Media Playback and Recording

CS193W - Spring 2016 - Lecture 3
Today

- Images and animated images
- Text input controller
- Media playback controller
- Inline video playback
- Playing extended audio
- Recording audio
Images
Ways to show an Image

• The **WKInterfaceImage** class displays a single image or a sequence of images as standalone content.

• The **WKInterfaceGroup**, **WKInterfaceButton**, and **WKInterfaceController** classes allow you to specify an image as the background for other content.

• The **WKInterfaceSlider** class can display custom images for the increment and decrement controls.

• The **WKInterfaceMovie** class displays a poster image for video or audio content.

• The **WKInterfacePicker** class displays items that can contain images.
WKInterfaceImage

- Displays a single image or an animated sequence of images.

- All images should be designed for retina displays and should have the @2x suffix

  e.g. myimage@2x.png
Image Asset Guidelines

• The preferred image types is PNG. JPEG is almost as good. Other image types can cause performance issues when rendering.

• Use the 8-bit color palette for PNG graphics that don’t require full 24-bit color.

• For JPEGs, make sure to set the quality no higher than necessary

• Avoid resizing images on the watch whenever possible. Create images at the desired size.

• Avoid transparency if possible
Image Caching

• In watchOS 1, because transferring between the extension to the app was expensive, there was a built-in image cache

• In watchOS 2, this is no longer needed
WKInterfaceImage API

func setImage(_ image: UIImage?)

func setImageData(_ imageData: NSData?)

func setImageNamed(_ imageName: String?)

func setTintColor(_ tintColor: UIColor?)
Where to Place Images

- You can place images in either your WatchKit App target or your WatchKit Extension target. Both allow you to use `setImageNamed:` and to use images in the storyboard.
Animatable Images

- Create an animatable `UIImage (not a WKImage)`

```swift
class func animatedImageNamed(_ name: String,
    duration duration: NSTimeInterval) -> UIImage
```

If `name` is `myimage`, then the images in your bundle should be named `myimage0`, `myimage1`, `myimage2`, etc.
Animating WKInterfaceImage

WKInterfaceImage conforms to the WKImageAnimatable protocol:

```swift
func startAnimating()
func stopAnimating()
func startAnimatingWithImagesInRange(_ imageRange: NSRange, duration duration: NSTimeInterval, repeatCount repeatCount: Int)
```

- `imageRange`: 0 represents the first image in the sequence
- `duration`: Loop time in seconds. Negative values cause the image to loop in reverse.
- `repeatCount`: Specify 0 to loop indefinitely.
Images in the Storyboard

- Fill in the image name, animate, and duration if desired
- Note that a question mark will show up for animated images; nothing is wrong.
Text Input Controllers
Text Input Controllers

- Suggestions
- Emoji
- Voice Dictation

Note: Voice dictation is not available in the simulator.
Emoji Pickers

Static Emoji

Animated Emoji
Presenting Text Input Controllers

```swift
func presentTextInputControllerWithSuggestions(_ suggestions: [String]?,
  allowedInputMode inputMode: WKTextInputMode,
  completion completion: ([AnyObject]?) -> Void)

enum WKTextInputMode : Int {
  case Plain
  case AllowEmoji
  case AllowAnimatedEmoji
}

func dismissTextInputController()

• The result will either be nil (if the user cancels) or an array of a single element (a String or BSData representing an image). Note that emoji are returned as Strings.

• Passing nil to suggestions results in the voice dictation screen being brought up directly.
Text Input Controller Example

```swift
self.presentTextInputControllerWithSuggestions(["foo", "bar", "baz"],
    allowedInputMode: .AllowAnimatedEmoji) {
(answers) -> Void in
    if (answers != nil) {
        if let resultString = answers?.first as? String {
            print(resultString);
        } else if let resultImageData = answers?.first as? NSData {
            let image = UIImage(data: resultImageData)
        }
    }
}
```
Using the Media Player Controller
Media Player

- A modal interface that can play audio or video
Media Player API

```swift
func presentMediaPlayerControllerWithURL(_ URL: NSURL,
    options options: [NSObject : AnyObject]?,
    completion completion: (Bool,
        NSTimeInterval, NSError?) -> Void)

func dismissMediaPlayerController()
```

Completion arguments:

- `didPlayToEnd` - true if the media playback completed
- `endTime` - the point at which playback was terminated, in seconds
- `error` - the error object, or nil

Note that calling `dismissMediaPlayerController` results in `endTime` being passed back as 0.0.
Media Player URL

- Can be a local URL or a remote one
- If it is remote, it must be secure (https)
- In the case of a remote URL, a progress indicator is shown while the media is downloading
Media Player Options

WKMediaPlayerControllerOptionsAutoplayKey
True if the media player starts playing automatically; the default is false.

WKMediaPlayerControllerOptionsStartTimeKey
The start time, in seconds.

WKMediaPlayerControllerOptionsVideoGravityKey
  .ResizeAspect - Size to fit, preserving aspect ratio. No cropping.
  .ResizeAspectFill - Size to fill, preserving aspect ratio. Allows cropping.
  .Resize - Size to fill, not preserving aspect ratio. No cropping.

WKMediaPlayerControllerOptionsLoopsKey
True if the content plays repeatedly in a loop
Playing Inline Videos
WKInterfaceMovie

func setMovieURL(_ URL: NSURL)

func setVideoGravity(_ videoGravity: WKVideoGravity)

func setLoops(_ loops: Bool)

func setPosterImage(_ posterImage: WKImage?)

An placeholder image to show while the movie is not playing.
Playing Extended Audio
Classes Involved in Background Audio Playback

WKAudioFileAsset
Stores a reference to an audio file and provides metadata access

WKAudioFilePlayerItem
Manages the state of an WKAudioFileAsset as it is being played

WKAudioFilePlayer
Controls playback of a single WKAudioFilePlayerItem

WKAudioFileQueuePlayer
Controls playback of multiple WKAudioFilePlayerItems
WKAudioFileAsset

Initializers

convenience init(URL URL: NSURL)

convenience init(URL URL: NSURL, 
    title title: String?, 
    albumTitle albumTitle: String?, 
    artist artist: String?)

Properties (all read-only)

URL: NSURL 
duration: NSTimeInterval 
title: String? 
albumTitle: String? 
artist: String?
WKAudioFilePlayerItem

Initializer

init(asset asset: WKAudioFileAsset)

Properties (all read-only)

asset: WKAudioFileAsset
status: WKAudioFilePlayerItemStatus { .Unknown, .ReadyToPlay, .Failed }
error: NSError — non-nil if status is .Failed
currentTime: NSTimeInterval — valid if status is .ReadyToPlay

Notifications

WKAudioFilePlayerDidPlayToEndTimeNotification
WKAudioFilePlayerItemFailedToPlayToEndTimeNotification
Initializer

```swift
convenience init(playerItem item: WKAudioFilePlayerItem)
```

Playing audio

```swift
var rate: Float

0.0  – Stopped
1.0  – Playing at regular speed
-1.0 – Playing at backwards at regular speed
0.5  – Playing at half speed
2.0  – Playing at double speed
```

```swift
func play()
Sets rate to 1.0
```

```swift
func pause()
Sets rate to 0.0
```

WKAudioPlayer also “passes through” the properties status, error, currentTime for its current item, accessed by:

```swift
var currentItem: WKAudioFilePlayerItem?
```
WKAudioFileQueuePlayer

A subclass of WKAudioFilePlayer

Initializer

convenience init(items items: [WKAudioFilePlayerItem])

Managing Items

var items: [WKAudioFilePlayerItem] { get }
func advanceToNextItem()
func appendItem(_ item: WKAudioFilePlayerItem)
func removeItem(_ item: WKAudioFilePlayerItem)
func removeAllItems()}
Keeping the App Open

• Normally, your app will go to sleep when you stop interacting with it.

• To prevent this while audio is playing, add the `UIBackgroundModes` key with the `audio` value to the `Info.plist` file of your watch app.
Recording Audio
func presentAudioRecorderControllerWithURL(_ URL: NSURL,  
    preset preset: WKAudioRecorderPreset,  
    options options: [NSObject : AnyObject]?,  
    completion completion: (Bool, NSError?) -> Void)
File System

- The file system on the watch is structured the same as the file system on the iPhone

- **Library** - storage for non-user-facing data
- **Documents** - storage for user-generated files
- **tmp** - short-term storage that may be purged
Accessing files in the Watch Extension

```swift
let path = NSBundle.mainBundle().pathForResource("sample", ofType: "mp4")!
let url = NSURL(fileURLWithPath: path)
```
File System URLs

Library

NSFileManager.defaultManager().URLsForDirectory(.LibraryDirectory, inDomains:.UserDomainMask).first

Documents

NSFileManager.defaultManager().URLsForDirectory(.DocumentDirectory, inDomains:.UserDomainMask).first

tmp

NSTemporaryDirectory()
Audio Recording

• Apple Watch can record audio as Linear PCM or as AAC.

• Linear PCM is raw, uncompressed sound data. LPCM is stored in .wav files.

• AAC is a lossy format that is more space efficient but has less fidelity. It is stored in .mp4 or .m4a files.

• LPCM vs AAC is analogous to bitmap vs jpeg.
WKAudioRecorderPreset
enum

NarrowBandSpeech
Suitable for voice messages

WideBandSpeech
Higher fidelity voice recording

HighQualityAudio
Suitable for recording music
Options

• A dictionary of options. Some notable options are:

  WKAudioRecorderControllerOptionsAutorecordKey
  True if the controller starts recording automatically, true is the default.

  WKAudioRecorderControllerOptionsActionTitleKey
  The title for the action button; “Save” is the default.
Recording Audio Example

```swift
let directoryURL = NSFileManager.defaultManager().URLsForDirectory(.DocumentDirectory, inDomains:.UserDomainMask).first

let fileURL = NSURL(fileURLWithPath: "audio.wav", isDirectory: false, relativeToURL:directoryURL)

self.presentAudioRecorderControllerWithOutputURL(fileURL, preset: .WideBandSpeech, options: [WKAudioRecorderControllerOptionsAutorecordKey:false, WKAudioRecorderControllerOptionsAlwaysShowActionTitleKey:false]) { (success, error) -> Void in
    print("done")
}
```