Solaris VISUAL Pixel Interfaces 6

Pixel-based Rendering Interfaces

The Solaris VISUAL environment provides a platform for pixel-based applications and APIs through the X foundation library, the standard Xlib API and the X11 protocol.

Pixel-based rendering is used to manipulate individual pixels of an image to be displayed. Pixel-based rendering is the lowest common denominator of all graphics rendering, since all graphics must, at some point, be reduced to individual pixels. Pixel-based routines operate in the 2-D integer device coordinate space of the image.

With some graphics devices, such as those used for geometry rendering, pixel manipulation takes place within the hardware. Such devices are optimized to process higher-level primitives such as vectors, triangles and quadrilaterals. Nonetheless, it is sometimes desirable to work with the pixels to be displayed. Image handling, window systems and graphical user interfaces are examples where pixel rendering is important.

Pixel-based Rendering Applications

Pixel-based applications cover a broad spectrum of areas, principally where close coupling with an X Windows server and individual pixel rendering is desired. The Solaris VISUAL pixel-based foundation library and Xlib offer complete interoperability with any X Windows compliant system, thus making them important to developers of multi-platform applications.

Windowing

Windowing applications, such as graphical user interface toolkits are commonly implemented on Xlib, relying on the pixel rendering capabilities which are critical to insuring accurate detail rendering of user interface widgets. Common applications based on pixel-based rendering interfaces include the Motif and OLIT user interface toolkits, a variety of window managers, and many third-party toolkit implementations.

Business Graphics

Business graphics applications do not generally have significant rendering or display demands. Commonly, business graphics applications such as presentation graphics, rely on simple 2-D primitives (lines, arcs, rectangles, etc.) and their requirements can be easily met by pixel-based rendering interfaces.

The X Window System Foundation

The Solaris VISUAL environment relies on the X Window System foundation for pixel-based rendering, as well as minimum protocol support for its client-server architecture.

Developers of Solaris platform display hardware must port the X foundation interface to their device to support Solaris VISUAL foundation libraries as well as the X graphics libraries and third-party APIs. If the display hardware has special characteristics such as features supporting 3-D geometry acceleration or image manipulation, the hardware implementor may choose to port additional foundation libraries for greater application performance and access to underlying hardware features.

The X Window System Foundation is based upon the MIT architecture for the device-independent (DIX) and device-dependent X (DDX) layers. Directly generating X Windows X11 protocol packets is more complicated than is desired by most application developers. Hence programming access to the X Window System generally occurs via the Xlib API. Figure 6.1 illustrates the X Windows foundation within Solaris VISUAL.

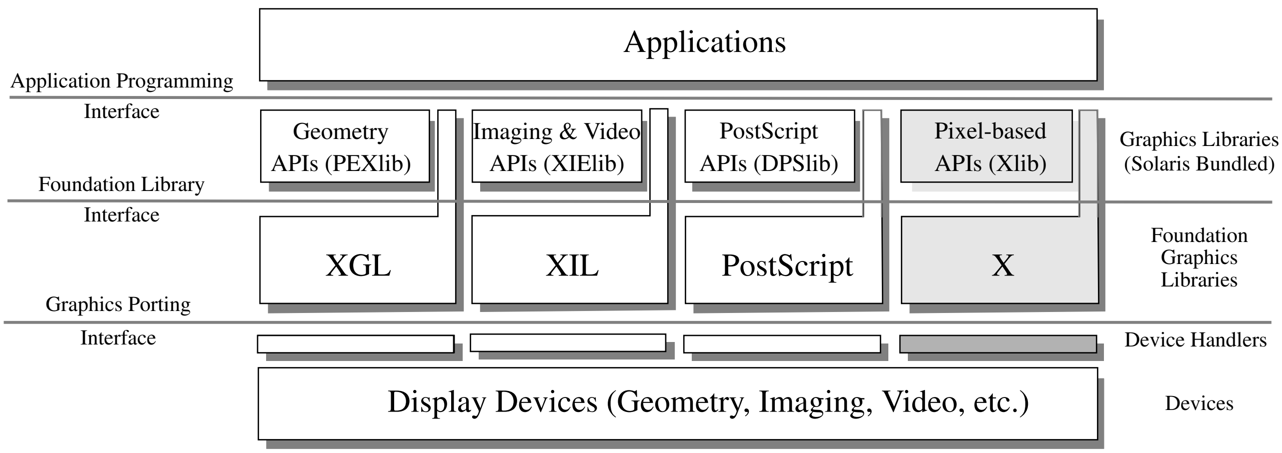


Figure 6.1 Solaris VISUAL Pixel-based Foundation Library and Layered Interfaces

The X Window System Protocol (X11)

The X11 protocol is the true definition of the X Window System. To be X Windows compliant, an application (such as a window manager or graphics library) must generate the X11 protocol requests to the server.

The X protocol specifies four types of messages: *requests*, *reply*s, *events* and *errors*, which can be transferred over the network between the client and server. These messages collectively enable client and server communication of mouse and keyboard input, graphics rendering, errors and inquiries regarding resources or state.

Xlib

The Solaris VISUAL Graphics Environment includes Xlib, the common rendering layer for most X Window System applications. Xlib provides the lowest level programming interface available for X, and a portable interface for applications across multiple vendors' hardware platforms.

Xlib is considered to be a *low-level* interface, where the programmer has complete access to the X Windows System environment, while still maintaining network transparency. Programmers using the Xlib interface do not need to concern themselves with the underlying network protocol, though they do need to manage aspects of the user interface and graphics rendering.

Xlib forms the fundamental basis of graphical user interface toolkits in the X Window environment. These toolkits either require or allow the application to call Xlib explicitly in conjunction with the toolkit.

Xlib includes far more than graphics rendering routines. Xlib was designed to permit virtually any style of user interface, and thus has very broad functionality. Xlib also provides functions for display connection management, error handling, font manipulation, client access control, keyboard mapping, resource management, text manipulation and inter-client communication.

Xlib maps closely to the X protocol, and as such, most of these calls correspond to the packet encodings of X11 Release 5, as defined in the *X Protocol Reference Manual*. Applications which use Xlib and follow the *Inter-Client Communications Convention Manual* (ICCCM) are considered fully compliant with the X Windows standard.

Direct Xlib

SunSoft provides a standard implementation of Xlib, together with a highly-optimized implementation, known as *Direct Xlib*. To a calling program, the Direct Xlib interface is identical to Xlib in every respect. When rendering remotely, Direct Xlib uses the X protocol for rendering via the X Windows server. However, when rendering locally, Direct Xlib utilizes SunSoft's Direct Graphics Access (DGA) to accelerated display hardware. Figure 6.2 illustrates both the normal Xlib rendering path and the Direct Xlib path.

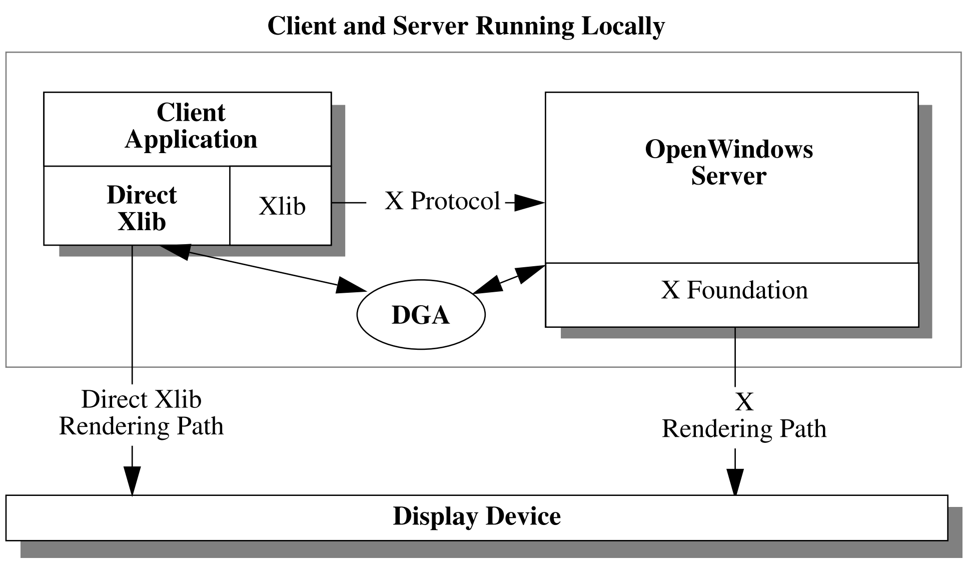


Figure 6.2 Direct and Server Rendering Paths

Xlib Rendering Primitives

Figure 6.3 illustrates the basic graphics primitives associated with Xlib. Other rendering routines exist within Xlib, though their functionality is generally duplicated. Additional routines, relating to color assignment, primitive color and pattern fill, image and pixel manipulation, and attribute access and control functions also comprise Xlib. For additional details on Xlib or the X protocol, consult the *Xlib Reference Manual*, and the *X Protocol Reference Manual*, published by O'Reilly and Associates.

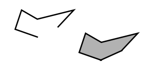
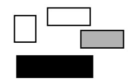
Primitive Call	Description	Example
XDrawLine(s) *XFillPolygon*	Line(s) connecting points	
XDrawPoint(s)	Point(s)	
XDrawRectangle(s) *XFillRectangle(s)*	Rectangle(s)	
XDrawSegments	Disconnected line segments	
XDrawArc(s) *XFillArc(s)*	Arc(s) fitting inside rectangle(s)	
XDrawString	1-byte (8-bit) text strings	123 abc
XDrawString16	2-byte (16-bit) text strings	αβχ δεζ
XDrawText	Multiple 1-byte text strings	123 abc 321 xyz
XDrawText16	Multiple 2-byte text strings	αβχ δεζ ξψζ φκλ

Figure 6.3 Xlib Graphics Routines (Rendering Primitives)

Remote Rendering using X11

A user wishing to run an X Windows application remotely can do so by identifying the hosts on which the application is to run and display. The local X Windows server must have granted appropriate access rights to the remote system (at initialization or via the xhost command). Finally, the remote application must indicate to Xlib, the system on which to display.

Once the application is invoked, it will connect to the appropriate server and proceed to emit X11 protocol to communicate with the server process.

Third-Party Pixel Interfaces

Because Xlib is the most common and portable graphics interface on X Windows-based systems, and due to its inherent portability, many application and API vendors have utilized it for supporting the graphics rendering within their products.

Many third-party graphics API vendors for example, provide an Xlib-based version of their product. Such products include a range of imaging and geometry interfaces including GKS, PHIGS, HOOPS, KHOROS and many others.