BY G. MICHAEL VOSE

MACINTOSH PASCAL

An interactive interpreter transforms Pascal into a language as easy to learn as it is expeditious to use

Pascal's evolution has mirrored the growth of the microcomputer industry—both seek to bring usable, learnable computer power to a generation of inquisitive, educated people looking toward the next century. Niklaus Wirth created Pascal to make learning computer programming an easy but still rigorous task. Even before Carl Helmers called six years ago in this journal for the widespread adoption of Pascal, colleges and universities worldwide were beginning to embrace the language as a primary tool for teaching programming.

Apple Pascal was released in 1979 and was one of the first microcomputer implementations. Pascal became the primary programming language within Apple Computer Inc. for the development of new products. With this strong tie to Pascal, there was a good chance that Apple would be instrumental in the adoption of significant new Pascal products. The first of these new products is the recently announced version for the Macintosh.

The version of Pascal that Apple Computer offers for its new Macintosh is called Macintosh Pascal. Although it will be marketed by Apple, Macintosh Pascal was created at Think Technologies Inc. (420 Bedford St., Lexington, MA 02173) by Melvin Conway, who conceived the project and wrote the prototype interpreter; Andrew Singer and John Hueras, who designed the product for the Macintosh; and Peter Maruhnic and Terry Lucas, who wrote the Macintosh version. Running initially on the Macintosh only, Macintosh Pascal will be available for Apple's Lisa running under the MacWorks operating system. Think Technologies promises separate versions of the language for all major educational microcomputers in the next 18 months. Macintosh Pascal will retail for $125.

Editor's Note: This article is a BYTE *Product Preview. It is not a review. We think this new product is significant and therefore offer this advance look at a prerelease version. An independent in-depth review, with appropriate benchmarks, will appear in a subsequent issue.*

A New Breed

An interactive interpreter is the most innovative feature of Macintosh Pascal. Programmers can write source code in Macintosh Pascal and run it immediately without going through a separate compilation step. Students can run individual commands to understand their functions. Using the Macintosh user interface—with its multiple windows, mouse, and data integration—makes Macintosh Pascal programming easy and efficient. New programmers can learn the language more quickly and effectively when they can interact with a program at the source-code level. Macintosh Pascal's program-development tools, including single-step execution, use of breakpoints, and an Observe window to track the alteration of variables, further enhance this process (see "Macintosh Pascal's Development Tools" later in the text).

Macintosh Pascal is a full implementation, not a subset, of Pascal, and it emulates as closely as possible both the ANSI (American National Standards Institute) standard Pascal and LisaPascal. The following paragraphs describe the major differences between Macintosh Pascal and LisaPascal and Macintosh Pascal and ANSI Pascal.

Macintosh Pascal varies slightly from LisaPascal, particularly in the way the latter uses extensions to the language definition. Also, the scope anomalies of LisaPascal are errors in Macintosh Pascal. The Macintosh version differs in other significant ways, including:

- use of up to 255 significant identifier characters
- no support of compiler commands or nested comments
- simpler rules for *integer* and *longint* arithmetic
- additional *real* data types: *longreal*, *extended*, and *computational*
- requirement of the *otherwise* statement within a CASE construct
- no support of the *external* directive
- no support of user-defined units or segmentation
- no support of the functions

exit	*halt*	*heapresult*
mark	*release*	*memavail*
pwroften	*moveleft*	*moveright*
scaneq	*scanne*	*fillchar*

- support of the *pack* and *unpack* procedures

The Macintosh Pascal manual lists other minor differences between the two.

Macintosh Pascal conforms most closely to the ANSI standard for Pascal and is closer to that standard than is LisaPascal. Macintosh Pascal's major departures from the ANSI/IEEE 770X3.97-1983 standard include:

- the special symbol @ is an operator and never treated as a ^
- only the standard file variables INPUT and OUTPUT can be used as program parameters
- all quoted character strings are STRING data types, but Macintosh Pascal's compatibility rules are nonetheless compatible with the standard's
- support of the word symbols *otherwise*, *string*, and *uses*
- support of the underscore character within an identifier
- all *integer* and *real* data type operands are converted to *extended* before real arithmetic is performed; the result is always *extended*
- support of predefined *libraries*
- support of a set of string procedures and functions
- support of the *pointer* and *sizeof* functions for LisaPascal compatibility

The Macintosh Pascal manual lists other minor differences from the ANSI standard, including errors not automatically detected and reported, in an appendix.

Macintosh Pascal also supports the graphics functions of the Macintosh QuickDraw program. Macintosh Pascal can take advantage of QuickDraw's functions by including the QuickDraw libraries. This is done with the *uses* clause; for example, *uses* QUICKDRAW1, QUICKDRAW2.

Macintosh Pascal also supports IEEE numerics conventions using the Pascal library SANE (Standard Apple Numeric Environment). The SANE package is the first implementation of IEEE numerics on a microcomputer.

Programming in Macintosh Pascal

Because the language is interpreted, programming in Macintosh Pascal is very similar to using interpreted BASIC.

(*text continued on page 138*)

G. Michael Vose is a BYTE *senior technical editor. He can be contacted at* POB 372, *Hancock*, NH 03449.

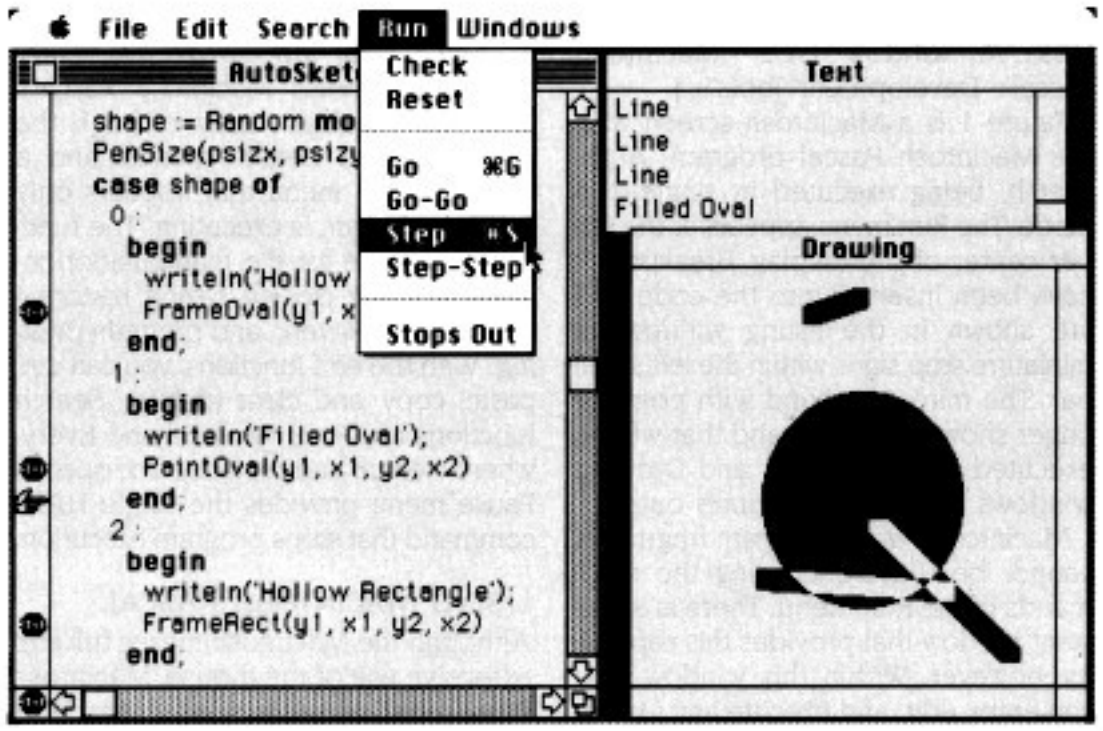

Figure 1: *The Macintosh Pascal AutoSketch program. The Run menu appears in the upper center of the screen. At the right are the Text and Drawing windows. The listing window, on the left, shows breakpoints indicated by stop signs; the finger points to the next instruction to be executed in single-step mode.*

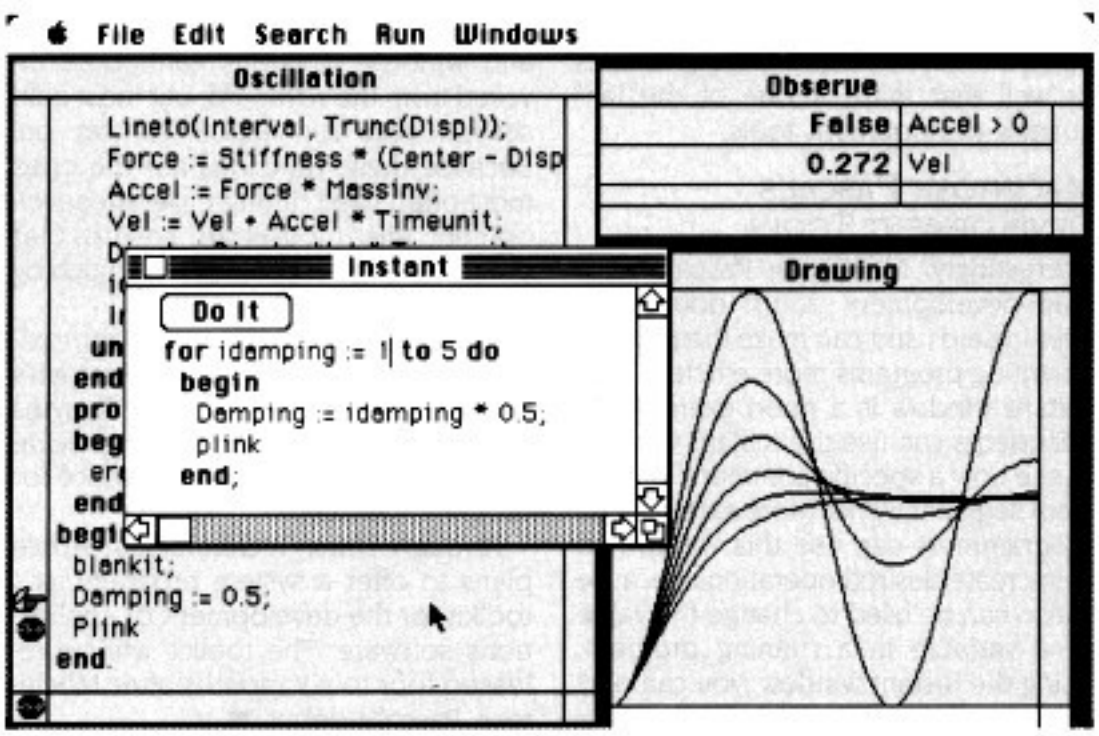

Figure 2: *The Oscillation program. The Observe window in the upper right shows the value of variables or expressions. The Instant window enables execution of code fragments and the changing of variables during program execution.*

(text continued from page 137)

You type in or load from disk the code you plan to run and then run it. Because Macintosh Pascal program lines are precompiled with the entry of a carriage return, errors are detected and reported immediately. Macintosh Pascal is thus even friendlier than traditional interpreted BASIC in detecting errors.

Where BASIC uses the RUN command to start program execution, Macintosh Pascal uses GO or ⌘-G, Apple's Cloverleaf command key followed by G in the manner of the Control-X keystroke sequence. Macintosh Pascal also enables execution of a program with breakpoints (called Stops) placed within the code (GO-GO), or single-step execution of the code with (STEP-STEP) or without (STEP or Cloverleaf-S) breakpoints. The GO-GO and STEP-STEP commands run a program with breakpoints, pause briefly at each Stop, and then continue, updating variables or expressions in the Observe window (see "Macintosh Pascal's Development Tools").

Figure 1 is a Macintosh screen with the Macintosh Pascal program, Auto-Sketch, being executed in single-step mode. The Run menu appears at the upper center of the display. Breakpoints have been inserted into the code and are shown in the listing window as miniature stop signs within the left scroll bar. The miniature hand with pointing finger shows the command that will be executed next. The Text and Drawing windows show the program's output.

Macintosh Pascal program fragments cannot be run alone using the commands in the Run menu. There is an Instant window that provides this capability, however. Within this window, you can enter, edit, and execute any Macintosh Pascal statement. The Instant window has great potential as an educational aid but has additional capabilities as well that make it one of the language's development tools.

Macintosh Pascal's Development Tools

Interestingly, Macintosh Pascal's program-development tools double as learning aids and can make the process of writing programs more efficient. The Instant window is a good example.

Students can use the Instant window to see how a specific command or program segment works. More experienced programmers can use this window to help create desired operations because it also can be used to change the value of a variable in a running program. Using the Instant window, you can play "what if" games with variables in a program while it is running.

This intraprogram interactivity is the guiding philosophy behind the language's program-development tools. Besides the Instant window, you can use an Observe window to watch the value of variables and expressions change as a program executes; the Text and Drawing windows to see the text and graphics output, respectively, of the current running program; or the Clipboard window, which provides access to the Clipboard system utility, used to move text or graphics from one window or program to any other program or window.

Figure 2 shows a Macintosh Pascal program called Oscillation in a display that includes the Instant and Observe windows. The Observe window, in the upper right corner, shows that the value of the *Accel*>0 expression is false, while the value of the variable *Vel* is 0.272. The Instant window enables the execution of a single *for* loop, with its result shown in the Drawing window.

You can access Macintosh Pascal's other development tools through the File, Edit, and Search menus, and a special Pause menu that appears only while a program is executing. The functions available for file manipulation include opening, closing, saving, restoring after editing (Revert), and program printing. With the edit functions, you can cut, paste, copy, and clear (delete). Search functions are Find, Replace, and Everywhere (search and replace). The special Pause menu provides the single HALT command that stops program execution.

Using Macintosh Pascal

Although the Macintosh makes full and extensive use of the mouse, Macintosh Pascal enables you to select many of its functions from the keyboard by using the Cloverleaf key as a control key. File and window functions cannot be invoked from the keyboard, but most edit, search, and run functions can be. Because these functions are the ones most often used during program development, this "mousetrap" ensures that programmers are not hindered much by the ubiquitous rodent.

Macintosh Pascal consumes approximately 50K bytes of the Macintosh's memory, leaving more than 35K bytes for programs. Program disks provide approximately 100K bytes of space for program storage.

Through Think Technologies, Apple plans to offer a system programmer's toolkit for the development of applications software. The toolkit will be released four to six months after Macintosh Pascal's debut. ■