What’s New in Core Audio

Session 507

Akshatha Nagesh ‘AudioEngine’er
Torrey Holbrook Walker Senior New Feature Salesperson
Agenda

AVAudioEngine
- Recap
- What’s New

Inter-device Audio

AVAudioSession
- What’s New

Audio Unit Extensions
AVAudioEngine

Recap
Core Audio Stack
iOS and OS X

AVAudioEngine
- Introduced in iOS 8.0/OS X 10.10
- Refer to WWDC14 session 502

AVAudioEngine in Practice
AVAudioEngine

Goals

Provide powerful, feature-rich API set
Achieve simple as well as complex tasks
Simplify real-time audio
AVAudioEngine

Features

Objective-C / Swift API set
Low latency, real-time audio
Play and record audio
Connect audio processing blocks
Capture audio at any point in the processing chain
Implement 3D audio for games
AVAudioEngine

Building blocks

Node-AVAudioNode

• Source nodes: Provide data for rendering
• Processing nodes: Process data
• Destination node: Terminating node connected to output hardware
Sample Engine Setup

Karaoke

InputNode → EffectNode (Delay) → MixerNode → OutputNode
InputNode → PlayerNode (Backing Track) → MixerNode
InputNode → PlayerNode (Sound Effects) → MixerNode
InputNode → NodeTapBlock (Analyze)
Sample Engine Setup

Karaoke

Source nodes:

1. InputNode
2. EffectNode (Delay)
3. PlayerNode (Backtrack)
4. NodeTapBlock (Analyze)
5. PlayerNode (Sound Effects)
6. MixerNode
7. OutputNode
Sample Engine Setup
Karaoke

InputNode -> EffectNode (Delay) -> PlayerNode (Backing Track) -> MixerNode -> OutputNode

NodeTapBlock (Analyze) -> PlayerNode (Sound Effects) -> MixerNode
Sample Engine Setup
Karaoke

InputNode → EffectNode (Delay) → PlayerNode (Backin Track) → MixerNode → OutputNode

NodeTapBlock (Analyze) → PlayerNode (Sound Effects) → MixerNode

Destination node
Mixer Nodes

AVAudioMixerNode
• Performs sample rate conversion, up/down mixing of channels
• Supports mono, stereo and multichannel inputs

AVAudioEnvironmentNode
• Simulates a 3D space (listener is implicit)
• Supports mono and stereo inputs
• Spatializes only mono inputs
AVAudioMixing Protocol

Defines properties for a mixer input bus

Source nodes conform to this protocol

- Properties cached when not connected to a mixer
- Properties applied on connection to a mixer
AVAudioMixing Protocol

Properties

Common (all mixer nodes)
- volume - `player.volume = 0.5`

Stereo (AVAudioMixerNode)
- pan - `player.pan = -1.0`

3D (AVAudioEnvironmentNode)
- position - `player.position = AVAudioMake3DPoint(-2.0, 0.0, 5.0)`
- obstruction, occlusion, renderingAlgorithm and more…
AVAudioMixing Protocol

Sample setup

- PlayerNode
- EnvironmentNode
- MixerNode (Main Mixer)
AVAudioMixing Protocol

Sample setup

properties are cached

Mixing Settings
volume = 0.5
pan = -1.0
position = (-1, 2, -5)

PlayerNode

EnvironmentNode

MixerNode

MixerNode (Main Mixer)
AVAudioMixing Protocol

Sample setup

PlayerNode

Mixing Settings
- volume = 0.5
- pan = -1.0
- position = (-1, 2, -5)

EnvironmentNode

MixerNode

MixerNode (Main Mixer)

volume and pan take effect
AVAudioMixing Protocol

Sample setup

PlayerNode

Mixing Settings
- volume = 0.5
- pan = -1.0
- position = (-1, 2, -5)

EnvironmentNode

MixerNode (Main Mixer)

volume and position take effect
Handling Multichannel Audio

Hardware setup

- OS X
- iOS

AVAudioEngine setup
Handling Multichannel Audio

Hardware setup: OS X

User can configure the hardware through Audio MIDI Setup.
Handling Multichannel Audio

Hardware setup: iOS

Configure AVAudioSession

- Playback use case:
  - Activate audio session
  - Check maximumOutputNumberOfChannels
  - Set preferredOutputNumberOfChannels
  - Verify actual outputNumberOfChannels
Handling Multichannel Audio

Hardware setup: iOS-code example

// example: audio playback use case
do {
    let desiredNumChannels = 6 // for 5.1 rendering

    let audioSession = AVAudioSession.sharedInstance()
    let category = AVAudioSessionCategoryPlayback
    try audioSession.setCategory(category)
    try audioSession.setActive(true)

    // check maximum available output number of channels
    let maxChannels = audioSession.maximumOutputNumberOfChannels
Handling Multichannel Audio
Hardware setup: iOS-code example

```swift
// example: audio playback use case
do {
    let desiredNumChannels = 6 // for 5.1 rendering

    let audioSession = AVAudioSession.sharedInstance()
    let category = AVAudioSessionCategoryPlayback
    try audioSession.setCategory(category)
    try audioSession.setActive(true)

    // check maximum available output number of channels
    let maxChannels = audioSession.maximumOutputNumberOfChannels
}
```
// example: audio playback use case
do {
    let desiredNumChannels = 6 // for 5.1 rendering

    let audioSession = AVAudioSession.sharedInstance()
    let category = AVAudioSessionCategoryPlayback
    try audioSession.setCategory(category)
    try audioSession.setActive(true)

    // check maximum available output number of channels
    let maxChannels = audioSession.maximumOutputNumberOfChannels
if maxChannels >= desiredNumChannels {
    // set preferred number of output channels
    try
        audioSession.setPreferredOutputNumberOfChannels(desiredNumChannels)
}
let actualChannelCount = audioSession.outputNumberOfChannels

    // adapt to the actual number of output channels
    ..
} catch {
    // handle errors
}
if maxChannels >= desiredNumChannels {
    // set preferred number of output channels
    try
        audioSession.setPreferredOutputNumberOfChannels(desiredNumChannels)
} 
let actualChannelCount = audioSession.outputNumberOfChannels

// adapt to the actual number of output channels
.. 
} catch {
    // handle errors
}
Handling Multichannel Audio

AVAudioEngine setup: iOS / OS X

Multichannel content
- AVAudioMixerNode

Spatialized content (games)
- AVAudioEnvironmentNode
Handling Multichannel Audio

AVAudioEngine setup: multichannel content
Handling Multichannel Audio

AVAudioEngine setup: multichannel content
Handling Multichannel Audio

AVAudioEngine setup: multichannel content

```swift
// get output hardware format
let output = engine.outputNode
let outputHWFormat = output.outputFormatForBus(0)

// connect mixer to output
let mainMixer = engine.mainMixerNode
engine.connect(mainMixer, to: output, format: outputHWFormat)
```
// get output hardware format
let output = engine.outputNode
let outputHWFormat = output.outputFormatForBus(0)

// connect mixer to output
let mainMixer = engine.mainMixerNode
engine.connect(mainMixer, to: output, format: outputHWFormat)
Handling Multichannel Audio

AVAUDIOEngine setup: multichannel content

Multichannel Content (5.1) → PlayerNode → MixerNode → OutputNode → Multichannel Hardware (5.1)

- get content format
- set connection format
do {
  // open multichannel file for reading
  let mcFile = try AVAudioFile(forReading: fileURL)

  // connect player to mixer
  engine.connect(player, to: mainMixer, format: mcFile.processingFormat)
} catch { .. }
// schedule file on player
// start engine
// start player
Handling Multichannel Audio

AVAudioEngine setup: spatialized content
Handling Multichannel Audio

AVAudioEngine setup: spatialized content

Mono Content → PlayerNode → Environment Node → OutputNode → Multichannel Hardware (5.1)

set compatible format
get hardware format
Handling Multichannel Audio

AVAudioEngine setup: spatialized content

// map output hardware channel layout to a compatible layout
let layoutTag = kAudioChannelLayoutTag_AudioUnit_5_0 // for 5.1 HW layout
let layout = AVAudioChannelLayout(layoutTag: layoutTag)
let envOutputFormat = AVAudioFormat(
    standardFormatWithSampleRate: outputHWFormat.sampleRate,
    channelLayout: layout);

engine.connect(envNode, to: output, format: envOutputFormat)
Handling Multichannel Audio

AVAudioEngine setup: spatialized content

// map output hardware channel layout to a compatible layout
let layoutTag = kAudioChannelLayoutTag_AudioUnit_5_0 // for 5.1 HW layout
let layout = AVAudioChannelLayout(layoutTag: layoutTag)
let envOutputFormat = AVAudioFormat(
    standardFormatWithSampleRate: outputHWFormat.sampleRate,
    channelLayout: layout);

engine.connect(envNode, to: output, format: envOutputFormat)
// map output hardware channel layout to a compatible layout
let layoutTag = kAudioChannelLayoutTag_AudioUnit_5_0 // for 5.1 HW layout
let layout = AVAudioChannelLayout(layoutTag: layoutTag)
let envOutputFormat = AVAudioFormat(
    standardFormatWithSampleRate: outputHWFormat.sampleRate,
    channelLayout: layout);

engine.connect(envNode, to: output, format: envOutputFormat)
Handling Multichannel Audio

AVAudioEngine setup: spatialized content

Mono Content → PlayerNode → Environment Node → OutputNode → Multichannel Hardware (5.1)

get content format
set connection format
let file = try AVAudioFile(forReading: fileURL)
engine.connect(player, to: envNode, format: file.processingFormat)

// set multichannel rendering algorithm
player.renderingAlgorithm = AVAudio3DMixingRenderingAlgorithm.SoundField
let file = try AVAudioFile(forReading: fileURL)
engine.connect(player, to: envNode, format: file.processingFormat)

// set multichannel rendering algorithm
player.renderingAlgorithm = AVAudio3DMixingRenderingAlgorithm.SoundField
Handling Multichannel Audio

AVAudioEngine setup: spatialized content

// schedule file on player
// start engine
// start player
AVAudioEngine

What’s new
AVAudioEngine

What’s new

Splitting support
Audio format conversion support
  • AVAudioCompressedBuffer
  • AVAudioConverter
AVAudioSequencer
AVAudioEngine

What’s new

Splitting support

Audio format conversion support
- AVAudioCompressedBuffer
- AVAudioConverter

AVAudioSequencer
Splitting Support

Sample setup

PlayerNode → EffectNode → MixerNode
Splitting Support

Sample setup

Only one-to-one connections were supported
Splitting Support

Sample setup

PlayerNode → MixerNode
Splitting Support
Sample setup

PlayerNode

EffectNode 1

EffectNode 2

MixerNode
Splitting Support

Code example
Splitting Support

Code example

Connection points—AVAudioConnectionPoint [node, bus]
Splitting Support

Code example

```swift
// Create an array of player connection points
let connPoints = [
    AVAudioConnectionPoint(node: effect1, bus: 0),
    AVAudioConnectionPoint(node: mixer, bus: 1),
    AVAudioConnectionPoint(node: effect2, bus: 0)]
```
Splitting Support

Code example

```swift
// Create an array of player connection points
let connPoints = [
    AVAudioConnectionPoint(node: effect1, bus: 0),
    AVAudioConnectionPoint(node: mixer, bus: 1),
    AVAudioConnectionPoint(node: effect2, bus: 0)]

// Make player connections
engine.connect(player, toConnectionPoints: connPoints, fromBus: 0,
    format: playerFormat)

// Make effect nodes to mixer connections
..
AVAudioMixing Protocol

With splitting

- PlayerNode
- EffectNode (Reverb)
- EffectNode (Distortion)
- MixerNode
AVAudioMixing Protocol

With splitting

// set player’s global mixing setting
player.volume = 0.5
AVAudioMixing Protocol

With splitting

// override mixing settings of mixer bus 0
if let mxDest0 = player.destinationForMixer(mixer, bus: 0) {
    mxDest0.volume = 0.8
}

volume = 0.8 (override)
AVAudioMixing Protocol

With splitting

// override mixing settings of mixer bus 1
if let mxDest1 = player.destinationForMixer(mixer, bus: 1) {
    mxDest1.volume = 0.2
}
AVAudioMixing Protocol
With splitting

// disconnect mixer bus1 input, corresponding mixing settings are lost
engine.disconnectNodeInput(mixer, bus: 1)
AVAudioMixing Protocol

With splitting

PlayerNode

Mixing Settings
volume = 0.5

EffectNode
(Distortion)

EffectNode
(Reverb)

Mixing Settings
volume = 0.8

MixerNode
AVAudioMixing Protocol
With splitting

// make a new connection, player's global mixing settings take effect
engine.connect(effect2, to:mixer, fromBus: 0, toBus: 1, format: format)
AVAudioMixing Protocol

With splitting

Source node with multiple mixer connections

- Properties changed on source node
  - Applied to all existing/new mixer connections
- Properties on individual mixer connections
  - Can be overridden
  - Not preserved on disconnections
Splitting support

Restrictions

From the split node to the mixer where all split paths terminate:

- Cannot have AVAudioUnitTimeEffect
- Cannot have any rate conversion
AVAudioEngine

What’s new

Splitting support

Audio format conversion support
• AVAudioCompressedBuffer
• AVAudioConverter

AVAudioSequencer
AVAudioBuffer

AVAudioPCMBuffer
- Uncompressed (PCM) audio data

AVAudioCompressedBuffer (new in iOS 9.0 / OS X El Capitan)
- Compressed audio data
- Used with AVAudioConverter
AVAudioConverter

Utility class, higher-level equivalent for AudioConverter

Audio format conversion

• PCM to PCM
  - Integer/float, bit depth, interleave/deinterleave, sample rate conversion

• PCM to/from compressed
  - Encoding
  - Decoding

Can be used in conjunction with AVAudioEngine
With AVAudioEngine: sample use case
AVAudioConverter

With AVAudioEngine: sample use case

Compressed Audio Stream

AVAudioEngine

PlayerNode → EffectNode → OutputNode
AVAudioConverter
With AVAudioEngine: sample use case
AVAudioConverter

With AVAudioEngine: sample use case
AVAudioConverter

Code example: encoding

// Input format: 44.1 kHz, 2 channel, non-interleaved, 16-bit signed integer
let inFormat = AVAudioFormat(
    commonFormat: AVAudioCommonFormat.PCMFormatInt16,
    sampleRate: 44100, channels: 2, interleaved: false)

// Output format: 44.1 kHz, 2 channel, AAC
var outDesc = AudioStreamBasicDescription(
    mSampleRate: 44100, mFormatID: kAudioFormatMPEG4AAC, mFormatFlags: 0,
    mBytesPerPacket: 0, mFramesPerPacket: 0, mBytesPerFrame: 0,
    mChannelsPerFrame: 2, mBitsPerChannel: 0, mReserved: 0)

let outFormat = AVAudioFormat(streamDescription: &outDesc)
// Input format: 44.1 kHz, 2 channel, non-interleaved, 16-bit signed integer
let inFormat = AVAudioFormat(
    commonFormat: AVAudioCommonFormat.PCMFormatInt16,
    sampleRate: 44100, channels: 2, interleaved: false)

// Output format: 44.1 kHz, 2 channel, AAC
var outDesc = AudioStreamBasicDescription(
    mSampleRate: 44100, mFormatID: kAudioFormatMPEG4AAC, mFormatFlags: 0,
    mBytesPerPacket: 0, mFramesPerPacket: 0, mBytesPerFrame: 0,
    mChannelsPerFrame: 2, mBitsPerChannel: 0, mReserved: 0)

let outFormat = AVAudioFormat(streamDescription: &outDesc)
// Create a converter
let converter = AVAudioConverter(fromFormat: inFormat, toFormat: outFormat)

// Allocate an input PCM buffer
let inBuffer = AVAudioPCMBuffer(PCMFormat: inFormat, frameCapacity: 1024)

// Allocate an output compressed buffer
let outBuffer = AVAudioCompressedBuffer(
    format: outFormat,
    packetCapacity: 8,
    maximumPacketSize: converter.maximumOutputPacketSize)
AVAudioConverter

Code example: encoding

// Create a converter
let converter = AVAudioConverter(fromFormat: inFormat, toFormat: outFormat)

// Allocate an input PCM buffer
let inBuffer = AVAudioPCMBuffer(PCMFormat: inFormat, frameCapacity: 1024)

// Allocate an output compressed buffer
let outBuffer = AVAudioCompressedBuffer(
    format: outFormat,
    packetCapacity: 8,
    maximumPacketSize: converter.maximumOutputPacketSize)
AVAudioConverter

Code example: encoding

// Create a converter
let converter = AVAudioConverter(fromFormat: inFormat, toFormat: outFormat)

// Allocate an input PCM buffer
let inBuffer = AVAudioPCMBuffer(PCMFormat: inFormat, frameCapacity: 1024)

// Allocate an output compressed buffer
let outBuffer = AVAudioCompressedBuffer(
    format: outFormat,
    packetCapacity: 8,
    maximumPacketSize: converter.maximumOutputPacketSize)
// Create an input block that’s called when converter needs input
let inputBlock : AVAudioConverterInputBlock = {
    inNumPackets, outStatus in
    if (<no_data_available>) {
        outStatus.memory = AVAudioConverterInputStatus.NoDataNow; return nil;
    } else if (<end_of_stream>) {
        outStatus.memory = AVAudioConverterInputStatus.EndOfStream; return nil;
    } else {
        ..
        outStatus.memory = AVAudioConverterInputStatus(HaveData;
        return inBuffer; // fill and return input buffer
    }
}
// Create an input block that’s called when converter needs input
let inputBlock : AVAudioConverterInputBlock = {
    inNumPackets, outStatus in
    if (<no_data_available>) {
        outStatus.memory = AVAudioConverterInputStatus.NoDataNow; return nil;
    } else if (<end_of_stream>) {
        outStatus.memory = AVAudioConverterInputStatus.EndOfStream; return nil;
    } else {
        ..
        outStatus.memory = AVAudioConverterInputStatus.HaveData;
        return inBuffer; // fill and return input buffer
    }
}
// Create an input block that's called when converter needs input
let inputBlock : AVAudioConverterInputBlock = {
    inNumPackets, outStatus in
    if (<no_data_available>) {
        outStatus.memory = AVAudioConverterInputStatus.NoDataNow; return nil;
    } else if (<end_of_stream>) {
        outStatus.memory = AVAudioConverterInputStatus.EndOfStream; return nil;
    } else {
        ..
        outStatus.memory = AVAudioConverterInputStatus.HaveData;
        return inBuffer; // fill and return input buffer
    }
}
AVAudioConverter

Code example: encoding

// Create an input block that's called when converter needs input
let inputBlock : AVAudioConverterInputBlock = {
    inNumPackets, outStatus in
    if (<no_data_available>) {
        outStatus.memory = AVAudioConverterInputStatus.NoDataNow; return nil;
    } else if (<end_of_stream>) {
        outStatus.memory = AVAudioConverterInputStatus.EndOfStream; return nil;
    } else {
        outStatus.memory = AVAudioConverterInputStatus.HaveData;
        return inBuffer; // fill and return input buffer
    }
}
// Conversion loop
outError = nil
while (true) {
    let status = converter.convertToBuffer(outBuffer, error: outError,
withInputFromBlock: inputBlock)
    if status == AVAudioConverterOutputStatus.EndOfStream ||
        status == AVAudioConverterOutputStatus.Error {
        break
    }
    // outBuffer contains output data
..
// Conversion loop
outError = nil
while (true) {
    let status = converter.convertToBuffer(outBuffer, error: outError, withInputFromBlock: inputBlock)
    if status == AVAudioConverterOutputStatus.EndOfStream ||
        status == AVAudioConverterOutputStatus.Error {
        break
    }
    // outBuffer contains output data
    ..
}
AVAudioConverter

Code example: encoding

// Conversion loop
outError = nil
while (true) {
    let status = converter.convertToBuffer(outBuffer, error: outError,
                                          withInputFromBlock: inputBlock)
    if status == AVAudioConverterOutputStatus.EndOfStream ||
        status == AVAudioConverterOutputStatus.Error {
        break
    }
    // outBuffer contains output data
    ..
}
AVAudioEngine

What’s new

Splitting support
Audio format conversion support
• AVAudioCompressedBuffer
• AVAudioConverter
AVAudioSequencer
AVAudioSequencer

Plays MIDI files

Associated with an AVAudioEngine during instantiation

Sends MIDI events to AVAudioUnitMIDIInstrument nodes in the engine
AVAudioSequencer

Sample setup
AVAudioSequencer

Sample setup

AVAudioSequencer

AVAudioEngine

InstrumentNode

MixerNode

OutputNode
AVAudioSequencer
Sample setup

AVAudioEngine

InstrumentNode

MixerNode

OutputNode
do {
    // setup instrument node (e.g. sampler)
    let sampler = AVAudioUnitSampler()
    engine.attachNode(sampler)
    engine.connect(sampler, to: engine.mainMixerNode, format: format)
    try sampler.loadInstrumentAtURL(instURL)

    // start the engine
    try engine.start()
} catch {
    // handle errors
}
do {
  // setup instrument node (e.g. sampler)
  let sampler = AVAudioUnitSampler()
  engine.attachNode(sampler)
  engine.connect(sampler, to: engine.mainMixerNode, format: format)
  try sampler.loadInstrumentAtURL(instURL)

  // start the engine
  try engine.start()
} catch {
  // handle errors
}
do {
    // create sequencer and associate with engine
    let sequencer = AVAudioSequencer(audioEngine: engine)

    // load MIDI file
    try sequencer.loadFromURL(fileURL, options: AVMusicSequenceLoadOptions.SMF_PreserveTracks)

    // start sequencer
    sequencer.prepareToPlay()
    try sequencer.start()
    // audio will start playing
} catch { // handle error
}
do {
    // create sequencer and associate with engine
    let sequencer = AVAudioSequencer(audioEngine: engine)

    // load MIDI file
    try sequencer.loadFromURL(fileURL,
        options: AVMusicSequenceLoadOptions.SMF_PreserveTracks)

    // start sequencer
    sequencer.prepareToPlay()
    try sequencer.start()
    // audio will start playing
} catch { // handle error
}
AVAudioSequencer

Code example: AVAudioSequencer setup

do {
    // create sequencer and associate with engine
    let sequencer = AVAudioSequencer(audioEngine: engine)

    // load MIDI file
    try sequencer.loadFromURL(fileURL,
                               options: AVMusicSequenceLoadOptions.SMF_PreserveTracks)

    // start sequencer
    sequencer.prepareToPlay()
    try sequencer.start()
    // audio will start playing
} catch { // handle error
}
AVAudioSequencer
Handling multiple tracks

AVAudioSequencer -> InstrumentNode 1
   t1  
   t2  
   InstrumentNode 2

AVAudioEngine
   MixerNode

   OutputNode
AVAudioSequencer

Handling multiple tracks

AVAudioSequencer

InstrumentNode 1

MixerNode

OutputNode

InstrumentNode 2
AVAudioSequencer

Code example: handling multiple tracks

// create and setup engine
// create sequencer
// load MIDI file
..

// send individual tracks to different instrument nodes in the engine
let tracks = sequencer.tracks
tracks[0].destinationAudioUnit = sampler
tracks[1].destinationAudioUnit = midiSynth

// start sequencer
..
AVAudioSequencer

Transport controls

Prepare to play, start, stop

Set playback position
  • Seconds or beats

Set playback rate
Demo

AVAudioEngine

Akshatha Nagesh ‘AudioEngine’er
Torrey Holbrook Walker Senior New Feature Salesperson
Summary
AVAudioEngine

Recap

- Handling multichannel audio

What’s new

- Splitting support
- Audio format conversion support
  - AVAudioCompressedBuffer
  - AVAudioConverter
- AVAudioSequencer
Inter-device Audio Mode for iOS

Torrey Holbrook Walker
Senior New Feature Salesperson
Recording From iOS Devices
Recording From iOS Devices
Recording From iOS Devices
Recording Audio From iOS

Digital recording is possible with USB host mode and USB hardware
3rd party software/frameworks
Couldn’t this be simpler?
Inter-device Audio Mode

Record audio digitally over the Lighting to USB cable
Stereo 24-bit @ 48 kHz stream format
USB 2.0 audio class-compliant implementation
Inter-device Audio Mode

No additional hardware
No additional software
No need to modify OS X or iOS applications
No system sounds routed to USB
Inter-device Audio Mode

Device can charge and sync

Temporarily disabled functionality

- Photo import
- Tethering
- QuickTime screen capture
Inter-device Audio Mode
Accessible via Audio MIDI Setup
Inter-device Audio Mode

iOS Device Browser

- 6 Minutes You're On: Enable
- 5C Notes Raining: Enable
- My Studio iPad: Disable
Demo
Inter-device audio mode

Torrey Holbrook Walker
Senior New Feature Salesperson
Inter-device Audio Mode

Requires OS X El Capitan and iOS 9
Works on all iPhones with a lightning connector
Works on all iPads with a lightning connector except first-gen iPad mini
Supports multiple devices simultaneously (if you’ve got the hubs)
@IBOutlet weak var viewContainer: NSView!
weak var iOSDeviceView: NSView?
var controller : CAInterDeviceAudioViewController?

@IBAction func toggleIOSDeviceView(sender: NSButton) {
    if iOSDeviceView == nil {
        controller = CAInterDeviceAudioViewController()
        iOSDeviceView = controller!.view
        viewContainer.addSubview(iOSDeviceView!)
    } else {
        iOSDeviceView!.removeFromSuperview()
        iOSDeviceView = nil
        controller = nil
    }
}
More CoreAudioKit View Controllers

CABTLEMIDIWindowController
Displays UI for configuring Bluetooth LE MIDI devices
NSWindowController subclass
CABTLEMIDIWindowController
More CoreAudioKit View Controllers

CANetworkBrowserWindowController
Displays UI for managing AVB audio devices
NSWindowController subclass
CoreAudioKit View Controller (cont.)
What’s New in AVAudioSession

Torrey Holbrook Walker
Senior New Feature Salesperson
Navigation Prompts and Podcasts

Problem

Listening to podcast while driving

Navigation prompts duck podcast audio

—> Bad user experience!
Navigation Prompts and Podcasts

Solution

Podcast and audio book apps:
• Use AVAudioSessionModeSpokenAudio

Navigation and fitness apps:
• Use AVAudioSessionCategoryOptions.InterruptSpokenAudioAndMixWithOthers

1st party apps have opted in:
• Maps, Podcasts, iBooks
do {
    let audioSession = AVAudioSession.sharedInstance()
    let category = AVAudioSessionCategoryPlayback
    var categoryOptions = AVAudioSessionCategoryOptions.DuckOthers
    if #available(iOS 9.0, *) {
        categoryOptions.unionInPlace(.InterruptSpokenAudioAndMixWithOthers)
    }
    try audioSession.setCategory(category, withOptions: categoryOptions)
} catch {
    // handle errors ...
}
func startNavPrompt(promptPath : NSURL) {
    do {
        let audioSession = AVAudioSession.sharedInstance()
        let player = try AVAudioPlayer(contentsOfURL: promptPath)
        player.delegate = self
        try audioSession.setActive(true)
        player.play()
    } catch {
        // handle errors ...
    }
}
// AVAudioPlayerDelegate method
func audioPlayerDidFinishPlaying(player: AVAudioPlayer, successfully flag: Bool) {
  do {
    let audioSession = AVAudioSession.sharedInstance()
    try audioSession.setActive(false,
        withOptions: .OptionNotifyOthersOnDeactivation)
  }
  catch {
    // handle errors ...
  }
}
do {
    let audioSession = AVAudioSession.sharedInstance()
    let category = AVAudioSessionCategoryPlayback
    var mode = AVAudioSessionModeDefault

    if #available(iOS 9.0, *) {
        mode = AVAudioSessionModeSpokenAudio
    }
    try audioSession.setCategory(category)
    try audioSession.setMode(mode)

    ...
}
Podcast Application

Session setup

...  

// add interruption handler
NSNotificationCenter.defaultCenter().addObserver(self, selector: 
"handleInterruption:", name:AVAudioSessionInterruptionNotification, object: 
audioSession)

// register for other important notifications

} catch {
  // handle errors ...
}
func handleInterruption(notification: NSNotification) {
    let userInfo = notification.userInfo as! [String: AnyObject]
    let type = userInfo[AVAudioSessionInterruptionTypeKey] as! AVAudioSessionInterruptionType

    switch type {
    case .Began:
        // update UI to indicate that playback has stopped
        if (state == isPlaying) {
            wasPlaying = true
            state = stopped
        }
    ...
}
case .Ended:
    if let flag = userInfo[AVAudioSessionInterruptionOptionKey] as? AVAudioSessionInterruptionOptions {
        if flag == .OptionShouldResume && wasPlaying {
            // rewind the audio a little
            player.play()
            state = isPlaying
            // and update the UI to reflect that playback has resumed
        }
    }
} // end switch
} // end func
Recap

Enhanced AVAudioEngine

Inter-device audio mode

CoreAudioKit View Controllers for IDAM, BLE MIDI, and AVB

AVAudioSessionModeSpokenAudio

AVAudioSessionCategoryOptions.InterruptSpokenAudioAndMixWithOthers
## Related Sessions

<table>
<thead>
<tr>
<th>Session</th>
<th>Location</th>
<th>Date and Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audio Unit Extensions</td>
<td>Nob Hill</td>
<td>Thursday 11:00 AM</td>
</tr>
<tr>
<td>What's New in SpriteKit</td>
<td>Mission</td>
<td>Wednesday 10:00 AM</td>
</tr>
<tr>
<td>Enhancements to SceneKit</td>
<td>Mission</td>
<td>Wednesday 2:30 PM</td>
</tr>
</tbody>
</table>
## Related Labs

<table>
<thead>
<tr>
<th>Audio Lab</th>
<th>Graphics, Games, and Media Lab A</th>
<th>Thursday 9:00 AM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audio Lab</td>
<td>Graphics, Games, and Media Lab A</td>
<td>Thursday 1:30 PM</td>
</tr>
</tbody>
</table>
More Information

Technical Support
Developer Technical Support
http://developer.apple.com/contact

Apple Developer Forums
http://developer.apple.com/forums

Bugs
http://bugreport.apple.com

General Inquiries
Craig Keithley, Technologies Evangelist
keithley@apple.com