Designing Interactive Systems 2

Lecture 2: Window System Architecture

Prof. Dr. Jan Borchers
Media Computing Group
RWTH Aachen University

hci.rwth-aachen.de/dis2
CHAPTER 4

Graphics & Event Library
## Graphics & Event Library

<table>
<thead>
<tr>
<th>Apps</th>
<th>UI</th>
<th>TK</th>
<th>WM</th>
<th>BWS</th>
<th>GEL</th>
<th>Hardware</th>
</tr>
</thead>
</table>

### Event queues
- Canonical events
- Driver-specific data

### Graphics objects & actions
- Logical coordinates
- Memory addresses

### Device drivers
- Graphics hardware

- Device-independent
- Device dependent
Graphics Models

RasterOp Model

Vector Model
Graphics Library Objects: Canvas

• A canvas is a memory area with a coordinate system and memory-to-pixel mapping

• Different formats
Graphics Library Objects: Output Objects

Elementary Objects

Complex Objects
Graphics Library Objects: Graphics Context

- State of the (virtual) graphics processor

- Goal: **reduce parameters** to pass when calling graphics operations

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Font</td>
<td>Gill Sans</td>
</tr>
<tr>
<td>Font size</td>
<td>24 pt</td>
</tr>
<tr>
<td>Font color</td>
<td>(0,0,0)</td>
</tr>
<tr>
<td>Line width</td>
<td>2 px</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

drawString(x, y, “Turtle”);
setFillColor(green);
fillPath(treePath);
setFillColor(orange);
fillPath(deerPath);
Damage Repair

- Mouse cursor is always drawn by GEL (performance)
- Restoring contents that were occluded by mouse cursor needed
## Event Library

### Canonical Mouse Event Queue

<table>
<thead>
<tr>
<th>Event</th>
<th>Type</th>
<th>Movement</th>
<th>Value</th>
<th>Timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Type</td>
<td>Movement</td>
<td>Value</td>
<td>Timestamp</td>
</tr>
<tr>
<td></td>
<td>Mouse Down</td>
<td>(0,2)</td>
<td></td>
<td>13:37:12.203</td>
</tr>
<tr>
<td></td>
<td>Mouse Down</td>
<td>(0,2)</td>
<td></td>
<td>13:37:12.271</td>
</tr>
<tr>
<td></td>
<td>Mouse Up</td>
<td>Left</td>
<td></td>
<td>13:37:12.289</td>
</tr>
<tr>
<td></td>
<td>Movement</td>
<td>(12,2)</td>
<td></td>
<td>13:37:12.416</td>
</tr>
</tbody>
</table>

### Mouse Driver Event Queue

<table>
<thead>
<tr>
<th>Event</th>
<th>Button 1</th>
<th>Button 2</th>
<th>Button 3</th>
<th>X</th>
<th>Y</th>
<th>Wheel</th>
<th>Timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>31267</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>31335</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>31421</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>31634</td>
</tr>
</tbody>
</table>
How Extensible is the GEL?

• Most systems: Not accessible to application developer

• GEL as library: extensible only with access to source code (X11)

• GEL access via interpreted language: extensible at runtime (NeWS)
  • NeWS example: Download PostScript code into GEL to draw triangles, gridlines, patterns,...
Graphics & Event Library

- Apps
- UITK
- WM
- BWS
- GEL

Event queues
- Canonical events
- Driver-specific data

Graphics objects & actions
- Logical coordinates
- Memory addresses

Device drivers

Device-independent
- Device-dependent

Graphics hardware

Hardware
CHAPTER 5

Base Window System
Base Window System

• **Core** of the Window System
  • Provides **WS-wide data structures** and operations
  • Manages **shared resources** to ensure consistency

• **Base window**: logical canvases that include the on-screen windows and / or widgets

• In general:
  1 WS — $k$ terminals, $m$ applications, $n$ objects per application (windows, fonts)
Base Window System

Dialog input, State messaging

<table>
<thead>
<tr>
<th>Access control</th>
<th>Request</th>
</tr>
</thead>
<tbody>
<tr>
<td>Demultiplex</td>
<td>Request</td>
</tr>
<tr>
<td>Multiplex</td>
<td>Mutual exclusion</td>
</tr>
<tr>
<td>Queue/Dequeue</td>
<td>Memory allocation</td>
</tr>
<tr>
<td>Events</td>
<td>Canvas</td>
</tr>
</tbody>
</table>

Requests, output, changes

Connection management
Resource operations
Synchronization
Elementary operations
Objects
Base Window System: Objects

Window & Canvas

Events

Graphics Context

Color Tables

Fonts
Base Window System: Objects

Components

• Owning application
• Using applications
• Size, depth, border, origin
• State variables (only for windows)

Operations

• Drawing in local coordinate system
• State changes
Base Window System: Objects

Components
- Event type
- Timestamp
- Type-specific data
- Location
- Window
- Application

Operations
- Read, write and filter events
Demultiplex

Order Events by Timestamp

Multiplex

Application 1
Application 2
Application m
Window Manager

Mouse Queue
Keyboard Queue
Trackpad Queue
Base Window System: Objects

Components

• Owner app, user apps
• Graphics attributes (line thickness, color index, copy function,...)
• Text attributes (color, skew, direction, copy function,...)
• Color table reference
Base Window System: Objects

Components

- Owner app, user apps
- Data fields for each color entry
- RGB, HSV, YIQ,…

Operations

- Provide default color
- Find close replacements of colors (fault tolerance)
Base Window System: Objects

Components

- Owner app, user apps
- Name, measurements (font size, kerning, ligatures, ...)
- One data field per character

Fonts
Base Window Management

- Trees are used to manage the window collection in the BWS
  - All child windows are inside their parent window
  - Simplifies event routing or setting visibility
Base Window Management

Exercise
Determine a valid tree structure for the window arrangement shown above.
Base Window Management
Base Window System: Shared Resources

- Reasons for sharing resources: Scarcity, collaboration
- **Problems**: Competition, consistency
Base Window System: Shared Resources

- Reasons for sharing resources: Scarcity, collaboration

- **Problems**: Competition, consistency

- **Synchronization** needed
  - At BWS entrance
  - Or on individual objects
Base Window System: OS Integration

Single address space

Apps
UITK
WM
BWS
GEL
HW
Base Window System: OS Integration

- Base Window System
- OS Integration

- Single address space
- User address space
- Kernel address space

- Apps
- UITK
- WM
- BWS
- GEL
- HW
Base Window System: OS Integration

Single address space

User address space

Kernel address space

Apps
UITK
WM
BWS
GEL
HW

Apps
UITK
WM
BWS
GEL
HW

Apps
UITK
WM
BWS
GEL
HW
## Base Window System

<table>
<thead>
<tr>
<th>Dialog input, State messaging</th>
<th>Requests, output, changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addressing</td>
<td>Access control</td>
</tr>
<tr>
<td>Demultiplex</td>
<td>Request</td>
</tr>
<tr>
<td>Multiplex</td>
<td>Mutual exclusion</td>
</tr>
<tr>
<td>Queue/Dequeue</td>
<td>Memory allocation</td>
</tr>
<tr>
<td>Events</td>
<td>Canvas</td>
</tr>
</tbody>
</table>

- Connection management
- Resource operations
- Synchronization
- Elementary operations
- Objects

- Event library
- Graphics library
Designing Interactive Systems 2

Lecture 3: Window System Architecture

Prof. Dr. Jan Borchers
Media Computing Group
RWTH Aachen University

hci.rwth-aachen.de/dis2
CHAPTER 6
Window Manager
Window Manager

• Allow users to control windows

• Provide Look & Feel for interaction with the WS
Window Manager

- Apps
- UITK
- WM
- BWS
- GEL
- Hardware

**Application-independent user interface**

<table>
<thead>
<tr>
<th>Behavior (“Feel”)</th>
<th>Appearance (“Look”)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pop-up menus at click</td>
<td>Tiling, overlapping, …</td>
</tr>
<tr>
<td>Fetch events</td>
<td>Request position change</td>
</tr>
</tbody>
</table>

**Events**

- Window positions

**Request drawing**

- Look & Feel
- Techniques
- Communicate with BWS

- Fetch events
- Request position change
- Communicate with BWS

- Hardware

Prof. Dr. Jan Borchers: Designing Interactive Systems II
Window Manager

Screen Management

• What is rendered where on screen?
• Where is empty space? Which apps are iconified?
Window Manager

Screen Management

• What is rendered where on screen?
• Where is empty space? Which apps are iconified?

Session Management

• Provide consistent ways to perform standard tasks
• Move window, start app, iconify window
Session Management

Menu Techniques

Pull-Down Menu

Cascading Menu

Fixed Menu Bar
Session Management

Window Decorations

Windows 7

Windows 10

macOS
Session Management

Layout Policy
Session Management

Virtual Desktops
Session Management

Direct manipulation

Icon technique

Input focus

Look & Feel
Late Refinement

- WM accompanies the session
- changing window positions
- changing app appearance
Late Refinement

- WM accompanies the session
  - changing window positions
  - changing app appearance

- Levels of late refinement
  - Per session or application
  - Per launch, user or for all users
Late Refinement

• WM accompanies the session
  • changing window positions
  • changing app appearance

• Levels of late refinement
  • Per session or application
  • Per launch, user or for all users

• Implementation with table files, internal database, or delta technique
Late Refinement: macOS Login Window

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple/DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
  <dict>
    <key>GuestEnabled</key>
    <false/>
    <key>lastUser</key>
    <string>loggedIn</string>
    <key>lastUserName</key>
    <string>borchers</string>
    <key>retriesUntilHint</key>
    <integer>3</integer>
  </dict>
</plist>
```
Window Manager: Location

Apps

WM
BWS
GEL

Upper part of BWS
Window Manager: Location

Upper part of BWS

Separate server
Window Manager: Location

Upper part of BWS

Separate server

Separate user process

Apps

WM

BWS

GEL

Apps

BWS

GEL

WM

Apps

BWS

GEL

WM

Apps

BWS

GEL

WM
Window Manager: Conclusions

- WM leads from system- to user-centered view of WS
- Provide different levels of consistency
- Accompanies user during session
- Potentially exchangeable
- WM requires UI Toolkit to implement same Look & Feel across applications
Window Manager

- Apps
- UITK
- WM
- BWS
- GEL
- Hardware

Application-independent user interface

<table>
<thead>
<tr>
<th>Behavior (&quot;Feel&quot;)</th>
<th>Appearance (&quot;Look&quot;)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pop-up menus at click</td>
<td>Tiling, overlapping, …</td>
</tr>
<tr>
<td>Fetch events</td>
<td>Request position change</td>
</tr>
</tbody>
</table>

Events
- Window positions

Request drawing

Look & Feel

Techniques

Communicate with BWS
CHAPTER 7

User Interface Toolkit
User Interface Toolkit

- Apps
- UITK
- WM
- BWS
- GEL
- Hardware

- UIDS / UIDL
- Interface Guidelines (Look & Feel)
- Composition and Refinement
- Basic Widgets

- Events
  - Position

- Actions
  - Drawing requests
UITK: Components

Widget Set

- Push Button
- Popup Button
- Radio Button
- Label
- Check Box
- Segment 1, Segment 2, Segment 3

User Interface Design System

Visual Studio Blend

Hello World!

Macintosh HD > Applications
UITK: Requirements

- Composition
- Reusability
- Communication
- Separation from app logic
UITK: Defining Widgets

Widget :=

\(<W=(w_1\ldots w_k), \ G=(g_1\ldots g_l), \ A=(a_1\ldots a_m), \ I=(i_1\ldots i_n)>\)
UITK: Defining Widgets

Exercise
What are the typical components $<W, G, A, I>$ of a button?

• $W$ = (text window, shadow window)
• $G$ = (size, color, font, shadow color,...)
• $A$ = (enter callback, leave callback, clicked callback)
• $I$ = (triggered with mouse, triggered with key, enter, leave)
UITK: Basic Widgets

- Push Button
- Check Box
- Menu Item 1
  - Menu Item 2
  - Menu Item 3
- Radio Button
- Container
UITK: Creating Complex Widgets

Composition

Minimal Button

Refinement

BUY
Dynamic Widget Hierarchy

What's new in this release?
- We fixed a major issue that caused the app to crash when saving
- We also added a new button in the main menu
Static Widget Hierarchy

NSView

NSControl

NSSlider

NSTextField

NSButton

MyButton
Late Refinement of Widgets
Style Guidelines
Types of UIDS

Language oriented (UIDL)

• Compiler implements style guidelines by checking constructs

Interactive

• Complex drawing programs to define look of UI
• Via lines connecting user input ($I$) to actions ($A$) (allowed by style guide)

Automatic

• Create UI automatically from spec of app logic (research)
User Interface Description Languages

```xml
<?xml version="1.0"?>
<?xml-stylesheet href="chrome://global/skin/" type="text/css"?>
<window id="findfile-window" title="Find Files" orient="horizontal" xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">
  <button label="Normal"/>
  <button label="Disabled" disabled="true"/>
</window>
```
User Interface Design Systems
User Interface Toolkit

UIDS / UIDL

Interface Guidelines (Look & Feel)

Composition and Refinement

Basic Widgets

Apps

WM

BWS

GEL

Hardware

Events

Position

Actions

Drawing requests