

## **X lib Programming**

(lecture programs)

- **X server:** Controls the Input/Output Resources of a host:

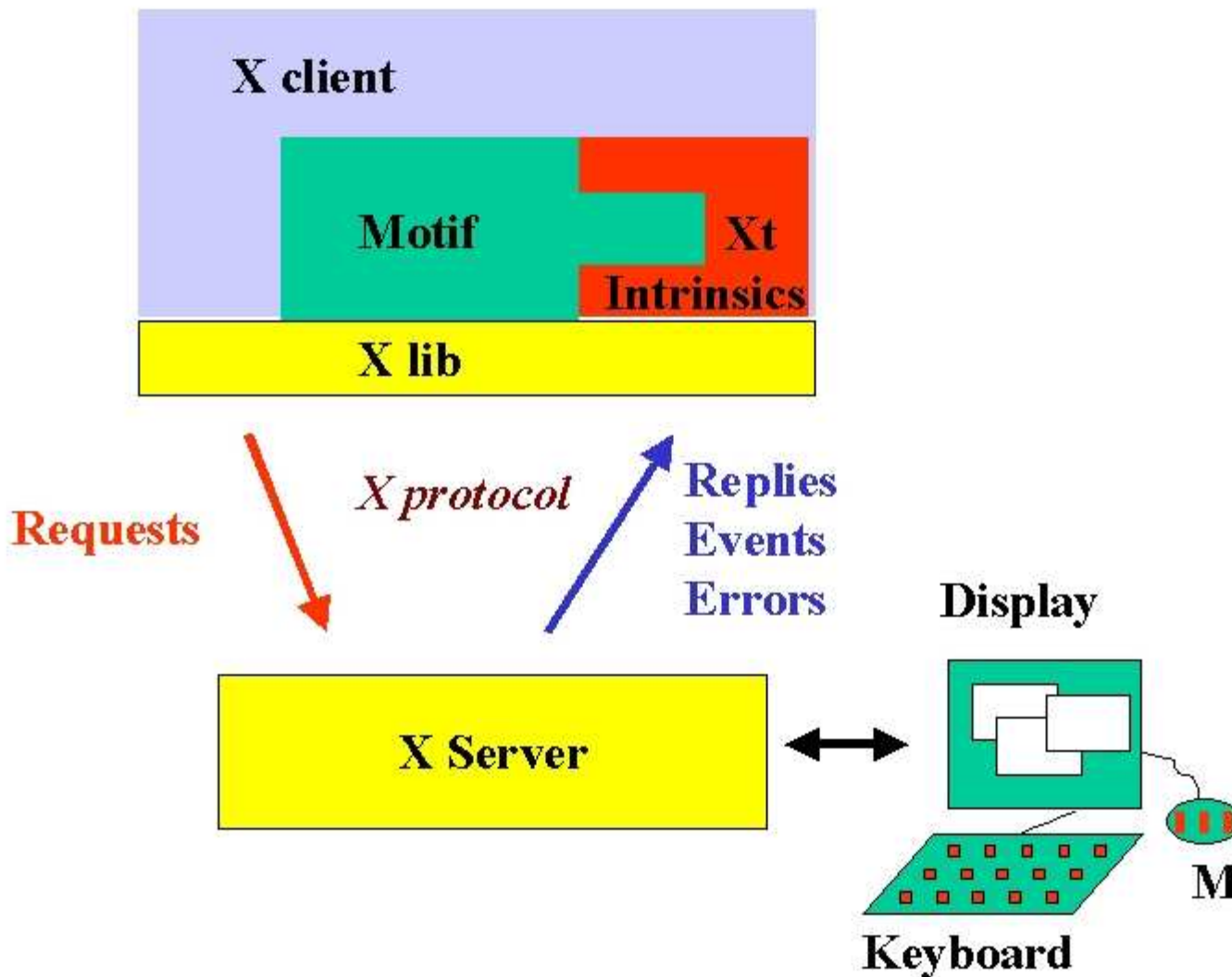
**Display, Keyboard and Mouse.**

- **X clients:** Applications that runs at any host in the Internet:

May be different from the X server's host.

- **TCP/IP:** Is used for communications between the clients and the server:

The default **port#** for the X server is **6000**.



### Running X clients on remote hosts

To run an X client (e.g., `xterm`) on a remote Unix host (e.g., `cash.cs.odu.edu`) and display the interface on your local window machine (e.g., `128.82.5.88`):

- Run the X server on your local window machine.
- Use ssh to login to `cash.cs.odu.edu`
- `% setenv DISPLAY 128.82.5.88:0`
- `% xterm`

---

## Examples of X lib Programs

### Example 1: Drawing Circles [xcircles.c](#)

```
main(argc,argv)
int argc;
char **argv;
{
    Display *display;
    Window root, window;
    long fgcolor, bgcolor;
    int screen, pointx, pointy;

    long eventmask = ButtonPressMask |
ExposureMask | KeyPressMask;

    XEvent event;
    XGCValues gcval;
    GC draw;
    Colormap cmap;
    XColor color, ignore;

    char *colorname = "red";

    int radius = 6;
```

The above are definitions that will be used throughout the program.

---

**Open Display**

```
if (!(display = XOpenDisplay (argv[1]))) {
    perror("XOpenDisplay");
    exit(1);
}
```

- Opens a TCP connection to an X server running at the host specified by `argv[1]`.
- If `argv[1]` is `NULL`, it contacts the server running at the *DISPLAY* machine.
- The format for `argv[1]` is: `host:0`

### Examples:

```
% xcircles
% xcircles 128.82.5.88:0
% xcircles necromancer.cs.odu.edu:0
```

---

### Create Root Window

---

```
root = RootWindow (display, screen =
DefaultScreen(display));
```

- In X every window must have a *parent* and this **root** is the parent of all other windows.

---

### Create Window

---

```
fgcolor = BlackPixel (display,screen);
bgcolor = WhitePixel (display,screen);
```

- Obtains the pixel values for the black and white colors.

```
    window = XCreateSimpleWindow (display,  
root, 0,0, 200,200, 2,
```

```
fgcolor, bgcolor);
```

- Creates the application main window on display as child for root at position 0,0.
- The window size is 200x200 with border of 2 pixels.
- The window's foreground color is black and its background color is white.

---

### Create Drawing Pen

---

```
char *colorname = "red";
```

```
    cmap = DefaultColormap (display,  
screen);  
    XAllocNamedColor (display, cmap,  
colorname, &color, &ignore);  
    fgcolor = color.pixel;  
    gcval.foreground = fgcolor;  
    gcval.background = bgcolor;  
    draw = XCreateGC  
(display, window, GCForeground | GCBackground,  
&gcval);
```

- The above statements are used to create a "red" pen called draw

---

### Select Input Events

---

```
XSelectInput (display, window,  
eventmask);
```

- Ask the server to report the events specified by eventmask

```
XMapWindow (display,window);
```

- Make the window visible on the screen.

### Handle Input Events

The following loop monitors and process the events sent by the X server

```
for (;;) {  
    XWindowEvent (display, window,  
eventmask, &event);
```

- This is a "blocking" call, i.e., the program will stop here until an event arrives from the X server.

```
switch (event.type) {  
  
    case Expose:  
  
        XClearWindow (display,window);  
        break;
```

- Whenever an **Expose** event arrives, the window is cleared. An expose event can be generated by e.g., covering and uncovering the window, closing and opening the window.

```
case ButtonPress:
```

```
    pointx = event.xbutton.x -  
radius;  
    pointy = event.xbutton.y -  
radius;  
    XFillArc(display, window,  
draw, pointx, pointy,  
2*radius, 2*radius,  
2*radius,0, 360*64);  
    break;
```

→ Whenever any **Button is Pressed** a **red** point is drawn at the x,y position where the event occurred.

```
case KeyPress:  
    exit(0);
```

→ Whenever any **Key is pressed** the program exits.

```
default:  
    fprintf(stderr, "Unexpected  
event: %d\n", event.type);
```

→ Any other event is unexpected and should not happen.

---

**Example 2:** *Drawing Lines* [xlines.c](#)

The program [xlines.c](#) is similar to [xcircles.c](#) but it draws lines.

The user **odd clicks** (1, 3, ...) draws a point while the **even clicks** (2, 4, ...) draws lines between the **current position** and the **previous position** of the mouse.

Here is the code that achieves that:

```
case ButtonPress:
    if (FirstPt) {
        FirstPt=FALSE;
        pointx = event.xbutton.x;
        pointy = event.xbutton.y;
        XDrawPoint (display,window,draw,
pointx, pointy);
        break;

    }
    else {
        FirstPt=TRUE;
        XDrawLine (display,window,draw,
pointx,pointy,
event.xbutton.x,
event.xbutton.y);
        break;
    }
}
```

→ Odd clicks draw a **point** while even clicks draw a **line** between the **previous** mouse position and the **current** position.