iPhone Programming
CMSC 498I – Fall 2010

UIKit and Views
Adam Porter
Today

- iPhone Application Architecture
  - Life Cycle
  - Resources
- Introduction to UIKit & UIViews
Applications

Life Cycle and Architecture
Application Ingredients

• Components
  ▪ Resources
  ▪ Frameworks
  ▪ App code

• Connectors
  ▪ Outlets
  ▪ Events

• Configuration
  ▪ System & user preferences
Resources

- Media
- Strings (localized)
- User Interface (XIBs)
Frameworks

• System libraries
  ▪ Foundation
  ▪ UIKit
  ▪ CoreLocation

• Third party
  ▪ JSON
  ▪ Cocos2d
  ▪ gdata-objectivec-client
Application Code

• Often follows Model-View-Controller (MVC) pattern
  ▪ Model Code - app specific data
  ▪ View Code - display code
  ▪ Controller Code - app logic that mediates between model and view
Code Connectors

- UI outlets
  - Set up in code, or using Interface Builder tool

- Events
  - Notifications, touch events, delegate callbacks
Configuration

• App Configuration – ‘Info.plist’
  • Property list that tells system about your application
    • E.g. Supported orientations, networking requirements, iPhone / iPod Touch compatibility, Icon, ...
Application Flow
User Launches App

main()

UIApplicationMain()

Main ‘xib’ file loaded
Done “Launching”

Event Loop

User Quits App

App Termination

System tells app to terminate
App Termination

UIApplicationMain( )

-applicationDidEnterBackground:

-applicationWillTerminate:

Your "Entry Points"

Actions
Notifications
Timers
Layout / Drawing

Sunday, September 19, 2010
UI Kit

main()

UIApplicationMain()

Main ‘xib’ file loaded

Done “Launching”

UI Kit

Your ‘xib’ file loaded

(applicationDidFinishLaunching:

Your “Entry Points”

Actions
Notifications
Timers
Layout / Drawing

You

System tells app to
Execution terminates

(applicationWillTerminate:

Sunday, September 19, 2010
UIApplicationMain( )

Main ‘xib’ file loaded
Done “Launching”

Your ‘xib’ file loaded
-aplicationDidFinishLaunching:

Your “Entry Points”

Actions
Notifications
Timers
Layout / Drawing

System tells app to
Execution terminates

-aplicationWillTerminate:
UIApplicationMain( )

Main ‘xib’ file loaded

Done “Launching”

You

Your ‘xib’ file loaded

-applicationDidFinishLaunching:

Your “Entry Points”

Actions

Notifications

Timers

Layout / Drawing

UIKit

main( )

UIApplicationMain( )

app will be loaded

Done “Launching”

System tells app to

Execution terminates

You

(applicationWillTerminate:

You
Nib Loading

• Application loads main Nib file
  ▪ NSMainNibFile key in Info.plist

• Nib file is “owned” by an object
  ▪ Often a UIViewController
    ▪ Owner can’t be defined in the Nib file

• Owner is the link between objects in the nib file and objects outside of it
File’s Owner

- In IB, the File’s Owner is a proxy for the owner object
NIB Loading Process

1. Load contents of Nib file

2. Unarchive Nib objects & initialize them
   - Standard objects receive `-initWithCoder:`
   - Custom objects receive `-init:`

3. Establish outlets & actions

4. Objects created during NIB loading receive
   - `-awakeFromNib:`
init*: & awakeFromNib: Methods

- IBOутlets are nil during your init methods
  - So -init: methods shouldn’t access IBOутlets

- IBOутlets are guaranteed to be set before –awakeFromNib: sent
Initializing Other Objects

- Some objects, such as the File’s Owner are not instantiated by NIB Loading

- IBOutlets will be valid after `loadNibNamed:owner:options: returns`

- For example, for UIViewController access IBOutlets in `-viewDidLoad`
Notifications

• Notifications: Mechanism for broadcasting messages
  ▪ System notifications, or your own

• Example of Observer pattern

• Observing

```c
NSNotificationCenter *nc = [NSNotificationCenter defaultCenter];
[nc addObserver:self
   selector:@selector(appDidLaunch:)
   name:UIApplicationDidFinishLaunchingNotification ...];
```

• Posting

```c
[nc postNotificationName:@"UserLoggedIn" ...];
```
Delegation

• One object acts on behalf of another

```c
// UIApplication delegate
- (void)applicationWillTerminate:(UIApplication *)app;

// UITableView delegate
- (BOOL)tableView:(UITableView *)tableView
canEditRowAtIndexPath:(NSIndexPath *)indexPath;
```

• Avoid subclassing when a delegate API is defined
UIKit

Views and Controls
**UIView**

- Your Application’s Canvas
- Provides services for subclasses
  - Handle touch events
  - Layout subviews and drawing
- Views arranged in hierarchy
  - Parent views can contain subviews
  - Rooted in a UIWindow (UIView subclass)
Creating UIViews

• Usually subclassed

• Create subclass instances of UIButton, etc.
  • aView = [[SomeView alloc] initWithFrame: frame]
    ▪ frame: rectangle on which UIView is drawn

• Installing view
  • [containerView addSubview: aView]
UIView Hierarchy

Container

SubviewA
SubviewB
SubviewC
SubviewD

Container

Subviews

SubviewA
SubviewB
SubviewC
SubviewD
Managing Hierarchy

@property nonatomic readonly UIView *superview;
@property nonatomic readonly NSArray *subviews;
@property nonatomic readonly UIWindow *window;

// Modifying Tree
- (void)removeFromSuperview;
- (void)insertSubview:(UIView *)view atIndex:(NSInteger)index;
- (void)addSubview:(UIView *)view;

// Layering
- (void)bringSubviewToFront:(UIView *)view;

// Finding out about hierarchy changes
- (void)willMoveToSuperview:(UIView *)newSuperview;
- (void)didMoveToSuperview;
- (void)willMoveToWindow:(UIWindow *)newWindow;
- (void)didMoveToWindow;
Geometry

- Uses structures found in CoreGraphics framework

```c
#import <CoreGraphics/CGGeometry.h>

struct CGPoint {
    CGFloat x;
    CGFloat y;
};
typedef struct CGPoint CGPoint;
```

- **CGPoint** – 2D coordinate (x,y)
- **CGSize** - 2D dimension (width, height)
- **CGRect** - CGPoint, and CGSize

- These are structures, not objects
Coordinates

• Positive Y is down
Position and Size

- **Bounds** – (origin, size) in local view coordinates
- **Frame** – (origin, size) in superview’s coordinates
- **Center** – view’s center in superview’s coordinates
- *Bounds, frame, and center* are dependent properties
  - When one changes, the others are updated
Bounds, Frame & Center

Superview

View

View’s bounds
origin {0, 0}
size {50, 30}

View’s frame
origin {50, 10}
size {50, 30}

View’s center
center {75, 25}
Using Frame & Bounds

• Rule of thumb
  ▪ Using a view – use frame
  ▪ Implementing a view – use bounds
Transforms

- Rotate, Translate & Scale
  - Relative to view’s center point
    - E.g. rotate about the center point

- Useful Items

```c
// Standard transforms
CGAffineTransform CGAffineTransformIdentity;

// Build transforms
CGAffineTransform CGAffineTransformScale(CGAffineTransform t, CGFloat sx, CGFloat sy);
CGAffineTransform CGAffineTransformRotate(CGAffineTransform t, CGFloat angle);
CGAffineTransform CGAffineTransformTranslate(CGAffineTransform t, CGFloat x, CGFloat y);

// Operations
CGAffineTransform CGAffineTransformInvert(CGAffineTransform t);
```
Sizing Views

• When can views resize?
  ▪ During UI rotation (landscape vs. portrait)
  ▪ In-Call UI (status bar height can change)
    ▪ You call setFrame:

• Parent views can cause subviews to resize
  ▪ Enable by setting property BOOL autoresizesSubviews
  ▪ Configure via UIViewAutoresizing autoresizingMask
    ▪ Specify which parts you want automatically changed in response to parent changes
Autoresizing Mask

• Defines how subview changes in response to superview change
• When superview’s autoresizesSubviews == YES

```swift
// If set, specified margin can grow/shrink
UIViewAutoresizingFlexibleLeftMargin
UIViewAutoresizingFlexibleRightMargin
UIViewAutoresizingFlexibleTopMargin
UIViewAutoresizingFlexibleBottomMargin

UIViewAutoresizingFlexibleWidth
UIViewAutoresizingFlexibleHeight

// A bit-mask, options are combined using bitwise-OR ‘|’
```

```swift
// View is auto-resizable in both dimensions
v.autoresizingMask = UIViewAutoresizingFlexibleWidth | UIViewAutoresizingFlexibleHeight;
```
Autoresizing Mask Example

```objc
UIView *view = [[SomeViewClass alloc] initWithFrame:f];
[container addSubview:view];

container.autoresizesSubviews = YES;
view.autoresizingMask = UIViewAutoResizingFlexibleRightMargin |
                         UIViewAutoResizingFlexibleBottomMargin |
                         UIViewAutoResizingFlexibleWidth;
```
UIView *view = [[SomeViewClass alloc] initWithFrame:f];
[container addSubview:view];

container.autoResizesSubviews = YES;
view.autoresizingMask = UIViewAutoresizingFlexibleRightMargin |
                        UIViewAutoresizingFlexibleBottomMargin |
                        UIViewAutoresizingFlexibleWidth;
Calculating Sizes

- UIView defines interface for picking good view size
- Views can resize to “best size”
  - Use - (void)sizeToFit
- Report “best size” with -(CGSize)sizeThatFits:(CGSize)size;

```swift
UILabel *label = [[UILabel alloc] initWithFrame: ...];
[label setText: @"Congrats, you won the game!"];
[label sizeToFit];
[container addSubview:label];
```
Reading

• “iPhone Application Programming Guide”
  ▪ Chapter 2, “Windows and Views” p.47 - 77

• Class References @ developer.apple.com
  ▪ Overview, and Tasks sections…
  ▪ UIView class reference
  ▪ UIImageView class reference
  ▪ UILabel class reference
UIView Subclasses

For reading at home
Concrete UIViews

• Commonly used UIView subclasses
  - UIImageView
  - UILabel
  -UIScrollView
UIImageView

- Create using an image
- Provide series of images for animation

```
@interface UI ImageView : UIView

-(id)initWithImage:(UIImage *)image;

@property(nonatomic,copy) NSArray *animationImages;

-(void)startAnimating;
-(void)stopAnimating;
-(BOOL)isAnimating;

@end
```
UIImage Example

// Load “Dot.png” into a UIImage
UIImage *image = [UIImage imageNamed:@”Dot.png”];

// Create an image view to display the dot
UIImageView *imageView = nil;
imageView = [[UIImageView alloc] initWithImage:image];

// Add the image view to the hierarchy
[contentView addSubview:imageView];
[imageView release];
UILabel

- Display text
- Control font, color, shadow, layout options
- Autosize text to fit

@interface UILabel : UIView
@property(nonatomic,copy) NSString *text;
@property(nonatomic,retain) UIFont *font;
@property(nonatomic,retain) UIColor*textColor;
@property(nonatomic,retain) UIColor*shadowColor;
@property(nonatomic) BOOL adjustsFontSizeToFitWidth;

// From UIView...
- (CGSize)sizeThatFits:(CGSize)size;
- (void)sizeToFit;
@end
UIScrollView

- Manages a content view which may be bigger than its own bounds
- Users scroll using swipe gestures
- Content view drawing clipped to scroll view frame
- Scroll view itself - no drawing except for scroll indicators
- Zooming, Bouncing API
- Normally don’t use UIScrollView directly
  - UITableView, UITextView are subclasses of UIScrollView