Solaris VISUAL Imaging Interfaces 4

Imaging Interfaces

The Solaris VISUAL environment addresses the needs of imaging applications and APIs through the XIL foundation library, the emerging XIE imaging extension and XIElib (when available from the X Consortium), and third-party and industry standard APIs.

XIE and third-party and industry standard APIs are not limited to any one set of customers, but rather provide the basic imaging capabilities required by a number of markets. Developers have the freedom to build their applications on the interfaces(s) which best address their needs, whether they are based upon performance, flexibility, features or portability.

Imaging Applications

The Solaris VISUAL imaging interfaces do not address any one set of customers, but rather provide the basic imaging capabilities required by a number of markets. Typically these markets emphasize WYSIWYG display of text and/or images, and often require fast rendering, manipulation, and display of images.

Commercial Document Imaging

Banks and other financial institutions, airlines and insurance companies process enormous numbers of paper documents in a typical day. Developers addressing this market must provide applications that read digital versions of documents from a scanning device, compress the images into CCITT Group 3 or 4 facsimile format, index and store them. Later, it may be necessary to retrieve the documents in a rapid manner, for additional image processing, scaling and enhancement.

Technical Document Imaging

This market handles fewer images than the commercial market, although the images are generally much larger. Potential applications in the technical document imaging market include textile design, cartography, and architecture. The Solaris VISUAL imaging interfaces will enable applications in this market to process very large images restricted only by the size of virtual memory.

Desktop Publishing

Desktop publishing products must be able to capture or import images. The Solaris VISUAL imaging interfaces enable developers of desktop publishing products to use imaging accelerators to resize images, and to filter and convert them from one data type to another. The advanced imaging functions of XIL and XIElib makes them better choice for managing images in desktop publishing applications than other pixel-oriented interfaces such as Xlib.

Color Prepress

A typical customer in this field wants to scan in color photographs and prepare them for printing on a color printer. The first step in this preparation might be to perform simple image processing operations on the images, such as sharpening, blurring, rotation or cropping. Finally, the images must be converted to the CMYK colorspace, which provides the digital equivalent of photographic color separation. Morphological dilation of selected image features (known as “trapping”) may be employed to avoid registration problems.

Technical Imaging

Technical imaging APIs and applications require broad base-level functionality for image manipulation, display, compression and color. Technical imaging applications frequently require APIs which address advanced image processing facilities including transformations, filtering and image enhancement. The Solaris VISUAL imaging interfaces provide a flexible foundation for technical imaging APIs, including emerging standards, as well as applications.

The XIL Foundation Library

XIL is the Solaris VISUAL foundation imaging library providing the common functions required by all imaging applications. As a foundation software layer for imaging applications, XIL defines how imaging operations, such as display, image manipulation, compression and decompression, are carried out.

XIL contains three primary components: a programming interface specification for basic imaging functionality, a high performance implementation of the specification and a standard hardware interface specification, which enables third-party hardware developers to readily support XIL and applications built upon XIL.

XIL is an object-oriented interface. As such, XIL hides the details of its implementation from the programmer. This enables the internals of XIL to be extended by SunSoft without changing the interface incompatibly. C language bindings provide compatibility with other tools available under Solaris including a large number of software development products. Figure 4.1 illustrates XIL in relationship to Solaris VISUAL.

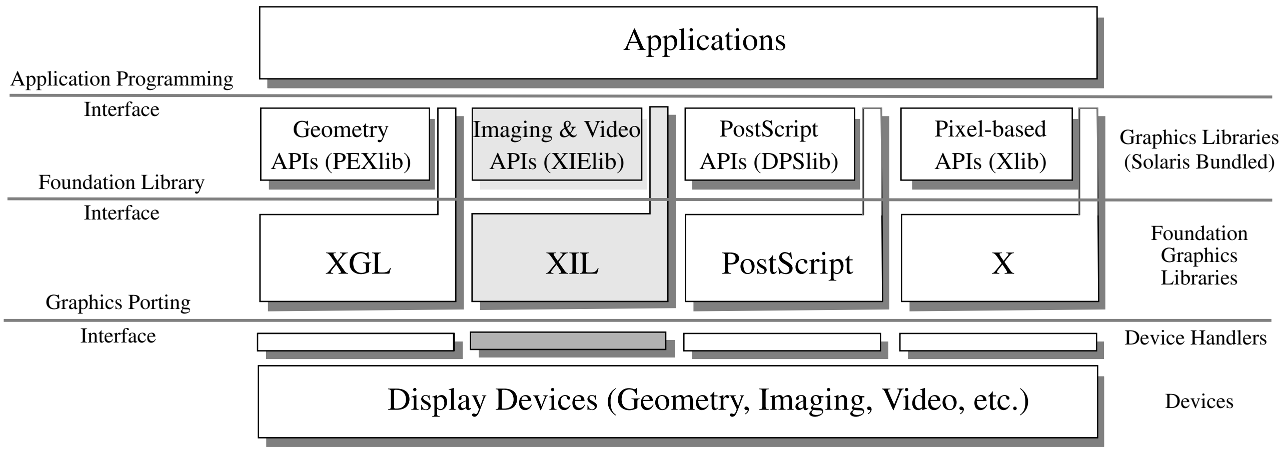

Figure 4.1 Solaris VISUAL XIL Foundation Library and Layered Interfaces

XIL Objects

In XIL, an *object* is an abstraction of a graphics resource. Objects consist of *state information* (known as attributes in XIL) and a set of operators. Operators perform predefined actions on objects of their related class. Figure 4.2 illustrates XIL's object hierarchy.

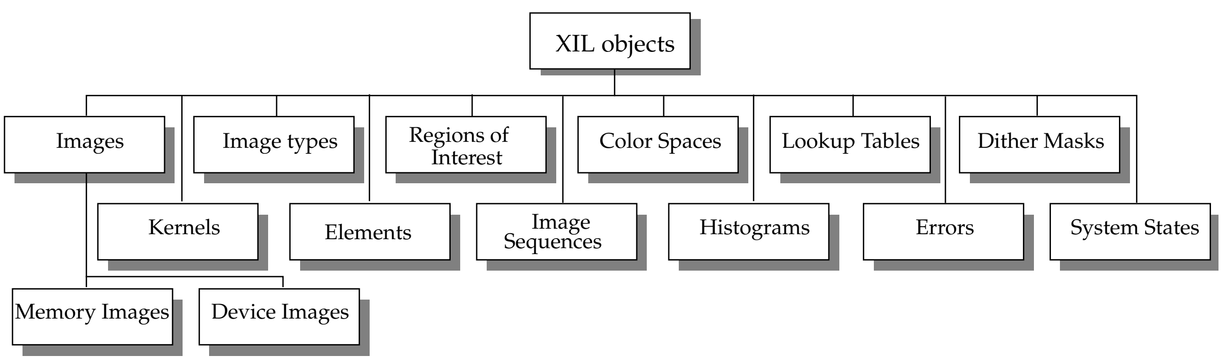

Figure 4.2 XIL Object Hierarchy

Images

The most important XIL data object is the image. This object stores a set of pixel values and values for a number of image attributes, such as width, height, number of bands, and data type. The library supports images of virtually any size: an image's width, height, and number of bands can each be as great as 2^{32}-1. In addition, the library supports three types of data: 1-bit data, 8-bit unsigned data, and 16-bit signed data per band.

When an application creates an image, the application receives an opaque handle to the image and, therefore, must process the image using XIL functions exclusively. However, the library enables applications that need to manipulate pixel values directly to *export* images. When an application exports an image, it gains access to information about how the image is stored in memory, including the address at which the image's pixel values are stored. The pointer in which this address is stored remains valid until the application *imports* the image back into XIL space.

Typically, when an XIL image is created, both the image structure that describes the image and the image's pixel values are stored in system memory. This type of image is called a *memory image*. All operations that do not read or write a device use memory images exclusively.

In addition to memory images, the XIL library supports *device images*. These images represent such devices as displays, cameras, frame grabbers, scanners, and printers. For each device image, the structure describing the image is stored in system memory, but the image's pixel values are read from or written to the device. A device image can be used as either the source or destination image for an XIL operation.

Display images are a special subclass of device images. A display image is associated with an X window and enables an XIL application to write directly to a screen.

Image Types

For every XIL image, there is a subset of attributes that constitutes its image type. These attributes are its width, height, number of bands, data type, and color space. An application might get an image's image type in preparation for creating an X window in which to display the image.

Regions of Interest

In the XIL scheme, a region of interest is attached to an image rather than to an operation. This region of interest is a single-bit mask that determines either which pixels in a source image can be read or which pixels in a destination image can be written. You build this mask by adding rectangles to an empty region of interest or by converting an XIL image or an X region to a region of interest.

The library calculates the region of interest for an operation by taking the intersection of the regions of interest of the images involved in the operation.

Color Spaces

A color space data object stores information about an image's color space. An application generally attaches color-space objects to images in preparation for converting an image from one color space to another. The color spaces supported by the XIL library include:

- CCIR Rec. 709 RGB
- Linear version of CCIR Rec. 709 RGB
- CCIR Rec. 709 YCC
- Luminance-only space derived from CCIR Rec. 709 YCC
- Linear version of the luminance-only space mentioned above
- CCIR Rec. 601 YCC
- Luminance-only space derived from CCIR Rec. 601 YCC
- YCC space from Eastman Kodak Company for its Photo CD™ product
- Linear CMY
- Linear CMYK

Lookup Tables

Lookup tables are used in two ways in XIL applications. First, they are used for general image modification. A single-band image of any data type can be converted to a single-band or multiband image of any data type. The correspondence between pixel values in the source and destination images is user defined.

Second, lookup tables are used in dithering operations. In a dithering operation, a single-band or multiband image of any data type is converted to a single-band image of any data type.

Dither Masks

A dither-mask object is one of the required parameters to the XIL library's ordered-dither operator. The mask is a three-dimensional array whose depth matches the number of bands in the image being dithered. The values in the mask act as threshold values that help eliminate contouring in the dithered image.

Kernels

A kernel is one of the required parameters to the convolution operator. The kernel is a two-dimensional array of single-precision floating-point values, one of which is designated as the kernel's key value. When the library performs a convolution operation, it aligns the kernel's key value with the source-image pixel it is currently processing.

The values in the kernel determine the effect that the convolution operation will have. For example, the operation may sharpen an image, smooth an image, or highlight the edges in an image.

Structuring Elements

A structuring element serves as a parameter to the erosion and dilation operators and defines the neighborhood of pixels in the source image to be used in calculating the value of each pixel in the destination image. This structuring element looks like a convolution kernel except that its member values are Boolean.

Compressed Image Sequences

A compressed image sequence is a container for compressed images. These images may represent a series of still images, frames from a movie, or pages from a document, and they may have been compressed in one of several formats:

- Cell (a format developed at SunSoft)
- JPEG baseline sequential or lossless
- CCITT Group 3 or Group 4

Histograms

When an XIL application calculates an image's histogram, it writes its results to a histogram object. This object contains a set of bins that store information about gray or color levels in the image.

Errors

When an XIL error occurs, the library generates an error object. This object contains such information as an error ID and a string containing a description of the error. The XIL library includes a default error handler that gets values from this object and prints a message to `stderr`. It also contains a set of error-handling functions that get information from this type of object. You can use these functions to construct a custom error handler.

System States

A system-state object is created when an application initializes the library. Among other things, this object is used to maintain a list of the data objects the application has created and to keep track of the current error handler.

XIL Functionality

Image Processing Functions

XIL includes a basic set of image processing operations, common to many imaging accelerators or sophisticated framebuffers. Additionally, XIL incorporates the functionality required to support more sophisticated imaging APIs.

Image processing operations include arithmetic and logical operations, geometric operations, and a set of miscellaneous functions including basic morphology and image manipulation.

Arithmetic and Logical Operations

The functions in this class can be used to lighten or darken an image, to increase or decrease contrast in an image, or to produce the negative of an image. Some of these functions require two source images while others require a source image and a constant. The library also contains the unary operator which takes the one's complement of the values in an image

Geometric Operations

XIL's geometric functions provide image scaling, rotation and translation capabilities (as well as a combination thereof, referred to as an affine transform). Using the scaling functions, XIL can scale images while preserving most of the information in the image. This feature provides a powerful mechanism for previewing electronic images such as from scanners and facsimile machines directly on the user's screen.

Compression Functions

XIL provides the ability to compress and decompress images and sequences of images. Compressed image sequences provide a flexible mechanism for applications to store and retrieve contiguous image data. Using SunSoft's Cell format for example, image compression of up to 98% can be achieved, and efficient decompression of the CIS results in a full 30 frames/second (640x480 pixels) on playback on a standard Solaris workstation.

Color Functions

In XIL, color space is an attribute of an image, and can be changed to any one of a number of supported color spaces. Color space conversion in XIL is useful for a number of reasons:

- Some operations are performed most easily on images with certain color spaces. For example, JPEG compression produces better results on color data when the input is in YUV or YC_bC_r format instead of RGB.
- Using nonlinear, or gamma corrected color spaces (such as YCbCr or nonlinear

RGB) can prevent contouring in low-intensity regions of 8-bit images.

- By supporting the CMY and CMYK color spaces, XIL enables the handling of color separation to produce images that can be sent to subtractive color printers.
- The ability to extract the luminance from color data allows the use of monochrome output devices.

Presentation Functions

XIL provides a set of general functions which deal with preparing images for display or for output to some device. These functions perform such tasks as dithering images, manipulating color maps and painting and blending images.

XIL is completely integrated into the OpenWindows window system, providing display to individual X windows with minimum programming on the part of the developer.

The XIE Extension to the X Protocol

The X Image Extension (XIE) provides a wire protocol for support of distributed imaging in the X Window environment. As such, XIE is optimized to support services needed by imaging applications and APIs. Like the X protocol, XIE is the result of a standards effort driven by the MIT X Consortium. SunSoft will offer XIE and XIElib once they become available from the X Consortium.

XIE provides a full range of services for imaging, divisible into subsets based upon requirements of the API or application above. Subsets are arranged in a nested hierarchy of service classes, and applications and APIs can be written to a given subset of the protocol. The initial public review specification of XIE defines two subsets: a full set of service features, and a more limited one providing for document imaging applications. Figure 4.3 illustrates the service classes defined in the current (4.12) specification draft.

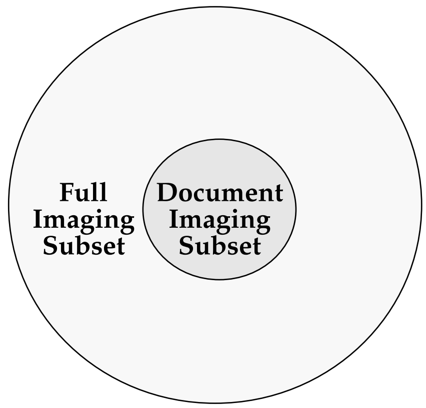

Figure 4.3 X Image Extension Service Classes: Full and Document Imaging Subsets (DIS)

XIElib

Like the X protocol, the XIE protocol defines the syntax, structure and semantics of XIE protocol elements in a language independent manner. For XIE to have broader appeal as an application (or imaging API) interface, a higher-level API with defined language bindings is desirable.

Ongoing work by the MIT X Consortium is taking place to define such an interface, currently referred to as XIElib. XIElib will be a client-side library callable from C and providing the services, features and distributed facilities of XIE under X Windows. SunSoft is participating in the standards effort associated with XIElib and will provide support for it when it becomes generally available.

Remote Rendering using XIE

Distributed rendering with XIE is managed in the same manner as with Xlib and the X protocol. XIElib utilizes the XIE protocol for interprocess communication for rendering. The client application and the X11 server may be on the same or different machines.

Figure 4.4 illustrates how XIE will fit into distributed window systems.

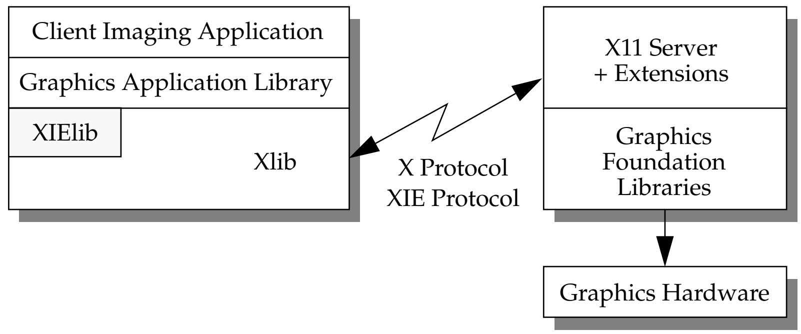

Figure 4.4 Distributed Imaging Application Under X Windows

Third-Party and Standard Imaging Interfaces

The Solaris VISUAL architecture is designed support third-party imaging libraries through the XIL foundation library and the XIE protocol (and library).

A variety of imaging APIs, both third-party and committee-defined are available today, but lack the optimized features of either a common display hardware-optimized foundation library, or a network distributed protocol. The Solaris VISUAL environment provides both of these through XIL and XIE.

The Programmer's Imaging Kernel System (PIKS)

The Programmer's Imaging Kernel System, or PIKS, is the result of almost five years of standards effort on behalf of the International Standards Organization (ISO) and the International Electrotechnical Commission (IEC). PIKS is one part of the Image Processing and Interchange (IPI) standards effort being undertaken by ISO and IEC with cooperation from DIN and ANSI. The IPI standard includes three parts: a Common Architecture for Imaging (CAI), providing an architectural framework for the IPI standard and defines the abstract description of image and non-image data objects; The Programmer's Imaging Kernel System (PIKS), which defines an API for image processing, including operators, tools and utilities; and the Image Interchange Facility (IIP), which specifies the means for transporting images and image-related data objects between applications.

All of the parts to the IPI standard are currently in draft review and will likely become approved standards during 1993. PIKS in particular is important because it provides broad imaging functionality (through an API) to a wide range of application areas, and because it represents the first significant technical image processing standard to emerge.

PIKS provides a rich set of both low-level and high-level services for image and image-related data objects, which can be used as building blocks for a broad range of common imaging applications. PIKS has no inherent display facilities and is independent of system hardware and window systems. PIKS applications must utilize the display facilities of the native window system or display hardware.

The PIKS definition is divided into three areas of increasing complexity: *Foundation*, *Technical* and *Full*. A Foundation implementation provides general imaging functionality approximately equivalent to the facilities in XIL. This level is of broad general appeal to the common imaging requirements of commercial document imaging, publishing and graphic arts. The Technical superset of PIKS includes additional features for complex data types, time sequences of images, multispectral data objects, and other capabilities. This level is of interest to specialized imaging markets such as scientific and medical imaging as well as other technical imaging markets.

The Full definition broadens PIKS to include sophisticated operations such as image object classification, morphology and features required by machine vision applications. This level of PIKS would be utilized by application developers in niche markets, industrial research and development, vision systems, etc.