iPhone Programming
CMSC 498I – Fall 2010

Views and Controls
Adam Porter
Creating A View – Code

• Example

CGRect frame = CGRectMake(0, 0, 100, 100);
PersonView *personView = [[PersonView alloc] initWithFrame:frame];

[containerView addSubview:personView];
[personView release];

• Views are retained by the view hierarchy
Creating A View – IB

- Drag out an existing view
- Or drag out a generic UIView, and set its class
Creating A View – IB

- Drag out an existing view
- Or drag out a generic UIView, and set its class
Creating A View – IB

- Drag out an existing view
- Or drag out a generic `UIView`, and set its class
Creating A View – IB

- Drag out an existing view
- Or drag out a generic UIView, and set its class

![Image of Interface Builder interface showing creation of a view with PersonView class](Image)
Connecting To A View

- If you have an outlet

```c
IBOutlet UISlider *mySlider;
```

- You can connect to it in IB by dragging

- During unarchiving IB will directly set instance variable
Connections

- If you define the outlet to be a property

```objective-c
@property (...) IBOutlet id anOutlet;
```

- Then setter’s implementation is used when setting the instance variable

```objective-c
-(void)setAnOutlet:(id)anOutlet {
    ....
}
```
### View Tags

- Views can be given an arbitrary integer “tag”
  - Another way to access views
- Set in IB or in code

```objective-c
#define VOLUME_SLIDER_TAG 10
[slider setTag: VOLUME_SLIDER_TAG];
```

- Recursively search a view using `viewWithTag:`

```objective-c
UIView *someView = ...;
UISlider *slider = [someView viewWithTag:VOLUME_SLIDER_TAG];
if (slider) {
    // found the slider, do something with it...
}
```
Subclassing UIView

Basics
Drawing

- Two basic ways to use UIViews
  - Drawing – subclass UIView and write custom drawing code
  - Composition – create and layout multiple existing views
Custom Views – Drawing

• Custom drawing code goes inside the `drawRect:` method

```c
- (void)drawRect:(CGRect)rect;
```

• Don’t call `drawRect:` directly

• Schedule `drawRect:` by calling `setNeedsDisplay`

```c
[view setNeedsDisplay]
```
Custom Views – Composition

• Use subviews as a display mechanism

• Create your subviews. Put composition code in -layoutSubviews

```
- (void)layoutSubviews;
```

• Schedule -layoutSubviews by calling -setNeedsLayout

```
[view setNeedsLayout]
```
To handle events, override

- `(void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event;
- `(void)touchesMoved:(NSSet *)touches withEvent:(UIEvent *)event;
- `(void)touchesEnded:(NSSet *)touches withEvent:(UIEvent *)event;

Tracking multi-touch events

```swift
view.multipleTouchEnabled = YES; // defaults to NO...
```

Ignoring input

```swift
view.userInteractionDisabled = YES;
```

Temporarily hiding

```swift
view.isHidden = YES;
```
UIResponder
UIResponder

- Superclass of UIApplication, UIView, etc.
- Defines an interface for responding to / handling events & messages
  - Touch event handling (e.g., tap)
  - Motion gestures - (e.g. shake)
  - Remote-control
  - Standard edit actions (e.g. copy/paste)
Responder Chain

- Responders linked in “Responder Chain”
  - Events travel up chain until handled or the chain ends
- Chain starts at
  - Hit-test view, the last touched view, for touch events
  - First responder for other events (e.g., motion, remote-control, action messages, editing-menu messages)
- If un-handled, passes to nextResponder
Determining The NextResponder

- Initial responder view passes the event / message to its view controller; if it has none, passes the event / message to its superview.

- If receiver can’t handle it, pass it to the view’s superview.

- Repeat until it arrives at the topmost view in the view hierarchy.
  - If still can’t be handled, the event / message passes to the window object.

- The window object can’t handle the event / message, passes it to the application object.

- If still not handled, it is dropped.
Responder Chain
First Responder

• To make your view become first responder

```swift
- (BOOL)canBecomeFirstResponder {
    return YES;
}

[myView becomeFirstResponder]
```

• To resign first responder status

```swift
[myView resignFirstResponder]
```
First Responder

- **nil** targeted actions
  - Action delivered to the first responder
  - Allows you to have actions that don’t hardcode a target
    - e.g., cut / copy / paste

- Text views automatically made first responder when tapped
  - UIKit shows / hides keyboard when the first responder knows how to deal with text (we’ll discuss what this means later)
Example

- MyViewController has copyable data (e.g. an image)
- Assume ViewA is the “first responder”
Example

“send -copy: message to *nil* target”
Example

“send -copy: message to *nil* target”
Example

“send -copy: message to *nil* target”
Example

“send -copy: message to *nil* target”
Example

“send -copy: message to *nil* target”
Example

“send -copy: message to *nil* target”

NextResponder

MyViewController

-(void)copy:(id)sender;

MyView

View A

First Responder
UIKitControls
UIKitControl

- Defines interface for UIViews with target/action behavior
  - Examples: button, slider, switch, date picker, etc…

- Features
  - Send actions when control events occur
  - Track touch events

- Properties
  - Control state – disabled, highlighted, selected
  - Tracking state – tracking, touchInside
Control Actions

• Maintains dispatch table of \{control event, target/action\}
  ▪ UIControlEvents – when to send actions
  ▪ Target / Action – what to send and where to send it

• UIControls can have multiple targets / actions

• Example

```swift
// UIButton is a subclass of UIControl
UIControl *button = [[UIButton alloc] initWithFrame: .. ];

[button addTarget:self action:@selector(buttonClicked:) forControlEvents:UIControlEventTouchUpInside];
[button addTarget:self action:@selector(logEvent:) forControlEvents:UIControlEventTouchUpInside];
```
Control Events

- Touch events - used by UIButtonns

```plaintext
UIControlEventTouchDown
UIControlEventTouchDragInside
UIControlEventTouchDragOutside
UIControlEventTouchUpInside
UIControlEventTouchOutside
UIControlEventTouchUpInside
UIControlEventTouchUpOutside
UIControlEventTouchCancel
```

- Value change events – used by UISlider

```plaintext
UIControlEventValueChanged
```

- Editing state events - used by UITextField

```plaintext
UIControlEventEditingDidBegin
UIControlEventEditingChanged
UIControlEventEditingDidEnd
```
Enabled State

• When isEnabled is NO
  ▪ UIControl will ignore touch events
  ▪ May draw differently

• UIViews can ignore touch events too
  ▪ Use view.userInteractionEnabled = NO to make a view ignore touch events
    ▪ Can be useful during animations for example
  ▪ May choose to ignore events so that other views receive them
UIButton

- Display Properties
  - Title, Font, Shadow
  - Image, Background Image

- Display Properties settable by state
  - Normal, Highlighted, Disabled, ...

- Use standard buttons, or make your own!
UIButton

• Properties settable by state

```c
// UIControl.h
enum {
    UIControlStateNormal,    // normal, not pressed or disabled, etc.
    UIControlStateHighlighted, // button’s pressed state
    UIControlStateDisabled
};
typedef NSUInteger UIControlState;
```

```c
// Setting Properties, By State
- (void)setTitle:(NSString *)title forState:(UIControlState)s;
- (void)setTitleColor:(UIColor *)color forState:(UIControlState)s;
- (void)setTitleShadowColor:(UIColor *)color forState:(UIControlState)s;
- (void)setImage:(UIImage *)image forState:(UIControlState)s;
- (void)setBackgroundImage:(UIImage *)image forState:(UIControlState)s;
```

• **UIControlStateNormal** is the fallback when a property is not specified for a state

• Each “setter” has matching getter
Button Types
Button Types

Custom Button
Button Types

Rounded Rect
Button Types

Add Button
Button Types

Disclosure
Button Types
**UITextField**

- Displays editable text
  - **Note:** **UITextView** – scrollable, multi-line, for “large” text display
- Sends target / action messages when editing ends
- Features
  - Text Attributes – text, placeholder
  - Text Attributes – font, color, alignment, autosizing (shrink to fit)
  - Appearance – border style, background
  - Drawing structure for subclasses
Keyboard

- Shown when text field becomes **first responder**

```objective-c
[textField becomeFirstResponder]
```

- Dismiss by sending `-resignFirstResponder` to text field

- Configure - `UITextField` conforms to `UITextInputTraits` protocol
  - Keyboard type - default, number pad, email addresses, ...
  - Auto correction on/off
  - Capitalization style

- More on keyboard, text input in weeks to come
Reading

- UIView, UIControl – become familiar with each
  - iPhone Application Programming Guide
    - Chapter 2, “Windows and Views” - p.47 - 75
  - UIControl Class Reference
    - Read - p.5, 11-12, 17-22

- Browse headers, docs for controls we discussed
  - UIKit Framework Reference
Other Controls

Quick Tour... for home reading
Other Controls

- Quick overview
- Some views we’ll cover in depth later
- Others are “exercise to the reader”
Other Controls

• View “Switcher” Affordance
  - UITabBar
  - UIPageControl
  - UISegmentedControl
Other Controls

- View “Switcher” Affordance
  - UITabBar
  - UIPageControl
  - UISegmentedControl

- UISwitch
  - On or off state
Other Controls

• View “Switcher” Affordance
  - UITabBar
  - UIPageControl
  - UISegmentedControl

• UISwitch
  - On or off state

• UISearchBar
  - UI for text searching
  - delegate does the search
Other Controls

- **UISlider**
  - Pick value from a range
  - Continuous or at end of tracking
Other Controls

- **UISlider**
  - Pick value from a range
  - Continuous or at end of tracking

- **UIProgressView**
  - Depict progress over time
Other Controls

• **UISlider**
  - Pick value from a range
  - Continuous or at end of tracking

• **UIProgressView**
  - Depict progress over time

• **UIActivityIndicator**
  - Indicate activity or indeterminate progress
Data Views & Controls

- UIPickerView
  - Spinning wheel for picking values
  - You provide rows as NSString, or UIView
Data Views & Controls

• UIPickerView
  - Spinning wheel for picking values
  - You provide rows as NSString, or UIView

• UIDatePicker
  - Picker designed for picking dates
  - Can control displayed components
Data Views & Controls

- **UITextView**
  - editable, scrollable, multiline text
  - custom display attributes for entire content
Data Views & Controls

• UITextView
  ▪ editable, scrollable, multiline text
  ▪ custom display attributes for entire content

• UIWebView
  ▪ Safari in a view!
  ▪ More complex attributes vs. UITextView
Data Views & Controls

- **UITextView**
  - editable, scrollable, multiline text
  - custom display attributes for entire content

- **UIWebView**
  - Safari in a view!
  - More complex attributes vs. **UITextView**

- **MKMapView**
  - Maps in a view! **MapKit.framework**
Data Views & Controls

• **UITextView**
  - editable, scrollable, multiline text
  - custom display attributes for entire content

• **UIWebView**
  - Safari in a view!
  - More complex attributes vs. **UITextView**

• **MKMapView**
  - Maps in a view! MapKit.framework

** Each will be covered in detail later in the semester **