Multimedia

Mobile Application Development in iOS

School of EECS

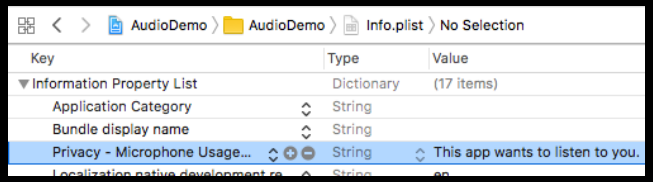
Washington State University

Instructor: Larry Holder

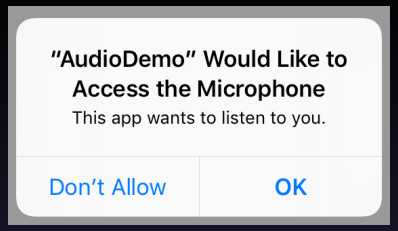
Outline

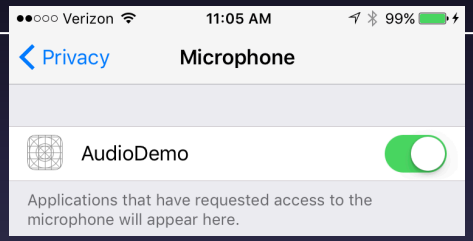
- Audio recording, access, and playback
  - Speech recognition and synthesis
- Image capture, access, and display
- Video recording, access, and playback

Audio Recording and Playback



- AVFoundation framework
- Configure AVAudioSession
  - Need permission to access microphone






```
import AVFoundation
class ViewController: UIViewController {
  var recordingAllowed = false

  override func viewDidLoad() {
    let session = AVAudioSession.sharedInstance()
    session.requestRecordPermission(audioPermissionHandler)
  }
  func audioPermissionHandler (allowed: Bool) {
    self.recordingAllowed = allowed
  }
}
```

Audio Recording and Playback

- Configure AVAudioSession
  - Category (e.g., AVAudioSessionCategoryPlayAndRecord)
  - Mode (e.g., AVAudioSessionModeSpokenAudio) [optional]
  - Options (e.g., defaultToSpeaker) [optional]
- Set active

```
let session = AVAudioSession.sharedInstance()
do {
  try session.setCategory(AVAudioSessionCategoryPlayAndRecord)
  try session.setActive(true)
} catch {
  print("error configuring audio session")
}
```

Audio Recording

- Initialize
  - AVAudioRecorder(url, settings) throws
    - Get URL to sound file in documents directory
    - Settings dictionary: Need at least AVFormatIDKey
- Main methods
  - prepareToRecord(), record(), stop()
- Delegate
  - AVAudioRecorderDelegate.audioRecorderDidFinishRecording

Audio Recording

```
class ViewController: UIViewController, AVAudioRecorderDelegate {
  var audioFileURL: URL!
  var audioRecorder: AVAudioRecorder!

  override func viewDidLoad() {
    // Get URL to audio file
    let paths = FileManager.default.urls(for: .documentDirectory,
                                         in: .userDomainMask)
    let docDir = paths[0]
    self.audioFileURL = docDir.appendingPathComponent("audioFile.m4a")

    // Setup audio recorder
    let settings = [AVFormatIDKey: kAudioFormatMPEG4AAC]
    do {
      self.audioRecorder = try AVAudioRecorder(url: self.audioFileURL,
                                               settings: settings)
      self.audioRecorder.delegate = self
    } catch {
      print("error creating audio recorder")
    }
  }
}
```

Audio Recording

- AVAudioRecorderDelegate method

```
func audioRecorderDidFinishRecording(_ recorder: AVAudioRecorder,
                                     successfully flag: Bool) {
  if !flag {
    print("recording terminated prematurely")
  }
  // Modify view: Change “Stop” to “Start”
  // Reset audio player (audioPlayer defined later...)
  do {
    self.audioPlayer = try AVAudioPlayer(contentsOf: self.audioFileURL)
    self.audioPlayer.delegate = self
    self.audioPlayer.prepareToPlay()
  } catch {
    print("error accessing audio player")
  }
}
```

Audio Playback

- Initialize
  - AVAudioPlayer(url) throws
    - Get URL to sound file in documents directory
- Main methods
  - prepareToPlay(), play(), stop()
- Delegate
  - AVAudioPlayerDelegate.audioPlayerDidFinishPlaying

Audio Playback

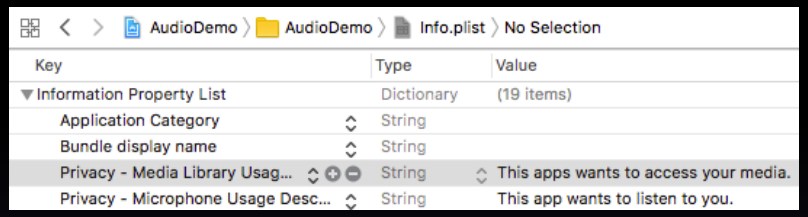
```
class ViewController: UIViewController, AVAudioPlayerDelegate {
  var audioFileURL: URL!
  var audioPlayer: AVAudioPlayer!

  override func viewDidLoad() {
    // Setup audio player
    do {
      self.audioPlayer = try AVAudioPlayer(contentsOf: self.audioFileURL)
      self.audioPlayer.delegate = self
      self.audioPlayer.prepareToPlay()
    } catch {
      print("error accessing audio player")
    }
  }
}
```

Audio Playback

- AVAudioPlayerDelegate method

```
func audioPlayerDidFinishPlaying(_ player: AVAudioPlayer,
                                 successfully flag: Bool) {
  if !flag {
    print("playback terminated prematurely")
  }
  // Modify view: Change “Stop” to “Start”
}
```

Accessing Audio Library



- Maintain privacy and DRM
- MediaPlayer framework
  - MPMediaPickerController to select audio
  - MPMediaPickerDelegate
    - mediaPicker(didPickMediaItems)
    - mediaPickerDidCancel
  - MPMediaPlayerController to play audio (not video)

Accessing Audio Library

```
import MediaPlayer

class ViewController: UIViewController, MPMediaPickerControllerDelegate {
  var mediaPlayer = MPMusicPlayerController()

  @IBAction func selectPlaySongTapped(_ sender: UIButton) {
    let mediaPicker = MPMediaPickerController(mediaTypes: .anyAudio)
    mediaPicker.delegate = self
    present(mediaPicker, animated: true, completion: {})
  }

  func mediaPicker(_ mediaPicker: MPMediaPickerController,
         didPickMediaItems mediaItemCollection: MPMediaItemCollection) {
    self.mediaPlayer.setQueue(with: mediaItemCollection)
    self.mediaPlayer.play()
    mediaPicker.dismiss(animated: true, completion: {})
  }

  func mediaPickerDidCancel(_ mediaPicker: MPMediaPickerController) {
    mediaPicker.dismiss(animated: true, completion: {})
  }
}
```

Speech Recognition: Setup

Key	Type	Value
Information Property List	Dictionary	(18 items)
Application Category	String	
Bundle display name	String	
Privacy - Microphone Usage Desc...	String	This app wants to listen to you.
Privacy - Speech Recognition...	String	This app wants to understand you.

- Speech framework
- SFSpeechRecognizer
  - requestAuthorization()
- SFSpeechRecognizerDelegate
  - speechRecognizer(availabilityDidChange)

Speech Recognition: Setup

```
import Speech
class ViewController: UIViewController, SFSpeechRecognizerDelegate {
  var speechRecognitionAllowed = false
  var speechRecognizer: SFSpeechRecognizer!

  override func viewDidLoad() {
    super.viewDidLoad()
    SFSpeechRecognizer.requestAuthorization(handleAuth)
  }

  func handleAuth (status: SFSpeechRecognizerAuthorizationStatus) {
    switch status {
    case .authorized:
      self.speechRecognitionAllowed = true
      self.speechRecognizer = SFSpeechRecognizer()
      self.speechRecognizer.delegate = self
    default:
      self.speechRecognitionAllowed = false
    }
  }
```

Speech Recognition: Delegate

```
// Delegate method
func speechRecognizer(_ speechRecognizer: SFSpeechRecognizer,
         availabilityDidChange available: Bool) {
  if available {
    self.speechRecognitionAllowed = true
  } else {
    self.speechRecognitionAllowed = false
  }
}
```

Speech Recognition

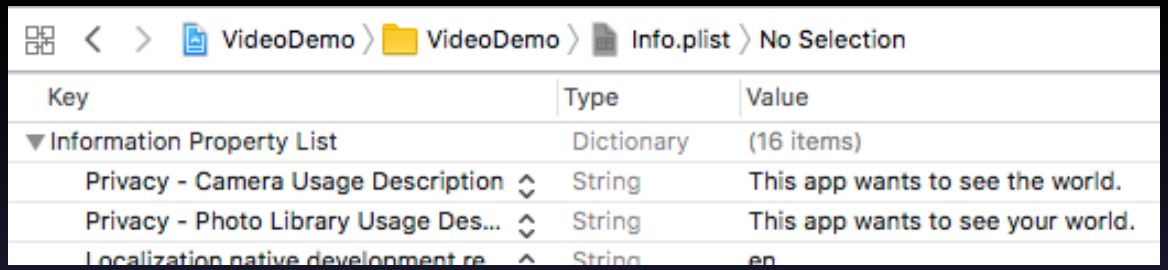
```
func recognizeSpeech () {
  if self.speechRecognitionAllowed {
    let request = SFSpeechURLRecognitionRequest(url: self.audioFileURL)
    request.shouldReportPartialResults = false
    self.speechRecognizer.recognitionTask(with: request,
                                          resultHandler: speechRecognitionHandler)
  }
}

func speechRecognitionHandler (result: SFSpeechRecognitionResult?,
                                error: Error?) {
  if let result = result {
    self.spokenTextView.text = result.bestTranscription.formattedString
  } else {
    print("speech recognition error")
  }
}
```

Speech Synthesis

```
import AVFoundation

let speechSynthesizer = AVSpeechSynthesizer()
let utterance = AVSpeechUtterance(string: “Hello, world”)
speechSynthesizer.speak(utterance)
```

Images and Video

- Add privacy description for access to camera and photos library



- UIImagePickerController
  - Take a picture or video
  - Select a picture or video from library
- AVFoundation framework for lower-level control of image and video assets

UIImagePickerController: Properties

- allowsEditing
- sourceType: .photoLibrary, .savedPhotosAlbum, .camera
- mediaTypes
  - kUTTypeImage as String, kUTTypeMovie as String
    - import MobileCoreServices
  - UIImagePickerController.availableMediaTypes(for)

UIImagePickerController: Delegates

- UIImagePickerControllerDelegate
  - imagePickerController(didFinishPickingMediaWithInfo info)
    - info[UIImagePickerControllerOriginalImage]
    - info[UIImagePickerControllerEditedImage]
  - imagePickerControllerDidCancel
- UINavigationControllerDelegate
  - Required, but not used

Select Image from Photo Library

```
import UIKit
import MobileCoreServices // to get kUTTypeImage

class ViewController: UIViewController,
    UIImagePickerControllerDelegate, UINavigationControllerDelegate {

  var selectedImage: UIImage!

  func selectImage () {
    if UIImagePickerController.isSourceTypeAvailable(.photoLibrary) {
      let picker = UIImagePickerController()
      picker.delegate = self
      picker.allowsEditing = false
      picker.sourceType = .photoLibrary // or see below
      picker.mediaTypes = [kUTTypeImage as String]
      self.present(picker, animated: true, completion: nil)
   } else {
      print("photo library not available")
   }
}

picker.mediaTypes = UIImagePickerController.availableMediaTypes(for: .photoLibrary)!
```

Select Image from Photo Library

```
// UIImagePickerControllerDelegate methods

func imagePickerController(_ picker: UIImagePickerController,
 didFinishPickingMediaWithInfo info: [String : Any]) {
  self.selectedImage = info[UIImagePickerControllerOriginalImage] as! UIImage
  self.imageView.image = self.selectedImage // contentMode = .scaleAspectFit
  picker.dismiss(animated: true, completion: nil)
}

func imagePickerControllerDidCancel(_ picker: UIImagePickerController) {
  picker.dismiss(animated: true, completion: nil)
}
```

Take Picture

```
func takePicture () { // same as selectImage, except use sourceType .camera
  if UIImagePickerController.isSourceTypeAvailable(.camera) {
    let picker = UIImagePickerController()
    picker.delegate = self
    picker.allowsEditing = false
    picker.sourceType = .camera
    picker.mediaTypes = [kUTTypeImage as String]
    self.present(picker, animated: true, completion: nil)
  } else {
    print("camera not available”)
  }
}
```

Save Picture

- UIImageWriteToSavedPhotosAlbum(image, completionTarget, completionSelector, contextInfo)
  - completionTarget = self
  - completionSelector = #selector(imageWriteHandler)
- func imageWriteHandler(_ image: UIImage, didFinishSavingWithError error: Error?, contextInfo: UnsafeRawPointer?)

Save Picture

```
func savePicture () {
  UIImageWriteToSavedPhotosAlbum(self.selectedImage, self,
    #selector(imageWriteHandler), nil)
}

func imageWriteHandler(_ image: UIImage, didFinishSavingWithError error: Error?,
                       contextInfo: UnsafeRawPointer?) {
  if let err = error {
    print("error saving image: \(err.localizedDescription)")
  } else {
    print("image saved to camera roll")
  }
}
```

Working with Video

- Selecting video from library similar to images
  - Still use UIImagePickerController
  - Use mediaType kUTTypeMovie
- Delegate method gets URL to video
- Play video using AVPlayer and AVPlayerViewController
- Recording video similar to images
  - Need to authorize use of microphone
    - See earlier slides
- Saving video similar to images

Select Video from Library

```
import UIKit
import MobileCoreServices // to get kUTTypeMovie

class ViewController: UIViewController,
    UIImagePickerControllerDelegate, UINavigationControllerDelegate {

  var videoURL: URL!

  func selectVideo () {
    if UIImagePickerController.isSourceTypeAvailable(.photoLibrary) {
      let picker = UIImagePickerController()
      picker.delegate = self
      picker.allowsEditing = false
      picker.sourceType = .photoLibrary
      picker.mediaTypes = [kUTTypeMovie as String]
      self.present(picker, animated: true, completion: nil)
   } else {
      print("video library not available")
   }
}
```

Select Video from Library

```
// UIImagePickerControllerDelegate methods

func imagePickerController(_ picker: UIImagePickerController,
 didFinishPickingMediaWithInfo info: [String : Any]) {
  self.videoURL = info[UIImagePickerControllerMediaURL] as! URL
  picker.dismiss(animated: true, completion: nil)
}

func imagePickerControllerDidCancel(_ picker: UIImagePickerController) {
  picker.dismiss(animated: true, completion: nil)
}
```

Play Video

```
import AVKit // for AVPlayerViewController
import AVFoundation // for AVPlayer

func playVideo () {
  if let url = self.videoURL {
    let playerView = AVPlayerViewController()
    playerView.player = AVPlayer(url: url)
    present(playerView, animated: true, completion: nil)
  }
}
```

Record Video

```
func recordVideo () { // same as selectVideo, except use sourceType .camera
  if UIImagePickerController.isSourceTypeAvailable(.camera) {
    let picker = UIImagePickerController()
    picker.delegate = self
    picker.allowsEditing = false
    picker.sourceType = .camera
    picker.mediaTypes = [kUTTypeMovie as String]
    self.present(picker, animated: true, completion: nil)
  } else {
    print("camera not available”)
  }
}
```

Save Video

- UISaveVideoAtPathToSavedPhotosAlbum(videoPath, completionTarget, completionSelector, contextInfo)
  - completionTarget = self
  - completionSelector = #selector(videoWriteHandler)
- func videoWriteHandler(_ videoPath: String, didFinishSavingWithError error: Error?, contextInfo: UnsafeRawPointer?)

Save Video

```
func saveVideo () {
  if let videoPath = self.videoURL?.path {
    UISaveVideoAtPathToSavedPhotosAlbum(videoPath, self,
      #selector(videoWriteHandler), nil)
  }
}

func videoWriteHandler(_ videoPath: String,
    didFinishSavingWithError error: Error?,
                        contextInfo: UnsafeRawPointer?) {
  if let err = error {
    print("error saving video: \(err.localizedDescription)")
  } else {
    print("video saved to library")
  }
}
```

Resources

- AVFoundation and AVKit
  - developer.apple.com/av-foundation/
  - developer.apple.com/reference/avkit
- MediaPlayer framework
  - developer.apple.com/reference/mediaplayer
- Speech framework
  - developer.apple.com/reference/speech
- UIImagePickerController
  - developer.apple.com/reference/uikit/uiimagepickercontroller