A GUIDE TO GUIs

Graphical user interfaces make computers easy to use; keeping them all straight is the hard part

Frank Hayes and Nick Baran

The world of graphical user interfaces (GUIs) seemed pretty simple in 1984, when Apple introduced the Macintosh. Back then, the genealogy was straightforward: Researchers at Xerox's Palo Alto Research Center begat the Xerox Star; Steve Jobs visited PARC, saw the Star, went back to Apple, and begat the Mac.

But five years later, the begats have become bewildering. The Mac begat Windows—or was it just a cousin? Windows begat Presentation Manager—which doesn't look much like the Mac at all, thanks to IBM, which begat Systems Application Architecture (SAA). MIT begat X Window, which crossbred with PM and NewWave to give birth to Motif. Tandy begat DeskMate, Japan, Inc., begat BTRON, Steve Jobs—back again for a second try—begat NextStep, and Apple has filed a paternity suit against Microsoft. What a mess.

But though there seem to be dozens of GUIs today, it's clear that they all still share similarities that reach below the surface.

Just One of the GUIs

Turn on a Macintosh, and you'll come face to face with the original definition of a GUI for desktop computers. The Mac defined the parts we've come to associate with a GUI:

- a pointing device, typically a mouse
- on-screen menus that can appear or disappear under pointing-device control
- windows that graphically display what the computer is doing
- icons that represent files, directories, and so on
- dialog boxes, buttons, sliders, check boxes, and a plethora of other graphical widgets that let you tell the computer what to do and how to do it

Of course, today's GUIs come in many varieties—not everything that's called a GUI has all these features. For example, some GUIs don't use icons. On others, the icons are optional or available only sometimes. Some require a mouse, while others will let you work from the keyboard.

GUIs are more similar beneath the surface. Although there are some hybrids, most GUIs consist of three major components: a windowing system, an imaging model, and an application program interface (API): (See figure 1.)

The windowing system is a set of programming tools and commands for building the windows, menus, and dialog boxes that appear on the screen. It controls how windows are created, sized, and moved on-screen, and how the user moves from one window to another, among other functions.

One example of a windowing system is X Window. X Window is *not* a complete GUI—it's just the windowing system shared by a group of different GUIs. Because all the X Window GUIs share the same windowing system, they can also share programming tools for developing applications. (By contrast, Microsoft Windows, for example, is a complete GUI with its own windowing system, imaging model, and API.)

The imaging model defines how fonts and graphics are actually created on-screen. For example, the typeface and size of text in a word processor or desktop publishing program is specified through the imaging model; so are the lines and curves of a CAD program. PostScript may be the best-known imaging model, familiar from laser printers; Display PostScript is a screen version of the PostScript imaging model. The Macintosh imaging model is QuickDraw, and Microsoft's PM for OS/2 uses an imaging model called GPI (for Graphic Programming Interface).

Some GUIs support more than one imaging model. For example, while Sun's NeWS (for Network Extensible Window System) is similar to the PostScript imaging model, it can also turn the screen over to a complete graphics imaging system (such as PHIGS or GKS) for controlling a CAD program.

The API is a set of programming-language function calls—it's how the programmer specifies which windows, menus, scroll bars, and icons will appear on the screen. Both PM and

continued

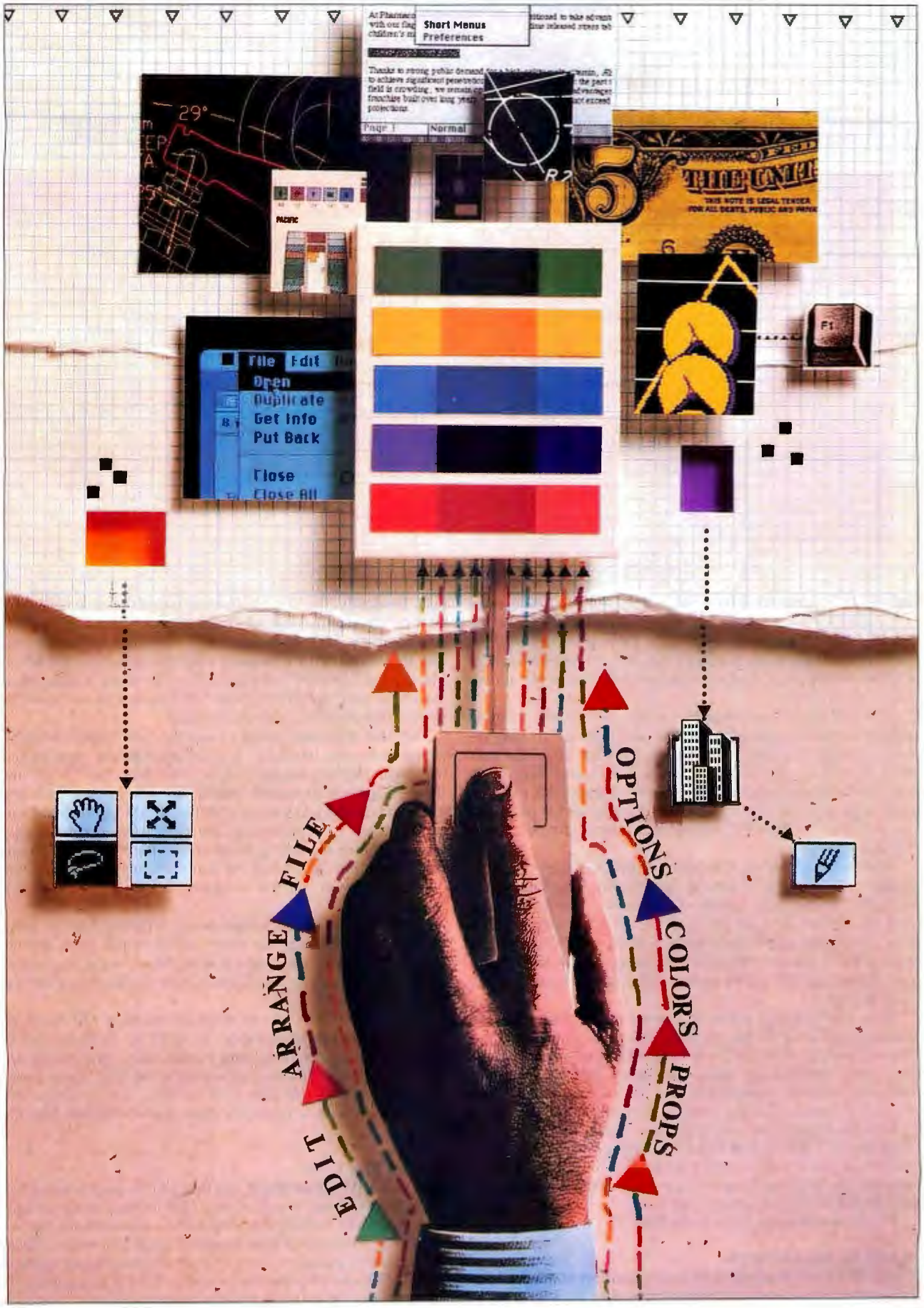
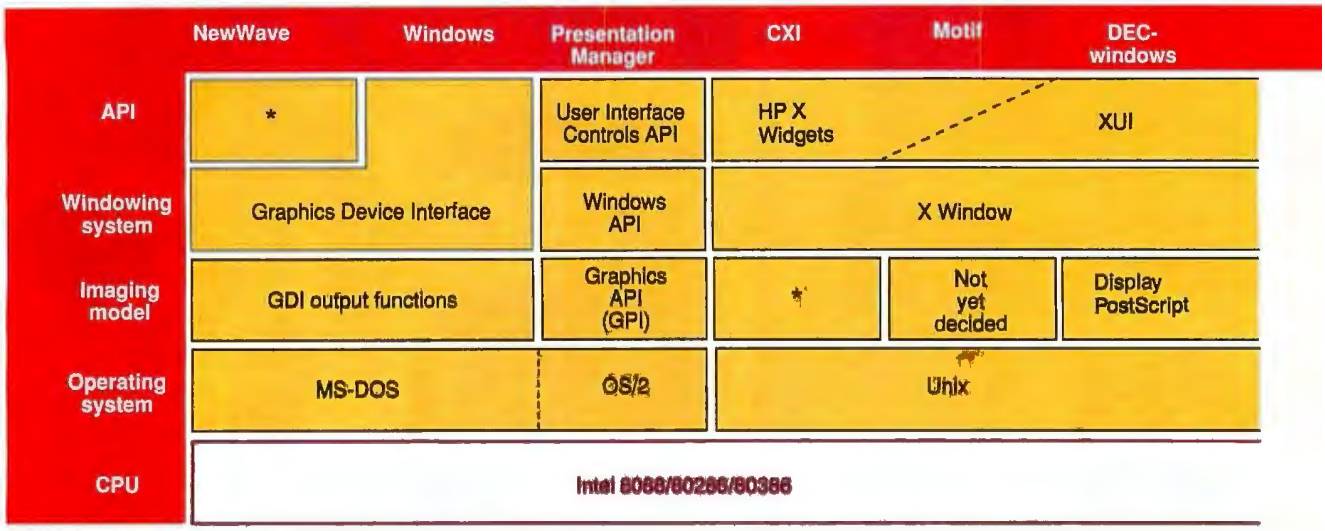


ILLUSTRATION: R. M. SCHNEIDER © 1989

Figure 1: *Graphical user interfaces tend to fall into a few camps: those based on IBM's Systems Application Architecture (primarily Windows and Presentation Manager), Unix systems generally built around X Window, and Mac-like systems*



Microsoft Windows have their own APIs. DECwindows uses an API called XUI (for X User Interface), which includes function calls for the X Window System. Open Look is the new API for Sun's operating system. NextStep uses its own API (defined by a library of objects called *kits*) and its own windowing system (the *window server*).

On top of these three elements—windowing system, imaging model, and API—some systems also have tools for creating interfaces and developing integrated applications. Hewlett-Packard's NewWave, for example, is not a user interface, but a method for integrating applications and objects from multiple applications—it's a development tool for application programmers. Similarly, NextStep includes a set of tools for object-oriented programming.

Another characteristic that varies widely is the level of integration between the GUI and the operating system. Some GUIs are tightly bound to the system—turn on a Mac, an Amiga, or a NeXT computer, and the GUI appears automatically. By contrast, you must specifically choose Microsoft Windows and most of the X Window GUIs that run under Unix—which could be a hindrance for Unix-based systems trying to appeal to a mass market.

Some GUIs provide access to a conventional command-line interface that lets you, for example, pass arguments to applications or view the text of a file without using the mouse, menus, and icons. NextStep has a console window that lets you get at the command line, whereas the Mac makes you use a desktop accessory to examine files, manipulate them, and so on.

With the similarities and differences defined, it's easier to break the GUI family tree into a few large groups: those based on the distinctive look of IBM's SAA; those built upon X Window and the Macintosh and its apparent offshoots; and a few hard-to-define hybrids and special cases.

We'll begin by going back to the Mac.

A GUI for the Rest of Us

The idea of a standard user interface, regardless of the machine the user is facing, was part of the dream that built the Macintosh. Ironically, the Mac has become one of the most isolated of GUI-based machines, largely because of Apple's litigiousness. Any company that even looked like it might be copying the Mac was threatened with a lawsuit. (See the text box "Of Mice, Menus, and Lawyers" on page 256). The result is that, while the whole world has followed the Macintosh in its use of GUIs, most of that world has gone its own way.

The Mac GUI (see photo 1) was the first widely available mouse-and-menu interface. It established several conventions that have reached beyond GUIs, including the "point and shoot" approach to menus. Before the Mac, you'd look at a menu and choose a key to type. After the Mac, your selections were limited to contextually correct answers—you simply couldn't choose something meaningless. Point-and-shoot interfaces—whether graphical or character-based—eliminated "wrong" answers, since it's impossible to select a choice that isn't available.

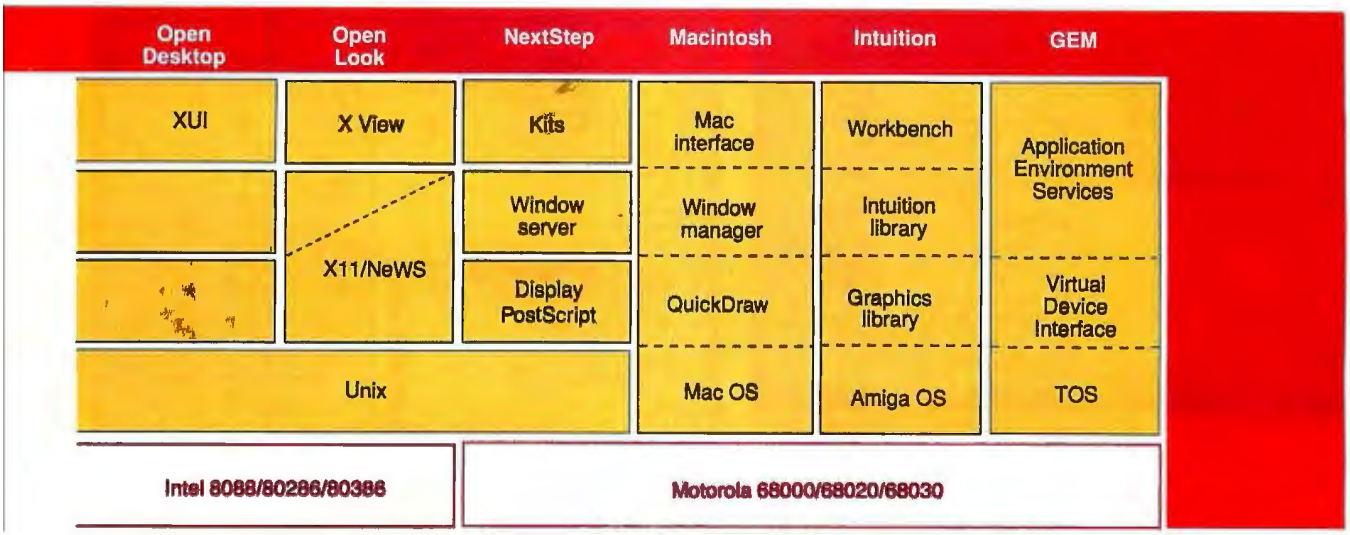
Although its stylistic guidelines are certainly heavily documented, the Mac interface really specifies just three distinct operating systems: the single-tasking Mac Finder, the multitasking MultiFinder, and Apple's own Finder clone for the Apple IIGS, ProDOS-16.

The Mac GUI combines all the functions of an API, windowing system, and imaging model in its ROM Toolbox, QuickDraw graphics primitives, and Finder, and these pieces are tightly integrated. The stark efficiency of the QuickDraw imaging model allows the Mac GUI to have reasonable performance, even with a relatively slow microprocessor like the 68000.

The Big Blue Look

IBM's SAA is both more and less than a GUI. SAA is actually a whole family of user interfaces that IBM defined two years ago. SAA interfaces include everything from ground-level character-only systems up to high-powered graphical workstations, and they span machines from PCs up to mainframes running IBM's MVS and VM operating systems. SAA is a complete system architecture, and as a result it covers things that most user

that tend to be tightly integrated and distinctive. In this figure, a dotted line indicates some overlap between the objects on either side of it. An asterisk indicates that the technology is proprietary or that the company has no specific name for it.



interfaces don't—including a standard for networking called the Systems Network Architecture (SNA), and one for database queries, the Structured Query Language (SQL). It also specifies, but doesn't rigorously define, the user interface. An SAA user interface isn't necessarily a GUI, complete with mouse and graphics. Remember, SAA is a standard for everything from glass teletypes on up, so SAA GUIs are really just a subset of SAA user interfaces.

SAA seeks to let any terminal handle any SAA application. Thus, while all SAA applications use the same style of drop-down menus, character-only systems will display only characters—and send only characters back to the application—while mouse-based graphics systems will let the user point and click. However, SAA does create a least-common-denominator situation: The application software ultimately has to choose what the minimum configuration for the SAA terminal is going to be. Fortunately, SAA applications that use terminals are much more likely to involve transaction processing—things like airline ticket reservation systems—rather than CAD systems or paint programs.

The PC-level GUIs that implement SAA are Windows for MS-DOS systems (see photo 2) and PM for OS/2 (see photo 3). Several GUIs based on X Window, including CXI, Motif, and PM/X (discussed below), have an SAA/Windows/PM look and feel designed to let users adapt easily from DOS-based systems to Unix-based systems. (In its original version, Windows was much more Mac-like in its appearance, but between a threatened lawsuit by Apple in 1985 and IBM's definition of SAA in 1987, it has come to look and act like the rest of its close brethren.)

The critical and most distinctive element of SAA GUIs is the fact that they don't depend on a mouse at all. You can do anything in an SAA GUI without a mouse, and, in fact, the system leans heavily on keyboard equivalents, including function keys. (You can gauge the pervasiveness of SAA's influence in the PC world by counting the number of DOS applications that now use the F1 key as the Help key.)

A characteristic element of the mouse-independent nature of SAA GUIs is the menu bar, which in SAA-speak is called the Action Bar. While the Mac interface requires a mouse-click to pull down a menu, you can do it in an SAA GUI by pressing a key instead.

Another characteristic of SAA GUIs is the style of windows they use. Unlike the Mac window, the size and shape of which you change by dragging the box in the lower right corner, an SAA window can be stretched by any of its borders. And under OS/2, there's an added feature: You can "minimize" a window down to an icon, and the program running in the window will continue to run. (You can also minimize a window under Microsoft Windows, but since Windows is not a multitasking operating system, the program in the window suspends operation until you "maximize" it again.)

While the DOS-based Windows and OS/2's PM share the SAA look and feel, each has its own API, imaging model, and windowing system. Although these parts are similar, they are not directly compatible, and porting an application from Windows to PM is not necessarily an easy task.

The emergence of powerful 80386 machines and the increasing acceptance of Unix as an operating system for them has led to a curious convergence between PM and Unix-based GUIs.

The Unix Brand: X

X Window user interfaces are a wide-ranging group—but underneath it all, X is X. The current version, X11, has become the most popular windowing system for Unix workstations, for two reasons. First, software that's written for the X Window System can (at least in theory) use any X Window display. The application program sends calls to the X Window library, which packages the display requests as X packets and sends them along to the X Window server, which decodes the X packets and displays them on the screen.

If that sounds a little complicated, it's because of X Window's second advantage: Since X Window is designed to work with networks, the software (called a *client application*) and the display may be on different computers. For example, the

continued

A GUI Gallery

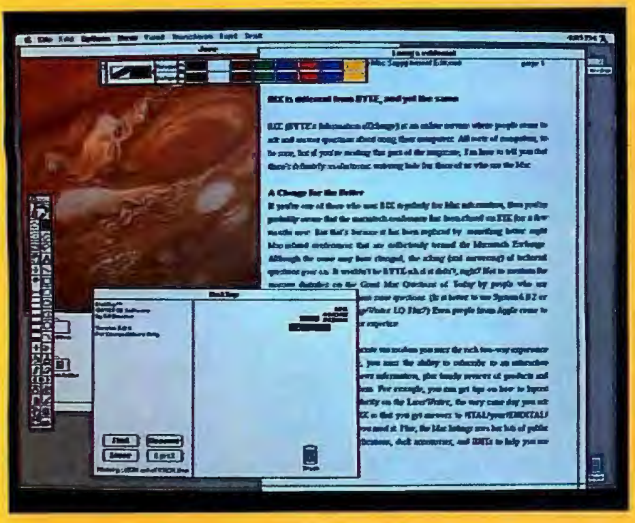


Photo 1: *The familiar Macintosh interface, with its windows, icons, and pull-down menus, launched a thousand graphical user interfaces—which promptly took off in their own directions.*

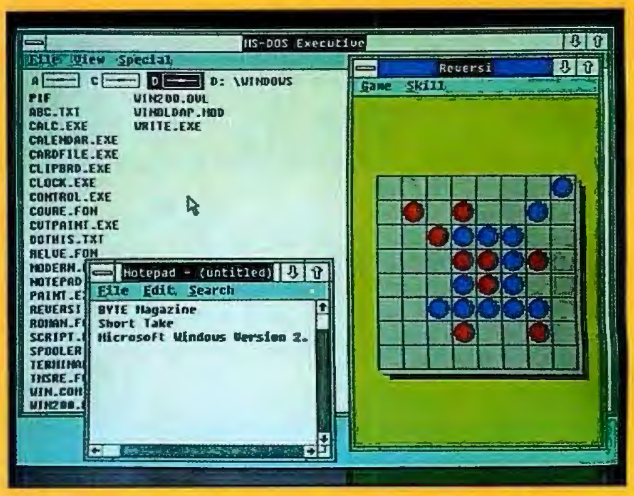


Photo 2: *In its original incarnation, Microsoft Windows looked more like the Macintosh interface; a threatened lawsuit from Apple, as well as IBM's solidification of its Systems Application Architecture, forced a shift to what is now the standard SAA look.*

Photo 5: *The Common X Interface (CXI), developed by Hewlett-Packard and Microsoft, features a Presentation Manager look on an X Window platform.*

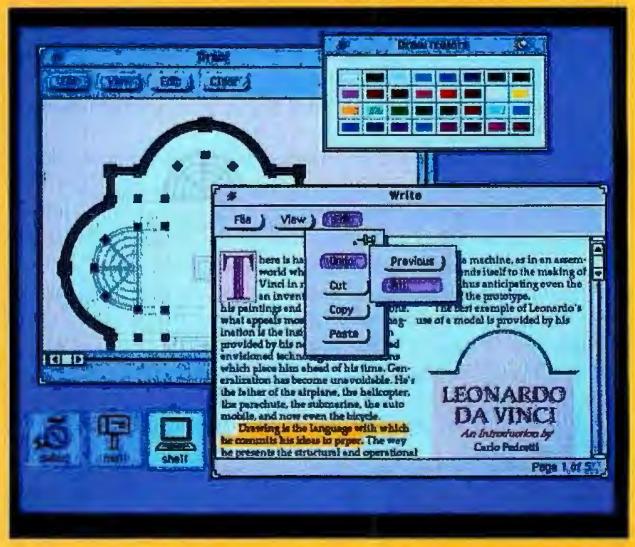


Photo 6: *Open Windows, from Sun Microsystems, features the Open Look interface, which provides several enhancements to the classic windows, icons, and pull-down menus interface.*

display can be on a workstation, while the application itself can be running on a mainframe or supercomputer. That's why the display requests have to be put into packets, so they can go zipping along the network as quickly as possible.

Exactly how those packets will be displayed on a workstation depends on the set of *widgets*, or predesigned window elements, the workstation uses. A radically different set of widgets could make the same program appear different on two separate workstations. But even if the look is different, the behavior of the program will be the same. For example, one workstation might have windows with a Close box in the upper left corner, while another might include it in a pop-up submenu. They'll look different—but whether you click on the Close box or select Close, the window will still close.

Photo 3: *OS/2's Presentation Manager is heir to the Microsoft Windows look and feel, although application developers have found that some similarities are only skin deep. Currently, several developers of graphical user interfaces for Unix systems are licensing the PM look for X Window–based interfaces.*

Photo 4: *DECwindows, Digital Equipment Corp.'s graphical user interface, was recently licensed by SCO for its integrated Open Desktop product.*

Photo 7: *Motif, the graphical user interface designed by the Open Software Foundation, combines DEC's XUI and HP's X Widgets with a Presentation Manager look and NewWave's three-dimensional windows on an X Window platform.*

Photo 8: *NextStep, the user interface for Steve Jobs's NeXT machine, includes a set of tools for object-oriented programming.*

Because X Window is so widespread on Unix workstations, hybrids have cropped up—on some systems, not all display operations are routed through it. For example, Sun's Open Windows system runs on NeWS in parallel with X Window; some display functions go through X Window, while others are handled by NeWS.

Currently, the "look" of X Window GUIs is divided into several camps: Hewlett-Packard uses an API called HP X Widgets. DEC based its DECwindows interface (see photo 4) on its XUI. Recently, Hewlett-Packard and Microsoft developed the Common X Interface (CXI) (see photo 5), with the look and feel of PM but working within an X Window environment. The Open Windows system from Sun Microsystems (see photo 6)

continued

Of Mice, Menus, and Lawyers

In 1985, Apple Computer threatened legal action against Digital Research, for its GEM operating environment, and Microsoft, for Windows. It claimed that the products infringed on Apple's copyright for the visual display of the Macintosh. Both companies signed agreements with Apple to resolve the disputes out of court.

According to the Apple-Microsoft agreement, Apple was willing to tolerate Windows 1.0 and several other programs (such as Excel) as long as Microsoft acknowledged that the displays of those programs were "derivative works of the visual displays generated by Apple's Lisa and Macintosh graphic user interface programs."

Then, in March 1988, Apple Computer filed a lawsuit against Microsoft and Hewlett-Packard, claiming that Microsoft Windows 2.03 and Hewlett-Packard's NewWave (which runs on top of Windows) infringed on the Macintosh's copyrighted visual display. Although versions 1.0 and 2.0 of Windows are not all that different (version 2.0 has overlapping windows, fatter screen borders, minimum/maximum icons for sizing windows, and mnemonic keyboard selections in menus and dialog boxes), Apple apparently thought that the program was beginning to look too much like the Mac interface.

Microsoft—mindful of its role as a major provider of Mac software—responded that the latest versions of Windows were covered by the 1985 agreement. Hewlett-Packard, which sells very little software for the Mac, went further, filing a countersuit against Apple. According to the Hewlett-Packard suit, the Macintosh copyrights were invalid because Apple didn't originate its displays but copied them from the work of windowing-interface pioneers such as Xerox's Smalltalk and Star interfaces. The suit also claimed that Apple had coerced Microsoft into signing the 1985 agreement and was trying to illegally prevent competition in the market for window-and-icon user interfaces.

While many observers thought that Apple could not win the suit, the judge in the case surprised them: In March, he ruled that version 2.03 of Windows was *not* covered by the 1985 agreement. (A ruling that it was covered would have ended the case in Microsoft's favor.)

At this writing, the case is headed for trial to determine whether or not Windows and NewWave infringe on Apple's copyrights. While an out-of-court settlement is again a possibility, some industry observers are concerned that a victory for Apple could spell trouble for other user interfaces and developers of software for those interfaces.

uses Sun's Open Look interface (see "Face to Face with Open Look" by Tony Hoeber, December 1988 BYTE).

Now, however, there is some movement toward a consensus, thanks in part to the Open Software Foundation. Last year, the OSF asked major software developers to submit GUI technologies for consideration as part of a standard operating environment for Unix. To most people's surprise, the OSF chose pieces from three companies—DEC, Hewlett-Packard, and Microsoft. Motif, as the OSF GUI is called, looks like PM, uses parts of the DEC and Hewlett-Packard APIs (as well as the three-dimensional windows from Hewlett-Packard's NewWave), and is based on X Window (see photo 7). The imaging model for Motif has not yet been selected.

Following the announcement of Motif, many companies announced support for the OSF standard and began tweaking their GUI software to be compatible with it. Hewlett-Packard and Microsoft are working on a version of PM for Unix (PM/X), with pieces similar to CXI and Motif. (While CXI merely *looks* like PM but is still based on X Window, PM/X will have its own windowing system. The idea is that PM/X will make it easy for application developers who have created applications under OS/2 to port those programs to Unix.)

Then, in February, The Santa Cruz Operation (SCO), which supplies Xenix, announced Open Desktop. This is a complete user interface for 80386-based Unix systems that incorporates the Motif GUI, DOS compatibility, SQL database facilities, and network support. Even IBM has announced support for Motif, despite the fact that it had earlier licensed the NextStep interface from NeXT. Although it's unlikely that IBM will support two different and incompatible user interfaces on its Unix platform, it could use some of the NextStep technology, such as the development toolkit and object-oriented programming features. Or IBM may have just been hedging its bets when it licensed NextStep, in case OSF failed to come up with an accepted standard interface.

Yet another GUI for X Window is X.Desktop, from IXI, Ltd., of Cambridge, England (see the text box "Managing the X Window Desktop" by Dick Pountain, page 356, January BYTE). X.Desktop incorporates its own API, although the company is working on implementations that use the Motif and Open Look APIs.

The multitude of SAA GUIs for Unix points up one of the major problems in trying to sort out GUIs—these things don't belong to simple categories. For example, CXI and Motif are X Window GUIs with an SAA look and feel. From the programmer's point of view, they belong to the X camp; from the user's standpoint, they've clearly got the PM look and feel.

Because X Window works on networks, it makes distributed computing a real possibility with mouse-and-menu GUIs. Unfortunately, anything that is graphics-intensive requires a lot of information to pass along a network, which can really slow down response time. X Window users complain that when you move the mouse, you have to wait several seconds for its on-screen pointer to catch up. On the other hand, X Window is the only GUI system that really does work in a multiuser, multicomputer, networked environment. For now, if you want to run windowing software on a Cray supercomputer and see the result on your personal desktop machine, X marks the spot.

The Mac-Like GUIs

Although the Macintosh essentially stands on an island in the GUI world, there are at least two other Mac-like GUIs. One is the original version of GEM from Digital Research (which survives on the Atari ST). Another is the user interface for Intuition, the operating system for the Commodore Amiga.

GEM was originally intended to be highly Mac-like, and so it was; so much so that in 1985, Apple threatened to sue Digital Research for copyright infringement. Digital Research responded by removing the offending features (including overlapping and movable windows) from the PC version of GEM, but the Atari ST version still has a Mac-like GUI. However, the ST lacks many of the Mac Desktop's niceties of implementation,

such as long filenames, the ability to remove things from the Trashcan, proportional typefaces, and automatic saving of the desktop.

While the Amiga's Intuition wasn't threatened with an Apple lawsuit when it first appeared, it too shared many Mac-like characteristics. But Intuition added a feature that Apple didn't include until several years later: It was the first widely used multitasking GUI. Unlike X Window and SAA, Intuition isn't really designed for remote applications—it's a single-user multitasking system. But if the Finder is the father of desktop computer GUIs, Intuition is arguably the father of MultiFinder.

The Next Wave

NextStep (see photo 8) represents the high end of GUIs for single-user computers. NextStep itself is a huge piece of the operating system of the NeXT computer, including a number of utilities that would probably be viewed as applications on most systems. More than any other GUI, NextStep resembles the Mac in its ambition—it wants to change the world. But it also scrupulously avoids being too Mac-like. Unlike the Mac, where a file-selector box can display the files of only one directory at a time, the NeXT GUI can display files in multiple hierarchical pop-ups. It's sort of an improved version of the Mac file-selector box.

NextStep does have application icons; in fact, you can drag an icon out of a window and onto the desktop for convenient use. The idea is to keep regular-use items handy.

The other way NextStep resembles the Mac philosophically is in its rejection of anyone else's standards. In the Unix world, the windowing standard is X Window—but NextStep doesn't use X Window. In fact, nothing in the networking world works with NextStep except NextStep—in many ways, it is designed for a powerful single-user PC running Unix rather than for a fully networked machine.

Another hard-to-classify system is Hewlett-Packard's NewWave, which the company likes to call a "software applications environment." Currently built upon Windows, NewWave features an Object Management Facility that lets you incorporate pieces from different types of applications—word processor, spreadsheet, graphics program, whatever—into NewWave documents. A task manager, called the Agent, acts as a kind of supermacro processor to let you automate repetitive tasks involving a number of different applications. As such, NewWave is part GUI, part "super-application." Hewlett-Packard is developing one version of NewWave that will run on top of PM and another that will run on Unix using the Motif GUI.

Windows of Opportunity

In the months and years to come, you can expect to see even more interesting things popping up in the windows on your screens: extremely high-resolution images, multimedia applications, full-motion video, and new ways of interacting with data. Programs like NextStep and NewWave point the way to the future, where intelligent interfaces may not only help you to automate everyday tasks, but may even anticipate your actions and thereby increase productivity.

The real question is no longer the one the Macintosh raised in 1984—whether to use a GUI. Today the issue is what *sort* of GUI: which elements are most important, and which you can sacrifice in favor of things like better network performance or low cost. ■

Frank Hayes is an associate news editor and Nick Baran is a senior technical editor for BYTE. They can be reached on BIX as "frankhayes" and "nickbaran."