X - UNIX windowing system developed at MIT
Sun - OpenWindow competitor
Free - took off, not developed for des
Develop to run more than one process and look at multiple output screens
- evolved to the desktop

X - policy free system - system with 2 parts (processes)
a) windowing system interfacing with hardware (Xserver)
b) window manager - just another client

Popular window managers - Motif (Xm), twm
Can change look and feel on the fly, just change Xwm

The drawback is interprocess communication (efficiency) - send as pixel data
- good if running on a remote machine

Designed to be universal
Fonts are little pictures, too! Xserver provides fonts (resource)
Color is another X resource
Resources loaded into servers addresses a communication problem (reduced traffic)
X bundler commands to reduce network traffic
X works better on high speed connections.

OS impacts performance (shared memory if local, build ethernet packets if remote)

X protocol interfaces with Xserver
- not a trivial task to learn -> established program interface
- don't use Athena graphical objects! written as demos...

Widgets - graphical objects
Motif is most used widget set
- OpenSource foundation - not free
- gadgets are Motif fast widgets

X intrinsic - XT - X Toolkit
- embed multiple associated low level calls

all Xlib - XLoadFont ← we will use Xlib for project
all XT toolkit - XTOpen
all Motif - Xm Puth
Can use CASE tool to create GUIs - has become easier
let tool build program

Colors
named Xlib/ lists for color names, cursors in Pixile's directory
- system will convert name → 3 integer RGB values (table lookup)

[v3 → v4 someone changed color names - big problem
color name not found → black ]

X two special color
White is the first color in the colormap 3 invisible, default color back
Black is the last color in the colormap

Window program architecture - 5 steps for interactive window

1 - Initialize system

2 - Build Graphical Objects
   Simple window - box with 2 pixel wide border

3 - Register Events and Callback
   Callback - registering a call to a section of the screen (button, menu)
   (defining areas)
   Events - type, timestamp, location ...
   - old days used polling - led to uncertainties
   - now use events
   - 32 events
     ButtonPress, ButtonRelease, KeyPress, KeyRelease ...
   queue events according to timestamp
   (move mouse by 5 pixels to trigger event - leave it)
   events can be used
   don't want an event queue to be filled up with events you don't care about
   - default → none included
   - must define which one to register
   * Want to hear about damaged windows
   Expose Event - covered window revealed
   - only once by default

   Window trapping - wmi's responsibility
   window contents - programmer's responsibility

4 - Realize Graphical Objects - displays stuff on screen

5 - Event Loop - process event queue
   - must register an event to kill program via callback
Button clicks
Motion events turn on & off
Expose event

See source listing handout

All windows need a parent
- Kill parent, all children go away
- Root window not displayed

Point, pointy - anchor point cursor - moving

Screen - up to 4 screens

Event mask - 32 events (32 bits)
- Build in mask in (Xlib) use "1" bit use "or"
  \[ 00100000 \]
  \[ 00001000 \]
  \[ 00101000 \]

X colors are defined as 3 byte color
- If older machine (e.g., 16 color) maps to closest one
- Exact - what you ask for
- Apparent - what you get

GC - graphics context (X server resource) holds drawing information
- includes foreground & background colors
- Resources are not loaded! Must explicitly load
draw, erase & same object with foregrounds changed

GC data structure - defines a mask to replace standard values

Display becomes an argument to all Xlib functions!
Program rubber_band.c

Screen = 0

Graphics Context used for drawing (lines, circles, text, fillers...)

masks - variable parameter list (in GC)
send only parts which are different (minimise communications)

Events - must "expose" window first time
(can set graphical exposure to false, need to look this up)

Realize - subwindows
   XUnmapWindow - to "hide"

Event loop:
(We will need fonts for project) - need to display a digital clock
   look at sprite/Xstuff - look at Xfonts and Xcolor
   \rightarrow write program to examine fonts
   XLoadGiftFont - ask if font is available

Xrtn list all X commands/functions \rightarrow Xrtn - both shown &

look at structure of Xevent, Xcolor, XEvent

Expose event - when starts, or when uncovered
MotionNotify

[will not need one in project]

Button Press 1,2,3 mouse buttons

check for MotionNotify

   eventmask & PointerMotionMask bit-wise AND
   if set \rightarrow non-zero (TRUE)

   turn off MotionNotify

   eventmask &= ~PointerMotionMask
   same as
   eventmask = eventmask & ~PointerMotionMask

   if 0 1 0 0 0 0 0 0
      \sim ~ 1 0 1 1 1 1 1 1
      \rightarrow \text{Expose}
      \rightarrow \text{MotionNotify turned off!}
"01" | 00010010 | off
     | 01000000 | on
     | 01010010 |

X Select Input (display, window, event mak)
#include <X11/Xlib.h>
#include <stdio.h>
#define FORE "light goldenrod" named colors
#define BACK "midnight blue"

main(argc, argv)
int argc;
char **argv;
{
    Display *display; ← communications channel, pointer to a server
    Window root, window; ← system needs a root window, parent of all windows
    long fgcolor, bgcolor;
    int screen, pointx, pointy, eventmask = ExposureMask|ButtonPressMask;
    int curx, cury;
    XEvent event; ← data structure for event queue
    XColor exact, apparent;
    XGCValues gcval;
    GC draw, erase;

    if (!((display = XOpenDisplay(NULL)))) {
        perror("XOpenDisplay");
        exit(1);
    }

    root = RootWindow(display, screen = DefaultScreen(display)); on screen 0 on this machine
    XAllocColor for integer values
    XAllocNamedColor(display, DefaultColormap(display, screen),
                     FORE, &apparent, &exact); pass by reference return values
    fgcolor = apparent.pixel;
    XAllocNamedColor(display, DefaultColormap(display, screen),
                     BACK, &apparent, &exact);
    bgcolor = apparent.pixel;

    window = XCreateSimpleWindow(display, root, 0, 0, 400, 400, 2, fgcolor, bgcolor);

    gcval.foreground = fgcolor;
    gcval.background = bgcolor;
    draw = XCreateGC(display, window, GCForeground|GCBackground, &gcval);
    gcval.foreground = bgcolor;
    erase = XCreateGC(display, window, GCForeground|GCBackground, &gcval);

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

    could check apparent = exact to see if we got what we ask for

    open display → open root → grab color →