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

• Use AVFoundation framework

• Configure AVAudioSession singleton class
  – Need permission to access microphone
    • AVAudioSession.recordPermission == .granted
    • AVAudioSession.requestRecordPermission((Bool -> Void))

```swift
import AVFoundation
class ViewController: UIViewController {

    override func viewDidLoad() {
        let session = AVAudioSession.sharedInstance()
        if session.recordPermission != .granted {
            session.requestRecordPermission({ Bool in })
        }
    }
}
```

“AudioDemo” Would Like to Access the Microphone
This app wants to listen to you.

Don’t Allow OK
Audio Recording and Playback

• Configure AVAudioSession
  – setCategory(category, mode, options)
  – Category (e.g., AVAudioSession.Category.playAndRecord)
  – Mode (e.g., AVAudioSession.Mode.spokenAudio)
  – Options (e.g., mixWithOthers, duckOthers, defaultToSpeaker)

• setActive(bool)
  – Request access to audio hardware
  – May fail if higher-priority task using audio
Audio Recording and Playback

• Configure and activate audio session

```swift
let session = AVAudioSession.sharedInstance()
do {
    try session.setCategory(AVAudioSession.Category.playAndRecord, mode: AVAudioSession.Mode.spokenAudio, options: [AVAudioSession.CategoryOptions.duckOthers, AVAudioSession.CategoryOptions.defaultToSpeaker])
    try session.setActive(true)
} catch {
    print("error starting audio session")
}
```
Audio Recording

• Initialize
  – AVAudioRecorder(url, settings) throws
    • Get URL to sound file in documents directory
    • Settings dictionary: Need at least AVFormatIDKey
        6-audio_data_format_identifiers

• Main methods
  – prepareToRecord(), record(), pause(), stop()

• AVAudioRecorderDelegate methods
  – audioRecorderDidFinishRecording
Audio Recording: Setup

class ViewController: UIViewController, AVAudioRecorderDelegate {
    let audioFile = "audioFile.m4a"
    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]
        audioFileURL = docDir.appendingPathComponent(audioFile)

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

- AVAudioRecorderDelegate method

```swift
func startRecording() {
    audioRecorder?.record()
}

func stopRecording() {
    audioRecorder?.stop()
}

func audioRecorderDidFinishRecording(_ recorder: AVAudioRecorder, successfully flag: Bool) {
    if flag {
        print("recording successful")
    } else {
        print("recording failed")
    }
}

// Modify view: Change "Stop" to "Start"
```
Audio Playback

• Initialize
  – AVAudioPlayer(\text{url}) \text{ throws}
    • Get URL to sound file in documents directory
    • Must be reinitialized when sound file rewritten

• Main methods
  – prepareToPlay(), play(), pause(), stop()
  – currentTime – set to 0 to return to front

• AVAudioPlayerDelegate methods
  – audioPlayerDidFinishPlaying
class ViewController: UIViewController, AVAudioPlayerDelegate {
    var audioFileURL: URL!
    var audioPlayer: AVAudioPlayer?

    func startPlaying() {
        // Setup audio player
        do {
            audioPlayer = try AVAudioPlayer(contentsOf: audioFileURL)
            audioPlayer?.delegate = self
        } catch {
            print("error accessing audio player")
        }
        audioPlayer?.play()
    }

    func stopPlaying() {
        audioPlayer?.stop()
    }
}
Audio Playback

• **AVAudioPlayerDelegate** method

```swift
func audioPlayerDidFinishPlaying(_ player: AVAudioPlayer, successfully flag: Bool) {
    if flag {
        print("playback finished")
    } else {
        print("playback error")
    }
    // Modify view: Change "Stop" to "Start"
}
```
Accessing Audio Library

- Maintain privacy and DRM
- MediaPlayer framework
  - MPMediaPickerController to select audio
  - MPMediaPickerControllerDelegate
    - mediaPicker(didPickMediaItems)
    - mediaPickerDidCancel
  - MPMediaPlayerController.applicationQueuePlayer
    - setQueue(mediaItemCollection)
    - play(), pause(), stop()

Note: No music library in iOS simulator. Need a real device.
import MediaPlayer

class ViewController: UIViewController, MPMediaPickerControllerDelegate {

    var mediaPlayer = MPMusicPlayerController().applicationQueuePlayer

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

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

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

• Speech framework

• SFSpeechRecognizer
  – requestAuthorization()

• SFSpeechRecognizerDelegate
  – speechRecognizer(availabilityDidChange)
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:
            speechRecognitionAllowed = true
            speechRecognizer = SFSpeechRecognizer()
            speechRecognizer?.delegate = self
        case .default:
            speechRecognitionAllowed = false
        }
    }
}
Speech Recognition: Delegate

```swift
// Delegate method
func speechRecognizer(_ speechRecognizer: SFSpeechRecognizer,
    availabilityDidChange available: Bool) {
    if available {
        speechRecognitionAllowed = true
    } else {
        speechRecognitionAllowed = false
    }
}
```
var spokenText: String = "Hello, world."

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

func speechRecognitionHandler (result: SFSpeechRecognitionResult?, 
    error: Error?) {
    if let res = result {
        self.spokenText = res.bestTranscription.formattedString
        print("spoke: \(self.spokenText)")
    } else {
        print("speech recognition error")
    }
}
Speech Synthesis

import AVFoundation

func synthesizeSpeech() {
    let speechSynthesizer = AVSpeechSynthesizer()
    let utterance = AVSpeechUtterance(string: self.spokenText)
    // utterance.voice = AVSpeechSynthesisVoice(identifier: "...")
    speechSynthesizer.speak(utterance)
}
Images and Video

• Add privacy properties for access to camera and photo library
  – Request authorization

  ![Privacy Property List](image)

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Privacy - Photo Library Additions Usage Description</td>
<td>String</td>
<td>This app wants to save the world.</td>
</tr>
<tr>
<td>Privacy - Photo Library Usage Description</td>
<td>String</td>
<td>This app wants to see your world.</td>
</tr>
<tr>
<td>Privacy - Camera Usage Description</td>
<td>String</td>
<td>This app wants to see the world.</td>
</tr>
</tbody>
</table>

• UIImagePickerController
  – Take a picture or video
  – Select a picture or video from library

• AVFoundation framework
  – Lower-level control of image and video assets

Note: iOS simulator cannot access Mac camera. Need real device for testing.

Can drag-and-drop images and videos into Photos app on iOS simulator.
UIImagePickerController: Properties

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

• UIWindowPickerControllerDelegate
  – imagePickerController(didFinishPickingMediaWithInfo info)
    • info[UIImagePickerController.originalImage]
    • info[UIImagePickerController.editedImage]
  – imagePickerControllerDidCancel

• UINavigationControllerDelegate
  – Required, but used implicitly
import UIKit
import MobileCoreServices // to get kUTTypeImage

class ViewController: UIViewController, UIImagePickerControllerDelegate, UINavigationControllerDelegate {

    var selectedImage: UIImage?

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

func imagePickerController(_ picker: UIImagePickerController, didFinishPickingMediaWithInfo info: [UIImagePickerController.InfoKey: Any]) {
    selectedImage = info[.originalImage] as? UIImage
    imageView.image = selectedImage
    picker.dismiss(animated: true, completion: nil)
}

func imagePickerControllerDidCancel(_ picker: UIImagePickerController) {
    picker.dismiss(animated: true, completion: nil)
}
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")
    }
}
func writeImage(_ image: UIImage, to fileName: String) {
    if let directoryURL = FileManager.default.urls(for: .documentDirectory, in: .userDomainMask).first {
        let fileURL = directoryURL.appendingPathComponent(fileName)
        // Can also use image.pngData, but rotates image
        if let imageData = image.jpegData(compressionQuality: 1.0) {
            do {
                try imageData.write(to: fileURL)
            } catch {
                print("\(error)"
            }
        } else {
            print("Unable to convert image to jpeg.")
        }
    } else {
        print("Error accessing document directory.")
    }
}
func readImage(from fileName: String) -> UIImage? {
    if let directoryURL = FileManager.default.urls(for: .documentDirectory, in: .userDomainMask).first {
        let fileURL = directoryURL.appendingPathComponent(fileName)
        do {
            let imageData = try Data(contentsOf: fileURL)
            let image = UIImage(data: imageData)
            return image
        } catch {
            print("\(error)"
        }
    } else {
        print("Error accessing document directory.")
    }
    return nil
}
Save Image to Photo Library


- `func imageWriteHandler(image: UIImage, didFinishSavingWithError error: Error?, contextInfo: UnsafeRawPointer?)`
func saveImage () {
    if let image = selectedImage {
        UIImageWriteToSavedPhotosAlbum(image, self, #selector(imageWriteHandler), nil)
    }
}

@objc 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
  – Authorize use of microphone

• Saving video similar to images
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")
        }
    }
}
// UIImagePickerControllerDelegate methods

func imagePickerController(_ picker: UIImagePickerController, didFinishPickingMediaWithInfo info: [UIImagePickerController.InfoKey: Any]) {
    videoURL = info[.mediaURL] as? URL
    picker.dismiss(animated: true, completion: nil)
}

func imagePickerControllerDidCancel(_ picker: UIImagePickerController) {
    picker.dismiss(animated: true, completion: nil)
}
import AVKit // for AVPlayerViewController
import AVFoundation // for AVPlayer

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

@IBOutlet weak var videoView: UIView!

func playVideoInSubView () {
  if let url = videoURL {
    let playerView = AVPlayerViewController()
    playerView.player = AVPlayer(url: url)
    playerView.view.frame = videoView.frame
    self.addChild(playerView) // initialize player view
    self.view.addSubview(playerView.view) // display player view
  }
}
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")
    }
}
let videoFileName = "myVideo.mov" // .mov extension important

func writeVideo(_ videoURL: URL?, to fileName: String) {
    if let directoryURL = FileManager.default.urls(for: .documentDirectory, in: .userDomainMask).first {
        let fileURL = directoryURL.appendingPathComponent(fileName)
        if let vidURL = videoURL {
            do {
                let videoData = try Data(contentsOf: vidURL)
                try videoData.write(to: fileURL)
            } catch {
                print("\(error)"
            }
        } else {
            print("Error accessing document directory.")
        }
    }
}
// Just return URL to video file
func getVideoFileURL(from fileName: String) -> URL? {
    if let directoryURL = FileManager.default.urls(for: .documentDirectory, in: .userDomainMask).first {
        let fileURL = directoryURL.appendingPathComponent(fileName)
        return fileURL
    } else {
        print("Error accessing document directory.\n")
    }
    return nil
}
Save Video to Library

- `UISaveVideoAtPathToSavedPhotosAlbum(videoPath: String, completionTarget: Any?, completionSelector: Selector?, contextInfo: UnsafeRawPointer?) // nil`

- `func videoWriteHandler(videoPath: String, didFinishSavingWithError error: Error?, contextInfo: UnsafeRawPointer?)`
func saveVideoToLibrary() {
    if let videoPath = videoURL?.path {
        UISaveVideoAtPathToSavedPhotosAlbum(videoPath, self, #selector(videoWriteHandler), nil)
    }
}

@objc 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 (more media tools here)
  – developer.apple.com/avfoundation/
  – developer.apple.com/documentation/avkit

• MediaPlayer framework
  – developer.apple.com/documentation/documentation/mediaplayer

• Speech framework
  – developer.apple.com/documentation/documentation/speech

• UIImagePickerController
  – developer.apple.com/documentation/documentation/uikit/uimagepickercontroller