Overview
Emphasis of this course

- Critical issues related to the synthesis, display, and manipulation of images
- Old way
  - proprietary hardware
  - incompatible GUI and API
  - single workstation
- New way
  - standardized hardware features (e.g. MMX, AGP)
  - small number of GUI (Xlib, widgets, Tcl/Tk) and API (OpenGL, Java3D)
  - networked graphics operations
Emphasis of this course

- Need be aware of
  - windowing environments
  - GUI and user interaction
  - graphics synthesis (both 2D & 3D)
- For Unix system, the choice is simple
  - X window systems
  - X widgets, Tcl/Tk, Java AWT, Swing
  - OpenGL (Mesa), Java3D, VRML
X Windows System

- Developed by Gettys, Seheifler, et al. at MIT for Project Athena in the ‘80s

- Features
  - hardware independent
  - network transparent
  - a mechanism supporting different interface designs and policies
**Example: Sun Windows**

- Fast
- A particular architecture
- Hardwired display information (single, fixed display environment)
X Windows: Client-Server model

Servers

library communication routines

Network

Client

application: drawline(x1,y1,x2,y2)
Features of the X Windows

- Slow (inter-process communication)
- Hardware independent
- Network transparent
- Only X server writes to screen
- Standard protocol
- One client -> multiple servers
- One server -> multiple clients
- Heterogeneous platforms
- Portable applications
Why Networked Operations?

- Distributed environment with a powerful compute server + many X terminals
- Sitting physically in front of a X terminal
- Slogin to compute server for computation

- Computation
  - X terminal: client
  - Compute server: server

- Display
  - X terminal: server
  - Compute server: client
Client-Server Model in X
Asynchronous communication

Server

% xhost +client

draw line
create window
query pointer position

Client

% setenv DISPLAY server:0.0

key pressed
pointer moved
window exposed

Computer Graphics
X Programming Environment #1

- **X Protocol**: define data structures used to transmit information between clients and servers
- **Xlib**: procedural interface for app. programmer
Xlib

- >300 C-callable routines
- For
  - windowing
  - interaction
  - 2D graphics
- Flexible
- Difficult to use
- Simple “hello world” program > 50 lines
Widgets

- High-level constructs
- Window data structures + Specific UI functions

Examples:

<table>
<thead>
<tr>
<th>Widgets</th>
<th>window data structures</th>
<th>UI operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>button</td>
<td>width, height</td>
<td>click</td>
</tr>
<tr>
<td>scroll bar</td>
<td>border, label</td>
<td>jump, scroll</td>
</tr>
<tr>
<td>menu</td>
<td>font, cursor</td>
<td>click, drag</td>
</tr>
<tr>
<td>canvas</td>
<td>bg, fg colors</td>
<td>...</td>
</tr>
</tbody>
</table>
Widgets

- Widget sets:
  - Athena
  - Motif
  - Open Look
  - Interview
  - Andrew
  - Dec Window

- X window is *policy free*
  - not dictate look-and-feel of applications
  - provide mechanisms supporting
    - 3D effects
    - non-overlapping windows
    - click-and-type
    - auto raise, etc.

<table>
<thead>
<tr>
<th>Widget Set</th>
<th>Sample Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Athena</td>
<td>sample distribution</td>
</tr>
<tr>
<td>Motif</td>
<td>OSF</td>
</tr>
<tr>
<td>Open Look</td>
<td>Unix International</td>
</tr>
<tr>
<td>Interview</td>
<td>Stanford</td>
</tr>
<tr>
<td>Andrew</td>
<td>CMU</td>
</tr>
<tr>
<td>Dec Window</td>
<td>Digital</td>
</tr>
</tbody>
</table>
Intrinsics

- Library routines for
  - creating widget classes
  - specifying widget resources
  - customizing widget behaviors
  - registration and invocation of widget procedures
  - etc.
X Programming Environment #2

Server

X protocol

Network

Client

X protocol

Xlib

Intrinsic

Toolkits

Computer Graphics
### External (desktop management) functions are common

- For screen resource sharing
- Implemented by a window manager (e.g. twm, tvtwm, mwm, fvwm, etc.)

<table>
<thead>
<tr>
<th>Applications</th>
<th>Internal</th>
<th>External</th>
</tr>
</thead>
<tbody>
<tr>
<td>xcalc</td>
<td>scientific calculator</td>
<td>move/resize</td>
</tr>
<tr>
<td>xbiff</td>
<td>mail alert</td>
<td>create/destroy</td>
</tr>
<tr>
<td>xterm</td>
<td>terminal emulator</td>
<td>iconfy/de-iconfy</td>
</tr>
<tr>
<td>netscape</td>
<td>WWW browser</td>
<td>expose/hide</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Window Managers

- An X client
- Use special X protocol to
  - manipulate window tree hierarchy (WM adopts all top-level application windows)
  - grab pointer & keyboard (input focus policy)
  - alter event dispatch
  - install colormaps when enter/exit from a window
  - create, delete, resize, iconify, expose, hide windows
  - decoration
  - etc.
**X Programming Environment #3**

Diagram showing the relationship between Server and Client, with X protocol, Xlib, Intrinsic Toolkits, and Window manager layers.
**GUI & 2D Graphics Options**

- Athena, Motif, etc.
  - C based
  - Unix only
  - Complicated and steep learning curve (a graphical *hello world* program can be > 100 lines)

- Other possibilities
  - Java AWT, JFC/Swing, Tcl/Tk, etc. for windowing & GUI
  - Java 2D, OpenGL for graphics
  - Not necessarily C based
  - Unix, MS Windows, Mac
  - Can be much simpler
3D Graphics

- Xlib, AWT, Swing, Tcl/Tk provide simple line, polygon, etc drawing and filling routines
- Limited to 2D
- 3D graphics is much more complicated, involving
  - view points
  - material and texture properties
  - lighting
  - animation, etc.
- Need to understand the underlying physics principle and an API for programming
OpenGL

- An open standard of SGI’s GL libraries
- Source is available and can be compiled into X Windows environment
- X server needs an GLX extension to understand 3D protocol
- Mesa
Java3D

- Fairly new, currently at bleeding edge
- May not work with your browsers
- Does not work under jdk 1.1.* (Get jdk 2.0 or above)
- Object-Oriented with many features
- Easier to learn than OpenGL
Another Classification

- User Interface
- 2D Graphics
- 3D Graphics
- Modeling
- Animation
This Course

- **What it is**
  - System oriented
  - Internal mechanism, algorithm
  - Teach you how to be a *graphics system programmer*

- **What it is *not***
  - Not a course that teaches you how to be *an animator*