iOS Dev Accelerator
Week 1 Day 3

- App Delegate
- UIWindow
- Views
- FirstResponder
- Textfields
- KVO Notifications

AppDelegate

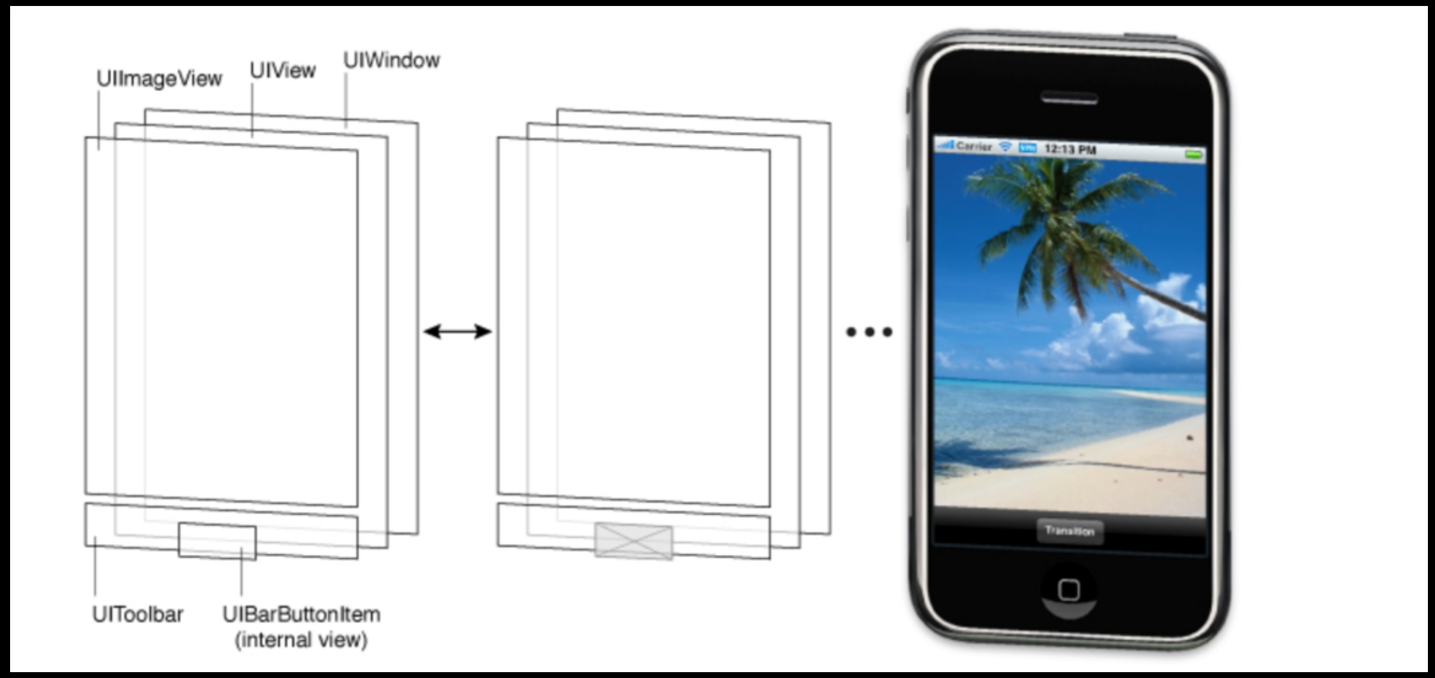
- The app delegate is a custom object created at launch
- The primary job of the app delegate is to bootstrap application startup, then handle transitions to and from the background.
- AppDelegate handles APN, openURL, NSURLSession downloads…
- Great place to manually set the root view controller.
- Demo

UIWindow

- UIWindow provides an area to to draw UIViews
- The most important property of a UIWindow is its `rootViewController` property
- Assigning a `rootViewController` sets the `rootViewController`'s view as the `contentView` of the window
- UIWindow also has a `screen` property; the screen displaying the window
- By default, all windows are created on the primary device screen
- Displaying views on an airplay display is an example of a non-primary screen

AppDelegate – Demo

Views



- “A view is an instance of the `UIView` class (or one of its subclasses) and manages a rectangular area in your application window”
- “Views are responsible for drawing content, handling multitouch events, and managing the layout of any subviews. “

Views

- To be displayed on screen, a `UIView` must be added as a subview of a parent view.
- All views can be a child view of a parent view, and all views can have children views of their own. This is called the view hierarchy.
- A UIView has many properties that dictate its appearance:
  - backgroundColor
  - alpha
  - hidden
  - opaque
  - tintColor
  - layer
  - clipsToBounds
- The most common `UIView` properties affect its location and size: `Frame and Bounds`

Frame and Bounds

- `Frame` - Describes the view's location and size **in its superview's coordinate system.**
- `Bounds` - Describes the view's location and size **in its own coordinate system.**
- Both are `CGRect` data structures
  - x - offset from the left edge
  - y - offset from the top edge (bottom edge in `AppKit/SpriteKit`)
  - width - width of the rectangle
  - height - height of the rectangle

`UIResponder`

- The `UIResponder` class defines an interface for objects that respond to and handle events
- `UIView` inherits from `UIResponder`, so all views can respond to and handle events
- Touch motion events are the two relevant event types in Cocoa Touch
- `firstResponder` is the object at the top of the `responderChain`
  - events are sent to the firstResponder for that event type
  - If the responder object resigns `firstResponder`, the event “bubbles up” the `responderChain` to the next responder

Responder Chain

- When a user-generated event happens, UIKit creates an event with all the pertinent info and places it in the event queue.
- Touch events send an `NSSet` of touches packed in a `UIEvent` object
  - motion events can include a number of different objects depending on the framework.
- The event then travels up the `responderChain` until it hits an object that can handle the event.
  - You will most likely never need to manually worry about the `responderChain` and what views a touch will interact with.
  - If you ever need to make a view completely ignore and pass touches through, just set its `enableUserInteraction` property to false.

`UITextField`

- A `UITextField` object is a control that displays editable text and sends an action message to a target object when the user presses the return button
- A `UITextField` can have a delegate to handle editing related notifications
- When a user taps into a `UITextField`, it becomes the `firstResponder` and it brings the keyboard on screen
- You are responsible for making sure the `UITextField` you are editing is not obscured by the keyboard

UIView Animation API

- Animates changes to properties over time in a `UIView`
  - can animate most properties, including `frame` and `bounds`
  - animates using linear keyframes by default
  - can use alternate key-frame timing (`easeIn, easeOut`, etc.)
- `UIView.animateWithDuration()`
  - Class method on `UIView` that takes 2 parameters
    - the `duration` of the animation
    - a closure containing the desired changes
- We will cover closures more in depth in Week 3, but they are just like blocks if you are coming from Objective-C.

UIView Animation API – Demo