Chapter 6

Application Libraries

Silicon Graphics provides several application libraries that you can use in writing your applications. These libraries provide tools for developing programs in the following areas:

- Graphics
- Image processing
- Digital media
- Printer/scanner management

Refer to Table 2-5 for a list of the manuals you can read to learn more about the topics discussed in this chapter.

Graphics Libraries

Silicon Graphics supports four graphics libraries.

- OpenGL provides low-level graphics routines and is an interface to the graphics hardware.
- Open Inventor is a toolkit, built on OpenGL, that allows you to create interactive graphics applications.
- IRIS Graphics Library is a library of subroutines for creating 2D and 3D color graphics and animation.
- IRIS Performer is a toolkit for creating real-time graphics and visual simulation applications.

OpenGL

OpenGL is a software interface to graphics hardware. It consists of about 120 commands that you can use to specify the objects and operations needed to create interactive programs that produce color images of moving 3D objects. OpenGL is an industry standard for 2D and 3D graphics rendering and is a part of the IDO.

OpenGL uses a client-server model for interpretation of commands. An application using OpenGL can run under IRIX on any Silicon Graphics platform and be rendered on a platform with another operating system and window system, provided the implementation of OpenGL on each platform conforms to the standard.

OpenGL doesn't include commands for performing windowing tasks or obtaining user input. For that you must work through the windowing system that controls your hardware. Since the OpenGL application programming interface (API) is independent of hardware platforms, window systems, and operating systems, porting among conforming implementations of OpenGL is an easy task.

OpenGL allows you to build the models you need from a small set of geometric primitives—points, lines, and polygons. It doesn't provide high-level commands for describing models of 3D objects—Silicon Graphics provides higher level graphics libraries for these tasks. One of these libraries is Open Inventor, which is built on OpenGL and uses OpenGL calls for rendering. You can write your application using the OpenGL API, although it's often easier to use one of the higher level libraries. Using the higher level graphics libraries makes rendering operations transparent to your application, which allows you to concentrate on your application rather than on the time-consuming details of rendering.

Note: IRIS Performer and the ImageVision Library are being converted from IRIS GL (the precursor to OpenGL) to the OpenGL standard.

The OpenGL library contains functions for

- rendering primitives (points, lines, polygons)
- controlling colors and lighting
- using texture mapping to add surface characteristics to geometry
- setting and controlling transformations

Figure 6-1 shows an object created by OpenGL—a sphere illuminated by a light source.

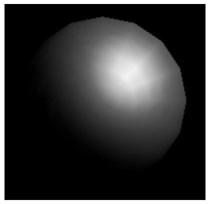

Figure 6-1 Using the Lighting Feature of OpenGL

Figure 6-2 shows another object created by OpenGL—a texture-mapped Bezier surface mesh.

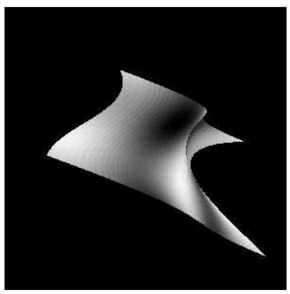

Figure 6-2 Using the Texture-Mapping Feature of OpenGL

OpenGL provides a small but powerful set of rendering commands, and all higher-level drawing must be done using these commands. To simplify your programming tasks, OpenGL provides a Utility Library (GLU) that includes routines that encapsulate OpenGL commands. These GLU routines perform tasks such as:

- drawing common objects such as spheres, cylinders, and disks
- manipulating images used in texturing
- handling simple non-convex polygons
- setting up matrices for a variety of viewing orientations and projections

Open Inventor

Open Inventor is an object-oriented toolkit that provides objects and methods for creating interactive 3D graphics applications. This toolkit contains 3D objects you can use to represent your 3D physical models, as well as objects that allow you to interactively operate on these models.

Figure 6-3 shows a single scene from a racing game created using Open Inventor. In this game, mouse buttons control the speed and position of a car as it moves along a track. Open Inventor creates scenes showing the car, the track, and the terrain that appears as the car moves along the track.

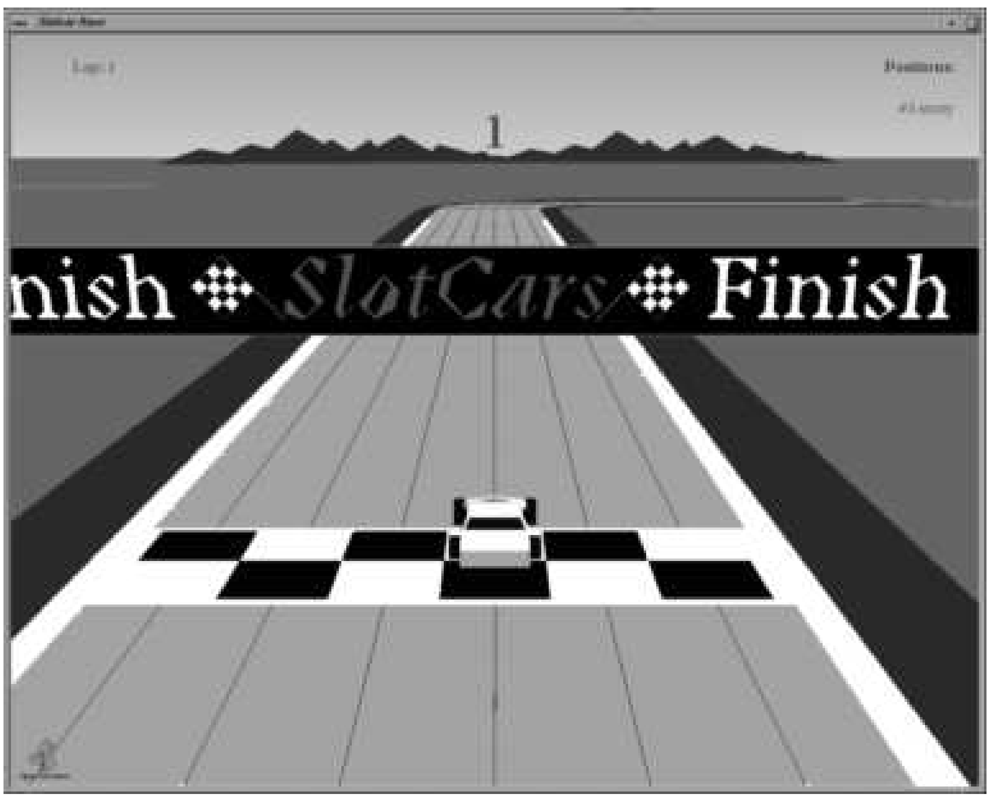

Figure 6-3 A Scene Created by Open Inventor

Open Inventor is written in C++ but also includes C bindings. It is object-oriented and extensible. The Inventor toolkit is based on OpenGL and provides a library of objects you can use, modify, and extend to meet your needs.

Figure 6-4 illustrates the architecture of an Open Inventor application. The Open Inventor components shown in the figure are described in the paragraphs following the figure.

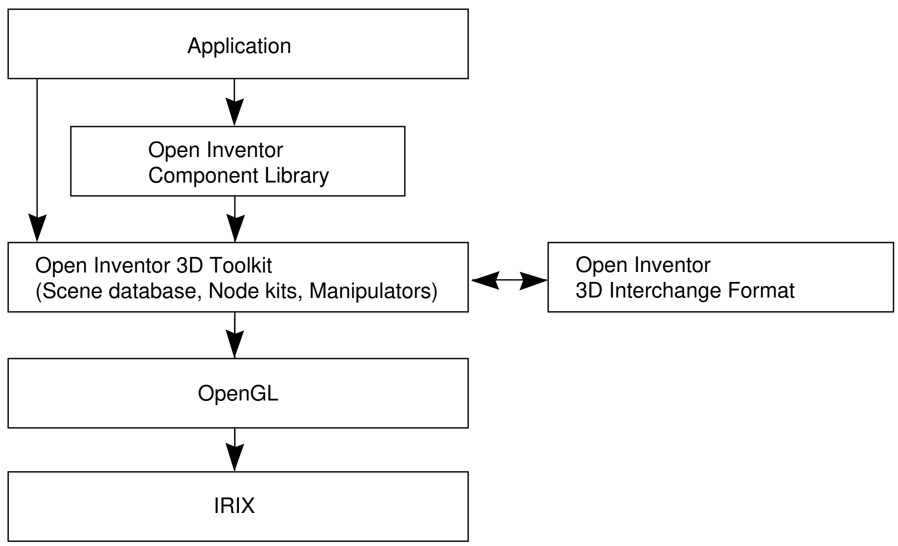

Figure 6-4 Open Inventor Architecture

Open Inventor Toolkit

The Open Inventor Toolkit provides three programming tools that you can use in your Open Inventor application.

Scene database A scene database is a collection of 3D objects and properties arranged to represent a 3D scene. A scene is composed of nodes that define all information about an object—its shape, size, coloring, surface texture, and location in 3D space. You can use this information to render the object or to vary it in a variety of ways—for example, to move the object or change the way it looks. Some objects, called engines, are used to animate part of a scene.

Manipulators	A manipulator is a special kind of node that reacts to user events. Manipulator objects allow users to interact with 3D objects on a screen. Manipulators allow rendering into a scene and provide a means for translating user-initiated events into changes to the scene database.
Node kits	A node kit is a collection of nodes grouped together to provide a simplified model. Open Inventor provides these ready-made kits to make building a structured scene database easier. The kit provides the basic structure of an object, but allows you to define information specific to your object. For example, the shape node kit describes the shape but allows you to define a geometric specification, material, a lighting model, texture, and other properties of the shape.

Component Library

The Component Library is a convenience library for programmers who use X Window System and X-based toolkits such as Xt and Motif. It contains an event translator that converts X events into Open Inventor events.

3D Interchange File Format

Open Inventor includes an interchange file format for exchanging 3D objects and scenes between applications. Objects in the scene database can be written to a file during the execution of your program, in either ASCII or binary form.

IRIS Graphics Library

The IRIS Graphics Library (IRIS GL) is a library of subroutines for creating 2D and 3D color graphics and animation.

Here are some of the things IRIS GL allows you to do:

- draw graphics primitives such as points, lines, polygons
- draw characters and define fonts
- use color modes and color maps to control the way colors are displayed
- use double buffering to create animated graphics
- perform coordinate transformations

- define and manipulate light sources to create lighted scenes
- use texture mapping to add surface characteristics to geometry

IRIS GL is a predecessor of OpenGL, the industry standard for graphics applications. Silicon Graphics' application libraries that were originally built on IRIS GL are moving to the OpenGL standard. Figure 6-5 shows relationship of an application to IRIX GL and to the ImageVision Library and IRIS Performer, the libraries currently built on IRIS GL.

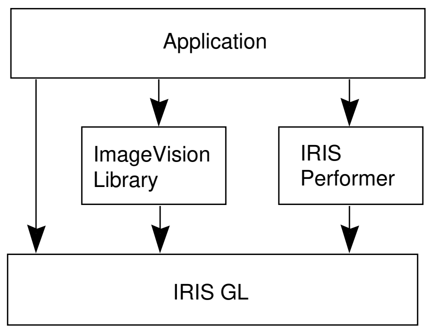

Figure 6-5 IRIS GL in the Developer Environment

Note: You should never reference both IRIS GL and OpenGL in a single application. This means you should not use a higher-level library based on OpenGL (for example, Open Inventor) in the same application in which you use a library based on IRIS GL (for example, the ImageVision Library).

IRIS Performer

IRIS Performer is a software development environment layered above the IRIS Graphics Library (IRIS GL). It provides high-level support for visual simulation, interactive entertainment, virtual reality, and graphics-intensive tasks. Applications that require real-time visuals and high-performance rendering benefit from using IRIS Performer.

The main components of IRIS Performer are the two libraries *libpr* and *libpf*.

- *libpr* is a low-level library that provides optimized rendering functions, state control, and other functions that are fundamental to real-time graphics. It provides highly optimized rendering loops for rendering a wide variety of geometric primitives.
- *libpf* is a visual simulation development environment that layers a multiprocessing database traversal and rendering system on *libpr*. It supports multiprocessing, hierarchical scene construction, multiple channels, culling to each channel's field-of-view, and frame-rate control.

Figure 6-6 shows the relationship between the IRIS Performer libraries and the IRIX system software.

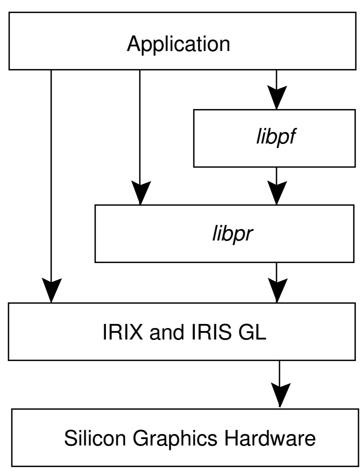

Figure 6-6 IRIS Performer Library Hierarchy

You can choose the IRIS Performer libraries that best suit your needs. You may want to build your own toolkits on top of *libpr*, the low-level, high-performance library, or you may choose to take advantage of the visual simulation environment that *libpf* provides. Note that functions from *libpf* make calls to *libpr* functions so you don't necessarily have to use the *libpr* functions directly.

IRIS Performer doesn't define a file format; it imports files from many standard database formats at run time. Some of the database formats supported by IRIS Performer are shown in Table 6-1.

Table 6-1 Database Formats Supported by Performer

Format Name	Description
BIN	Silicon Graphics format
DWB	Designer's Workbench format
DXF	AutoCAD® format
FLT	MultiGen™ FLIGHT format
IV	Inventor format
OBJ	Wavefront Technologies *Model* format

ImageVision Library

Unlike the graphics libraries, which build a display from a series of geometric objects, image processing applications start with an image consisting of pixel information stored in a file. These images can originate as a photographic image that's scanned or obtained from a Kodak™ CD, data obtained from medical imaging equipment, satellite data, or numerous other sources. An image processing application can manipulate this image in ways that are meaningful to the user of the application.

The ImageVision Library (IL) is a set of tools designed for developers of image processing applications. The IL is written in C++ but has interfaces for C and Fortran. You can use this library to import, manipulate, display, and store images.

The IL is an object-oriented toolkit whose modularity provides an easy and efficient means to create and maintain programs that use it for image manipulation and display. This modular structure also makes it easy to extend the IL—for example, to augment the image operators supplied by the IL or to design new ones.

The ImageVision Library contains objects and methods that allow an image processing application to:

- Import images created in a variety of different file types. The supported file formats include TIFF, GIF, Kodak Photo CD™, SGI, and FIT.
- Process images using any sequence of the image processing operators supported by the IL. Image processing functions include color conversion, arithmetic operations on pixel data, radiometric and geometric transformations, generation of statistical data for an image, spatial and non-spatial domain transformations, and edge, line, and spot detection.
- Display one or more images in an X Window. The IL provides many ways to control image displays, including stacking images or aligning them side by side, roaming a large image, or doing a wipe to move one edge of an image to reveal what is stacked beneath it.
- Store processed images on disk.

The following figures illustrate an ImageVision Library operation. Figure 6-7 shows an image imported from a Kodak Photo CD.

Figure 6-7 A Display Created by ImageVision Library

Figure 6-8 shows the image created when the ImageVision library performs a geometric transformation called a warp on the image in Figure 6-7.

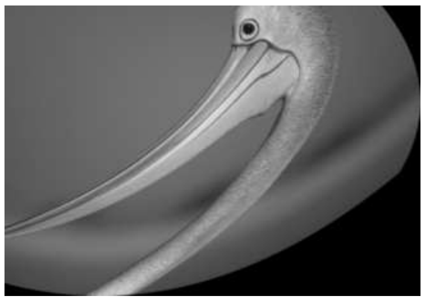

Figure 6-8 Using the ImageVision Library to Transform an Image

Figure 6-9 illustrates a simple ImageVision application that reads an image from disk, applies a rotation transformation, displays both images on a monitor, and writes the rotated image to disk.

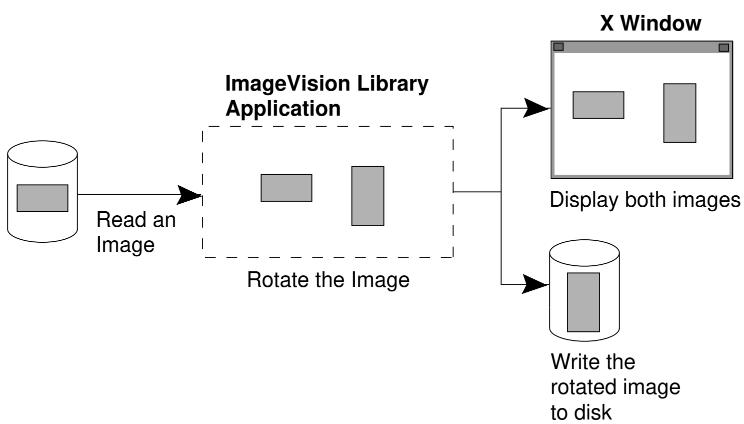

Figure 6-9 A Simple ImageVision Library Application

The ImageVision Library toolkit provides an application program interface (API) that is common across all Silicon Graphics workstations. The IL uses Xlib for window management and allows you to use either IRIS GL or X to render into an X window.

Figure 6-10 shows the architecture of an IL application.

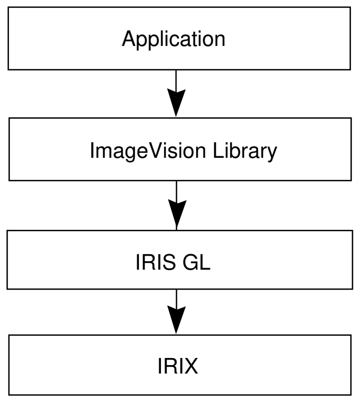

Figure 6-10 Architecture of an ImageVision Library Application

The IL implements an execution model that optimizes memory usage and performance as image data is processed. This execution model

- supports the parallel processing features of Silicon Graphics workstations
- caches image data to minimize file access
- is demand-driven so that only image data needed for output is processed
- chains operations together, which saves time because intermediate results don't have to be stored
- uses hardware acceleration of graphics operations whenever possible to improve performance of IL operations

IRIS Digital Media Development Environment

The IRIS Digital Media Development Environment provides digital media libraries and tools for developers of media applications.

IRIS Digital Media Libraries

The IRIS Digital Media Libraries provide programming support for digital media development on Silicon Graphics platforms. These libraries are included with the IDO. The term *digital media* describes digitally sampled audio and video (including still image) data, MIDI event streams, and other associated information such as time codes. Sampled audio/video data can be digitally encoded in a variety of uncompressed and compressed formats

The IRIS Digital Media Development Environment provides programming support for digital audio, digital video, and MIDI applications. The libraries provide programming interfaces to:

- audio, video, and MIDI I/O subsystems
- data format conversion (including compression)
- digital media file importation/exportation
- high-level playback functions

The five libraries that comprise the IRIS Digital Media Development Environment are briefly described in the following paragraphs.

Digital Audio Libraries

The digital audio libraries provide a device-independent programming interface to the digital audio I/O subsystems built into Silicon Graphics workstations. You can use the digital audio libraries individually or in combination. Table 6-2 describes the libraries contained in the digital audio library set.

Table 6-2 Digital Audio Libraries

Library	Library Function
Audio	Provides an interface for configuring the audio system, managing audio I/O between the application program and audio hardware, specifying attributes of digital audio data, and facilitating real-time programming.
Audio File	Provides an interface for reading and writing the standard digital audio file formats AIFF and AIFF-C standards.
CD Audio	Provides an interface for optional Silicon Graphics SCSI CD-ROM drives. This interface features a special mode that allows it to read audio CD format as well as CD-ROM format.
DAT Audio	Provides an interface for optional Silicon Graphics SCSI DAT drives.

Figure 6-11 diagrams the interaction between an audio application and the audio libraries, the device drivers, the IRIX filesystem, the audio hardware, and the optional SCSI devices.

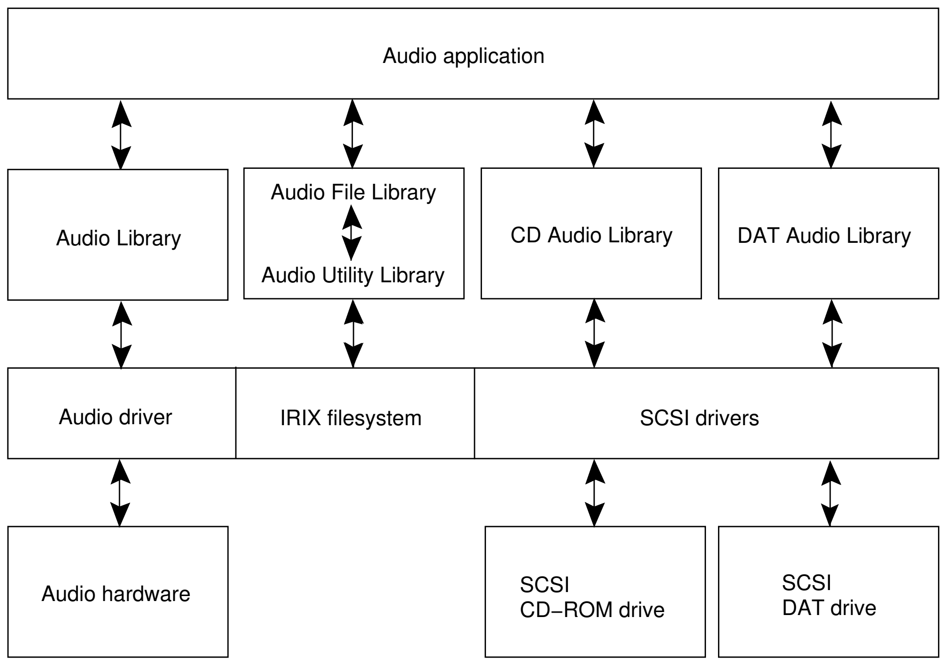

Figure 6-11 Interaction of Digital Audio System Components

MIDI library

The Musical Instrument Digital Interface (MIDI) Library provides a programming interface for timestamped MIDI input/output via serial ports.

Video Library

The Video Library provides both device-independent and device-dependent interfaces to the on-board Indy VINO, and to video options such as Indy Video™, Galileo Video™, Indigo^2 Video™, and Sirius Video™.

Compression Library

The Compression Library (CL) provides an algorithm-independent, extensible interface for compressing and decompressing animation, video, audio, and image data. The CL interface supports the Cosmo Compress JPEG codec (available for Indigo R4000, Indy, Indigo 2) in addition to several software codecs, including software JPEG. Cosmo Compress connects to a Galileo-family video device to allow realtime JPEG video capture and playback. In this configuration, Cosmo operates as a component of the video I/O system.

Movie Library

The Movie Library is a collection of routines that provides a C language API for creating, reading, writing, editing, and playing movie files. Supported file formats include Silicon Graphics Movie File (SGIMF) format and the Apple® QuickTime™ movie file format.

Digital Media Tools

IRIX includes several interactive media tools that are built on the IRIS Digital Media Development Environment and make it easy to perform basic media functions. These tools have command-line interfaces that allow you to incorporate them into your digital media application. Table 6-3 lists the digital media tools.

Table 6-3 Digital Media Tools

Tool	Tool Function
Capture	Record audio, video input, audio, and still images on your system disk and import them into multimedia applications.
Movie Maker	Combine audio, video, and still images to create a movie you can play on your workstation.
Movie Player	Play movies on your workstation using controls that allow you to play, stop, reverse, and fast forward.
Sound Editor	Record sound into and edit audio files.
Sound Filer	Play audio files on your system and convert audio files to different formats and sizes.
CD Manager	Play and record from a compact disc drive attached to your workstation SCSI port.
DAT Manager	Play and record to and from DAT tapes on a DAT drive attached to your workstation SCSI port

Read the *Media Tools User's Guide*, available in IRIS Insight, or the reference page for each tool to learn more about these media tools.

Printer/Scanner Management

Silicon Graphics provides a printing and scanning environment for IRIS workstations. This environment, Impressario, has features for a wide range of IRIX audiences: printer and scanner driver developers, application program developers, and end users. Impressario allows files of different types to be printed on a wide variety of printers and allows images to be scanned from a scanning device, an IRIS screen, or a Silicon Graphics image file.

Impressario provides the following libraries for application developers:

libspool — Provides an API to the IRIX printer spooling system and functions, allowing you to submit print jobs, query their status, and so on.

libprintui — Provides a graphical interface for printing that's compatible with IRIS IM.

libpod — Provides a network-transparent interface to the Printer Object Database (POD). Each printer has a POD that contains configuration, status, and other information about that printer.

libscan — Provides an interface to the Impressario scanning system.

libstiff — Allows you to read and write Stream TIFF (STIFF) files.

libimp — Allows you to read and write Silicon Graphics image files in RGB format.

Impressario is built on top of the System V print spooling system. It provides model files, filters, and drivers to convert ISO text files, SGI images files, PostScript® documents, and a wide variety of other file formats to a format suitable for both raster and PostsScript printers. Figure 6-12 shows the relationship between an application program, the Impressario libraries, and the spooling system in IRIX.

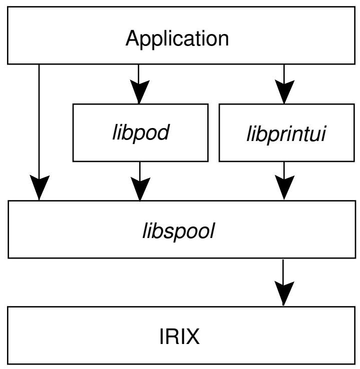

Figure 6-12 Interface to the Spooling System

Refer to the *Impressario Programming Guide* for information about writing a driver for a printer or scanner device for which no driver is available.