Multimedia

Mobile Application Development in iOS

School of EECS

Washington State University

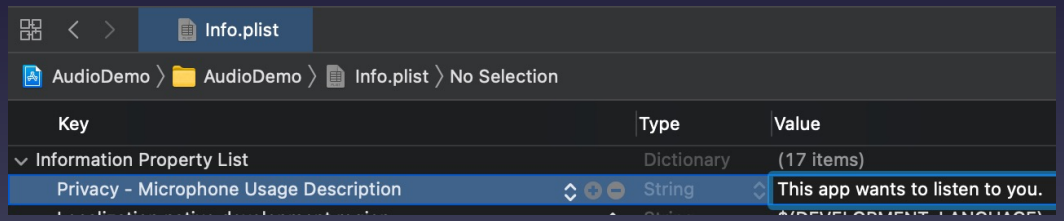
Instructor: Larry Holder

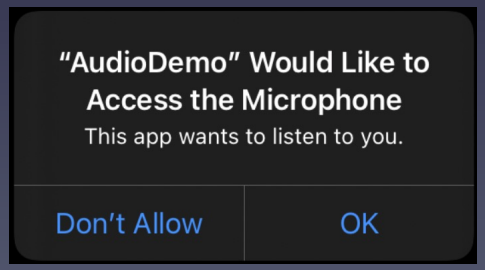
Outline

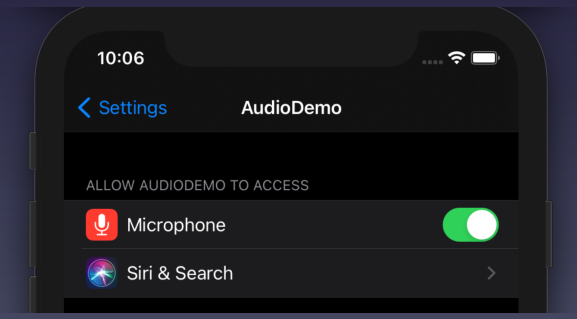
- Audio recording, access, and playback
- Image capture, access, and display
- Video recording, access, and playback

Audio Recording and Playback

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







Audio Recording and Playback

```
import AVFoundation

class ViewController: UIViewController {

    override func viewDidLoad() {
        let session = AVAudioSession.sharedInstance()
        if session.recordPermission != .granted {
            session.requestRecordPermission({permission in
                if permission {
                    self.initializeAudioSession()
                } else {
                    print("permission denied")
                }
            })
        } else {
            initializeAudioSession()
        }
    }
}
```

Audio Recording and Playback

- Configure AVAudioSession
  - setCategory(category, mode, options) throws
  - Category (e.g., AVAudioSession.Category.playAndRecord)
  - Mode (e.g., AVAudioSession.Mode.spokenAudio)
  - Options (e.g., mixWithOthers, duckOthers, defaultToSpeaker)
- setActive(Bool) throws
  - Request access to audio hardware
  - May fail if higher-priority task using audio

Audio Recording and Playback

- Configure and activate audio session

```
func initializeAudioSession() {
    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
      - https://developer.apple.com/documentation/coreaudio/core_audio_data_types/1572096-audio_data_format_identifiers
- Main methods
  - prepareToRecord(), record(), pause(), stop()
- AVAudioRecorderDelegate methods
  - audioRecorderDidFinishRecording

Note: iOS simulator can access Mac's microphone.

Audio Recording: Setup

```
class ViewController: UIViewController, AVAudioRecorderDelegate {
    let audioFile = "audioFile.m4a"
    var audioFileURL: URL!
    var audioRecorder: AVAudioRecorder?

    func initializeAudioRecorder() {
        // Get URL to audio file
        if let path = FileManager.default.urls(for: .documentDirectory,
                                               in: .userDomainMask).first {
            audioFileURL = path.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

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

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

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

Audio Playback

- Initialize
  - AVAudioPlayer(url) 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

Audio Playback: Setup

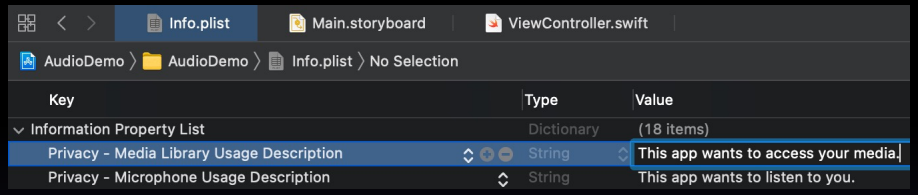
```
class ViewController: UIViewController, AVAudioPlayerDelegate {
    var audioFileURL: URL!
    var audioPlayer: AVAudioPlayer?

    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

```
func audioPlayerDidFinishPlaying(_ player: AVAudioPlayer,
                                 successfully flag: Bool) {
    if flag {
        print("playback finished")
    } else {
        print("playback error")
    }
    // Modify view: Change "Stop" to "Play"
    playStopButton.setTitle("Play", for: .normal)
    playing = false
}
```

Accessing Audio Library



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

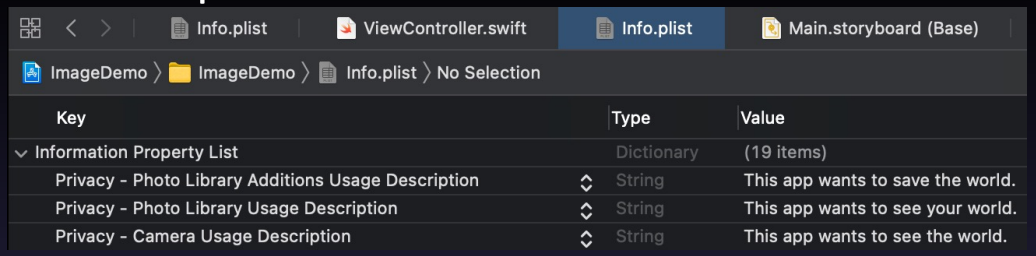
Note: No music library in iOS simulator. Need a real device.

Accessing Audio Library

```
import MediaPlayer

class ViewController: UIViewController, MPMediaPickerControllerDelegate {
    var mediaPlayer = MPMusicPlayerController.applicationQueuePlayer

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

    func mediaPicker(_ mediaPicker: MPMediaPickerController,
            didPickMediaItems mediaItemCollection: MPMediaItemCollection) {
        mediaPicker.dismiss(animated: true, completion: {})
        songLabel.text = mediaItemCollection.items[0].title
        mediaPlayer.setQueue(with: mediaItemCollection)
    }

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

Images and Video

- Add privacy properties for access to camera and photo library
  - Authorization requested on first access



- 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

- allowsEditing
- sourceType: .photoLibrary, .camera
- isSourceTypeAvailable(sourceType)

UIImagePickerControllerDelegate

- imagePickerController(didFinishPickingMediaWithInfo info)
  - info[UIImagePickerController.originalImage]
  - info[UIImagePickerController.editedImage]
- imagePickerControllerDidCancel
- UINavigationControllerDelegate
  - Required, but used implicitly

Select Image from Photo Library

```
import UIKit

class ViewController: UIViewController,
        UIImagePickerControllerDelegate, UINavigationControllerDelegate {

    func selectImageFromLibrary () {
        if UIImagePickerController.isSourceTypeAvailable(.photoLibrary) {
            let picker = UIImagePickerController()
            picker.delegate = self
            picker.allowsEditing = false
            picker.sourceType = .photoLibrary
            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]) {
    picker.dismiss(animated: true, completion: nil)
    if let image = info[.originalImage] as? UIImage {
        imageView.image = image
    }
}

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
        self.present(picker, animated: true, completion: nil)
    } else {
        print("camera not available")
    }
}
```

Save Image to File

```
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.")
    }
}
```

Read Image from File

```
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

- UIImageWriteToSavedPhotosAlbum(image: UIImage,

  completionTarget: Any?, // self

  completionSelector: Selector?, // #selector(imageWriteHandler)

  contextInfo: UnsafeRawPointer?) // nil

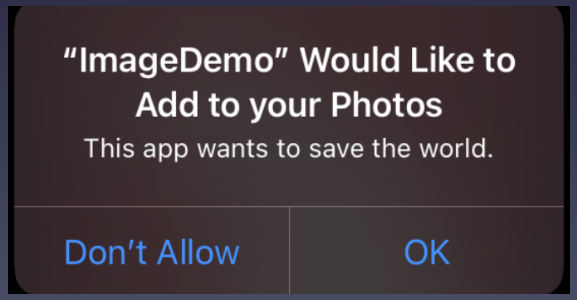
- func imageWriteHandler(image: UIImage,

  didFinishSavingWithError error: Error?,

  contextInfo: UnsafeRawPointer?)

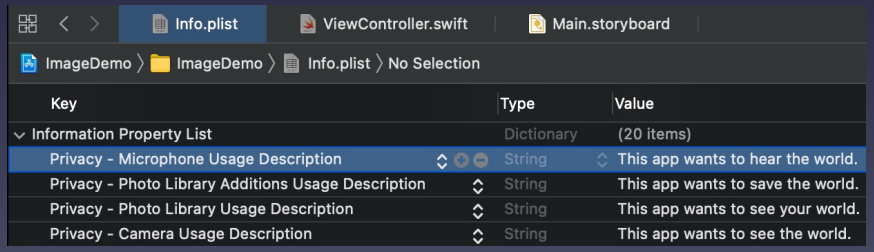
Save Image to Photo Library

```
func saveImageToPhotoLibrary(_ image: UIImage) {
    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 photo library")
    }
}
```



Working with Video

- Selecting video from library similar to images
  - Still use UIImagePickerController
  - Set picker.mediaType = ["public.movie"]
- 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



Select Video from Library

```
class ViewController: UIViewController,
        UIImagePickerControllerDelegate, UINavigationControllerDelegate {

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

Select Video from Library

```
// UIImagePickerControllerDelegate methods

var videoURL: URL?

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

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 = 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 = ["public.movie"]
        self.present(picker, animated: true, completion: nil)
    } else {
        print("camera not available")
    }
}
```

Write Video to File

```
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.")
    }
}
```

Read Video from File

```
// 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.")
    }
    return nil
}
```

Save Video to Library

- UISaveVideoAtPathToSavedPhotosAlbum(videoPath: String,

  completionTarget: Any?, // self

  completionSelector: Selector?, // #selector(videoWriteHandler)

  contextInfo: UnsafeRawPointer?) // nil

- func videoWriteHandler(videoPath: String,

  didFinishSavingWithError error: Error?,

  contextInfo: UnsafeRawPointer?)

Save Video to Library

```
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/mediaplayer
- UIImagePickerController
  - developer.apple.com/documentation/uikit/uiimagepickercontroller
- PhotoKit (iOS 14+)
  - developer.apple.com/documentation/photokit