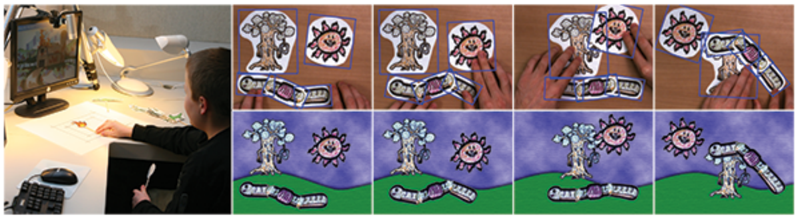
CS160: User Interface Design

Widgets, Events, MVC 02/29/12

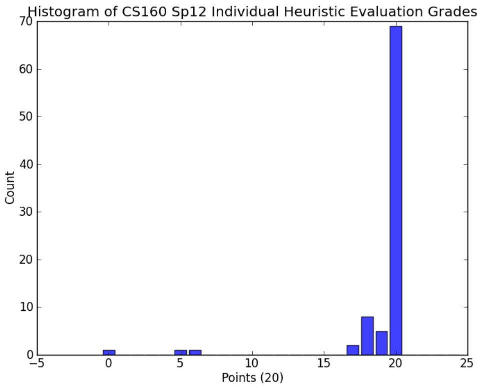
Berkeley
UNIVERSITY OF CALIFORNIA

Video Puppetry: SIGGRAPH Asia 2008



Authors: Connelly Barnes, David E. Jacobs, Jason Sanders, Dan B Goldman, Szymon Rusinkiewicz, Adam Finkelstein, Maneesh Agrawala

Results: Indiv. Heuristic Evaluation

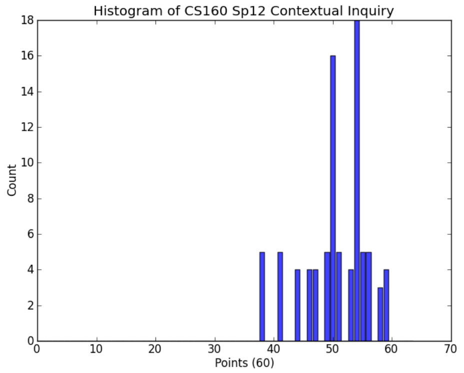


Stats:
Num: 87
Mean: 19.13
Median: 20.0
Stddev: 3.05

Grades on bSpace now

Regrades: Write down where you think you deserve more points and submit physical copy to us. We will regrade entire assignment. Your grade **can decrease** during regrading.

Results: Contextual Inquiry



Stats:
Num: 87
Mean: 50.48
Median: 51.0
Stddev: 5.36

Grades on bSpace now

Regrades: Write down where you think you deserve more points and submit physical copy to us. We will regrade entire assignment. Your grade **can decrease** during regrading.

Contextual Inquiry

Group: Pajama Party

Ordering fast food for deaf users

http://www.youtube.com/watch?v=o1sswVMmSO4&feature=player_embedded

Assignment: Low Fidelity Prototype

Due Mar 5

Identify project mission statement

Create **a low-fidelity paper prototype** that supports 3 tasks

1 easy, 1 moderate, 1 difficult task

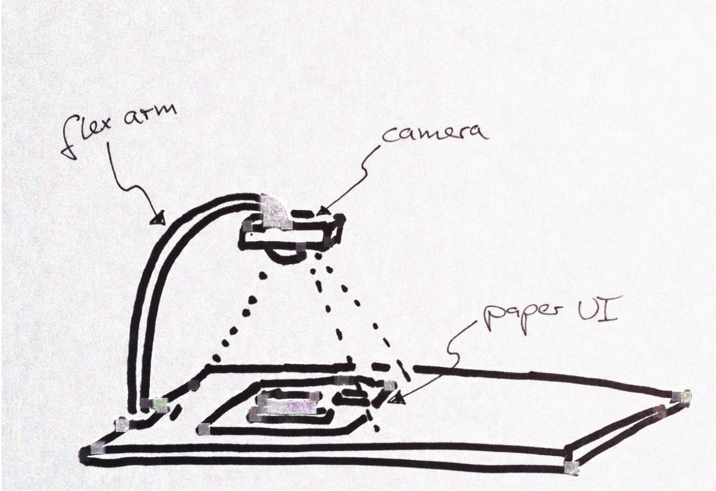
Create a video showing your prototype:

How it supports the 3 tasks

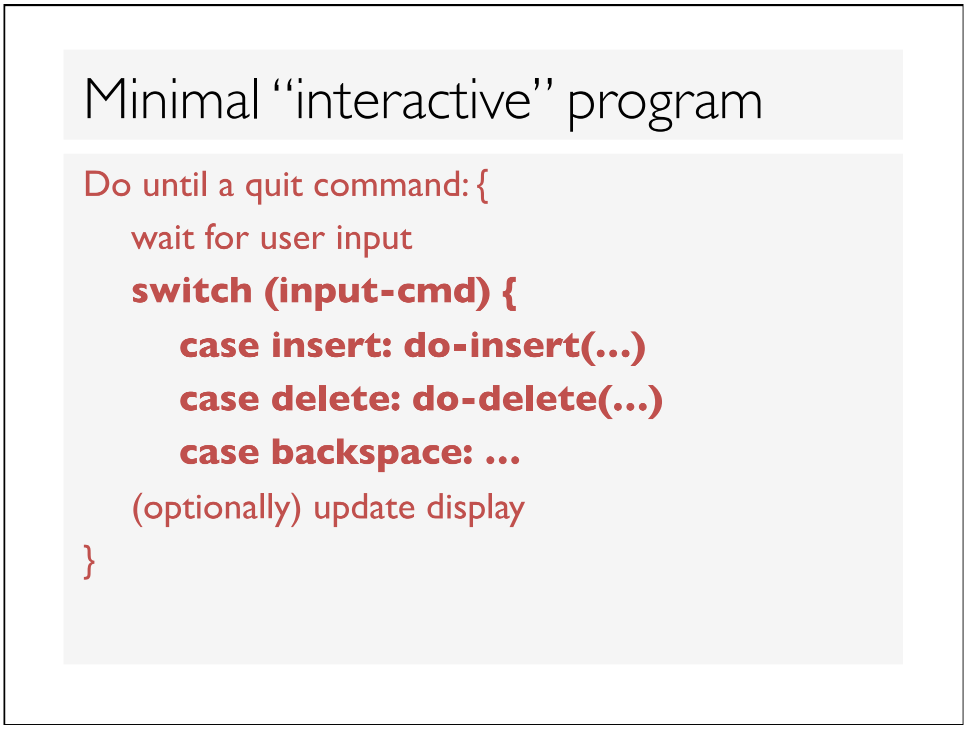
Context in which is will be used (back story)

Your video must include narration!

Flex arms available for your video

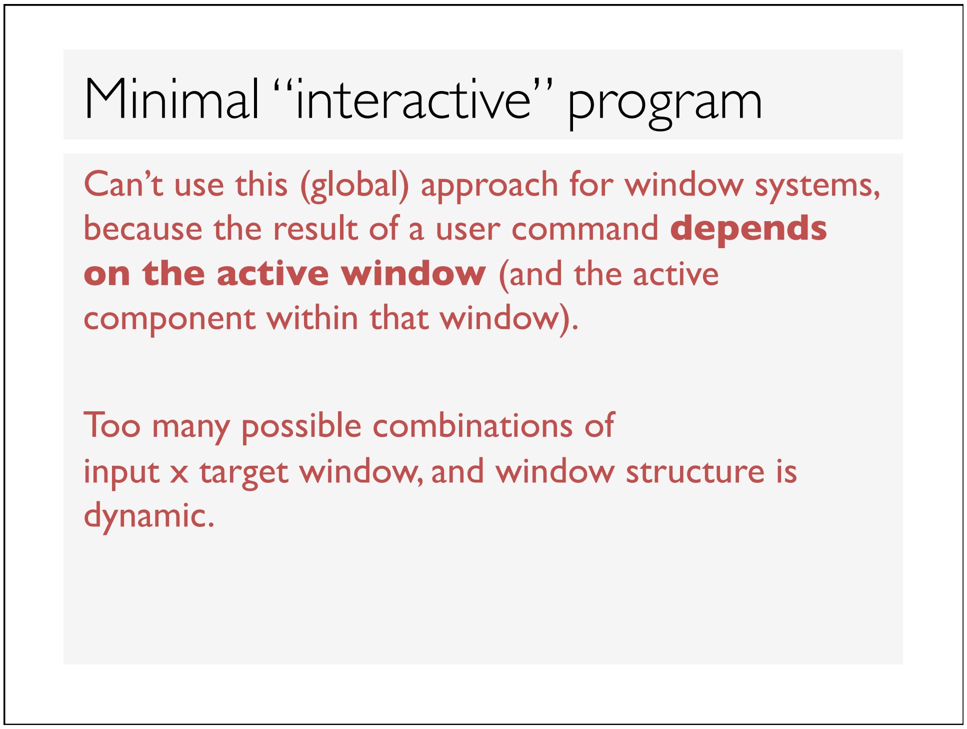


Widgets, Layouts, Events



Minimal "interactive" program

```
Do until a quit command: {
    wait for user input
    switch (input-cmd) {
        case insert: do-insert(…)
        case delete: do-delete(…)
        case backspace: …
    (optionally) update display
}
```



Minimal "interactive" program

Can't use this (global) approach for window systems, because the result of a user command **depends on the active window** (and the active component within that window).

Too many possible combinations of input x target window, and window structure is dynamic.

GUI Toolkits

Most user interfaces today are written using toolkits (e.g., QT, Cocoa, Java Swing, GTK, Android SDK,…)

Toolkits come with *libraries* of interactive elements (widgets) and layouts

Frequently used interactive components

Toolkits also define an *architecture*:

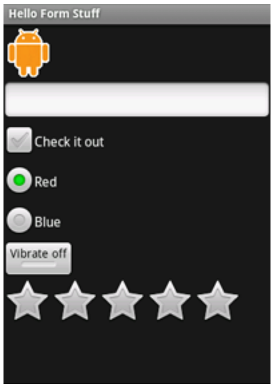
A standard way to handle input and output

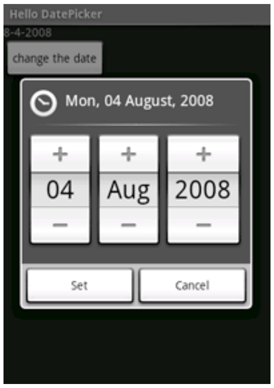
Usually wrap main() – application programmer writes pieces of code that plug into the architecture

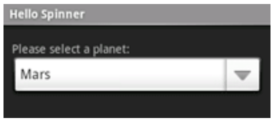
The architecture specifies how to write new widgets for the library

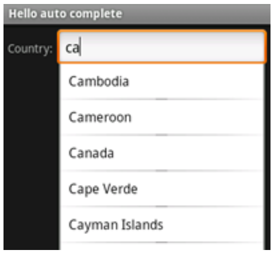
Widgets

Android Widgets

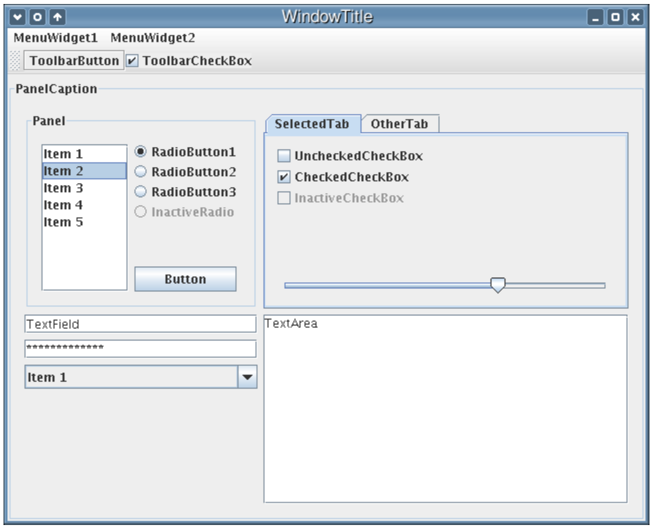




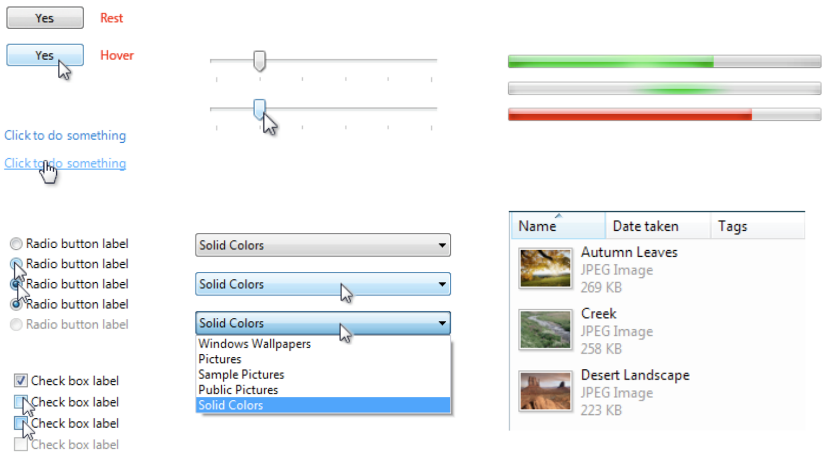




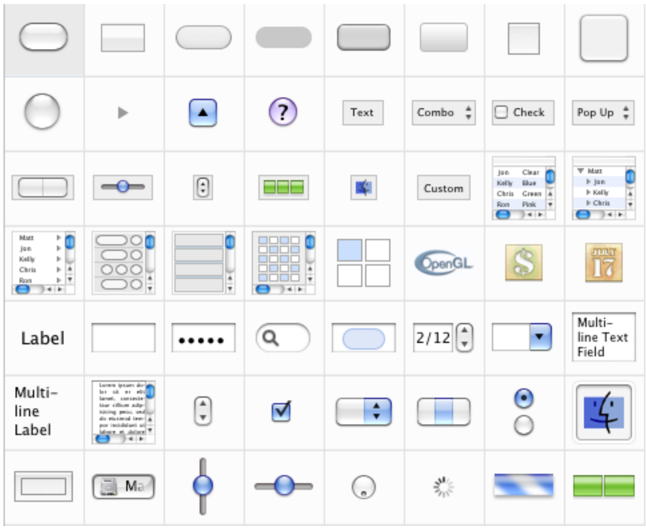
Java Swing Widgets



Windows Vista Widgets



Mac Cocoa Widgets



Interface Builder - Library

Widgets

Encapsulation and organization of interactive controls

Class hierarchy encapsulating widgets

Top-level "Component" class

Implements 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

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
}
```

Cocoa:

(void)**drawRect:**(NSRect)*rect*

2D Graphics Model

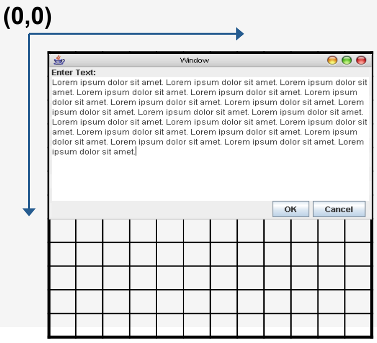
Widget canvas and coordinate system

Origin often at top-left, increasing down and to the right

Units depend on output medium (e.g., pixels for screen)

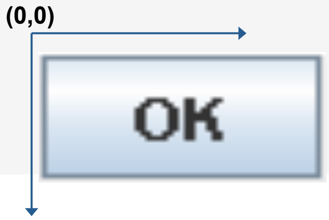
Rendering methods

- Draw, fill shapes
- Draw text strings
- Draw images



Enter Text:

Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet.



Working with Widgets

Make the common case fast and the uncommon case possible.

Common case: assemble standard widgets into a layout

Uncommon case: write your own widget.

Working with Widgets

Make the common case fast and the uncommon case possible.

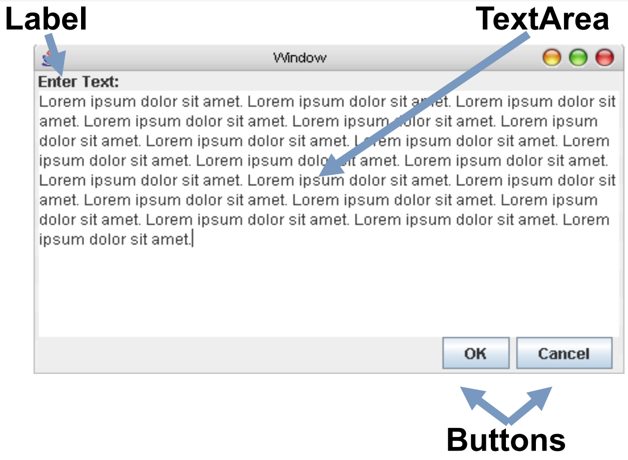
Common case: assemble standard widgets into a layout

Uncommon case: write your own widget.

Custom Components in AndroidSDK :

- Extend `View` class
- Paint method: Override `onDraw()`
- Bounding box: Override `onMeasure()`
- Callbacks: Override `onTouchEvent(), onKeyDown,` …

Composing a User Interface

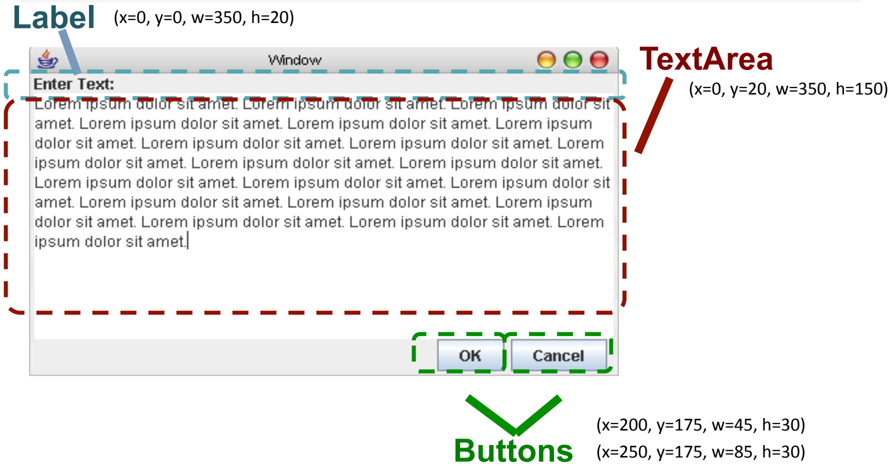


Enter Text:

Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet.

How might we instruct the computer to generate this layout?

Absolute Layout

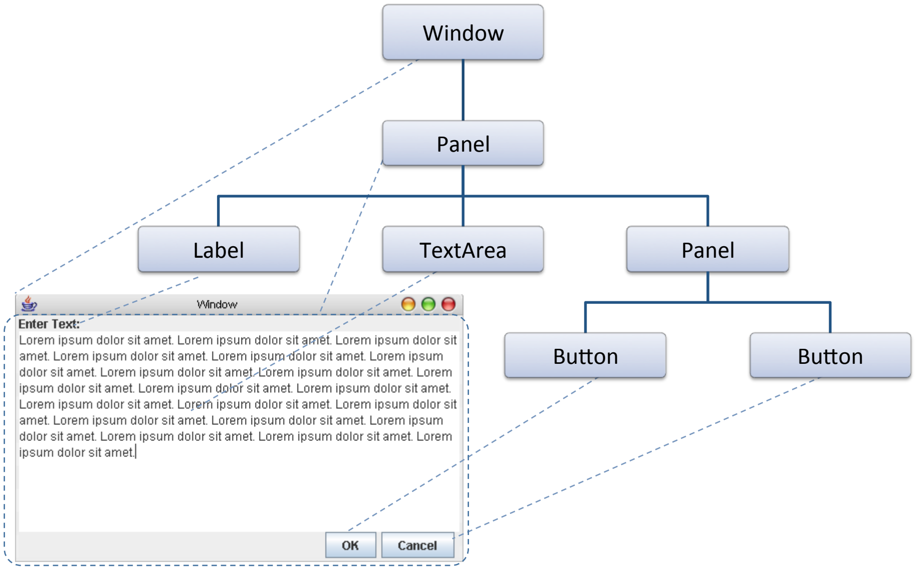

Label
(x=0, y=0, w=350, h=20)
Window
TextArea
(x=0, y=20, w=350, h=150)
OK
Cancel
Buttons
(x=200, y=175, w=45, h=30)
(x=250, y=175, w=85, h=30)

Enter Text:

Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet.

Absolute layout is inflexible and doesn't scale or resize well.
(But: great for prototyping because it's fast!)

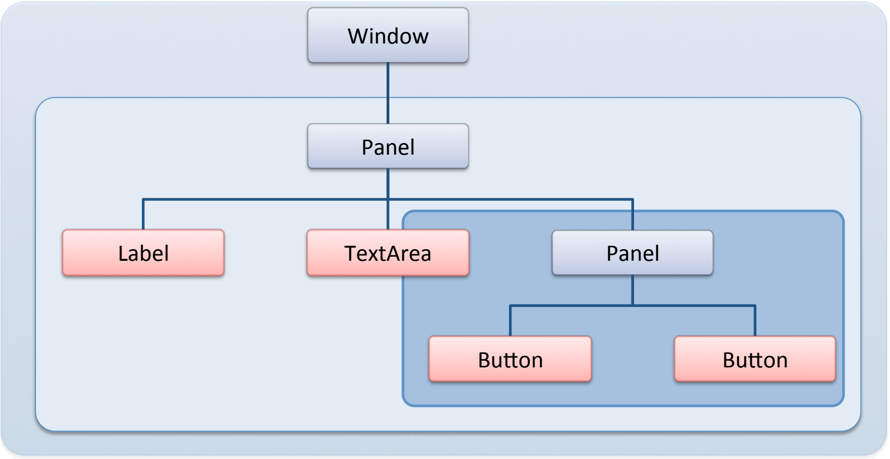
Containment Hierarchy



Enter Text:

Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet.

Containment Hierarchy



Principle: Each container is responsible for allocating space and positioning its contents.

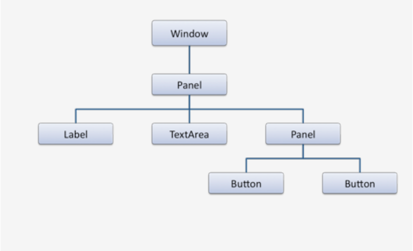
Common Hierarchical Layouts

1D Horizontal or Vertical List

2D Grid

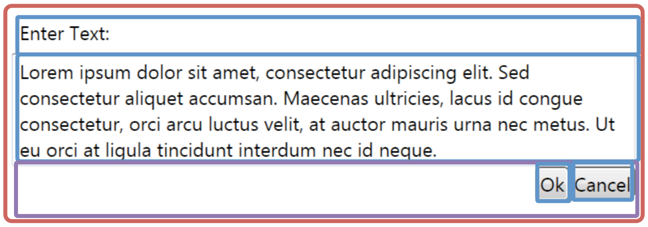
Constraint-based Layout (Struts+Springs)

Example Declarative Layout (WPF)



```
<StackPanel>
  <Label>Enter Text:</Label>
  <TextBox TextWrapping="Wrap”>…</TextBox>
  <StackPanel Orientation="Horizontal"
              HorizontalAlignment="Right">
    <Button>Ok</Button>
    <Button>Cancel</Button>
  </StackPanel>
</StackPanel>
```

Example Declarative Layout (WPF)

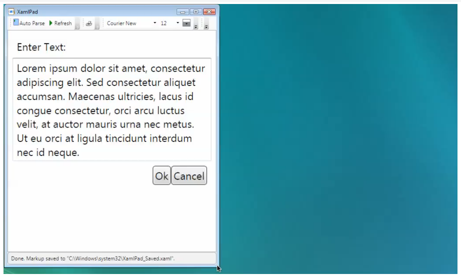


Enter Text:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed consectetur aliquet accumsan. Maecenas ultricies, lacus id congue consectetur, orci arcu luctus velit, at auctor mauris urna nec metus. Ut eu orci at ligula tincidunt interdum nec id neque.

```
<StackPanel>
  <Label>Enter Text:</Label>
  <TextBox TextWrapping="Wrap”>…</TextBox>
  <StackPanel Orientation="Horizontal"
              HorizontalAlignment="Right">
    <Button>Ok</Button>
    <Button>Cancel</Button>
  </StackPanel>
</StackPanel>
```

Example Declarative Layout (WPF)



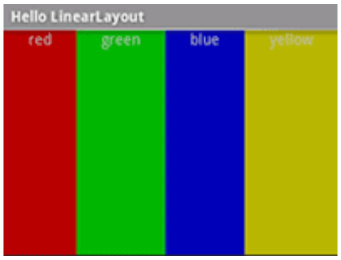
Enter Text:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed consectetur aliquet accumsan. Maecenas ultricies, lacus id congue consectetur, orci arcu luctus velit, at auctor mauris urna nec metus. Ut eu orci at ligula tincidunt interdum nec id neque.

Done. Markup saved to "C:\Windows\system32\XamlPad_Saved.xaml".

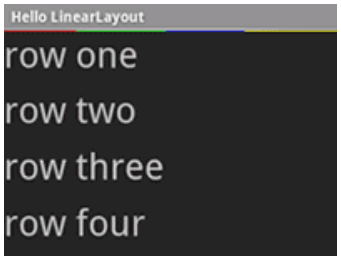
```
<StackPanel>
  <Label>Enter Text:</Label>
  <TextBox TextWrapping="Wrap”>…</TextBox>
  <StackPanel Orientation="Horizontal"
              HorizontalAlignment="Right">
    <Button>Ok</Button>
    <Button>Cancel</Button>
  </StackPanel>
</StackPanel>
```

Android Layouts



```
<LinearLayout orientation="horizontal">
   <TextView text="red" background=”…"/>
   <TextView text="green" background=”…"/>
   <TextView text="blue" background=”…”/>
   <TextView text="yellow" background=”…”/>
</LinearLayout>
```

Android Layouts



Hello LinearLayout

row one

row two

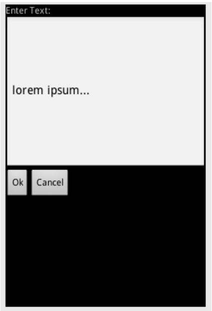
row three

row four

```
<LinearLayout orientation="vertical">
   <TextView text="row one" .../>
   <TextView text="row two" .../>
   <TextView text="row three" .../>
   <TextView text="row four" .../>
</LinearLayout>
```

In Android

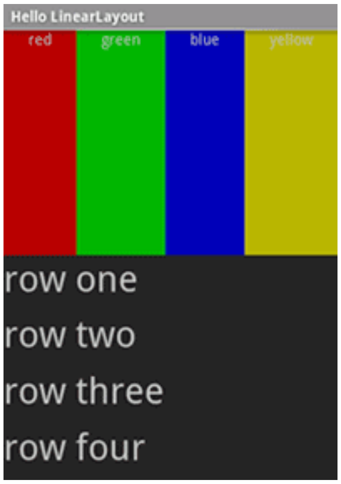
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout orientation="vertical”…>

  <TextView  text="Enter Text:”></TextView>
  <EditText text="lorem ipsum...” ></EditText>
  <LinearLayout orientation="horizontal”>
    <Button text=“Ok”></Button>
    <Button text=“Cancel”></Button>
  </LinearLayout>
</LinearLayout>
```



Enter Text:

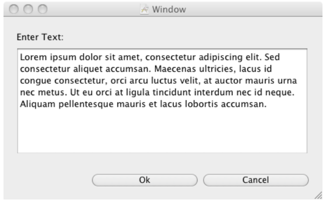
lorem ipsum...

Android Layouts



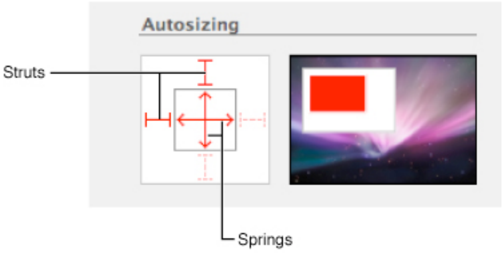
```
<LinearLayout orientation="vertical”>
  <LinearLayout orientation="horizontal">
   <TextView text="red" background=”…"/>
   <TextView text="green" background=”…"/>
   <TextView text="blue" background=”…”/>
   <TextView text="yellow" background=”…”/>
  </LinearLayout>

  <LinearLayout orientation="vertical">
   <TextView text="row one" .../>
   <TextView text="row two" .../>
   <TextView text="row three" .../>
   <TextView text="row four" .../>
  </LinearLayout>
</LinearLayout>
```

Layout in Cocoa: Springs + Struts



Enter Text:

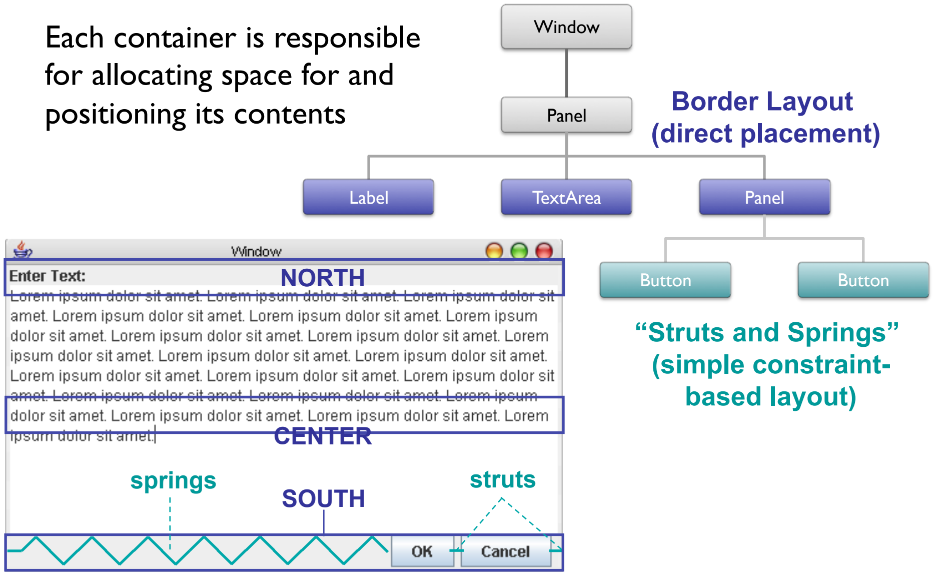
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed consectetur aliquet accumsan. Maecenas ultricies, lacus id congue consectetur, orci arcu luctus velit, at auctor mauris urna nec metus. Ut eu orci at ligula tincidunt interdum nec id neque. Aliquam pellentesque mauris et lacus lobortis accumsan.



Interface Builder Demo

Component Layout

Each container is responsible for allocating space for and positioning its contents



Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet.

Border Layout
(direct placement)

"Struts and Springs"
(simple constraint-based layout)

Specifying Layout

Declarative
e.g., HTML, XAML, MXML,…

```
<StackPanel>
  <Label>Enter Text:</Label>
  <TextBox TextWrapping="Wrap”>…</TextBox>
  <StackPanel Orientation="Horizontal"
              HorizontalAlignment="Right">
    <Button>Ok</Button>
    <Button>Cancel</Button>
  </StackPanel>
</StackPanel>
```

Procedural
e.g., Java Swing

GUI Builders exist for either approach (but generating procedural code is brittle)

Is your UI layout determined statically or dynamically at runtime? If at runtime, may need procedural approach.

Specifying Layout

Declarative
e.g., HTML, XAML, MXML,…

Procedural
e.g., Java Swing

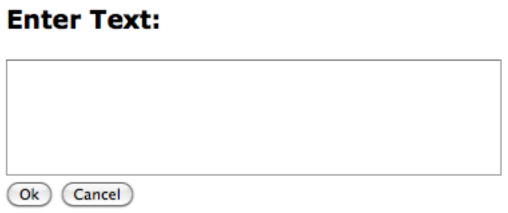
GUI Builders exist for either approach (but generating procedural code is brittle)

Is your UI layout determined statically or dynamically at runtime? If at runtime, may need procedural approach.

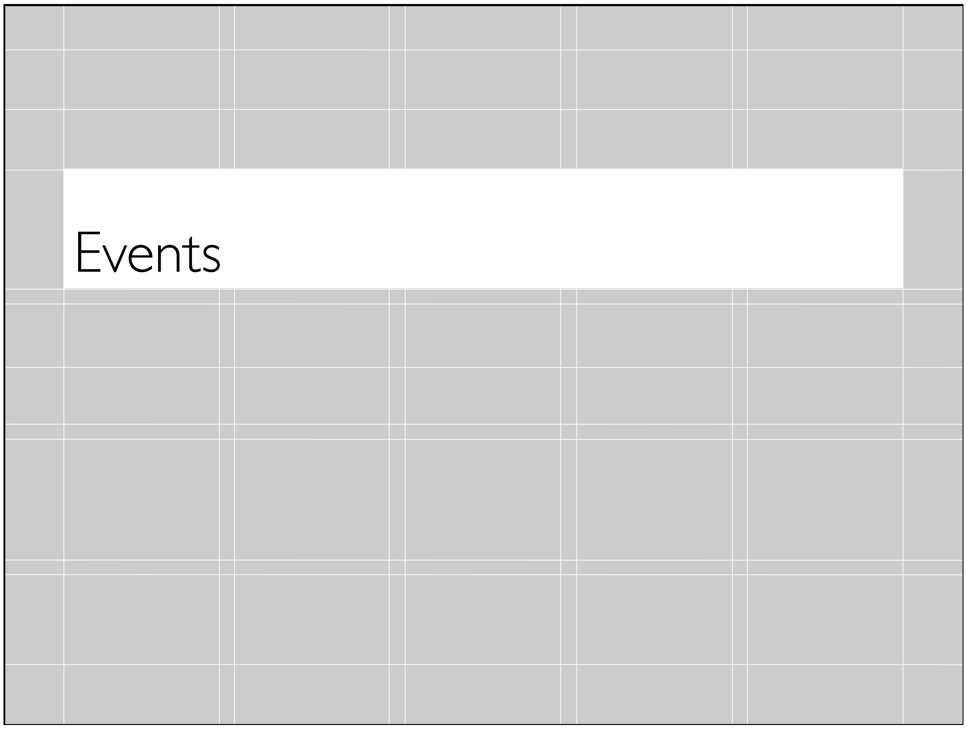
```
public void init() {
    Container c = getContentPane();
    c.setLayout(new BorderLayout());
    c.add(new JButton("One"),
         BorderLayout.NORTH);
    c.add(new JButton("Two"),
         BorderLayout.WEST);
    c.add(new JButton("Three"),
        BorderLayout.CENTER);
  }
```

HTML – What kind of Layout?

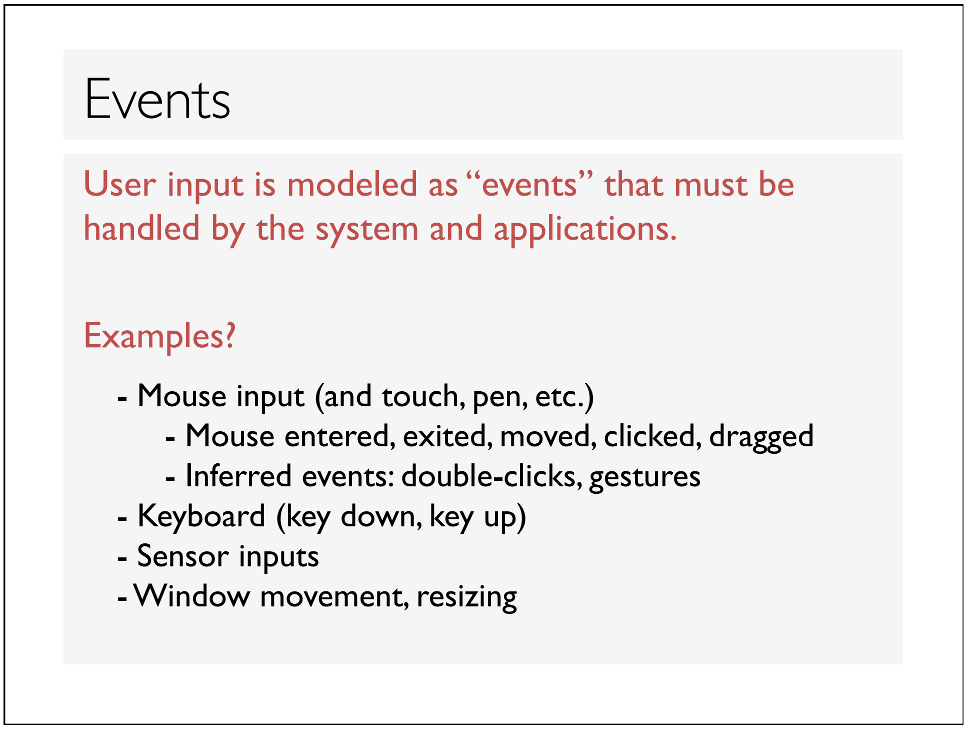
```
<h3>Enter Text:</h3>
<form method="post" action="">
    <textarea name="theText" cols="45" rows="5"></textarea>
    <br/>
    <input type="submit" name="btnOK"  value="Ok" />
    <input type="submit" name="btnCancel" id="button" value="Cancel" />
</form>
```



Enter Text:



Events

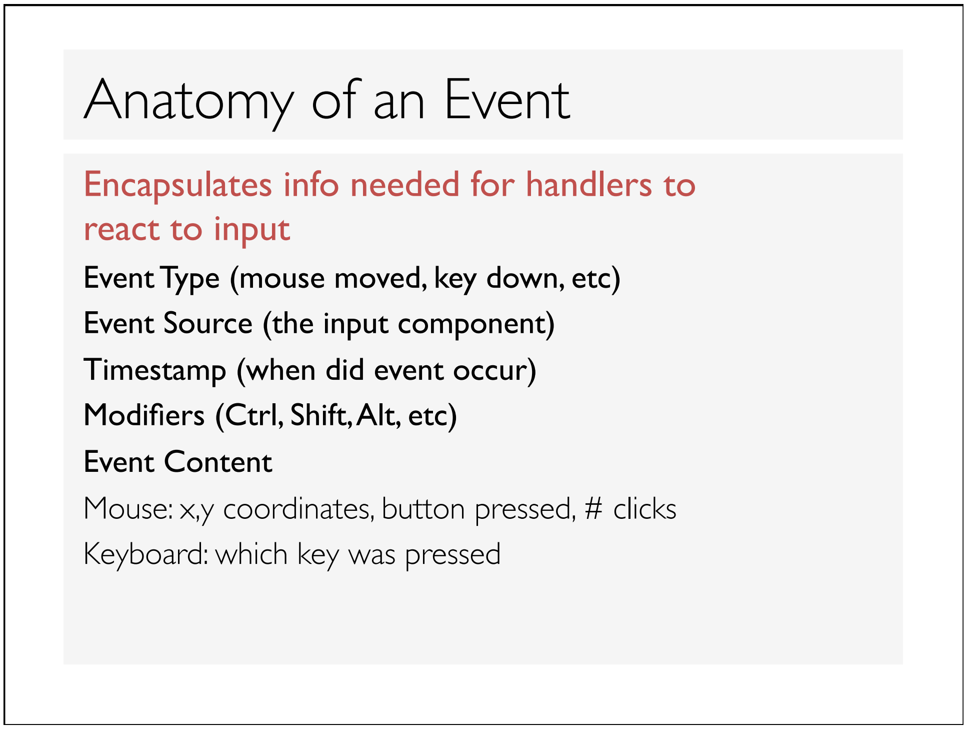


Events

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

Examples?

- 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

Encapsulates info needed for handlers to react to input

Event Type (mouse moved, key down, etc)

Event Source (the input component)

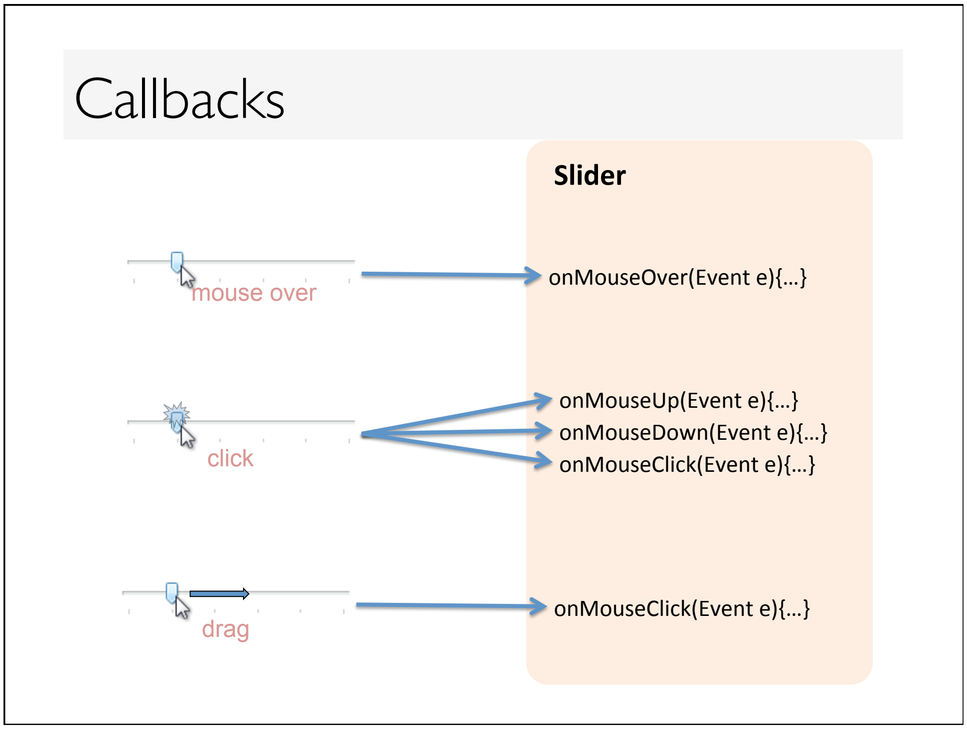
Timestamp (when did event occur)

Modifiers (Ctrl, Shift, Alt, etc)

Event Content

Mouse: x,y coordinates, button pressed, # clicks

Keyboard: which key was pressed



Callbacks

Slider

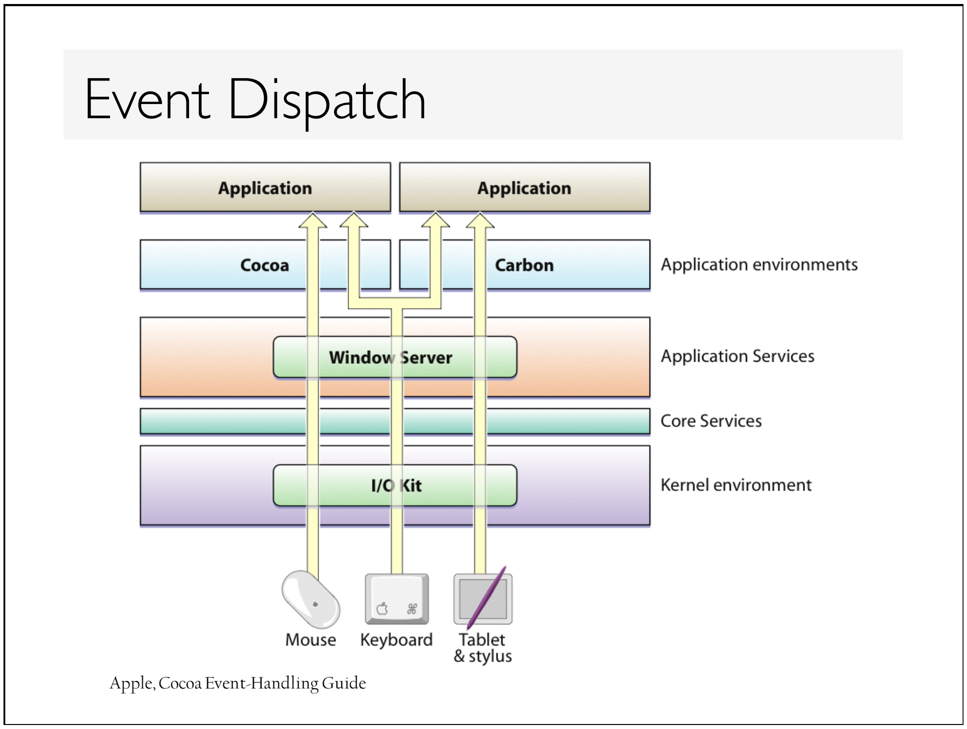
onMouseOver(Event e){…}

onMouseUp(Event e){…}

onMouseDown(Event e){…}

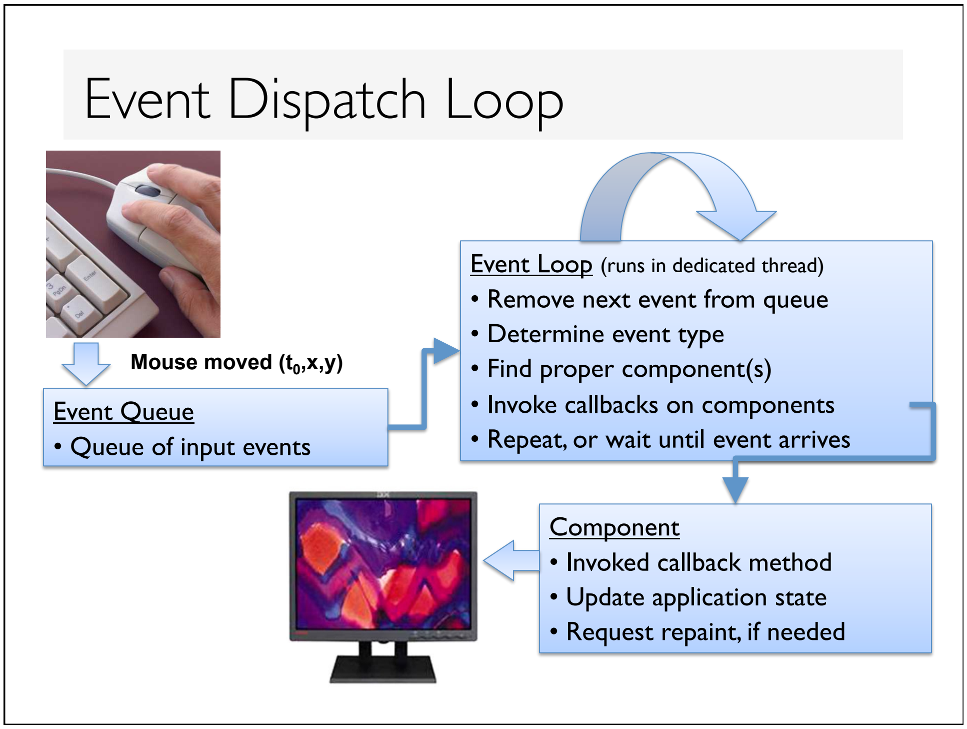
onMouseClick(Event e){…}

onMouseClick(Event e){…}



Event Dispatch

Apple, Cocoa Event-Handling Guide



Event Dispatch Loop

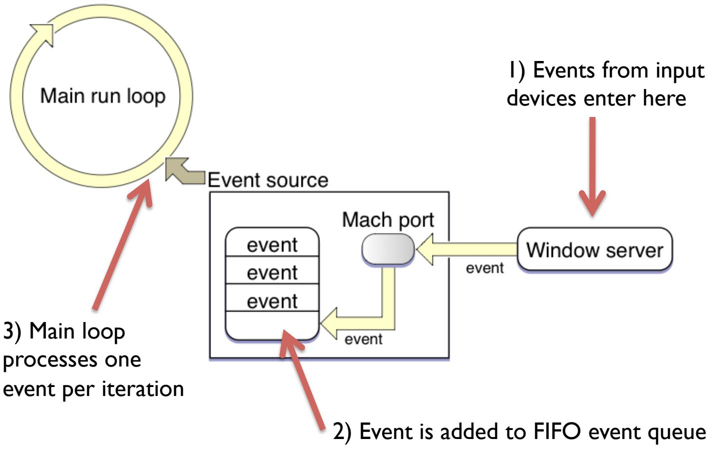
Mouse moved (t_0,x,y)

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

Event Dispatch Loop



1) Events from input devices enter here

2) Event is added to FIFO event queue

3) Main loop processes one event per iteration

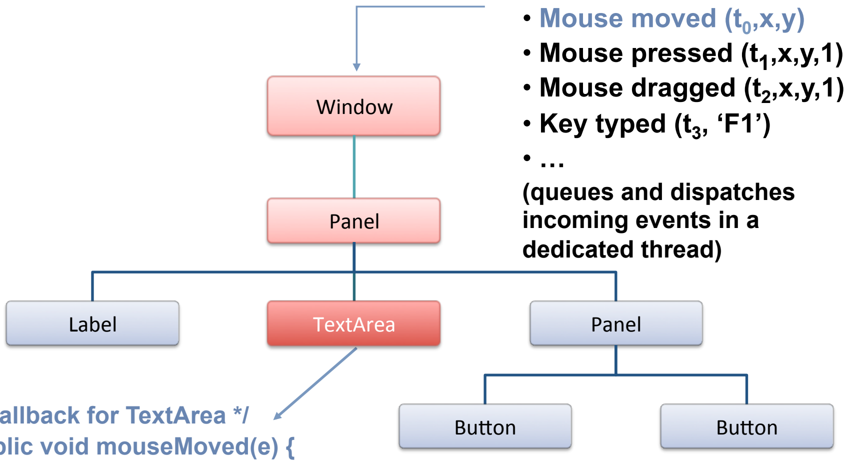
Apple, Cocoa Event-Handling Guide

Event Dispatch

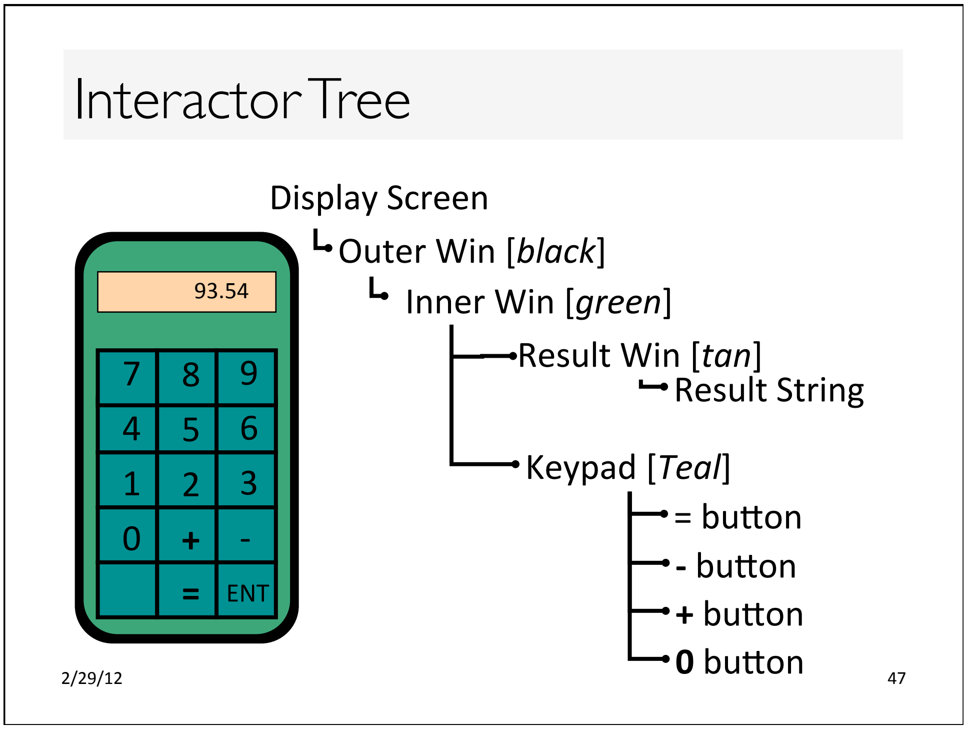
Event Queue

- 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)



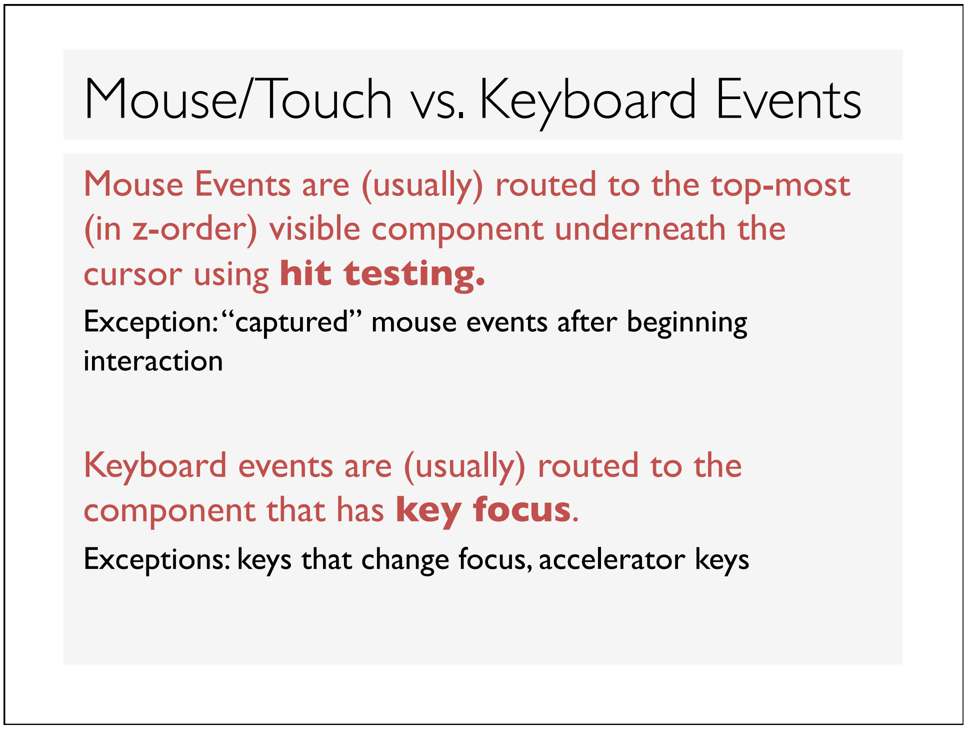
```
/* callback for TextArea */
public void mouseMoved(e) {
    // process mouse moved event
}
```



Interactor Tree

Display Screen
- Outer Win [*black*]
  - Inner Win [*green*]
    - Result Win [*tan*]
      - Result String
    - Keypad [*Teal*]
      - = button
      - - button
      - + button
      - **0** button

2/29/12 47



Mouse/Touch vs. Keyboard Events

Mouse Events are (usually) routed to the top-most (in z-order) visible component underneath the cursor using **hit testing.**

Exception: "captured" mouse events after beginning interaction

Keyboard events are (usually) routed to the component that has **key focus**.

Exceptions: keys that change focus, accelerator keys

Event Dispatch in ObjC / Cocoa

Mouse events:

Dispatched to NSView of object under cursor

Keyboard events:

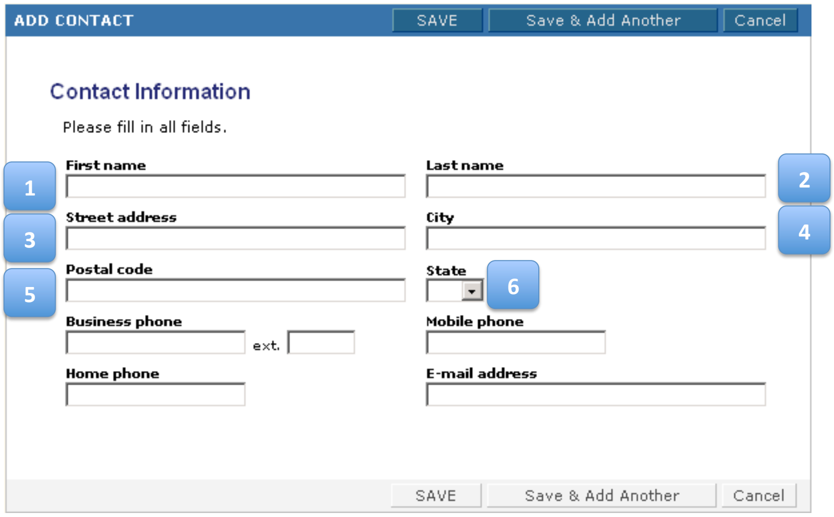
Dispatched to "first responder" (i.e., object in focus)

Default NSView implementation does not handle, forwards to "next responder":

"the event, if not handled, proceeds up the view hierarchy to the NSWindow object representing the window itself." (Apple)

If view does

Key Focus: Form Example



Contact Information

Please fill in all fields.

First name

Last name

Street address

City

Postal code

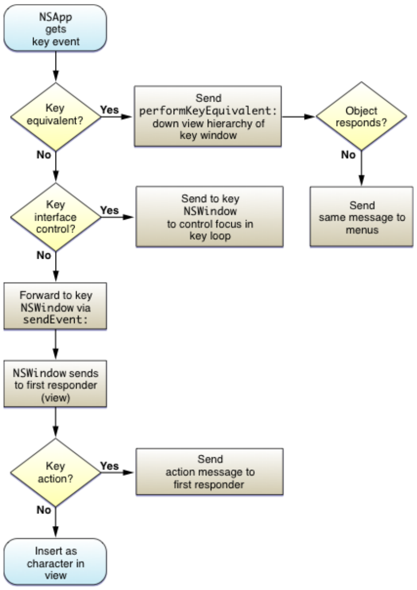
State

Business phone

Mobile phone

Home phone

E-mail address



NSApp gets key event

Key equivalent?

Yes

Send performKeyEquivalent: down view hierarchy of key window

Object responds?

No

Send same message to menus

No

Key interface control?

Yes

Send to key NSWindow to control focus in key loop

No

Forward to key NSWindow via sendEvent:

NSWindow sends to first responder (view)

Key action?

Yes

Send action message to first responder

No

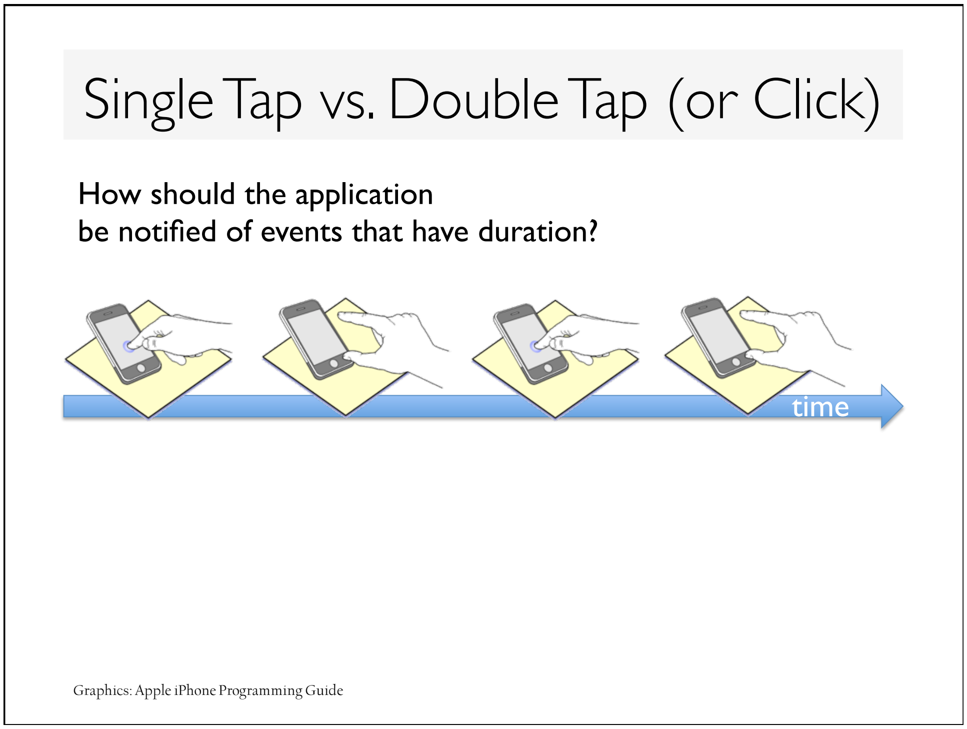
Insert as character in view

Abstracting Events

Level of abstraction may vary. Consider:

Mouse down vs. **double click** vs. **drag**

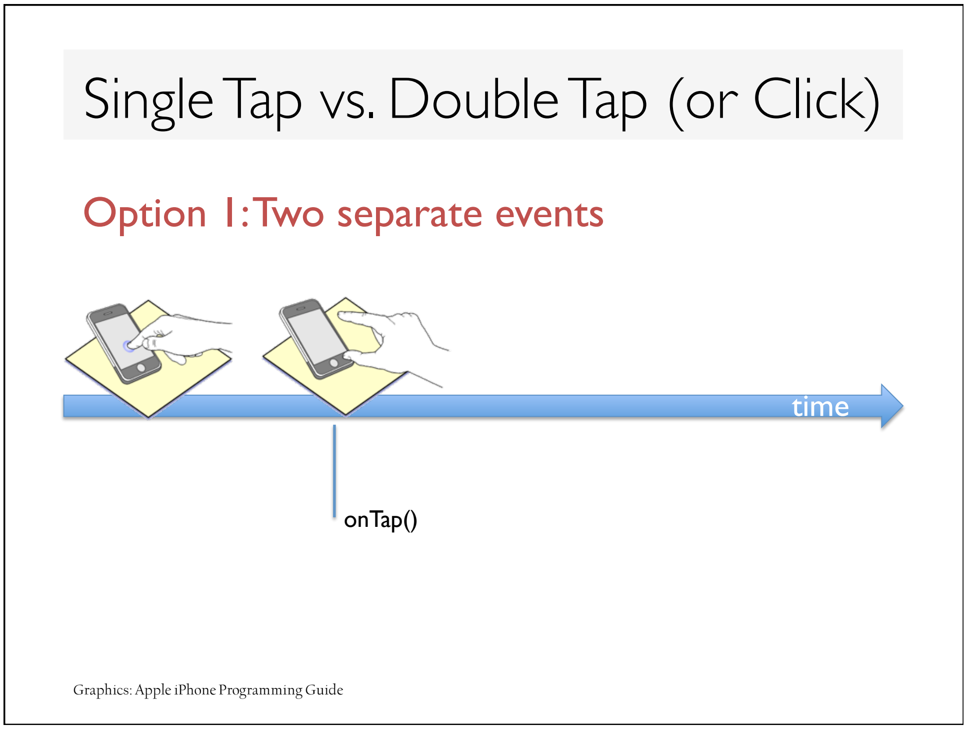
Pen move vs. **gesture**



Single Tap vs. Double Tap (or Click)

How should the application
be notified of events that have duration?

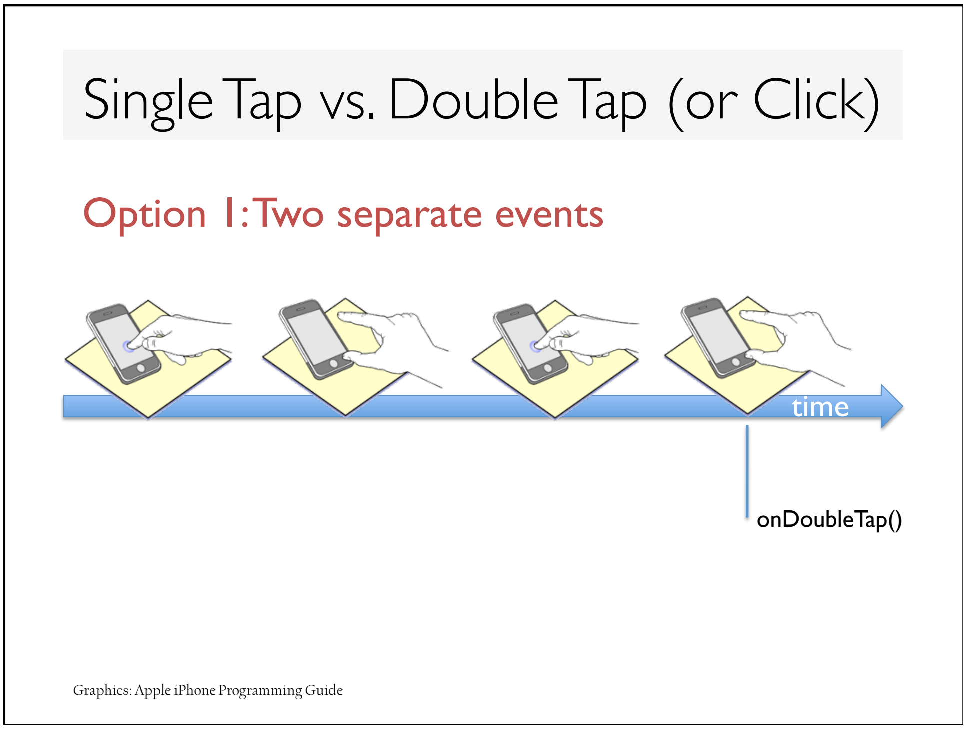
Graphics: Apple iPhone Programming Guide



Single Tap vs. Double Tap (or Click)

Option 1: Two separate events

Graphics: Apple iPhone Programming Guide

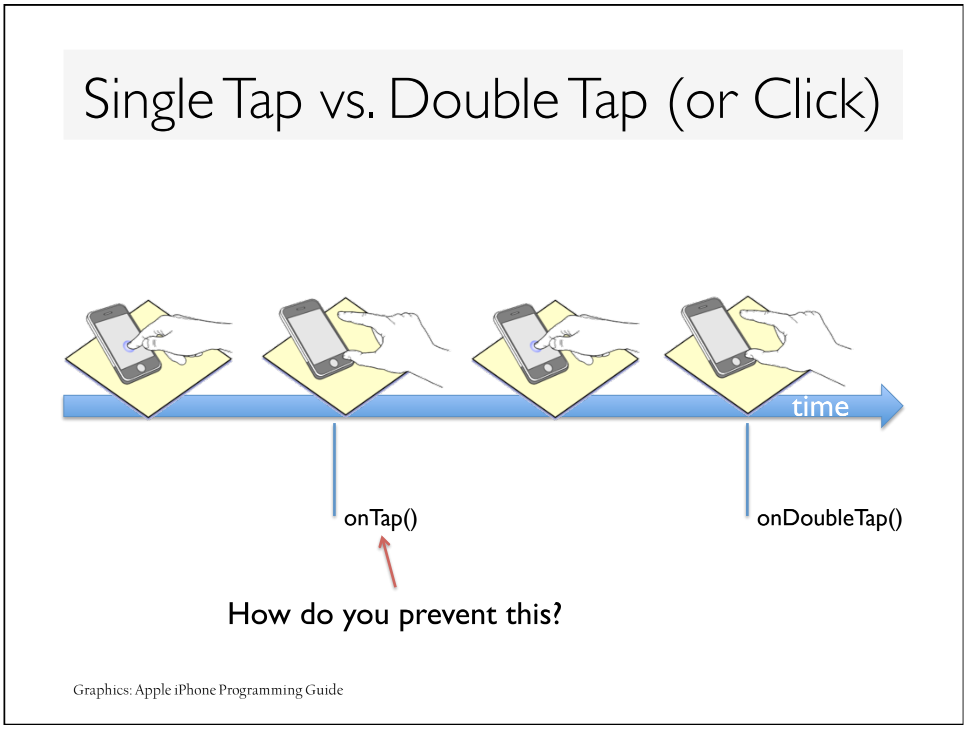


Single Tap vs. Double Tap (or Click)

Option 1: Two separate events

onDoubleTap()

Graphics: Apple iPhone Programming Guide



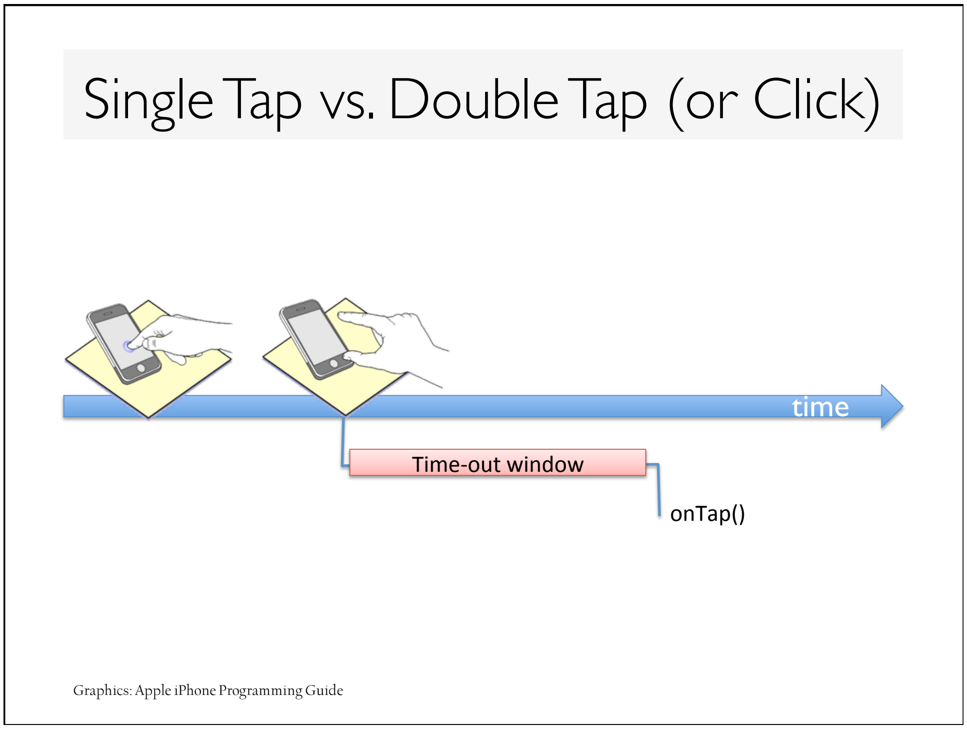
Single Tap vs. Double Tap (or Click)

onTap()

onDoubleTap()

How do you prevent this?

Graphics: Apple iPhone Programming Guide



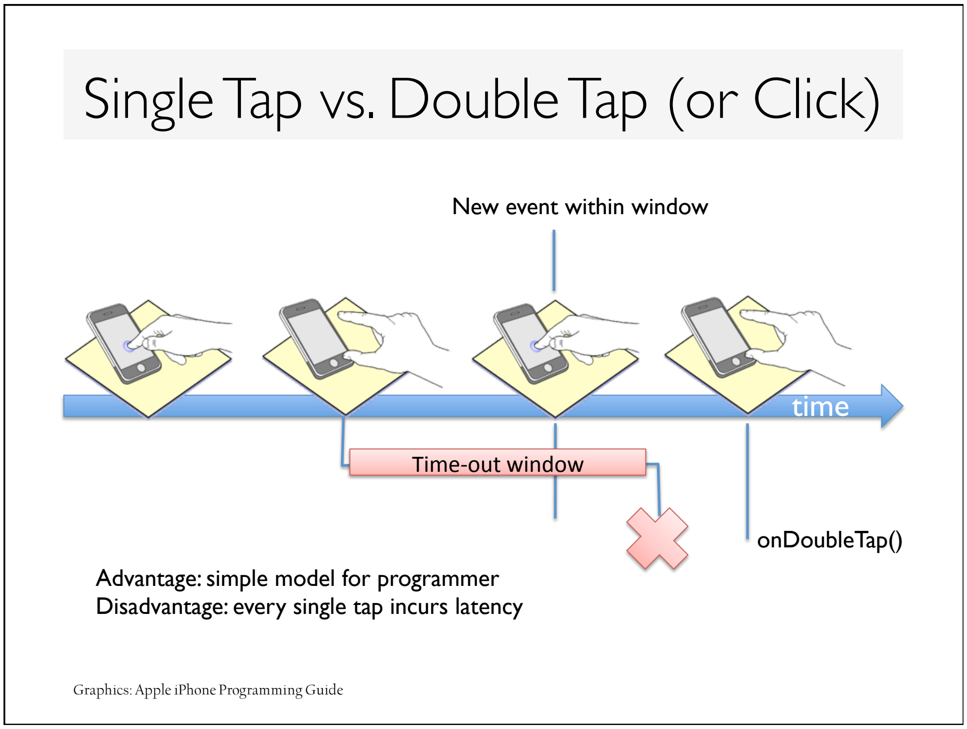
Single Tap vs. Double Tap (or Click)

time

Time-out window

onTap()

Graphics: Apple iPhone Programming Guide



Single Tap vs. Double Tap (or Click)

New event within window

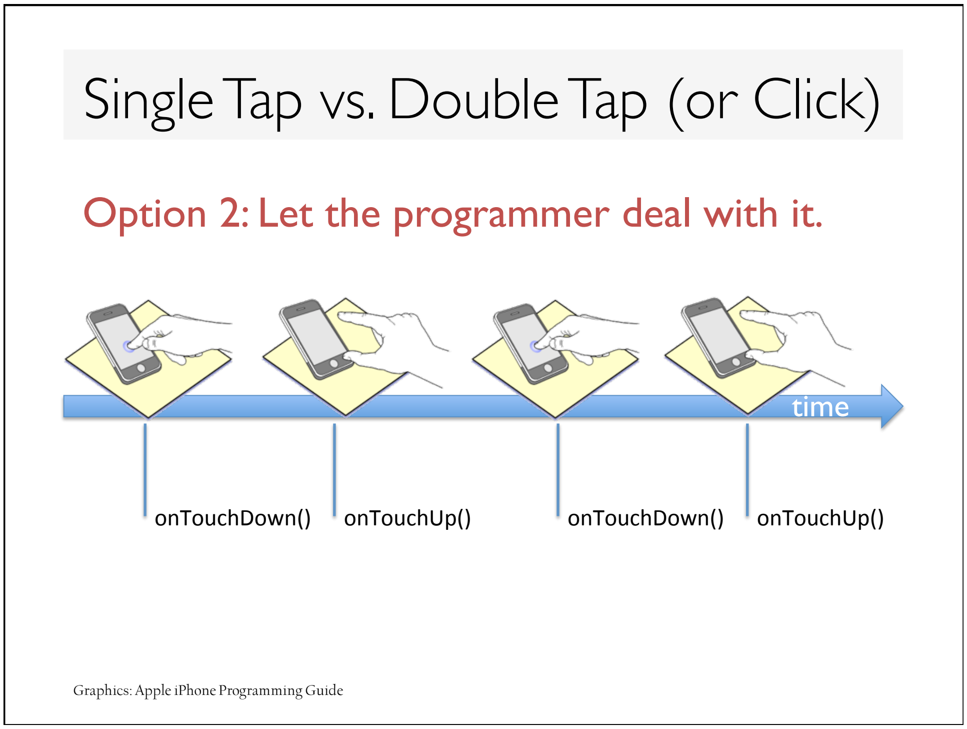
time

Time-out window

onDoubleTap()

Advantage: simple model for programmer
Disadvantage: every single tap incurs latency

Graphics: Apple iPhone Programming Guide



Single Tap vs. Double Tap (or Click)

Option 2: Let the programmer deal with it.

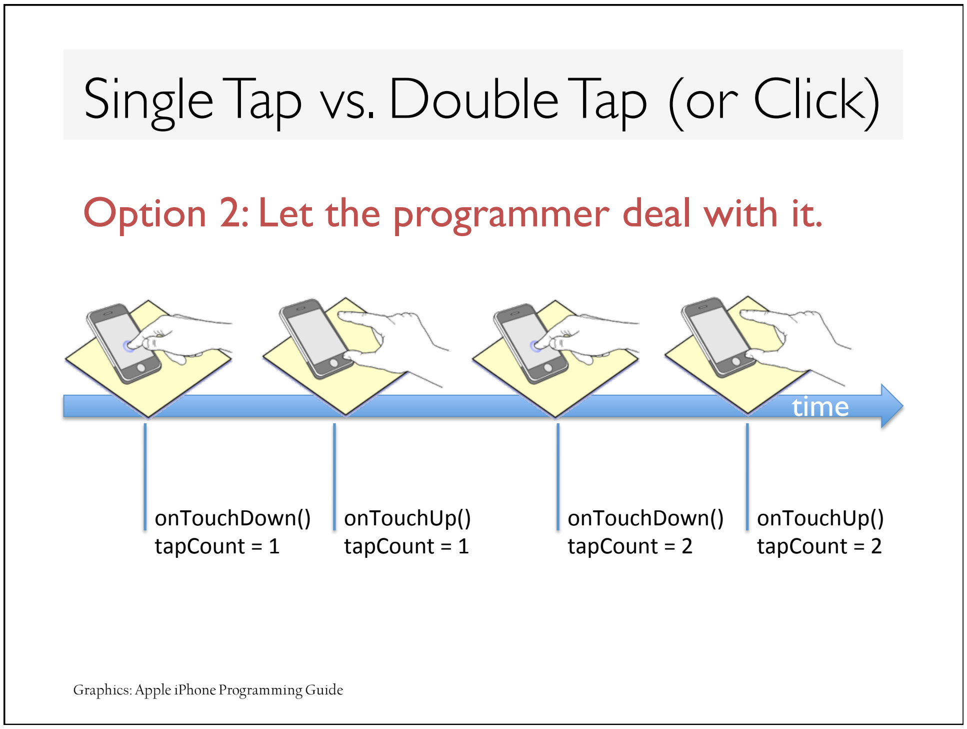
onTouchDown()

onTouchUp()

onTouchDown()

onTouchUp()

Graphics: Apple iPhone Programming Guide



Single Tap vs. Double Tap (or Click)

Option 2: Let the programmer deal with it.

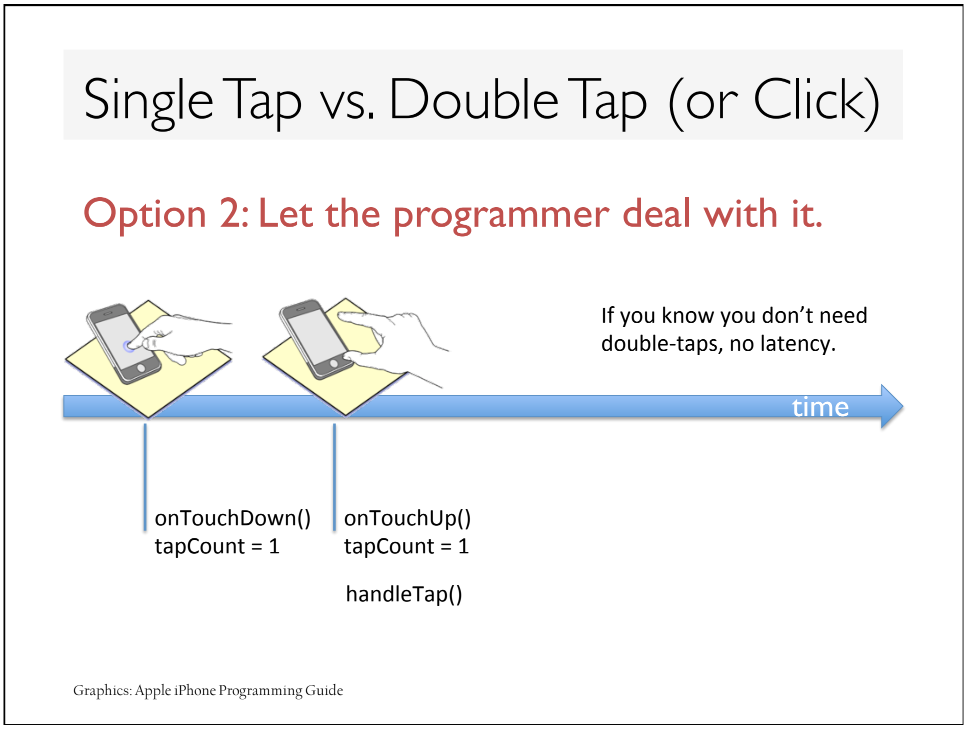
onTouchDown()
tapCount = 1

onTouchUp()
tapCount = 1

onTouchDown()
tapCount = 2

onTouchUp()
tapCount = 2

Graphics: Apple iPhone Programming Guide



Single Tap vs. Double Tap (or Click)

Option 2: Let the programmer deal with it.

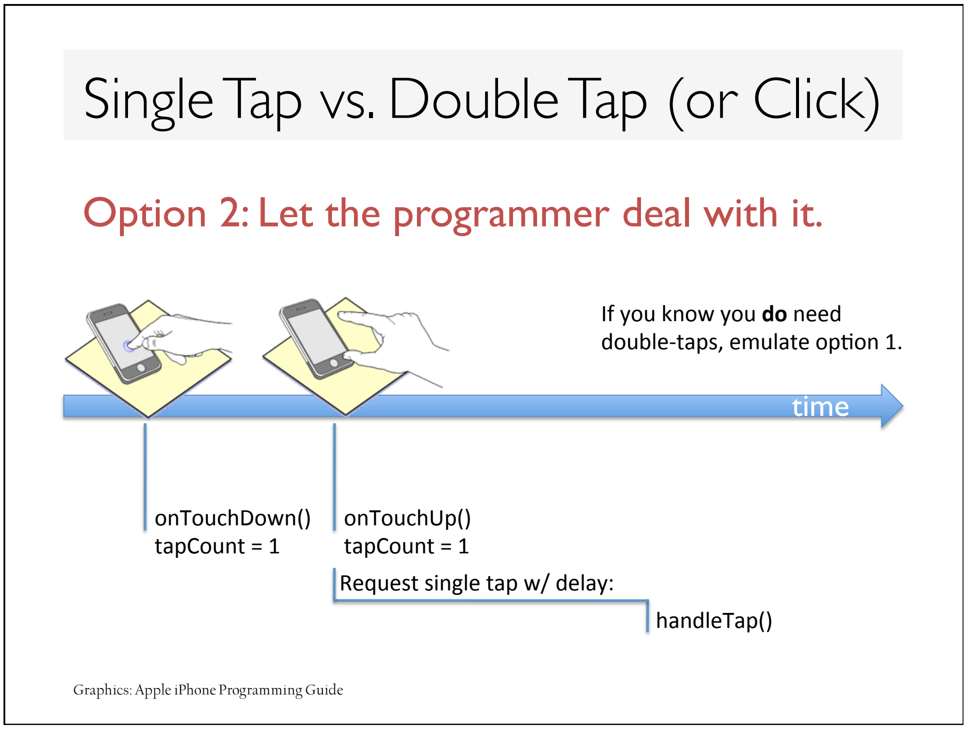
If you know you don't need double-taps, no latency.

onTouchDown()
tapCount = 1

onTouchUp()
tapCount = 1

handleTap()

Graphics: Apple iPhone Programming Guide



Single Tap vs. Double Tap (or Click)

Option 2: Let the programmer deal with it.

If you know you **do** need double-taps, emulate option 1.

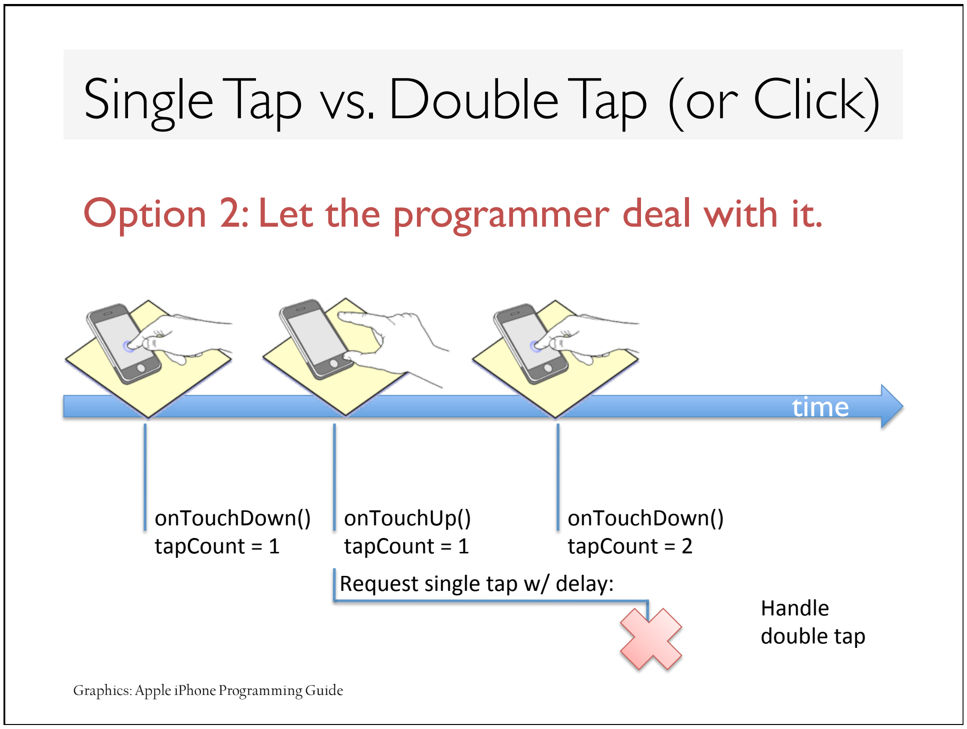
onTouchDown()
tapCount = 1

onTouchUp()
tapCount = 1

Request single tap w/ delay:

handleTap()

Graphics: Apple iPhone Programming Guide



Single Tap vs. Double Tap (or Click)

Option 2: Let the programmer deal with it.

onTouchDown()
tapCount = 1

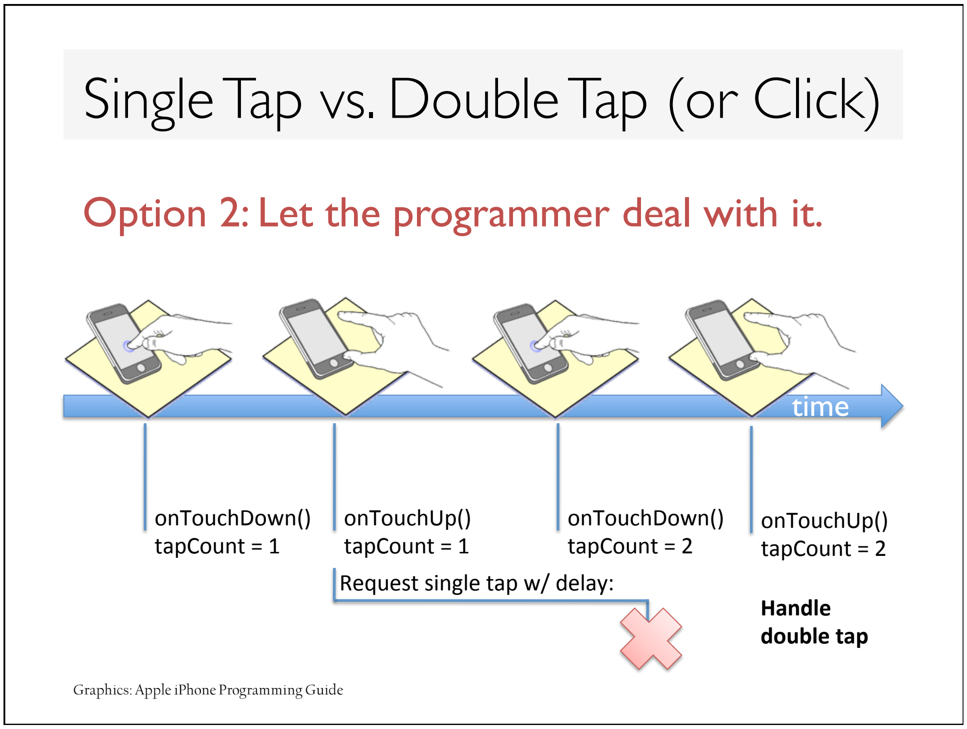
onTouchUp()
tapCount = 1

onTouchDown()
tapCount = 2

Request single tap w/ delay:

Handle
double tap

Graphics: Apple iPhone Programming Guide



Single Tap vs. Double Tap (or Click)

Option 2: Let the programmer deal with it.

onTouchDown()
tapCount = 1

onTouchUp()
tapCount = 1

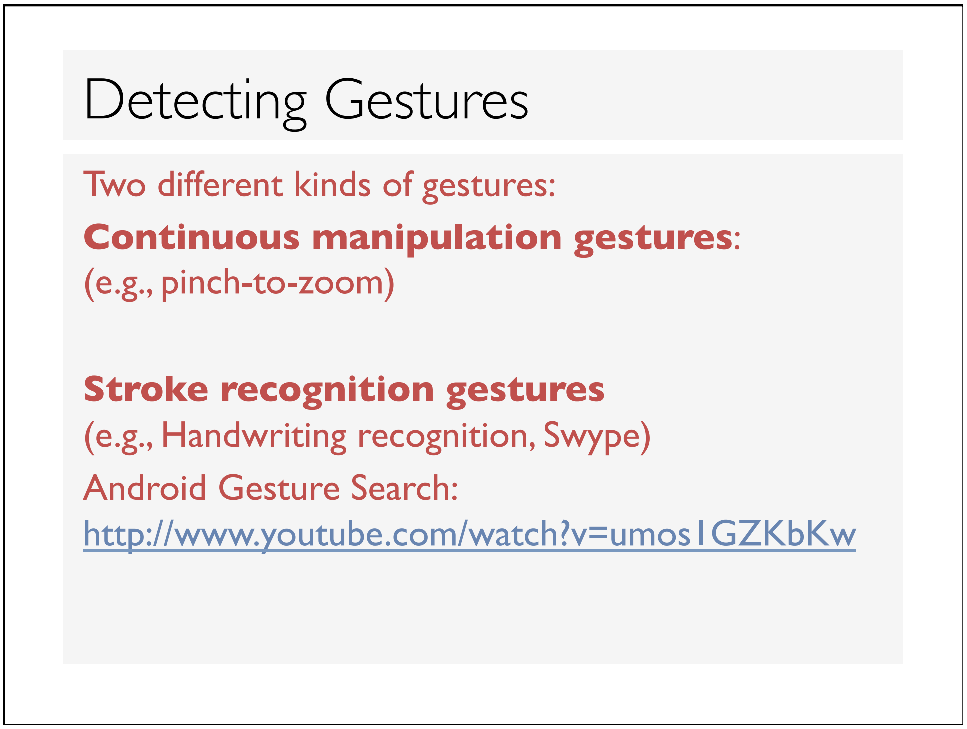
onTouchDown()
tapCount = 2

onTouchUp()
tapCount = 2

Request single tap w/ delay:

Handle
double tap

Graphics: Apple iPhone Programming Guide



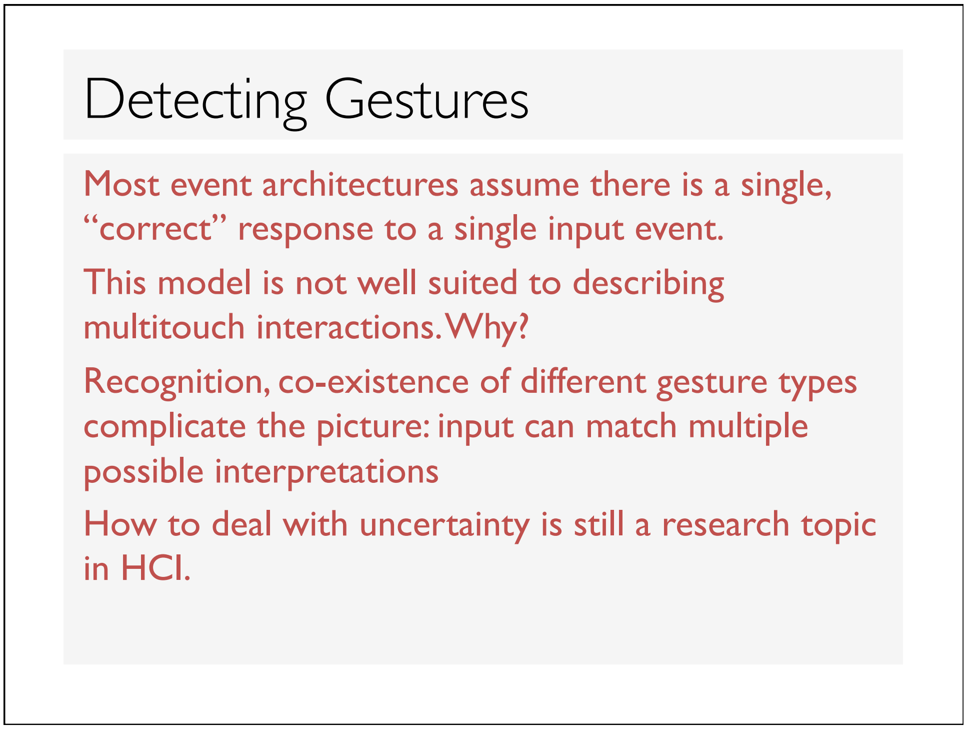
Detecting Gestures

Two different kinds of gestures:

Continuous manipulation gestures: (e.g., pinch-to-zoom)

Stroke recognition gestures (e.g., Handwriting recognition, Swype)

Android Gesture Search: http://www.youtube.com/watch?v=umos1GZKbKw



Detecting Gestures

Most event architectures assume there is a single, “correct” response to a single input event.

This model is not well suited to describing multitouch interactions. Why?

Recognition, co-existence of different gesture types complicate the picture: input can match multiple possible interpretations

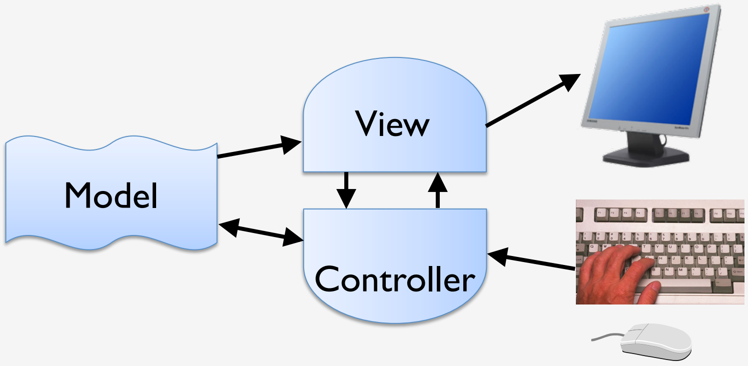
How to deal with uncertainty is still a research topic in HCI.

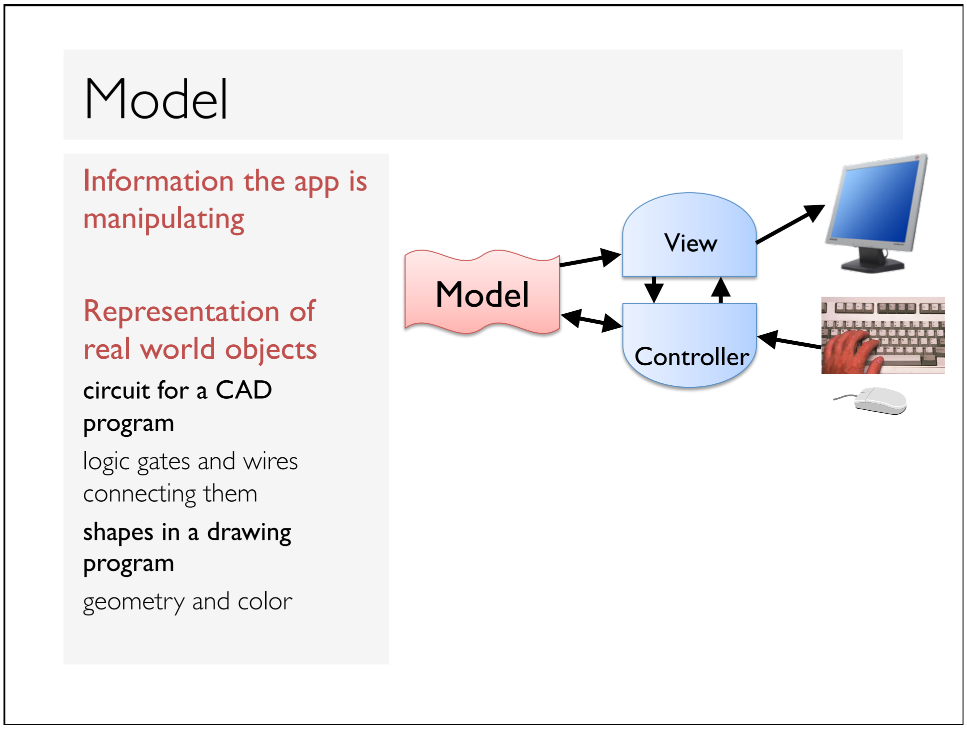
Model-View-Controller Architecture

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

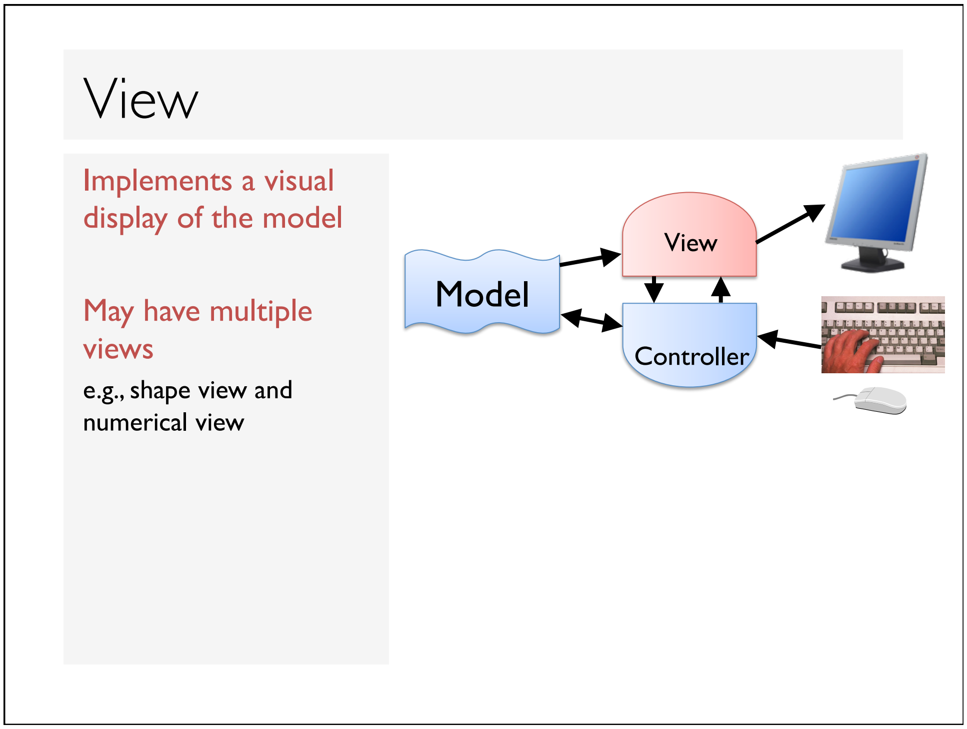
shapes in a drawing program

geometry and color

Model

View

Controller



View

Implements a visual display of the model

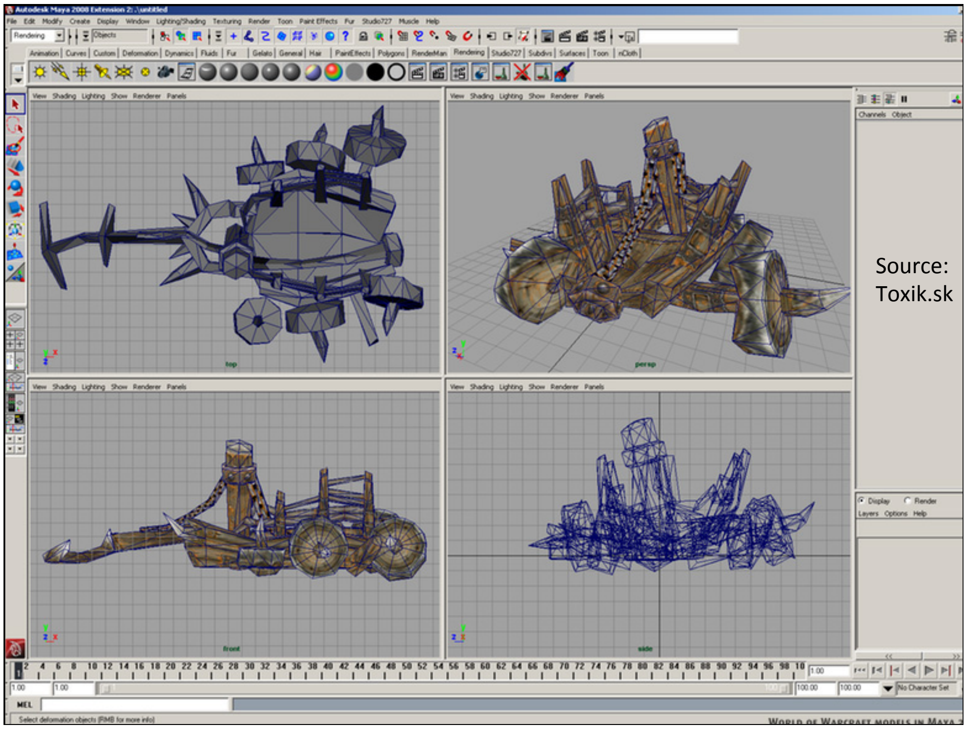
May have multiple views

e.g., shape view and numerical view

Model

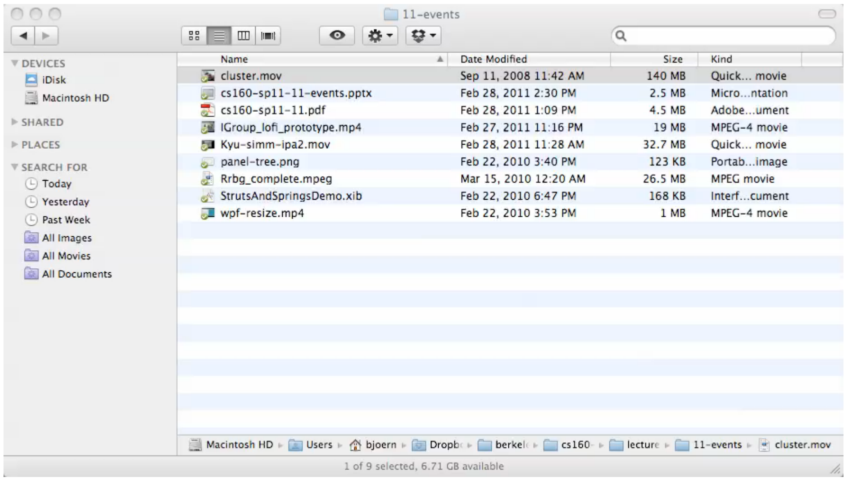
View

Controller


Autodesk Maya 2008 Extension 2: .\untitled
top
persp
front
side
World of Warcraft models in Maya

Source: Toxik.sk

Multiple Views



Name	Date Modified	Size	Kind
cluster.mov	Sep 11, 2008 11:42 AM	140 MB	Quick... movie
cs160-sp11-11-events.pptx	Feb 28, 2011 2:30 PM	2.5 MB	Micro...ntation
cs160-sp11-11.pdf	Feb 28, 2011 1:09 PM	4.5 MB	Adobe...ument
IGroup_lofi_prototype.mp4	Feb 27, 2011 11:16 PM	19 MB	MPEG-4 movie
Kyu-simm-ipa2.mov	Feb 28, 2011 11:28 AM	32.7 MB	Quick... movie
panel-tree.png	Feb 22, 2010 3:40 PM	123 KB	Portab...image
Rrbg_complete.mpeg	Mar 15, 2010 12:20 AM	26.5 MB	MPEG movie
StrutsAndSpringsDemo.xib	Feb 22, 2010 6:47 PM	168 KB	Interf...cument
wpf-resize.mp4	Feb 22, 2010 3:53 PM	1 MB	MPEG-4 movie

1 of 9 selected, 6.71 GB available

View

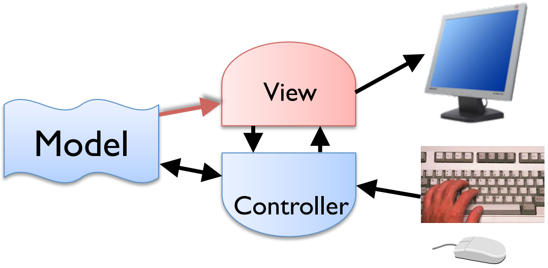
Implements a visual display of the model

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

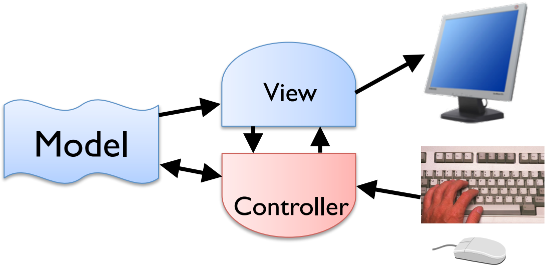
Receives all input events from the user

Decides what events mean and what to do

communicates with view to determine the objects being manipulated (e.g., selection)

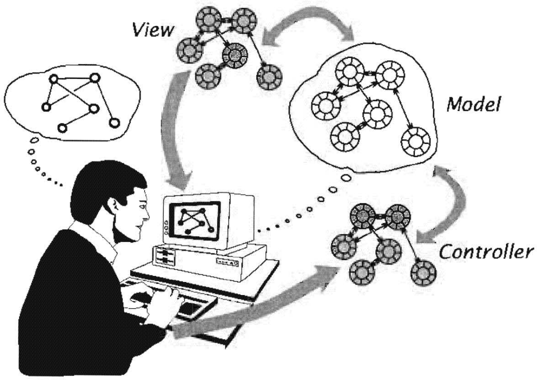
calls model methods to make changes on objects

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

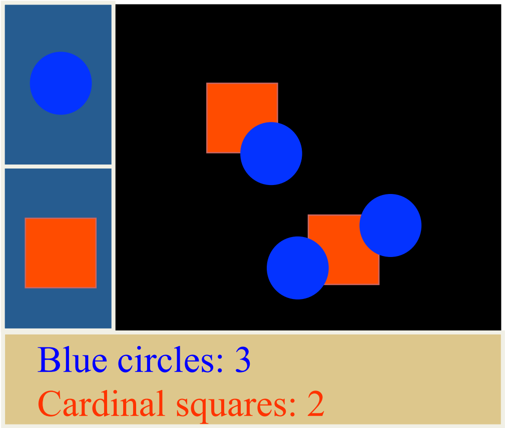
easy to add a new view later

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

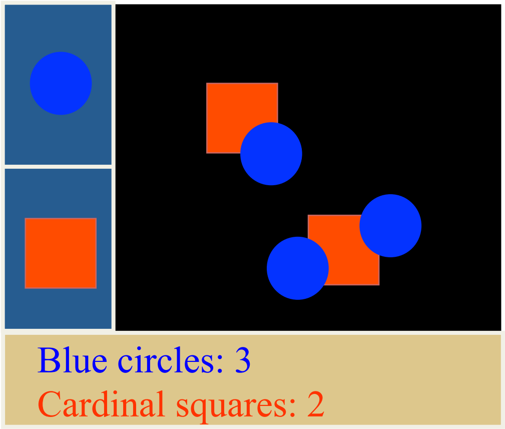


Blue circles: 3
Cardinal squares: 2

Model

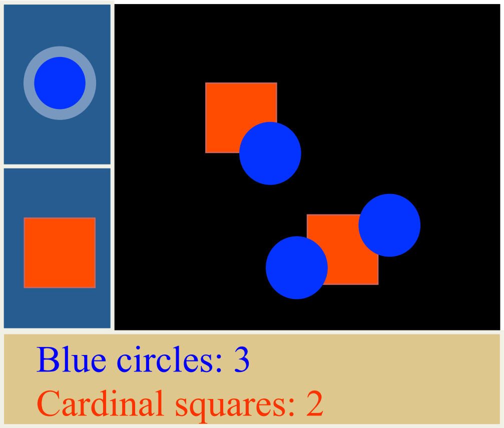
```
Class AppModel {
  ArrayList<Point> rectangles;
  ArrayList<Point> circles;
  Color rectangleColor;
  Color circleColor;


  …
}
```

Controller



Blue circles: 3
Cardinal squares: 2

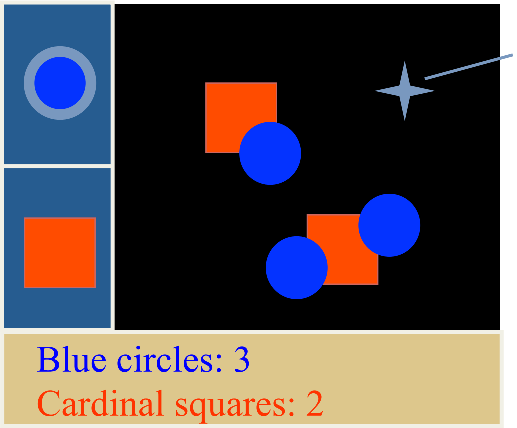
Controller



Blue circles: 3

Cardinal squares: 2

Controller

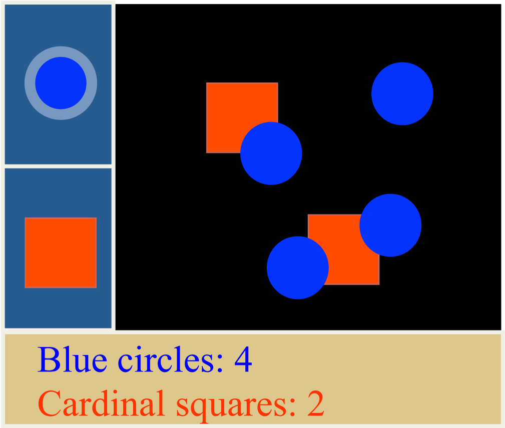


Click!

Blue circles: 3

Cardinal squares: 2

Controller



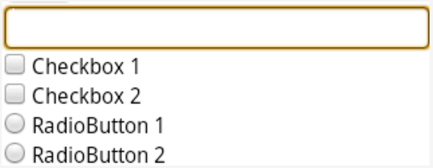
Blue circles: 4
Cardinal squares: 2

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

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



□ Checkbox 1
□ Checkbox 2
○ RadioButton 1
○ RadioButton 2

But controller must usually contact view to interpret what user events mean (e.g., selection)

Combining View & Controller

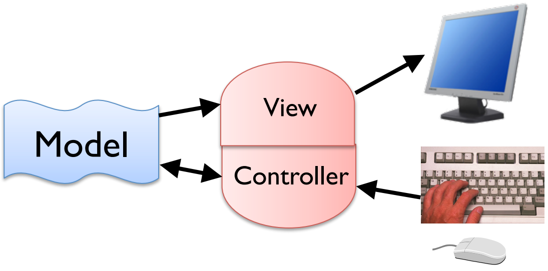
View and controller are tightly intertwined

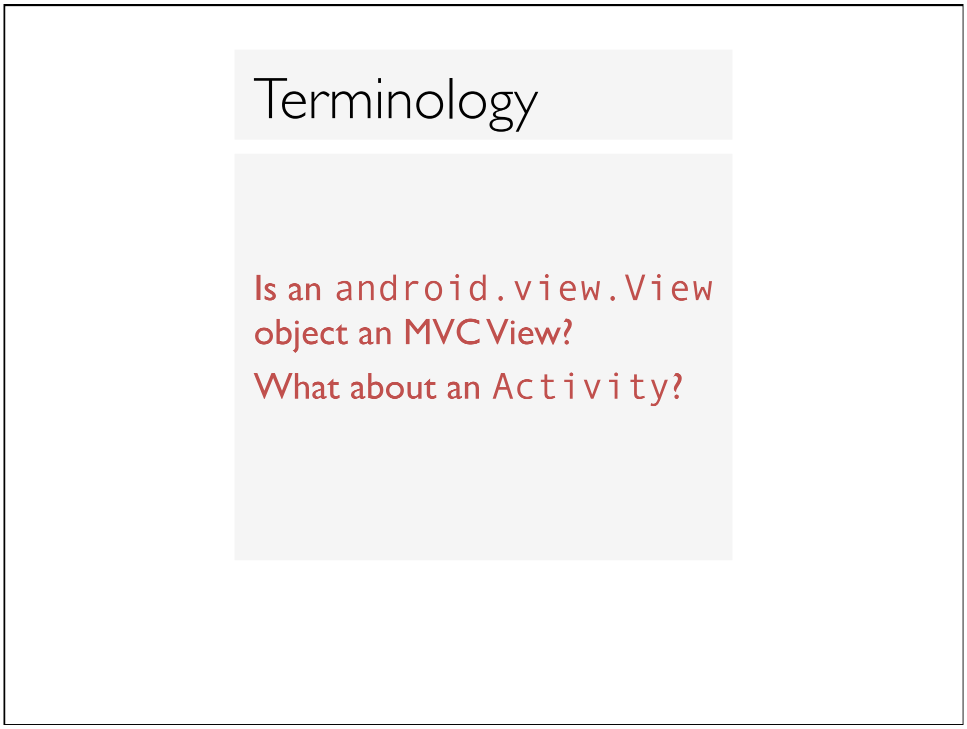
lots of communication between the two

Almost always occur in pairs

i.e., for each view, need a separate controller

Many architectures combine into a single class ("VC")

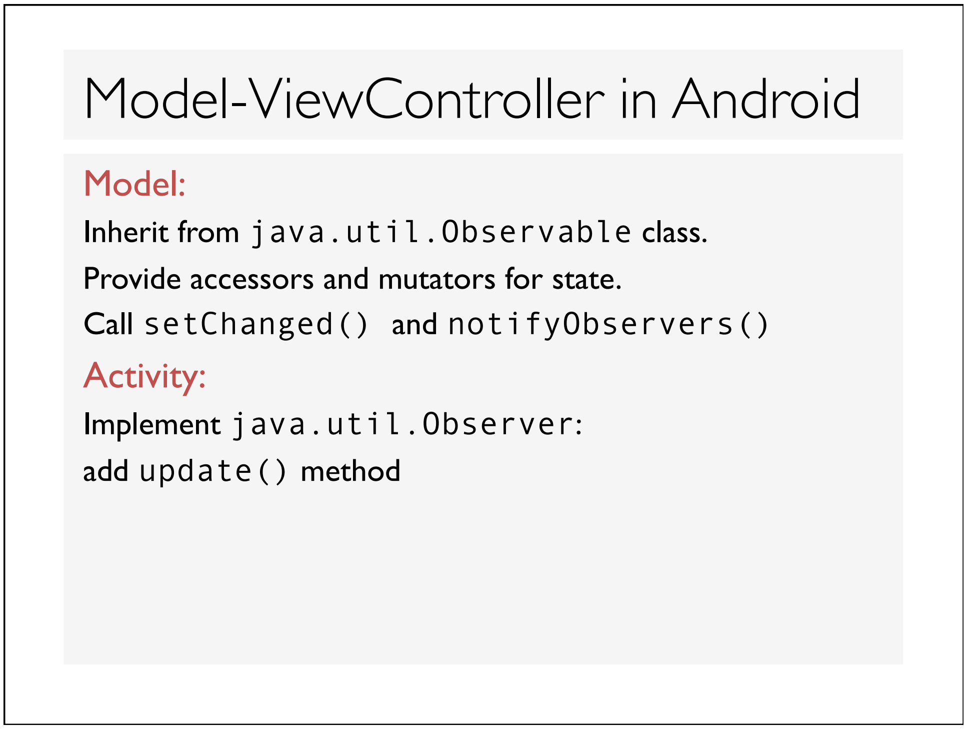




Terminology

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:

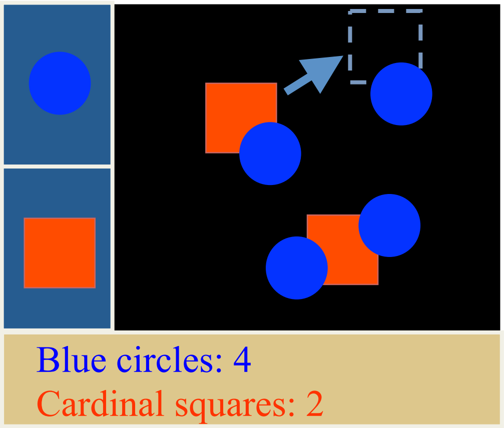
Implement `java.util.Observer`:

add `update()` method

Changing the Display

How do we redraw graphics when a shape moves?

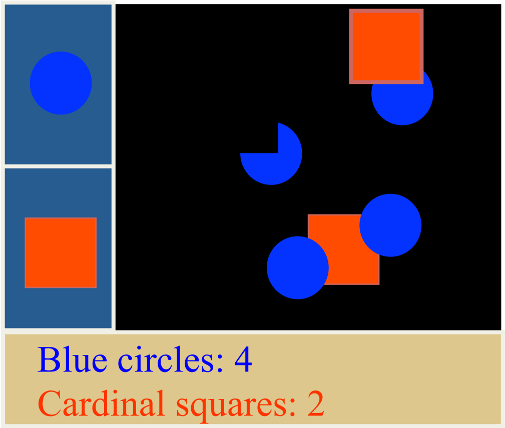
Moving Cardinal Square



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

Damage / Redraw Method

View informs windowing system of areas that are damaged

does not redraw them right away…

Windowing system

batches updates

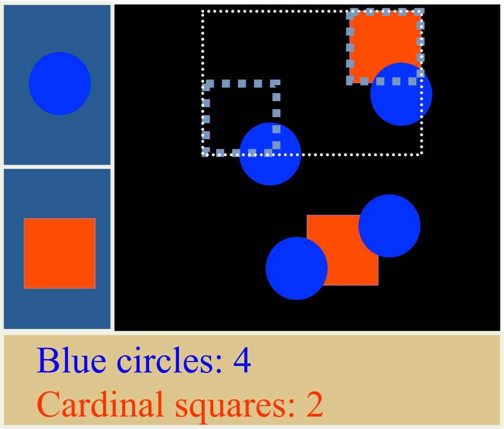
clips them to visible portions of window

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

From the Android Reference:

HOW ANDROID DRAWS VIEWS

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

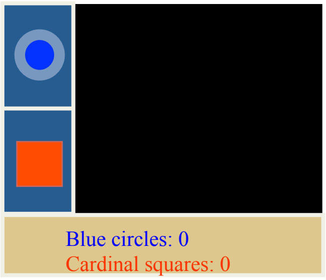
Drawing begins with the root node of the layout. Drawing is handled by walking the tree and rendering each View that intersects the *invalid region*. The framework will not draw Views that are not in the invalid region.[...]

You can force a View to draw, by calling *invalidate()*.

MVC Event Flow

What happens when the user creates a new shape?

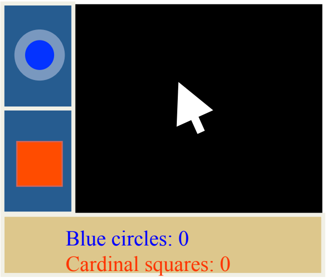
Event Flow (cont.)



Blue circles: 0
Cardinal squares: 0

Assume blue circle selected

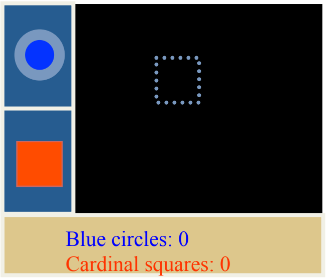
Event Flow (cont.)



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

Event Flow (cont.)



AddCircle() adds new circle to model’s list of objects

Model then notifies list of views of change

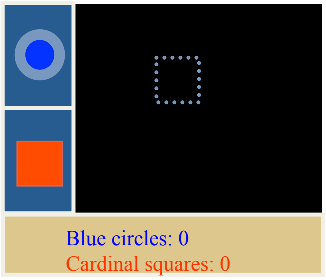
drawing area view and text summary view

Views notifies windowing system of damage

both views notify WS without making changes yet!

model may override

Event Flow (cont.)



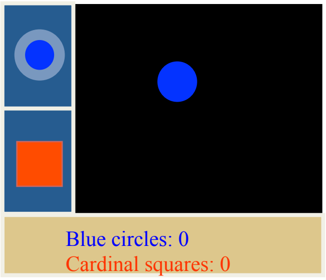
Views return to model, which returns to controller

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 Flow (cont.)



Blue circles: 0
Cardinal squares: 0

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

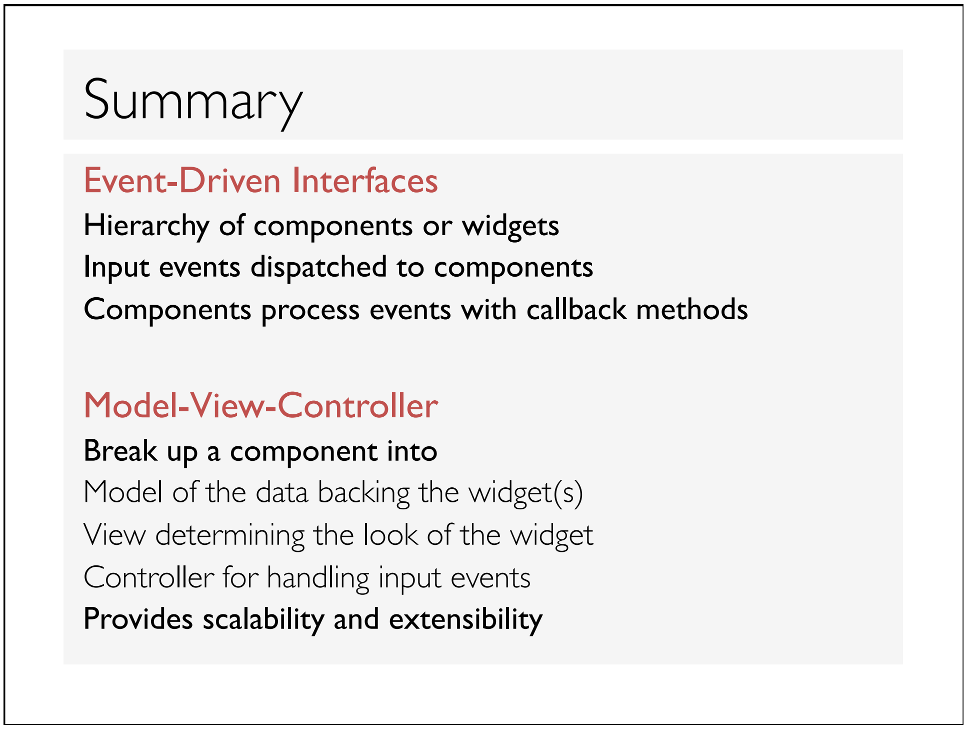
Solutions

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

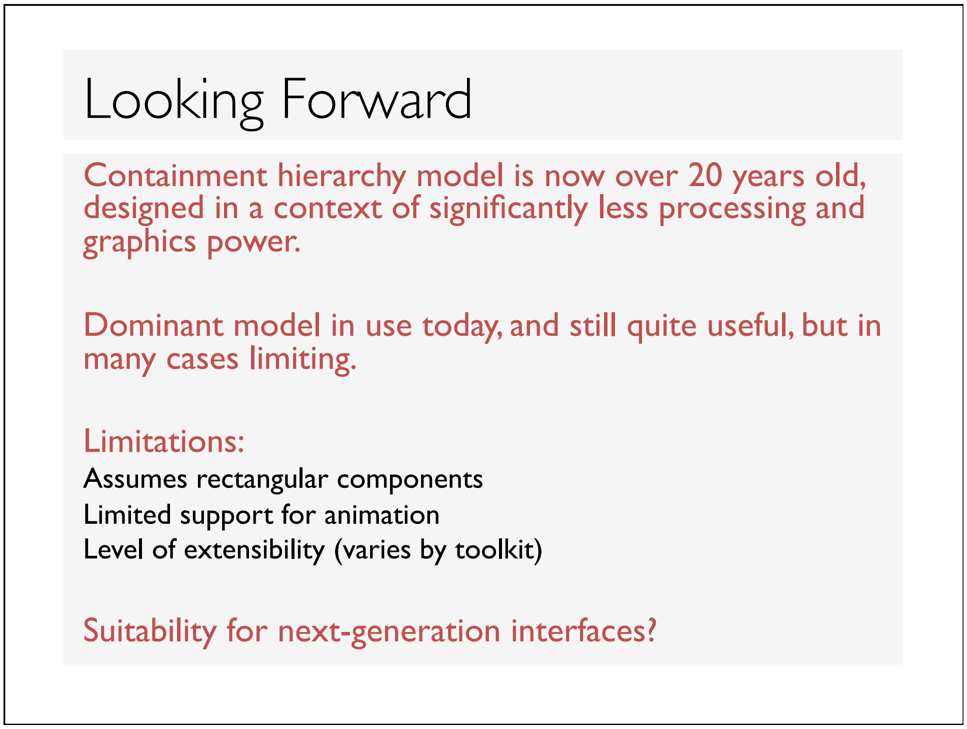
Break up a component into

Model of the data backing the widget(s)

View determining the look of the widget

Controller for handling input events

Provides scalability and extensibility



Looking Forward

Containment hierarchy model is now over 20 years old, designed in a context of significantly less processing and graphics power.

Dominant model in use today, and still quite useful, but in many cases limiting.

Limitations:

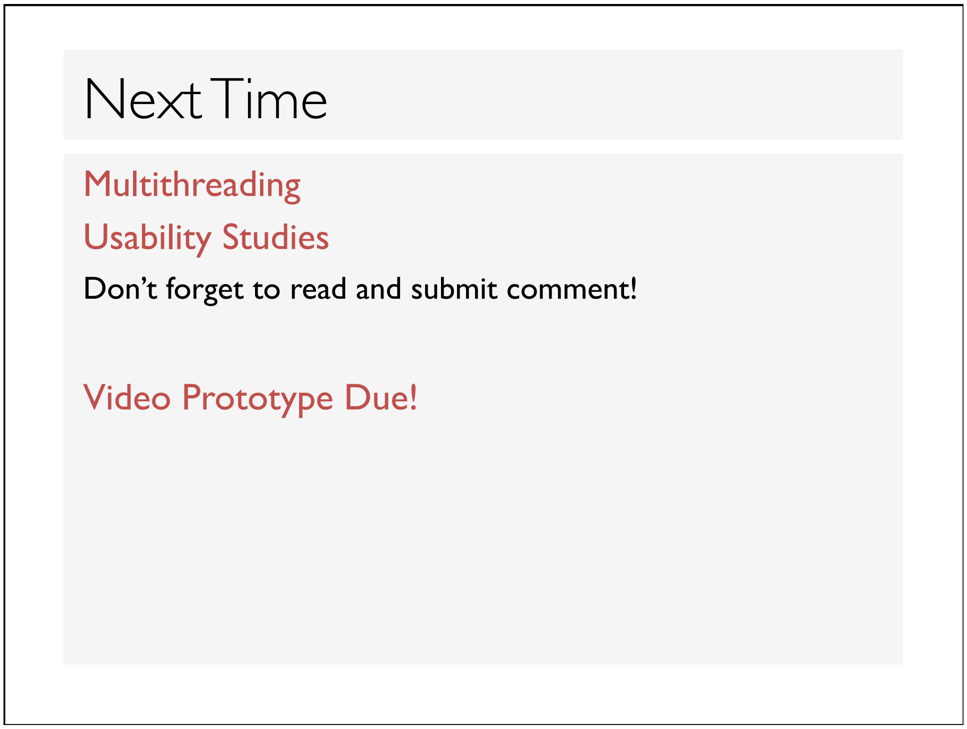
Assumes rectangular components

Limited support for animation

Level of extensibility (varies by toolkit)

Suitability for next-generation interfaces?

Multithreading in GUIs



Next Time

Multithreading

Usability Studies

Don’t forget to read and submit comment!

Video Prototype Due!