Designing Interactive Systems II

Computer Science Graduate Programme SS 2010

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

http://hci.rwth-aachen.de/dis2
Review

- What is the difference between Smalltalk, Squeak, and Morphic?
- How did the original Smalltalk implement the window system layer architecture?
- What are the most particular qualities of Morphic as a UI toolkit?
- What are morphs, and what is special about them?
- How does Morphic implement widget layout?
The Apple Macintosh

- Introduced in 1984
- Based on PARC Smalltalk, Star, Tajo
- Few technical innovations (QuickDraw)
  - Otherwise, rather steps back
- But landmark in UI design and consistency policies
  - First commercially successful GUI machine
  - Advertised with what is sometimes considered the best commercial in history:
20 Years Later...
Macintosh: Architecture

- One address space, communication with procedure calls
- "No" OS—app is in charge, everything else is a subroutine library ("Toolbox")
- Functional, not object-oriented (originally written in Pascal)
- Organized into Managers
- Mostly located in "the Mac ROM"
Event Manager

- Event loop core of any Mac app
- Processes events (from user or system) and responds
- Event Manager offers functions to deal with events
  - extern pascal Boolean
    GetNextEvent(short
eventMask, EventRecord
*theEvent);
- Cooperative Multitasking
  - External: App must allow user to switch to other apps
  - Internal: App must surrender processor to system regularly

struct EventRecord {
  short what;   // type of event
  long message; // varies depending // on type
  long when;    // Timestamp in ticks
  Point where;  // mouse position // in global coords
  short modifiers; // modifier keys held down
};

Event types
enum {
  nullEvent   =  0,
  mouseDown   =  1,
  mouseUp     =  2,
  keyDown     =  3,
  keyUp       =  4,
  autoKey     =  5,
  updateEvt   =  6,
  diskEvt     =  7,
  activateEvt =  8,
  osEvt       = 15,
};
Control Manager

• Controls: Buttons, checkboxes, radio buttons, pop-up menus, scroll bars,...
• Control Manager: Create, manipulate, redraw, track & respond to user actions

Dialog Manager

• Create and manage dialogs and alerts
• (System-) modal, movable (application-modal), or modeless dialog boxes—choice depends on task!
Window Manager(!)

- Not the Window Manager from our layer model
- Create, move, size, zoom, update windows
- App needs to ensure background windows look deactivated (blank scrollbars,...)

Menu Manager

- Offers menu bar, pull-down, hierarch. & pop-up menus
- Guidelines: any app must support Apple, File, Edit, Help, Keyboard, and Application menus
Finder Interface

- Defining icons for applications and documents
- Interacting with the Finder

Other Managers

- Scrap Manager for cut&paste among apps
- Standard File Package for file dialogs
- Help Manager for balloon help
- TextEdit for editing and displaying styled text
- Memory Manager for the heap
- List Manager, Sound Manager, Sound Input Manager,...
Resource Manager

- Resources are basic elements of any Mac app: Descriptions of menus, dialog boxes, controls, sounds, fonts, icons,...
  - Makes it easier to update, translate apps
- Stored in resource fork of each file
  - Each Mac file has data & resource fork
  - Data fork keeps application-specific data (File Manager)
  - Resource fork keeps resources in structured format (Resource Manager)
    - For documents: Preferences, icon, window position
    - For apps: Menus, windows, controls, icons, code(!)
Resource Manager

- Identified by type (4 chars) and ID (integer)
  - Standard resource types (WIND, ALRT, ICON, ...)
  - Custom resource types (defined by app)
- Read and cached by Resource Manager upon request
  - Priorities through search order when looking for resource
    - Last opened document, other open docs, app, system
- Can write resources to app or document resource fork
  - E.g., last window position
- Graphical Resource Editor (Apple)
- Overview of resources in resource fork of any file (app or doc), sorted by resource type
- Opening a type shows resources of that type sorted by their ID
- Editors for basic resource types built in (ICON, DLOG, ...)
- Big productivity improvement over loading resources as byte streams
Macintosh: Evaluation

- **Availability**: high (apps from 1984 still ran on machines from 2005)
- **Productivity**: originally low (few tools except ResEdit; Mac was designed for users, not programmers)
- **Parallelism**: originally none, later external+internal
  - External: Desk accessories, Switcher, MultiFinder
  - Internal: Multi-processor support in mid-90's
Macintosh: Evaluation

- **Performance**: high (first Mac was 68000@1 MHz, 128K RAM) – improvement over Smalltalk
- **Graphic model**: QuickDraw (RasterOp+fonts, curves...)
- **Style**: most consistent to this day (HI Guidelines, Toolbox)
- **Extensibility**: low (Toolbox in ROM, later extended via System file)
Macintosh: Evaluation

• **Adaptability**: medium (System/app/doc preferences in resources, but limited ways to change look&feel)
• **Resource sharing**: medium (fonts, menu bar shared by apps,...)
• **Distribution**: none
• **API structure**: procedural (originally Pascal)
• **API comfort**: high (complete set of widgets)
• **Independency**: Medium (most UI code in Toolbox)
• **Communication**: originally limited to cut&paste
In-Class Exercise: Simple Mac Application

- Write a simple Macintosh application that opens a window and exits upon mouseclick
void main (void)
{
    WindowPtr window;
    Rect rect;

    InitGraf (&qd.thePort); // must be called before any other TB Manager (IM IX 2-36)
    InitFonts ();  // after ig, call just to be sure (IM IX 4-51)
    FlushEvents(everyEvent,0); // ignore left-over (finder) events during startup
    InitWindows (); // must call ig & if before (IM Toolbox Essentials 4-75; IM I 280)

    InitCursor (); // show arrow cursor to indicate that we are ready

    SetRect (&rect, 100, 100, 400, 300);

    window = NewCWindow (NULL, &rect, "My Test", true, documentProc,
                          (WindowPtr) -1, FALSE, 0);

    do {
    }
    while (!Button());

    DisposeWindow (window);
}
Windows: History

- 1985: Windows 1.0
  - no virtual memory, shared memory space
  - tiled windows only, no composite widgets
  - dev tools: DOS only
Windows: History

- 1987: Windows 2.0 - windows can overlap
  - composite widgets (dialog boxes)
  - Windows 2.04: address memory >1MB
  - SDK w/ MS C 5.0, can develop within Windows
Windows: History

• 1990: Windows 3.0
  • virtual memory
  • BMP format adopted
Windows: History

- 1993: Windows NT 3.1
  - rewritten from scratch, 32-bit (Win32)
  - pre-emptive multitasking, processes
  - ran on x86, MIPS, Alpha, PowerPC
int PASCAL WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, 
     LPSTR lpszCmdLine, int nCmdShow)
{
    static char szAppName [] = "DIS II goes Windows" ;
    MSG msg;
    WNDCLASS wndclass ;

    if (!hPrevInstance)
    {
        wndclass.style = CS_HREDRAW | CS_VREDRAW ;
        wndclass.lpfnWndProc = WndProc ;
        wndclass.hInstance = hInstance ;
        wndclass.hIcon = LoadIcon (hInstance, IDI_APPLICATION) ;
        wndclass.hCursor = LoadCursor (NULL, IDC_ARROW) ;
        wndclass.lpszMenuName = "AppMenu";
        wndclass.lpszClassName = szAppName ;
        ... 
        RegisterClass (&wndclass) ;
    }

    HWND hwnd = CreateWindow (szAppName, "DIS II",
        WS_OVERLAPPEDWINDOW, CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT, 
        NULL, NULL, hInstance, NULL) ;

    ShowWindow (hwnd, nCmdShow) ; //show window
    UpdateWindow (hwnd); //initial update

    while (GetMessage (&msg, NULL, 0, 0))
    {
        TranslateMessage (&msg);
        DispatchMessage (&msg);
    }

    return msg.wParam ;
}