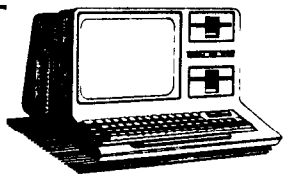
TRENDS IN
HARDWARE/SOFTWARE



Charles E. Crume
Cleborne D. Maddux

LogoWriter: An Introduction and Critique

Logo is a relatively new computer language that has recently become popular in education. Although Logo has gained a reputation as an excellent graphics language for children, many educators and programmers do not realize that it is also a powerful general purpose programming language that includes advanced features such as subprocedures, list processing, and recursion.

WHAT IS LOGO?

Logo was developed by Seymour Papert, an MIT mathematician and computer scientist who spent five years in Geneva studying developmental child psychology with Jean Piaget. When Papert returned to MIT, he put together "the Logo group" and developed a

CHARLES E. CRUME is a Software Systems Analyst and Technical Consultant, University of Nevada System Computing Services, Reno, NV 89557-0023.
CLEBORNE D. MADDUX is Professor of Education and Chairman, Department of Curriculum and Instruction, University of Nevada, Reno, NV 89557-0023.

Computers in the Schools, Vol. 7(3) 1990

new computer language based on Piagetian developmental psychology (Papert, 1980). The language was Logo.

Papert developed Logo by modifying LISP, an existing computer language commonly used by researchers in artificial intelligence experiments. It is ironic that Logo is sometimes regarded merely as a children's graphics language, since neither LISP nor the original version of Logo included any graphics capability.

THE LOGO CONTROVERSY

Recently, Logo has sparked a controversy among educational researchers. This controversy is based on claims that have been made about benefits to be gained from teaching the language to children. Some of these claims have been made by Papert himself. The most notable is his contention that Logo is capable of lowering the boundary between child and adult thinking. Although this revolutionary claim has often been attacked, Papert's critics sometimes overlook the fact that his hypothesis called for specialized teaching methods coupled with implementation of sweeping educational changes in our schools. Because these changes (suspension of evaluation, a computer for every child, etc.) have not occurred and are not likely to occur, Papert's hypothesis may never be fairly tested outside of a few artificial laboratory settings (Maddux & Johnson, 1988).

Other controversial claims have been made by a heterogeneous group of Logo zealots. These claims consist of diverse contentions about Logo's ability to effect changes in children's cognitive processes. Such advocates have been accused of promoting Logo as *Wheaties of the mind*. For an in-depth discussion of these and other issues surrounding Logo, the interested reader is referred to Maddux and Johnson (1988).

WHY TEACH LOGO?

While the controversy over Logo continues, we believe that Logo is an excellent language for introducing children to computers and programming. The learning of languages such as BASIC, FORTRAN, and Pascal require lengthy study and practice before

enough is learned to permit doing anything interesting with the language. (In fact, it could be argued that children at most levels of mastery are seldom asked or encouraged to do interesting and creative things with such languages.) This is true because most computer languages were designed to be useful for the solving of repetitive, routine tasks on cumbersome mainframe computers. These earlier languages were designed to run in batch mode and employed arcane syntax rules and obscure, abstract, mathematically oriented statement structure. Logo, on the other hand, was designed to be attractive and highly motivating to beginners. Logo's graphics commands, for example, allow children to do interesting things after only a brief introduction to the language. At the same time, Logo's list processing commands, subprocedures, and recursive capability allow the writing of sophisticated programs in a simple but elegant manner. This flexibility has prompted the observation that Logo has "no threshold and no ceiling" (Billstein, Libeskind, & Lott, 1985, p. 1).

Due to its simplicity, Logo eliminates many barriers to mastery and seems to motivate many children to independent exploration of Logo's capabilities. At the same time, Logo's power makes it a candidate for inclusion in the junior and senior high school computer curriculum.

PURPOSE OF THE ARTICLE

The first commercial implementation of Logo was in the late seventies for the ill-fated Texas Instruments TI99/4A microcomputer. As Logo's popularity among educators has grown, numerous dialects of the language have been developed for a variety of hardware. A new dialect that adds word processing to the language is marketed by LCSI (Logo Computer Systems, Inc.) and is called *LogoWriter.* According to Papert (1986), LogoWriter is a new medium that will aid children in learning the "high-order" skills of writing. The purpose of this article is to provide an introduction to *LogoWriter* and an analysis of its strengths and weaknesses.

LOGOWRITER

LogoWriter version 1.0 was released in April of 1986. Designed to run on the Apple II, IBM PC, IBM PC Junior, and Commodore 64 microcomputers, it was priced at $395 with a yearly renewal fee of $99. For this price, schools received a master disk for each of the above microcomputers and an activity kit. Version 1.0 was limited by several characteristics including the inability to leave a page without saving it and (worse yet) an extremely limited work space.

LogoWriter version 1.1 was released in April of 1987. Version 1.1 rectified a number of bugs present in version 1.0. More importantly, LCSI had realized that many school officials objected to the $99 annual renewal fee, and the company reacted by substituting a one-time fee of $450. For this price, schools received four master disks, one for each brand of microcomputer listed above. This astute marketing decision contributed to LogoWriter's present popularity.

LogoWriter version 2.0 was released in September 1988. This version was designed to run on the Apple II and IBM PC microcomputers. (The IBM PC Junior and Commodore 64 versions have not been revised, and remain at level 1.1.) LCSI amended the one-time fee of $450 to include a single master disk. (Buyers were required to select either IBM or Apple II.) Version 2.0 increased the work space from 64k to 128k and added network capabilities. Version 2.0 also added many new primitives. Among these are GETSHAPES, LEAVEPAGE, LOCK, and UNLOCK for scrapbook operations; INT, ROUND, and SQRT for mathematical operations; and COPYFILE, ERASEFILE, FILELIST, LOADPIC, LOADTEXT, SAVEPIC, and SAVETEXT for file operations. New features specific to the Apple II version of LogoWriter include ProDOS compatibility, multiple directories, a shapes page per subdirectory, and color printing capability. New features specific to the IBM version include the ability to customize Logo for use with the 16 color enhanced graphics adaptor (EGA) and the 256 color multi-color graphics array (MCGA) and the ability to employ up to 90 shapes per shapes page. The price to upgrade from version 1.1 to version 2.0 is $35 for the Apple II and $50 for the IBM PC.

The following discussion focuses on LogoWriter version 2.0 for

the Apple II and IBM PC microcomputers. It is divided into three areas: (a) strengths of LogoWriter, (b) weaknesses of LogoWriter, and (c) suggested enhancements that might make LogoWriter more useful. These three areas will be addressed in turn.

STRENGTHS OF LOGOWRITER

1. *The ability to easily combine text and graphics is LogoWriter's most highly touted feature (Papert, 1986).* The commands provided for text manipulation are easy to understand and use. The most frequently used commands such as MARK, CUT, COPY, and PASTE are easily accessed through use of the function keys.

2. *The ability to control multiple turtles concurrently.* LogoWriter provides four turtles that facilitate complex drawings and animation. Their use could also provide practice in skills related to identity (determining which turtle needs to be moved) and sequencing (determining the order that the turtles should be moved to accomplish some task).

3. *The commands ASK, EACH, TELL, and WHO.* The ability to control multiple turtles presents the problem of achieving collective control in some cases, and independent control in other cases. The TELL primitive is used to determine which turtles will respond to succeeding commands. The ASK primitive is used to temporarily override the TELL primitive. The WHO primitive reports which turtles are currently set to respond to commands. The EACH primitive is used to make each turtle respond to a list of commands in turn. These four primitives greatly facilitate complex programming tasks. For example, the following procedure instructs all four turtles to draw a square concurrently, then instructs each turtle to draw a circle in turn:

```
TO DEMO
  TELL [0 1 2 3]
  ST
  REPEAT 4 [FD 5 RT 90]
  EACH [REPEAT 90 [FD 2 RT 4]]
END
```

4. *The ability to create "tool" procedures.* Tool procedures occupy computer memory but are not routinely accessible for editing (they can only be executed). This prevents the inadvertent alteration of their instructions. Additionally, the computer uses its memory more efficiently, thereby allowing larger programs to be written. Procedures frequently needed for a variety of projects are often loaded as tool procedures. We have developed a variety of tools, such as the TEST, IFTRUE, and IFFALSE procedures referred to in point 4 of the weaknesses section.

5. *Automatic loading and execution of a LogoWriter program.* If a page named STARTUP is on the disk when LogoWriter begins, LogoWriter will load that page into the computer's memory automatically. If a page contains a procedure named STARTUP, LogoWriter will execute the STARTUP procedure whenever that page is loaded. To cause the automatic execution of a program whenever LogoWriter is started, the user can place a STARTUP procedure within a page named STARTUP on the disk.

6. *The ability to "stamp" shapes on the page.* Shapes facilitate the design of small complex figures and eliminate the need to actually include turtle commands to draw figures such as a forest, a flock of birds, a fleet of cars, and oversized letters and numerals. Oversized letters and numerals are handy for use with younger children.

7. *Use of an actual turtle shape to represent the Logo turtle.* Many other dialects use a triangle to represent the turtle. This needless abstraction could be confusing to some younger children. If users find the literal turtle shape objectionable, they can, of course, easily choose one of 30 predefined shapes or make up a shape of their own design.

8. *Availability of a site license.* Instead of charging for each copy of LogoWriter, LCSI charges a one-time fee of $450. A school is allowed to make copies of the software (but not printed materials) for each computer they own. When more computers are acquired, additional copies of the software can be made without additional fees.

9. *Availability of a home use agreement option.* In addition to the site license, a home use agreement can be purchased for an addi-

tional one-time fee of $150. This permits children to take a copy of LogoWriter home and use it on their own computers.

WEAKNESSES OF LOGOWRITER

1. *LogoWriter allows more than one procedure to have the same name*. However, only the procedure closest to the end of the page is used. This can cause considerable confusion, since editing the first procedure will have no effect on the way the second procedure with that same name works. (Logo searches from the bottom up.)

2. *The four turtles are numbered 0 through 3*. Ask anyone what number they start counting with and the answer will most likely be *one* rather than *zero*. Even though memory locations in the computer start at zero, and it is thus understandable that programmers would choose to begin with zero, it is difficult for most people (especially children) to make this adjustment. We agree with Kemeny and Kurtz (1985) who included the following among their design criteria for BASIC: (a) advanced features should be added in such a way that any inconvenience or difficulty affects the expert rather than the novice, (b) no hardware expertise should be necessary, and (c) the user should be shielded from the operating system. We recommend that LCSI modify LogoWriter to number the turtles from 1 to 4.

3. *LogoWriter lacks the LOCAL primitive*. This complicates the writing of programs and tool procedures. For example, if a user creates a variable and inadvertently assigns it a name used by an existing procedure, the results are unpredictable. Possible scenarios follow: (a) The program might stop with an error message; (b) the program might continue to run properly (if the variable is not subsequently accessed); (c) the program might continue to run properly but terminate at a later time when the variable is subsequently accessed; or (d) the program might continue to run, but do so incorrectly. Scenarios (c) and (d) cause problems that are obscure and difficult to diagnose. The technical support staff at LCSI advises, via telephone, that the LOCAL primitive was removed from LogoWriter because educators were "abusing" it. The staff suggested that the abuse consisted of using local variables in procedures in order to save main memory. We do not agree that such a practice

constitutes abuse, and we feel the LOCAL primitive should be reinstituted in LogoWriter.

4. *The syntax and operation of the IF and IFELSE commands are difficult for beginners to understand.* Both children and adults sometimes have difficulty with these statements. Some other dialects of Logo provide a TEST primitive (to test a condition), an IFTRUE primitive (to execute one or more commands if the TEST was true), and an IFFALSE primitive (to execute one or more commands if the condition was false). We feel these primitives are less abstract, and are therefore better choices than IF and IFELSE. Therefore, we have written TEST, IFTRUE, and IFFALSE primitives for LogoWriter (Crume & Maddux, 1989). The source code for the primitives is duplicated here:

```
TO TEST :MYLIST
  IF NOT LIST? :MYLIST [(TYPE [TEST
    DOESN'T LIKE] :MYLIST [AS INPUT]
    CHAR 13) STOP]
  IFELSE RUN :MYLIST
    [MAKE "TEST.RESULT "TRUE]
    [MAKE "TEST.RESULT "FALSE]
END

TO IFTRUE :LIST.OF.COMMANDS
  IF NOT LIST? :LIST.OF.COMMANDS
    [(TYPE [IFTRUE DOESN'T LIKE]
    :LIST.OF.COMMANDS
    [AS INPUT] CHAR 13) STOP]
  IF NAME? "TEST.RESULT
    [IF :TEST.RESULT = "TRUE
    [RUN :LIST.OF.COMMANDS]]
END

TO IFFALSE :LIST.OF.COMMANDS
  IF NOT LIST? :LIST.OF.COMMANDS
    [(TYPE [IFFALSE DOESN'T LIKE]
    :LIST.OF.COMMANDS
    [AS INPUT] CHAR 13) STOP]
  IF NAME? "TEST.RESULT
```

```
    [IF :TEST.RESULT = "FALSE
    [RUN :LIST.OF.COMMANDS]]
END
```

These procedures do not work exactly like the real primitives of other Logo dialects. First, the TEST primitive requires a list as input. This allows the input to be error trapped and also reminds us that a user defined version of TEST is being used. Second, TEST stores its result in a variable called TEST.RESULT. IFTRUE and IFFALSE use the contents of TEST.RESULT to determine whether they should execute the commands in their input. Using a variable limits the capability to nest TEST primitives—but there are ways to circumvent this limitation. For example, the procedure WINTER (containing a TEST command) is shown below:

```
TO WINTER
  TEST [:WEATHER = "SNOWING]
  IFTRUE [PRINT [IT IS SNOWING]]
  IFFALSE [PRINT [IT IS NOT SNOWING]]
END
```

This procedure executes without any problems. However, if the following commands (which use winter) are executed, a "bug" is encountered:

```
MAKE "MONTH "DECEMBER
MAKE "WEATHER "RAINING
TEST [OR :MONTH = "NOVEMBER OR :MONTH =
"DECEMBER
  :MONTH = "JANUARY]
IFTRUE [WINTER]
IFFALSE [PRINT [IT MUST NOT BE WINTER]]
```

The problem is both the IFTRUE and IFFALSE commands will be executed when only the IFTRUE is appropriate. IFTRUE is properly executed because the test for the month is true. IFFALSE is erroneously executed because the TEST command in WINTER changes the value of TEST.RESULT to false. This problem can be avoided by: (a) Not nesting TEST, IFTRUE and IFFALSE com-

mands; or (b) inserting a duplicate TEST command in front of the IFFALSE command as shown below:

```
MAKE "MONTH "DECEMBER
MAKE "WEATHER "RAINING
TEST [OR :MONTH = "NOVEMBER OR :MONTH =
"DECEMBER
  :MONTH = "JANUARY]
IFTRUE [WINTER]
TEST [OR :MONTH = "NOVEMBER OR :MONTH =
"DECEMBER
  :MONTH = "JANUARY]
IFFALSE [PRINT [IT MUST NOT BE WINTER]]
```

Both solutions are inconvenient. However, there are many situations in which this nesting limitation will not be a problem and TEST, IFTRUE, and IFFALSE primitives are convenient for those who prefer them over IF and IFELSE.

5. *LogoWriter does not include a "FENCE" primitive.* The FENCE primitive in other versions of Logo prevents the turtle from leaving the screen. Young children might benefit from such a primitive as they sometimes do not understand what is happening when the turtle leaves one side of the screen and appears on the other side.

6. *LogoWriter does not include a "WINDOW" primitive.* Whereas the FENCE primitive prevents the turtle from leaving the screen, WINDOW allows it to leave without appearing on the other side. This primitive (available in many other dialects of Logo) is useful when graphing mathematical functions in which the value of the function exceeds the scaling factor of the window. In LogoWriter, the lack of a WINDOW primitive causes the tip of the graph to wrap around and appear on the other side of the screen, creating a cluttered display. To prevent this, extra programming must be done to account for such function values.

7. *The turtle is the only shape that pivots on its axis when its heading is changed.* Employing this feature in conjunction with the command SLOWTURTLE (which slows the turtle's movement) allows teachers to explain and demonstrate the turtle's response to

various commands. The following procedure, which draws a square, demonstrates this:

```
TO DEMO
  SLOWTURTLE
  REPEAT 4 [FD 50 RT 90]
END
```

Unfortunately, the other shapes cannot pivot, even if their headings are changed. This inconsistency can be confusing to beginners who change the shape of the turtle.

8. *There is no way to easily execute operating system commands from within LogoWriter.* The ability to perform certain tasks such as formatting floppy disks from within the LogoWriter program would be beneficial for many users.

9. *The clipboard reports as a single word whatever is placed into it.* This makes it difficult to analyze the clipboard's contents. For example, if the text "Hello there, John. How are you today?" is placed in the clipboard, and the command PRINT FIRST CLIPBOARD is issued, the result is the letter "H" rather than the word "Hello." The primitive PARSE will convert the words into a list (according to LogoWriter's internal rules), but this is too advanced for most beginning users.

10. *LogoWriter gave up the ability to type TO followed by the name of a procedure to enter procedure writing mode. Also eliminated was the ability to type EDIT followed by the name of a procedure to enter procedure editing mode.* Other versions of Logo provide these two capabilities. In LogoWriter, one must go to the "flip" side of the page to enter a procedure. (The flip side is accessed by holding down a control key and striking the F key. This procedure is intended to be analogous to turning to the *flip* side of a piece of paper. This may be an advantage for first-time users of Logo but is confusing to users who have previous experience with other dialects of the language.) To edit in LogoWriter, one must go to the flip side and position the cursor to the line to be edited. Finding the correct line in the desired procedure can be difficult and time consuming.

We have written the procedures EDIT, EDIT!, and BLANKS.

These procedures search the flip side for the named procedure and either position its first line at the top of the screen (if found) or issue an error message (if not found). Then, to edit the procedure, the user simply goes to the flip side. The procedures are shown below:

```
TO EDIT :PROCEDURE
  IF FRONT? [FLIP]
  TOP EDIT! :PROCEDURE 1
  CU SEARCH :PROCEDURE UNSELECT SOL
  REPEAT 18 [CD] REPEAT 18 [CU]
  FLIP
END

TO EDIT! :X :C
  SEARCH WORD "TO WORD BLANKS :C :X
  IF FOUND? [STOP]
  IF :C > 10 [(SHOW :PROCEDURE :X "NOT "FOUND)
  FLIP STOPALL]
  EDIT! :X :C + 1
END

TO BLANKS :C
  IFELSE :C = 1 [OP CHAR 32]
       [OP WORD CHAR 32 BLANKS :C - 1]
END
```

11. *Flexible disk management commands are not included.* The ability to read and write directly to a disk file is an important feature lacking in LogoWriter. Although one can read and write an entire file via LOADTEXT and SAVETEXT, these do not allow processing information on a record-by-record basis. Excellent disk management commands can be found in Logo II, another LCSI dialect that is, so far, not available for machines other than the Apple II.

12. *LogoWriter does not include the DRIBBLE primitive.* The dribble primitive allows setting up dribble files to record all keystrokes. This primitive is handy for teaching and for research. DRIBBLE is included in the Logo II version referred to above.

13. *LogoWriter does not provide a way to switch between color palettes on an IBM PC equipped with a CGA (color graphics adaptor).* The CGA provides both a white/cyan/magenta and a red/yel-

low/green palette. The ability to switch between the two would enhance graphics applications for CGA-based computers.

14. *LogoWriter lack of a nice selection of tool procedures.* Tool procedures add a great deal of power and flexibility to LogoWriter. However, most educators do not have the time to develop their own tools. Perhaps LCSI could set up a "toolbox" into which educators could deposit tool procedures others might deem beneficial, and locate tools which they might be able to employ in their own curriculum.

15. *LogoWriter's documentation is poor.* Although we have seen much worse documentation, LogoWriter's reference manual is not oriented toward the beginning user or the teacher with no experience with Logo. Although the purchase price covers an attractive kit with impressive-looking activity cards, there is no introduction to the language, and the manual is confusing and incomplete.

ENHANCEMENTS TO LOGOWRITER

1. *The limit of three foreground colors and a background color on CGA-equipped computers limits various applications.* We propose a low/medium resolution graphics mode that would allow the use of all supported colors for text applications.

2. *The inclusion of sprites (independently executing processes) would provide for sophisticated animation.* Although one can address up to four turtles concurrently, independent execution of commands by each turtle is not allowed. Although the IBM PC does not have a sprite controller chip (as did the Texas Instruments TI99/4A), it should be possible to insert procedures into the "process control stack" and allow the PC's standard clock (which ticks 18.2 times a second) to execute such code at a rate acceptable for numerous animation applications.

CONCLUSIONS

Logo, based upon developmental Piagetian child psychology, is the first (and only) computer language developed specifically for children. It is easy to learn and use, yet provides a powerful learning tool for children. Although considered a "toy" computer lan-

guage by the uninformed, Logo is powerful and flexible enough for many sophisticated programming projects. As these facts become more widely known, Logo (and LogoWriter) may surpass other, more traditional computer languages as the choice in most curricula, not only for teaching beginning programming, but also for use as a powerful general purpose programming language.

REFERENCES

Billstein, R., Libeskind, S., & Lott, J. (1985). *Logo: MIT Logo for the Apple*. Menlo Park, CA: Benjamin/Cummings Publishing Co.

Crume, C. E., & Maddux, C. D. (1989). Adding a set of alternative conditional primitives to LogoWriter. *Logo Exchange*, *8*(1), 26-27.

Kemeny, J. G., & Kurtz, T. E. (1985). *Back to BASIC: The history, corruption, and future of the language*. Addison-Wesley.

Logo Computer Systems, Inc. (1988). *LogoWriter reference guide*. Vaudreuil, Quebec: Author.

Maddux, C. D., & Johnson, D. L. (1988). *Logo: Methods and curriculum for teachers*. New York: The Haworth Press.

Papert, S. (1980). *Mind-Storms: Children, computers, and powerful ideas*. New York: Basic Books.

Papert, S. (1986). The next step: LogoWriter. *Classroom Computer Learning*, *6*(7), 38-40.