Chapter 1

OpenGL on Silicon Graphics Systems

Silicon Graphics systems allow you to write OpenGL applications that are portable and run well across the Silicon Graphics workstation product line. This chapter introduces the basic issues you need to know about if you want to write an OpenGL application for Silicon Graphics systems. The chapter contains the following topics, which are all discussed in more detail elsewhere in this guide:

- "Using OpenGL With the X Window System" on page 1
- "Extensions to OpenGL" on page 6
- "Debugging and Performance Optimization" on page 7

Using OpenGL With the X Window System

OpenGL is a window-system-independent graphics library. The platform's window system determines where and how the OpenGL application is displayed and how events (user input or other interruptions) are handled. Currently, OpenGL is available for the X Window System, for OS/2™, for Windows NT™, and for Windows™ 95. If you intend your application to run under several window systems, the application's OpenGL calls can remain unchanged but window system calls are different for each window system.

Note: If you plan to have your application run under different window systems, be sure to keep the windowing code isolated as much as possible to minimize the number of files that must be special for each implementation.

All Silicon Graphics systems use the X Window System. Applications on a Silicon Graphics system rely on Xlib calls to manipulate windows and obtain input. An X-based window manager (usually *4Dwm*) handles iconification, window borders, and overlapping windows. The Indigo Magic™ desktop environment is based on X, as is the Silicon Graphics widget set, IRIS IM. IRIS IM is the Silicon Graphics port of OSF/Motif.™

A full introduction to X is beyond the scope of this guide; for detailed information about X, see the sources listed in "Background Reading" on page xxv.

The GLX Extension to the X Window System

The OpenGL extension to the X Window System (GLX) provides a means of creating an OpenGL context and associating it with a drawable window on a computer that uses the X Window System. GLX is provided by Silicon Graphics and other vendors as an adjunct to OpenGL.

For additional information on using GLX, see "The GLX Extension to X" on page 10. More detailed information is in Appendix D, "OpenGL Extensions to the X Window System" of the *OpenGL Programming Guide.* The *glxintro* reference page also provides a good introduction to the topic.

Libraries, Tools, Toolkits, and Widget Sets

When you prepare a program to run with the X Window System, you can choose the level of complexity and control that suits you best, depending on how much time you have and how much control you need.

This section looks at different tools and libraries for working with OpenGL in an X Window System environment, moving from easy-to-use toolkits and libraries with less control to the Xlib library, which is more primitive but offers more control. Most application developers usually write at a higher level than Xlib, but you may find it helpful to understand the basic facts about the lower levels discussed in this guide.

Note that the different tools are not mutually exclusive: You may design most of the interface with one of the higher-level tools, then use Xlib to fine-tune a specific aspect or add something that's otherwise unavailable. Figure 1-1 illustrates the layering:

- IRIS ViewKit™ and Open Inventor™ are layered on top of IRIS IM, which is on top of Xlib.
- GLX links Xlib and OpenGL.
- Open Inventor uses GLX and OpenGL.

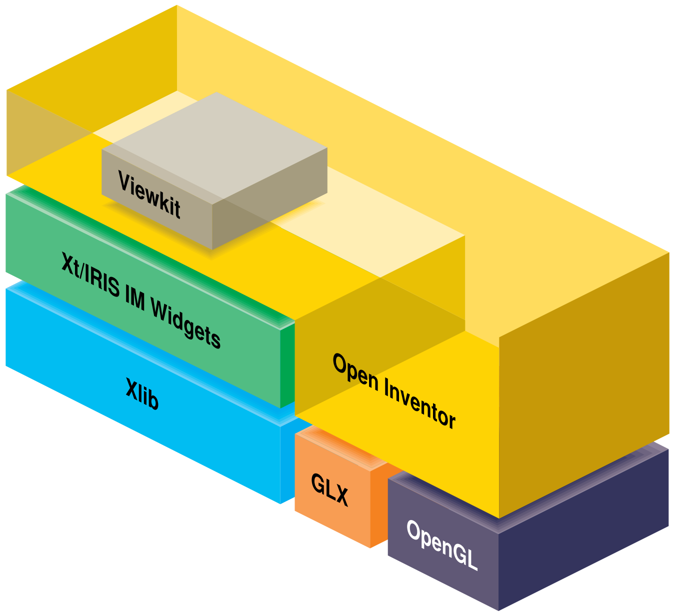

Figure 1-1 How X, OpenGL, and Toolkits are Layered

Note: If you write an application using IRIS Viewkit, OpenInventor, or RapidApp™, the graphical user interface will be visually consistent with the Indigo Magic desktop.

RapidApp

RapidApp is a graphical tool, available from Silicon Graphics, that allows developers to interactively design the user-interface portion of their application. It generates C++ code utilizing IRIS ViewKit (see “IRIS ViewKit” on page 5) for each user-interface component as well as the overall application framework. As with all applications based on ViewKit, IRIS IM (Motif) widgets are the basic building blocks for the user interface. RapidApp is not included in Figure 1-1 because it generates ViewKit and IRIS IM code and is therefore dependent on them in a way different from the rest of the hierarchy.

To speed the development cycle, RapidApp is integrated with a number of the Developer Magic™ tools. This allows developers to quickly design, compile, and test object-oriented applications.

RapidApp also provides easy access to Silicon Graphics specific widgets and components. For instance, you can add an OpenGL widget to a program without having to know much about the underlying details of integrating OpenGL and X.

For more information, see the *Developer Magic: RapidApp User's Guide*, also available online through IRIS InSight.

Open Inventor

The Open Inventor library uses an object-oriented approach to make the creation of interactive 3D graphics applications as easy as possible by letting you use its high-level rendering primitives in a scene graph. It is a useful tool for bypassing the complexity of X and widget sets, as well as many of the complex details of OpenGL.

Open Inventor provides prepackaged tools for viewing, manipulating, and animating 3D objects. It also provides widgets for easy interaction with X and Xt, and a full event-handling system.

In most cases, you use Open Inventor, not the lower-level OpenGL library for rendering from Open Inventor. However, the Open Inventor library provides several widgets for use with X and OpenGL (in subclasses of the SoXtGLWidget class) that you can use if OpenGL rendering is desired. For instance, the SoXtRenderArea widget and its viewer subclasses can all perform OpenGL rendering. SoXtGLWidget is, in turn, a subclass of SoXtComponent, the general Open Inventor class for widgets that perform 3D editing.

Components provide functions to show and hide the associated widgets, set various parameters (such as title and size of the windows), and use callbacks to send data to the calling application. The viewer components based on SoXtRenderArea handle many subsidiary tasks related to viewing 3D objects. Other components handle anything from editing materials and lights in a 3D scene, to copying and pasting 3D objects.

Note that if you're using libInventorXt, you only need to link with libInventorXt (it automatically "exports" all of the routines in libInventor, so you never need to use **-lInventorXt -lInventor**, you only need **-lInventorXt**).

For detailed information on Open Inventor, see *The Inventor Mentor: Programming Object-Oriented 3D Graphics with Open Inventor,™ Release 2*, published by Addison-Wesley and available online through IRIS InSight.

IRIS ViewKit

The IRIS ViewKit library is a C++ application framework designed to simplify the task of developing applications based on the IRIS IM widget set. The ViewKit framework promotes consistency by providing a common architecture for applications and improves programmer productivity by providing high-level, and in many cases automatic, support for commonly-needed operations.

When you use Viewkit in conjunction with OpenGL, it provides drawing areas that OpenGL can render to.

For more information, see the *IRIS ViewKit Programmer's Guide*, available online through IRIS InSight.

The IRIS IM Widget Set

The IRIS IM widget set is an implementation of OSF/Motif provided by Silicon Graphics. You're strongly encouraged to use IRIS IM when writing software for Silicon Graphics systems. IRIS IM integrates your application with the desktop's interface. If you use it, your application conforms to a consistent look and feel for Silicon Graphics applications. See the sources listed in "Background Reading" on page xxv for further details.

The Xlib Library

The X library, Xlib, provides function calls at a lower level than most application developers want to use. Note that while Xlib offers the greatest amount of control, it also requires that you attend to many details you could otherwise ignore. If you do decide to use Xlib, you are responsible for maintaining the Silicon Graphics user interface standards.

A Note to IRIS GL Users

An application that uses both IRIS GL and X is called a mixed-model program. If you prepared your IRIS GL application to run as a mixed-model program, porting to OpenGL becomes much easier. For porting information, see the *OpenGL Porting Guide*.

Many IRIS GL programs use the built-in windowing interface provided by IRIS GL. In contrast, OpenGL relies on X for all its windowing functionality. If your application uses

IRIS GL functions like **winopen()**, your windowing code needs to be rewritten for X. See the *OpenGL Porting Guide* for more information.

Note that the term "mixed-model program" is no longer relevant when you work with OpenGL, since all OpenGL programs use the native window system for display and event handling. (The OpenGL API, unlike IRIS GL, has no windowing calls).

Extensions to OpenGL

The OpenGL standard is designed to be as portable as possible and also to be expandable with extensions. Extensions may provide new functionality, such as several video extensions, or extend existing functionality, such as blending extensions.

An extension's functions and tokens use a suffix that indicates the availability of that extension:

- EXT is used for extensions reviewed and approved by more than one OpenGL vendor.
- SGI is used for extensions found across the Silicon Graphics product line, although the support for all products may not appear in the same release.
- SGIS is used for extensions found only on a subset of Silicon Graphics platforms.
- SGIX is used for experimental extensions: In future releases, the API for these extensions may change, or they may not be supported at all.

The *glintro* reference page provides a useful introduction to extensions; many extensions are also discussed in detail in the following chapters in this guide:

- Chapter 5, "Introduction to OpenGL Extensions"
- Chapter 6, "Texturing Extensions"
- Chapter 7, "Imaging and Blending Extensions"
- Chapter 8, "Miscellaneous OpenGL Extensions"
- Chapter 9, "Extensions to GLX"

Note that both the X Window System and OpenGL support extensions. GLX is an X extension to support OpenGL. Keep in mind that OpenGL (and GLX) extensions are different from X extensions.

Debugging and Performance Optimization

If you want a fast application, think about performance from the start. While making sure the program runs reliably and bug free is important, it's also essential that you think about performance early on. Applications designed and written without performance considerations can almost never be suitably tuned.

If you want high performance, read the performance chapters in this guide before you start writing the application.

Debugging Your Program

Silicon Graphics provides a variety of debugging tools for use with OpenGL programs:

- The *ogldebug* tool helps you find OpenGL programming errors and discover OpenGL programming style that may slow down your application. You can set breakpoints, step through your program, and collect a variety of information.
- For general-purpose debugging, you can use standard UNIX debugging tools such as *dbx*.
- Also available (for general-purpose debugging) are the CASE tools. For more information on the CASE tools, see *ProDev WorkShop and MegaDev Overview* and *CASEVision/Workshop User's Guide.*

Tuning Your OpenGL Application

The process of tuning graphics applications differs from that of tuning other kinds of applications. This guide provides platform-independent information about tuning your OpenGL application in these chapters:

- Chapter 11, "Tuning Graphics Applications: Fundamentals"
- Chapter 12, "Tuning the Pipeline"
- Chapter 13, "Tuning Graphics Applications: Examples"

In addition, there are tuning issues for particular hardware platforms. They are discussed in Chapter 14, "System-Specific Tuning."

Maximizing Performance With IRIS Performer

The IRIS Performer™ application development environment from Silicon Graphics automatically optimizes graphical applications on the full range of Silicon Graphics systems without changes or recompilation. Performance features supported by IRIS Performer include data structures to use the CPU, cache, and memory system architecture efficiently; tuned rendering loops to convert the system CPU into an optimized data management engine; and state management control to minimize overhead.

Location of Example Source Code

All complete example programs (though not the short code fragments) are available in */usr/share/src/OpenGL* if you have the *ogl_dev.sw.samples* subsystem installed.