iOS Dev Accelerator
Week 3 Day 1

- Using the Camera
- UIImage & UIImageView
- UIAlertController

Camera Programming

- 2 ways for interfacing with the camera in your app:

  1. UIImagePickerController (basic)

  2. AVFoundation Framework (advanced)

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.

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 method:
  1. imagePickerControllerDidCancel:
  2. imagePickerController:didFinishPickingMediaWithInfo:

Info Dictionary

The info dictionary has a number of items related to the image that was taken:

```
NSString *const UIImagePickerControllerMediaType;
NSString *const UIImagePickerControllerOriginalImage;
NSString *const UIImagePickerControllerEditedImage;
NSString *const UIImagePickerControllerCropRect;
NSString *const UIImagePickerControllerMediaURL;
NSString *const UIImagePickerControllerReferenceURL;
NSString *const UIImagePickerControllerMediaMetadata;
```

MediaType is either kUTTypeImage or kUTTypeMovie

Memory Considerations

- Un-edited, un-compressed images are big (~30MB on iPhone5)
- Displaying/editing/caching large images will raise memory pressure
- GPU-based filters (typically used for video) need to consider the max texture size if tiling is not available (mostly fixed in iOS 8)
- You should avoid creating `UIImage` objects that are greater than 1024 x 1024 in size

UIImage

- UIKit object wrapper around an image in memory.
- UIImage's are immutable, you cannot change them after creation. Which means they are thread safe.
- The only way to change a UIImage to make a copy of it.
- Since they are immutable, you are not allowed to access their underlying binary image data.
- Use the UIImagePNGRepresentation or UIImageJPEGRepresenation functions to get an NSData from a UIImage.
- In low-memory situations, image data may be purged from a `UIImage` object to free up memory on the system.

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

- `class func imageNamed(String) -> UIImage`
- Returns an image object associated with a specified file name.
- Uses caching. If the image isn't in the cache, it then loads the image from the main bundle, caches it, and 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 though!
- .PNG should still be used whenever possible for optimal performance
- If you know this image is only going to be used once, and don't need the caching, use the method `imageWithContentsOfFile` which doesn't cache. Saves memory.

UIImage Methods

- `class func imageWithData(NSData) ->UIImage`
- The data passed in can be from a file or data you created.
- returns nil if it could not initialize the image from the specified data.
- no caching.

UIImageView

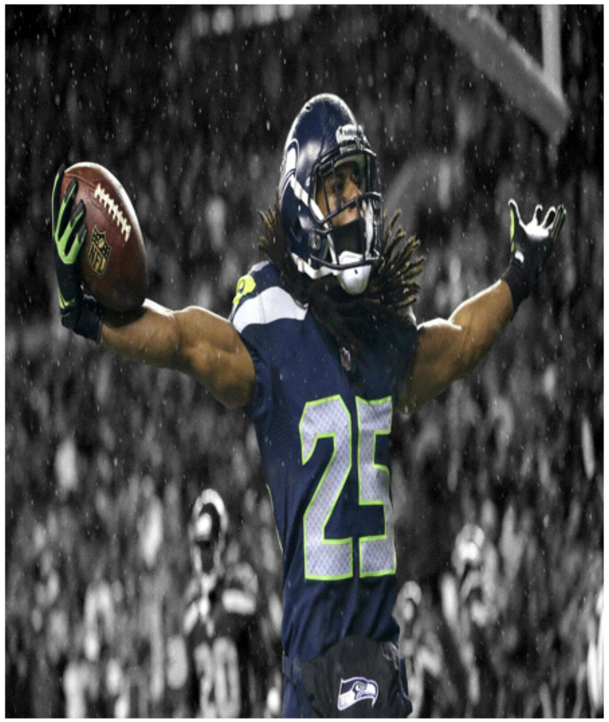
- View based container for displaying images(s)
- The image displayed is sized, scaled to fit, or positioned in the image view based on the imageView's contentMode.
- Apple recommends images displayed using the same imageView be the same size.

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

UIImageView Optimization

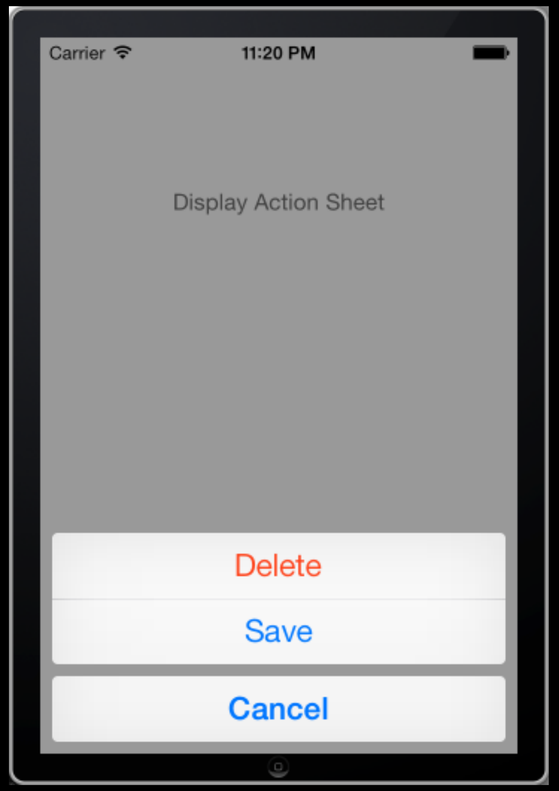
- Pre-scale your images: if the imageView you using is very small, generate thumbnails of your images in a cache vs scaling large images to fit the small view.
- Disable alpha blending: Unless you are intentionally working transparency, you should mark your imageView's as Opaque.

UIImageView Animation

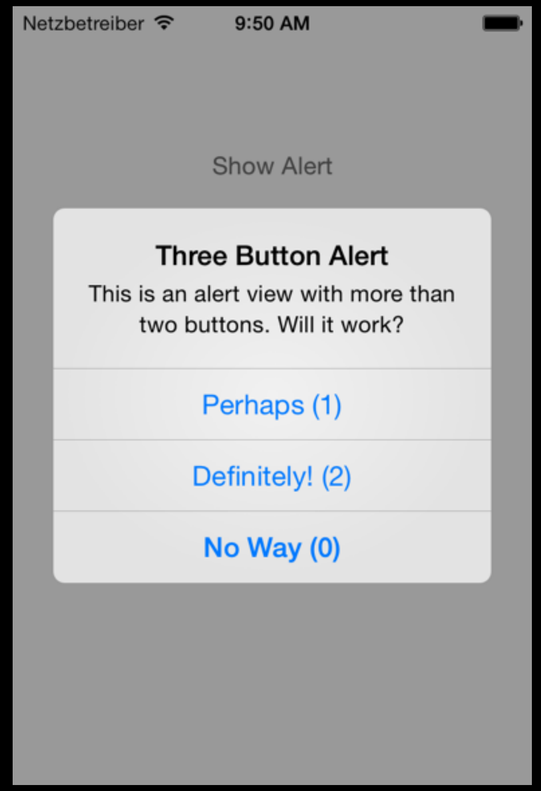
- Workflow for getting an image based animation:
    1. Create an array of the animation frame images
    2. set your imageView's .animationImages to the array
    3. set your imageView's animationDuration
    4. call startAnimating on your imageView

UIAlertController

- “UIAlertController object displays an alert message to the user”
- Replaces both UIActionSheet and UIAlertView
- After configuring the Alert Controller present it with presentViewController:animated: Completion:



ActionSheet



AlertView

UIAlertController Setup

`init(title:message:preferredStyle:)`

Creates and returns a view controller for displaying an alert to the user.

Declaration

SWIFT

```
convenience init(title title: String!,
                 message message: String!,
           preferredStyle preferredStyle: UIAlertControllerStyle)
```

Parameters

title	The title of the alert. Use this string to get the user's attention and communicate the reason for the alert.
message	Descriptive text that provides additional details about the reason for the alert.
preferredStyle	The style to use when presenting the alert controller. Use this parameter to configure the alert controller as an action sheet or as a modal alert.

UIAlertController Configuration

- In order to add buttons to your alert controller, you need to add actions.
- An action is a instance of the UIAlertAction class.
- “A UIAlertAction object represents an action that can be taken when tapping a button in an alert”
- Uses a closure to define the behavior of when the button is pressed. This is called the handler.

UIAlertAction Setup

`init(title:style:handler:)`

Create and return an action with the specified title and behavior.

Declaration

SWIFT

```
convenience init(title title: String!,
                 style style: UIAlertActionStyle,
               handler handler: ((UIAlertAction!) -> Void)!)
```

Parameters

title	The text to use for the button title. The value you specify should be localized for the user's current language. This parameter must not be `nil`.
style	Additional styling information to apply to the button. Use the style information to convey the type of action that is performed by the button. For a list of possible values, see the constants in UIAlertActionStyle.
handler	A block to execute when the user selects the action. This block has no return value and takes the selected action object as its only parameter.

Return Value

A new alert action object.

Feedback

Adding Actions

- Adding actions to the AlertController is as easy as calling addAction: on your AlertController and passing in the UIAlertAction(s)
- The order in which you add those actions determines their order in the resulting AlertController.