INSIDE DIGITAL RESEARCH'S FLEXOS 386 AND X/GEM PART 4

Multiuser/Multitasking Graphics
by Peter Ruber

X/GEM represents Digital Research's second-generation graphical interface, based on the GEM technology that was begun in 1983 and released in 1985. Although the underlying system software architecture is radically different from the single-user GEM product, there exists a relationship between the two.

This relationship can be described on two levels: X/GEM maintains the same user interface, and it provides a run-time environment that is independent of the underlying hardware and system software.

Unlike Microsoft Windows, which is married to the Intel microprocessor technology and is, therefore, hardware-dependent, X/GEM (like its predecessor) can be ported to other CPU platforms. Also, unlike Windows, which is dependent on DOS software, X/GEM is adaptable to any operating system standard.

GEM, which began in the DOS environment (running under PC/MS-DOS, DR DOS, and Concurrent DOS) is also available for the 68000 CPU as the operating system on the Atari ST computers. It has already been ported to Presentation Manager under OS/2.

X/GEM has also been developed and ported to the Unix X-Windows environment for the Intel 80386 and Motorola 68000 CPUs, and is now available as an integrated component of the FlexOS 386 operating system running on 386 desktops and 386 Multibus II minicomputers. (X/GEM has also been implemented for 80286 system.) This gives GEM and X/GEM a software universality that is not available with any other system software.

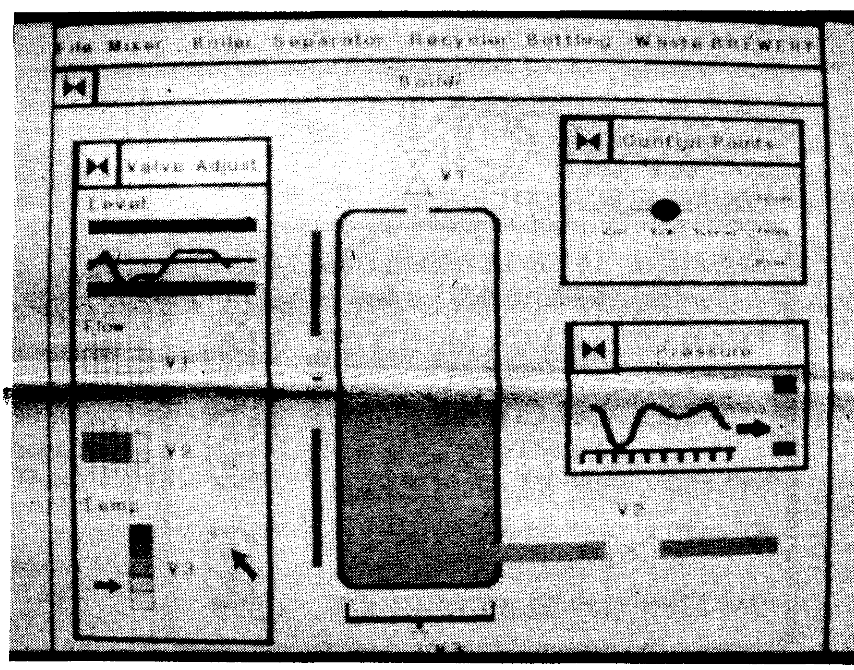


Figure 1: X/GEM has realtime multitasking functionality under FlexOS 386.

With respect to the X/GEM implementation under Unix's X-Windows System version 11, Digital Research is working with a number of Unix vendors on application development. There is no urgent user demand for X-Windows applications either, as the X-Windows user interface is still in development. That will come soon enough, though, because Unix needs a friendlier user interface in order to compete in the DOS and MAC marketplace.

In September 1988, the group of Unix hardware developers who had formed the Open Software Foun-

◆ INSIDE DIGITAL RESEARCH'S FLEXOS 386

dation to create a standard graphical interface, held a fair to review available graphics technology. X/GEM was one of about two dozen products submitted for examination.

X/GEM was the only graphical interface sufficiently along the development path to be released within a matter of months. X/GEM met all the criteria of the OSF specifications, but the conference instead decided to merge interface technologies from several companies (DEC, Microsoft, Hewlett-Packard).

Several participants did not go along with the OSF selection. IBM, which had already acquired technology from Metaphor and NeXT, went its own way. Big Blue was thinking about its AIX derivation of Unix as a hedge against the possible future failure of OS/2 and Presentation Manager. IBM also acquired a financial stake in about 10 other companies developing object-oriented interface technologies.

The Santa Cruz Operation, the largest Unix developer, also decided to do its own thing. The company poured millions of development dollars into Open Desktop and OSF/Motif, then had to sell a chunk of the company to Microsoft in order to stay afloat.

Aside from the need to develop a consistent user interface for X-Windows, the motivation to create the OSF was fueled by AT&T's cross-development agreement with Sun Microsystems.

Hardware and software developers believed the AT&T-Sun alliance would unfairly tip the scales in Sun's favor. The accord would give Sun an important advantage by affording early access to AT&T's graphics technology months before they were allowed to license it.

In this highly competitive industry, the first to market with a product has a hands-on advantage over the competition. A second consideration was concern that the IBM-Microsoft push for OS/2 and Presentation Manager might steamroll over Unix.

The OSF seriously erred by overlooking X/GEM. Politics got in the way of judgment when the group ignored a graphics interface it could adopt and put to immediate use to challenge Presentation Manager.

As a result, the OSF's software development tools will not be ready for at least a few months. Don't expect applications until 1991 or 1992. A 32-bit version of OS/2 and Presentation Manager may be upon us before then.

Realtime Automation Graphics

It took about two years to develop the multitasking X/GEM environment. During that time, the X-Windows and FlexOS versions were done in tandem. In both cases, the X/GEM system software uses the multitasking/multiuser resources and the interprocess communications services of the operating systems.

Since multitasking is not a DOS attribute, Windows must provide those services to its application base. Therefore, it's more complicated to program for the Windows environment than for GEM applications, which are single-tasking.

Windows programming is different than programming for DOS. Instead of the programmer being in control, Windows is in the pilot's seat. The programmer finds himself writing an event handler and then responding to Windows messages. Some programmers compare this to learning to drive on the wrong side of the road.

In many respects, Windows can be said to be an operating system running on top of an operating system. So is GEM, for that matter. GEM is the graphical interface that manages DOS services for input/output to disks and serial/parallel ports, and file management services. By building operating system services into the GEM environment, Digital Research and Atari created a pure graphics environment similar to the Macintosh.

Presentation Manager is more or less a superset of the Windows environment, but it doesn't provide a consistent programming interface, forcing application developers to master yet another new programming environment.

This will hold true even of Windows 3.0, which will reportedly look exactly like the Presentation Manager interface. When GEM made the move to X/GEM for multitasking

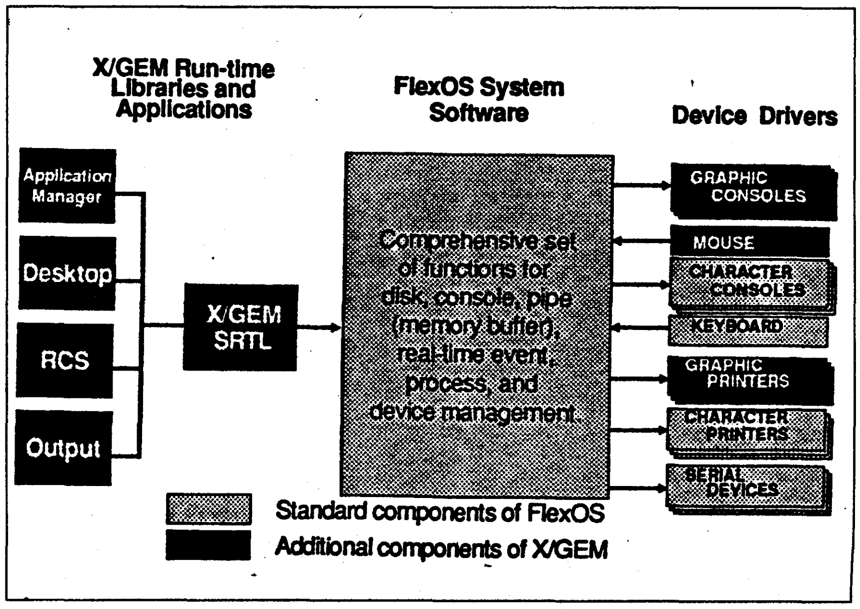


Figure 2: X/GEM is a loadable shared run-time library, set of applications, and set of drivers.

and multiuser environments, Digital Research maintained source-code compatibility between the two in order to provide an easy upgrade path for application developers. To make a GEM application run on X/GEM under FlexOS 286 and 386 and Unix, a programmer merely links certain routines required by the operating system, and then does a recompile of the source code. A single programmer can generally do this in two to four weeks, depending on the complexity of the program.

There is a benefit attached to running X/GEM under FlexOS and Unix. Since X/GEM is independent of the underlying hardware and software, it doesn't become a rigid or concrete-like interface that Presentation Manager is under OS/2. X/GEM applications can run concurrently with FlexOS native-mode applications.

Microsoft's OS/2 has to be customized by an OEM before it will boot. This customizing includes resolving ROM BIOS incompatibilities and writing device drivers for various hardware components. Therefore, one manufacturer's implementation of OS/2 is not guaranteed to run on another's. This tends to lock users into a single hardware platform. If the user wishes to add a faster hard disk controller or a new graphics board, he cannot do so unless his manufacturer supports this hardware. FlexOS 286 and 386 include drivers for hard disk and floppy controllers, Monochrome, EGA and VGA graphics, and serial/parallel I/O, so that the operating system will boot from any off-the-shelf 286 or 386 hardware.

In the FlexOS version, X/GEM also utilizes the operating system's realtime response attributes and event-handling capabilities by facilitating the implementation of graphics in factory automation. X/GEM provides a straightforward solution for emergency handling in those environments when timely operator response is mandatory.

For example, consider a vision system monitoring part alignment on an assembly line. When a misaligned part is detected, a monitoring process flashes an alert to the line person and writes to a semaphore pipe. This wakes a second process which retrieves the proper alignment specifications from a database and writes the information to another pipe, which wakes a third process which then reads the data, realigns the part, and signals the fix on the console. On FlexOS, processes two and three do not exert any CPU load except when they're awake. (See Figure 1.)

Realtime also has its applications in office environments. Here realtime may not be such a critical requirement, but it can be deployed in a variety of ways. Electronic mail can interrupt a user's application by displaying a message, or a user's

◆ INSIDE DIGITAL RESEARCH'S FLEXOS 386

planner program can pop up and tell him when it's time to go to a meeting.

A Consistent Programming Interface

Having studied the GEM Programmer's Toolkits from their early releases through the current X/GEM product, I have noticed that Digital Research has adhered to a consistent development methodology that has addressed the following issues: modularity and structured design; file and function header formats; module, file, variable, and constant naming conventions; coding syntax, indentation, code style, and commenting; portability considerations, including portable type declarations; source-code control; and unit test, system test, and external beta test cycles.

X/GEM's open architecture makes no assumptions or restrictions about language bindings or development environment. Bindings have been developed for several versions of C, as well as for FORTRAN, Pascal, COBOL, Lisp, PL/I, BASIC, Modula II, Assembler, and Ada. The development of new language bindings is typically as simple as modifying the existing bindings for the language's parameter-passing conventions and numerical representations.

By not placing restrictions on the application developer regarding his/her choice of development environment, developers are free to use virtually any language, editor, and debugger they are familiar with. Additionally, the X/GEM API makes no assumptions about the form of input device. Whether a mouse, touch-screen, keyboard, or some other form of input device is being utilized, the X/GEM API presents a uniform set of input primitives that are independent of type of input device. The X/GEM Desktop itself is, in fact, usable in keyboard-only mode.

Aside from the clean, layered architecture of the X/GEM interface, it has been designed for extensibility and upward compatibility. The API has gone through three major versions in its lifetime, as well as ports to very disparate environments, always maintaining upward compatibility with previous versions. While GEM's archrival Windows has also gone through numerous versions with enhanced features, there exists little upward compatibility in making use of these features unless the application software undergoes major revisions.

The System Software

X/GEM is composed of function libraries, applications, and graphics device drivers. The function libraries provide a run-time environment, independent of the underlying hardware and system software, because of its layered structure. When an application makes a call to either the operating system or the CPU, it doesn't do it directly; X/GEM inter-

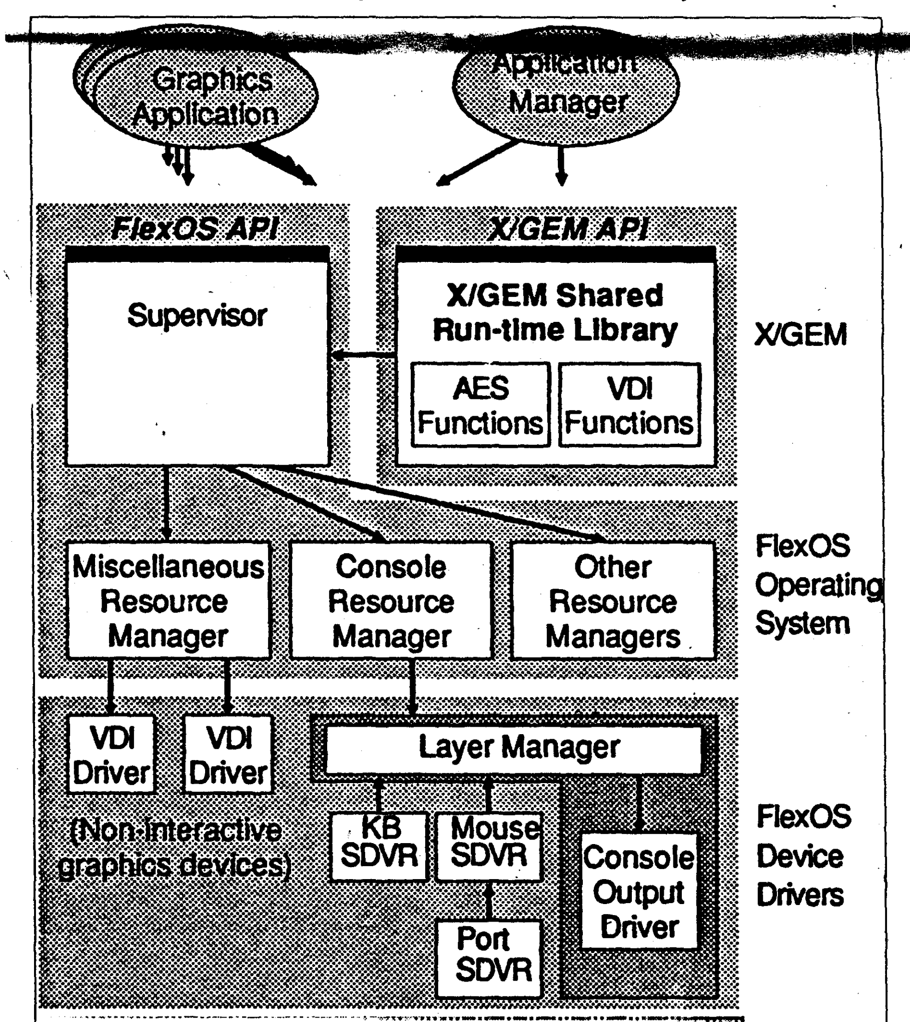


Figure 3: Internally, the X/GEM Shared Run-time Library has two logical parts: AES and VDI functions.

cepts the call and handles the communication to the OS and CPU. As such, it doesn't matter what the operating system is or what the CPU is, the application cannot talk directly to either. The applications consist of an output display utility, an icon-based file management utility, a graphics application development aid, and an application manager. The manipulation of the screen display and information, and the management of the keyboard and pointing device input, reside in two primary modules: the Virtual Device Interface (VDI—a precursor to the Computer Graphics Interface standard), and the Application Environment Services (AES). The graphics capabilities are built into the VDI, and the user interactions are handled by the AES. Input and graphics output devices are managed through drivers.

VDI graphics routines are two-dimensional and primarily vector-oriented. Additionally, GEM's VDI has been extended for text-oriented raster operations. The interface is optimized to run on systems with a very wide performance range, and was built especially for applications requiring a high-quality user interface.

The VDI was not intended for applications requiring highly rendered 3-D pictures or large raster image processing. These functions add significantly to the processing power requirements of the system, and require a great deal of software to support—both expensive items—and add negligible value for automation graphics. Three-dimensional icons are being incorporated into SCO's OSF/Motif and into Presentation Manager. While they add a nice visual touch, they also consume extra system memory, which can affect performance.

AES functions provide a number of diverse services, including the display of and user interaction with menus and dialogues (forms of information). In addition, the AES can handle graphics resources—logical groups of graphical objects and text strings—which comprise the menus and dialogues and also allow for language independence of applications. This capability is especially important to multinational vendors, because it allows them to modify program text without rewriting and recompiling the program code for each language.

Most important to the GEM programmer are the event-handling mechanisms provided in the AES. The event routines dictate a pro-

◆ INSIDE DIGITAL RESEARCH'S FLEXOS 386

I have also learned there is now an OEM version of MicroFocus Level II COBOL available for FlexOS. Also, in December 1989, Digital Research will release a series of linkable libraries that will be object code-compatible with DOS C languages. Users will be able to develop FlexOS applications using Microsoft C and Microsoft TURBO C.

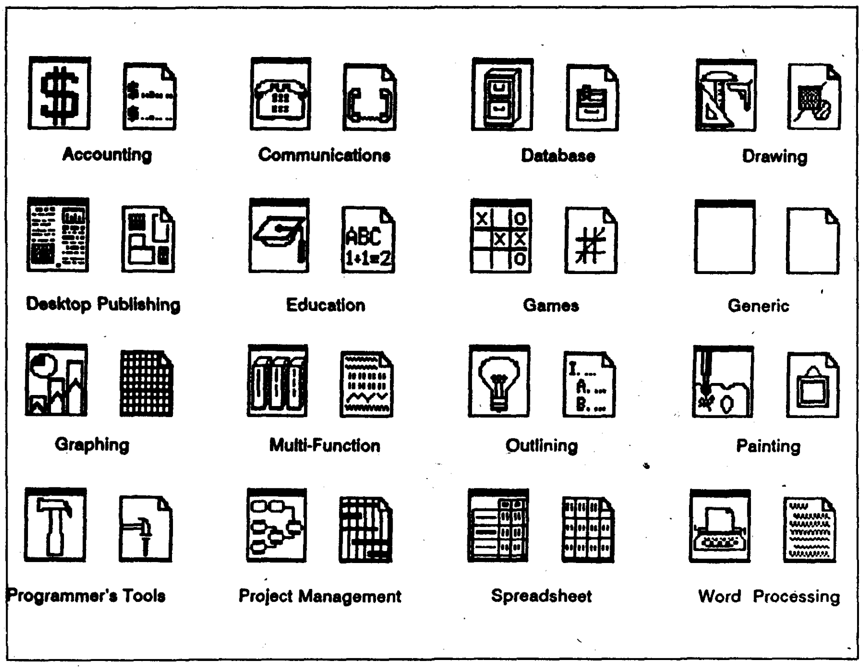


Figure 5: A sampling of X/GEM application icons.

Demo disk: According to DRI's FlexOS marketing group, they will have a demonstration disk available in late December 1989 that will illustrate some of FlexOS' key features, plus examples of realtime applications. As of late August, this project was just being assigned to a firm that specializes in developing demonstration disks. Since this issue will be appearing on or about the time the FlexOS demo disk is scheduled for release, you might want to call DRI for a copy.

The Desktop Windows

There are two windows in the X/GEM environment. The first is the Desktop Window, which can also be described as a virtual console. Its use is reserved by the X/GEM operating environment. It consists of a series of border components, as shown in Figure 4. There are seven buttons that enable you to manipulate the Application Window. You can move it up or down, and left or right. Dragging the Size Box button allows you to shrink or enlarge the Application Window Box. The Full Box lets you return a box to its original state. The Close Box button removes the current window and brings up the previous folder or the drive icons. By continuously clicking the Close button you move through all previous folders and close them. When all elements of an active window have been closed, the next window underneath—if there is one— becomes active.

The Scroll Bar and Slider allow you to view other areas of the Application Window that may not be visible. In a standard 80x25 screen, you are able to view less than 25 percent of the actual area available to an application. Applications employ a "view" feature in which

◆ INSIDE DIGITAL RESEARCH'S FLEXOS 386

you can "zoom" in and out of a window and/or select predetermined sizes of the working screen. On the top of the Desktop Window are the Title Bar and Information Line. The Title Bar displays the directory drive and path information and the name of the file active in the Application Window. The Information Line is a reserved area of the screen managed by the Application Environment Services (AES) that contains programmer-defined information. It displays Files and Options that interact with the operating system for such operations as Open a file, Info/Rename a file, Delete a file, Format, go to Output, Quit, Assign a drive Icon, Register Application, Set Preferences, Save Desktop, and Enter Shell Commands. (The X/GEM Shell Command under FlexOS/386 is an important feature that I'll discuss later.)

When an application is loaded, menus pertinent to that application join the Files and Options of the Desktop. All menu selections have drop-down windows presenting choices the user can access. Options within drop-down windows generate pop-up windows called Dialogs and Alerts.

Dialogs request information from the user for such diverse actions as filenames, drive location, list of available files, font sizes, colors, preferences, etc. It is possible to have nested Dialogs (or message trees) for a single option. Alerts let users know if they neglected to provide a piece of information requested by a Dialog. They also prompt users to save files before deleting a window.

All the activity is managed by the user through a mouse or other pointing device.

There is also a Move Bar function that occupies the same space as the Title Bar, when present. When the user presses the mouse button while the mouse form is on the Move Bar, the AES displays an XORed outline of the window. The user can drag this outline around the desktop as long as the button is held down. When the user releases the button, the outline disappears, and the AES sends a WM_MOVED message to tell the application that the user wants the window moved to the location indicated by the outline's last position.

Mouse Input

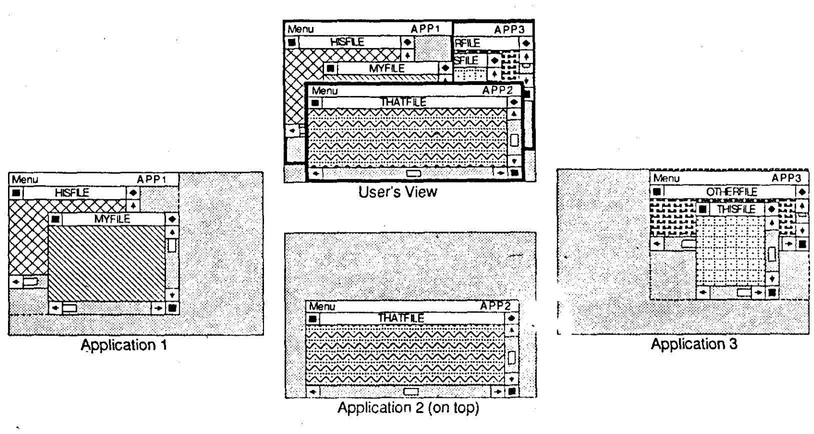
All the activity is managed by the user through a mouse or other pointing device, such as a tablet or stylus. The mouse is a catalyst for events such as mouse clicks, dragging an object, opening pull-down menus, or other user interaction. Alternatively, events can be enabled from the keyboard by using the arrow and select editing keys. Certain file management tasks that are accessed from within pull-down menus can also be accessed using function keys or Alt and Ctrl key sequences. The basic mouse techniques are:

Click: Press the button once. This selects an object which is then highlighted. An icon or a menu option is shown in reverse video. Images or text in the view window are blocked off.

Double-click: Press the button twice to open an object for editing. If you double-click a drive icon, for example, a list of filenames appears on screen. If you click the drive icon only once, the mouse has to pull down the File menu and then click the Open command.

Drag: (1) Press and hold down the button; (2) move the mouse; (3) release the button. This action copies parts of an object from a parts box to the view window; moves an object inside a view window; or moves an object to or from clipboard.

Multiple Graphic Applications On X/GEM



Currently running applications can start new applications and achieve true multitasking on a single physical console. In a multi-user environment, each graphical workstation can have multiple applications running.

Figure 6

◆ INSIDE DIGITAL RESEARCH'S FLEXOS 386

* Application Environment Services (AES): Sets of functions that manage drop-down menus, windows, icons, and dialog boxes on the screen, and handle mouse and keyboard input from the user.
* Virtual Device Interface (VDI): An extensive collection of device-independent functions that manage the interface to the physical graphics devices.

These X/GEM components do not replace the computer's operating system; they supplement the operating system's functions with the AES and VDI functions. Applications use the operating system for file management and the X/GEM functions to manage user input and graphics device output.

The AES

The AES is composed of 12 function libraries. The term "library" in this context means "a set of related functions." These are called RTLs, or run-time libraries. The AES interface to the screen has two fundamental components: the menu bar and the desktop window.

Each AES library addresses a different aspect of the application programming interface:

Application: Application initialization and communication.

Event: Input and event management.

Menu: Menu bar display and management.

Object: Object tree display and management.

Form: Form display and management.

Graphics: Rectangle display and management.

Scrap: Data interchange between applications.

File Selector: Directory display and file selection.

Window: Window management.

Resource: Resource file and object address management.

Shell: Shell information retrieval and management.

Extended Graphics: Supplemental rectangle functions.

The VDI

The VDI run-time libraries provide device-independent functions for opening and closing, setting attributes for, drawing on, and getting information from graphics devices. The VDI functions complement the AES functions, each serving a different purpose. The AES calls initialize the application, manage menus and window control areas, and perform object-based operations within the work area. The VDI functions open and initialize each graphics device and output graphics and text.

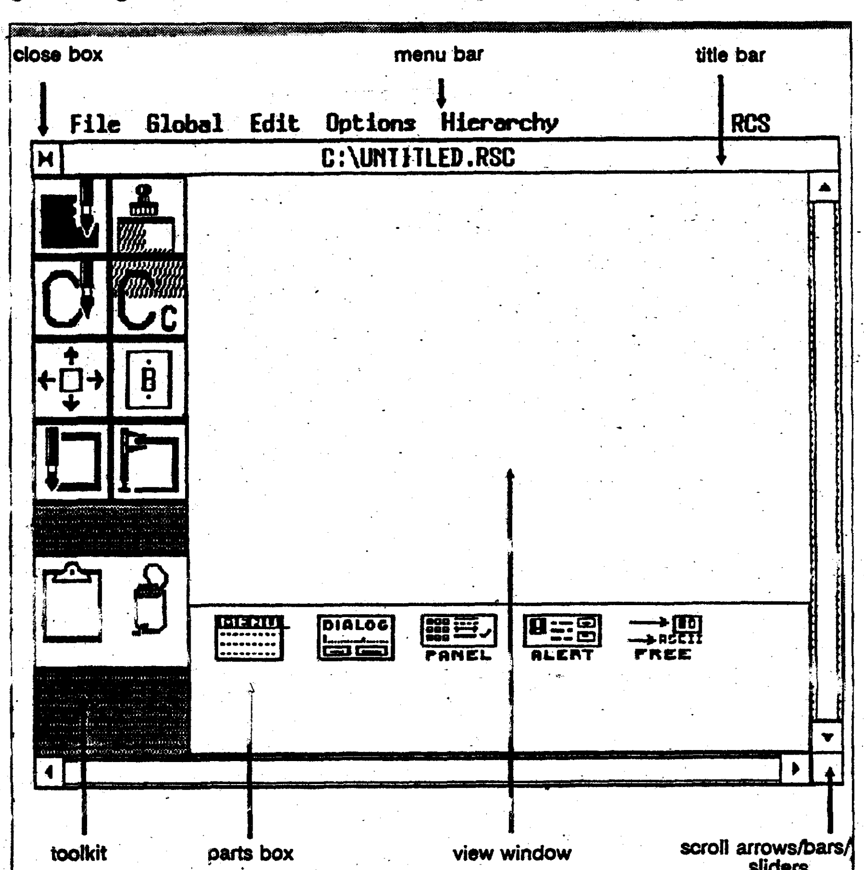


Figure 8: The Desktop Window of the Resource Construction Set.

All computer graphics are displayed using a coordinate system to reference individual points on the workstation.

The VDI provides the workstation control functions. A workstation under X/GEM is a generic term for any graphics device. Common graphics devices are a screen, a mouse, a keyboard, a graphics printer, and a plotter. Metafiles— recorded versions of an image—are treated in much the same way as graphics devices.

Each graphics device has two sets of attributes. One set consists of output attributes, such as color, line style, and text face, that you can set dynamically during program execution. The other set consists of device characteristics, such as maximum addressable width and height, pixel size, minimum line width, and maximum number of colors, that cannot be changed at run-time.

All computer graphics are displayed using a coordinate system to reference individual points on the workstation. The VDI supports two coordinate systems: normalized device coordinates (NDC) and raster coordinates. NDCs provide a standard numerical addressing scheme independent of the number of actual picture elements (pixels) supported by the device. These are more commonly known as bit-image files which have an .IMG filename extension. The address space defined by NDCs appears at the same relative location on the surface, regardless of the device.

Raster coordinates provide an addressing scheme that references locations according to their X,Y pixel coordinates. These are also called object-oriented files and carry a .GEM filename extension. The number of pixels on each axis is defined by the device driver. The VDI translates NDCs to raster coordinates before outputting point references to a device driver. Because the VDI maps the full NDC range to each axis, the aspect ratio (the ratio of the horizontal to vertical dimensions) is not 1-to-1 for devices with an unequal number of pixels on the axes. The VDI compensates for the aspect ratio by outputting the image on the screen in the same general proportions to the highest resolution of the printing device. If your display device has 640x480 VGA resolution, the output on a 300x300 dpi laser will have exactly the same aspect ratio. Therefore, the printed image will not be distorted because of the dissimilar resolution between the devices.

The final component is the DOS run-time library (DOSRTL). This is a collection of routines responsible for operating system services not covered by the AESRTL and VDIRTL. This includes the use of disk files and memory management. The DOSRTL does not depend on either the AESRTL or the VDIRTL.

When X/GEM is installed under FlexOS, it does not take complete control of the operating system as does the single-user GEM environment. X/GEM applications will run concurrently with FlexOS applica-

◆ INSIDE DIGITAL RESEARCH'S FLEXOS 386

tions. FlexOS applications can also run under X/GEM. Explaining this can become complicated, so I'm going to sift through this carefully.

You will have to suspend your conceptual understanding of how DOS or even OS/2 functions, because FlexOS/386 is rooted in minicomputer and mainframe operating systems technology and can provide a higher level of functionality on desktop 386 and 386 Multibus II computers. I expect that the new 486-based desktops running FlexOS/386 (and the expected 486-specific version) will give some classes of minicomputers a run for their money.

FlexOS/386 is rooted in minicomputer and mainframe operating systems technology.

FlexOS/386 is a 32-bit realtime protected-mode multiuser, multitasking operating system. Multitasking (see Figure 6) is performed on the host machine and at workstations through virtual consoles. Each virtual console can allocate and manage its own memory partition and run its own applications. The number of virtual consoles a host or workstation can generate is governed by application memory requirements in relation to the total amount of memory resources available on the system. (See Figure 7.)

On startup, the system defaults to the native FlexOS mode. In its present configuration, a single DOS console can be set up to run any certified DOS application that might be required in certain situations—database, spreadsheet, word processing, languages, or even single-user GEM applications.

The FlexOS system developer can then set up virtual consoles to run realtime applications (as described

◆ INSIDE DIGITAL RESEARCH'S FLEXOS 386

earlier in this series), and X/GEM applications. X/GEM applications have access to all the realtime and memory-management services available to FlexOS native-mode applications.

Since X/GEM has been designed as a multitasking environment, it doesn't require the resources of separate virtual consoles from the FlexOS host. It can spawn multiple parent-child processes through a single FlexOS console. These can be multiple GEM graphics-based applications or a combination of graphics and character-based native-mode FlexOS applications.

X/GEM processes can run in the background just as FlexOS processes can.

To run character-based applications in a graphics environment, users first have to call up the FlexOS "shell" which I mentioned previously in this article. This informs the operating system that the application to be loaded in this X/GEM window will be character-based and will not require the X/GEM graphics environment services. Multiple X/GEM processes in a single FlexOS virtual console can dynamically exchange images in a simple cut-and-paste operation. Users mark the image to be moved or copied with a "rubber rectangle" and move it to any other process running on the screen.

X/GEM processes can run in the background just as FlexOS processes can. These can be communications, data acquisition, factory automation, environmental control, process control, and other realtime applications, while a CAD/CAM applications is worked on in the foreground. Workstations can be simple PC-type terminals, or high-end graphics terminals. Additional 386s can be connected to the host machine through FlexNet, DRI's networking software for FlexOS. A FlexOS system will also communicate with Unix, OS/2, DOS, and Concurrent DOS 386; and minicomputer and mainframe systems via gateways through industry-standard protocols. A FlexOS/386-X/GEM system can be as simple or as complex as a situation requires.

The "shells" are modifiable by FlexOS developers. In addition to the FlexOS shell in X/GEM, the operating system's Window Manager is also a shell. This is a character-based interface that can be reconstructed with the FlexOS development tools to suit specific applications. There are no limits to the number of virtual consoles that can be set up on the Window Manager, except for the limits of system memory.

Resource Construction Set

The Resource Construction Set (RCS) is an application in the X/GEM Programmer's Toolkit that is used to create resources (menus, dialogs, alerts, toolkit panels, and icons) for application programs. In graphics applications, resources appear on the screen but are not actually part of the program code. Resources are kept in a separate resource file, which has several advantages:

* The resource file can be created by a non-programmer.

* The resource file can be modified or updated (again by a non-programmer) often without having to recompile the application code.

* An application can exist in different national "editions," using the same program code and different resource files for each language and nationality. Application developers can create foreign-language editions literally overnight.

* An application can operate in different machine environments, using resources generated from the same source code.

Resources are made up of objects. "Object," in this sense, is a technical term referring to a specific set of images that can appear on the screen, including empty boxes, boxes containing text, text strings, and the like. To create a resource—a menu, for example—you combine objects to form an object tree. The relationship between the objects in the tree is described in family terms: the first object is the parent; the objects contained within the parent are the children.

The RCS uses many of the techniques associated with an X/GEM Desktop application—pull-down menus, dialogs, alerts, and bit-mapped graphics editing. (See Figure 8.) The AES enforces an absolute limit of 64K for a resource file. If a larger resource is needed for a particular section of an application menu, two resource files can be chained together.

Linking the resource files into an X/GEM application is performed by RSCREATE. This is a C language utility program that has two intended purposes: to create a resource file from a hand-edited file; and to port resource files between microprocessor environments. RSCREATE uses an .RSH file as an include file. An .RSH file is an ASCII file that can be edited with a text editor or word processor. It is at this stage, prior to the compiling, that the application developer can begin a port of his GEM software to another environment. First the .RSH file is moved to the new environment. Then a header is added to make the .RSH file compatible with the target environment. If the port, for example, is to be made to a 68K environment, the header is commented out in RSCREATE. The target-format .RSH is entered in RSCREATE as an include file. Lastly, RSCREATE is compiled, linked, and executed on the target environment with Lattice "C" or another full language implementation.

The RCS is also very conducive to the creation of online help and training materials. The facilities provided in the AES (dialog boxes, alerts, etc.) are powerful tools in the construction of such materials. The KnowledgeSet knowledge-retrieval system is one example of a GEM-based application containing extensive help and training facilities—it can provide the user with up to 550 megabytes of help or training information, stored on one 5.25-inch CD-ROM disk. The KnowledgeSet group, a pioneer in CD-ROM applications and technology, also offers the Grolier Encyclopedia under a GEM-based user interface, and all the Boeing aircraft maintenance manuals, engineering specifications, and wiring diagrams cross-referenced for instant retrieval.

Update

According to a news story, Digital Research is working with IBM on a Presentation Manager implementation of X/GEM. IBM, which is DRI's largest FlexOS licensee, is interested in the potential of having the large worldwide base of GEM applications ported to PM.

While there are several dozen PM applications available, with a score of new applications scheduled to hit the market early this year, these are not enough to encourage an industry migration to Presentation Manager and OS/2. Applications sell operating systems, and under existing ground rules, creating PM applications is a long, complicated, and expensive investment for developers to make in the hope that it will pay off somewhere down the road.

With a GEM-Presentation Manager toolkit just about in place, and an X/GEM-Presentation Manager toolkit planned for a first-quarter 1990 release, DRI's technology may provide the impetus that IBM needs to make OS/2 a viable player in the operating systems market.

As the computer-industry turmoil continues its never-ending quest for the next higher levels of operating systems and graphics functionality, it will be interesting to see where X/GEM will put its roots. So far, Digital Research is moving in all directions—from FlexOS, to Unix, to OS/2's Presentation Manager, and to the Intel and Motorola CPUs.

Summing Up

I don't see the market fragmenting, by any means, but since it has grown so large, its needs have become more specialized. FlexOS/386 has become the clear choice for realtime multitasking, multiuser environments. OS/2 will gravitate into the corporate environment as the host in powerful networks. Unix will dominate in the multiuser workgroup environments.

Each provides a palpable solution to the needs of the business community. There are still major hurdles and issues to be resolved on all fronts. Digital Research has, perhaps, the best software engineering technology in place among all the dominating systems-software specialists. Their marketing skills and aggressiveness have yet to become as highly focused. Perhaps the focus they have had with FlexOS will filter down to their other operating-system product groups.

Microsoft, on the other hand, is a marketing-driven company that is highly successful with its general-purpose software, but its systems-software group cannot come to grips with the technology, promising more than it delivers, and frequently has to buy technology or form alliances to keep a product alive.

Unix, highly modular in structure and portable to diverse hardware platforms, lacks a consistent user-interface and operating environment. Although it was ostensibly the purpose of the OSF to unify the user-interface, this has been slow to materialize. There are also too many implementations of the systems software on the market that prevent applications for one platform from running on another.

Within this murky sphere is a conglomeration of networking standards, graphics file formats, communications protocols, programming toolkits, and a rash to kludge kits that attempt to bridge some of the incompatibilities. Yes, 1990 should be an interesting year.

For information about the FlexOS/386 and X/GEM Programmer's Toolkits and System Builder's Kits, contact Digital Research, Inc. at Box DRI, 70 Garden Ct., Monterey, CA 93942 (1-800-443-4200), or call their North American sales office at (408) 982-0700.

Peter Ruber is a freelance writer and consultant.