioning techniques, and classroom environment.

Work like this will provide credibility to the claims we have all been making about the value of instruction in and with Logo. What these authors are saying is that under certain types of conditions, certain gains can be made. Most importantly, these conditions are defined in such detail that teachers will not have to guess whether their methods meet the criteria for effective instruction.

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

British Logo User Group Conference 1988

This year's conference of the British Logo User Group will be held in the College of Saint Paul and Saint Mary, Cheltenham, from Friday 2 September to Sunday 4 September, 1988.

This conference is intended for primary and secondary school teachers, advisory teachers, teacher trainers, researchers in education and anyone else interested in the use of microm-computers in schools, in particular in the Logo programming language.

This conference will include a series of activities for Beginners in Logo, and a chance to hear about teachers' experiences in their classrooms. There will be a trade exhibition.

Conference fees:

	Resident	Non-resident	Sat. only
Members	£80	£55	£25
Non-members	£90	£65	£35

For bookings or further details write to:

The British Logo User Group
P.O. Box 79
WALSALL, West Midlands
WS5 3RW

International Logo News

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

Logo Exchange Continental Editors

AFRICA	ASIA	AUSTRALIA	Europe	Latin America
Fatimata Seye Sylla	Hillel Weintraub	Anne McDougall	Harry Pinxteren	Eduardo Cavallo &
Lab Informatique et Ed	Doshisha Int. Sch.	Faculty of Education	Logo Centrum Nederland	Patricia Dowling
BP 5036 Dakar	Tatara, Tanabe-Cho	Monash University	P.O. Box 1408	Instituto Bayard
Senegal, West Africa	Kyoto-fu Japan	Clayton, Victoria 3168	BK Nijmegen 650	Salguero 2969
	610-03	Australia	Netherlands	Buenos Aries,Argentina

Greetings from "America's Paradise." No, I am not on an extended vacation but am now an associate professor of educational technology at the University of the Virgin Islands on St. Thomas. Having spent the last two years in Singapore and Helsinki, this stop offers another opportunity to learn about computer education and Logo in a new region of the world.

This month's column summarizes a five-year Logo effort in Belgium and a five-minute Logo experience on the island of Tortola in the Caribbean's British Virgin Islands.

The British Virgin Islands have approximately 200 primary and secondary teachers. As part of a teachers' professional day on the fourth of January, the BVI Department of Education conducted three one-hour workshops on computer uses in education. Seven topics were to be included in the one-hour session of 70 teachers each — data bases, spreadsheets, thought processing, graphics, word processing, CAI and Logo. The BVI had recently received three IIe, two IIGS, and two Macintosh computers.

The plan was to divide each of the three groups of 70 teachers into seven groups of ten and have them spend five minutes seeing a demonstration of each type of software. Every five minutes the teachers would rotate to the next application. I was to arrive a day early and "train" the teachers who were chosen to demonstrate the software. The director of math and science for the BVI told me he had "heard of Logo" and would like to give the 21 five minute Logo demonstrations. Welcome to the developing world!

I was beginning to ask myself why I accepted this assignment. Twenty years' experience teaching told me this was never going to work. However, after hours of training the volunteers and fine tuning the timing, the feeling was that the next day would be fruitful.

Fifteen minutes before the presentations were to begin I was informed that the thought processor and Logo tutor would be unable to make it. However, not to worry, they had found another volunteer who had once browsed through a BASIC book and could probably pick up enough Logo in ten minutes to introduce all the BVI's 200 teachers to the language.

The day following the workshop, education ministry officials informed me that, according to answers on questionnaires, the teachers were very enthused to learn more about computers — especially Logo. Somehow the introductions had had an effect. This experience certainly demonstrates another reason why Logo is such a powerful language — it only takes "ten minutes" to train teachers how to both use it and train others!

Now that you are probably breaking out in a cold sweat, I had better quickly go on to a much more thorough Logo project conducted at the State University of Gent in Belgium. This information comes to us via our European correspondent, Harry Pinxteren, and it was written by M. Valcke who was involved in a project that integrated Logo learning environments into the mathematics curriculum of the Flemish primary schools of Belgium. Valcke reports:

The Gent Logo project can be split into two parts: a formative part (1983-85) and a summative part (1985-87). In the formative part the potential of working with the Logo language was explored, evaluation instruments were constructed, a special handbook was written, a curriculum analysis was carried out... During the summative part of the project, two research approaches were adopted. A first study focused on the qualitative evolution of planning behavior of pupils when working with Logo. Pupils had already been working for two years with a special Logo resource package that not only influenced mathematical concepts and skills, but also their planning skills. A second study is of a qualitative nature. The first results of this study were presented during the EuroLogo conference, Dublin, Ireland in September 1987 and also at the 'Leren met Logo' conference in Nijmegen, Holland in November.

During the exploratory formative phase, a lot of problems emerged in relation to the integration of Logo activities *into normal classroom practice*. This forced us to adopt our approach and to change our evaluation study into an innovation study. The following

International Logo News - Continued

observations directed our thinking and the actual study.

1. The school environment is a very complex research setting. Bringing a new learning tool into this setting not only influences the parameter 'media' of the educational environment, but also other variables. We tried to control or to influence in the following variables:

   - context (infrastructure, time table, classroom organization),
   - personal teacher characteristics (aspirations, philosophy, motivation level),
   - training status (initial training, additional background),
   - teaching practice (objectives, media/tools, teaching/learning strategies),
   - evaluation (methods to observe or analyse learning benefits),
   - educational support structure (availability of continuous training opportunities, availability of support facilities).

2. Although much is learned when children use Logo, this Piagetian 'learning' is of an implicit nature. In order to link this learning to the study of mathematics and problem solving, it is necessary to explicate the link between Logo and the course objectives. This is especially true when transfer effects are expected.

3. Logo is a toy and a tool for the learning of mathematics, but its 'tool-nature' is too primitive. In our view, children working with Logo are still bothered too much with technical programming skills and too little with content-related topics. Therefore, a large set of microworlds has been developed to sustain specific mathematical concepts or skills.

With these observations in mind, a Logo resource package was created and is considered the core of the innovation project. The package presents a set of Logo programming activities. Each activity is a kind of restricted Logo environment or microworld, directed towards specific objectives. Most activities do still give opportunities to the pupils to freely explore certain aspects of Logo, but on the other hand, exploration is restricted by the nature of the activities. This enhances the goal-directedness of the learning experiences, reduces the variety of goals to be taken into account by the teacher at the same time, and helps to reduce technical barriers for the pupils and the teacher.

In the quantitative study a pretest/posttest design was adopted with an experimental and control group. The experimental group consisted of 478 pupils while the control group had 335 children of primary school age. No initial differences in variance (IQ, mathematics ability, etc.) were detected.

For one year an experimental group received a once a week Logo 'lesson'. These Logo activities were based on the Logo resource package and were directed by the regular classroom teacher. A researcher attended the activities in order to gather observational data. The control group received no special training. Since the Logo activities were directed towards the normal classroom objectives, Logo was considered as another 'tool' to sustain teaching and learning.

A special instrument was constructed to 'measure' transfer effects on specific mathematical skills and concepts on a wide variety of topics. This test (WIBAI) was presented to both the control and experimental group in September 1986 and again in June 1987.

Although the general WIBAI scores revealed no significant differences in overall scores, there were significant differences on specific mathematical concepts. The experimental group did significantly better in the areas of:

- estimating and comparing distances,
- restructuring geometric shapes,
- using non-conventional measures, and
- applying scales.

This summary of the Belgium Logo Project only gives a brief overview of the results of the study. Those interested in further details should write to the *LX* European editor, Harry Pinxteren. Moreover, the results still have to be confronted with observational data to put them in context. At the moment, all attention of the project has shifted towards the identification of changes in the problem solving skills of the pupils in the qualitative study. Reports will be published in the near future.

British Logo User Group Conference 1988

This year's conference of the British Logo User Group will be held in the College of Saint Paul and Saint Mary, Cheltenham, from Friday 2 September to Sunday 4 September, 1988.

This conference is intended for primary and secondary school teachers, advisory teachers, teacher trainers, researchers in education and anyone else interested in the use of micromcomputers in schools, in particular in the Logo programming language.

This conference will include a series of activities for Beginners in Logo, and a chance to hear about teachers' experiences in their classrooms. There will be a trade exhibition.

Conference fees:

	Resident	Non-resident	Sat. only
Members	£80	£55	£25
Non-members	£90	£65	£35

For bookings or further details write to:
The British Logo User Group; P.O. Box 79; WALSALL, West Midlands; WS5 3RW

What Your Love of Logo Reveals about You

by Cara Garcia

Are you enthralled with the idea of your students busily engaged at the keyboard, seemingly lost in the Logo microworld? ...debating the finer points of how to best control a floor turtle? ...using Logo as database tool to solve some problem which they've just identified? If the answer to any of these questions is, "Yes," then you are a "LogoLover," a person who appreciates the power and promise of Logo in learning.

Like everything else in life, our passions reveal something about our nature, so, without looking ahead in this article, complete the questionnaire below to find what your love of Logo reveals about you. Again, warning: DO NOT read ahead or you will miss your only chance to get a pure measure of what your love of Logo reveals about you...

Questionnaire for Logo Lovers

Step 1: On the lines below, identify a Logo lesson you've taught and would like to analyze. For example, you might write "How to use a colon for variability."

____

____

Proceed **only** after you've completed step 1.

Step 2: You are to make statements about the three parts of the lessson you've identified: (1) preparing to teach it, (2) actually teaching it, and (3) evaluating it. For each part, there are two statements given below. Circle the statement on the left or right, whichever one best describes the way you would approach that part of the lesson you identified.

(1) Preparing for Instruction

A. I plan the steps leading to the goal. I figure the time it will take for me to cover the material. The parts equal the whole. The goal is measurable and observable behavior. For example, the goal is that students will be able to name and use a variable correctly.

B. I plan how to get the activity started, not knowing how or when the goal will be reached. The whole is more than the sum of the parts. For example, the goal is to discover what it's like to name and use variables in statements.

(2) Teaching the Lesson

A. I state the objective of the lesson for the students. My methods are define, give examples, and practice. For example, I would teach naming and using variables, show examples of these in programs, and then have the students practice with problems I give them. Then I would tell them to use them in their own programs. I go from the knowledge level of Bloom's Taxonomy to the application level.

B. I state the objective of the lesson. My methods are to show one example and then dialogue with the students as they define the function of the new concept. Then we analyze the program listing to determine the steps for writing variables. I would probe to see if they know variables in other situations. I go from the application level of Bloom's Taxonomy to the knowledge level.

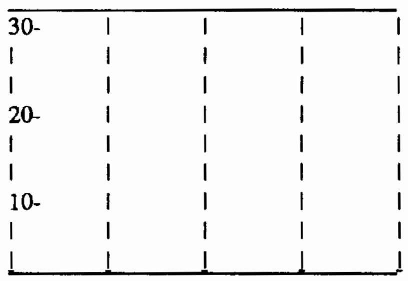
(3) Evaluating the Lesson

A. I evaluate the lesson by testing the students to find out if they know and can use what I have taught. For example, I give them a problem involving variables and grade them on how well they can do it.

B. The students evaluate the lesson by telling me if they understand. I decide if I trust their self-assessment. For example, I ask, "Do you understand variables well enough to move on?"

After marking your choices, go to step 3 below.

Step 3: Now you are to summarize your responses on two bar graphs. Give yourself ten points on the left bar graph for each A statement you circled, and ten points on the right bar graph for the B statements.



Interpreting Your Scores

A score pattern of Left Graph 0 - Right Graph 30 means that your love of Logo is true-blue—you chose a totally cognitivistic approach to each section of the lesson you identified.

What Your Love of Logo Reveals about You- CONTINUED

A score pattern of Left 10 - Right 20 means that your love of Logo is half-hearted (technically 66 2/3% hearted)— for one of the three parts of the lesson you chose a behavioristic practice.

A score pattern of Left 20 - Right 10 means that your love of Logo is fickle—two out of three times you were lured away by the call of behaviorism.

A score pattern of Left 30 - Right 0 means that your love of Logo is flirtatious—all your practices were behavioristic even though you may have said you loved the theory.

Logo: A Cognitivistic Metaphor

Mainly, we tend towards one of two major theoretical poles: behavioral or cognitive. The behavioral approach uses telling, showing, and practice to develop habits. Learning is behavior modification. The cognitive approach uses questioning, hands-on experimentation, and creative projects to solve problems or just experience something. Learning is insight.[1]

Logo is designed to get students thinking about their thinking. It is associated with the work with Jean Piaget, a genetic epistomologist, an influential figure in developmental cognitive psychology. Logo is intended to be an alternative approach to behaviorism; that is, one way to understand Logo is by constant comparison and contrast to behaviorism: in large measure, Logo is what behaviorism isn't. If you examine the statements in the "Logo love" questionnaire, you'll find that they are polar.

Both behaviorism and cognitivism are useful orientations for our teaching practice. But each theory fits best in certain places in the curriculum. For example, on the one hand, we don't want students to discover how to handle floppy disks so we use a teacher-directed behavioral approach to train them directly. On the other hand, we don't want students to learn content area subjects by rote, spouting facts without comprehension, so we use a student-centered approach to facilitate their understanding, appreciation, and skill in using information. We choose the theory that fits the situation.

Yet, it is important for us to recognize that behaviorism and cognitivism are mutually exclusive approaches at any given time. Like it or not, we cannot be both teacher-directed and student-centered simultaneously. Either the teacher is in control or the student is in control. Even in the cooperative learning projects which are becoming popular, the control fluctuates between the teacher and students. In fact, one of the reasons why cooperative learning (Johnson, Johnson, Holubee, Roy, 1984) has been so sucessful is because it helps everyone know who is in control at a given time.

The skillful use of both behaviorism and cognitivism is the mark of a good teacher, but most teachers manage intuitively, with less than a thorough grounding in the theories that underlie their practices. The skilled teacher lets students know when they are expected to follow the teacher's line of thinking and when to proceed independently. Without such clarity, students receive a mixed message... follow me, while proceeding on your own.

The Need for Consciousness-Raising

Many teachers avoid taking a stand on theory, identifying with both behaviorism and cognitivism. When asked how they stand on such issues as use of the computer as a tool or tutor, they respond, "I do whatever works," or "I'm eclectic," or "Well, that's just theory." Some have developed lessons in teaching Logo that are totally behavioral: "Now, class, I'll show you how to draw a box with three right turns." Some of these practices are used by teachers who love Logo and agree that it can empower children to become critical thinkers. Yet these same teachers often believe that first the students must understand "the basics."[2] Unfortunately, the insistance on "basics first" can stifle the creativity and enthusiasm of the student who needs to use Logo to satisfy immediate personal interests.[3] In other words, the cost of such behaviorism is high, purchased at the expense of cognitive and affective growth.

Hence, the LogoLove questionaire is for those who love Logo but need to analyze how that love is or is not being translated into practice. Basically, the questionnaire is a consciousness-raising device. If we teachers are theoretically attracted to cognitivism as embodied in Logo, but are using behavioral practices, I think our first step is to bring our contradiction to light.

Keep Those Cards and Letters Coming In...

I'd appreciate hearing from those who use the questionnaire. If you've been thinking about your thinking, I'd like to know. Also, let me know what your love of Logo reveals about you.

Phone (213) 568-5643 or write me at
Pepperdine University
400 Corporate Pointe
Los Angeles, California 90230

CompuServe 70057,1255.

Notes

1. Behaviorism rests on positivistic traditions in philosophy while cognitivism rests on relativistic traditions. Without belaboring the point, suffice it to say that the assumption in behaviorism is that there is one objective reality which can be learned and in cognitivism, there are multiple subjective realities which are being learned.

2. This reasoning is all too familiar to me. In my graduate work, I majored in reading, and for years I've encountered teachers who insist that decoding must precede comprehension. I've concluded that in such circumstances the students must learn to read in spite of the teacher's theory.

3. For those who need to learn how to teach Logo by using Logo, consider the language experience approach. See Garcia and

What Your Love of Logo Reveals about You- CONTINUED

Polin. A language experience approach to teaching Logo. (Unpublished manuscript. Los Angeles, California: Pepperdine University).

Acknowledgement

Many thanks to Dr. Linda Polin, Assistant Professor of Education in the Graduate School of Education and Psychology at Pepperdine University, who teaches me Logo.

Reference

Johnson, D.W., Johnson, R.T., Holubee, E.J., Roy, P. Circles of Learning: Cooperation in the Classroom. Alexandria, Virginia: Association for Supervision and Curriculum Development, 1984.

—

Cara Garcia, Ph.D. is Professor of Education in the Graduate School of Education and Psychology at Pepperdine University in Los Angeles, California. She teaches educational psychology, curriculum development, and educational computing. Her research deals with the integration of cognitive psychology and gestalt psycholotherapy for cases of learning blocks.

University of Michigan
Ann Arbor, Michigan

Co-Sponsored by
University of Michigan's Professional Development Office and the Michigan Association for Computer Users in Learning (MACUL) SIGLOGO

June 27-30, 1988

June 27
Morning — Introduction to Logo for Novices
Afternoon — Presentations by Michigan-area teachers on using Logo in their classrooms.

June 28 Introduction to LogoWriter (no Logo experience necessary)

June 29 More Logo Writer (on using LogoWriter music, animation, and linking multiple scrapbook pages together in one project — some Logo- Writer experience required)

June 30
Morning — Introduction to Lego TC Logo
Afternoon — The Phantom Fish Tank and Other Logo Ideas

The prices for the sessions are:
Half Day: $20, One Day:$40, Two Days: $70, Three Days:$90, Four Days:$100

For more information, Contact:
Christine Canning; Professional Development Office; 1225 School of Education; The University of Michigan; Ann Arbor, Michigan 48109-1259; 313/763-9497

The Logo Institute
Lesley College
Cambridge, Massachusetts
with Dan Watt and Molly Watt

What are your plans for professional renewal this summer?

Consider The Logo Institute, August 7-21,1988:

- Experience a two-week immersion in a learning environment for Logo-using educators;
- Use the full scope and power of Logo and its educational philosophy to create programming, research, teacher education and curriculum projects; and
- Receive three graduate credits from Lesley College,with three additional credits optional.

Participate with An Outstanding Faculty:

- Dan Watt and Molly Watt, Codirectors;
- Plus: Alison Birch, Sharon Burrowes Yoder, Ricky Carter, Dorothy Fitch, and Michael Tempel.
- Faculty will offer lectures and demonstrations, lead short courses, provide technical assistance and coaching, and support participants in completing their own creative projects.

Attend The Logo Institute Conference, August 20-21, 1988

- An intimate weekend conference to which the broader education community and participants in previous Logo Institutes are invited.
- Keynote speaker: Steve Ocko, codeveloper of Lego-Logo: Tracing the History of Technology Through Clocks, Auto mobiles and Robots.
- Speakers and workshop leaders include Logo Institute faculty and participants and leading Logo-using educators.
- Conference participants may register for one optional graduate credit.

For further information and registration materials, contact: **Prof. Ricky Carter, The Logo Institute, Lesley College, 29 Everett St. , Cambridge, MA 02138-2790; ph. 617/868-9600.**

Join the Leading Professional Organization for Computer Educators!

Whether you are an administrator, curriculum specialist, classroom teacher, media specialist or special educator, you will benefit from membership in ICCE.

ICCE is the leading U.S. and international professional organization for computer educators. It is non-profit, supported by more than 50 organizations of computer-using educators worldwide.

Membership in ICCE includes a subscription to *The Computing Teacher* journal. The journal has long been respected as an important source of information for computer educators, providing accurate, responsible and innovative information. Nine times per year, its articles, departments and reviews keep you in touch with the constant changes taking place in educational technology.

Members will also receive discounts on all ICCE publications and SIG memberships, and information on ICCE committee activities.

As educational technology continues to grow and change, look to ICCE and *The Computing Teacher* journal to keep you on the leading edge.

Join today!

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

INTERNATIONAL COUNCIL FOR COMPUTERS IN EDUCATION
ICCE