



# **X WINDOWS**

**Richard Wilkinson**

**TESSELLA SUPPORT SERVICES PLC**

Issue V1.R3.M0  
February 2003



## **Introduction**

X-Windows has become the standard system for graphical user interfaces on UNIX and VMS computers and X-Windows applications may also be displayed on IBM compatible PCs and Apple Macintoshes running X servers. Its strengths are its flexibility, portability and the ease with which it can be used over networks. This is achieved by using a client-server model with a platform independent protocol for interaction between the X display server and client applications.

## **The X Windows System**

The X-Windows system consists of an X server and any number of client applications, which may be running on the same or different hardware. The X server controls the display and monitors the keyboard and mouse for user input. It may run on a standard workstation or a dedicated X-terminal. Applications send display requests to the server and the server directs events (key presses, mouse movements and button presses, display related events such as the exposure of a previously concealed window and communication events generated by applications) to the appropriate application. The window manager influences the layout and “look and feel” of the X display. This is a special client application that controls aspects of the state of windows such as position and size, and determines “input focus”, ie which window and application receives keyboard input. The other clients are the interactive applications that users run.

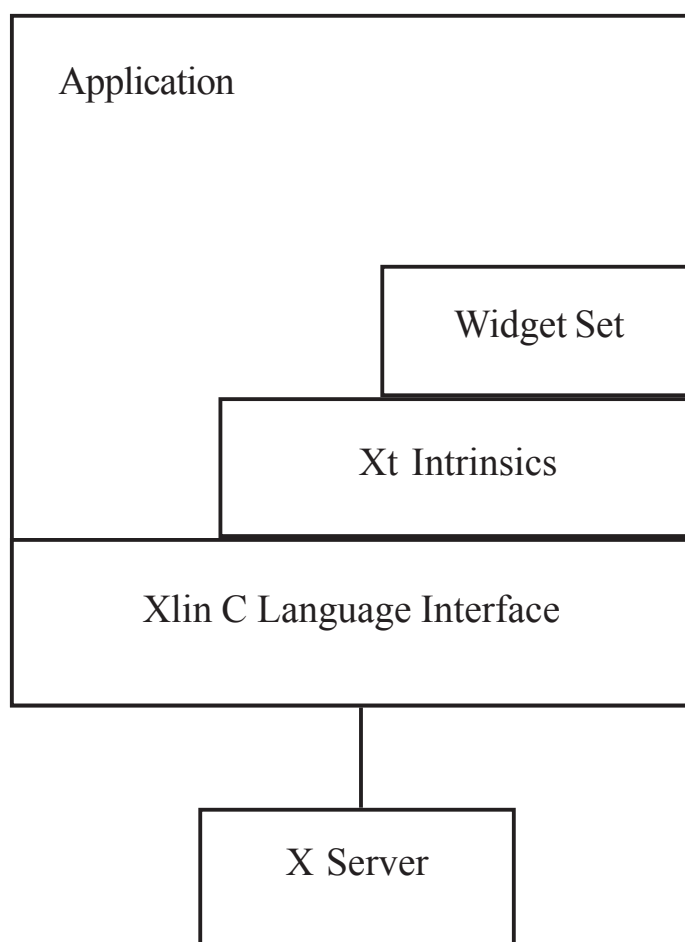
## **X-Windows Programming**

Programmers using X usually base their applications on libraries that provide an interface to the base window system. Xlib is the standard library written in C. It consists of a set of functions that provide access to and control of the display, windows and input devices. There are functions to enable drawing of areas, lines and text. There is a large choice of fonts and there are several different ways of controlling colour. Drawing is in 2D integer co-ordinates that refer to individual pixels. A number of sophisticated operations may be used to affect the generation and modification of the raster image. Drawing is directed to windows; these can be created, moved, reshaped and destroyed. There are also functions to receive events that have been queued for the application by the X server so that they can be acted on.

The Xlib library works at a very low level and can be tedious and difficult to use. Programmers generally prefer to use one of the higher level toolkits designed to be used with Xlib. The main body of application programming is carried out using a widget set with frequent use of a library that interfaces to Xlib called the Xt Intrinsics. The architecture of an application based on a widget set and the Xt Intrinsics is shown in figure 1. Both the Xt Intrinsics and the widget set are written in C and built on top of

Xlib. The widget set is essentially a library of pre-programmed graphics routines. It contains routines for creating and controlling user interface components such as menus, pushbuttons, text fields, labels, scrollbars and window frames, together with layout control widgets. The Xt Intrinsic provide a framework that allows the application programmer to combine these components to produce a user interface.

Each widget (a scrollbar, a pushbutton etc.) is a member of a widget class. The basic behaviour of the widget, and what kind of data is stored in it, is defined by its class. The widgets can be put together to form an application interface as in the following example: a text widget (a viewspace allowing text reading or text entry) and two scrollbar widgets (fitting on one side and underneath the text widget and allowing scrolling through the text) could make a simple interface (see figure 2).



*Figure 1: The X-Window System Architecture*

Each widget can be thought of as an object in object oriented programming parlance.

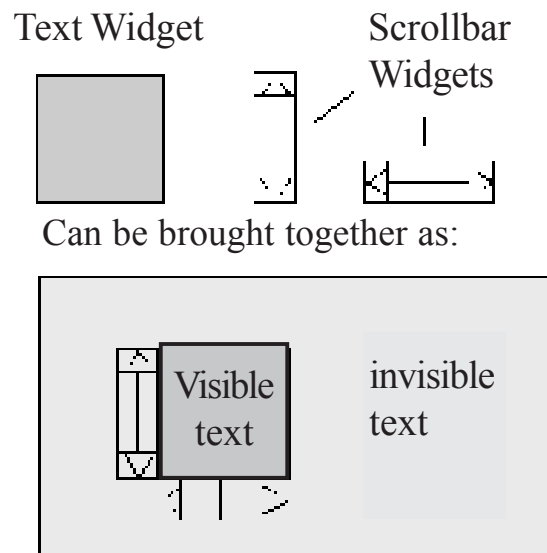
Widgets inherit features from classes higher up in the (object oriented) tree, providing a consistency that makes for easier programming. Associated with this is the idea of resources, which can be supplied for whole classes of widgets and for individual widgets. Resources are data defining the appearance or behaviour of widgets, such as colours and fonts, and they can be supplied in resource files, which may be modified to customise an application according to a user's preferences.

### Widget Sets

There are three main reasons for using widget sets:

- ❑ From the user's point of view they provide consistency between applications (especially if user interface guidelines are followed).
- ❑ From the programmer's point of view they are straightforward to use, greatly simplifying the job of creating standard user interface components.
- ❑ They are portable (open systems), which is advantageous from both the user's and programmer's point of view.

There are several toolkits (widget sets) on the market but the two main ones are the Open Software Foundation (OSF), Motif and the OPEN LOOK Intrinsic Toolkit (OLIT) widget sets. The features provided by these are similar.



*Figure 2: A text widget and two scrollbar widgets can make a simple application interface*

OLIT started life as a user interface specification developed by SUN micro-systems

with AT & T backing and was widely circulated for comment before any implementation was done. It was intended to be implementation independent and implemented separately by many vendors.

Digital and Hewlett-Packard, both OSF sponsors, originally developed motif. The look and feel of the widget set was proposed by HP/Microsoft and is designed to emulate the look and feel of the IBM/Microsoft Presentation Manager. The Application Programmers Interface (API) is based on DEC windows. Some underlying enhancements and supporting utilities to the Xt Intrinsics were also provided by Digital.

When comparing the Motif and OLIT widget sets it is important to keep in mind that the look and feel of an application is controlled not only by the widget set but also by the window manager. Window managers that complement the widget sets are available from both suppliers.

There are advantages and disadvantages to each of the toolkits but the choice is likely to be controlled by factors such as availability - Motif is available on all common UNIX and VMS platforms.

### **Application Development and GUI Designer Tools**

A typical, straightforward X-Windows application will contain the following elements:

- Function calls to create and display widgets, and to set resources.
- Definitions of callback routines: these are called to take actions in response to events, e.g., to terminate an application if an Exit menu option is selected.
- A main event loop in which events are received and the appropriate callbacks are called.

The layout of a window is usually controlled by “manager” widgets that are used to align, or otherwise position, a number of widgets relative to one another. The separation of code into display definition and callback parts is good for program maintainability and facilitates the building of new user interfaces for old applications, since it may well be possible to re-use much of the old code in the callbacks without major modification. It also facilitates the use of GUI design tools to generate the display definition code.

The need for tools that allow the programmer to interactively design user interfaces soon becomes apparent when one writes an application with more than a few widgets. If the application requires several buttons and a host of other widgets (scrollbars, view

areas, menus etc.) then the code can become quite involved and writing it is very repetitive. In most cases explicit positioning for each widget is also required. This in itself can become complicated and frustrating.

Widget set development tools provide the quickest way to set out a widget framework, i.e., to position the widgets in desired places and to produce the code for the widgets and callback routines. All the programmer then has to do is supply the functional code required when pressing a button, moving a scrollbar etc. (i.e., the code in, and called from, the callback routines).

The tools are usually used in the following way: the desired user interface is built up by dragging the chosen widgets (buttons, menus, etc.) from a palette to the required positions until the interface is finished. The tool generates code in C or C++, which can be modified and extra functions added. Code generated by user interface design tools can be stored separately from the application programmer's own additions to the code. This allows for easy updating and revising of the application. Such tools also encourage a modular approach to programming applications, which is particularly important when designing and writing an interface using object oriented techniques (probably using C++). This modular approach again helps in maintenance.

Design tools allow inexperienced user interface programmers to learn whilst quickly producing a useful product. Meanwhile the experienced interface programmer can use the tools to produce the frame of the application quickly whilst concentrating on the more complicated aspects of X-Windows programming. Claims have been made that use of an application development tool can reduce the time spent on coding by up to 90%. While this figure may apply only to the "cleanest" of applications (with no complicated X calls) it is certainly worth considering the use of a development tool for medium to large applications or if several small applications are to be developed. They are also very useful if several prototype designs are to be produced.

A number of design tools are available for Motif, OLIT and other widget sets such as XView. Examples are UIM/X and X-Designer for Motif, and GUIDE for OLIT (these lists are by no means exhaustive).

### **Third Party Widgets**

While the main widget sets provide most of what is required for many applications, there are cases where it is better to use additional widgets. These cases would include:

- Improved manager widgets - the facilities for aligning widgets in the standard manager widgets are often insufficient when the sizes of the widgets are unpredictable or will be re-sized.
- Widgets that perform commonly needed functions, such as text entry widgets that

perform validation of entered data.

❑ Specialised widgets such as 3D graph widgets.

Add-on widgets such as the XRT widgets from the KL Group may well be worth considering rather than writing new widgets from scratch or putting the required functionality into extra callback routines.

### **X-Windows Graphics**

There are a number of quite different approaches that can be taken to display graphics in an X-Window application. It is possible to use primitive Xlib functions to perform simple drawing and to write functions to scale co-ordinates, perform geometric transformations and allow the picking of points with the mouse. Moreover, Xlib allows a variety of functions for manipulating the rasterised image that may not be provided by the alternatives. For example, one can create images that respond very rapidly such as a “rubber band” which selects portions of the image for enlargement. A more sophisticated example is a ghostly outline of a 3D image, which the user rotates until satisfied with the orientation. This outline will be much more responsive than a constantly re-drawn full 3D image. PEX is a graphics extension to X and is very closely related to the PHIGS standard. It was released by MIT with X11 R5. PHIGS, and now PEX, is effectively a database in which the components of a picture are stored: primitives such as lines and text, rendering instructions to give colour and style, transformations such as projections and rotations, etc. The programming interface permits the data to be loaded, organised, modified and reorganised, allowing dynamic control over the picture. The graphics system can also produce advanced rendering effects such as hidden surface removal or shading.

The PEX programming functions are very similar to those of PHIGS, but the PEX standard also makes statements about the underlying system. The organisation of PEX is analogous to that of X: the programmer’s interface transmits protocols to a PEX server. This server runs beside the X server and is able to take advantage of the graphics hardware to optimise the performance of a display.

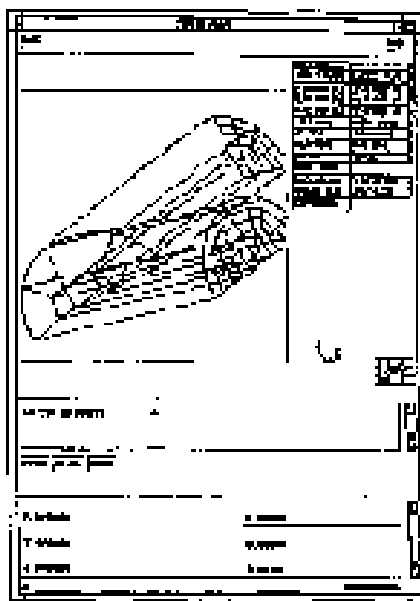
PEX offers distinct advantages over X. First, its database holds graphics information in a form that can be modified and reordered by function calls. X has no memory of what it has drawn, other than the raster image itself. Secondly the user can opt for a co-ordinate system based upon real values of any scale, rather than addressing pixel values. Thirdly the PEX system provides many impressive components of 3D rendering such as hidden surface removal and shading which are often essential to 3D imaging but are difficult to produce in-house. Lastly, implementations of PEX are likely to make use of the hardware’s graphics accelerators, which leads to a dramatic

improvement in performance. Of course a PEX-based program would still make use of X or X toolkit calls to create the GUI and gather input from the user.

The design of PEX and its integration into the X system confers important advantages over PHIGS. In particular, the PHIGS standard does not consider the creation and control of windows for the display of images, which has caused the proliferation of implementation specific extensions to PHIGS. By contrast PEX is designed to display in windows which are created by X. Further PEX uses the client-server architecture that is familiar in X, thus conferring portability and network transparency.

Alternative graphics libraries are GINO, OpenGL, and HOOPS. If the facilities of these are insufficient it may be better to integrate the application with a data visualisation tool such as AVS or PV-Wave.

Figure 3 shows the GUI for the CFDS, AEA Technology grid generator. This is a specialised CAD tool, which is used to describe 3D geometries in which the flow of fluids or gases can be simulated. There is a matching Visualisation tool which displays simulation results as contours, vectors etc. The model is a section of a junction between pipes. The image is manipulated in 3D with the buttons to its right. Below the image window are: a message display window; a process control area with buttons; and a data entry area. The menu in the illustration offers the means of entering 3D points, which will be used to build more geometry. Tessella has made a substantial contribution to this application. The graphics and GUI are created with Xlib and the Motif widget set.



*Figure 3: A GUI developed for CFDS  
(By kind permission of CFDS.)*

## **X-Windows and Microsoft Windows**

Because of the prevalence of IBM compatible PCs and the desirability of having applications that are consistent across platforms, it is worth considering the options for developing applications that can run under X-Windows and Microsoft Windows.

This can be done by having a single application on a server and displaying via window systems on users' workstations or by using a development technique that generates applications for both types of platform. The resulting possibilities include:

- 1 Write the application for MS Windows and run it on an NT server together with special display clients on X-Windows (and PCs). An example of this type of system is NTrigue.
- 2 Write the application for MS Windows and use a port of Windows to UNIX (e.g., MainWin or Wind/U).
- 3 Write the application for X-Windows and run an X emulator (such as Hummingbird eXceed) on a PC. Replacement Motif libraries are available to give the application a MS Windows look and feel, (e.g., the IXI Wintif library).
- 4 Write the application using a development tool that supports both X-Windows and MS Windows. Examples include zApp, SUN's ProWorks/Visual XP and X-Designer. Generally these provide a set of functionality that can be reproduced under both window systems and will generate versions of applications that use the native window system of the platform. It is often preferable to use a solution that gives users the look and feel they expect for the platform they are using. This is possible with approaches 2 or 3.

## **Conclusions**

X-Windows is a powerful and flexible window system, which achieves its flexibility through its multi-layer hierarchy of programming libraries and tools. To make the best use of it the correct mix is required of Xlib and widget toolkit programming, GUI design tools, additional components such as extra widgets or graphics libraries and environmental aspects such as window managers and PC X servers.

**Tessella Support Services plc**  
**Creating Software for Science and Engineering**

Tessella's services range from feasibility studies, through system design, development, implementation and ongoing support. Our expertise includes:

Data Analysis Software  
Data Capture  
Simulation Software  
Advanced Graphics  
Systems Support  
Database Applications

**Other Technical Supplements available include:**

- |   |  |
|---|--|
| <input type="checkbox"/> Archiving of Electronic Info   | <input type="checkbox"/> Object Oriented Programming   |
| <input type="checkbox"/> Active Server Pages            | <input type="checkbox"/> Pocket PC                     |
| <input type="checkbox"/> Automated GUI Testing          | <input type="checkbox"/> Portable GUI Development      |
| <input type="checkbox"/> Bayesian Statistics            | <input type="checkbox"/> Printer Technology Guide      |
| <input type="checkbox"/> Beowulf Clusters               | <input type="checkbox"/> Real Time Systems             |
| <input type="checkbox"/> C++                            | <input type="checkbox"/> Regression Testing            |
| <input type="checkbox"/> Client-Server Technology       | <input type="checkbox"/> Security and the Internet     |
| <input type="checkbox"/> COM                            | <input type="checkbox"/> Simulation                    |
| <input type="checkbox"/> Computational Fluid Dynamics   | <input type="checkbox"/> Soft Computing                |
| <input type="checkbox"/> Computer Image Processing      | <input type="checkbox"/> Software Design Methodologies |
| <input type="checkbox"/> Decision Support Systems       | <input type="checkbox"/> Software Development Cycle    |
| <input type="checkbox"/> Electronic Data Capture        | <input type="checkbox"/> Software Documentation        |
| <input type="checkbox"/> Electronic Lab Notebooks       | <input type="checkbox"/> Software Portability          |
| <input type="checkbox"/> Excel                          | <input type="checkbox"/> Software Re-engineering       |
| <input type="checkbox"/> Extending the Life of Software | <input type="checkbox"/> Software Specification        |
| <input type="checkbox"/> Federal Drug Administration    | <input type="checkbox"/> SQL                           |
| <input type="checkbox"/> FORTRAN 90                     | <input type="checkbox"/> UNIX Inter-Process Comms      |
| <input type="checkbox"/> Grid Computing                 | <input type="checkbox"/> UNIX Systems Performance      |
| <input type="checkbox"/> High Throughput Screening      | <input type="checkbox"/> UNIX Workstations             |
| <input type="checkbox"/> Instrumentation                | <input type="checkbox"/> Visual Basic 6                |
| <input type="checkbox"/> Integrated Lab Systems         | <input type="checkbox"/> WAP                           |
| <input type="checkbox"/> J2EE                           | <input type="checkbox"/> Web Services                  |
| <input type="checkbox"/> Java                           | <input type="checkbox"/> Windows 2000 Services         |
| <input type="checkbox"/> Lims                           | <input type="checkbox"/> XML                           |
| <input type="checkbox"/> Linux                          | <input type="checkbox"/> X Windows                     |
| <input type="checkbox"/> Microsoft Net                  |  |



INVESTOR IN PEOPLE

**Tessella Support Services plc**

3 Vineyard Chambers, Abingdon, Oxon, OX14 3PX, England

Tel: (+44) (0) 1235 555511 Fax: (+44) (0) 1235 553301

E-mail: [info@tessella.com](mailto:info@tessella.com) Web Address: <http://www.tessella.com>