Examples:

**Hello:**

prints “hello Hussein” in the middle of the window at position: line 10, column 20.

```c
#include <curses.h>
main()
{
    initscr();
    clear();
    move(10,20);
    addstr("Hello Hussein");
    refresh();
    endwin();
}
```

**Motion:**

prints “hello Hussein” in the middle of the window and move it across the line from position: line 10, column 0 to column 30.

```c
main()
{
    int i;
initscr();
clear();
for (i=0; i<30;i++){
    move(10,i);
    addstr("Hello Hussein");
    refresh();
    usleep(100000); /* sleep 100 milli sconds */
    move(10,i);
    addstr("     ");
}
getch();
endwin();
}```
moved

prints “hello Hussein” in the middle of the window and move it across the line from position: line 10, column 0 to column 30. If the user type “z”, it move to next line.

```c
main()
{
    int i;
    int row;
    char ch;
    initscr();
    clear();
    nodelay(stdscr, TRUE); /* non block getch()*/
    row=10;
    noecho(); /* stop echo of input */
    for (i=0; i<30;i++){
        move(row,i);
        addstr("Hello Hussein");
        refresh();
        usleep(100000);
        move(row,i);
        addstr("            ");
        ch=getch();
        if (ch == 'z') {
            row++;
            i=0;
        }
    }
    getch();
    endwin();
}
```
X lib Programming
(lecture programs)

**X server:** Controls the input/output resources of a host:
*Display, Keyboard & Mouse.*

**X clients:** Applications that runs at any host in the Internet:
May be different from the X server's host.

**TCP/IP:** used for communications between clients and 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., fast.cs.odu.edu) or Linux host (e.g., linux.cs.odu.edu) and to display the interface on your local window machine (e.g., 128.82.5.154):

1. Run the X server on your local window machine.
2. Use ssh to login to fast.cs.odu.edu or linux.cs.odu.edu
   % setenv DISPLAY 128.82.5.154:0
   % xterm
   OR
   click on Settings of the ssh button (3rd from last)
   click on Tunneling
   check on the Tunnel X11 connections.

Examples of X lib Programs

Example 1: Drawing Circles \texttt{xcircles.c}

\begin{verbatim}
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 radious = 6;

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

    Open Display

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

    :: Opens a TCP connection to an X server running at the host specified by \texttt{argv[1]}.
    :: If argv[1] is NULL, it contacts the server running at the \texttt{DISPLAY} machine.
    :: The format for argv[1] is: \texttt{host:0}
\end{verbatim}
Examples:

% who am i
cs476 pts/4 2011-09-01 14:36 (dhcp-154.cs.odu.edu)
% host dhcp-154.cs.odu.edu
dhcp-154.cs.odu.edu has address 128.82.5.154

% xcircles
OR
% xcircles 128.82.5.154:0
OR
% xcircles dhcp-154.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 "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 generated by e.g., covering/uncovering window, closing/opening window.

    case ButtonPress:
        pointx = event.xbutton.x - radious;
        pointy = event.xbutton.y - radious;
        XFillArc(display, window, draw, pointx,
Whenever any Button is Pressed a red point is drawn at the x,y position where the event occurred.

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

Whenever any Key is pressed the program exits.

```c
default:
    fprintf(stderr,"Unexpected event:");
```

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.

Odd clicks (1, 3, ...) draws a point
Even clicks (2, 4, ...) draws line between current and previous mouse positions.

Here is the code that achieves that:

```c
case ButtonPress:
    if (FirstPt) {
        FirstPt=FALSE;
        pointx = event.xbutton.x;
        pointy = event.xbutton.y;
        XDrawPoint (display,window,draw, pointx, pointy);
    } else {
        FirstPt=TRUE;
        XDrawLine (display,window,draw,pointx,pointy,
                    event.xbutton.x, event.xbutton.y);
    }
```
Motif Programming
(lectures programs)

Example 1  xecute.c:
Executes any typed *command* after the user confirms the action.
The interface is composed of: A container widget: *BulletinBoard* Widget, that contains two other widgets:

1. *Label* Widget
2. *PushButton* Widget

Example Usage:

```
% xecute "list all files?"  "ls -lt"
```

This produces the following interface:

![Diagram of xecute.c interface]

**xecute.c outline:**

```c
void main ( int argc, char **argv )
{
    Widget         shell, msg, bb, yes, no;
    XtAppContext   app;
    XmString       xmstr;
    Dimension      height;

    shell = XtAppInitialize( &app, "Xecute", NULL, 0, &argc, argv,
                                NULL, NULL, 0 );

    bb = XtVaCreateManagedWidget
        ("bboard", xmBulletinBoardWidgetClass,shell,NULL );
```
xmstr = XmStringCreate ( argv[1], XmFONTLIST_DEFAULT_TAG );

// Create msg, yes and no widgets
msg = XtVaCreateManagedWidget ( "message", xmLabelWidgetClass, bb, XmNlabelString, xmstr, XmNx, 0, XmNy, 0, NULL );
yes = XtVaCreateManagedWidget ( "yes", xmPushButtonWidgetClass, bb, XmNx, 0, XmNy, 30, NULL );
no = XtVaCreateManagedWidget ( "no", xmPushButtonWidgetClass, bb, XmNx, 200, XmNy, 30, NULL );

// Add callback functions to yes and no widgets
XtAddCallback ( yes, XmNactivateCallback, YesCallback, ( XtPointer ) argv[2] );
XtAddCallback ( no, XmNactivateCallback, NoCallback, NULL );
XtRealizeWidget ( shell );
XtAppMainLoop ( app );

} void YesCallback ( Widget w, XtPointer clientData, XtPointer callData )
{
char * cmd = (char *) clientData;
system ( cmd );
exit ( 0 );
}

} void NoCallback ( Widget w, XtPointer clientData, XtPointer callData )
{
exit ( 0 );
}
Example 2: A drawing editor: `editor.c`

```c
void main ( int argc, char **argv )
{
    Widget shell, canvas, panel, commands, options;  
    XtAppContext app;
    shell = XtAppInitialize( &app, "Editor", NULL, 0, &argc, argv, NULL,
                           NULL, 0 );
    panel = XtCreateManagedWidget ( "panel", xmFormWidgetClass, shell,
                                    NULL, 0 );
    commands = XtVaCreateManagedWidget ( "commands", xmRowColumnWidgetClass, panel,
                                          XmNnumColumns, 3,
                                          XmNorientation, XmHORIZONTAL,
                                          XmNtopAttachment, XmATTACH_FORM,
                                          XmNrightAttachment, XmATTACH_FORM,
                                          XmNleftAttachment, XmATTACH_FORM,
                                          XmNbottomAttachment, XmATTACH_NONE, NULL );
```
options =
XtVaCreateManagedWidget ( "options", xmRowColumnWidgetClass, panel,
XmNorientation, XmVERTICAL,
XmNtopAttachment, XmATTACH_WIDGET,
XmNtopWidget, commands,
XmNrightAttachment, XmATTACH_NONE,
XmNleftAttachment, XmATTACH_FORM,
XmNbottomAttachment,XmATTACH_FORM, NULL );

canvas =
XtVaCreateManagedWidget ( "canvas", xmDrawingAreaWidgetClass, panel,
XmNtopAttachment, XmATTACH_WIDGET,
XmNtopWidget, commands,
XmNleftAttachment, XmATTACH_WIDGET,
XmNleftWidget, options,
XmNrightAttachment, XmATTACH_FORM,
XmNbottomAttachment,XmATTACH_FORM, NULL );
XtCreateManagedWidget ( "button1", xmPushButtonWidgetClass, Commands,
NULL, 0 );
XtCreateManagedWidget ( "button2", xmPushButtonWidgetClass, commands,
NULL, 0 );
XtCreateManagedWidget ( "button3", xmPushButtonWidgetClass, commands,
NULL, 0 );
XtCreateManagedWidget ( "button1", xmPushButtonWidgetClass, options,
NULL, 0 );
XtCreateManagedWidget ( "button2", xmPushButtonWidgetClass, options,
NULL, 0 );
XtCreateManagedWidget ( "button3", xmPushButtonWidgetClass, options,
NULL, 0 );
XtRealizeWidget ( shell );
XtAppMainLoop ( app );

<table>
<thead>
<tr>
<th>Resource files:</th>
<th>Editor &amp; Editor.color</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>options</em>button1*labelString:</td>
<td>Option 1</td>
</tr>
<tr>
<td><em>options</em>button2*labelString:</td>
<td>Option 2</td>
</tr>
<tr>
<td><em>options</em>button3*labelString:</td>
<td>Option 3</td>
</tr>
<tr>
<td><em>commands</em>button1*labelString:</td>
<td>Command One</td>
</tr>
<tr>
<td><em>commands</em>button2*labelString:</td>
<td>Command Two</td>
</tr>
<tr>
<td><em>commands</em>button3*labelString:</td>
<td>Command Three</td>
</tr>
</tbody>
</table>

! Color resources.

| *foreground: | white |
| *XmDrawingArea*background: | khaki |
| *XmDrawingArea*height: | 500 |
| *XmDrawingArea*width: | 400 |
| *options*background: | Blue |
| *commands*background: | Green |
| *options*XmPushButton*background: | Red |
| *commands*XmPushButton*background: | grey50 |
To associate the resource file with the program before execution, use:

```bash
% setenv XENVIRONMENT Editor
% editor
```

To get colors:

```bash
% setenv XENVIRONMENT Editor.color
% editor
```

To use no resource file:

```bash
% setenv XENVIRONMENT
```

Adding life to editor:

**Example 3: editor2.c  Quit & Draw Squares**

```c
Widget quit;
Display *display;
int screen;
long fgcolor, bgcolor;
XGCValues gcval;
GC draw;
Window window;
int pointx, pointy;
int FirstPt = TRUE;

void main ( int argc, char **argv )
{
    .......
    code here is almost the same as editor.c with the changes below.
    .......
    change:
    XtCreateManagedWidget
    ("button1",xmPushButtonWidgetClass,commands,NULL, 0);
    XtCreateManagedWidget
    ("button2",xmPushButtonWidgetClass,commands,NULL, 0);
    XtCreateManagedWidget
    ("button3",xmPushButtonWidgetClass,commands,NULL, 0);
    to:
    quit  = XtCreateManagedWidget
    ("Quit",xmPushButtonWidgetClass,commands,NULL, 0 );
    no    = XtCreateManagedWidget
    ("no",xmPushButtonWidgetClass,commands,NULL, 0 );
    yes   = XtCreateManagedWidget
    ("yes",xmPushButtonWidgetClass,commands,NULL, 0 );
```
Add the callback function:

```c
Add the callback function:

XtAddCallback ( quit, XmNactivateCallback, quitCallback, NULL );
XtAddCallback ( no, XmNactivateCallback, noCallback, NULL );
XtAddCallback ( yes, XmNactivateCallback, yesCallback, NULL );
```

Add the following code to draw lines on the canvas:

```c
Add the following code to draw lines on the canvas:

display = XtDisplay(shell);
screen = DefaultScreen(display);
XtVaGetValues ( canvas, XmNforeground, &gcval.foreground, XmNbackground, &gcval.background, NULL );
draw = XtGetGC ( canvas, GCForeground | GCBackground , &gcval );
XtAddEventHandler(canvas,ButtonPressMask,FALSE, HandleBoardEvents, NULL);
.....
```

```c
void quitCallback ( Widget w, XtPointer clientData, XtPointer callData )
{
    Colormap cmap;
    XColor color, ignore;
    char *colorname = "red";

    XtSetArg(wargs[0], XmNlabelString, XmStringCreateLocalized("Are you sure?") );
    XtSetValues(quit, wargs, 1);
    XtVaSetValues( quit, XmNsensitive, False, NULL);
    XtMapWidget(no);
    XtMapWidget(yes);
    cmap = DefaultColormap (display, screen);
    XAllocNamedColor(display, cmap, colorname, &color, &ignore);
    XtSetArg(wargs[0], XmNbackground, color.pixel);
    XtSetValues(quit, wargs, 1);
}
```

```c
void yesCallback ( Widget w, XtPointer clientData, XtPointer callData )
{
    exit(0);
}
```

```c
void noCallback (Widget w, XtPointer clientData, XtPointer callData) 
{
    XtSetArg(wargs[0], XmNlabelString, XmStringCreateLocalized("QUIT "));
    XtSetValues(quit, wargs, 1);
}
```
XtVaSetValues( quit, XmNsensitive, True, NULL);
XtMapWidget(quit);
XtUnmapWidget(no);
XtUnmapWidget(yes);
}

void HandleBoardEvents( Widget w, XtPointer clientData, XEvent *event, Boolean *flag )
{
    XDrawRectangle(XtDisplay(w), XtWindow(w), draw,
                   event->xbutton.x, event->xbutton.y, 10, 10);
}

To associate the resource file with the program before execution.

    % setenv      XENVIRONMENT      Editor.color

Then type:

    % editor2