# AudioSession and MultiRoute Audio

Session 505

**Torrey Holbrook Walker**
Core Audio Engineering

# Outline

- AudioSession Overview
  - Configuring the AudioSession
  - Audio routes and route change notifications
- AVAudioPlayer
  - Multichannel audio
  - Channel assignments
- MultiRoute category
- Using I/O units with AudioSession

# Managed Audio Experience on iOS

- Users carry iOS devices everywhere
- Goal: Consistent user experience
- Mobile device market is varied
- Choose the right APIs to communicate the app's intentions

# AudioSession Management

## Focus on the audio user experience

# AudioSession Management

## Focus on the audio user experience

- Make your app sounds
  - Behave according to user expectations
  - Be consistent with built-in apps

# AudioSession Management

## Focus on the audio user experience

- Make your app sounds
  - Behave according to user expectations
  - Be consistent with built-in apps
- What there is to do
  - Categorize your application
  - Respond to interruptions
  - Handle routing changes

# Using AudioSession in iOS 6

# AVAudioSession

- Use AVAudioSession class

  `<AVFoundation/AVAudioSession.h>`

  - Objective-C API for all AudioSession functionality

# Using AVAudioSession

## Five tasks

1. Set up the session and notification handler
2. Choose and set a category
   - Choose and set mode
3. Make session active
4. Handle interruptions
5. Handle route changes

# Set Up the Session


- Retrieve the AVAudioSession instance

```
AVAudioSession *session = [ AVAudioSession sharedInstance ];
```

- Register for notifications

```
[[NSNotificationCenter defaultCenter] addObserver: myObject
    selector: @selector(handleInterruption:)
    name: AVAudioSessionInterruptionNotification
    object: session];
```

# Choose and Set a Category

## Based on role of audio in your app

2.

Playback

Play and Record


Ambient


MultiRoute

Record

Audio Processing

Solo Ambient

# Choose and Set a Category

2.

```
// Retrieve session instance
AVAudioSession *session = [ AVAudioSession sharedInstance ];

// Register for notifications
...

// Request the “Play and Record” category
[ session setCategory:AVAudioSessionCategoryPlayAndRecord error:&errRet ];

// Set our session to be active
[ session setActive:YES error:&errRet ];

// Set up AVAudioPlayer or OpenAL or ..
...

// Handle interruptions
```

# Choose and Set a Category


```
// Retrieve session instance
AVAudioSession *session = [ AVAudioSession sharedInstance ];

// Register for notifications
...

// Request the “Play and Record” category
[ session setCategory:AVAudioSessionCategoryPlayAndRecord error:&errRet ];

// Set our session to be active
[ session setActive:YES error:&errRet ];

// Set up AVAudioPlayer or OpenAL or ..
...

// Handle interruptions
```

# Choose and Set a Mode

Voice Chat

Measurement


Movie Playback

Video Recording

[Default]

# Choose and Set a Mode


```
// Retrieve session instance
AVAudioSession *session = [ AVAudioSession sharedInstance ];

// Register for notifications
...

// Request the “Play and Record” category
[ session setCategory:AVAudioSessionCategoryPlayAndRecord error:&errRet ];

// Request the “Video Recording” mode
[ session setMode:AVAudioSessionModeVideoRecording error:&errRet ];

// Set our session to be active
[ session setActive:YES error:&errRet ];

// Set up AVAudioPlayer or AURemoteIO, etc.
```

# Choose and Set a Mode


```
// Retrieve session instance
AVAudioSession *session = [ AVAudioSession sharedInstance ];

// Register for notifications
...

// Request the “Play and Record” category
[ session setCategory:AVAudioSessionCategoryPlayAndRecord error:&errRet ];

// Request the “Video Recording” mode
[ session setMode:AVAudioSessionModeVideoRecording error:&errRet ];

// Set our session to be active
[ session setActive:YES error:&errRet ];

// Set up AVAudioPlayer or AURemoteIO, etc.
```

# Make Session Active


```
// Retrieve session instance
AVAudioSession *session = [ AVAudioSession sharedInstance ];

// Register for notifications
...

// Request the “Play and Record” category
[ session setCategory:AVAudioSessionCategoryPlayAndRecord error:&errRet ];

// Request the “Video Recording” mode
[ session setMode:AVAudioSessionModeVideoRecording error:&errRet ];

// Set our session to be active
[ session setActive:YES error:&errRet ];

// Set up AVAudioPlayer or AURemoteIO, etc.
```

# Make Session Active


```
// Retrieve session instance
AVAudioSession *session = [ AVAudioSession sharedInstance ];

// Register for notifications
...

// Request the “Play and Record” category
[ session setCategory:AVAudioSessionCategoryPlayAndRecord error:&errRet ];

// Request the “Video Recording” mode
[ session setMode:AVAudioSessionModeVideoRecording error:&errRet ];

// Set our session to be active
[ session setActive:YES error:&errRet ];

// Set up AVAudioPlayer or AURemoteIO, etc.
```

# Make Session Active


```
// Retrieve session instance
AVAudioSession *session = [ AVAudioSession sharedInstance ];

// Register for notifications
...

// Request the “Play and Record” category
[ session setCategory:AVAudioSessionCategoryPlayAndRecord error:&errRet ];

// Request the “Video Recording” mode
[ session setMode:AVAudioSessionModeVideoRecording error:&errRet ];

// Set our session to be active
[ session setActive:YES error:&errRet ];

// Set up AVAudioPlayer or AURemoteIO, etc.
```

# Make Session Active


```
// Retrieve session instance
AVAudioSession *session = [ AVAudioSession sharedInstance ];

// Register for notifications
...

// Request the “Play and Record” category
[ session setCategory:AVAudioSessionCategoryPlayAndRecord error:&errRet ];

// Request the “Video Recording” mode
[ session setMode:AVAudioSessionModeVideoRecording error:&errRet ];

// Set our session to be active
[ session setActive:YES error:&errRet ];

// Set up AVAudioPlayer or AURemoteIO, etc.
```

# Handle Interruptions

4.

- Session may be interrupted by higher priority audio
- Interruption makes your session inactive
  - Audio currently playing is stopped
- After the interruption is over
  - Reactivate certain state (API specific)
  - Become active again (if appropriate)

# Get Notified

## AVAudioSessionInterruptionNotification

```
userInfo
     AVAudioSessionInterruptionTypeKey (NSNumber)
       AVAudioSessionInterruptionTypeBegan
       AVAudioSessionInterruptionTypeEnded
     AVAudioSessionInterruptionOptionKey (AVAudioSessionInterruptionOptions)
       AVAudioSessionInterruptionOptionShouldResume
```

# Get Notified

AVAudioSessionInterruptionNotification

# Get Notified

## AVAudioSessionInterruptionNotification

- `AVAudioSessionInterruptionTypeBegan`
  - Audio has stopped, already inactive
  - Change state of UI, etc., to reflect non-playing state

# Get Notified

## AVAudioSessionInterruptionNotification

- `AVAudioSessionInterruptionTypeBegan`
  - Audio has stopped, already inactive
  - Change state of UI, etc., to reflect non-playing state
- `AVAudioSessionInterruptionTypeEnded`
  - Make session active
  - Update user interface
  - `AVAudioSessionInterruptionOptionShouldResume` option

# Other Notifications

AVAudioSessionMediaServicesWereResetNotification

# Other Notifications

## AVAudioSessionMediaServicesWereResetNotification

- If the media server resets for any reason, handle this notification to reconfigure audio or do any housekeeping, if necessary

# Other Notifications

## AVAudioSessionMediaServicesWereResetNotification

- If the media server resets for any reason, handle this notification to reconfigure audio or do any housekeeping, if necessary
- No userInfo dictionary for this notification
- Audio streaming objects are invalidated (zombies)
- Handle this notification by fully reconfiguring audio

# Handle Route Changes

## What users expect

5.

# Handle Route Changes

## What users expect

5.

- Last in wins

# Handle Route Changes

## What users expect

5.

- Last in wins
  - Plugging in
    - Routed to headset/headphone
    - Continues playing without pause

# Handle Route Changes

## What users expect

5.

- Last in wins
  - Plugging in
    - Routed to headset/headphone
    - Continues playing without pause
  - Unplugging
    - Routed to previous output
    - Audio playback should pause

# New Route en Route

## AVAudioSessionRouteChangeNotification

```
AVAudioSessionRouteChangeReasonKey (NSNumber)
     AVAudioSessionRouteChangeReasonNewDeviceAvailable
     AVAudioSessionRouteChangeReasonCategoryChange
     ...
AVAudioSessionRouteChangePreviousRouteKey (AVAudioSessioRouteDescription)
```

# Querying the Route

## The current route is a collection of inputs and outputs

- `[session currentRoute]`
  - AVAudioSessionRouteDescription with detailed information about the route
- `[[session currentRoute] inputs]`
  - Value is an array of input AVAudioSessionPortDescription objects
- `[[session currentRoute] outputs]`
  - Value is an array of output AVAudioSessionPortDescription objects

# AVAudioSessionPortDescription

## A port is a single hardware input or output on a device

```
@property(readonly) NSString * portType

/* eg.,  AVAudioSessionPortLineOut, AVAudioSessionPortHeadphones */

@property(readonly) NSString * portName

/* eg., “Line Out” or “Headphones” */

@property(readonly) NSString * UID      // system-assigned

@property(readonly) NSArray * channels  // AVAudioSessionChannelDescription
```

# AVAudioSessionChannelDescription

## A channel description is a single channel on a port

```
@property(readonly) NSString * channelName

/* eg.,  “Headphone Left” or “HDMI Output 1” */

@property(readonly) NSString * owningPortUID  // system-assigned

@property(readonly) NSUInteger channelNumber  // 1-based channel index
```

# Using Audio Session

## Summary

1. Set up the session and notification handler
2. Choose and set a category
   - Choose and set mode
3. Make session active
4. Handle interruptions
5. Handle route changes

# No More Mix and Match

- Audio Session Services (Deprecated)

```
<AudioToolbox/AudioServices.h>
```

- All functionality from the C API has been moved to AVAudioSession

# A Few More Changes

- AVAudioSessionDelegate has been deprecated
  - Register for the NSNotifications instead
- Some properties have been deprecated to make naming consistent

# Using AVAudioPlayer

# AVAudioPlayer

- Use to play caf, m4a, mp3, aif, wav, au, snd, aac
- Play, pause, seek, stop
- Volume, panning, looping, rate control

# AVAudioPlayer

## Creating a player

- Create from a file URL

```
// Create the player from local file
    NSURL *url = ...
    AVAudioPlayer *player = [[AVAudioPlayer alloc] initWithContentsOfURL:url
    withError:&error];
```

# AVAudioPlayer

## Playing a file from the music library

- Obtain the reference to an MPMediaItem object from the device's music library (e.g. using MPMediaPickerController)

```
// Create the player using a user-selected file
NSURL *mediaUrl =
      [myMediaItem valueForProperty:MPMediaItemPropertyAssetURL];
AVAudioPlayer *player = [[AVAudioPlayer alloc]
     initWithContentsOfURL:mediaUrl withError:&error];
```

# Audio, Interrupted

## AVAudioPlayerDelegate methods

- `- (void) audioPlayerBeginInterruption`
  - Playback has stopped, already inactive
  - Change state of UI, etc., to reflect non-playing state
- `- (void) audioPlayerEndInterruption:withOption:`
  `AVAudioSessionInterruptionOptionShouldResume`
  - Update user interface
  - Resume playback

# Multichannel Audio on iOS

# When Is Multichannel Available?

- USB Inputs > 2 were available on iOS 5
- USB Outputs > 2 are now available on iOS 6

# Stereo and Mono with Multichannel

## What if you don't need that many channels?

- When the audio route contains more than two output channels, stereo content plays to the first two channels
- Mono also plays to the first two channels
- For recording, mono records the first channel only

# Channel Selection

## Choose which inputs and outputs to use

- Set an array of AVAudioSessionChannelDescription(s)
- Can be used with AVAudioPlayer and AVAudioRecorder
- There are also congruent channel selection methods for
  - AudioQueue
  - AURemoteIO

# AudioSession MultiRoute Category

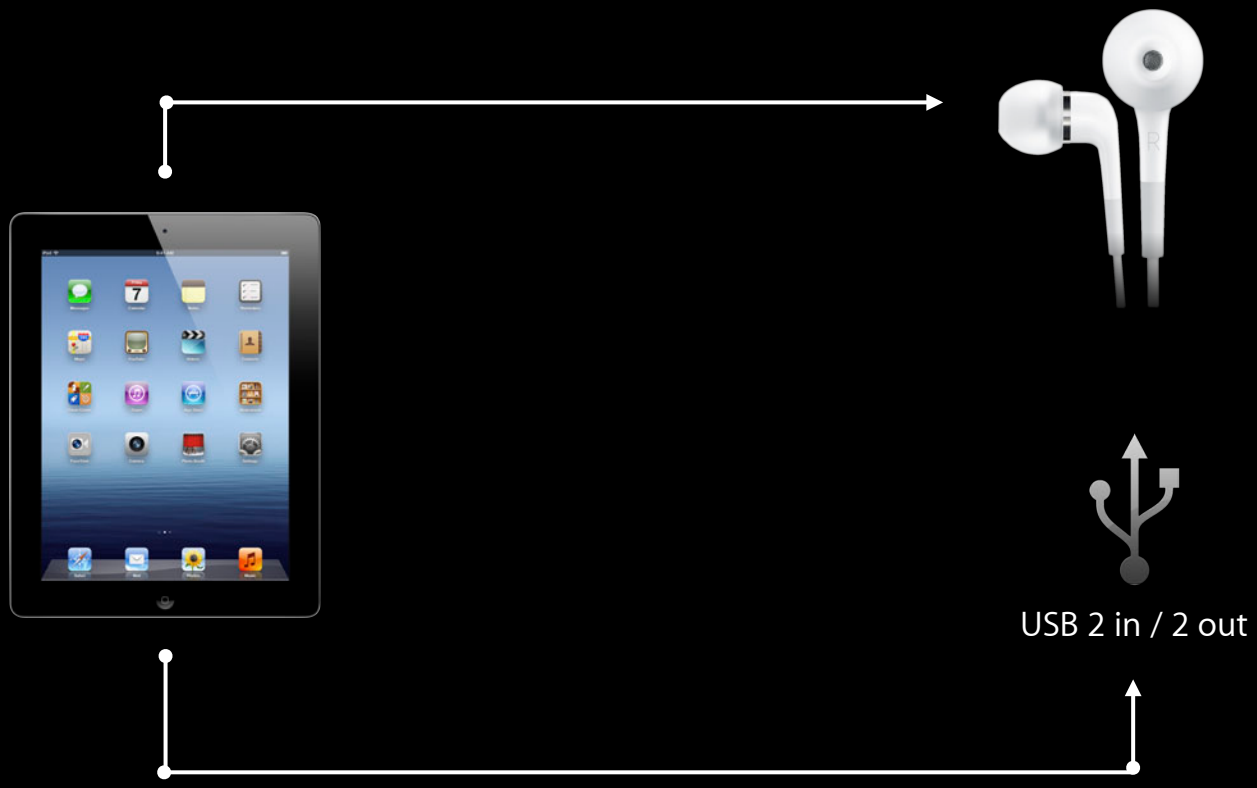
# Last in Wins

## Audio configuration is determined by the last route change

USB 2 in / 2 out

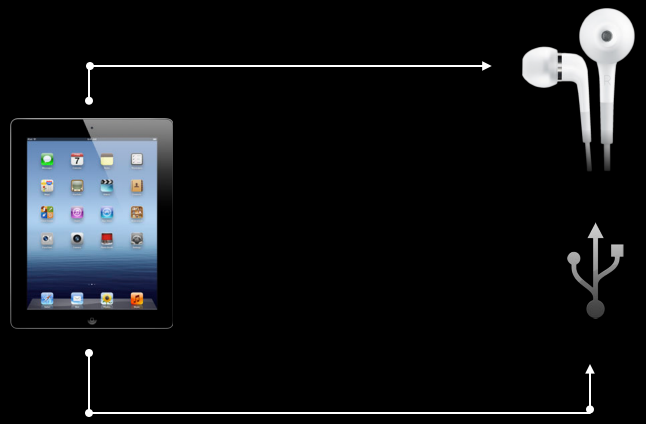
# Last in Wins

## Audio configuration is determined by the last route change



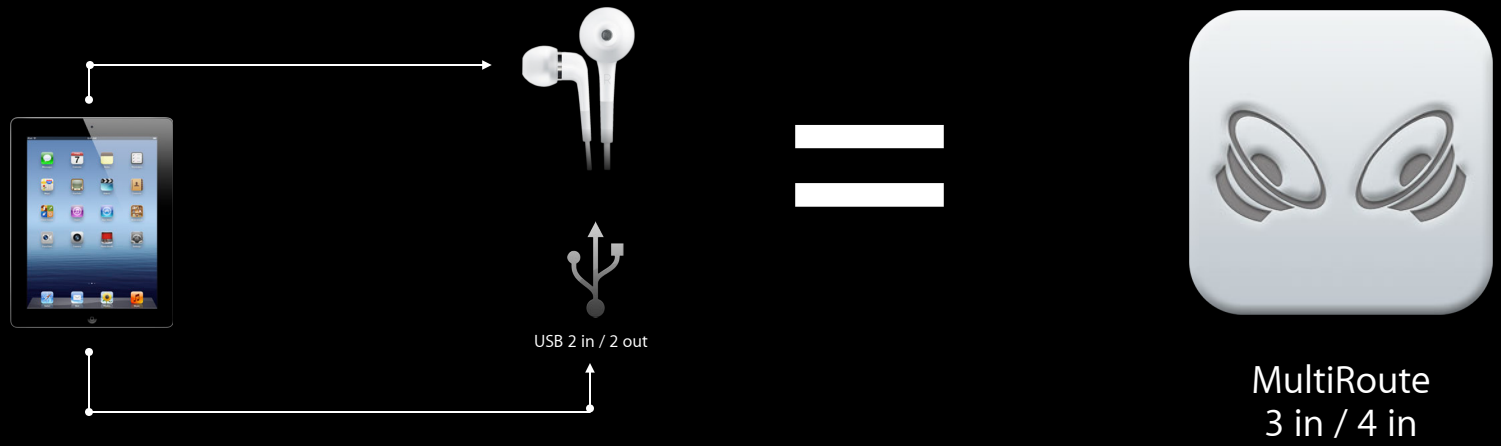
# In MultiRoute, Everybody Wins

## Capable inputs and outputs are treated as a single route



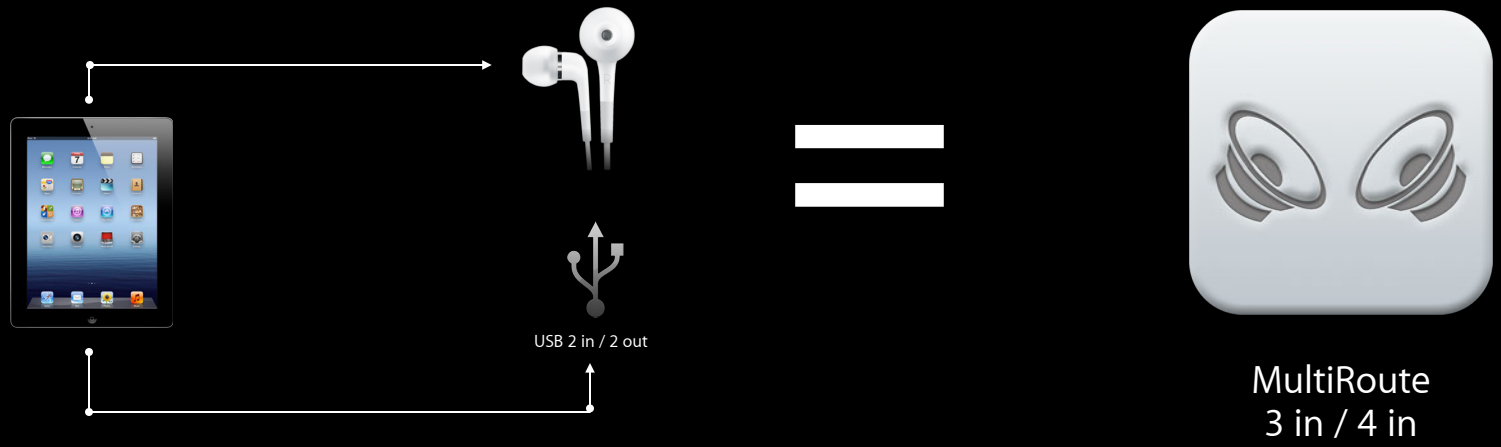
# In MultiRoute, Everybody Wins

## Capable inputs and outputs are treated as a single route



# In MultiRoute, Everybody Wins

## Capable inputs and outputs are treated as a single route




**Go wild.**

# MultiRoute Demo

**Harry Tormey**
Core Audio Engineering

# MultiRoute Demo

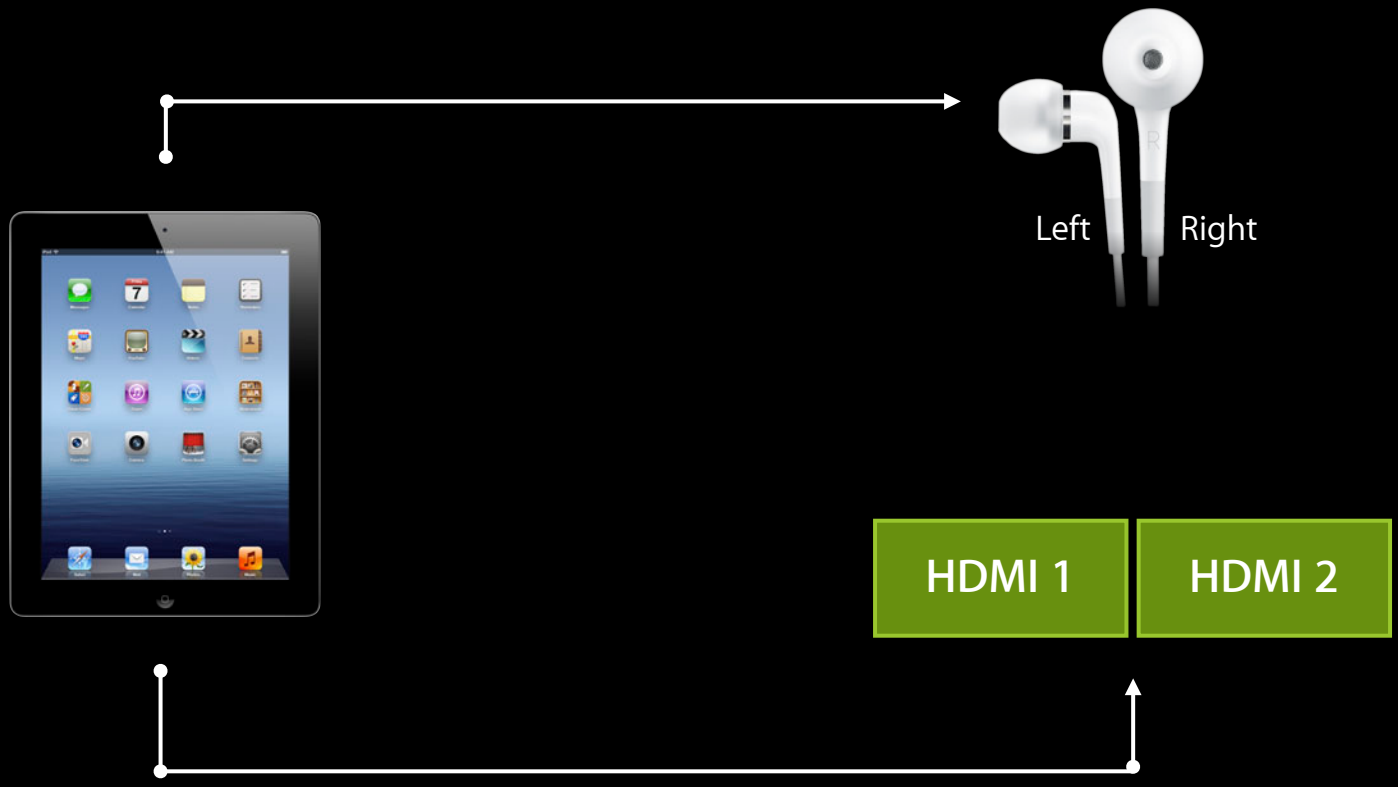
## Before MultiRoute category

HDMI 1

HDMI 2

# MultiRoute Demo

## Before MultiRoute category



# MultiRoute Demo

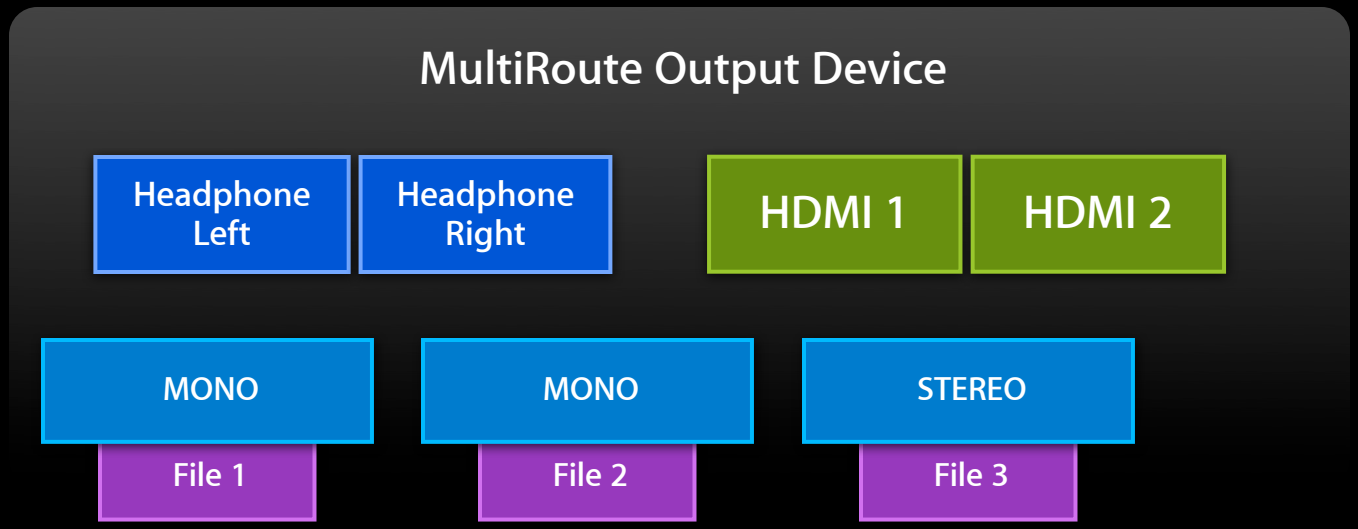
## After MultiRoute category

# MultiRoute Demo

## After MultiRoute category

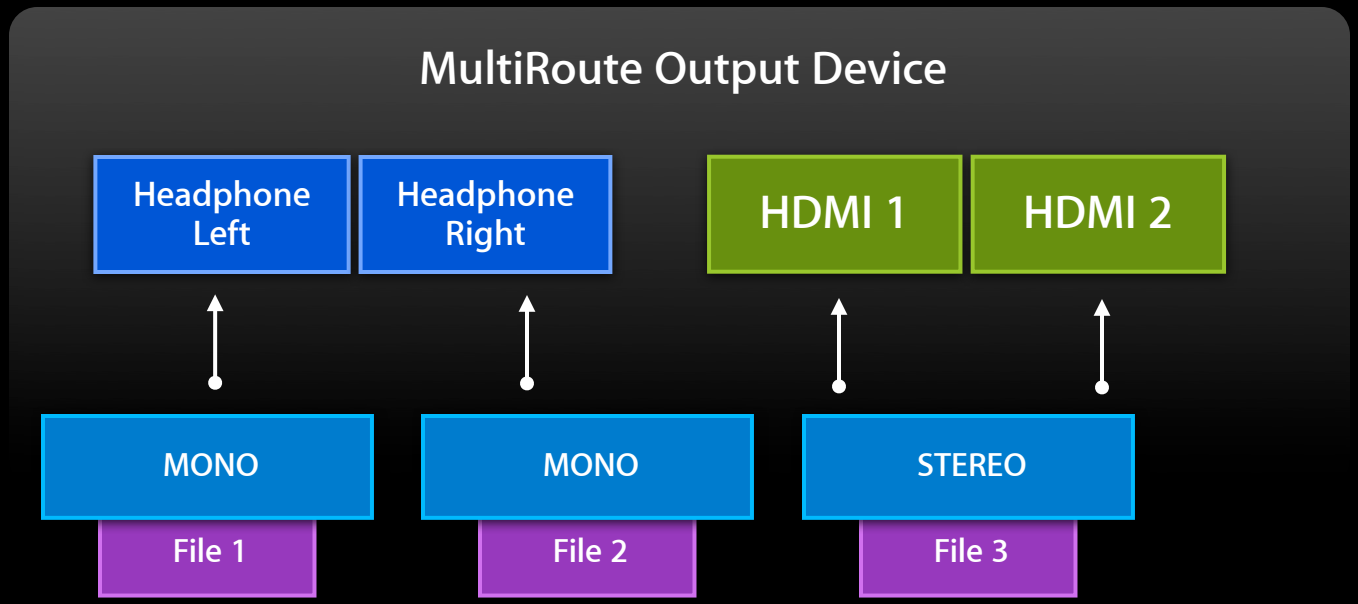
# MultiRoute Demo

## Select files to play



# MultiRoute Demo

## Channel assignments



# Using MultiRoute

## How much control do you need?

- DJ applications (stereo cue mix and house mix)
- Digital Audio Workstation software
- Multichannel software instruments
- Unique audio software applications

# AVAudioPlayer Setup

## Register for route change notifications

```
// Retrieve session instance
AVAudioSession *session = [ AVAudioSession sharedInstance ];

// Register for Route Change notifications
[[NSNotificationCenter defaultCenter] addObserver: myObject
     selector: @selector(handleRouteChange:)
     name: AVAudioSessionRouteChangeNotification
     object: session];

// Request the MultiRoute category
[ session setCategory:AVAudioSessionCategoryMultiRoute error:&errRet ];

// Set our session to be active
[ session setActive:YES error:&errRet ];

// Set up AVAudioPlayer, etc.
```

# AVAudioPlayer Setup

## Register for route change notifications

```
// Retrieve session instance
AVAudioSession *session = [ AVAudioSession sharedInstance ];

// Register for Route Change notifications
[[NSNotificationCenter defaultCenter] addObserver: myObject
     selector: @selector(handleRouteChange:)
     name: AVAudioSessionRouteChangeNotification
     object: session];

// Request the MultiRoute category
[ session setCategory:AVAudioSessionCategoryMultiRoute error:&errRet ];

// Set our session to be active
[ session setActive:YES error:&errRet ];

// Set up AVAudioPlayer, etc.
```

# Set MultiRoute Category

```
// Retrieve session instance
AVAudioSession *session = [ AVAudioSession sharedInstance ];

// Register for Route Change notifications
...

// Request the MultiRoute category
[ session setCategory:AVAudioSessionCategoryMultiRoute error:&errRet ];

// Set our session to be active
[ session setActive:YES error:&errRet ];

// Set up AVAudioPlayer, etc.
```

# Set MultiRoute Category

```
// Retrieve session instance
AVAudioSession *session = [ AVAudioSession sharedInstance ];

// Register for Route Change notifications
...

// Request the MultiRoute category
[ session setCategory:AVAudioSessionCategoryMultiRoute error:&errRet ];

// Set our session to be active
[ session setActive:YES error:&errRet ];

// Set up AVAudioPlayer, etc.
```

# AVAudioPlayer Setup

## Get the route

```
// Retrieve the route information
AVAudioSessionRouteDescription *route = [ session currentRoute ];

NSArray *outputs = [route outputs];

// Display the route
```

# AVAudioPlayer Setup

## Get the route

```
// Retrieve the route information
AVAudioSessionRouteDescription *route = [ session currentRoute ];

NSArray *outputs = [route outputs];

// Display the route
```

# AVAudioPlayer Setup

## Display the route (optional)

```
// Display the route
NSLog(@"Route %@", [session currentRoute]);

<AVAudioSessionRouteDescription: ...,
  inputs = (
   0: type = MicrophoneBuiltIn; name = iPhone Microphone;
      UID = Built-In Microphone;
       channels = (
      0: name = MicrophoneBuiltIn; number = 1; port UID = Built-In
Microphone ))

  outputs = (
   0: type = Headphones; name = Headphones; UID = Wired Headphones;
       channels = (
      0: name = Headphone Left; number = 1; port UID = Wired Headphones,
      1: name = Headphone Right; number = 2; port UID = Wired Headphones ))>
```

# AVAudioPlayer Setup

## Create the player

```
// Create an AVAudioPlayer
AVAudioPlayer *player = [[AVAudioPlayer alloc] initWithContentsOfUrl:myURL
     error:&err];

// Select channel(s)
AVAudioSessionChannelDescription *desiredChannel =
     [[[outputs objectAtIndex:0] channels] objectAtIndex:0];

// Create an array of desired channels
NSArray *channelDescriptions = [NSArray arrayWithObject:desiredChannel];

// Assign the channels
player.channelAssignments = channelDescriptions;

// Play audio
[player play];
```

# AVAudioPlayer Setup

## Create the player

```
// Create an AVAudioPlayer
AVAudioPlayer *player = [[AVAudioPlayer alloc] initWithContentsOfUrl:myURL
     error:&err];

// Select channel(s)
AVAudioSessionChannelDescription *desiredChannel =
     [[[outputs objectAtIndex:0] channels] objectAtIndex:0];

// Create an array of desired channels
NSArray *channelDescriptions = [NSArray arrayWithObject:desiredChannel];

// Assign the channels
player.channelAssignments = channelDescriptions;

// Play audio
[player play];
```

# AVAudioPlayer Setup

## Choose the channels you want

```
// Create an AVAudioPlayer
AVAudioPlayer *player = [[AVAudioPlayer alloc] initWithContentsOfUrl:myURL
     error:&err];

// Select channel(s)
AVAudioSessionChannelDescription *desiredChannel =
     [[[outputs objectAtIndex:0] channels] objectAtIndex:0];

// Create an array of desired channels
NSArray *channelDescriptions = [NSArray arrayWithObject:desiredChannel];

// Assign the channels
player.channelAssignments = channelDescriptions;

// Play audio
[player play];
```

# AVAudioPlayer Setup

## Choose the channels you want

```
// Create an AVAudioPlayer
AVAudioPlayer *player = [[AVAudioPlayer alloc] initWithContentsOfUrl:myURL
     error:&err];

// Select channel(s)
AVAudioSessionChannelDescription *desiredChannel =
     [[[outputs objectAtIndex:0] channels] objectAtIndex:0];

// Create an array of desired channels
NSArray *channelDescriptions = [NSArray arrayWithObject:desiredChannel];

// Assign the channels
player.channelAssignments = channelDescriptions;

// Play audio
[player play];
```

# AVAudioPlayer Setup

## Create an array of channels and assign them

```
// Create an AVAudioPlayer
AVAudioPlayer *player = [[AVAudioPlayer alloc] initWithContentsOfUrl:myURL
     error:&err];

// Select channel(s)
AVAudioSessionChannelDescription *desiredChannel =
     [[[outputs objectAtIndex:0] channels] objectAtIndex:0];

// Create an array of desired channels
NSArray *channelDescriptions = [NSArray arrayWithObject:desiredChannel];

// Assign the channels
player.channelAssignments = channelDescriptions;

// Play audio
[player play];
```

# AVAudioPlayer Setup

## Create an array of channels and assign them

```
// Create an AVAudioPlayer
AVAudioPlayer *player = [[AVAudioPlayer alloc] initWithContentsOfUrl:myURL
     error:&err];

// Select channel(s)
AVAudioSessionChannelDescription *desiredChannel =
     [[[outputs objectAtIndex:0] channels] objectAtIndex:0];

// Create an array of desired channels
NSArray *channelDescriptions = [NSArray arrayWithObject:desiredChannel];

// Assign the channels
player.channelAssignments = channelDescriptions;

// Play audio
[player play];
```

# AVAudioPlayer Setup

```
// Create an AVAudioPlayer
AVAudioPlayer *player = [[AVAudioPlayer alloc] initWithContentsOfUrl:myURL
     error:&err];

// Select channel(s)
AVAudioSessionChannelDescription *desiredChannel =
     [[[outputs objectAtIndex:0] channels] objectAtIndex:0];

// Create an array of desired channels
NSArray *channelDescriptions = [NSArray arrayWithObject:desiredChannel];

// Assign the channels
player.channelAssignments = channelDescriptions;

// Play audio
[player play];
```

# AVAudioPlayer Setup

```
// Create an AVAudioPlayer
AVAudioPlayer *player = [[AVAudioPlayer alloc] initWithContentsOfUrl:myURL
     error:&err];

// Select channel(s)
AVAudioSessionChannelDescription *desiredChannel =
     [[[outputs objectAtIndex:0] channels] objectAtIndex:0];

// Create an array of desired channels
NSArray *channelDescriptions = [NSArray arrayWithObject:desiredChannel];

// Assign the channels
player.channelAssignments = channelDescriptions;

// Play audio
[player play];
```

# Using I/O Units with AudioSession

**William Stewart**
Core Audio Engineering

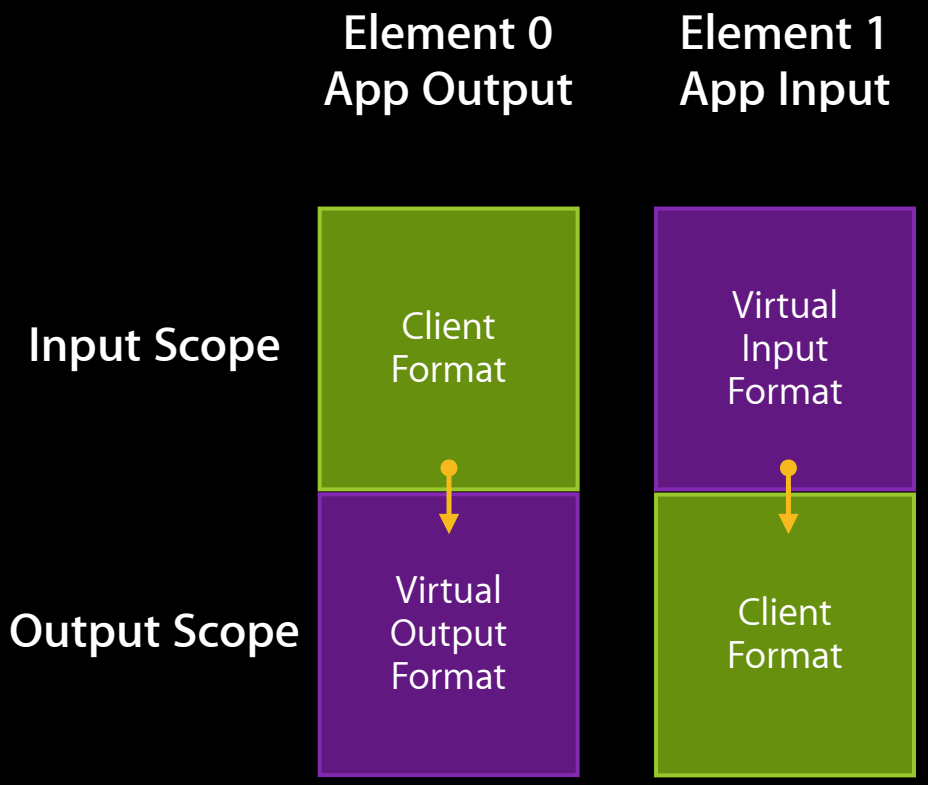
# AURemoteIO

- This is an audio unit that interfaces directly to Audio input and output
- Provides low latency audio I/O
  - Less than 10 msec depending on audio devices in use
- Used by
  - OpenAL implementation
  - Games with their own audio engines
  - Music apps
  - VoIP apps

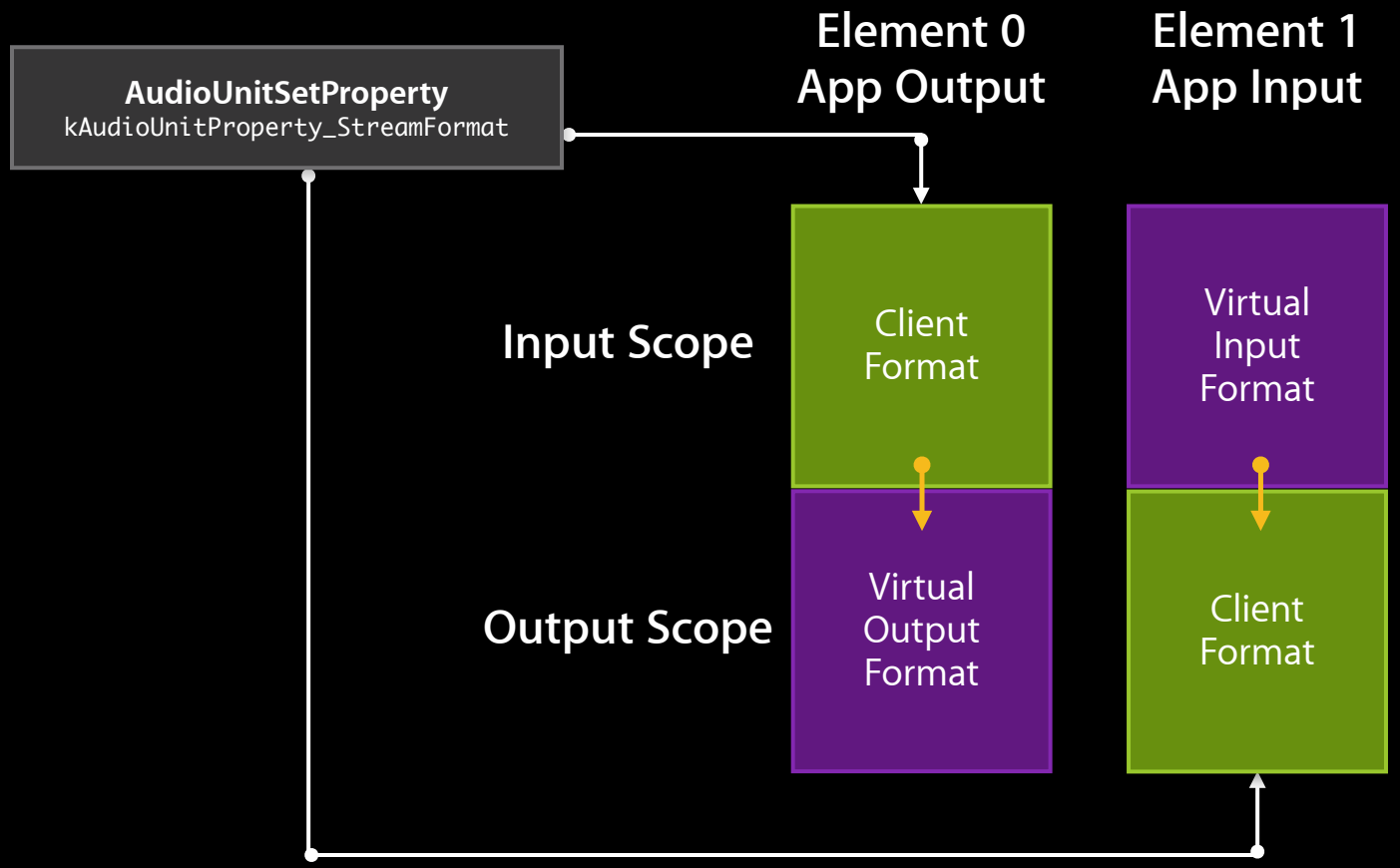
# I/O Units Usage Pattern

- Examine the I/O formats
- Set up the client input and output formats
- Initialize the AudioUnit so you can use it
- Establish your data mechanisms
- Start audio I/O

# Anatomy of an I/O Unit



# Setting Your Client Stream Formats



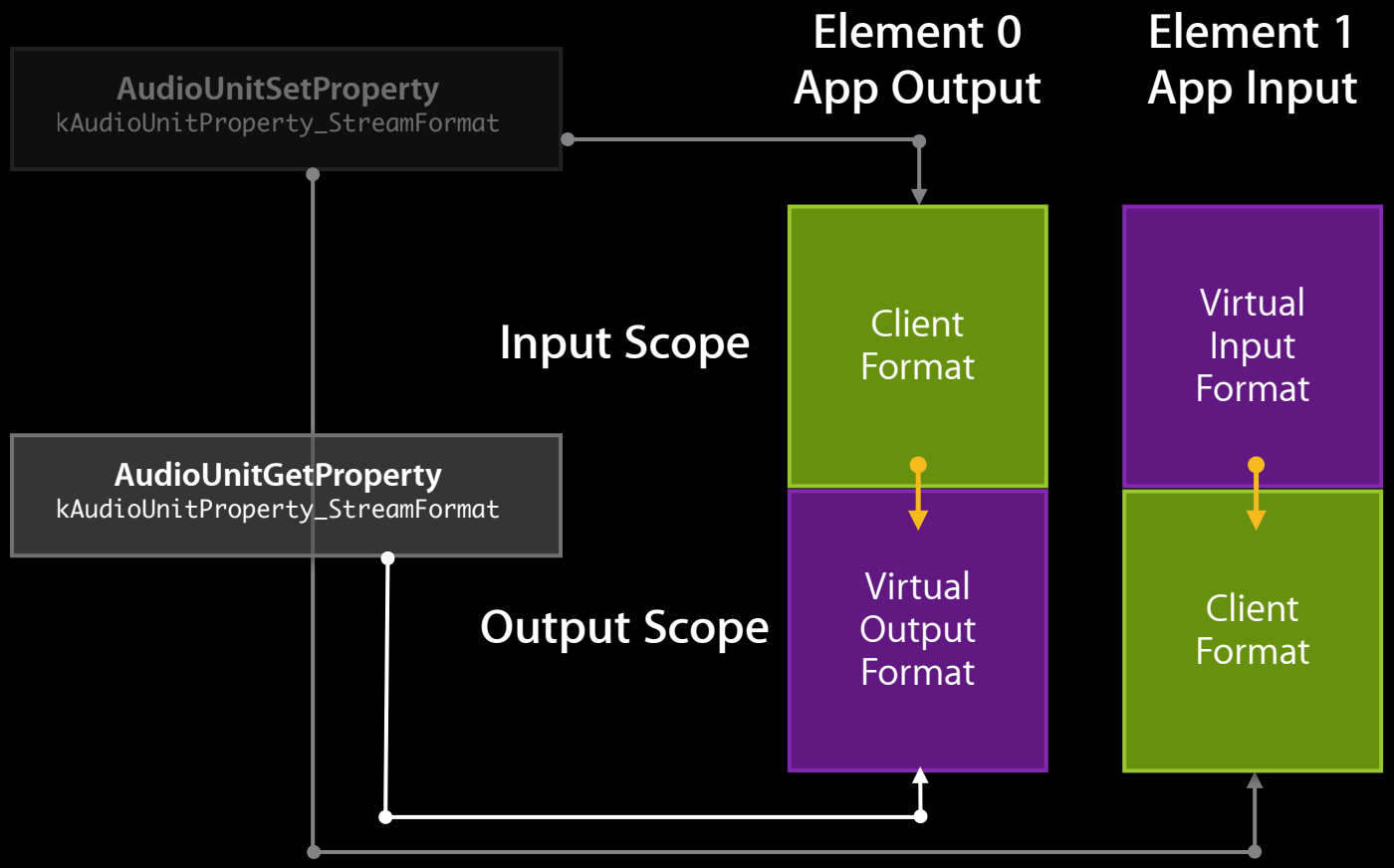
# Audio Route Formats and I/O Unit

- AURemoteIO
  - Stream format for Output Scope, Bus 0
- AVAudioSession route description

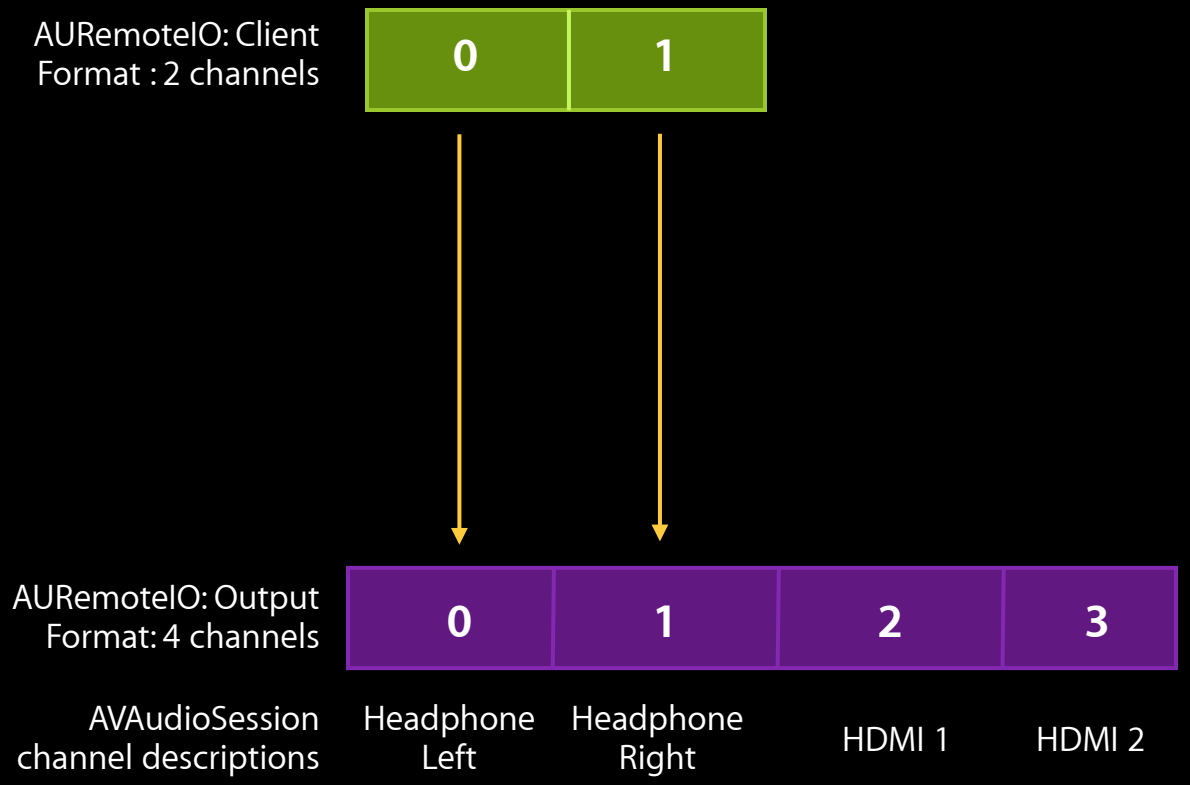
```
[[session currentRoute] outputs]
```

  - For instance, from the demo
    - 2 channels for headphone
    - 2 channels for HDMI output

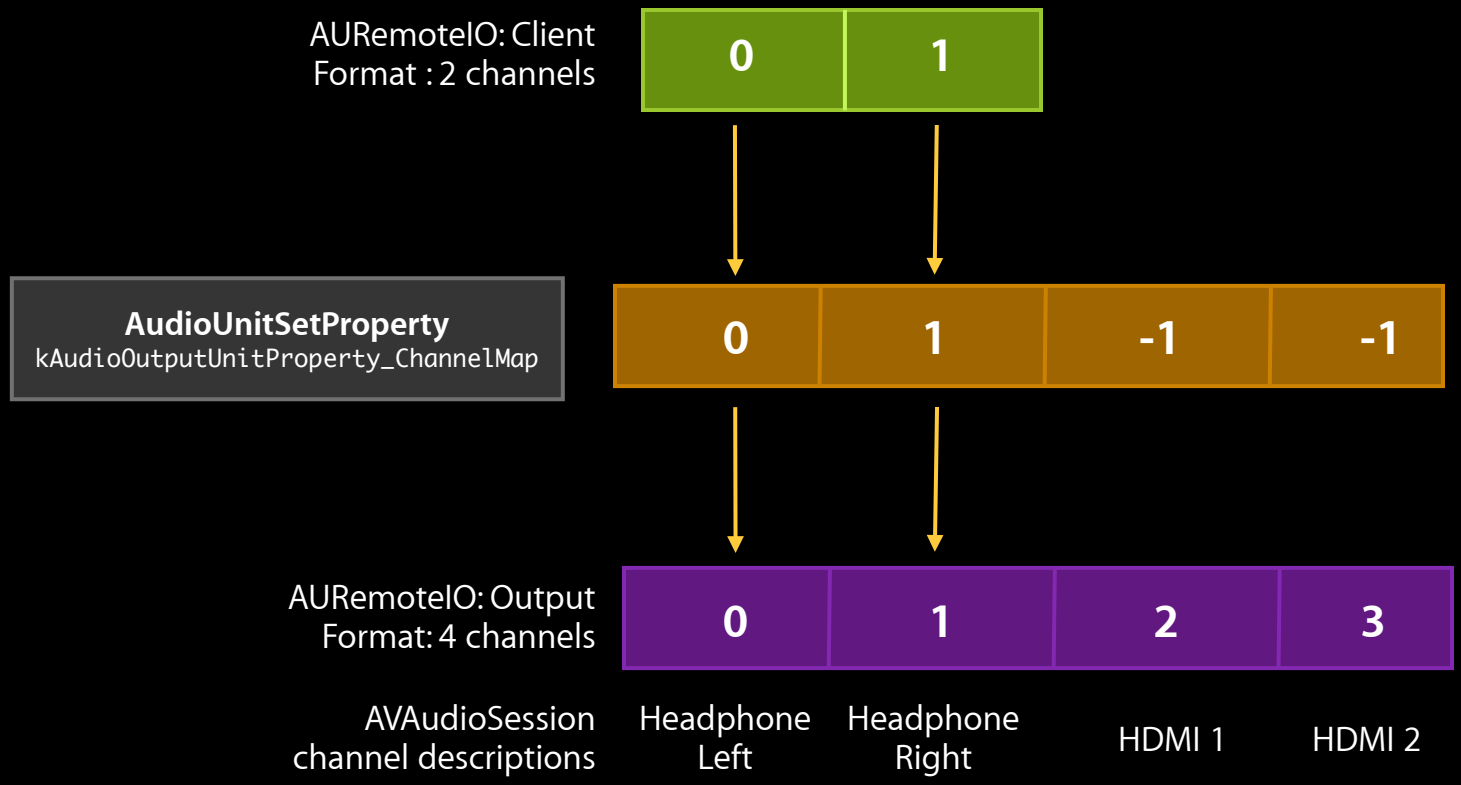
# Anatomy of an I/O Unit



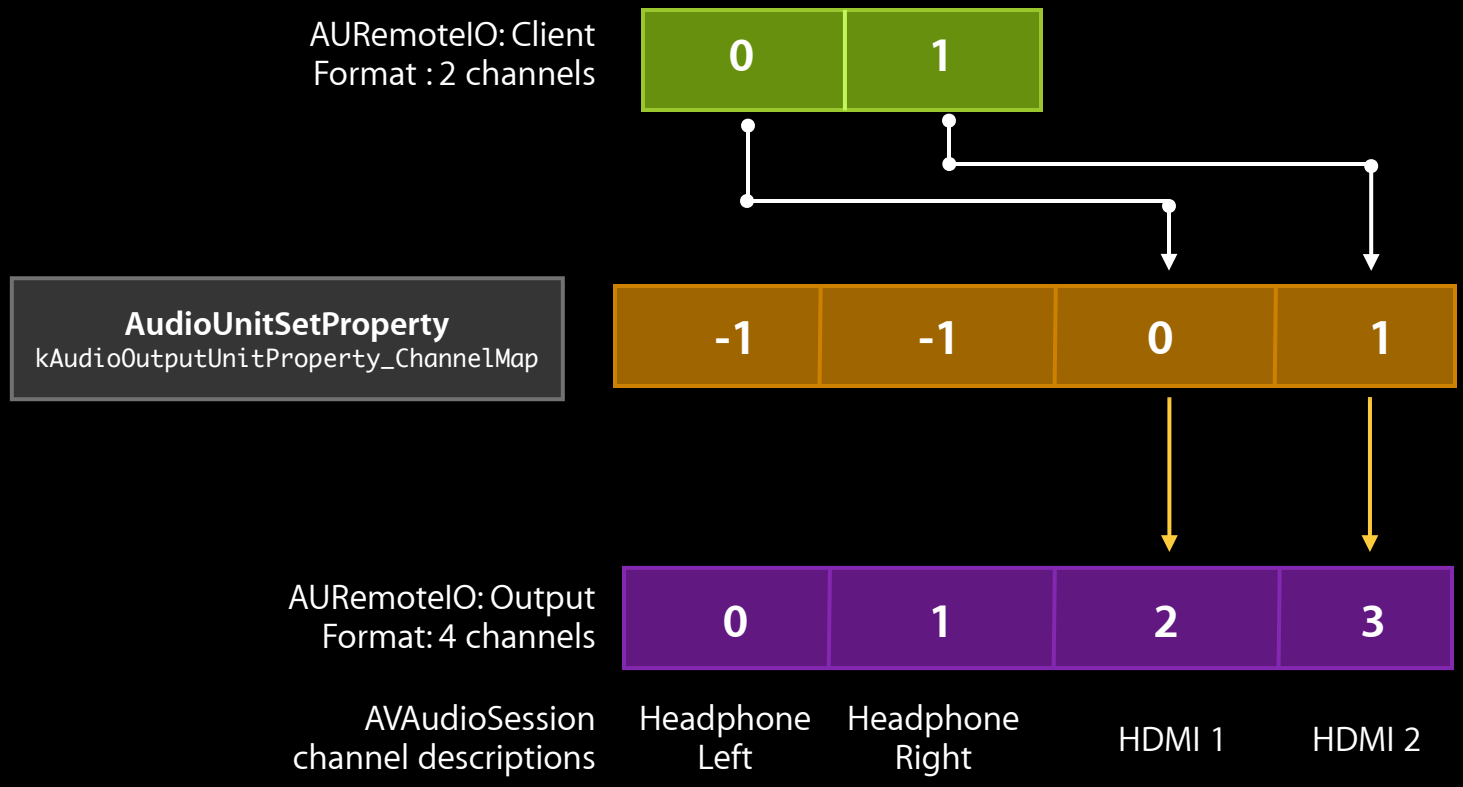
# Channel Map on Output



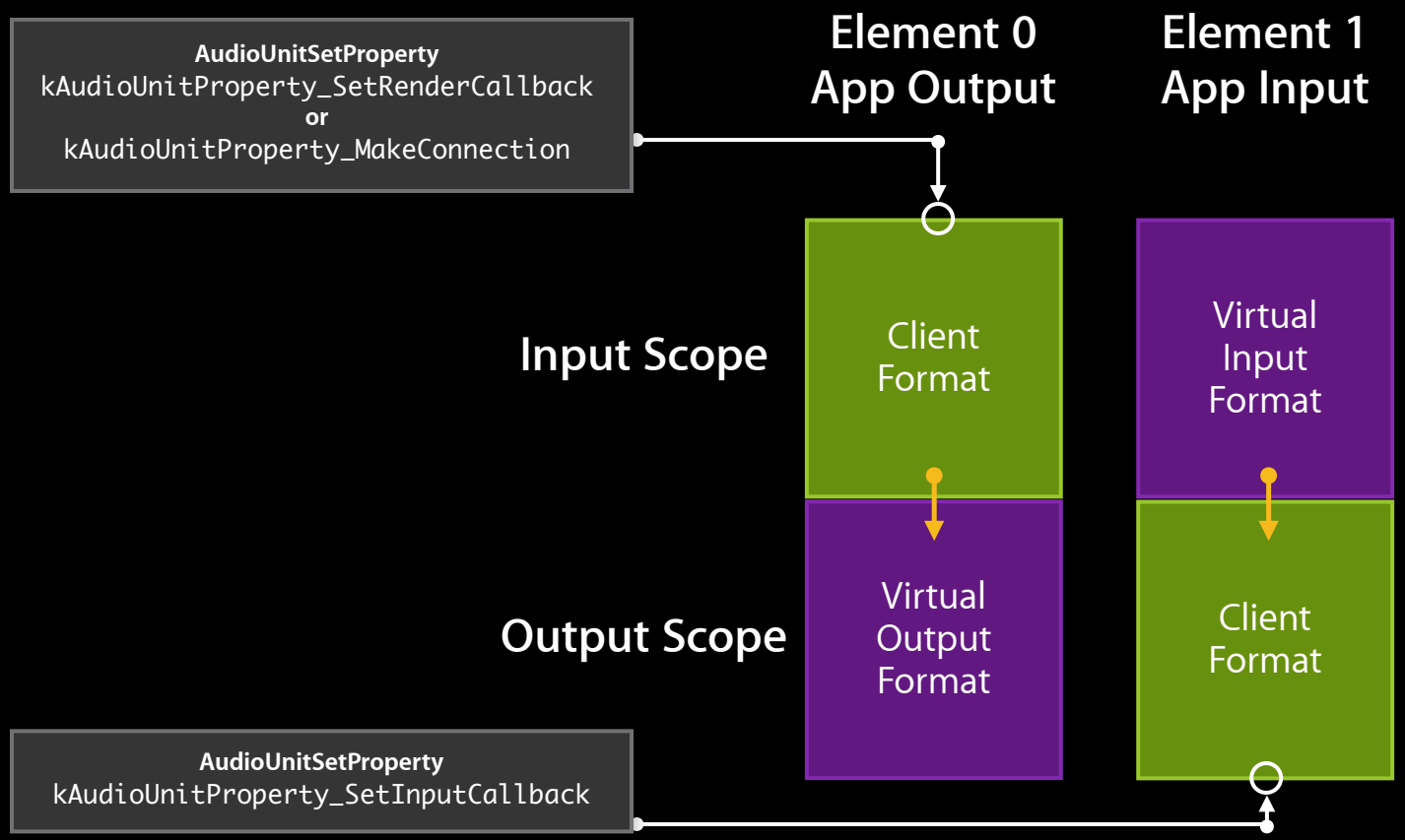
# Channel Map on Output



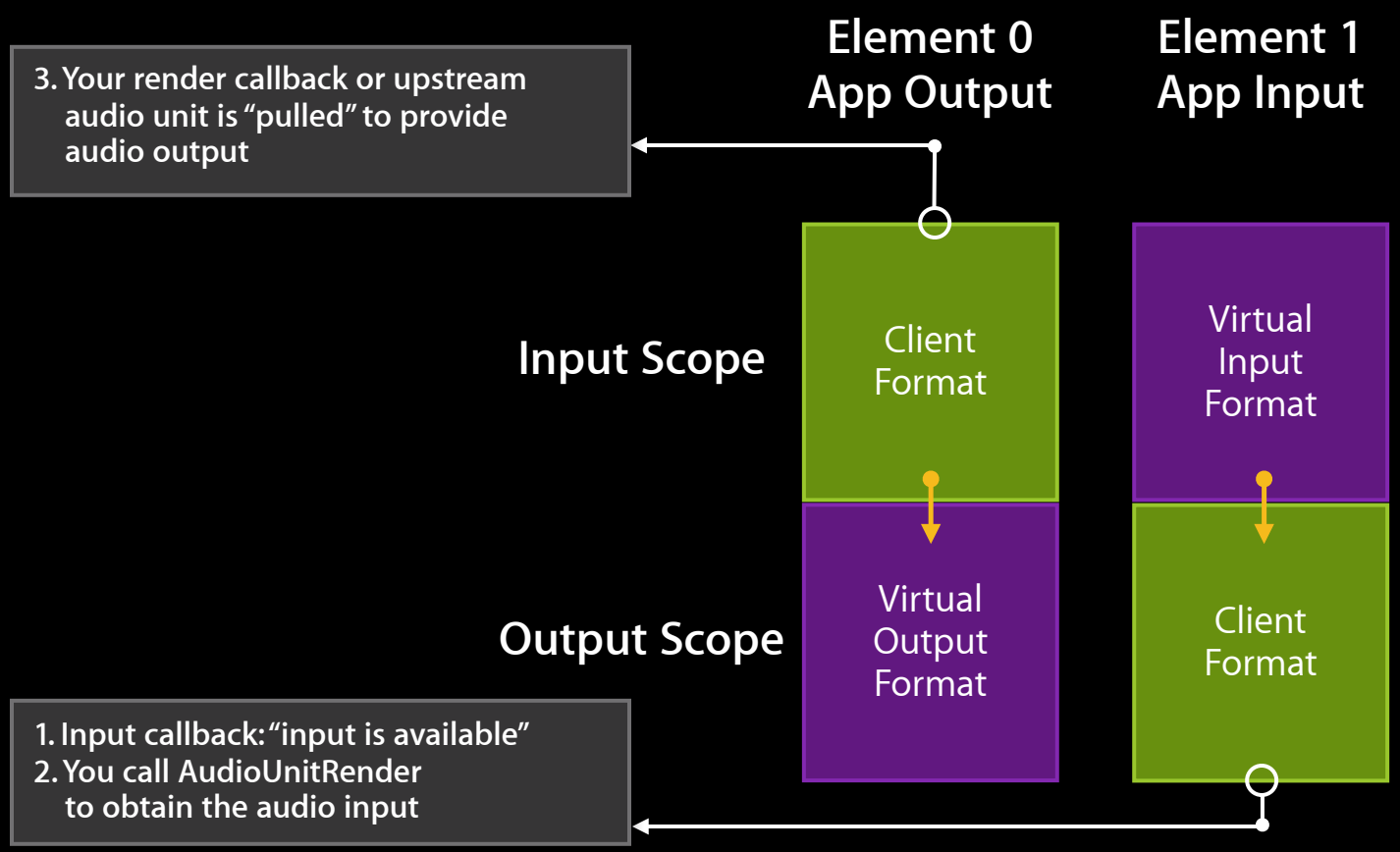
# Channel Map on Output



# I/O Cycle Setup



# I/O Cycle


3. Your render callback or upstream audio unit is "pulled" to provide audio output
Element 0
App Output
Element 1
App Input
Input Scope
Client Format
Virtual Input Format
Output Scope
Virtual Output Format
Client Format
1. Input callback: "input is available"
2. You call AudioUnitRender to obtain the audio input

# AVAudioSession—Game

- Typically use `AVAudioSessionCategoryAmbient`
  - Your audio is mixable
  - Output only
  - Audio is silent if ringer switch is on

# AVAudioSession—Music App

- If just doing output
  - Use `AVAudioSessionCategoryPlayback`
- If doing input and output
  - Use `AVAudioSessionCategoryPlayAndRecord`
- Both of these play through the ringer switch

# AVAudioSession—Music App

- Advisable to set `AVAudioSessionCategoryOptionMixWithOthers`
  - Allow other apps to make sound as well
  - Can still play audio in background

```
[session setCategory:AVAudioSessionCategoryPlayAndRecord
            withOptions:AVAudioSessionCategoryOptionMixWithOthers
            error:&outError];
```

# AVAudioSession—Music App

- If you want to be the main app
- Do not set `AVAudioSessionCategoryOptionMixWithOthers`
  - Your app will interrupt other apps that are not mixable when your app goes active
  - Other apps that are mixable will still play

```
[session setCategory:AVAudioSessionCategoryPlayAndRecord error:&outError];
```

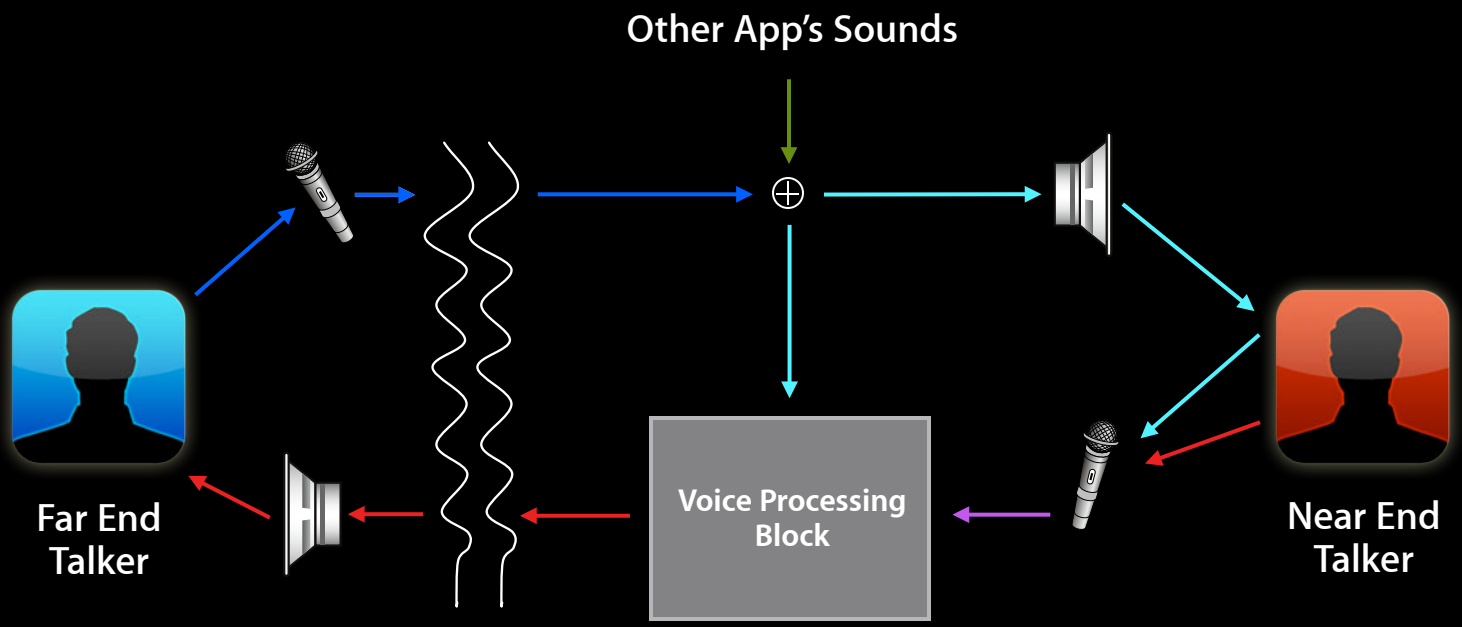
# AVAudioSession Additions

- `[session setPreferredIODuration: error:]`
  - Allows you to request a preferred I/O size
  - Controls the I/O latency of all applications
- `[session setPreferredSampleRate: error:]`
  - Request a preferred sample rate

# Voice Processing Audio Unit

- AUVoiceProcessingIO
  - Extension of AURemoteIO
  - Adds Voice Processing
    - Acoustic Echo Cancellation
    - Noise Suppression
    - Automatic Gain Correction
- Designed for high-quality chat and optimized per route and use case
- Available on iOS, Mac OS X Lion, or later

# How Does Voice Processing Work?



# Voice is Primary (VoIP App)

## AVAudioSession

- Need to be in play and record category
  - Establish the fact you need input and output

```
[session setCategory: AVAudioSessionCategoryPlayAndRecord error:
&outErr];
```

- Set chat mode
  - Establishes routes that are valid for a voice (or video) call

```
[session setMode: AVAudioSessionModeVoiceChat error: &outErr];
```

# Voice is Primary (VoIP App)

## AVAudioSession

- Sample rate should be set to what you need

```
[session setPreferredSampleRate: 24000.0 error: &outErr];
```

  - Preserves the fidelity of the voice
  - If you don't do it, you get whatever the system is set to

- I/O buffer duration can be set to control latency

```
[session setPreferredIODuration: .02 error:]
```

# Other Voice Related Properties

- AVAudioSession
  - Speaker output override (for speakerphone)
- AUVoiceIO properties
  - Defined in the `<AudioUnit/AudioUnitProperties.h>` header file

```
kAUVoiceIOProperty_BypassVoiceProcessing
kAUVoiceIOProperty_VoiceProcessingEnableAGC
kAUVoiceIOProperty_MuteOutput
```

# For GameKit—Game Audio and Chat

- Chat fits in with the game's use of the audio hardware
- Game uses sample rate as appropriate for game
  - 44.1kHz, 48kHz, etc.
- Network (chat) sampling rate: 16kHz
- AUVoiceProcessingIO will take care of matching these sample rates as appropriate

# Summary

- AVAudioSession usage
  - AVAudioPlayer and multichannel playback
- New MultiRoute category
- Using I/O units and AudioSession

# Labs

| | |
|---|---|
| Audio Lab | Graphics, Media & Games Lab D<br>Tuesday 2:00PM |
| Audio Lab | Graphics, Media & Games Lab C<br>Wednesday 2:00PM |

# More Information

**Eryk Vershen**
Media Technologies Evangelist
evershen@apple.com

**Developer Support**
http://developer.apple.com/audio

**Apple Developer Forums**
http://devforums.apple.com


WWDC2012