CHAPTER 21

What Makes a Great Windows 95 Application?

Although MS-DOS–based and Win16–based applications can run under Windows 95, users benefit from the additional functionality supported by Win32–based applications. This functionality includes the preemptive multitasking architecture of Windows 95 and the increased robustness and protection for running applications. In addition, the following six key contributors make a Windows 95-based application great from the user's perspective:

- The Win32 Application Programming Interface (API)
- OLE functionality
- The *Windows 95 User Interface Design Guidelines*
- Support for handling Plug and Play events
- Support for quickly identifying files
- Adherence to common setup guidelines for consistent software installation

The next section discusses why these components make these applications great for users.

The Win32 Application Programming Interface

Microsoft supports the use of the Win32 API on three operating system platforms: Windows NT, Windows 95, and Windows 3.1 with Win32s. Each operating system supports a common set of Win32 APIs, allowing applications developed using a single API set to run on multiple platforms. As a result, application developers and corporate developers can learn a single API set and leverage development resources to support a broad base of hardware systems. Users benefit from being able to run the same application on multiple platforms and from increased system reliability under Windows 95 because of the improved robustness and memory protection available to 32-bit applications.

Windows 95 delivers a robust and powerful 32-bit platform on which 32-bit applications are preemptively multitasked, run in private address spaces, and can spawn multiple threads of execution. Preemptive multitasking ensures excellent system responsiveness, allowing users to run multiple applications simultaneously and integrate personal productivity and business-critical applications in a smooth manner. (This model is similar to the one used by Windows NT.) The use of a private address space for each Win32–based application ensures that multiple applications can run simultaneously without interfering with each other or the operating system itself. Windows 95 is able to provide

smooth, preemptive multitasking and protected virtual memory because it is based on a redesigned 32-bit protected-mode kernel and a 32-bit protected-mode driver model.

Running 32-bit applications under Windows 95 provides the following improvements from a user perspective:

- Running multiple applications is smoother because of the preemptive multitasking architecture.
- Overall system performance is improved because of 32-bit operating system components.
- Robustness and system reliability are improved because of 32-bit memory protection and separate message queues.
- Applications have new functionality because of Win32 and other operating system services.
- File manipulation is easier because of long filename support.

OLE Functionality

Users are buying and using more applications per PC than ever before. In 1992, InfoCorp reported that the average number of applications purchased per desktop running the Windows operating system increased to more than seven programs, up from an average of 3.4 programs for customers using the MS-DOS operating system in 1986. Users who learn one Windows–based application find learning a second or third application easy, and research shows that users cite the ability to move and share information among applications as the most important reason for using Windows–based applications.

People are not only acquiring more applications, but they are also using them together, accessing several applications in order to create compound documents. The mechanism that allows applications to interoperate effectively, and thereby enables users to work more productively, is OLE. Users of OLE applications can create and manage compound documents that seamlessly incorporate data, called *objects*, of different formats. Sound clips, spreadsheets, text, and bitmaps are some examples of objects commonly found in compound documents. Each object is created and maintained by its server application, but through the use of OLE, the services of the different server applications are integrated. Users of OLE-enabled applications feel as if they are working with a single application that has all the functionality of each of the server applications. They don't need to be concerned with managing and switching between the various server applications. Instead, they can focus on the compound document they are creating and the task they are performing using OLE-based features.

The Features of OLE

With OLE, Windows 95 increases the degree of application integration available to any application that takes advantage of the services. Tight integration gives users tangible benefits, allowing them to share data and functionality across applications and combine the data as they please. Because OLE is based on an open industry standard, users can extend their applications with additional third-party products, further expanding their choices and flexibility.

OLE provides the following features to allow users to easily combine information from multiple applications:

- **Cross-application drag and drop.** Users can drag and drop graphs, tables, and pictures directly onto slides, worksheets, and documents to mix text, data, and graphics into compound documents.
- **Visual editing.** Users can double-click an object to directly edit it while remaining in the original document.

Cross-Application Drag and Drop

Drag and drop is a new, intuitive method of moving data between applications. Until recently, this method was available only for moving information *within* applications. The most widely used way of transferring data *between* applications has been to use the Clipboard, but this method involves multiple steps—using the Copy or Cut command, moving to the destination application, and using the Paste command. With the current release of OLE, drag and drop now works between applications. Users simply select an object in one application, drag it to its destination in another application, and drop it into place. Objects can also be dragged over the desktop to system resource icons, such as printers and mailboxes, making the sending, printing, and sharing of files faster and easier.

Visual Editing

Visual editing makes revising a compound document faster, easier, and more intuitive. For example, Figure 129 shows a Microsoft Excel worksheet embedded as an object in a Word document. When the user double-clicks the worksheet object, the menus and toolbars necessary to interact with the Microsoft Excel worksheet temporarily replace Word's menus and controls. Microsoft Excel, the application that is needed to edit or modify the worksheet, partially "takes over" the Word document window, as shown in Figure 130. The user can then interact with the Microsoft Excel worksheet without switching to a different application or window. (Unlike the first release of the OLE technology, the current release of OLE does not launch users into a separate Excel window to work on the spreadsheet data.) When users move on to work on the word processing portion of the document, the focus returns to Word, and the original Word menus and controls are restored.

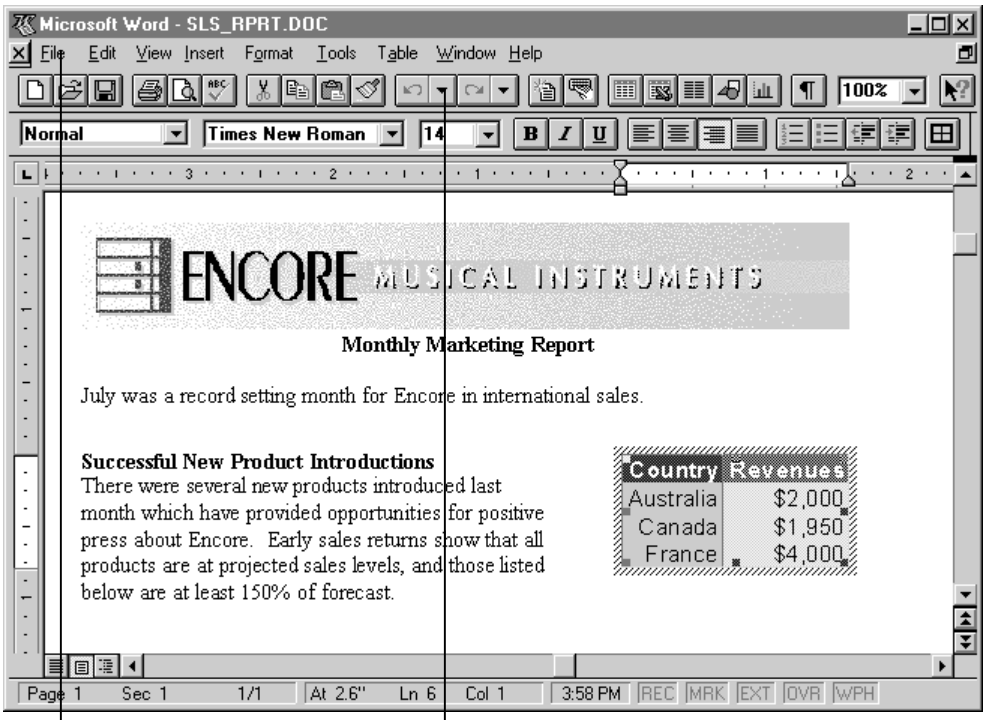

Figure 129. A Microsoft Excel worksheet object embedded in a Word document

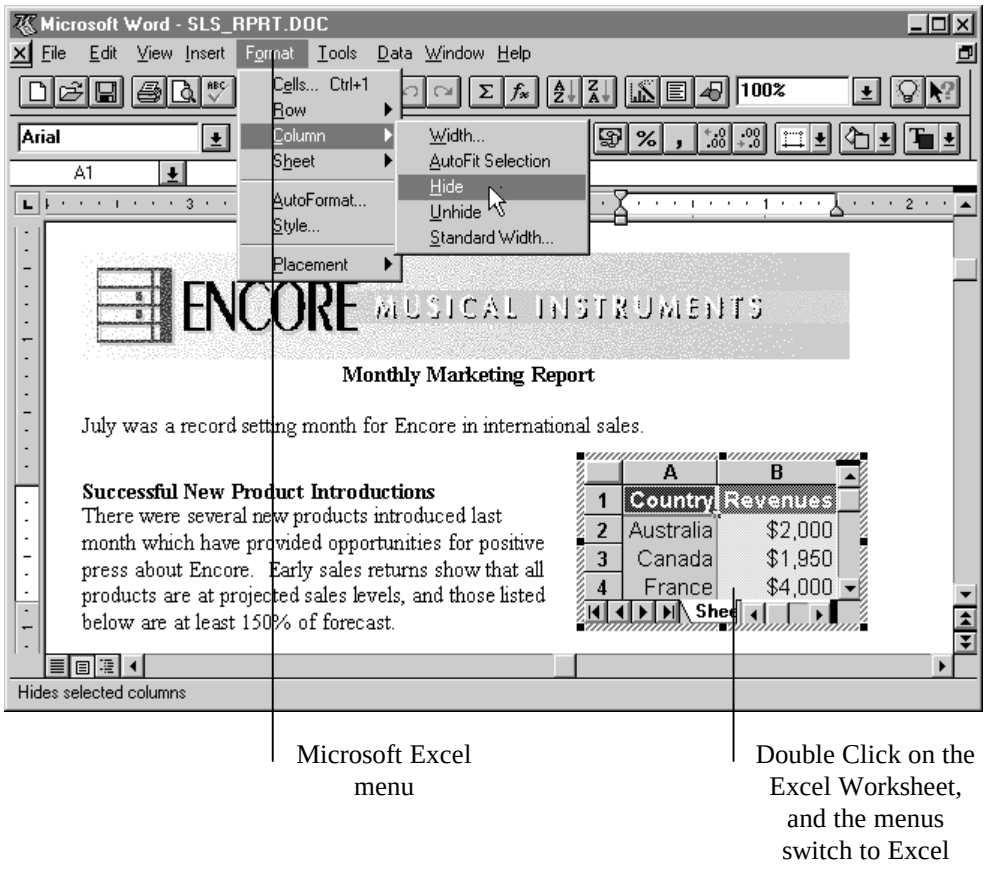

Figure 130. Activating the Excel worksheet object, which displays the Excel menus in the Word environment

The advantage of visual editing is even greater when compound documents include large numbers of objects created by different applications, such as Microsoft Excel worksheets and charts, PowerPoint graphics, sound, video clips, and so on. Instead of switching back and forth among different windows to update the objects, users can work in a single document window, which provides one location for editing and otherwise interacting with the data. Visual editing thus offers users a more document-centric approach, putting the primary focus on the creation and manipulation of information rather than on the operation of the environment and its applications.

The *Windows 95 User Interface Design Guidelines*

As with previous versions of Windows, one of the reasons that Windows 95 applications are easy to learn is the fact that they look and act alike. Microsoft has taken great steps to improve the basic common controls that all applications can share. These controls, which have evolved based on user feedback and extensive usability testing, are among the features described in the *Windows 95 User Interface Design Guidelines.*

Applications that use the basic controls provide their users with common, improved features, such as being able to create new folders in the Save As dialog box without having to switch to the Windows Explorer or File Manager (see Figure 131).

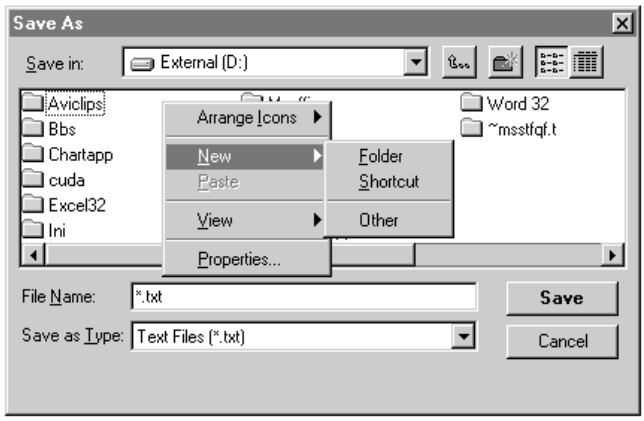

Figure 131. The Save As common dialog box

The new Printer property sheet, shown in Figure 132, illustrates some of the controls that make accessing features easier for users. At the top of the property sheet are tabs on which different categories of properties are arranged. Clicking any tab displays that category of properties. Figure 132 also shows an example of a spin control, which increments and decrements the number in the Copies box.

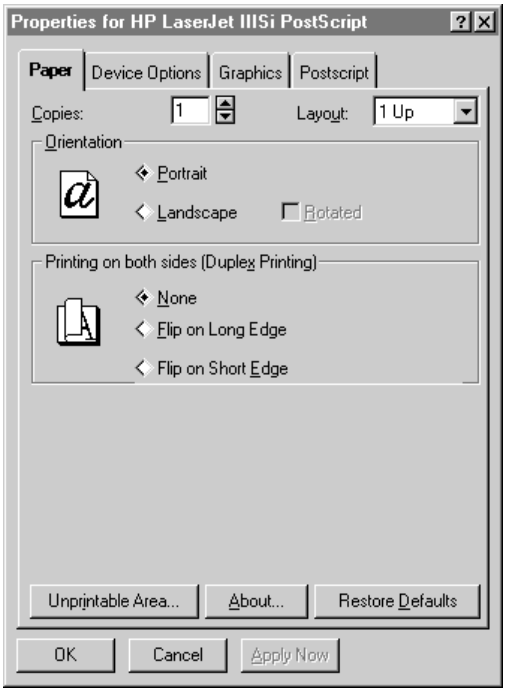

Figure 132. The tabbed Printer property sheet

The new Open common dialog box, shown in Figure 133, allows users to see long filenames and navigate around the entire PC or network to look for the files they want to open. As shown in Figure 134, this dialog box uses tree lists to display the hierarchy of the PC's hard disk and the network to which the PC is connected.

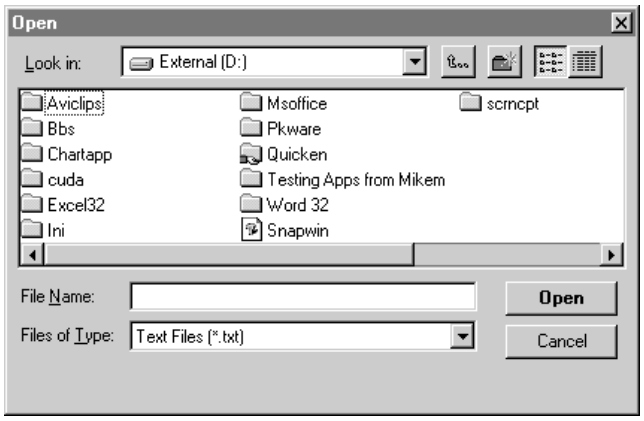

Figure 133. The Open common dialog box

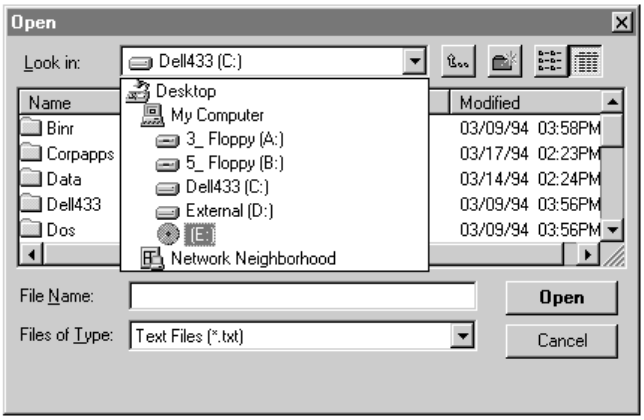

Figure 134. The tree list in the Open common dialog box

Figure 135 shows another tree list control that makes viewing and accessing hierarchical information easier. This tree list is found in the property sheet for the Device Manager in the Control Panel's System tool. As users expand and collapse the tree, they can see information relevant to their chosen topic.

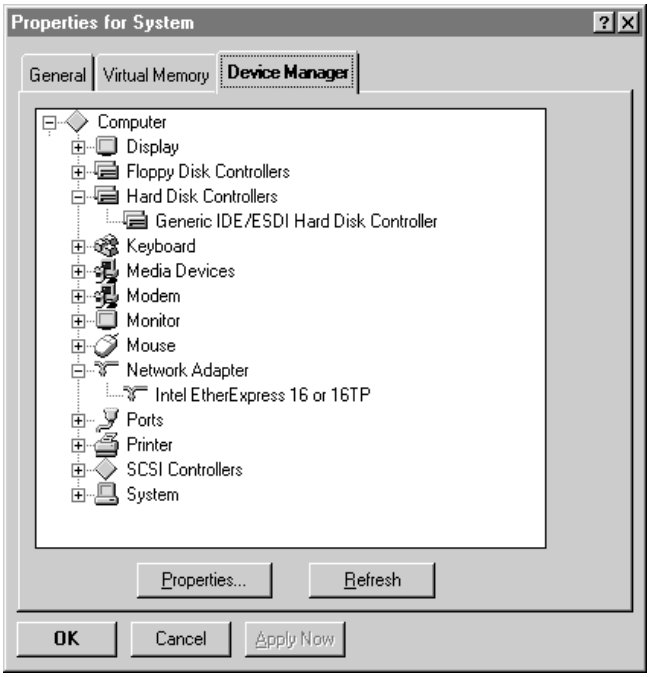

Figure 135. The tree list in the Device Manager property sheet

Applications no longer have to include their own custom slider controls. Figure 136 shows the Mouse property sheet, where the pointer's Size option is controlled by the slider control included in Windows 95.

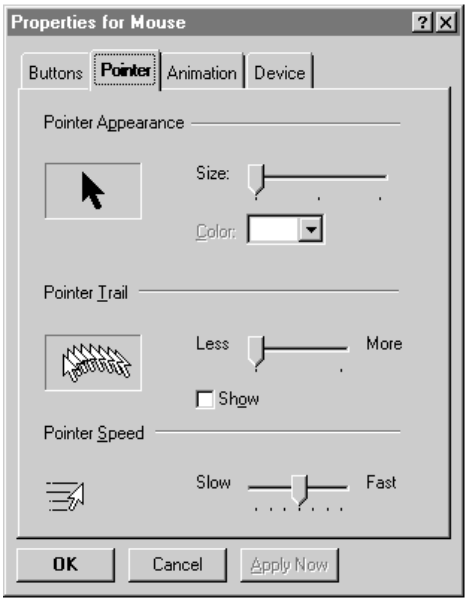

Figure 136. A slider control in the Mouse property sheet

Many new common controls are included in Windows 95, such as toolbars, the status bar, column headings, tabs, sliders, progress indicators, rich-text controls, list views, and tree views. Great Windows 95-based applications will use these controls to make user access

to features consistent across applications, thereby making the entire system much easier to use.

Support for Handling Plug and Play Events

Applications that provide Plug and Play event awareness help users by seamlessly adapting to hardware configuration changes. Users reap the following two key benefits:

- **Applications automatically recognize and respond to hardware changes.** Consider this scenario: A user has a mobile PC installed in a docking station and is using an external monitor running at a resolution of 1024x768. When the user undocks the PC, the desktop Control Panel recognizes this action and switches the resolution for the mobile PC to 640x480. When this change occurs, Plug and Play–aware applications resize their windows and toolbars accordingly. The user doesn't have to do a thing; it's all automatic.

  Here's another scenario: A user is working on a document on a mobile PC. Battery power is running low. The computer sends a message to all the active Plug and Play–aware applications, telling them to save the user's data and shut down because the power is going off in a few minutes.

- **Applications warn users about open network files when hot-undocking their computers.** Consider this scenario: A user has a PCMCIA network card installed in a laptop computer. Before leaving the office, the user switches PCMCIA cards and installs a modem for dial-up network access. With Plug and Play, the user doesn't have to fuss with software configuration. Windows 95 simply knows that the network has been replaced by a modem and passes this information on to a Plug and Play–aware e-mail application. The application then knows that it now needs to use the modem to make connections.

Applications that are Plug and Play–aware provide seamless adaptation to changes in the hardware configuration so that users can focus on their work, not their configuration.

Support for Quickly Identifying Files

As discussed in Chapter 3, "The Windows 95 User Interface," the UI for Windows 95 is much improved. But the UI itself is only part of the benefit for users. Applications that take advantage of the new UI support can offer their users long filenames and direct file viewing. Long filename support means that document names are no longer limited to eight characters; they can now have up to 255 characters, as shown in Figure 137. Instead of 23ISM_JB.DOC, users can name a file *Status report July 23 regarding the ISM project for my boss Jim Bernstein*—a title that clearly identifies the document's contents. Applications that support the Quick View capabilities of Windows 95 provide users with a quick and easy way to identify files by viewing them directly from the UI without launching the applications that created them.

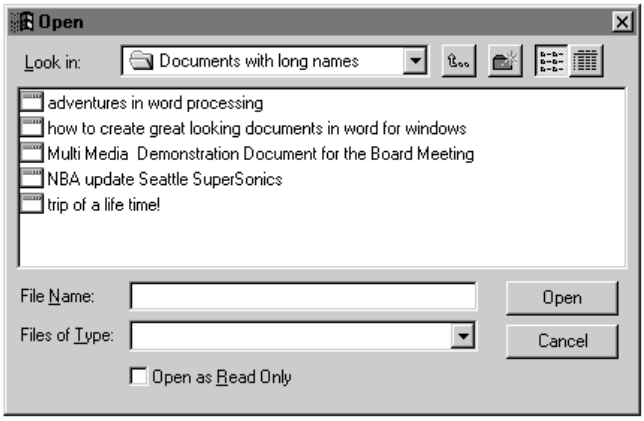

Figure 137. Sample long filenames supported by Windows 95

Consistent Setup Guidelines

In the past, users have generally had an easy way to set up new applications, but removing applications from their hard disks was not so simple. Most users of Windows 3.1 eventually ended up with hard disks clogged with files that were never used because they belonged to a deleted application. Because many applications use the same library files, users of Windows 3.1 quite commonly had several copies of a file stored in different places on their hard disks—an inefficient use of precious disk space. Another common problem with Windows 3.1 was that applications put files not only in their own directories, but also in the Windows directory, in the System directory, and even in the root directory, creating a nightmare for users who were trying to keep track of what was where.

The *Windows 95 Software Development Kit* offers some guidelines to developers for consistent installation locations and uninstall functionality in their applications. Common libraries can be shared by applications, thereby reducing the amount of disk space consumed by duplication. The guidelines also set standards for where developers should put files on the hard disk so as to provide an easy, powerful, and compatible structure for users. Setup programs that follow the guidelines all operate similarly, use consistent naming conventions, and offer the same setup options, thereby reducing the learning curve for users, improving manageability and support for corporations, and increasing the efficiency of remote administration of installed software.