1 Introduction

The X Windows environment, also known as xlib and X Windows, is a cross-platform windowing environment developed at MIT in the 80’s and 90’s. It is an attempt to create a specification for a graphical windows environment that hardware and software vendors can follow. Currently, almost all flavors of Unix support X, and there are even X add-ins for Microsoft Windows and Macintosh.

X works as a client-server protocol. The client and server in X are the opposite of what you might encounter in other similar protocols. The way X works is as follows. The client sends X messages to the server. The server then draws (or paints) a window on the computer screen. User input, such as key presses and mouse movement is processed by the window the window that is currently open (by the server), and that user input is then sent back to the client. This loop continues until the program terminates. Figure 1 illustrates this concept.

Often, the client and server will be the same physical machine. When you sit down at a Sun workstation, for example, and log in and run a program, the client and server are both that physical machine. You may however find yourself in a position where you’re doing your work on a remote machine, but displaying the graphical X results on a different machine. In this case, the client is the remote machine, and the server is the machine you’re seeing the graphic results on.

2 Structure of an X Program

In order to actually send drawing commands to an X Window, there are a few steps to take care of first. Accordingly, the first step in an X program is to initialize the X connection. Remember, X is a protocol, not just an API (application programming interface). Since there is communication that must take place between two computers (which may be the same computer), an initialization has to take place. Next, windows must be opened on the display machine (server). After the window is opened, the programmer must specify which X messages this particular window should process. Finally, the main loop is entered, which
Figure 1: The client sends drawing commands to the server, and the server sends user input to the client.

The client sends drawing commands to the server, and the server sends user input to the client.

consists of basically 2 steps - process X commands, and do something that responds to those commands, such as redraw the window. Figure 2 shows the basic flow of an X program.

3 Parts of an X Display

There are 3 parts of an X display. Each part is initialized and then used in subsequent parts of the X program.

3.1 Display

Display is an X struct which encapsulates the parameters of the current X server. Remember, by server, we mean the computer on which the graphics will be displayed. A Display pointer is declared, and is initialized with a call to XOpenDisplay, as follows:

```c
Display *display;
...
display=XOpenDisplay(NULL);
```

3.2 Window

A Window is the actual window that appears on the X server. There are a number of ways to create an X window. The simplest is to use the function XCreateSimpleWindow function. This returns a handle (of type Window) to the newly created X window. For example:
Figure 2: A typical X application first initializes communication, opens an X window, and goes into the main loop.
Window w;
...
w=XCreateSimpleWindow(display,DefaultRootWindow(display),0,0,
    200,100,0,white,white);

### 3.3 Graphics Context

Now that you have the window, you need to get a handle to the canvas - that is, you need to get a handle to the location where you actually do the drawing. This area is called the graphics context. You obtain it with a call to the function XCreateGC. This function returns an X struct GC. For example:

```c
GC gc;
  gc=XCreateGC(display,w,0,NULL);
```

### 4 A sample X Program

Let’s take a look at a complete X program. This program simply sets up all of the X constructs and then simply draws a line in a window.

```c
#include <X11/Xlib.h>
#include <unistd.h>
#include <stdio.h>

#define bool char
#define true 1
#define false 0

bool Running;
Display *display;
Window w;
GC gc;
int width,height;

void ProcessEvent(XEvent e);
void Draw();

main()
{
  int black,white,Notify;
  display=XOpenDisplay(NULL);
  black=BlackPixel(display,DefaultScreen(display));
  white=WhitePixel(display,DefaultScreen(display));
  Notify=0;
}
Notify|=StructureNotifyMask;
Notify|=KeyPressMask|ButtonPressMask;

w=XCreateSimpleWindow(display,DefaultRootWindow(display),0,0,
    200,100,0,white,white);
if(display)
{
    XSelectInput(display,w,Notify);
    XMapWindow(display,w);
    gc=XCreateGC(display,w,0,NULL);
    XSetForeground(display,gc,black);
    //wait for window map notify
    Running=true;
    while(Running)
    {
        XEvent e;
        XNextEvent(display,&e);
        ProcessEvent(e);
        Draw();
    }
}

void ProcessEvent(XEvent e)
{
    switch(e.type)
    {
    case ButtonPress:
        printf("button press\n");
        break;

    case ConfigureNotify:
        width=e.xconfigure.width;
        height=e.xconfigure.height;
        printf("resized to %d x %d\n",width,height);
        break;

    case KeyPress:
        printf("key pressed\n");
        Running=false;
        break;
    }
}

void Draw()
{

XDrawLine(display,w,gc,10,60,180,20);
XFlush(display);
}