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 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 })
        }
    }
}
```
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(Boolean)
  – 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`
- Main methods
  - `prepareToRecord()`, `record()`, `pause()`, `stop()`
- `AVAudioRecorderDelegate` methods
  - `audioRecorderDidFinishRecording`
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]
        audioFileURL = docDir.appendingPathComponent("audioFile.m4a")

        // 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 audioRecorderDidFinishRecording(_ recorder: AVAudioRecorder, successfully flag: Bool) {
    if !flag {
        print("recording terminated prematurely")
    }
    // Setup new audio player
    do {
        audioPlayer = try AVAudioPlayer(contentsOf: audioFileURL)
        audioPlayer?.delegate = self
    } catch {
        print("error accessing audio player")
    }
    // Modify view: Change "Stop" to "Start"
}
```
Audio Playback

• Initialize
  – AVAudioPlayer(url) throws
    • Get URL to sound file in documents directory

• 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?

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

- AVAudioPlayerDelegate method

```swift
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**
  - `MPMediaPlayerPickerController` to select audio
  - `MPMediaPlayerDelegate`
    - `mediaPicker(didPickMediaItems)`
    - `mediaPickerDidCancel`
  - `MPMediaPlayerController.applicationQueuePlayer`
    - `setQueue(medialItemCollection)`
    - `play()`, `pause()`, `stop()`

Note: Can drag-and-drop images and videos into iOS simulator.
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)
        mediaPlayer.play()
        mediaPicker.dismiss(animated: true, completion: {})
    }

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

Mobile Application Development in iOS
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(
                delegate = self
        default:
            speechRecognitionAllowed = false
        }
    }
}
Speech Recognition: Delegate

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

```swift
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")
    }
}
```
import AVFoundation

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

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information Property List</td>
<td>Dictionary</td>
<td>(17 items)</td>
</tr>
<tr>
<td>Privacy - Camera Usage Description</td>
<td>String</td>
<td>This app wants to see 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 - Photo Library Additions Usage Description</td>
<td>String</td>
<td>This app wants to save 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.
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[UIImagePickerController_originalImage]
    • info[UIImagePickerControllerEditedImage]
  – imagePickerControllerDidCancel

• UINavigationControllerDelegate
  – Required, but used implicitly
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
            picker.mediaTypes = [kUTTypeImage as String]
            self.present(picker, animated: true, completion: nil)
        } else {
            print("photo library not available")
        }
    }
}
Select Image from Photo Library

// 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")
    }
}
Save Picture


- `func imageWriteHandler(image: UIImage, didFinishSavingWithError error: Error?, contextInfo: UnsafeRawPointer?)`
func savePicture () {
    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")
        }
    }
}
Select Video from Library

// 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 playVideo () {
    if let url = videoURL {
        let playerView = AVPlayerViewController()
        playerView.player = AVPlayer(url: url)
        present(playerView, animated: true, completion: nil)
    }
}
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: String`,
  - `completionTarget: Any?`, // self
  - `completionSelector: Selector?`, // #selector(videoWriteHandler)
  - `contextInfo: UnsafeRawPointer?`) // nil

- **func videoWriteHandler**
  - `videoPath: String`,
  - `didFinishSavingWithError error: Error?`,
  - `contextInfo: UnsafeRawPointer?`)
func saveVideo () {
    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/mediaplayer

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

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