12-1988

Subprocedure Anti-APPENDectomies

Judi Harris
College of William and Mary

Logo LinX

Subprocedure Anti-APPENDectomies

by Judi Harris

I once heard of a chemistry professor that immediately erased with his left hand what he wrote on the chalkboard with his right. The idea fascinated me, although I was thankful that I wasn't in his class. Logo work in the immediate mode is a bit like the professor's quick-to-disappear chemical equations. The result may be admired for its precision, beauty, or conceptual power, but the process by which it was formed has disappeared into chalkdust or computer ether.

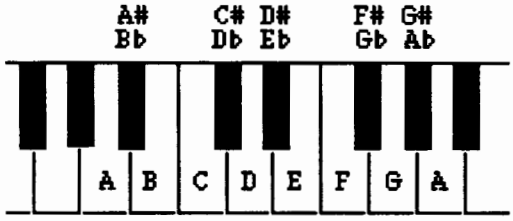
Logo novices may be hesitant to do what you are about to suggest: write procedures that can be stored in memory and invoked with a single command. The leap from immediate mode doodling to preplanned, hierarchically structured procedure writing can be a large one.

It is possible to bridge these two experiences with a simple Logo tool that I call APPEND. When added to all student- or teacher-defined procedures, APPEND constructs a sequential record of subprocedures as they are used in the immediate mode. This list can then be defined as an operable superprocedure with another tool procedure called TITLE.

A Melodious Microworld

Consider, for example, a project that a Pennsylvania fifth- grader conceived and coded. The idea of Logo music intrigued him, especially because he was taking clarinet lessons, and reasoned that if he wrote Logo procedures to play practice pieces for him, then he would be sure to play the pieces correctly on the clarinet.

He proceeded to make eighteen subprocedures, one for each note in the chromatic scale beginning with A below middle C. All shared a common local variable, TIME, used to express the duration of the notes.




```
TO A :TIME
TONE 440 :TIME
END

TO B :TIME
TONE 494 :TIME
END

TO C :TIME
TONE 523 :TIME
END

TO D :TIME
TONE 587 :TIME
END

TO E :TIME
TONE 659 :TIME
END

TO F :TIME
TONE 698 :TIME
END

TO G :TIME
TONE 784 :TIME
END

TO A2 :TIME
TONE 880 :TIME
END
```

```
TO A# :TIME
TONE 466 :TIME
END

TO Bb :TIME
TONE 466 :TIME
END

TO C# :TIME
TONE 554 :TIME
END

TO Db :TIME
TONE 554 :TIME
END

TO D# :TIME
TONE 622 :TIME
END

TO Eb :TIME
TONE 622 :TIME
END

TO F# :TIME
TONE 740 :TIME
END

TO Gb :TIME
TONE 740 :TIME
END

TO G# :TIME
TONE 830 :TIME
END

TO Ab :TIME
TONE 830 :TIME
END
```

(Note: LCSI Logo II and Apple Logo II use the command TOOT instead of TONE, but the frequency numbers are the same.)

The student could then easily combine the subprocedures into superprocedures that played practice songs.

```
TO CHROMATIC.SCALE
A 24
A# 24
B 24
C 24
C# 24
D 24
D# 24
```

```
E 24
F 24
F# 24
G 24
G# 24
A2 24
END

TO TWINKLE
A 12 A 12 G 12 G 12 A2 12 A2 12 G 24
F 12 F 12 E 12 E 12 D 12 D 12 C 24
G 12 G 12 F 12 F 12 E 12 E 12 D 24
G 12 G 12 F 12 F 12 E 12 E 12 D 24
A 12 A 12 G 12 G 12 A2 12 A2 12 G 24
F 12 F 12 E 12 E 12 D 12 D 12 C 24
END
```

But when the student's younger brother stopped by after school, it was clear that he was not ready to write tune procedures like the one above. Instead, he began playing with the note subprocedures in an immediate mode format. His frustration was, of course, that a particular sequence had to be recorded on paper or memorized to be repeated.

Subprocedure APPENDages

An APPEND tool at the end of each of the note procedures would have automatically built a sequenced list of the notes that the younger brother used in his melody experiments.

```
TO APPEND :COMMAND
MAKE "CREATION LPUT
   :COMMAND :CREATION
END

TO A :TIME
TONE 440 :TIME
APPEND "A
APPEND :TIME
END

TO A# :TIME
TONE 466 :TIME
APPEND "A#
APPEND :TIME
END

TO Bb :TIME
TONE 466 :TIME
APPEND "Bb
APPEND :TIME
END
```

In the APPEND tool, CREATION is a global variable name that refers to the record of commands, in list format, that forms sequentially in the computer's memory as the music program user builds a melody. CREATION must be set to [ ] (an empty list) before the musical record of user keystrokes can be started.

```
MAKE "CREATION []
```

Another simple tool procedure, TITLE, takes the current value of CREATION and defines it as a procedure with a title that the user chooses. It then restarts "CREATION, so that the next tone sequence may be recorded separately.

```
TO TITLE
CLEARTEXT
TYPE [TITLE?]
DEFINE READWORD LIST [ ] :CREATION
MAKE "CREATION [ ]
CLEARTEXT
END
```

You're probably thinking, "but what happens if the user types a note, then wants to change it?" As a devoted facilitator, I'd reply, "that's a fascinating programming challenge! How might you go about coding a tool to allow users to alter the string of commands in the computer's memory?"

INSTANT History

One of the most popular examples of action-recording microworlds is INSTANT, a single-keystroke adaptation of Logo graphics commands for new or physically challenged users. Most versions of INSTANT allow users to enter turtle commands with single letters (i.e., F(orward), B(ack), R(ight), and L(eft)). This limits the number of commands that can be created to the number of characters on the keyboard, and further constricts inputs to these commands. Most single-keystroke INSTANTs, for example, make the turtle move 10 steps forward when the user presses an F, and turn the turtle 10 or 90 degrees to the right or left when users type R or L. Angle estimation and length approximation become much less conscious activities when users don't have to type numbers as inputs to movement or heading commands.

If users are physically able to press the Enter or Return key after typing a command, then command options can be greatly increased. Microworld function names can be one or more letters long, can accept variable input, and if APPEND is used, an automatic record of user actions can still be maintained. Moreover, new commands (in the form of short procedures) can be added to the microworlds by either students or teachers without having to change existing code.

APPENDix of Ideas

Although the appropriateness of using INSTANT-like microworlds has been debated (see the articles listed in

Logo LinX -- continued

the References at the end of this column), user-based, self-recording microworlds need not be limited to INSTANT-like programs. Microworlds are best used when they are integrated into subject-centered study. Here is a list of possibilities. Hopefully, these ideas will spark others that can be infused into curricular study in your classroom.

- rebus maker (language arts)
- hieroglyphic typewriter (social studies)
- chemical element diagram-maker (science)
- pattern block microworld (mathematics)
- animation tool (SETSHAPE sequences; art)
- Lego Logo obstacle course path for a car (science)
- sound effect sequences (music)
- braille, Morse code, or sign language translator (language arts)
- choreography tool (music/dance)

If you would like to see examples of how procedures like APPEND and TITLE can be used in subject-centered microworld construction, I would be glad to send you a copy of a pattern blocks/parquetry blocks microworld that I constructed for elementary school use and/or an upgraded version of INSTANT, written for kindergarten students, that I call DRAW. Send a blank, double-sided diskette and a disk mailer with sufficient return postage to the address listed at the end of this article. I would also love to hear from you if you create or revise microworlds using procedures like APPEND and TITLE.

References

Bull, G. L. & Cochran, P. S. (1986). Instant tools. *National Logo Exchange*, 4(4), 8-11.

Lough, T. M. (1983). Slow turtle moves clearly. *National Logo Exchange*, 2(1), 7-8.

Tipps, S. (1983). The issue of instant. *National Logo Exchange*, 2(4), 6-10.

Watt, D. & Watt, M. (1986). Starting Logo with young children? Slow down the turtle! *Logo Exchange*, 5(1), 3-5.

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

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

Creative Writing

Developing Creative Writing Projects with Text Files in Terrapin Logo

by
Mel Levin

Using Logo as a Text Editor

The Logo language is designed to read and save files that contain Logo procedures. By modifying the system, one can use Logo to read and save text files (and thus be able to use the Logo editor purely as a text editor).

The advantage to using the Logo editor as a text editor rather than as a procedure editor is two-fold. First, writing text in this manner uses very little memory in the workspace. This is helpful if you are writing a program which contains a lot of text material (instructions, explanations, written passages, etc.) If you wrote the text as procedures, you would be using primitives which take up lots of memory in the Logo workspace. If your program is large, there is danger that you might encounter the dreaded "no storage left" message. Using the Logo editor as a text editor helps eliminate the problem.

Secondly, you are using the Logo editor as a quasi-word processor, which means that you write text as you would in English and not as Logo code. This means there is no code syntax problem (no hassles with PRINT, SENTENCE, WORD, [,], etc.). You also have the freedom to format the text as you would with a word processor. This is not to imply that the Logo editor has terrific word processing capabilities. It is, in fact, quite limited as a word processor. The point, however, is that text can be formatted relatively easily within the restrictions of the Logo environment.

Normally, to enter the editor you type TO followed by the name of a procedure to be created or edited. To work with just text in Logo, it is necessary for the edit buffer in memory to be empty. Entering the edit buffer is achieved by typing TO followed by Return. At this point text is written and edited the same way as a procedure is written and edited. You use the same cursor keys to move around the editor. In order to save the text, it is necessary to exit the editor. Instead of typing Control-C, which defines a procedure, Control-G is used. This exits the editor without modifying what is in the edit buffer. Thus, the text will be printed as written, and not reformatted as would a procedure. Next, you must save the buffer as a file on the disk. At this point, you must not edit any procedures or enter the graphics mode before you save the buffer or it will be lost. While you can save the text with any name, you cannot save in the usual way.