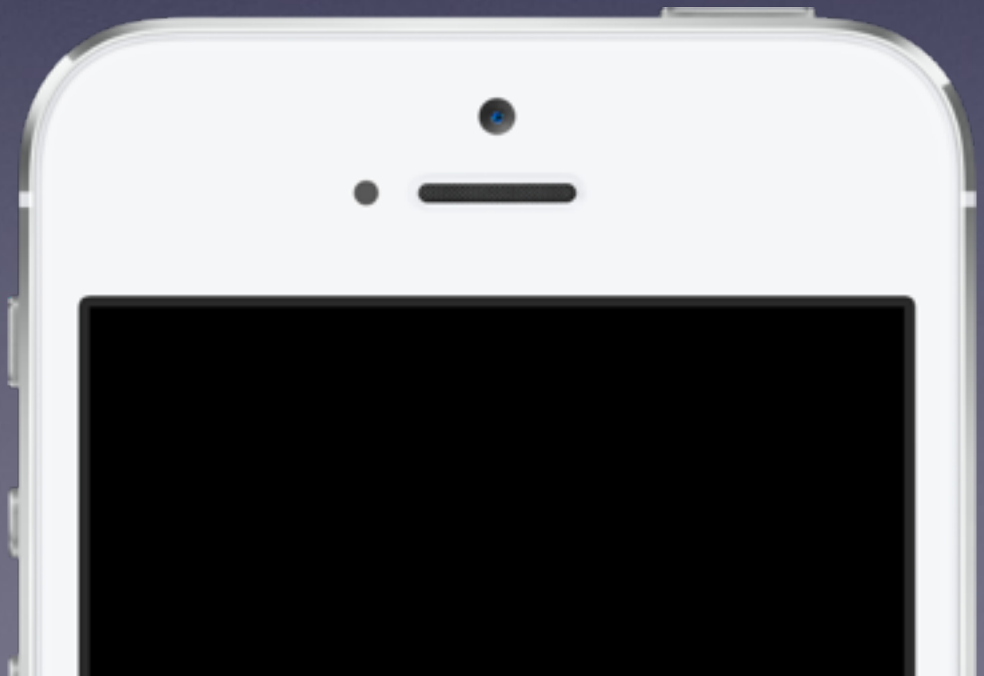
COMS W3101: Programming for iOS

Michael Vitrano



Images, ScrollViews, and Frameworks

- UIImage
- UIImagePickerController
- UIScrollView
- NSNotification Center
- Frameworks
- Demo - Integrating Aviary’s SDK

UIImage

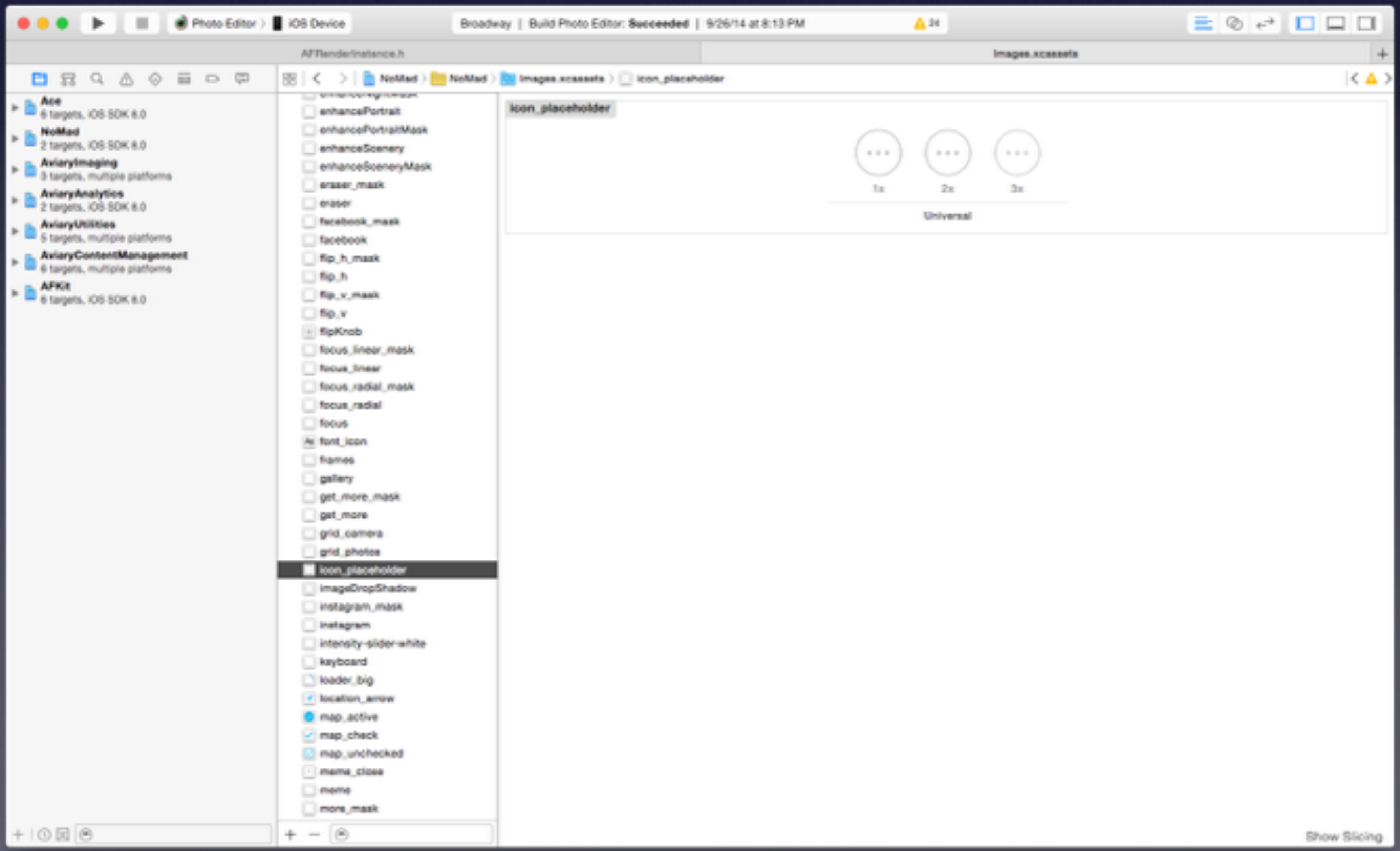
- **UIImage** is UIKit’s object used to represent image data
- Images can be created in a number of ways:
  - Read from a file on disk
  - Created from data downloaded off the network
  - Drawn using Core Graphics
- UIImage instances are immutable
- Instances do not provide access to the raw data that represents the image
  - You can get NSData representations by using the **UIImagePNGRepresentation** and **UIImageJPGRepresentation** functions

Using Bundled Images

- Images are oftentimes essential for creating the UI of your app (e.g. icons, buttons)
- For each image to be used, you need three copies, one at the desired size, one at twice that resolution and another at three times the resolution
  - The image with twice and three times the resolution are used for devices with Retina and Retina HD screens respectively
  - The naming convention for these images is: <image_name>.png,<image_name>@2x.png, <image_name>@3x.png
- Images to be used in your app are added to the special **Images.xcassets** folder in your Xcode project

XCAssets

- Your **Images.xcassets** folder is a drag and drop based system for organizing your assets



Accessing and Displaying Images

- Accessing an app’s bundled images is done through the **+imageNamed:** class method on UIImage
- Creating an image from data is done through the **-initWithData:** initializer
- **UIImageView** is UIKit’s builtin class for displaying image in your interface
  - A useful property is the view’s **contentMode**, which allows customization of the stretching of the image
    - **UIViewContentModeScaleToFill**, **UIViewContentModeScaleAspectFit**, **UIViewContentModeScaleAspectFill**

Accessing a User’s Images

- **UIImagePickerController** is a built-in VC for taking pictures and choosing from pictures saved to the camera
- Before presentation, changing the **sourceType** property adjusts the source of the images shown for the use to choose
- Adjusting the **mediaTypes** property allows configuration of whether to allow the user to choose photos, videos or both

UIImagePickerControllerDelegate

- UIImagePickerController instances' delegate object must conform to both the **UINavigationControllerDelegate** and the **UIImagePickerViewControllerDelegate** protocols
- The **UIImagePickerViewControllerDelegate** has two methods for responding to the controller actions:
    - **imagePickerController:didFinishPickingMediaWithInfo:**
    - **imagePickerDidCancel:**

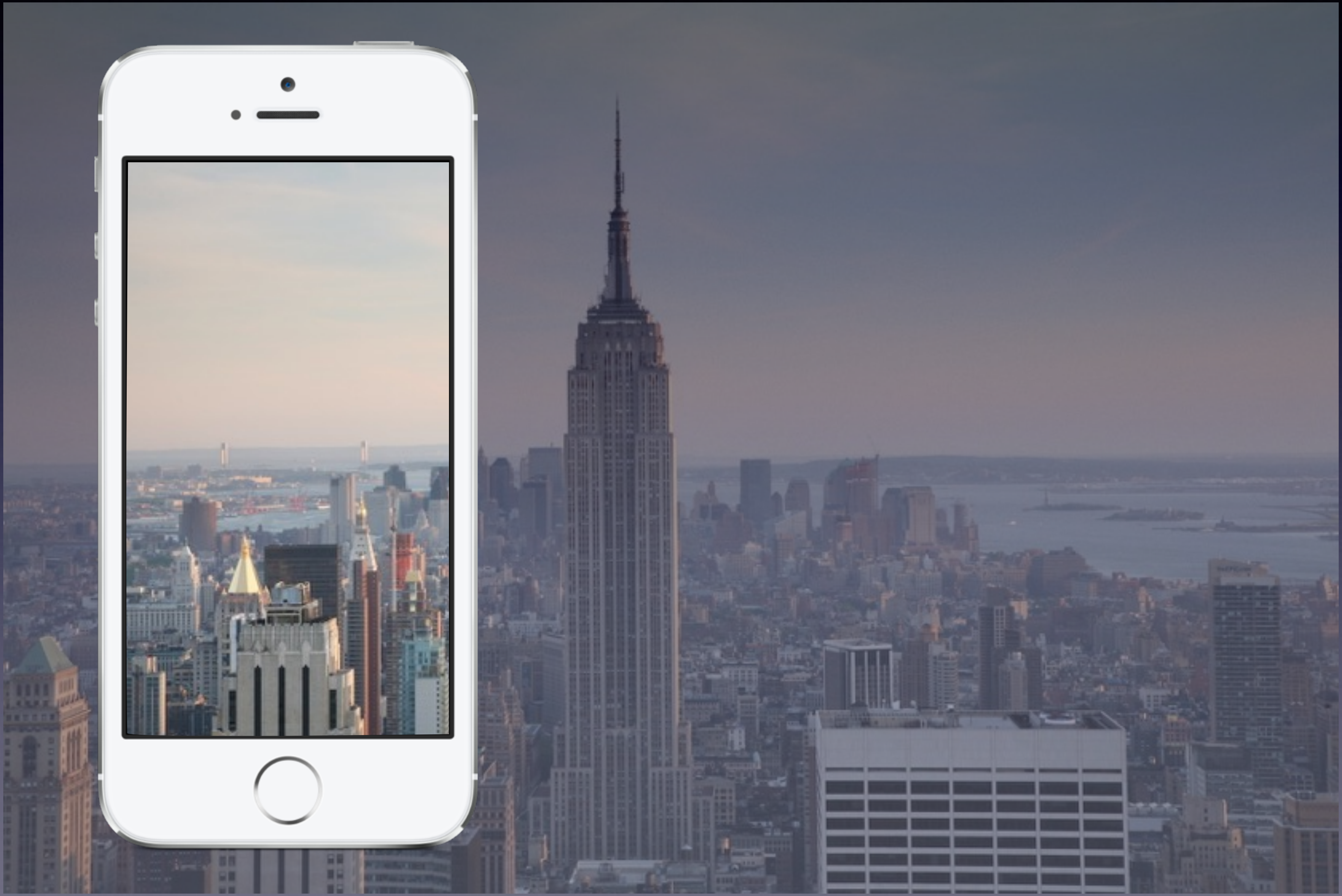
UIImagePickerControllerDelegate

- **imagePickerController:didFinishPickingMediaWithInfo:**
  - This method returns a dictionary that contains all of the information regarding the media that was chosen by the user
  - In most cases, you will want the UIImage representation of the image and can get it using the **UIImagePickerControllerOriginalImage** key
- **imagePickerDidCancel:**
- Both methods are responsible for dismissing the image picker

UIScrollView

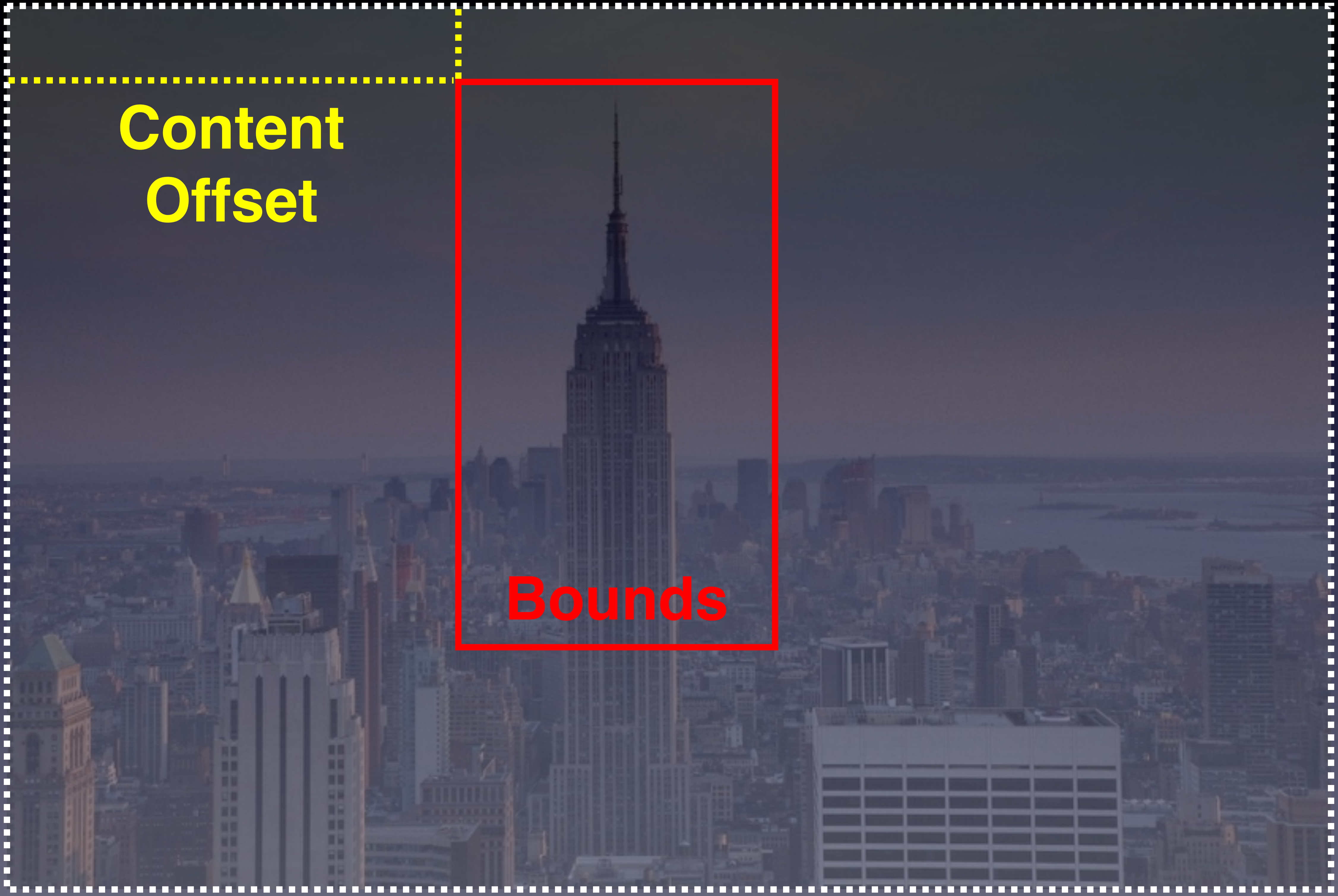
- UIScrollView is a UIView subclass that provides support for displaying content that is larger than the view’s frame
- A scroll view’s **bounds** define a viewport onto the content contained in it
- Scroll Views contain gesture recognizers that handle the user’s interaction with scrolling around that content

UIScrollView



UIScrollView

- The scroll view's **contentSize** property represents the size of the canvas that a user can scroll within
- The scroll view's **contentOffset** property is the origin of the viewport which is currently being displayed by the scroll view
  - This is also the origin of the scroll view's **bounds** property
  - Scrolling around the content simply adjusts the **contentOffset**



Content Size

NSNotificationCenter

- **NSNotificationCenter** is used to broadcast information within your app
    - Oftentimes used by model objects to alert potential observers of changes
    - Also used by UIKit to provide notification of changes to device orientation and keyboard status
- Access the shared notification center with the **+defaultCenter** class method

NSNotificationCenter

- Adding an observer for a notification is done with the **-addObserver:selector:name:object:** instance method
    - The **observer** is the object which will have the **selector** called on it
    - **name** is a string value which uniquely identifies the notification
    - By specifying an **object**, you will only receive notifications posted by that object. This is oftentimes **nil**

NSNotificationCenter

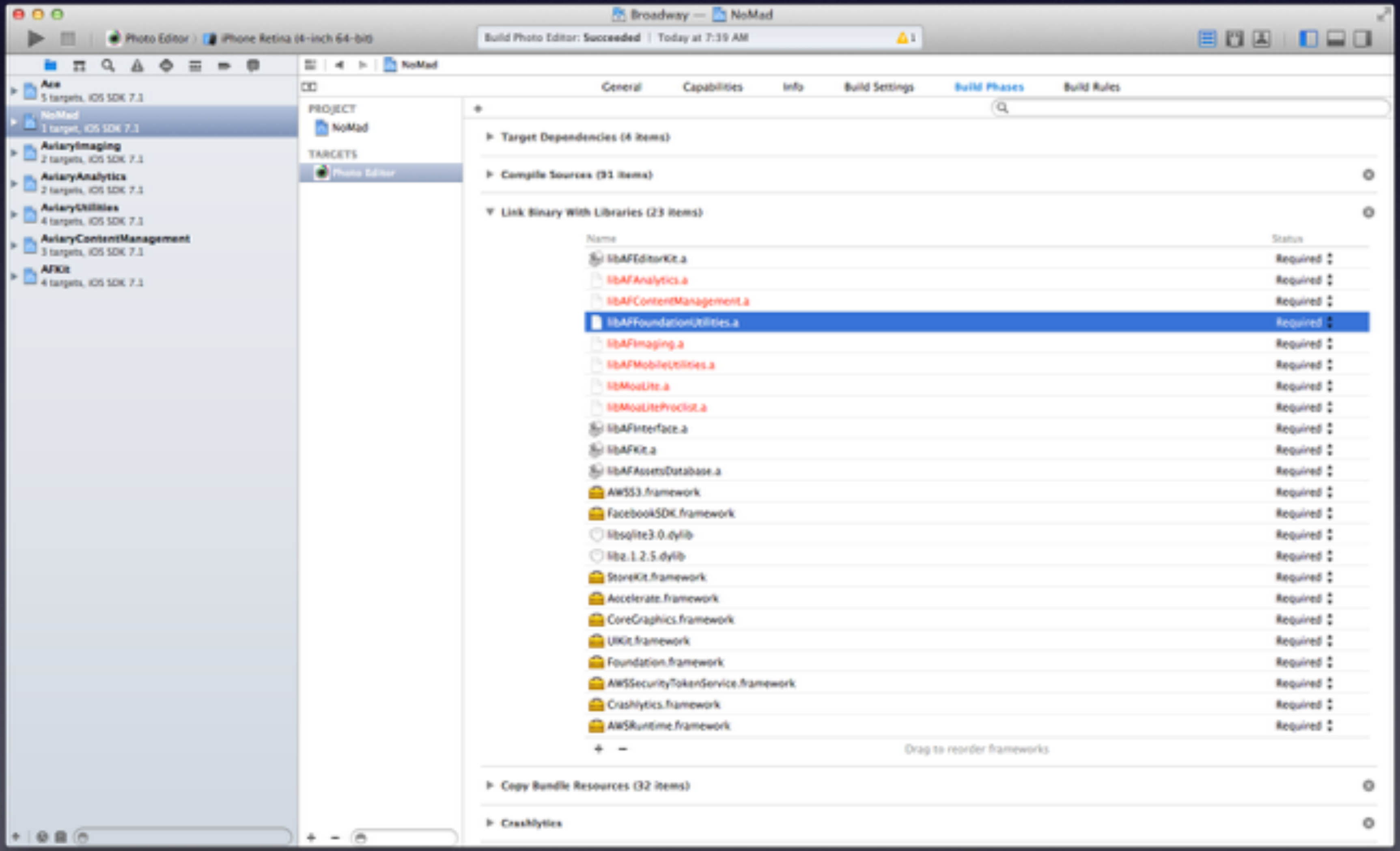
- You must deregister an observer before it is deallocated using the **-removeObserver:** instance method
  - This is typically done by overriding the **-dealloc** method of your observing object
- To post a custom notification, use the **-postNotificationName:object:** instance method

Frameworks and Libraries

- Frameworks and libraries allow you to add additional functionality to your app that are provided by Apple or Third-parties
- In the iOS 8 SDK, there are over a hundred Apple provided frameworks that allow you to do things like send SMSs, work with SQLite databases, use the deice's bluetooth hardware, etc
- Third parties like Facebook or Aviary use Frameworks to ship their SDKs to developers

Adding Frameworks

- Adding a Framework or library is done in the **Build Phases** of your project file:



Broadway — NoMad

Photo Editor › iPhone Retina (4-inch 64-bit)

Build Photo Editor: **Succeeded** | Today at 7:39 AM

- Ace — 5 targets, iOS SDK 7.1
- NoMad — 1 target, iOS SDK 7.1
- AviaryImaging — 2 targets, iOS SDK 7.1
- AviaryAnalytics — 2 targets, iOS SDK 7.1
- AviaryUtilities — 4 targets, iOS SDK 7.1
- AviaryContentManagement — 3 targets, iOS SDK 7.1
- AFKit — 4 targets, iOS SDK 7.1

PROJECT: NoMad

TARGETS: Photo Editor

General | Capabilities | Info | Build Settings | Build Phases | Build Rules

▸ Target Dependencies (4 items)

▸ Compile Sources (91 items)

▾ Link Binary With Libraries (23 items)

Name	Status
libAFEditorKit.a	Required
libAFAnalytics.a	Required
libAFContentManagement.a	Required
libAFFoundationUtilities.a	Required
libAFImaging.a	Required
libAFMobileUtilities.a	Required
libMoaLite.a	Required
libMoaLiteProclist.a	Required
libAFInterface.a	Required
libAFKit.a	Required
libAFAssetsDatabase.a	Required
AWSS3.framework	Required
FacebookSDK.framework	Required
libsqlite3.0.dylib	Required
libz.1.2.5.dylib	Required
StoreKit.framework	Required
Accelerate.framework	Required
CoreGraphics.framework	Required
UIKit.framework	Required
Foundation.framework	Required
AWSSecurityTokenService.framework	Required
Crashlytics.framework	Required
AWSRuntime.framework	Required

Drag to reorder frameworks

▸ Copy Bundle Resources (32 items)

▸ Crashlytics

Accessing a Framework

- Once you have added a framework to your project file, you can reference it in code by importing the framework header:

```
// Importing the header for the Core Data framework
#import <CoreData/CoreData.h>

// Importing the Facebook SDK
#import <FacebookSDK/FacebookSDK.h>

// Importing Aviary’s SDK
#import <AviarySDK/AviarySDK.h>
```