Designing Interactive Systems 2

Lecture 4: The X Window System, Smalltalk

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

hci.rwth-aachen.de/dis2
CHAPTER 8

The X Window System
The X Window System

- Origin: Window system for V OS
  - Moved BWS&GEL to remote machine
  - Simplified porting to new architectures, but slow under Unix

- MIT: X improvement over W
  - Asynchronous calls: much-improved performance
  - Application = client
X: Architecture

Application
- Widget Set
- Xt Intrinsics
- Xlib

Network
- X Server
- Kernel (OS)
- Hardware

UITK & WM
- Window Manager
- Xlib

BWS & GEL
X Server

- Responsible for one keyboard (one EL)
- Can manage multiple physical screens (GLs)
- Provides base windows as canvas for clients (BWS)
X: Protocol

Client

First packet

Accept / Refuse

Request

Reply

Request

Error

Event

Server
Xlib

- Implements X protocol client
- Checks for events from server & creates queue on client
- Xlib offers functions to create, delete, and modify server resources
Typical Xlib application

```c
#include <Xlib.h>, <Xutil.h>
Display *d; int screen; GC gc; Window w; XEvent e;
main () {
    d = XOpenDisplay("171.64.77.1:0");
    screen = DefaultScreen(d);
    w = XCreateSimpleWindow(d, DefaultRootWindow(d), x,y,w,h,
                           border, BlackPixel(d), WhitePixel(d)); //fore- & background
    XMapWindow(d, w);
    // Graphics Context setup left out here
    gc = XCreateGC(d, w, mask, attributes);
    XSelectInput(d, w, ExposureMask|ButtonPressMask);
    while (TRUE) {
        XNextEvent(d, &e);
        switch (e.type) {
            case Expose: XDrawLine (d, w, gc, x,y, w,h); break;
            case ButtonPress: exit(0);
        }
    }
```
X Toolkit Intrinsics

- Xt Functions are generic to work with all widget classes
X Toolkit Intrinsics

• Xt Functions are generic to work with all widget classes

• At runtime widgets have four states: Created, managed, realized, mapped
X Toolkit Intrinsics

- Xt Functions are generic to work with all widget classes
- At runtime widgets have four states: Created, managed, realized, mapped
- Dispatches events
Widget Set

- Programming model already given in intrinsics
- Collection of several different user interface components
- Defines the look & feel of the system together with the WM
Athena Widget Set

- **Simple** — Base class for all other Athena widgets
  - Does nothing, but adds new resources such as cursor and border pixmap
- **Standard widgets**
- **Special widgets**
Motif: More than a Widget Set

• **Style Guide** (book) for application developer

• **Widget set** implementing style guide

• **Window Manager** (mwm)

• **UIDL**
Motif: Widget Set
Programming in X

```c
#include <X11/Intrinsic.h>
#include <X11/StringDefs.h>
#include <X11/Xlib.h>
#include <Xm/Xm.h>
#include <Xm/PushB.h>

void ExitCB (Widget w, caddr_t client_data, XmAnyCallbackStruct *call_data)
{
    XtCloseDisplay (XtDisplay (w));
    exit (0);
}

void main(int argc, char *argv[])
{
    Widget toplevel, pushbutton;
    toplevel = XtInitialize (argv [0], "Hello", NULL, 0, &argc, argv);
    pushbutton = XmCreatePushButton (toplevel, "pushbutton", NULL, 0);
    XtManageChild (pushbutton);
    XtAddCallback (pushbutton, XmNactivateCallback, (void *) ExitCB, NULL);
    XtRealizeWidget (toplevel);
    XtMainLoop ();
}
```
X: Window Manager

- Ordinary client to the BWS
- Communicates with apps via hints in X Server
- Look&Feel mechanisms are separated from Look&Feel policy
- Late refinement
- Exchangeable at runtime
X: Demo
CHAPTER 9
Wayland
Wayland: Motivation

• X rendering pipeline designed in the 1980s

• Modern clients use libraries instead of referring to X
  • Hence, the X Server has lost one of its core functionalities

• Communication overhead
  • X was designed as a distributed system
  • 3D effects
Wayland: Motivation

Where is the mouse cursor?

• In screen coordinates: (0.5, 0.5)
• In desktop coordinates: (0.2, 0.5)

The WS does not know
X: Communication

Network

App (X Client) → X Server → Kernel (OS) → Hardware

WM (Compositor)
Wayland

• Wayland is…
  • A communication protocol between the compositor and its clients (similar to Xlib)
  • An implementation of that protocol as a C library
• No network transparency
  Clients and compositor talk to each other via IPC
Wayland: Direct Rendering

• Graphics memory shared between clients and compositor

• Applications render directly into a memory buffer

• Compositor uses buffers from all clients and recomposites the screen

• Saves communication overhead
X as Wayland Client

- Provide backwards compatibility to X clients
- XWayland is an X Server implementation with changes that allow to run X on Wayland
CHAPTER 10

Smalltalk
Smalltalk

- The common ancestor of all window systems
- Operating system, window system, OO programming language
- Introduced the MVC Pattern
Smalltalk

• The common ancestor of all window systems

• Operating system, window system, OO programming language

• Introduced the MVC Pattern

• UITK with modeless editor
Smalltalk

- The common ancestor of all window systems

- Operating system, window system, OO programming language

- Introduced the MVC Pattern

- UITK with modeless editor

- Inspect and modify the system’s code while it is running
<table>
<thead>
<tr>
<th>AllClasses</th>
<th>Date</th>
<th>ClassDefinition</th>
</tr>
</thead>
<tbody>
<tr>
<td>SystemOrganization</td>
<td>Float</td>
<td>ClassOrganization</td>
</tr>
<tr>
<td>'Kernel Classes'</td>
<td>Integer</td>
<td>'Arithmetic'</td>
</tr>
<tr>
<td>'Numbers'</td>
<td>LargeInteger</td>
<td>'Conversion'</td>
</tr>
<tr>
<td>'Basic Data Structures'</td>
<td>MachineDouble</td>
<td>'Math functions'</td>
</tr>
<tr>
<td>'Sets and Dictionaries'</td>
<td>Natural</td>
<td>'Printing'</td>
</tr>
<tr>
<td>'Graphical Objects'</td>
<td>Number</td>
<td>'Initialization'</td>
</tr>
<tr>
<td>'Text Objects'</td>
<td>Time</td>
<td></td>
</tr>
<tr>
<td>'Windows'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'Panes and Menus'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'Files'</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
sqrt | t1 t2
[[self ≤ 0.0 ⇒
  [self = 0.0 ⇒ [↑0.0]
  user notify: 'sqrt invalid for x<0.']]

  t1 ← self + 0.0.
  t1 instfield: 1 ← (t1 instfield: 1) / 4 * 2.
  for t2 to: 5 do8
    [t1 ← self - (t1 * t1) / (t1 * 2.0) + t1].
  [↑t1]
```
Smalltalk: Architecture

- Single process, single address space

- Machine-dependent **virtual machine**
  (byte-code interpreter)

- Machine-independent **virtual image**
  (Smalltalk classes)

- Initially OS & WS merged,
  later WS on top of OS
Model-View-Controller

User SEES Controller
Controller USES View
View MANIPULATES Model
Model UPDATES Problem Domain Application

Prof. Dr. Jan Borchers: Designing Interactive Systems 2
Morphic

- UI construction environment for Smalltalk

- Key concepts: **Directness** and **liveness**

- Widgets are called **morphs**
  - Every morph can be a container for other morphs
  - Used for reification of widget structure and layout
  - Morphs can have autonomous behavior, usually appearing as animation
Squeak: Demo
Morphic: Implementing Layout

Exercise
Algorithm to determine the layout of a morph that includes a tree of submorphs?

• **1st pass:** Compute minimum size of all submorphs bottom-up

• **2nd pass:** Distribute available space between submorphs top-down

• Optimizations?
  • Deferred layout
  • Pruning
  • Site selection
Morphic: Managing Redraws

- Damage List
  - Add bounding box of each changed morph to list
  - Each frame, redraw all morphs intersecting each bounding box in damage list
  - Double buffering prevents the user from seeing the construction of an animation

- Improvements?
History

1970 Smalltalk
1977 Tajo
1977 Dlisp
1980 Docs
1981 Star
1981 NU
1984 X
1984 Macintosh
1985 Windows
1986 NeWS
1983 Andrew