

Going Further

If you've made it this far, you've learned quite a bit about putting a Macintosh application together. You've seen how to create and manage menus, windows, dialog boxes, and preference files. You know how to get information about the user's actions, and you know how to respond to many of those actions. You also know, at least in overview, how your application shares the available system resources with the Operating System and other open applications. Congratulations; that's a lot to learn in less than 200 pages.

No doubt, however, you want to learn more. The Venn Diagrammer application fails to implement a number of very fundamental elements of a typical Macintosh application. It provides no text-input or editing capabilities, no support for user drawing, no support for color, and virtually no support for the many important features introduced in System 7. The following section briefly describes some of these capabilities and refers you to the *Inside Macintosh* books that give more information about implementing those capabilities.

This afterword also provides some hints on writing your application so that it is compatible with all existing Macintosh computers and system software versions and so that it can be easily localized to different languages. This afterword ends with a list of additional developer services provided by Apple Computer, Inc.

Implementing Further Features

Venn Diagrammer succeeds in its basic goal, which is to illustrate how to implement many of the essential user interface components of a typical Macintosh application and to introduce the very simplest features of the Operating System. It shows how to do basic drawing in a window and how to handle many user actions. Best of all, it's a real application that does useful, albeit limited, work.

It's important to realize that although some parts of the source code presented throughout this book are purposely simplified, other parts are not. The code for handling dialog boxes, for instance, is designed to be easily amplified to handle other modeless dialog boxes. The basic event loop and the menu-handling code are also quite typical of what you'd find in a commercial Macintosh application. The Venn Diagrammer source code is not intended as a shell on which to base your application, but chances are you'll do at least a few things in the same way.

Still, the Venn Diagrammer source code fails to illustrate how to implement a number of important Macintosh features. Here's a moderately complete list of what's missing and where you can look to get the information you need to add these features to your application:

- **Windows.** The document windows created by the Venn Diagrammer application are of fixed size, so they don't need to contain zoom boxes, size boxes, or scroll bars. In all likelihood, however, your application will allow the user to enter and edit information (such as text or graphics) that will usually not fit in a fixed-size window. As a result, you will probably want to include support for these window elements. To learn how to handle zoom and size boxes, see the chapter "Window Manager" in *Inside Macintosh: Macintosh Toolbox Essentials*. To learn how to implement scroll bars, see the chapter "Control Manager" in that same book.
- **Menus.** The Macintosh system software provides support for several kinds of menus in addition to the standard "pull-down" menus used by the Venn Diagrammer application. A very useful adaptation of the pull-down menu is the *pop-up menu*, which you can put in dialog boxes and document windows. Moreover, both pop-up menus and pull-down menus can contain *hierarchical menus*, where an entire menu is attached to a menu item. For information about these additional kinds of menus, see the chapter "Menu Manager" in *Inside Macintosh: Macintosh Toolbox Essentials*. That chapter also shows how to modify a menu item's text and style, how to add a mark to a menu item, and how to associate an icon with a menu item. Because pop-up menus are actually very complex controls, you'll also need to read the chapter "Control Manager" in *Inside Macintosh: Macintosh Toolbox Essentials* to learn how to handle pop-up menus.
- **Text.** Most Macintosh applications support some form of text entry and editing, even if just to solicit some piece of information from the user in a dialog box. The system software includes TextEdit, which you can use to provide basic text-handling capabilities for your application. Although TextEdit was originally designed to handle edit fields in a dialog box, you can also use it for other purposes. For example, if you're writing a spreadsheet application, you might use TextEdit to handle small amounts of text. TextEdit is not, however, suitable for large amounts of text (greater than about 32,000 characters). If you're writing a word-processing application, you'll need to write your own custom text-handling routines. To learn how to handle text entry and editing in dialog boxes, see the chapter "Dialog Manager" in *Inside Macintosh: Macintosh Toolbox Essentials*. To learn how to use TextEdit directly, see the chapter "TextEdit" in *Inside Macintosh: Text*. This latter book also describes a number of other text-related facilities provided by the Macintosh system software, such as support for multiple fonts and non-Roman character sets.
- **Files.** The Venn Diagrammer application can create, read, and write resource files only (which contain the user's preferences). Most applications allow the user to create and edit information of arbitrary size, and they store that information in a file's data fork. The data fork can contain any kind of information you care to put there. You read and write data from a file's data fork using the File Manager, and you present the

standard user interface for opening and saving files using the Standard File Package. The chapter “Introduction to File Management” in *Inside Macintosh: Files* shows how to use these and other services to implement the typical File menu commands (Open, Save, Save As, Revert, and so forth). Other chapters in that book provide more detailed information about the structure of the file system used on Macintosh computers and about the system software managers you can use to manipulate objects in the file system. For more complete information on reading and writing resource files, see the chapter “Resource Manager” in *Inside Macintosh: More Macintosh Toolbox*.

- **Icons.** To learn how to define icons for your application and its document files, see the chapter “Finder Interface” in *Inside Macintosh: Macintosh Toolbox Essentials*.
- **Help.** Every application should include the resources necessary to allow the Help Manager to display help balloons after the user has chosen the Show Balloons command from the Help menu. Usually you can add support for help balloons simply by adding resources to your application’s resource fork, without having to change or recompile its source code. In some cases, however, you might also need to modify the source code to provide help balloons. For complete details on implementing help balloons, see the chapter “Help Manager” in *Inside Macintosh: More Macintosh Toolbox*.
- **Printing.** One of the easiest features to add to the Venn Diagrammer application is the capability to print a Venn diagram window. Printing essentially involves just drawing the window into a special graphics port called a printing graphics port. Before doing that, however, you need to present the standard dialog boxes to set up a page and to send a print job to a printer. If, as is usually the case, there are multiple pages to be printed, you’ll want to structure your printing code into a printing loop. A complete printing loop is provided in the chapter “QuickDraw Printing Manager” in *Inside Macintosh: Imaging*. That chapter also shows how to handle a number of other printing-related tasks.
- **Memory.** The Venn Diagrammer application is surprisingly naive in its management of the memory in its own partition. For the most part, it simply tries to allocate the memory it needs for some particular operation, and if it fails to get that memory, it just does the safest thing it can to work around that failure. You’ll want to implement a much more robust scheme to manage the memory you’re allocated when your application starts up. You need to make sure that your application’s memory requirements don’t consume too much of your partition, because many system software routines (especially many QuickDraw routines) also use memory in your application partition. For a simple but effective memory-management strategy, see the chapter “Introduction to Memory Management” in *Inside Macintosh: Memory*. For some advice on how to segment your application’s executable code to minimize its memory footprint, see the chapter “Segment Manager” in *Inside Macintosh: Processes*.

- **Interapplication Communication.** To take full advantage of the cooperative multitasking environment provided in system software versions 7.0 and later, your application should be able to communicate effectively with other open applications. The system software provides several ways in which you can interact with other applications. You can support the publish and subscribe capabilities of the Edition Manager (described earlier in “Interapplication Communication” beginning on page 14) and you can support high-level events such as Apple events. For complete details on how to communicate and share data with other applications, see the book *Inside Macintosh: Interapplication Communication*.
- **Sound.** You can enhance the perceived quality of your application by appropriately including sounds in its user interface. When, for example, the user asks the Venn Diagrammer application to check the user’s diagram, the application might play some agreeable sound if the diagram is correct and some discordant sound otherwise. Sound can provide user feedback that is not achievable using text and graphics alone. Other applications are more directly involved with recording or producing sound. To learn how to add sound capabilities to your application, see the chapter “Introduction to Sound” in *Inside Macintosh: Sound*.
- **Color.** Like sound, color might be either an enhancement to or a fundamental feature of your application. For example, Venn Diagrammer might allow the user to fill empty regions with colored patterns. You can use QuickDraw to draw shapes, regions, and even text in any color supported by the available video devices. For complete information on supporting color in your application, see the appropriate chapters in *Inside Macintosh: Imaging*.

IMPORTANT

You don’t have to read all of the books mentioned in this list to develop a Macintosh application. Which of the many *Inside Macintosh* books you’ll need depends on the particular requirements of your application. (The Venn Diagrammer application, for instance, draws mainly on four books only: *Inside Macintosh: Macintosh Toolbox Essentials*, *Inside Macintosh: More Macintosh Toolbox*, *Inside Macintosh: Memory*, and *Inside Macintosh: Imaging*.) Moreover, you don’t necessarily have to read all of a chapter to get started using a certain manager. Most chapters in *Inside Macintosh* contain advanced material that is likely to be of interest only to developers with very specialized needs. ▲

Maintaining Compatibility

Compatibility is the ability of an application to execute properly in different operating environments. Compatibility is important if you want to write software that runs, with little or no modification, on all members of the Macintosh family and in all system software versions.

The key to achieving compatibility is not to depend on things that may change. *Inside Macintosh* contains numerous warnings about which information is likely to change. As the Operating System and Toolbox evolve to accommodate the needs of developers and users, many of their elements will vary. Whenever possible, Apple Computer strives to add features without altering existing programming interfaces. In general, you can assume that Operating System and Toolbox routines are less likely to change than data structures. Therefore, you should never directly manipulate data structures that are internal to a manager or system software routine, even if their structure is documented. Instead, you should manipulate those structures only indirectly, by calling Operating System and Toolbox routines that achieve the desired effect. In particular, you should never alter any portion of a data structure marked as unused or reserved.

Another key to writing compatible code is to code defensively. Do not assume that users perform actions in a particular order, and do not assume that function and procedure calls always succeed. You should always test the return values of routines for errors, as illustrated in most of the code samples presented in this book.

Here are some more specific guidelines to keep in mind as you write your application:

- Never address hardware directly; whenever possible, use the routines provided by the various device drivers and managers to send data to the available hardware. The addresses of memory-mapped hardware are always subject to change, as is the hardware itself. More important, direct access to such hardware is not possible in every operating environment. In multi-user systems like A/UX, for instance, the operating system manipulates all hardware; applications simply cannot write directly to hardware addresses.
- Avoid writing directly to the screen. Use QuickDraw routines whenever possible to draw on the screen. If you absolutely must write directly to the screen, do not assume that the screen is a fixed size or that it is in a fixed location. The location, size, and bit depth of the screen differ in various machines.

- Don't rely on system global variables. Many of these variables are documented in *Inside Macintosh*, but many are not. In particular, you must avoid undocumented system global variables because they are most likely to change. But you should try to avoid even well-known system global variables because they may not be available in all environments or in the future. In general, you can avoid using system global variables by using available routines that return the same information. (For example, the `TickCount` function returns the same value that is contained in the system global variable `Ticks`.)

Making Your Application Localizable

Localization is the process of adapting an application to a specific language, culture, and region. By planning ahead and making localization relatively painless, you'll ensure that your product is ready for international markets in the future. This section provides a brief overview of what you need to do to make it easy to localize your application. For the complete account of writing software that is compatible with Macintosh computers throughout the world, see *Inside Macintosh: Text and Guide to Macintosh Software Localization*.

The key to easy localization is to store region-dependent information used by your application as resources (rather than within the application's code). Text seen by the user can then be translated without modifying the code. In addition, storing such information in resources means that your application can be adapted for a different area of the world simply by substituting the appropriate resources. Make sure that at least the following kinds of information are stored in resources:

- all text, including special characters and delimiters
- menus and keyboard equivalents for menu commands (if available)
- character, word, phrase, and text translation tables
- address formats, including zip codes and telephone numbers

When you create resources for your applications, remember the following key points:

- text needs room to grow (up, down, and sideways)
 - translated text is often 50 percent larger than the U.S. English text
 - diacritical marks, widely used outside the United States, may extend up to the ascent line
 - some system fonts contain characters that extend to *both* the ascent and descent lines
- text location within a window should be easy to change

Using Developer Services

In addition to the *Inside Macintosh* library of books, Apple Computer provides a number of other services that you can use to learn more about programming for Macintosh computers and simplify your software development process. Apple's goal in making these services available is to provide you with the resources you need to create outstanding Macintosh applications. These services include

- books and other technical publications
- programming languages and tools
- programming classes and self-paced training materials
- conferences and workshops
- technical support

Most of these products and services are available to anyone interested in programming for Macintosh computers. You can get information about them by contacting APDA, Apple's source for developer tools. See the Preface (page xv) for details on contacting APDA.

Some of the services just listed—in particular, technical support and invitations to some developer conferences and workshops—are provided only to members of the Apple Associates and Partners Program. For information about Apple's support programs for commercial developers, call the Developer Hotline at (408) 974-4897. These programs are available to developers in the United States and Canada only.

Technical Publications

Apple provides a number of technical publications that can assist you in writing Macintosh applications. Here's a brief description of three books that you'll probably need right now:

- *Macintosh Human Interface Guidelines*. A complete description of the Apple Desktop Interface and an indispensable set of guidelines governing the appearance and behavior of Macintosh applications. You will need this book to ensure that your application conforms to those guidelines.
- *Technical Introduction to the Macintosh Family*. A general introduction to the family of Macintosh computers, with emphasis on the features that make it a desirable platform for application developers. This book also provides details on Macintosh hardware and on A/UX, Apple's version of the UNIX[®] operating system.

- *Guide to Macintosh Software Localization*. A guide to the process of localizing application software for Macintosh computers around the world. You'll want to read this book for essential information about making your product marketable worldwide.

If you are an Apple Associate or Partner, you'll automatically receive a subscription to *develop*, *The Apple Technical Journal*. This magazine is intended to complement other reference materials like *Inside Macintosh*. It doesn't try to replace or reword those books; instead, it's designed to help you understand them by illustrating some of the techniques they describe. For subscription information, contact

develop

Apple Computer, Inc.

P.O. Box 531

Mount Morris, IL 61054-7858

Telephone 800-877-5548 (United States)
815-734-6309 (All other countries)

Fax 815-734-4205

AppleLink DEV.SUBS

Training

Apple Developer University offers a broad range of Macintosh programming instruction through hands-on classes and self-paced training products. Classes are offered in Cupertino, at Apple training facilities worldwide, on an on-site basis, and through selected third-party University and Corporate trainers.

Developer University provides expert instruction for all levels of Macintosh programmers. These course teach programmers to produce fast, efficient code that takes maximum advantage of the Macintosh Toolbox and Operating System.

Apple Developer University is open to all individuals worldwide who have an interest in mastering leading-edge technology. To reserve your place in a class, schedule an on-site training class, or for more information, contact

Apple Developer University Training Registrar

Apple Computer, Inc.

20525 Mariani Avenue

M/S 75-6U

Cupertino, CA 95014

Telephone 408-974-6215 (United States)

Fax 408-974-0544

AppleLink DEVUNIV

Technical Support

If you are an Apple Associate or Partner, you'll have access to various levels of technical support from Apple. Both Associates and Partners receive monthly mailings that include a newsletter, Apple II and Macintosh Technical Notes, pertinent Developer Programs information, and the latest news relating to Apple products. Mailings also usually include the latest developer CD-ROM, which contains system software, programming utilities, code samples illustrating how to use various parts of the Macintosh system software, and the latest on-line technical documentation.

In addition, Apple Partners receive discounts on Apple equipment and technical assistance from the staff of Apple's Developer Technical Support department.

