Interface Builder & iPhone UI Programming

Lecture 3
The Model

• This is your data model; maintains state
• Has no idea about UI or how the data is presented
• Example: a Fraction class for a fraction calculator app

Paraphrased from the Stanford Spring 2009 lecture slides
The View

• The part the user sees; the UI
• Displays the model
• Does not store data (except maybe caching its state)

Paraphrased from the Stanford Spring 2009 lecture slides
The Controller

- The middleman between the model and view
- Updates the view when the model changes
- Tells the model to update when the user manipulates the view
Demo: Making a UI
Outlets & Actions

• How do we connect our code to the UI (and vice-versa)?

• **Outlets**: create references in code to UI controls

• **Actions**: allow UI controls to send messages to objects

• Control-drag to attach
Nib Loading

• A nib is *not* generated code.
• Rather, it is a collection of *serialized* objects
• Connects outlets & actions automatically
  • Then calls - (void) awakeFromNib on each object
Outlets

• `IBOutlet UIButton *button;`
• `IBOutlet` is completely ignored by the compiler
  • It’s simply an indicator for Interface Builder
• Caution! Outlets may not be loaded (connected) until `- (void) awakeFromNib`
Actions

- (IBAction) buttonClicked: (UIButton *) sender;

- **IBAction** is defined as **void**

- On Mac, the `sender` argument is required

- On iOS, `sender` is optional
Target-Action

- Actions can be set manually on controls
- On Mac, controls have one target-action
  - (void) setTarget: (id) target;
  - (void) setAction: (SEL) action;
- On iOS, controls can have many target-actions
  - (void) addTarget: (id) target action: (SEL) action forControlEvents: (UIControlEvents) ctrlEvents;
MVC Example

Controller

- (IBAction)emailClicked:(id)sender

Model (Person)
- (UIImage *)photo;
- (NSString *)name;
- (NSString *)email;

View (Address Card)
Steve Jobs
steve@mac.com
Demo: Slider
NS(UI)Application

- Manages the main run loop of your application:
  - Waits for events from the mouse, keyboard, touches (iOS)
  - Dispatches events to the relevant objects (of class NS(UI)Responder)
- Owner of MainMenu.xib (Mac)/MainWindow.xib (iOS)
Application Life Cycle

• Application starts
• MainMenu.xib/MainWindow.xib loads
• Run loop:
  • Wait for event
  • Handle event
• Application terminates
NSView/UIView

- Superclass for all views in Cocoa/Cocoa Touch
- Position and size in `[view frame]`
- `[view addSubview:]` and friends used for programmatic layout
- Often subclassed; we’ll talk about making your own views later
NSWindow

• Manages window frame

• Views in window are subviews of [window contentView]
The View Hierarchy

NSWindow

NSView

[window contentView]

[contentView subviews]

[box subviews]
NS(UI)Control

- Subclass of NS(UI)View
- Concept of value: [control intValue], [control floatValue], stringValue, and so on (Mac only)
- Superclass for text fields, buttons, etc.
Delegation

- Common pattern in Cocoa; avoids need to subclass in simple cases
- Objects notify their delegates when something happens through delegate methods
- Application delegate: start, terminate
- Window delegate: resize, close
Application Life Cycle

[delegate applicationDidFinishLaunching:notification];

- Application starts
- MainMenu.nib loads
- Event loop
- Application terminates
Application Life Cycle

(delegate applicationWillTerminate:notification);

- Application starts
- MainMenu.nib loads
- Event loop
- Application terminates