Example of the Day:

http://www.apple.com/ipad/#video

iPad 2
Thinner. Lighter. Faster.
FaceTime. Smart Covers. 10-hour battery.
How worthwhile is the course?

1 - Not worthwhile  2  3  4  5  6  7 - Very worthwhile

Bar chart showing the distribution of ratings for how worthwhile the course is.
What you like

1. Examples of Good UIs & Bad UIs
2. Demonstrations and Videos
3. Practical, hands-on orientation of class: real-world examples and assignments
4. Big project – in stages
Areas for improvement

1. We don’t teach how to program Android
2. Readings: Long
3. Reading Responses
4. Focus of discussion sections?
5. Group issues?
Conflicting Opinions

1. More group assignments vs. fewer group assignments
2. More design process, “fuzzy” topics vs. more engineering
Requests that are queued up

1. Lecture on Graphic Design and Aesthetics
2. Discussion of Web Apps
Flex arms available for your video
Review: Widgets, Layouts, Events
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:
```java
public void paint(Graphics g) {
    g.fillRect(...); // interior
    g.drawString(...); // label
    g.drawRect(...); // outline
}
```

Cocoa:
```cocoa
(void)drawRect:(NSRect)rect
```
Layout: Containment Hierarchy

- Window
  - Panel
    - Label
    - TextArea
    - Panel
      - Button
      - Button

Enter Text:
Anatomy of an Event

Encapsulates info needed for handlers to react to 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
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 \((t_0, x, y)\)
In Android

<table>
<thead>
<tr>
<th>Concept</th>
<th>Android SDK Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Widget (Control)</td>
<td>android.view.View</td>
</tr>
<tr>
<td>Layout</td>
<td>android.view.ViewGroup</td>
</tr>
<tr>
<td>Event</td>
<td>android.view.Event</td>
</tr>
</tbody>
</table>
Layout: Containment Hierarchy

- Activity
  - ViewGroup
    - View
    - View
    - ViewGroup
      - View
      - View
Event Listeners in Android

```java
final EditText edittext = (EditText) findViewById(R.id.edittext);
editttext.setOnKeyListener(new OnKeyListener() {
    public boolean onKey(View v, int keyCode, KeyEvent event) {
        // If the event is a key-down event on the "enter" button
        if ((event.getAction() == KeyEvent.ACTION_DOWN) &&
            (keyCode == KeyEvent.KEYCODE_ENTER)) {
            // Perform action on key press
            Toast.makeText(HelloFormStuff.this, edittext.getText(),
                           Toast.LENGTH_SHORT).show();
            return true;
        }
    return false;
}
});
```
Model-View-Controller Architecture
Model-View-Controller

OO Architecture for interactive applications
introduced by Smalltalk developers at PARC ca. 1983
Model

Information the app is manipulating

Representation of real world objects
- circuit for a CAD program
- logic gates and wires connecting them
- shapes in a drawing program
- geometry and color
View

Implements a visual display of the model

May have multiple views

e.g., shape view and numerical view
Multiple Views
View

Implements a visual display of the model

May have multiple views
  e.g., shape view and numerical view

Any time model changes each view must be notified so it can update
  e.g., adding a new shape
Controller

Receives all input events from the user

Decides what events mean and what to do
communicates with view to determine the objects being manipulated (e.g., selection)
calls model methods to make changes on objects
model makes change and notifies views to update
Why MVC?
Why MVC?

“The user's conceptual model of the system captures the semantics of objects, relationships, and behavior” (Collins)
Why MVC?

**Combining MVC into one class will not scale**
model may have more than one view
each is different and needs update 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 3D
flexibility of changing input handling when using separate controllers
Example Application

Blue circles: 4
Cardinal squares: 2
Model

Class AppModel {
    ArrayList<Point> rectangles;
    ArrayList<Point> circles;
    Color rectangleColor;
    Color circleColor;

    ...

}

Multiple Views

Blue circles: 4
Cardinal squares: 2
Controller

Blue circles: 3
Cardinal squares: 2
Controller

Blue circles: 3
Cardinal squares: 2
Controller

Blue circles: 3
Cardinal squares: 2

Click!
Controller

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

But controller must usually 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

Many architectures combine into a single class (“MV”)
Terminology

Is an `android.view.View` object an MVC View?
What about an `Activity`?
Model-ViewController in Android

**Model:**
Inherit from `java.util.Observable` class.
Provide accessors and mutators for state.
Call `setChanged()` and `notifyObservers()`

**Activity:**
Implement `java.util.Observer`:
add `update()` method
Changing the Display

How do we redraw graphics when a shape moves?
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

...slow for large / complex drawings

flashing! (can solve with double buffering)
Damage / Redraw Method

View informs windowing system of areas that are damaged
does not redraw them right away…

Windowing system
batches updates
cips them to visible portions of window

Next time waiting for input
windowing system calls Repaint() method
passes region that needs to be updated
Damage old, Change position in model, Damage new

Blue circles: 4
Cardinal squares: 2
HOW ANDROID DRAWS VIEWS

“When an Activity receives focus, it will be requested to draw its layout. […]”

Drawing begins with the root node of the layout. Drawing is handled by walking the tree and rendering each View that intersects the invalid region. The framework will not draw Views that are not in the invalid region. […]

You can force a View to draw, by calling invalidate().
MVC Event Flow

What happens when the user creates a new shape?
Event Flow (cont.)

Assume blue circle selected

Blue circles: 0
Cardinal squares: 0
• 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!
model may override
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’ `Repaint()` methods with damaged areas.
Views redraw all objects in model that are in damaged area.
Dragging at Interactive Speeds

**Damage old, move, damage new method may be too slow**

must take less than ~100 ms to be smooth

**Solutions**

don’t draw object, draw an outline (cartoon)
save portion of frame buffer before dragging
draw bitmap rather than redraw the component
modern hardware often alleviates the problem
Summary

Event-Driven Interfaces
Hierarchy of components or widgets
Input events dispatched to components
Components process events with callback methods

Model-View-Controller
Break up a component into
Model of the data backing the widget(s)
View determining the look of the widget
Controller for handling input events
Provides scalability and extensibility
Next Time

**Multithreading**

**Usability Studies**

Don’t forget to read and submit comment!

**Video Prototype Due!**

**Continue work on Programming Assignment III.**