UIMS/Windowing Systems

Reading #3: "Chapter 4.1-4.2 Basics of Event Handling" by Dan Olsen, *Developing User Interfaces*, 1998, pp. 89-104.

Seeheim Model (1985)

- Definition
  - Separates UI code from application code
  - Provides UI tools to programmer
- User-Interface Management System (UIMS)
  - analogy to DBMS
  - UIMS is the run-time system
  - UIDE (User-Interface Design Environment) is the development tool

User Interface Operating Systems

Types

- Kernel-based Operating Systems
- Client-Server Operating Systems

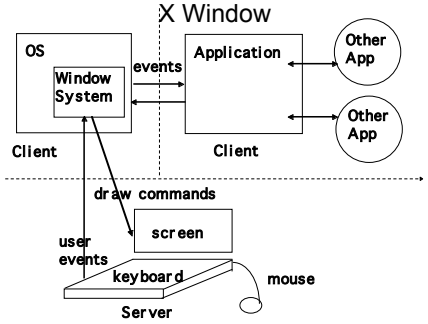
User Interface Operating Systems

- Kernel-based Systems
  - Services provided through code in the machine's operating system
  - Application makes calls to the operating system for interface system resources
  - Device-dependent code
  - Fast and efficient, not extensible
  - Examples: Classic Macintosh Toolbox and Microsoft Windows

User Interface Operating Systems

- Client-Server Systems
  - "Clients" or application programs communicate with the "server" to request resources (such as a new window) and services
  - Several clients can access one server
  - A client can have connections to multiple servers
  - Applications can share resources
  - Can be distributed over a network
  - Device-independent code
  - Extensible, not part of any machine's operating system
  - Examples: XWindow and NeWS

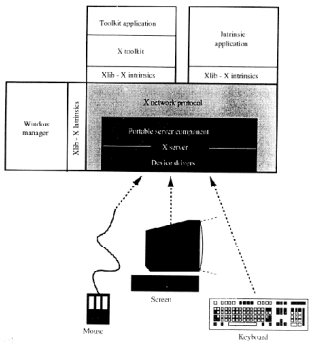
Client-Server Model



X Environment Processes

- X server for each display
  - manages display's hardware and window hierarchy
  - draws graphics, generates events
  - if Xterm, server is on client side system
  - if workstation, server is in workstation
- Window manager, one for each display
  - allows user to manipulate top-level window in uniform way for all applications
  - displays decorative frame around window
  - provides controls: move, resize, iconify, deiconify
  - title
- Application
  - communicates with X servers and window managers
    - uses network protocol such as TCP/IP

X Window Components



Types of UIMS Software

X Window: Xbuild, UIMX **Mac: MacApp, Hypercard**	User Interface Interactive Design Environments
X Window: Interviews **Next: Next-Step**	User Interface Class Library
X Window: Motif, Tk **Mac: Toolbox**	Toolkits
X Window: XLib, Xtrinsics **Mac: Window Manager**	Window Manager
X Window: GKS, Phigs **Mac: Quickdraw**	Graphics Languages
Bit-mapped display, mouse, keyboard	Device Drivers

X Window System Program

(at the Window Manager Level)

```
/* This is the skeleton of an X window program that produces
        "Hello World" in a window.

   It consists of just the X calls to do this. None of the code to set
    up the parameters for the calls is displayed.   */

XOpenDisplay;                   /*   open the display */

XLoadQueryFont;                 /*   load the correct font with error
                                    return*/

XCreateSimpleWindow;            /*   create the window */

XSetStandardProperties;         /*   set the properties to instruct */

XsetWMHints;                    /*   the window manager where
                                     to place the window      */

XChangeWindowAttribute;         /*   set the window's color map*/

XCreateGC;                          /* create the graphic context for
                                       text*/

XSelectInput;                       /* select events of interest, only
                                       expose events */

XMapWindow;                         /* make the window visible */

while (not_done) {                  /* loop forever processing events */

        XNextEvent;             /*          get expose event */

        XDrawString;            /*          place string on screen */
}
```

Kernel vs Client-Server Systems: Summary

- Kernel-based Systems
  - Device dependent
  - Not extensible by programmer or user
  - Single System UIMS
  - Fast
  - Sparse code
- Client-Server Systems
  - Device independent
  - Extensible
  - Network-based UIMS
  - Slower
  - Huge code

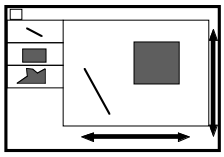
Who does what to whom where? Simple Drawing Application

Who draws what?

- Drawing a rectangle inside the draw window
- Moving and resizing a window

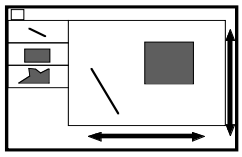
How does the app know what to do and when?

- Click outside window
- Click in window close box
- Click in icon panel
- Click inside draw window



Division of Responsibility

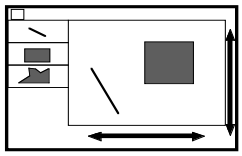
- What the UIMS must do
  - Draw each icon on the screen
  - Draw each object in the drawing window
  - Dialog handling
    - get a mouse input & decide which icon was selected
    - based on the icon selected, gets inputs for the object to be drawn
    - draws the object & adds it to the list of objects maintained by the application.



Division of Responsibility

Application Code

```
if line-icon chosen,
   DrawLine(X1,Y1,X2,Y2)
if rect-icon chosen,
   DrawRect(X1,Y1,X2,Y2)
if poly-icon chosen,
  do:
  get X,Y points
  StartPoly(X,Y)
  AddPolyPoint(X,Y)
  if poly-complete,
        EndPoly( )
```



Who Does What? Output

- Window Manager
  - Arbitrates which event goes where
    - UIMS
    - application
    - Examples
      - move a window doesn't matter to the application
      - menu item selected matters to the application
  - Manages redrawing and repainting screen
    - Knows where windows are
    - Examples
      - cursor movement
      - resize window

Who Does What? Input

- Window Manager
  - Arbitrates which input event goes where
  - Mouse/Keyboard Hardware
    - generates stream of keyboard and mouse events
    - Passed to appropriate application

Who Does What? Control

- Window Manager
  - Controls all events to the UI
    - Sequentially ordered and processed
    - Events ordered by timestamp and priority
      - Events typed by priority
      - Terminate application (“quit”) highest priority
  - Uses “event-loop” and a priority queue
    - Events handled asynchronously (older systems used “polling” system)
    - Macintosh and Microsoft Windows have only one queue
    - Multi-tasking OS (e.g. X Window) has a queue for each process

UIMS Event Types

- Input Events
  - Mouse Buttons
  - Modifier Keys (Shift, Control, Meta, Option, etc.)
  - Double-Clicking, triple-clicking
  - Function Keys
  - Mouse Movement
  - Mouse-Enter & Exit
  - Keyboard
- Windowing Events
  - Create, Destroy, Open, Close, Iconify, Deiconify, Resize
- Redrawing Events
- Pseudo-Events: communication between objects

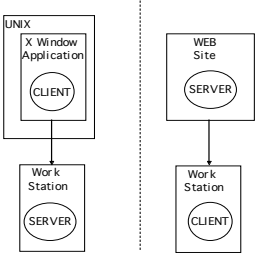
Who Does What?

- Application
  - in-window event handling then passed to UIMS
  - creates within window images
  - triggers redraw

What is a window?

- Windows can be much more than the traditional window
  - Widgets
  - Toolkits of widgets incorporated into the window manager

Some confusing terminology!



Seeheim Model

Advantages

- UI Code Advantages
  - Re-useable therefore economical to produce
  - UI consistency
  - Flexibility in design - easy to change
  - Allows non-specialist involvement
- Application Code Advantages
  - Code is better structured (decomposition by function) therefore fewer bugs
  - Reliability is high since high-level tool generates UI code
  - Device dependence isolated in UI therefore easier to port

A Question to Keep Asking Yourself

Can we completely separate the UI from the application?