Event Driven UIs and Model-View-Controller

CS160: User Interfaces
John Canny

Includes slides based on those of James Landay & Jeffrey Heer
Reminder

Archos 5 hardware available in class this Weds – one per group.

Pls bring a check for $200 to UC Regents as a deposit.

You need the Archos for individual programming assignment 4 – no BT in emulator. OK to work in pairs. But still submit individually.

Anyone else need Wii remote?
Topics

Interactive application programming
- Component Model
- Event-Driven User Interfaces
- Model-View Controller pattern
Interactive Application Programming
In the beginning...

bash-2.05b$ pwd
/home/dstone
bash-2.05b$ cd /usr/portage/app-shells/bash
bash-2.05b$ ls -al
total 60
-rwxr-xr-x  3 root  root  4096 May 14 12:05 .
-rwxr-xr-x 26 root  root  4096 May 17 02:36 ...
-rw-r--r--  1 root  root 13710 May  3 22:35 ChangeLog
-rw-r--r--  1 root  root  2924 May 14 12:05 Manifest
-rw-r--r--  1 root  root  3720 May 14 12:05 bash-2.05b-r11.ebuild
-rw-r--r--  1 root  root  3516 May  2 20:05 bash-2.05b-r9.ebuild
-rw-r--r--  1 root  root  5883 May  3 22:35 bash-3.8-r11.ebuild
-rw-r--r--  1 root  root  4038 May 14 12:05 bash-3.8-r7.ebuild
-rw-r--r--  1 root  root  3931 May 14 12:05 bash-3.8-r8.ebuild
-rw-r--r--  1 root  root  4267 Mar 29 21:11 bash-3.8-r9.ebuild
drwxr-xr-x  2 root  root  4096 May  3 22:35 files
-rw-r--r--  1 root  root  164 Dec 29 2003 metadata.xml
bash-2.05b$ cat metadata.xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE pkgmetadata SYSTEM "http://www.gentoo.org.dtd/metadata.dtd">
<pkgmetadata>
<herd>base-system</herd>
</pkgmetadata>
bash-2.05b$ sudo /etc/init.d/bluetooth status
Password:
* status: stopped
bash-2.05b$ ping -q -c 1 en.wikipedia.org
PING rr.chtpa.wikimedia.org (207.142.131.247) 56(404) bytes of data.

--- rr.chtpa.wikimedia.org ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 112.076/112.076/112.076/0.000 ms
bash-2.05b$ grep -i /dev/sda /etc/fstab | cut --fields=3
/dev/sda1  /mnt/usbkey
/dev/sda2  /mnt/ipod
bash-2.05b$ date
Wed May 25 11:36:56 PDT 2005
bash-2.05b$ Isnod
Module Size Used by
joudev 8256 0
ipw2200 17512 0
ieee80211 44228 1 ipw2200
ieee80211_crypt 4972 2 ipw2200,ieee80211
c1000 64468 0
bash-2.05b$
The Xerox Alto (1973)
Event-Driven UIs

Old model (e.g., UNIX shell, DOS)
– Interaction controlled by system, user queried for input when needed by system

Event-Driven Interfaces (e.g., GUIs)
– Interaction controlled by user
– System waits for user actions and then reacts
– Complex, dynamic screen content requires a more complicated system.
Event-Driven UIs

i.e. A "console" program looks like this:

- do some work...
- prompt user for input
- wait for user input
- process it...
Event-Driven UIs

i.e. An “interactive” program does at least this:

Do until a quit command: 
{
    wait for user input
    process it...
    (optionally) update display
}
Event-Driven UIs

“Console”-style input processing:

Switch (input-cmd) {
    case insert: do-insert(...) 
    case delete: do-delete(...) 
    case backspace: ...
}
Event-Driven UIs

Can’t use this approach for window systems. The result of a user command depends on what is the active window - usually the one under the mouse.

There are too many possible combinations of input x target window, and the window structure is dynamic.

Instead, code and data associated with a active screen element are packaged together in a “widget” or “component.”
Component/Widget Model

Encapsulation and organization of interactive components ("widgets")

- Typically using a class hierarchy with a top-level "Component" type implementing basic bounds management, and event processing

Drawn using underlying 2D graphics library

Input event processing and handling

- Typically mouse and keyboard events

Bounds management (damage/redraw)

- Only redraw areas in need of updating
What are Some Examples of Components?
What are Some Examples of Components?

- Windows
- Layout panels
- Drawing panes
- Buttons
- Sliders
- Scrollbars
- Images
- Dropdown boxes
- Toolbars
- Menus
- Dialogue Boxes

- Progress indicators
- Video
- Icons
- Links
- Checkboxes
- Radio buttons
- Etc.
Periodic Table of Motif Widgets

The Periodic Table of Motif Widgets
The Open Group
Java Swing Widgets
Windows Vista/.Net Widgets
Apple Cocoa Widgets
Android Widgets
User Interface Components

Each component is an object with

- Bounding box
- Paint method for drawing itself
  - Drawn in the component’s coordinate system
- Callbacks to process input events
  - Mouse clicks, typed keys

```java
public void paint(Graphics g) {
    g.fillRect(...); // interior
    g.drawString(...); // label
    g.drawRect(...); // outline
}
```
2D Graphics Model

Every component is a clipped drawing canvas with a coordinate system

- Origin typically at top-left, increasing down and to the right
- Units depend on the output medium (e.g., pixels for screen)
- Rendering methods
  - Draw, fill shapes
  - Draw text strings
  - Draw images
Composing a User Interface

Label

TextArea

Buttons

How might we instruct the computer to generate this layout?
Absolute Layout

- **Label** (x=0, y=0, w=350, h=20)
- **TextArea** (x=0, y=20, w=350, h=150)
- **Buttons** (x=200, y=175, w=45, h=30)
  (x=250, y=175, w=85, h=30)

But this is inflexible and doesn’t scale or resize well.
In Android

Declarative layout, main.xml:

```xml
<?xml version="1.0" encoding="utf-8" ?>
  <TextView android:layout_width="wrap_content" android:layout_height="wrap_content" android:text="@string/enter_amt" />
  <EditText android:id="@+id/editText1" android:layout_width="200px" android:layout_height="wrap_content" android:numeric="decimal" />
  <FrameLayout android:layout_width="fill_parent" android:layout_height="0px" android:layout_weight="1" />
  <TextView android:layout_width="wrap_content" android:layout_height="wrap_content" android:text="@string/choose_tip" />
  <RadioGroup android:layout_width="wrap_content" android:layout_height="wrap_content" android:orientation="horizontal" android:id="@+id/tipGroup1">
    <RadioButton android:checked="false" android:id="@+id/option1" android:text="10%" />
    <RadioButton android:checked="false" android:id="@+id/option2" android:text="15%" />
    <RadioButton android:checked="false" android:id="@+id/option3" android:text="20%" />
  </RadioGroup>
  <FrameLayout android:layout_width="fill_parent" android:layout_height="0px" android:layout_weight="1" />
  <TextView android:id="@+id/tipVal" android:layout_width="wrap_content" android:layout_height="wrap_content" android:text="Tip: $0" />
  <TextView android:id="@+id/totalVal" android:layout_width="wrap_content" android:layout_height="wrap_content" android:text="Total: $0" />
  <FrameLayout android:layout_width="fill_parent" android:layout_height="0px" android:layout_weight="1" />
  <Button android:id="@+id/splitButton" android:layout_width="wrap_content" android:layout_height="wrap_content" android:text="Split bill" />
  <FrameLayout android:layout_width="fill_parent" android:layout_height="0px" android:layout_weight="8" />
</LinearLayout>
```
In Android
Root View

LinearLayout (vertical)
Next level

TextView
EditText
FrameLayout (wt 1)
TextView
RadioGroup (horizontal)
FrameLayout (wt 1)
TextView
FrameLayout (wt 1)
Button
FrameLayout (wt 8)
Leaves

RadioButtons

Tip Calculator
Enter Check Amount

Choose Tip Amount

10% 15% 20%

Tip: $0
Total: $0

Split bill
c.f. Layout in Cocoa: Springs + Struts
In Android

You can always use code for your layout (procedural layout):

```java
lay1 = (LinearLayout) this.findViewById(R.id.mainFrame);
tv1 = new TextView(this);
lay1.add(tv1); // Makes the textView a child of the LinearLayout View
fl1 = new FrameLayout(this);
fl1.setLayoutParams(MATCH_PARENT, 0px, 1); // width, height, weight
lay1.add(fl1);
```

And create struct and spring space with variables. Then to update, override the `onLayout` method.
Events
Events

User input is modeled as events that must be handled by the system and applications.

- Mouse input (and touch, pen, etc.)
  - Mouse entered, exited, moved, clicked, dragged
  - **Inferred events**: double-clicks, gestures
- Keyboard (key down, key up)
- Sensor inputs
- Window movement, resizing
Anatomy of an Event

An event encapsulates the information needed for handlers to react to the input

– Event Type (mouse moved, key down, etc)
– Event Source (the input component)
– Timestamp (when did event occur)
– Modifiers (Ctrl, Shift, Alt, etc)
– Event Content
  • Mouse: x,y coordinates, button pressed, # clicks
  • Keyboard: which key was pressed
Events

Level of abstraction may vary. Consider:

- **Mouse down** vs. **double click** vs. **drag**
- **Pen move** vs. **gesture**
Callbacks

- onMouseOver(Event e){…}
- onClick(Event e){…}
- onMouseClick(Event e){…}
- onMouseDown(Event e){…}
- onMouseUp(Event e){…}

Slider
Event Dispatch Loop

- Event Queue
  - Queue of input events

- Event Loop (runs in dedicated thread)
  - Remove next event from queue
  - Determine event type
  - Find proper component(s)
  - Invoke callbacks on components
  - Repeat, or wait until event arrives

- Component
  - Invoked callback method
  - Update application state
  - Request repaint, if needed

Mouse moved \((t_0, x, y)\)
Event Dispatch

Event Queue
- Mouse moved \((t_0,x,y)\)
- Mouse pressed \((t_1,x,y,1)\)
- Mouse dragged \((t_2,x,y,1)\)
- Key typed \((t_3, ‘F1’)\)
- ...
(queues and dispatches incoming events in a dedicated thread)

/* callback for TextArea */
public void mouseMoved(e) {
    // process mouse moved event
}
Interactor Tree

Display Screen

\[
\begin{array}{c}
\text{Outer Win [black]} \\
\text{Inner Win [green]} \\
\text{Result Win [tan]} \\
\text{Result String} \\
\text{Keypad [Teal]} \\
\text{= button} \\
\text{- button} \\
\text{+ button} \\
\text{0 button}
\end{array}
\]
“Front” window gets event, decides whether to handle it. - may decide to pass it on (by boolean return value).
Overlays

Sometimes the complexity in an existing widget makes it difficult to add event-handlers or draw methods to it.

e.g. MapView
Overlays

An overlay sits “above” the other View and receives events first.

Handles onTouch() and onTap()

It should always forward events it doesn’t want to deal with (return false in Android) so the main widget still works.
There are several ways to use events:

```java
public class myApp extends Activity
    implements View.OnClickListener, View.someOtherListener {

    public void onClick(View v)
        // figure out who handles the event

    public void onSomeOtherEvent(View v)
        // figure out who handles the event
```
Code Style

Or:

Public class myApp extends Activity {

    Public class myClickListener implements View.OnClickListener {
        // handle the event for one widget
        Public void onClick(View w) {
            ...
        }
    }

    Public void onCreate(...) {
        myWidget.setOnClickListener(new myClickListener());
    }
}
Code Style

Tradeoffs:

- Activity-level:
  - Big switch statement to funnel event to right widget
  - Less modular

- Inner class:
  - Per-widget code only in onClick() methods
  - Much more modular
  - More memory to create classes? Perhaps – but they are very small objects.
Model-View-Controller Architecture
Model-View-Controller

Architecture for interactive apps
- introduced by Smalltalk developers at PARC
Partitions application in a way that is
- maintainable
- extensible
- scalable
Idea is that the controller mediates between views (different representations of data) and the data itself, which is in the model.
Model-View-Controller

Is a general software design pattern
Very explicit in the Cocoa Framework
Model-View-Controller

Similar to the architecture of client-server systems:
- Amazon
- Spreadsheet views of DBs
- MMORPG’s (World of Warcraft etc.)
- Email
Example Application

Blue circles: 4
Cardinal squares: 2
Information the app is trying to manipulate

Representation of real world state
- shapes in a drawing program
- document contents
- game state: players plus world
- email DB
The model should be **standalone**. i.e. it shouldn’t depend on the view or controller logic.

Ideally, its schema should be fixed since the logic of several controllers may depend on it.
Implements a visual display of (part of) the model data. May have multiple views
- e.g., board view and schematic for CAD tool
- Outline and page view for Powerpoint
The view system contains information about “how” the viewer is viewing the data, e.g.

- Where they are looking in 3D game
- What scale and page you are on in an editor

But information about content resides in the model, so that the view can be rebuilt from it.
Views

View only needs to know current page number and scale and size of main view and context view.
Multiple Views

Blue circles: 4
Cardinal squares: 2
Shouldn’t *directly* depend on model data, but certainly depends on the *abstract* content of the model.

i.e. the schema in the model might change, but the view need not since the controller is mediating.
Since some view components are for input, the “input device” may really be a screen widget in the view.
Controller

Mediating function:

• Converts information from model format to what the view needs.
• Interprets view state (e.g. the page you’re on) to make appropriate updates and queries from the model.
Control function:

- Makes things happen: processes events from input devices (or view widgets), and propagates changes by other apps from model to view.
Non-MVC design

Avoid too much data stored in view.
Adding a circle

Blue circles: 3
Cardinal squares: 2
Adding a circle

Blue circles: 3
Cardinal squares: 2
Adding a circle

Blue circles: 3
Cardinal squares: 2
Adding a circle

Blue circles: 4
Cardinal squares: 2
Relationship of View & Controller

“pattern of behavior in response to user events (controller issues) is independent of visual geometry (view issues)” –Olsen, Chapter 5.2
Relationship of View & Controller

“pattern of behavior in response to user events (controller issues) is independent of visual geometry (view issues)”

Controller must contact view to interpret what user events mean (e.g., selection)
Combining View & Controller

View and controller are tightly intertwined
  – lots of communication between the two
Almost always occur in pairs
  – i.e., for each view, need a separate controller
Some apps combine into a single class
Apple’s view

To more fully modularize, Cocoa separates some controllers into view-facing and model-facing subparts. Cocoa’s document classes:
In Android

MVC architecture much less obvious, although it is claimed. Component control logic “internally MVC”.

One concrete example is `AdapterView`, which includes
- `ExpandableListView`
- `Gallery`
- `GridView`
- `ListView`
- `Spinner`

An Adapter class mediates between a View class and some data.
Why MVC?

Combining MVC into one class will not scale
  – model may have more than one view
  – Each view is different and needs updating when model changes

Separation eases maintenance and extensibility
  – easy to add a new view later
  – model info can be extended, but old views still work
  – can change a view later, e.g., draw shapes in 3-d (recall, view handles selection)
  – flexibility of changing input handling when using separate controllers
Adding Views Later

Blue circles: 4
Cardinal squares: 2
MVC and connections

Cocoa and certain other systems (Nokia’s Qt) support a dynamic connection model between widgets and code. In Xcode, you edit these graphically with Interface Builder.
MVC and connections

This leads naturally to a clean view/controller boundary. Similarly, Cocoa’s “core data” provides data encapsulation, creating a clean controller/model boundary.
Changing the Display

How do we redraw when shape moves?

Widgets with an internal MVC structure make this easy because they have a model for the data to be drawn.
Moving Cardinal Square

Blue circles: 4
Cardinal squares: 2
Erase w/ Background Color and Redraw

Blue circles: 4
Cardinal squares: 2
Changing the Display

Erase and redraw
  – using background color to erase fails
  – drawing shape in new position loses ordering
Move in model and then redraw view
  – change position of shapes in model
  – model keeps shapes in a desired order
  – tell all views to redraw themselves in order
  – slow for large / complex drawings
Damage / Redraw Method

View informs windowing system of areas that need to be updated (i.e., damaged) with `invalidate();`
  – does not redraw them at this time...

Windowing system
  – batches updates
  – clips them to visible portions of window

Next time waiting for input
  – windowing system calls `draw` method

Damage old, Change position in model, Damage new

Blue circles: 4
Cardinal squares: 2
Example - Event Flow

Creating a new shape
Event Flow (cont.)

Assume blue circle selected
Press mouse over tentative position
Windowing system identifies proper window for event
Controller for drawing area gets mouse click event
Checks mode and sees “circle”
Calls model’s AddCircle method with new position
AddCircle adds new circle to model’s list of objects
Model then notifies list of views of change
  – drawing area view and text summary view
Views notifies windowing system of damage
  – both views notify WS without making changes yet!
Event Flow (cont.)

Views return to model, which returns to controller
Controller returns to event handler
Event handler notices damage requests pending and responds
If one of the views was obscured, it would be ignored
Event handler calls views’ `draw()` methods with damaged areas.
Views redraw all objects in the model that are in the damaged area.
Review

Component Model
Event Handling
Model-View-Controller
  – Break up a component into
    • **Model** of the data supporting the App
    • **View** determining the look of the application
    • **Controller** for mediating and event management
  – Provides scalability and extensibility
  – Damage-repair drawing
Reminder

Archos 5 hardware available in class this Weds – one per group.

Pls bring a check for $200 to UC Regents as a deposit.

You need the Archos for individual programming assignment 4 – no BT in emulator. OK to work in pairs. But still submit individually.