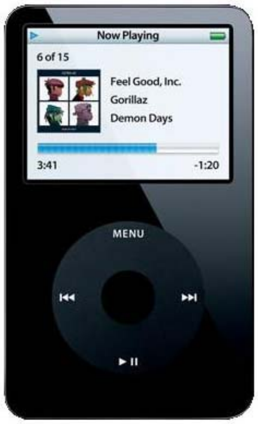
Model View Controller and Event-Driven UI in Flash/Flex

CS160: User Interfaces

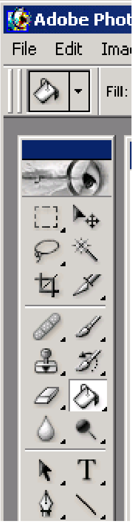
Maneesh Agrawala and Nicholas Kong

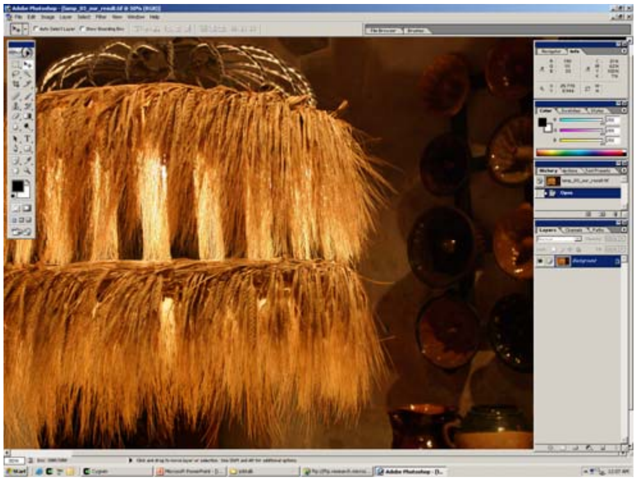
http://www.youtube.com/watch?v=WHxQU4RhyLk



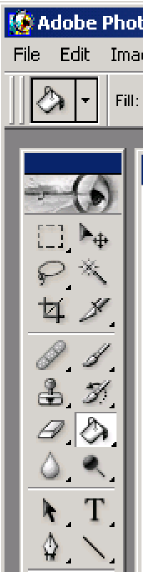
- Most heavily used features directly mapped (volume, play/pause)
- Circular movements mapped to linear operations

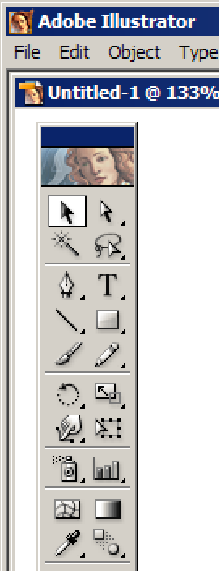
Review: Metaphor





Review: Metaphor





Review: Cognition

Jef Raskin

Cognetics

- Ergonomics of the mind
- Study of "engineering scope of our mental abilities"

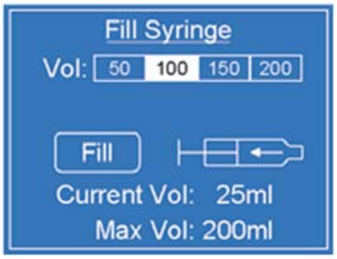
Cognitive Conscious/Unconscious

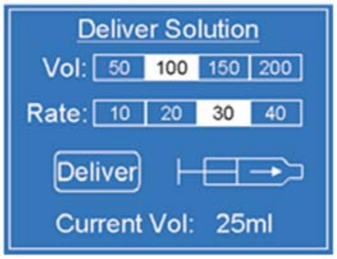
- What is the last letter in your first name?

Locus of Attention

- Idea/object/event which you are intently thinking about
- Focus implies volition; locus not always consciously controlled

Review: Modes





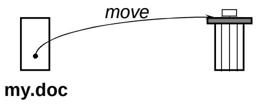
Noun-Verb *VS* Verb-Noun

Noun-Verb: Select object, *then* do action

- Emphasizes 'nouns' (visible objects) rather than 'verbs' (actions)

Advantages

- Closer to real world
- Modeless interaction
- *Actions* always within context of object
  - inappropriate ones can be hidden
- *Generic commands*
  - the same type of action can be performed on the object
  - e.g. drag ‘n drop:



Individual Programming Assignment
(due Mar 2)

Design and Implementation Components

- Sketches of 3 alternatives, pick a favorite
- "Discount" user studies in section (Feb 25-26)
- Write up what you learned from the study
- Note how you changed your interface as a result
- Implement user interface

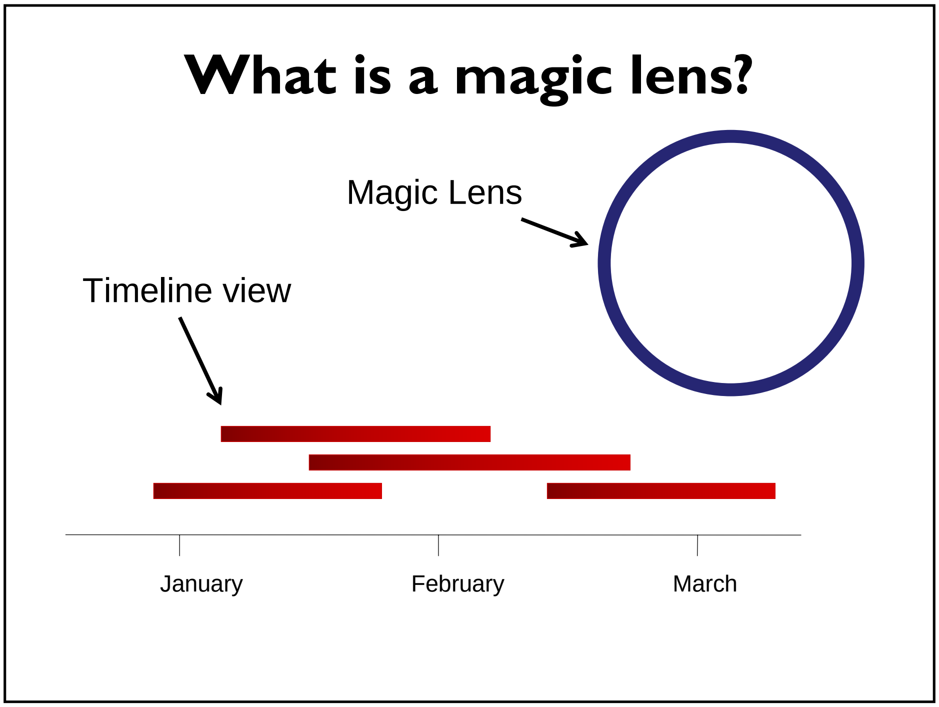
Application area: Project Management/To-Do List

- Items should have start and end date
- Traditional to-do list checklist view
- Timeline view
- Magic lens: http://dohistory.org/diary/exercises/lens/index.html

Individual Programming Assignment
(due Mar 2)

Project Management/To-Do List

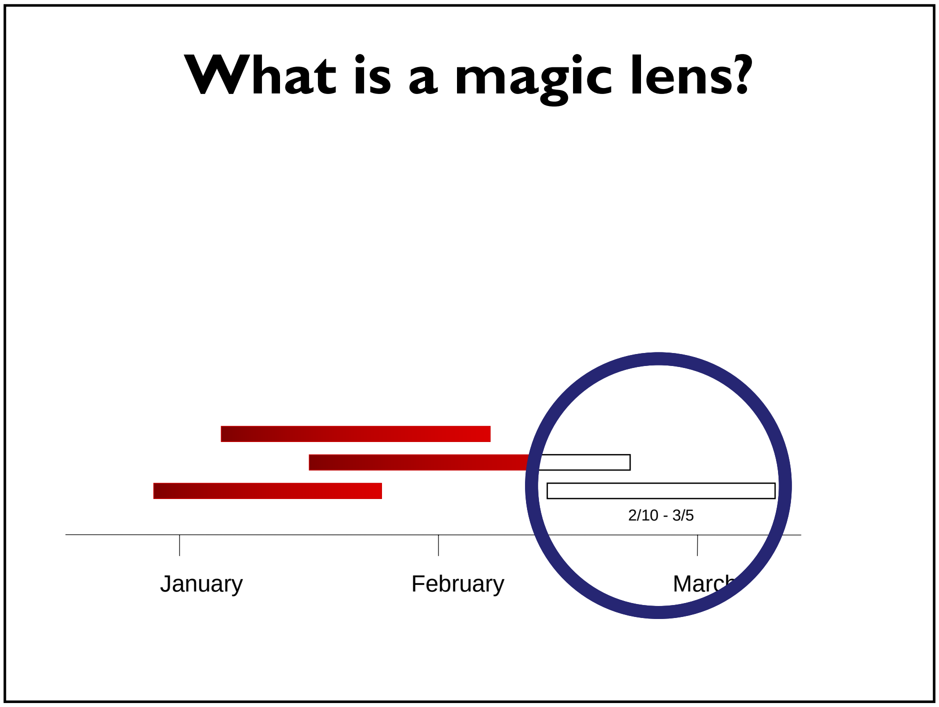
- Tasks have the following properties:
  - Task Name
  - Percentage Completed (0-100%)
  - Start and End date
  - Priority
  - List of people assigned to the task
  - URL related to the task
- Checklist view
  - Include checkbox to automatically set completion percentage to 100%
  - You should be able to see the completion percentage
- Timeline view
- Magic lens: http://dohistory.org/diary/exercises/lens/index.html



What is a magic lens?

Magic Lens

Timeline view



What is a magic lens?

Topics

Interactive application programming

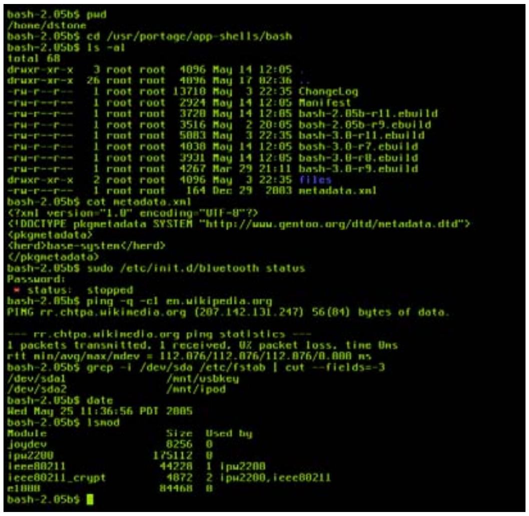
- Component Model
- Event-Driven User Interfaces

Model-View-Controller

- Architecture for interactive components
- Why do we need it?
- Changing the display

Interactive Application Programming

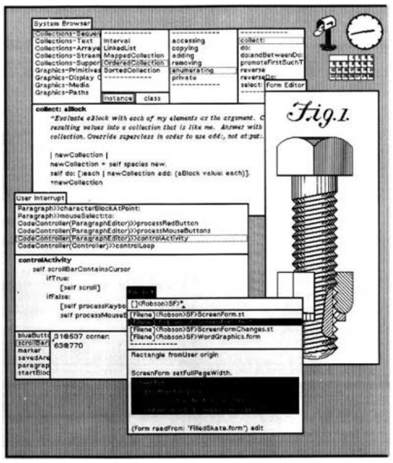
In the beginning…



```
bash-2.05b$ pwd
/home/dstone
bash-2.05b$ cd /usr/portage/app-shells/bash
bash-2.05b$ ls -al
total 68
drwxr-xr-x   3 root root  4096 May 14 12:05 .
drwxr-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  5083 May  3 22:35 bash-3.0-r11.ebuild
-rw-r--r--   1 root root  4038 May 14 12:05 bash-3.0-r7.ebuild
-rw-r--r--   1 root root  3931 May 14 12:05 bash-3.0-r8.ebuild
-rw-r--r--   1 root root  4267 Mar 29 21:11 bash-3.0-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 -c1 en.wikipedia.org
PING rr.chtpa.wikimedia.org (207.142.131.247) 56(84) 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$ lsmod
Module                  Size  Used by
joydev                  8256  0
ipw2200               175112  0
ieee80211              44228  1 ipw2200
ieee80211_crypt         4872  2 ipw2200,ieee80211
e1000                  84468  0
bash-2.05b$ 
```

http://www.cryptonomicon.com/beginning.html

The Xerox Alto (1973)



```
collect: aBlock
    "Evaluate aBlock with each of my elements as the argument. C
    resulting values into a collection that is like me.  Answer with
    collection. Override superclass in order to use add:, not at:put:.

    | newCollection |
    newCollection ← self species new.
    self do: [:each | newCollection add: (aBlock value: each)].
    ↑newCollection
```

```
Paragraph>>characterBlockAtPoint:
Paragraph>>mouseSelect:to:
CodeController(ParagraphEditor)>>processRedButton
CodeController(ParagraphEditor)>>processMouseButtons
CodeController(ParagraphEditor)>>controlActivity
CodeController(Controller)>>controlLoop
```

```
controlActivity
    self scrollBarContainsCursor
        ifTrue:
            [self scroll]
        ifFalse:
            [self processKeybo
            self processMouseB
```

```
31@537 corner:
63@770
```

```
[]<Robson>SF>*
[Filene]<Robson>SF>ScreenForm.st
[Filene]<Robson>SF>ScreenFormChanges.st
[Filene]<Robson>SF>WordGraphics.form
```

```
Rectangle fromUser origin

ScreenForm setFullPageWidth.

(Form readFrom: 'FilledSkate.form') edit
```

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
- More complicated programming and architecture

Widgets

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

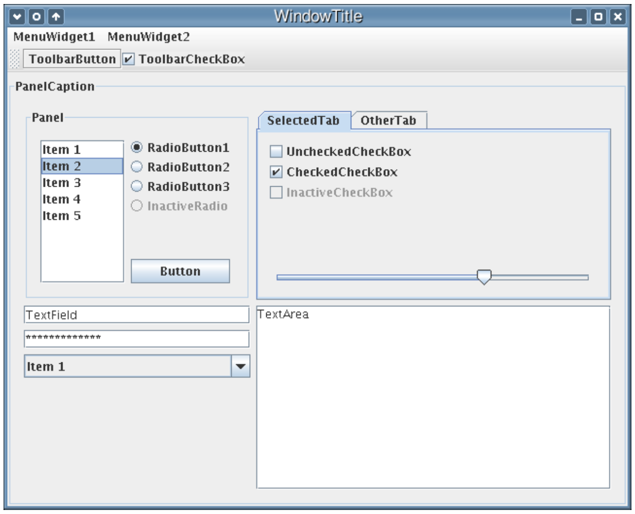
Input event processing and handling

- Typically mouse and keyboard events

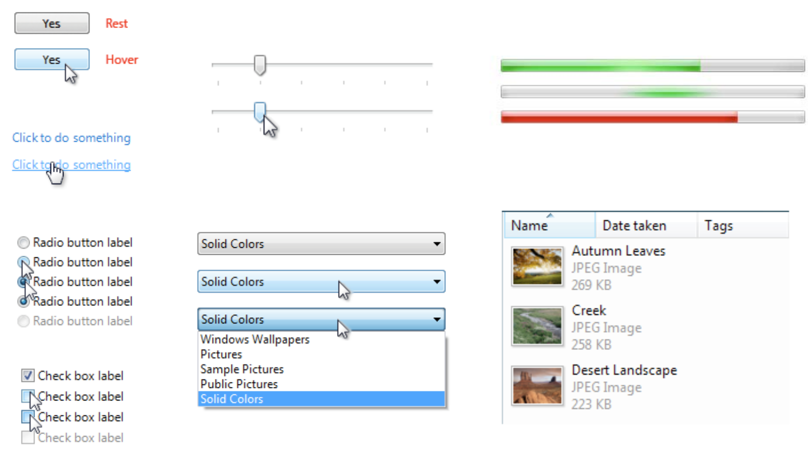
Bounds management (damage/redraw)

- Only redraw areas in need of updating

Java Swing Widgets



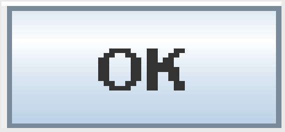
Windows Vista Widgets



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

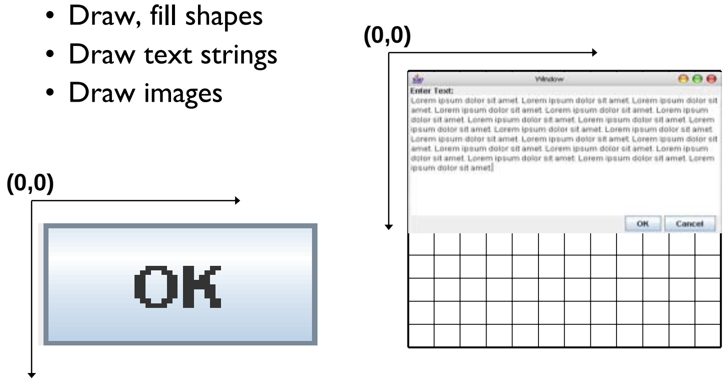


```
•  public void paint(Graphics g) {
•      g.fillRect(…); // interior
•      g.drawString(…); // label
•      g.drawRect(…); // outline
•  }
```

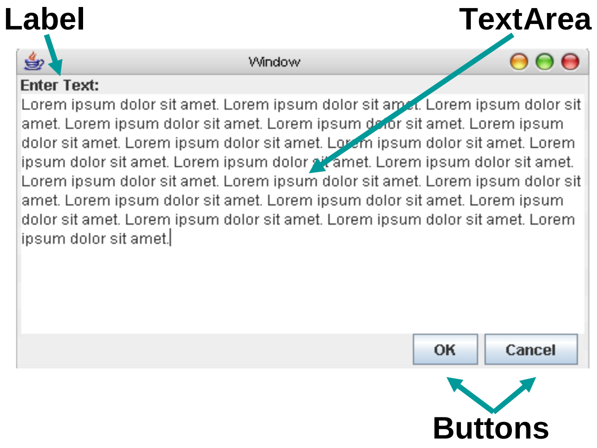
2D Graphics Model

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



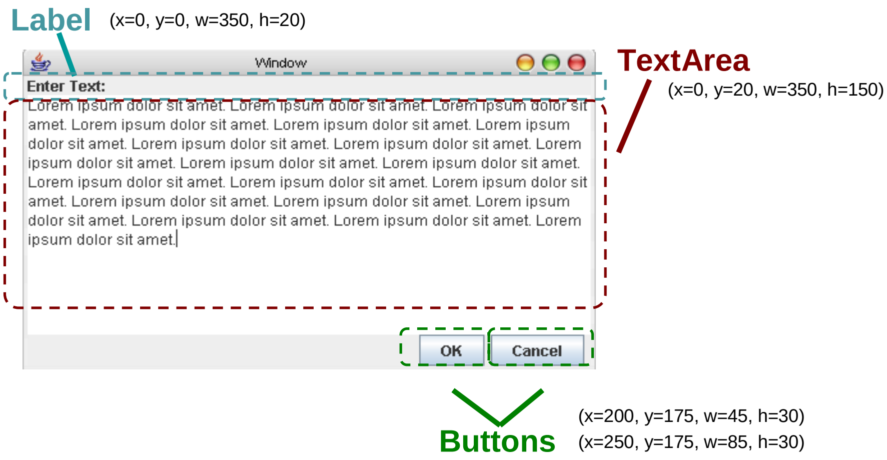
Composing a User Interface



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)

TextArea (x=0, y=20, w=350, h=150)

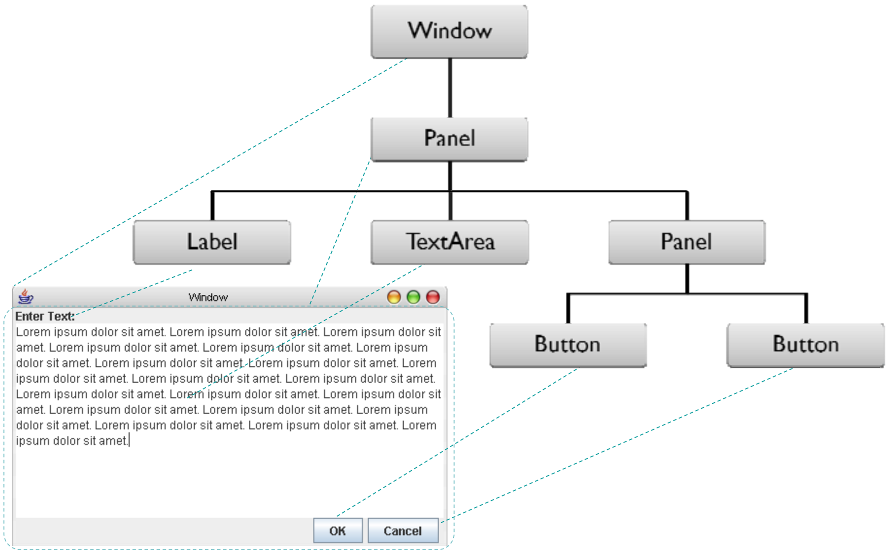
Buttons (x=200, y=175, w=45, h=30) (x=250, y=175, w=85, h=30)

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.

But this is inflexible and doesn't scale or resize well.

Containment Hierarchy



Window

Panel

Label

TextArea

Panel

Button

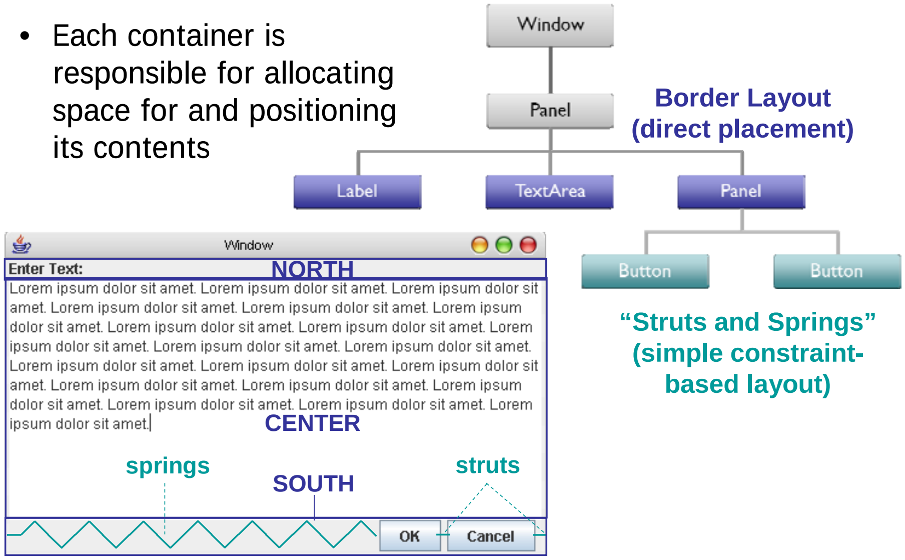
Button

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.

Component Layout

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



Enter Text:

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

Border Layout (direct placement)

“Struts and Springs” (simple constraint-based layout)

Layout in Flash/Flex

What are Flash and Flex?

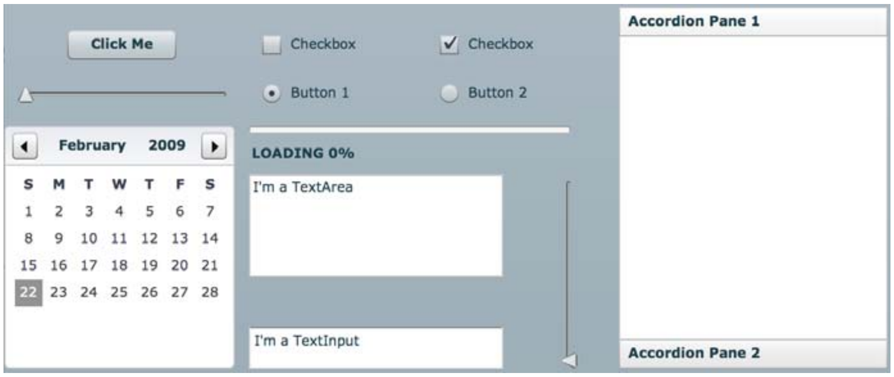
Flex

- Framework for web applications
- Implemented using *MXML* and *ActionScript*
- Contains library of components
- Quickly prototype interfaces in MXML

Flash (actually, ActionScript)

- What Flash Player runs
- JavaScript-like syntax
- Object-oriented, procedural language
- Use to create custom components, event handling

Flex Widgets



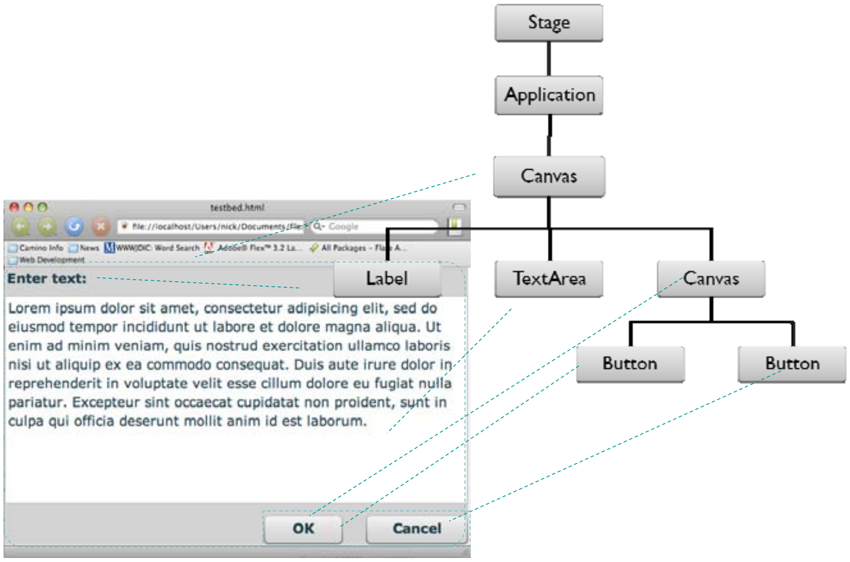
February 2009

S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28

I'm a TextArea

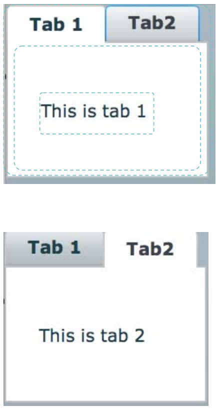
I'm a TextInput

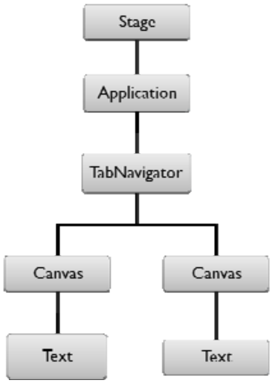
Component Layout in Flex



Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

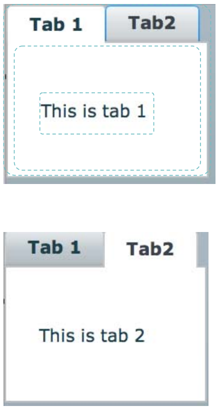
One Flex Layout





TabNavigator

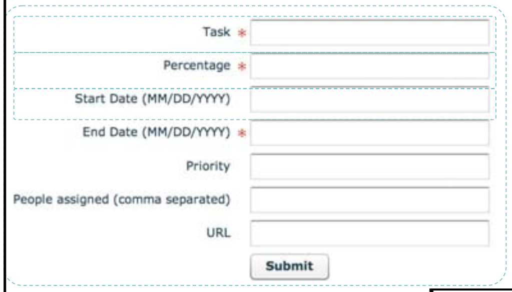
One Flex Layout (XML)



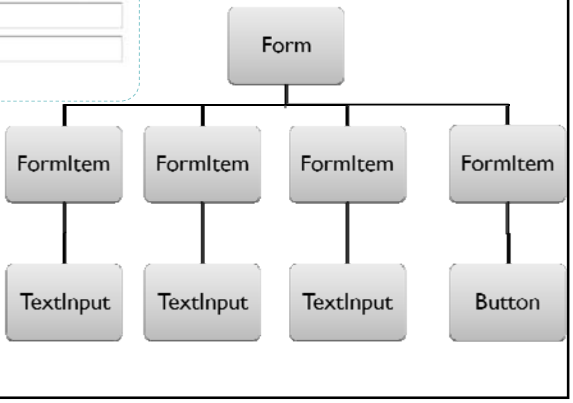
TabNavigator

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application
xmlns:mx="http://www.adobe.com/2006/mxml"
layout="absolute">

      <mx:Canvas>
            <mx:TabNavigator width="117" height="100">
                  <mx:Canvas label="Tab 1" width="100%"
                  height="100%">
                        <mx:Text x="17" y="21" width="81"
                  height="25" text="This is tab 1"/>
                  </mx:Canvas>
                  <mx:Canvas label="Tab2" width="100%"
                  height="100%">
                        <mx:Text x="17" y="21" width="81"
                  height="25" text="This is tab 2"/>
                  </mx:Canvas>
            </mx:TabNavigator>
      </mx:Canvas>

</mx:Application>
```

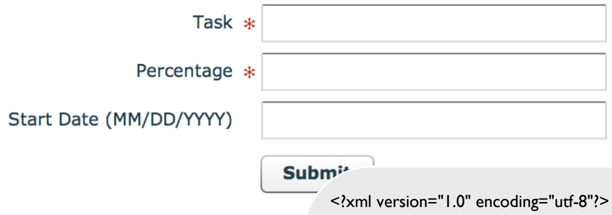
Another Flex Layout



Form



Flex Layout XML



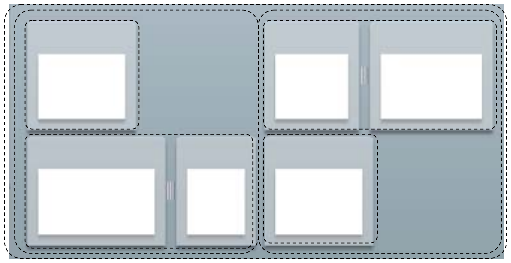
Task *

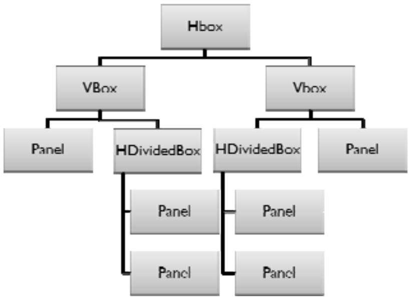
Percentage *

Start Date (MM/DD/YYYY)

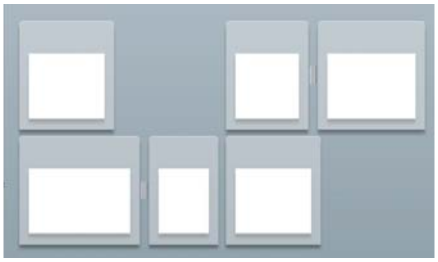
```
<?xml version="1.0" encoding="utf-8"?>
<mx:Form xmlns:mx="http://www.adobe.com/2006/mxml"
initialize="{init();}">
    <mx:FormItem label="Task" required="true">
        <mx:TextInput id="taskname" width="200"/>
    </mx:FormItem>
    <mx:FormItem label="Percentage" required="true">
        <mx:TextInput id="percentage" width="200"/>
    </mx:FormItem>
    <mx:FormItem label="Start Date (MM/DD/YYYY)" required="false">
        <mx:TextInput id="startDate" width="200"/>
    </mx:FormItem>

      <mx:FormItem>
      <!-- User clicks Button to trigger validation. -->
      <mx:Button id="submit" label="Submit" click="{addTask();}”/>
      </mx:FormItem>
</mx:Form>
```

Roll your own...



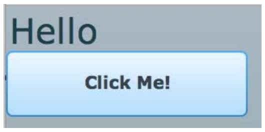


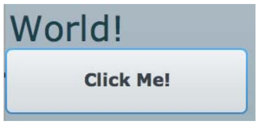
Roll your own…



```
<mx:HBox x="10" y="10" width="100%" scaleX="1.5"
scaleY="1.5">
    <mx:VBox height="100%">
        <mx:Panel width="100" height="100">
        </mx:Panel>
        <mx:HDividedBox width="100%">
            <mx:Panel width="100" height="100">
            </mx:Panel>
            <mx:Panel width="100" height="100">
            </mx:Panel>
        </mx:HDividedBox>
    </mx:VBox>
    <mx:VBox height="100%">
        <mx:HDividedBox width="100%">
            <mx:Panel width="100" height="100">
            </mx:Panel>
            <mx:Panel width="100" height="100">
            </mx:Panel>
        </mx:HDividedBox>
        <mx:Panel width="100" height="100">
        </mx:Panel>
    </mx:VBox>
</mx:HBox>
```

Flex Event Handling





- Every component (i.e., objects that extend *UIComponent*) dispatch events corresponding to different interactions.
- Classes that extend EventDispatcher can dispatch and listen to events, pre- or user-defined

- Examples events include:
- MouseEvent.MOUSE_MOVE, .CLICK
- KeyboardEvent.KEY_DOWN
- FlexEvent.BUTTON_DOWN

Flex Event Handling

Three phases: *Capturing, Targeting, Bubbling*

Capturing

Flash Player traverses the display list from root to the target's parent for event listeners.

Targeting

The event listener is called on the target.

Bubbling (certain events)

Flash Player traverses the display list from target to root.

Flex Event Handling

There are a few ways to specify event handlers in Flex. The code below shows inline specification in MXML. You can also use the addEventListener() function in ActionScript.

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute">

    <mx:Canvas scaleX="2" scaleY="2">

        <mx:Label x="0" y="0" id="lab" width="140" height="28" fontSize="20"/>
        <mx:Button x="0" y="26" label="Click Me!"
        buttonDown="lab.text='Hello';" click="lab.text='World!';"
        width="140" height="38"/>

    </mx:Canvas>

</mx:Application>
```

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

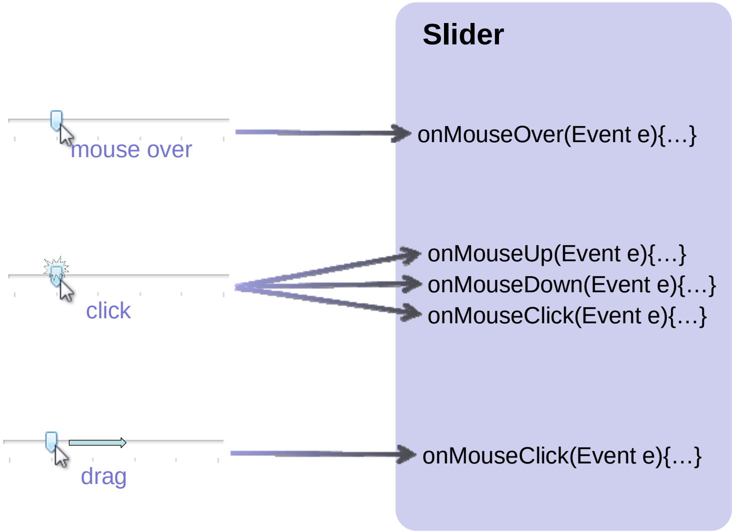
- Event Type (mouse moved, key down, etc)
- Event Source (the input component)
- Timestamp (when did event occur)
- Modifiers (Ctrl, Shift, Alt, etc)
- Event Content
  - Mouse: x,y coordinates, button pressed, # clicks
  - Keyboard: which key was pressed

Abstracting Events

Level of abstraction may vary. Consider:

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

Callbacks



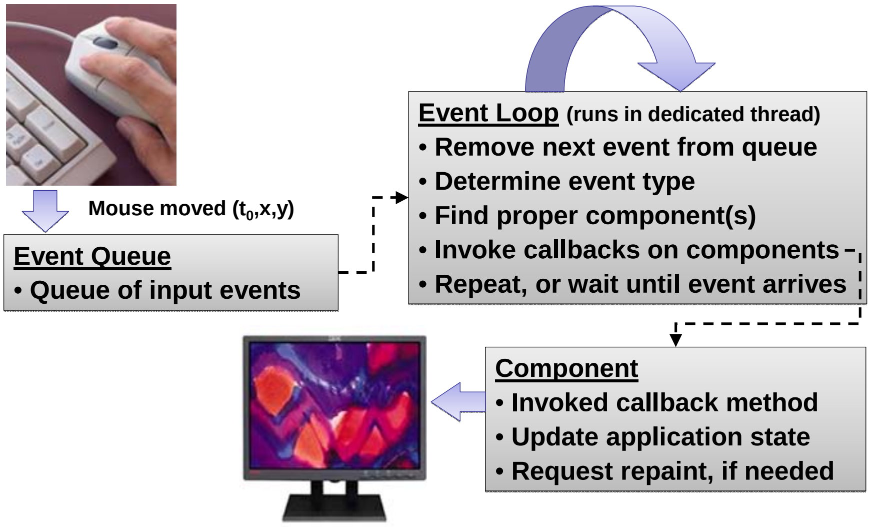
Slider

mouse over → onMouseOver(Event e){…}

click → onMouseUp(Event e){…}
onMouseDown(Event e){…}
onMouseClick(Event e){…}

drag → onMouseClick(Event e){…}

Event Dispatch Loop

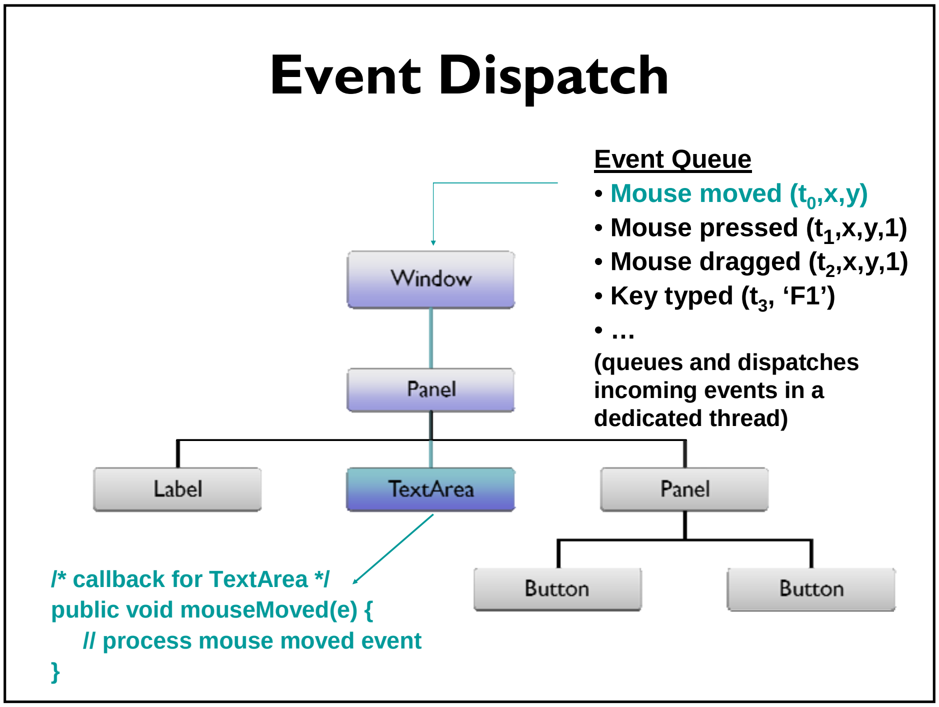


Mouse moved (t_0,x,y)

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



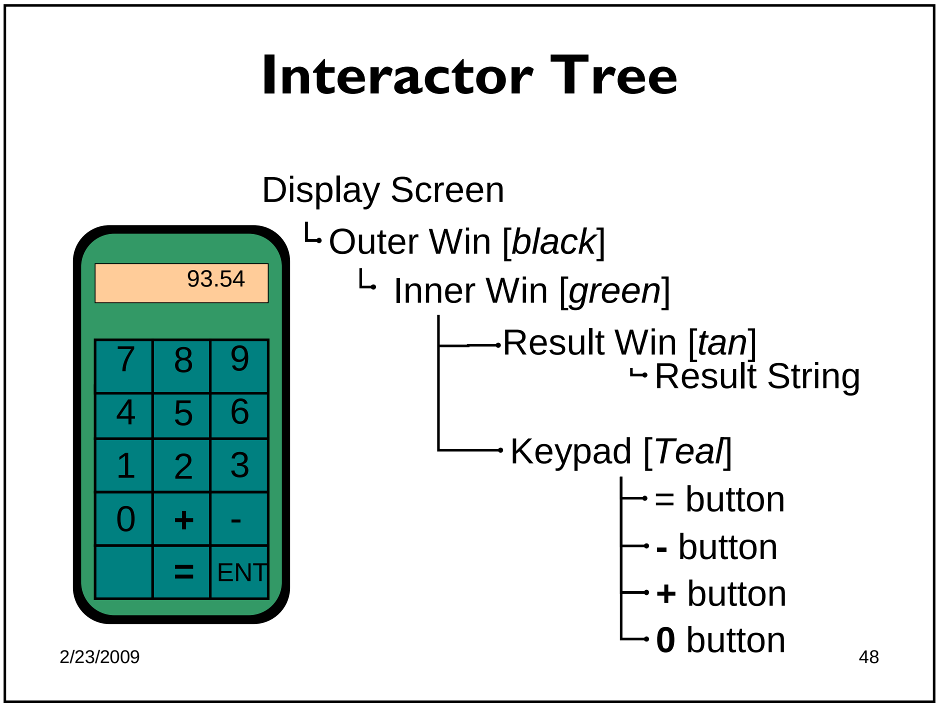
Event Dispatch

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

93.54		
7	8	9
4	5	6
1	2	3
0	**+**	-
	=	ENT

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

2/23/2009

Demo

- Walk through example code for layouts we saw earlier and the sample code for the first assignment
- Explore ActionScript's event handling model