iOS Foundations II
Session 6

- Stacked Learning
- Day 5 Homework Discussion
- UITextField: String & Keyboard
- Reloading the table view
- UIImage & UIImageView
- Camera Programming & UIImagePickerController
- Optionals

Stacked Learning

- https://www.codefellows.org/blog/how-to-accelerate-your-learning-with-stacked-modules

Day 5 Homework Discussion

- Auto-Layout
  - Size and position
  - Testing: 4 orientations (upside-down is ok? Tilt head?)
  - Testing: iPhone4, iPad Retina
- UITextField and keyboard
  - Dismiss keyboard on [Return]
  - [Command]-K to reset default

UITextField: String & Keyboard

- Handle specific **widget** event (e.g. button click callback)

```
@IBAction func myButtonClickEventHandlingFunction(sender: UIButton) {
        myTextField.resignFirstResponder(); // This UITextField pulls up the kbd
                                            //   Mnemonic: “resign” = “dismiss”
        txtEcho.text = myTextField.text; // Aside: access String data in UITextField
}
```

- Handle click **outside** UITextField (where click is **not** on another widget)

```
override func touchesBegan(touches: Set<NSObject>, withEvent event: UIEvent?) {
    super.touchesBegan(touches, withEvent: event!)
    myTextField.resignFirstResponder()
 // myTextField.endEditing(true)  // Also dismisses kb
}
```

- Dismiss keyboard on [**Return**] key press in UITextField

```
func textFieldShouldReturn(txtfield: UITextField) -> Bool {
    myTextField.resignFirstResponder();
    return true;
}
```

Demo 6.1: UITextField: String & Keyboard

UIImage

- High-level way to display images (no need to deal with lower-level considerations such as codecs, sub-formats, pixel values, etc.)
- Immutable. You can’t change it after creation, so they’re thread safe.
- The only way to “change” a UIImage is to make a modified copy of it.
- You not allowed to access the underlying binary image data.
- Use UIImagePNGRepresentation or UIImageJPEGRepresenation functions to get an NSData from a UIImage.

UIImage Supported Formats

Format	Filename extensions
Tagged Image File Format (TIFF)	`.tiff`, `.tif`
Joint Photographic Experts Group (JPEG)	`.jpg`, `.jpeg`
Graphic Interchange Format (GIF)	`.gif`
Portable Network Graphic (PNG)	`.png`
Windows Bitmap Format (DIB)	`.bmp`, `.BMPf`
Windows Icon Format	`.ico`
Windows Cursor	`.cur`
X Window System bitmap	`.xbm`

UIImage Methods

- Loading an image you've dragged into your project:

```
myUIImageView.image = UIImage(named: "silverPiano.jpg")
```

- Returns image object associated w/ specified image name (folder name not needed).
- Uses caching. If image isn't in cache, image loads from the main bundle, caches it, then returns the image.
- Since iOS 4, it is not no longer necessary to put .png into your string if its a png file. Still need .jpeg or .jpg though!

  If you know this image is only going to be used once and don't need caching, use this to save memory:

```
UIImage(contentsOfFile: "/Users/me/bigBuxGeneratingApps/CastleWars/burningCastle.jpg")
```

UIImage Methods

- `myUIImageView.image = UIImage(CGImage: myCGImage)`

```
func CGImageCreate(_ width: Int,
                   _ height: Int,
                   _ bitsPerComponent: Int,
                   _ bitsPerPixel: Int,
                   _ bytesPerRow: Int,
                   _ colorspace: CGColorSpace!,
                   _ bitmapInfo: CGBitmapInfo,
                   _ provider: CGDataProvider!,
                   _ decode: UnsafePointer<CGFloat>,
                   _ shouldInterpolate: Bool,
                   _ intent: CGColorRenderingIntent) -> CGImage!
```

- Data passed in can be from a file or from data you created in RAM.
- Returns *nil* if it could not initialize the image from the specified data.
- No caching (“direct copy”)

UIImageView

- View based container for displaying images(s)
- The image displayed is sized, scaled to fit, or positioned in the image view based on the imageView's contentMode.
- Apple recommends images displayed using the same imageView to be the same size. If image sizes are very different, the system will create a new scaled-down image, which consumes more RAM!

UIImageView Optimization

- Pre-scale your images: if the imageView you using is very small, generate thumbnails of your images in a cache vs scaling large images to fit the small view.
- Disable alpha blending: Unless you are intentionally working transparency, you should mark your imageView's as Opaque.

Demo 6.2: UIImageView

Camera / Photo Library

- 2 ways for interfacing with the camera in your app:
  1. UIImagePickerController (basic)
  2. AVFoundation Framework (advanced)
- We are going to use **UIImagePickerController** in the demo.
- UIImagePickerController is a view controller you configure then present.
- Its built-in features let a user pick a photo from the photo library or by using the camera.

UIImagePickerController

- The workflow of using UIImagePickerController is 3 steps:
    1. Instantiate and modally present the UIImagePickerController
    2. ImagePicker manages the user's interaction with the camera or photo library
    3. The system invokes your image picker controller delegate methods to handle the user being done with the picker.

UIImagePickerController Setup

- The first thing you have to account for is checking if the device has a camera.
- If your app absolutely relies on a camera, add a UIRequiredDeviceCapabilities key in your info.plist
- Use the `isSourceTypeAvailable` ***class method*** on UIImagePickerController to check if camera is available (it wont be available on the simulator)

UIImagePickerController Setup

- Next make sure something is setup to be the delegate of the picker. This is usually the view controller that is spawning the picker.
- The final step is to actually create the UIImagePicker with a sourceType of UIImagePickerControllerSourceTypeCamera.
- Media Types: Used to specify if the camera should be locked to photos, videos, or both.
- AllowsEditing property to set if the user is able to modify the photo in the picker after taking the photo.

UIImagePickerControllerDelegate

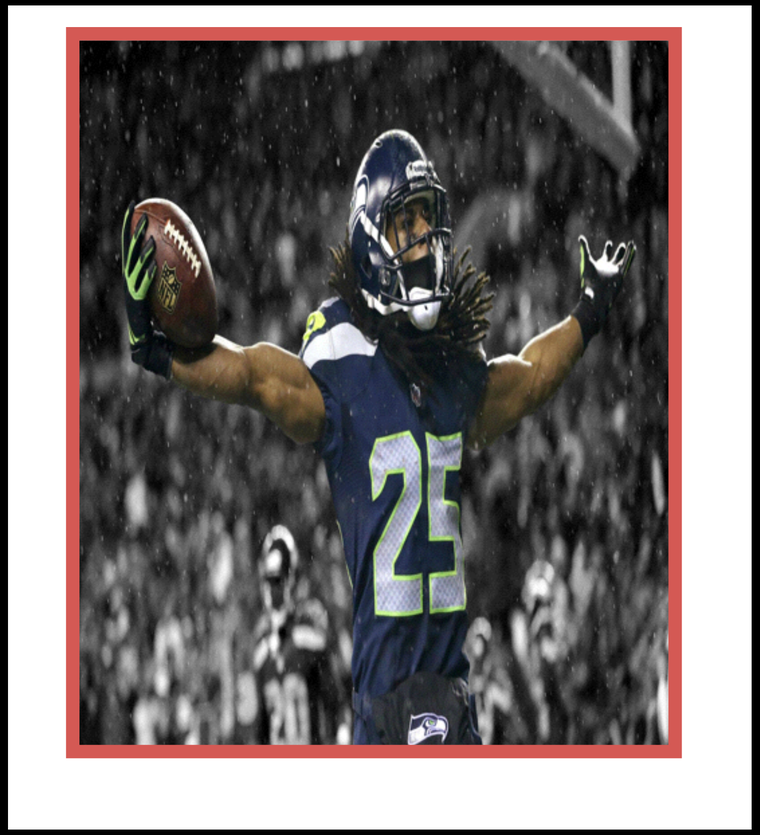
- The Delegate methods control what happens after the user is done using the picker. 2 big methods:
  1. imagePickerControllerDidCancel:
  2. imagePickerController:didFinishPickingMediaWithInfo:

UIImageView Content Mode

```
typedef enum {
   UIViewContentModeScaleToFill,
   UIViewContentModeScaleAspectFit,
   UIViewContentModeScaleAspectFill,
   UIViewContentModeRedraw,
   UIViewContentModeCenter,
   UIViewContentModeTop,
   UIViewContentModeBottom,
   UIViewContentModeLeft,
   UIViewContentModeRight,
   UIViewContentModeTopLeft,
   UIViewContentModeTopRight,
   UIViewContentModeBottomLeft,
   UIViewContentModeBottomRight,
} UIViewContentMode;
```

- ScaleToFill: scale content to fit the size of itself by changing the aspect ratio if necessary
- ScaleAspectFit: scale content to fit the size of the view by maintaining aspect ratios. Any remaining area of the view's bounds is transparent.
- ScaleAspectFill:scale content to fill the size of the view. Some content maybe be clipped to fill the view's bounds.

UIImageView Content Mode



ScaleToFill:

AspectFit:

AspectFill

Demo 6.3: UIImagePickerController

Optionals

Optionals

- You use optionals in situations where a value may be absent.
- An optional says:
  - There is a value and it equals x

    OR

  - There isn't a value **at all**. **nil** indicates this condition.

    The "value" is not zero (0) or "false".
- The concept of optionals does not exist in Objective-C, C, or C++!

Swift is strict!

- Swift does not let you leave properties in an undetermined state.
- They must be…
  - …given a default value

    OR
  - …set to a value in an initializer statement or init()

    OR
  - …marked as **optional**

Marking an Optional

- A question mark (?) at the end of a variable definition indicates that the value it contains is optional, meaning it might contain a value or it might be empty:

```
class Person {

    var firstName : String?
```

Accessing a value inside an optional

To access the value inside an optional variable, you must force an unwrapping with an exclamation mark:

```
// return the first and last names in a string
func returnFullName() -> String {
    return "\(self.firstName!) \(self.lastName)"
}
```

You can also use a question mark to do “optional chaining”, a process for querying and calling properties,methods, and subscripts on an optional that might currently be nil.

Implicitly Unwrapped

Instead of marking an optional with a ?, using an exclamation point ! will mark it as implicit unwrapped:

```
class Person {

    var firstName : String!
```

This makes it so you don't have to unwrap this optional to access its value.
So basically you are saying “I’m making this an optional, but this thing will always have a value when I need to access it”

Demo 6.4: Optionals