Chapter 4

User Interface Tools

This chapter describes the user interface libraries and toolkits you can use to create a graphical interface between your application and its users. Refer to Table 2-3 for a list of the manuals you can read to learn more about the topics discussed in this chapter.

X Window System

A window system allows a user to run several tasks at the same time and to view and control each of these tasks from a separate window. Silicon Graphics implements the X Window System, a hardware- and operating system-independent windowing system. The X Window System is a portable, network-based windowing system whose portability allows you to create applications that can run on many different workstations. You can compile and execute portable X application code in IRIX without modification.

The Silicon Graphics native X Window System is based on the X Version 11 Release 6 (X11R6) standard. The IRIX extensions to X include

- support for multiple visuals
- overlay windows
- the Display PostScript extension
- the Shape Extension that supports non-rectangular windows
- the X Input Extension that supports devices other than keyboard and mouse
- the Shared Memory Extension
- simultaneous display on multiple graphics monitors
- support for OpenGL
- PEX Version 5.1 supports 3D graphics

Xlib and Xt

The X library, known as *Xlib*, is the C-language programming interface to the X Window System. It is the lowest level programming interface to X. You can use *Xlib* to build a graphical user interface for your application, although most application developers choose higher level tools.

It's difficult to build applications with a graphical user interface that uses a low-level library such as *Xlib*. The *Xt* Intrinsics library (or simply *Xt*) simplifies this task by providing a library of C language routines designed to facilitate the interface to Xlib. Xt is a standard established by the X Consortium that provides an object-oriented programming style in the C language. Xt routines can be used to create interface components called widgets. OSF/Motif, for example, uses Xt to create its widget set (see "OSF/Motif and IRIS IM").

Silicon Graphics implements Xlib and Xt as defined by the standard and supports them as dynamic shared libraries.

4Dwm

The IRIS Extended Motif Window Manager, *4Dwm*, is an X Window System client based on *mwm*, the Motif Window Manager. In addition to the standard functions found in *mwm*, *4Dwm* provides:

- support for multiple screens
- session management functions that support the Indigo Magic desktop
- support for the desk overview feature of the Indigo Magic desktop, which allows the user to organize windows into related groups called desks
- control of background images on desks, which switches the screen background automatically when the user switches desks
- communication between the application and *4Dwm* using the *tellwm* program

For more detailed information about *mwm* and *4Dwm*, read the *mwm* and *4Dmw* reference pages.

User Interface Toolkits

Silicon Graphics provides developers with several user-interface toolkits that simplify the development of a graphical user interface. These toolkits supply a set of objects that appear in graphical form on the screen and allow users to interact with an application by manipulating these objects.

OSF/Motif and IRIS IM

The industry-standard OSF/Motif library provides user-interface objects (called widgets) to be used with Xt. The objects defined by OSF/Motif include menus, scrollbars, dialog boxes, and command buttons.

IRIS IM is Silicon Graphics' port of OSF/Motif for use on Silicon Graphics IRIS workstations. Figure 4-1 shows the relationship of an application to the various user interface libraries you might choose to use.

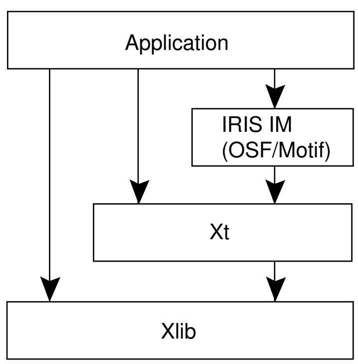

Figure 4-1 Hierarchy of User Interface Toolkits

The application programming interface (API) to IRIS IM is the same as that for OSF/Motif. However, Silicon Graphics has modified the appearance of some IRIS IM widgets so that they conform to the Silicon Graphics user interface style. You can select either the OSF/Motif or the IRIS IM widget set for use with your application, although using IRIS IM widgets is preferable because their appearance conforms to the Indigo Magic style.

In addition to the low-level building blocks such as buttons and scrollbars that IRIS IM provides, it also supplies some related software that isn't part of the standard OSF/Motif. This includes the GLwMDrawingArea widget for IRIS IM programs that use OpenGL to draw to a window within an IRIS IM application.

IRIS ViewKit

IRIS ViewKit is a C++ toolkit that provides user interface facilities for applications. It defines a collection of high-level components that you typically must implement in all applications, for example, components such as windows, menus, and dialogs. IRIS ViewKit components are designed to implement as many commonly used features as possible. To build a user interface component for your application, you can create a subclass of an IRIS ViewKit component and define any application-specific behavior. Also, with ViewKit classes as a base, you can create your own library of reusable components.

IRIS ViewKit is based on IRIS IM and uses IRIS IM widgets to implement all of its user interface components. You can make IRIS IM and X calls directly from your IRIS ViewKit application, but using ViewKit components directly is simpler and faster than creating your own components from low-level widgets. Figure 4-2 shows the relationship of an application to IRIS ViewKit and the libraries upon which it is built.

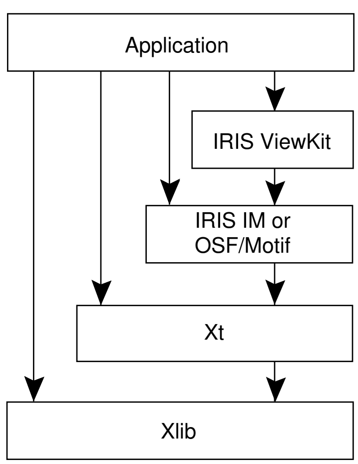

Figure 4-2 IRIS ViewKit in the Developer Environment

An IRIS ViewKit component is a C++ class that encapsulates sets of widgets and methods for their manipulation. Figure 4-3 shows a graphical representation of a portion of the ViewKit class hierarchy.

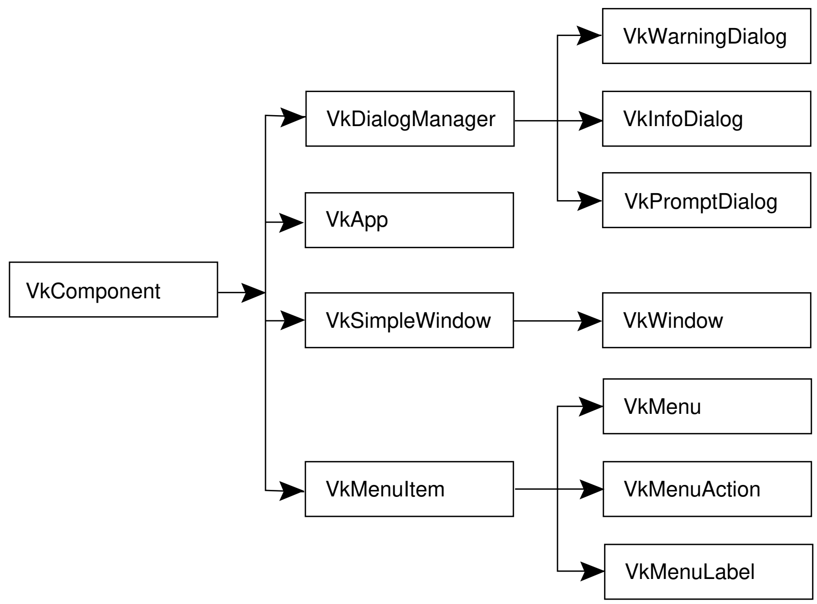

Figure 4-3 A Portion of the IRIS ViewKit Class Hierarchy

The functions of some of these IRIS ViewKit base classes are:

- VkComponent—This abstract class defines the basic structure and protocol for all ViewKit components.
- VkDialog Manager—This is the base class for specific dialog classes.
- VkApp—This base class handles application-level tasks such as Xt initialization, event handling, window management, and cursor control.
- VkSimpleWindow—This base class implements a simple top-level window (one that does not require a menu bar).
- VkMenu—This base class provides a standard set of functions for accessing and manipulating menu items.

Indigo Magic Desktop Integration

The Indigo Magic desktop is an end-user environment that provides an icon-oriented interface to the IRIX operating system and filesystem. Users can launch applications and select files using icons instead of typing shell commands and pathnames. Integrating your product into this desktop is an important part of your application. It is much easier for users of your application who are familiar with the desktop to get started if your interface conforms to the Indigo Magic standard.

Learning About Silicon Graphics User Interface Style

Silicon Graphics provides user interface style guidelines for developers of software products used on Silicon Graphics workstations. The *Silicon Graphics User Interface Guidelines* describes these style guidelines. Its purpose is to help you create products that are consistent with other applications and that integrate seamlessly into the Indigo Magic desktop environment.

The guidelines cover these areas:

- The design of icons for your application programs and files—they should be meaningful and behave appropriately in response to user actions.
- The appearance of your application's windows and the expected behavior these windows should support—such as when users should be able to size, move, and minimize windows.
- The individual components of your desktop interface, such as menus, controls, dialogs, and use of color.

Integrating Your Application Into the Desktop

Integrating your application into the desktop is an important step in creating your product. Since your users are already familiar with the appearance and behavior of applications in the desktop environment, you can simplify their use of your application by conforming to this standard. Here are the steps to follow to integrate your application:

1. Achieve the Indigo Magic look and feel
2. Create desktop icons for your application
3. Package your application for installation

Achieving the Indigo Magic Look and Feel

Here are some of the things you want to consider to achieve the Indigo Magic look and feel:

- Use schemes—Silicon Graphics includes schemes in its implementation of Xt. Schemes allow you to provide default colors and fonts for your application while also ensuring that users can easily select other colors and fonts according to their individual needs and preferences. If you provide default colors, your application will use the same colors and fonts as other applications on the desktop using default colors.
- Use enhanced widgets—IRIS IM provides new and enhanced widgets that are part of the Indigo Magic look and feel. Some of these widgets are the Color Chooser, the Dial, and the Thumbwheel. You can use these widgets by linking to the appropriate library.
- Provide window, session, and desk management for your application—Most users of Silicon Graphics systems use *4Dwm*, which is based on *mwm* (the Motif window manager).
- Create minimized windows—You can customize the minimized version of your application's window so it's easily recognized when a user clicks the minimize button.

Creating Desktop Icons

When you create icons for your application, perform these tasks:

- Draw the icons—Use IconSmith™, a tool for drawing desktop icons.
- Program the icon—Define the look and behavior of the icons in your application, for example, the actions to be taken if a user double-clicks an icon.
- Install the icon in the icon catalog—Use the *iconbookedit* command to install your icon in the desktop's Icon Catalog.

Packaging Your Application for Installation

Silicon Graphics recommends that you use the software packaging tool Software Packager (*swpkg*) to package your application for installation. This allows your users to easily install your application using Software Manager (*swmgr*), Silicon Graphics software installation program.