Audio Development for iPhone OS

Part 1

William Stewart
Core Audio Eng. Mgr.

Introduction

Platform for audio

Media Player

OpenAL

AV Foundation

Audio Toolbox

Audio Unit

Welcome

Audio Development for iPhone OS, Part 1	Mission Wednesday 9:00AM
Fundamentals of Digital Audio for Mac OS X and iPhone OS	Mission Wednesday 10:15AM
Audio Development for iPhone OS, Part 2	Mission Wednesday 11:30AM

AV Foundation

Allan Schaffer
Graphics and Game Technologies Evangelist

AV Foundation Framework

High-level classes for audio

- Audio player (AVAudioPlayer)
  - Provides playback from audio file or memory
- Audio recorder (AVAudioRecorder)
  - Records audio to a file
- Audio session (AVAudioSession)
  - Manage audio behavior and hardware characteristics
- Stream parser (AVPlayer)
  - Play local or streamed audio

AVAudioPlayer

Simplest way to play a sound

- Plays file types supported by AudioFile API, including:
  - caf, m4a, mp3, aif, wav, au, snd, aac
- Provides basic playback operations
  - Create, prepare, play, pause, skip, stop
- Multiple sounds?
  - Use multiple AVAudioPlayer objects
- Supports volume control, metering, looping
  - New—stereo panning, synchronized playback

AVAudioPlayer

Creating a player

- Can be created from a file URL, or NSData in memory

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

// Create the player from NSData
NSData *data = ...
AVAudioPlayer *player =
    [AVAudioPlayer alloc] initWithData:data withError:&error];
```

AVAudioPlayer

Creating a player

- Can be created from a file URL, or NSData in memory

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

// Create the player from NSData
NSData *data = ...
AVAudioPlayer *player =
    [AVAudioPlayer alloc] initWithData:data withError:&error];
```

AVAudioPlayer

Setting properties for playback

- Control of volume, panning, looping, playback position

```
player.volume = 1.0;        // 100% of current system volume
player.pan = -1.0;          // pan to left side
player.numberOfLoops = 2;   // loop 3 times
player.currentTime = 5.0;   // playback position starts 5 seconds in
player.delegate = self;     // delegate
```

- Other properties
  - Metering
  - Duration (read-only)
  - Number of channels (read-only)
  - Play state (read-only)

AVAudioPlayer

Playback controls

- `[player prepareToPlay];`
- `[player play];`
- `[player pause];`
- `[player stop];`

AVAudioPlayer

Playback controls

- `[player prepareToPlay];`
- `[player play];`
- `[player pause];`
- `[player stop];`

- Gets ready to play the sound
  - Allocates buffers
  - Performs priming
- Helps responsiveness of -play:

AVAudioPlayer

Playback controls

- `[player prepareToPlay];`
- `[player play];`
- `[player pause];`
- `[player stop];`

- Starts playing the sound
- Resumes playing if paused or stopped
- Note:
  - Set `currentTime` property to zero to reset playback position

AVAudioPlayer

Playback controls

- `[player prepareToPlay];`
- `[player play];`
- `[player pause];`
- `[player stop];`

- Stops (pauses) playback
- Player remains prepared to play
  - Queue and buffers are still allocated
- Call "play" to resume from where it left off

AVAudioPlayer

Playback controls

- `[player prepareToPlay];`
- `[player play];`
- `[player pause];`
- `[player stop];`

- Stops playback
- Player is no longer “prepared”
  - Queue and buffers are disposed
- To resume:
  - Need to “prepare” again

AVAudioPlayer

Delegate methods

- When certain events happen, your delegate gets called
  - e.g., the player finished playing

```
-(void)audioPlayerDidFinishPlaying:(AVAudioPlayer *)player
                      successfully:(BOOL)flag
```

- Others
  - There was a decode error
  - An interruption began
  - An interruption ended
  - An interruption ended (“with flags”)

AVAudioPlayer

Playing a sound

```
-(void)playAudioFile:(NSURL *)url withError:(NSError **)error
{
  // Create the player
  AVAudioPlayer *player =
      [AVAudioPlayer alloc] initWithContentsOfURL:url withError:error];

  // Set properties
  player.delegate = self;

  // Get ready to play the sound
  [ player prepareToPlay ];

  // Play !
  [ player play ];

}
```

AVAudioPlayer

Playing a sound

```
-(void)playAudioFile:(NSURL *)url withError:(NSError **)error
{
  // Create the player
  AVAudioPlayer *player =
      [AVAudioPlayer alloc] initWithContentsOfURL:url withError:error];

  // Set properties
  player.delegate = self;

  // Get ready to play the sound
  [ player prepareToPlay ];

  // Play !
  [ player play ];

}
```

AVAudioPlayer

Playing a sound

```
-(void)playAudioFile:(NSURL *)url withError:(NSError **)error
{
  // Create the player
  AVAudioPlayer *player =
      [AVAudioPlayer alloc] initWithContentsOfURL:url withError:error];

  // Set properties
  player.delegate = self;

  // Get ready to play the sound
  [ player prepareToPlay ];

  // Play !
  [ player play ];

}
```

AVAudioPlayer

Playing a sound

```
-(void)playAudioFile:(NSURL *)url withError:(NSError **)error
{
  // Create the player
  AVAudioPlayer *player =
      [AVAudioPlayer alloc] initWithContentsOfURL:url withError:error];

  // Set properties
  player.delegate = self;

  // Get ready to play the sound
  [ player prepareToPlay ];

  // Play !
  [ player play ];

}
```

AVAudioRecorder

AVAudioRecorder

- Records sound to a file
  - Records until stopped
  - Or for a specified duration
- Supports a variety of encoding formats
  - AAC
  - ALAC
  - Linear PCM
  - IMA/ADPCM (IMA4)
  - μ-law and a-law
  - iLBC

AVAudioRecorder

API overview

- Creation

```
- (id)initWithURL:(NSURL*)url
         settings:(NSDictionary*)settings
            error:(NSError **)error
```

- Recording controls

```
- (BOOL)prepareToRecord;
- (BOOL)record;
- (BOOL)recordForDuration:(NSInterval)duration;
- (void)pause;
- (void)stop;
```

- Properties

```
settings
recording
currentTime
delegate
```

AVAudioRecorder

Settings dictionary

- Used to specify recording settings
  - What format to record
  - Sample rate
  - Number of channels
  - For LPCM
    - Bit depth, endian-ness
  - For encoded formats
    - Quality, bit rate, etc.

AVAudioRecorder

```
id keys[5], values[5];

keys[0] = AVFormatIDKey;            values[0] = [ NSNumber numberWithInt: kAudioFormatMPEG4AAC ];
keys[1] = AVSampleRateKey;          values[1] = [ NSNumber numberWithFloat: 44100. ];
keys[2] = AVNumberOfChannelsKey;    values[2] = [ NSNumber numberWithInt: 2 ];
keys[3] = AVEncoderBitRateKey;      values[3] = [ NSNumber numberWithInt: 128000 ];
keys[4] = AVEncoderAudioQualityKey; values[4] = [ NSNumber numberWithInt: AVAudioQualityMax ];

// Create the dictionary
NSMutableDictionary* settings = [[NSMutableDictionary alloc] initWithObjects: values
                                                                     forKeys: keys
                                                                       count: 5 ];

// Create the recorder
AVAudioRecorder* recorder = [[AVAudioRecorder alloc] initWithURL: url
                                                        settings: settings
                                                           error: nil ];

// Prepare queues & buffers
[ recorder prepareToRecord ];

// Record!
[ recorder record ];
```

AVAudioRecorder

```
id keys[5], values[5];

keys[0] = AVFormatIDKey;            values[0] = [ NSNumber numberWithInt: kAudioFormatMPEG4AAC ];
keys[1] = AVSampleRateKey;          values[1] = [ NSNumber numberWithFloat: 44100. ];
keys[2] = AVNumberOfChannelsKey;    values[2] = [ NSNumber numberWithInt: 2 ];
keys[3] = AVEncoderBitRateKey;      values[3] = [ NSNumber numberWithInt: 128000 ];
keys[4] = AVEncoderAudioQualityKey; values[4] = [ NSNumber numberWithInt: AVAudioQualityMax ];

// Create the dictionary
NSMutableDictionary* settings = [[NSMutableDictionary alloc] initWithObjects: values
                                                                     forKeys: keys
                                                                       count: 5 ];

// Create the recorder
AVAudioRecorder* recorder = [[AVAudioRecorder alloc] initWithURL: url
                                                        settings: settings
                                                           error: nil ];

// Prepare queues & buffers
[ recorder prepareToRecord ];

// Record!
[ recorder record ];
```

AVAudioRecorder

```
id keys[5], values[5];

keys[0] = AVFormatIDKey;            values[0] = [ NSNumber numberWithInt: kAudioFormatMPEG4AAC ];
keys[1] = AVSampleRateKey;          values[1] = [ NSNumber numberWithFloat: 44100. ];
keys[2] = AVNumberOfChannelsKey;    values[2] = [ NSNumber numberWithInt: 2 ];
keys[3] = AVEncoderBitRateKey;      values[3] = [ NSNumber numberWithInt: 128000 ];
keys[4] = AVEncoderAudioQualityKey; values[4] = [ NSNumber numberWithInt: AVAudioQualityMax ];

// Create the dictionary
NSMutableDictionary* settings = [[NSMutableDictionary alloc] initWithObjects: values
                                                                     forKeys: keys
                                                                       count: 5 ];

// Create the recorder
AVAudioRecorder* recorder = [[AVAudioRecorder alloc] initWithURL: url
                                                        settings: settings
                                                           error: nil ];

// Prepare queues & buffers
[ recorder prepareToRecord ];

// Record!
[ recorder record ];
```

AVAudioRecorder

```
id keys[5], values[5];

keys[0] = AVFormatIDKey;            values[0] = [ NSNumber numberWithInt: kAudioFormatMPEG4AAC ];
keys[1] = AVSampleRateKey;          values[1] = [ NSNumber numberWithFloat: 44100. ];
keys[2] = AVNumberOfChannelsKey;    values[2] = [ NSNumber numberWithInt: 2 ];
keys[3] = AVEncoderBitRateKey;      values[3] = [ NSNumber numberWithInt: 128000 ];
keys[4] = AVEncoderAudioQualityKey; values[4] = [ NSNumber numberWithInt: AVAudioQualityMax ];

// Create the dictionary
NSMutableDictionary* settings = [[NSMutableDictionary alloc] initWithObjects: values
                                                                     forKeys: keys
                                                                       count: 5 ];

// Create the recorder
AVAudioRecorder* recorder = [[AVAudioRecorder alloc] initWithURL: url
                                                        settings: settings
                                                           error: nil ];

// Prepare queues & buffers
[ recorder prepareToRecord ];

// Record!
[ recorder record ];
```

AVAudioRecorder

```
id keys[5], values[5];

keys[0] = AVFormatIDKey;            values[0] = [ NSNumber numberWithInt: kAudioFormatMPEG4AAC ];
keys[1] = AVSampleRateKey;          values[1] = [ NSNumber numberWithFloat: 44100. ];
keys[2] = AVNumberOfChannelsKey;    values[2] = [ NSNumber numberWithInt: 2 ];
keys[3] = AVEncoderBitRateKey;      values[3] = [ NSNumber numberWithInt: 128000 ];
keys[4] = AVEncoderAudioQualityKey; values[4] = [ NSNumber numberWithInt: AVAudioQualityMax ];

// Create the dictionary
NSMutableDictionary* settings = [[NSMutableDictionary alloc] initWithObjects: values
                                                                     forKeys: keys
                                                                       count: 5 ];

// Create the recorder
AVAudioRecorder* recorder = [[AVAudioRecorder alloc] initWithURL: url
                                                        settings: settings
                                                           error: nil ];

// Prepare queues & buffers
[ recorder prepareToRecord ];

// Record!
[ recorder record ];
```

Summary

AVAudioPlayer and AVAudioRecorder

- Feature-rich playback and recording using Objective-C
  - Synchronized playback
  - Flexible recording
- Recommended as the starting point
  - Unless you require:
    - Access to audio samples for processing: use Audio Units
    - Spatial 3D positioning: use OpenAL
    - Networked streaming: use AVPlayer

Audio Session Management

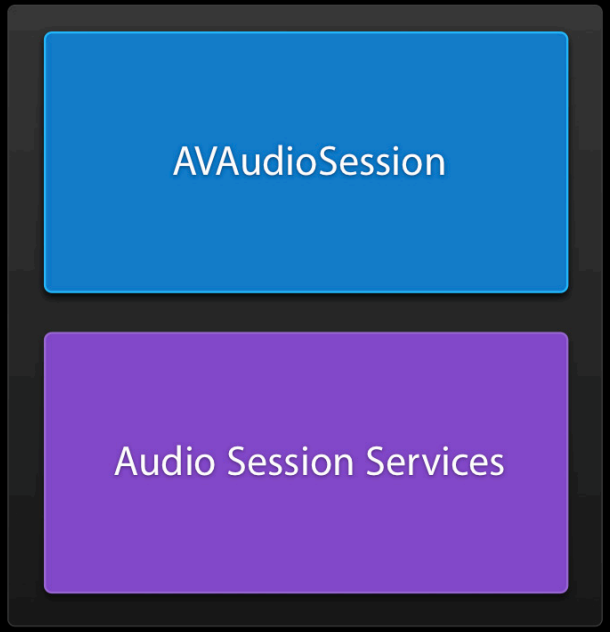
Audio Session Management

Focus on the audio user experience

- How to make your app's sounds:
  - Behave according to user's expectations
  - Be consistent with built-in apps
- Categorize your application
- Mix with background audio
- Respond to interruptions
- Handle routing changes

Two Audio Session APIs

- AVAudioSession class
  - <AVFoundation/AVAudioSession.h>
  - High-level wrapper for the most common functionality
- Audio Session Services
  - <AudioToolbox/AudioServices.h>
  - All of the implementation
  - C Based, lower-level
- Mix and match OK



Using AVAudioSession

Five basic tasks

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

1 Set up the Session

- Retrieve the AVAudioSession instance

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

- Set a delegate for notifications

```
session.delegate = self;
```

- Set preferred hardware settings (optional)

```
[ session setPreferredHardwareSampleRate:44100.0f error:&errRet ];
```

2 Choose and Set a Category

Based on role of audio in your app

Playback

Play and Record

Ambient

Record

Audio Processing

Solo Ambient

Category Name	Intended Usage	Obey Ringer Switch	Obey Screen Lock	Mix with Others	Audio Input	Audio Output	Allowed In Background
 Playback	Audio Players, Video Players			Optional		●	●
 Record	Audio Recorders, Voice Capture				●		●
 Play and Record	VOIP, Voice Chat			Optional	●	●	●
 Audio Processing	Offline Conversion, Offline Processing						●
 Ambient	Games, Productivity Apps	●	●	●		●	
 Solo Ambient	Games, Productivity Apps	●	●			●	

Choose and Set a Category

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

// set a delegate for interruptions & state changes
session.delegate = self;

// Request the “Ambient” category
[ session setCategory:AVAudioSessionCategoryAmbient error:&errRet ];

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

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

// Handle interruptions
...
```

3 Make Session Active

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

// set a delegate for interruptions & state changes
session.delegate = self;

// Request the “Ambient” category
[ session setCategory:AVAudioSessionCategoryAmbient error:&errRet ];

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

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

// Handle interruptions
...
```

3 Make Session Active

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

// set a delegate for interruptions & state changes
session.delegate = self;

// Request the “Ambient” category
[ session setCategory:AVAudioSessionCategoryAmbient error:&errRet ];

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

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

// Handle interruptions
...
```

Quick Tip

Interaction with background audio

- Most apps can just go “Active” and stay that way
- Some apps shouldn’t always be “Active”
  - Recorders
  - VOIP apps
  - Turn-by-turn navigation apps
- These should only be “Active” while in use
  - While recording, on a VOIP call, or announcing the next turn
  - Then become inactive when done

Quick Tip

VOIP/Voice chat app

- Go active when the call begins
  - Interrupts background app playback
- Go inactive when the call ends
  - System will notify background app it can resume

```
- (void)myCallDidFinish
{
    int flags = AVAudioSessionSetActiveFlags_NotifyOthersOnDeactivation;

    // Call is done
    [ session setActive:NO withFlags:flags error:&errRet ];

    // update UI, ...
}
```

Quick Tip

Turn-by-turn Navigation Instructions


```
- (void)setup
{
    UInt32 mix = 1, duck = 1;

    [ session setCategory:AVAudioSessionCategoryPlayback
                    error:&errRet ];

    AudioSessionSetProperty (
        kAudioSessionProperty_OverrideCategoryMixWithOthers,
        sizeof(mix), &mix );

    AudioSessionSetProperty (
        kAudioSessionProperty_OtherMixableAudioShouldDuck,
        sizeof(duck), &duck );
}
```

Quick Tip

Turn-by-turn Navigation Instructions

```
- (void)playThePreloadedInstructions
{
    [ session setActive:YES error:&errRet ];   // duck other audio
    [ myAudioPlayer play ];
}


- (void)audioPlayerDidFinishPlaying:(AVAudioPlayer *)player
                       successfully:(BOOL)flag
{
    [ session setActive:NO error:&errRet ];   // un-duck
}
```

Quick Tip

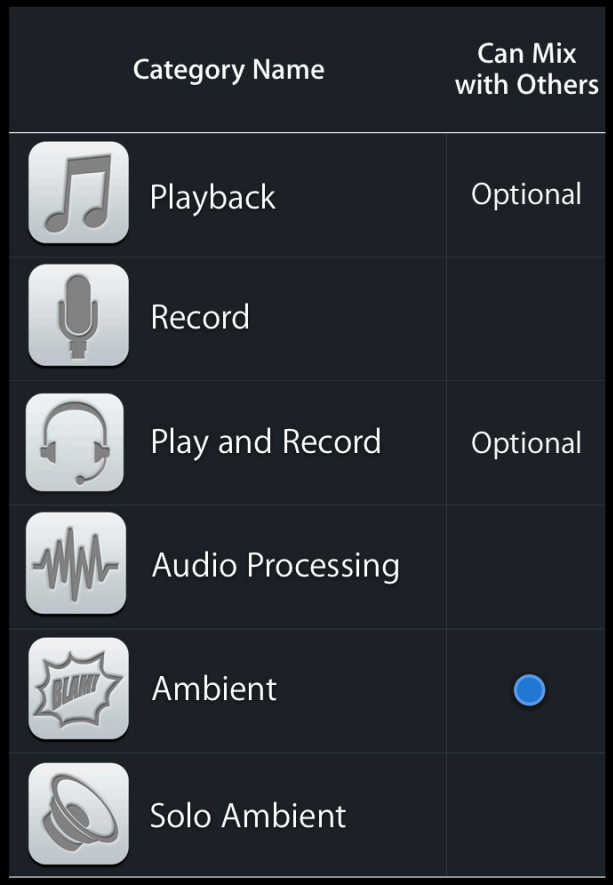
Turn-by-turn Navigation Instructions

```
- (void)playThePreloadedInstructions
{
    [ session setActive:YES error:&errRet ];   // duck other audio
    [ myAudioPlayer play ];
}



- (void)audioPlayerDidFinishPlaying:(AVAudioPlayer *)player
                       successfully:(BOOL)flag
{
    [ session setActive:NO error:&errRet ];   // un-duck
}
```

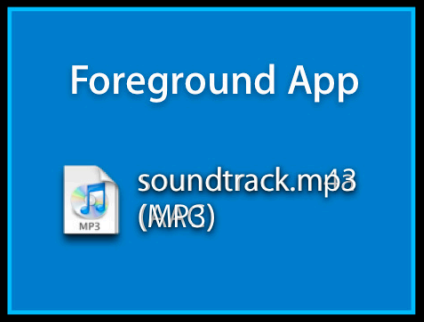
Mixing with Background Audio

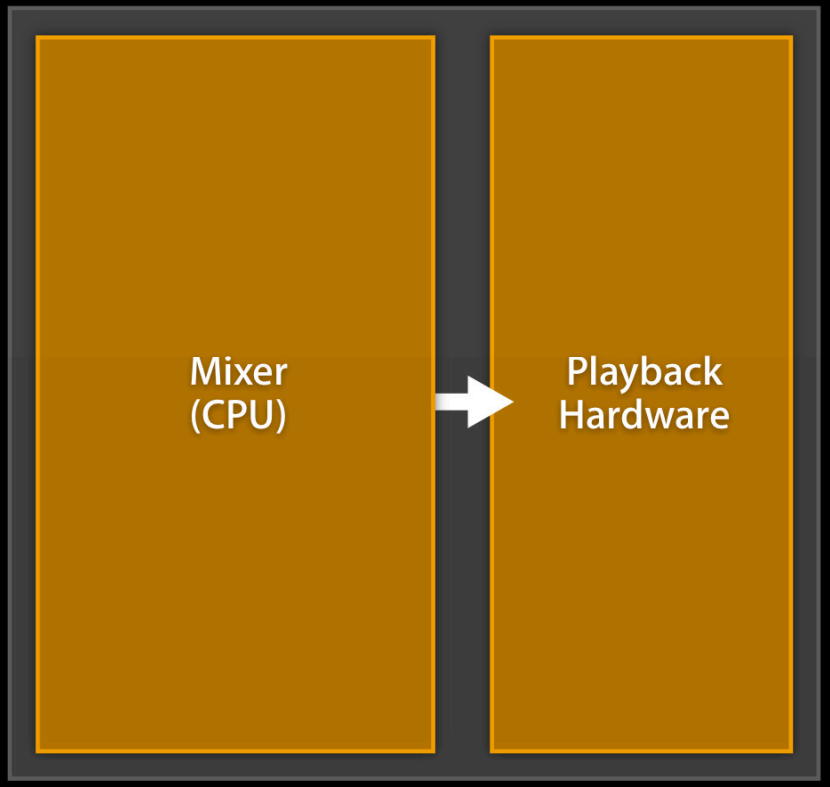
Mixing with Background Audio

- Your application may be playing a variety of sounds
- Another app may be playing its own sounds
- What is heard?
  - Depends on both apps:
    - Whether category can mix with others
    - Whether "Mix with others" override is set
  - "Mixable" or "Non-mixable"

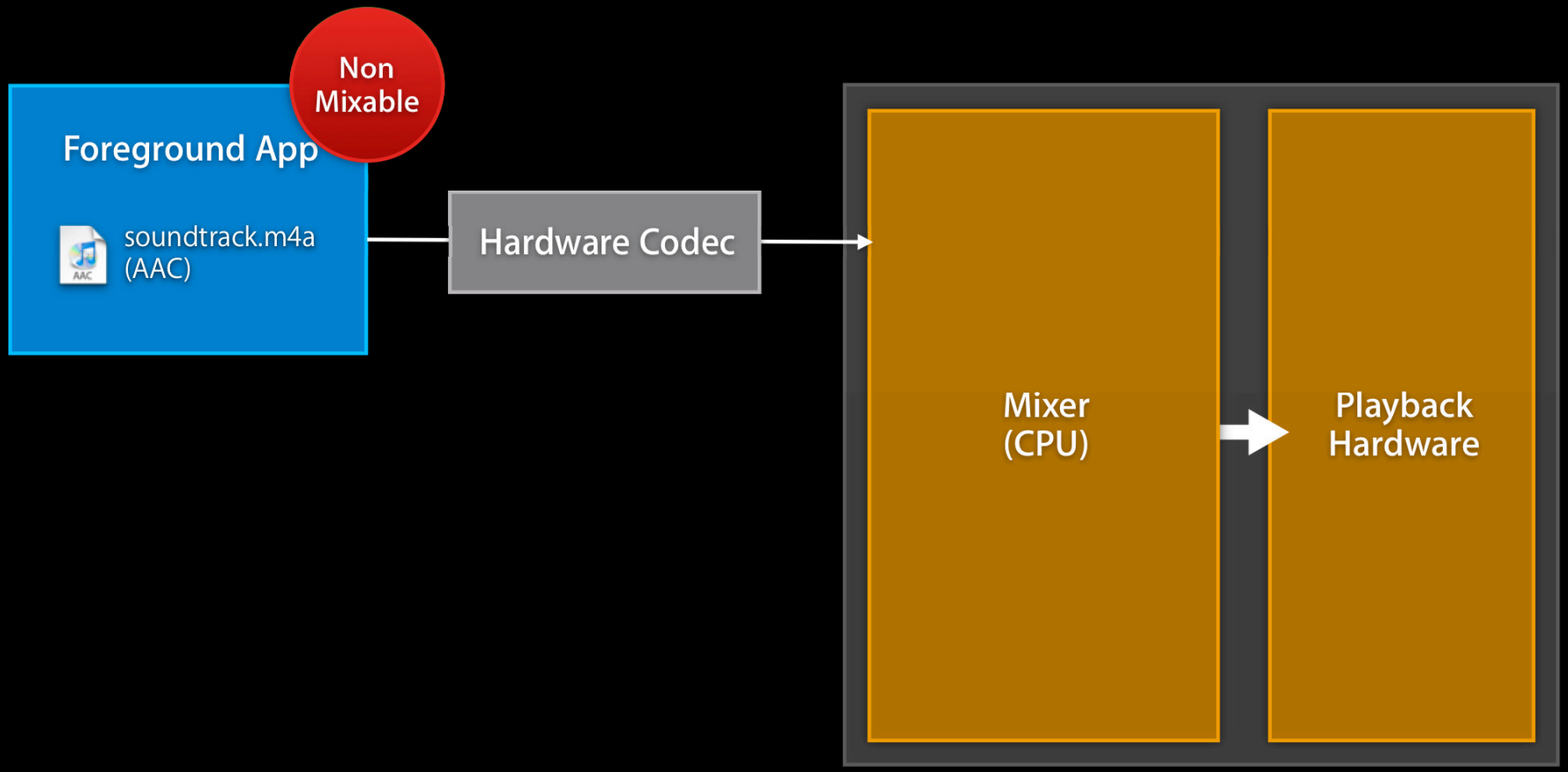


Mixing with Background Audio

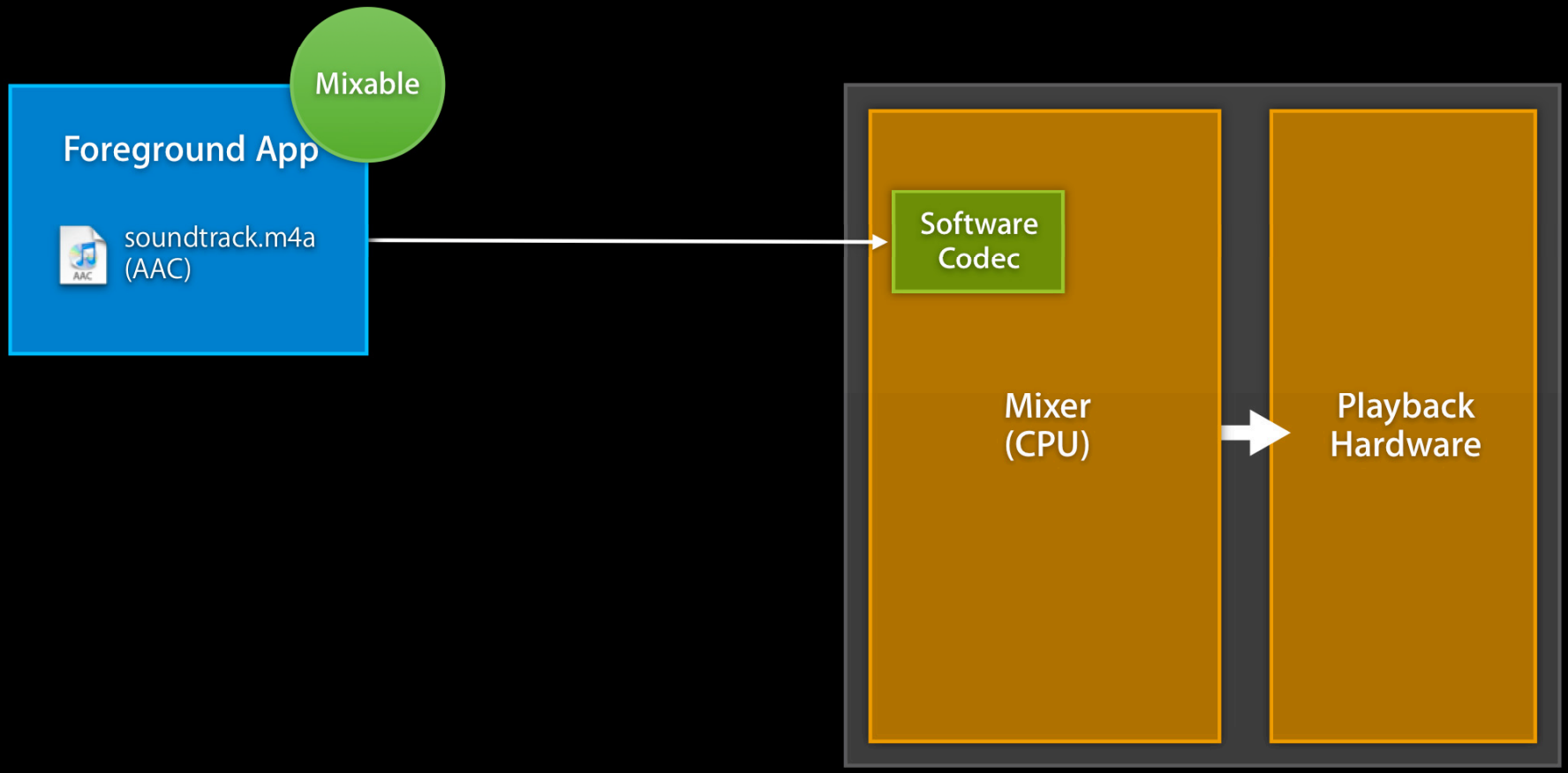




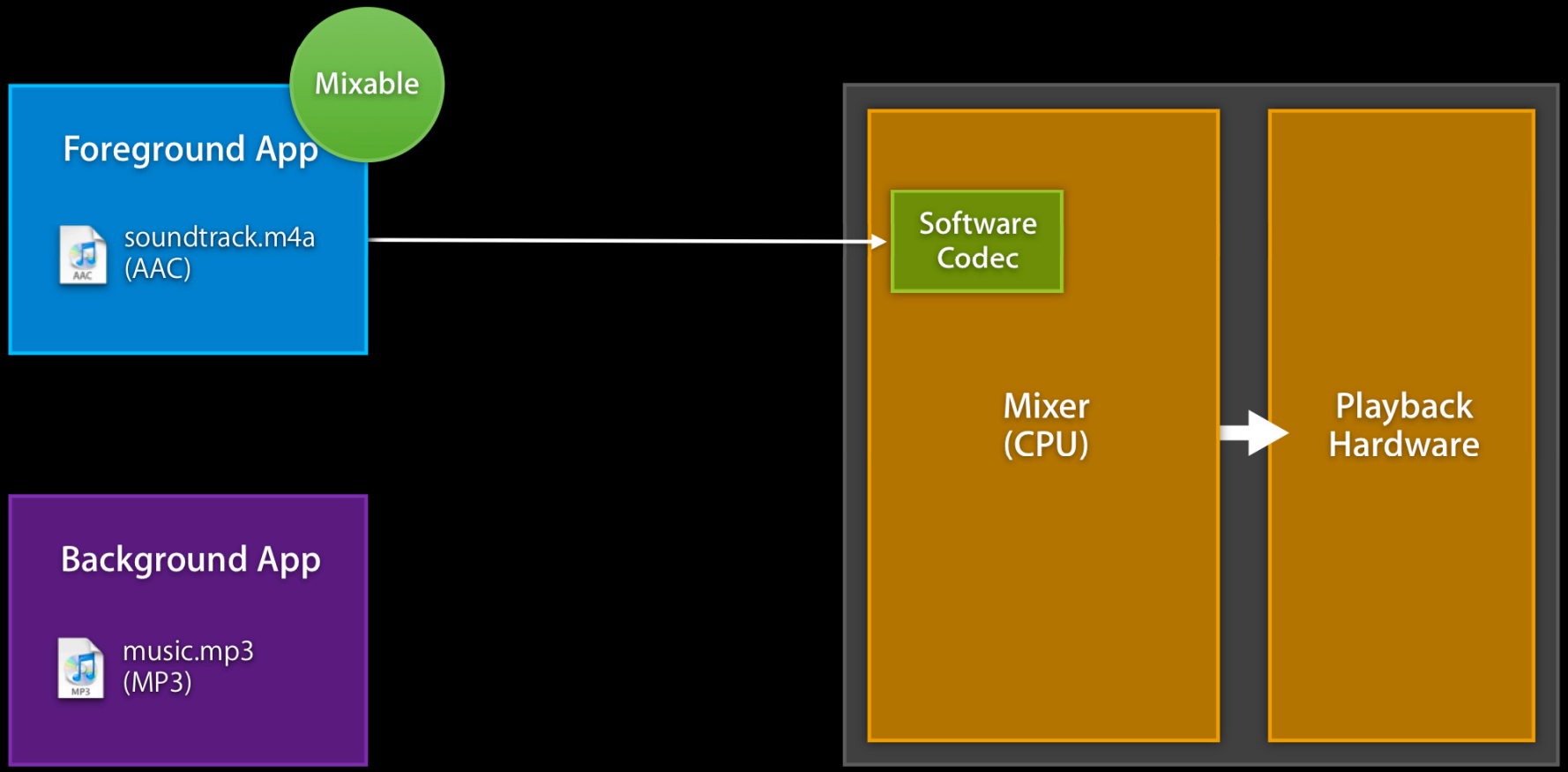
Mixing with Background Audio



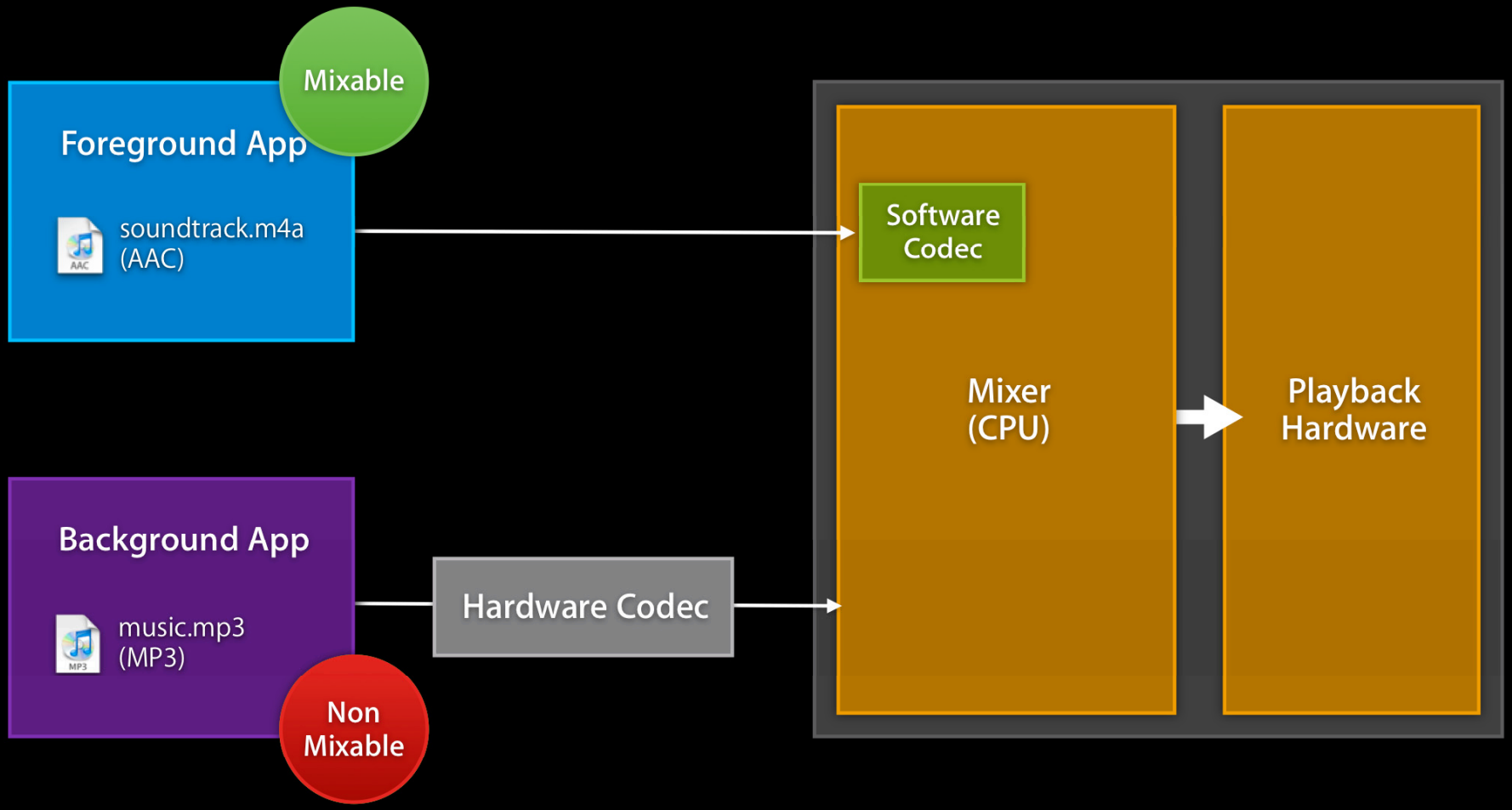
Mixing with Background Audio



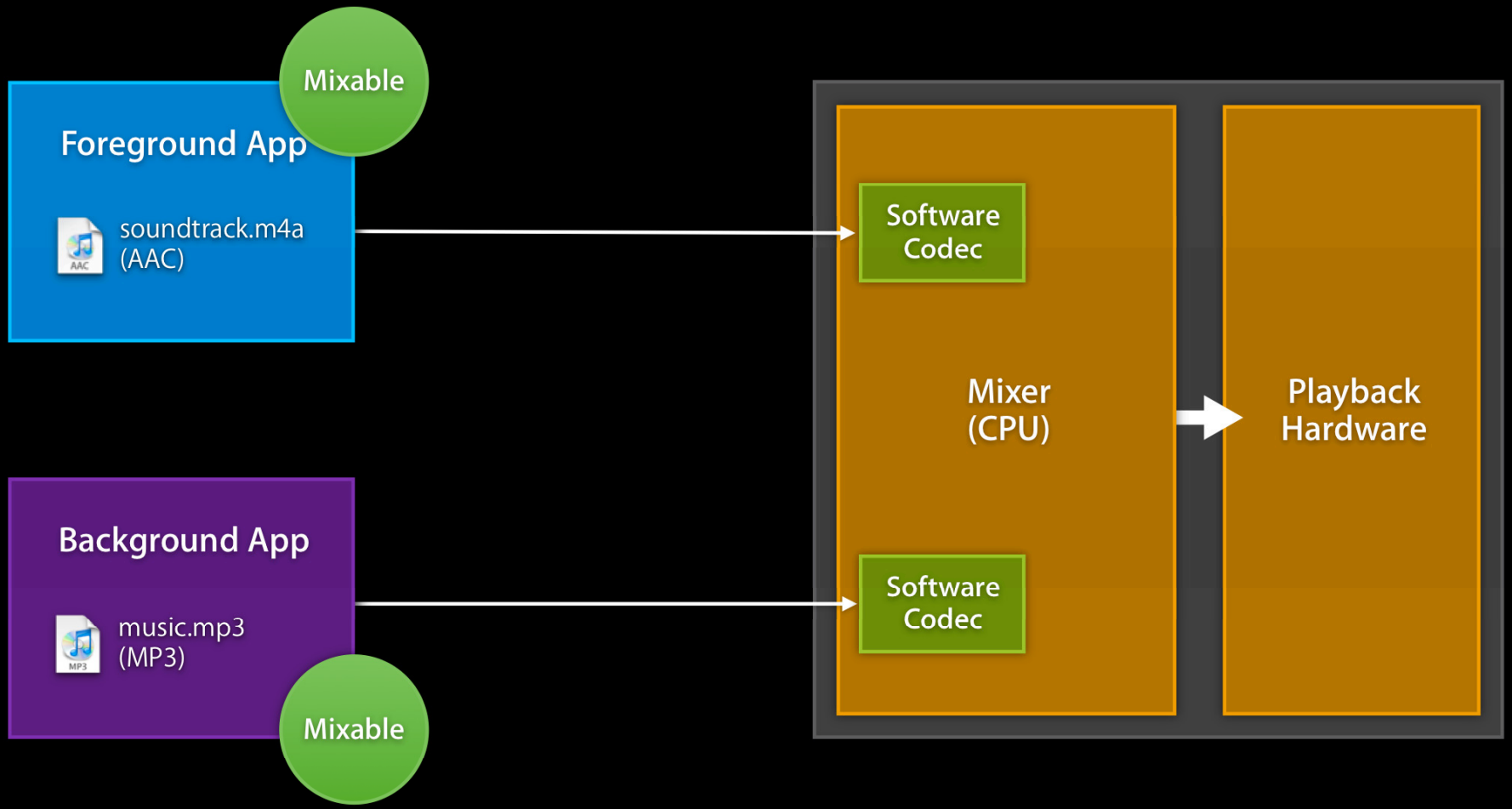
Mixing with Background Audio



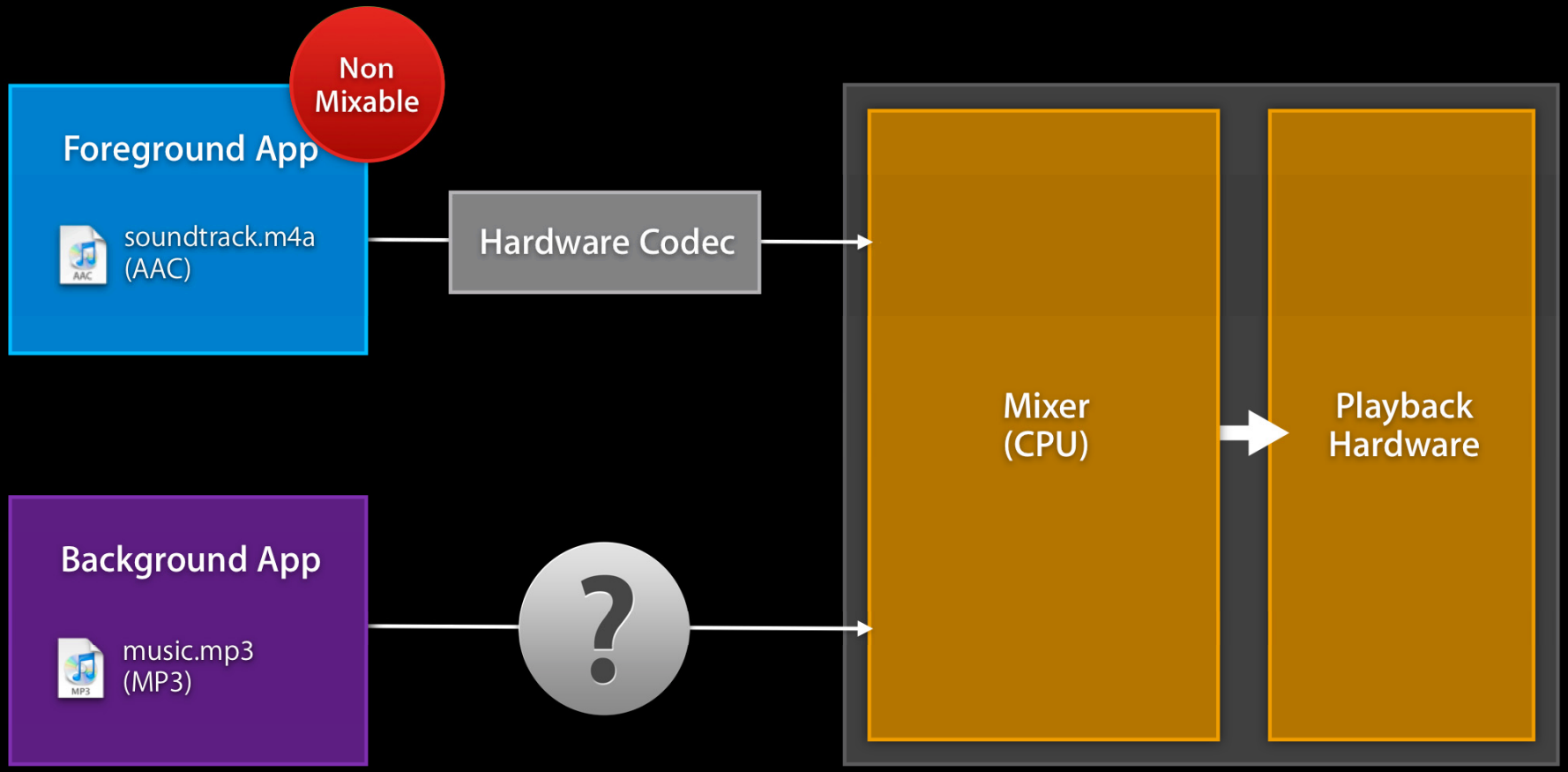
Mixing with Background Audio



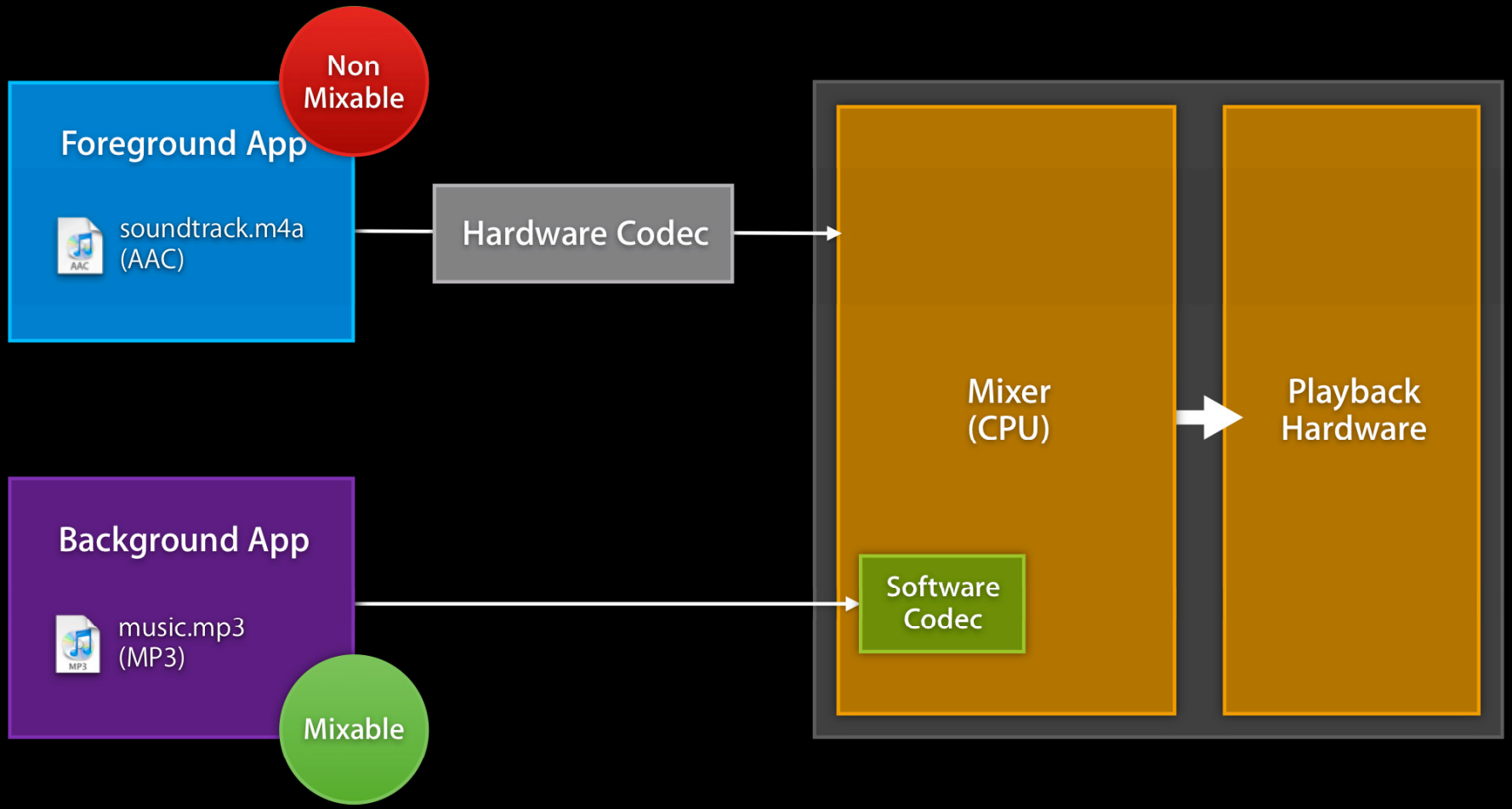
Mixing with Background Audio



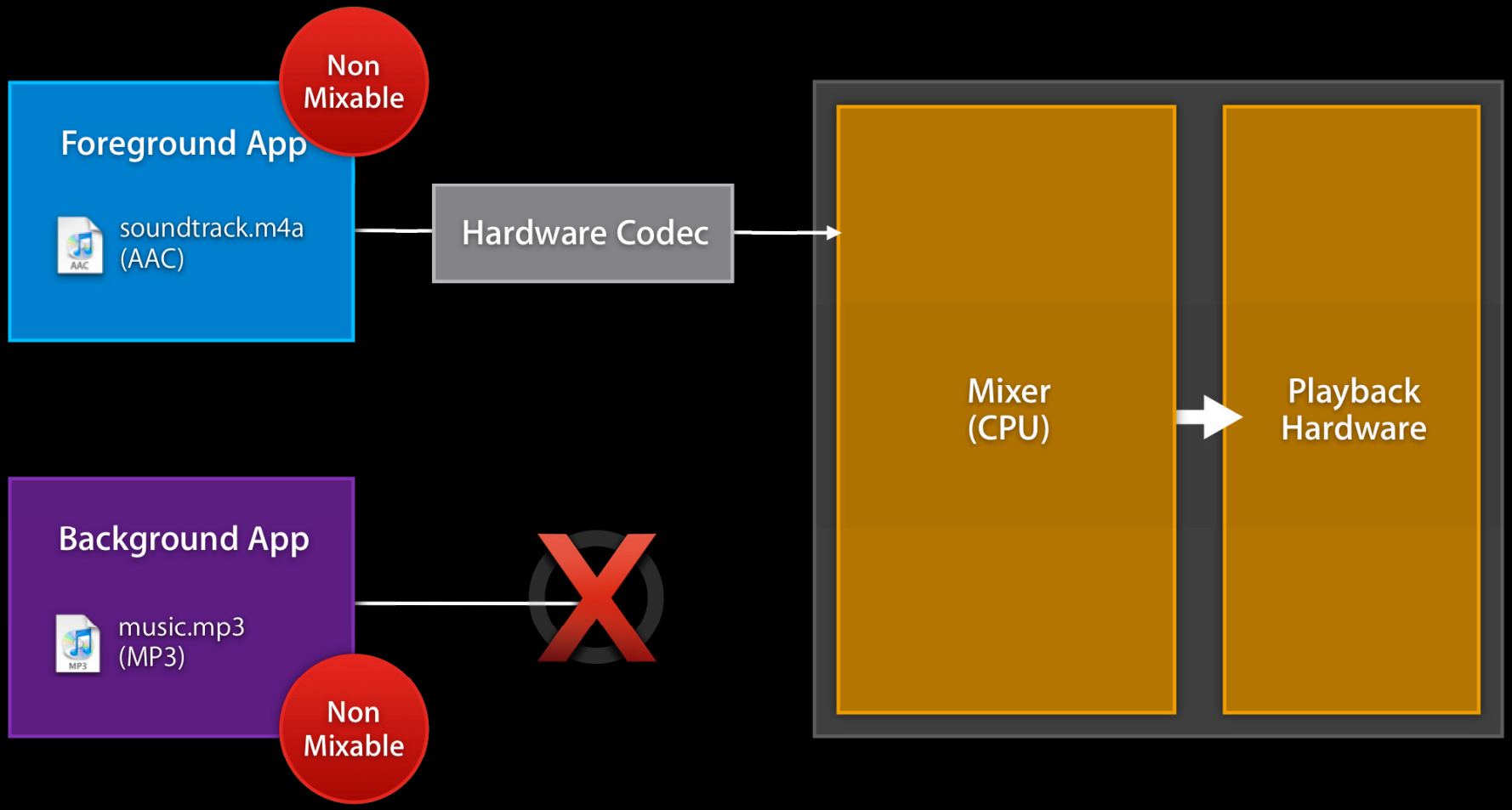
Mixing with Background Audio



Mixing with Background Audio

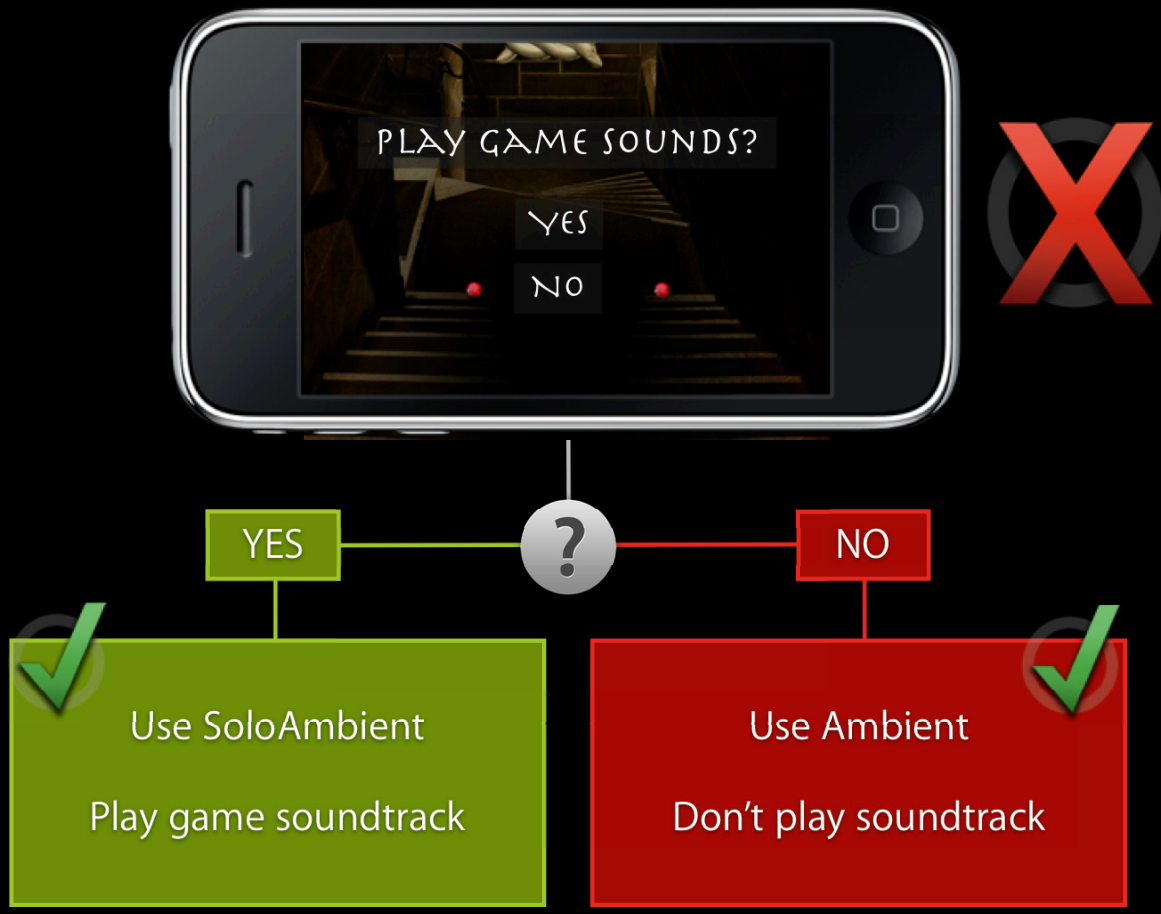


Mixing with Background Audio



Detecting Background Audio


PLAY GAME SOUNDS?
YES
NO


PLAY GAME SOUNDS?
YES
NO
YES
?
NO
Use SoloAmbient
Play game soundtrack
Use Ambient
Don't play soundtrack

Detecting Background Audio

- kAudioSessionProperty_OtherAudioIsPlaying
  - Use this to decide
  - Play game music or not?
  - Which category to use?

```
UInt32 otherAudioIsPlaying;
UInt32 propertySize = sizeof(otherAudioIsPlaying);

AudioSessionGetProperty(
    kAudioSessionProperty_OtherAudioIsPlaying,
    &propertySize, &otherAudioIsPlaying);
```

  - Important: Check again in -applicationDidBecomeActive:

MPMoviePlayerController

Quick Tip

MPMoviePlayerController

Behavior change

- Prior to iPhone OS 3.2
  - MPMoviePlayerController had its own session
    - Interrupted your session
    - Silenced other audio
- In iPhone OS 3.2 and above
  - MPMoviePlayerController uses your audio session
    - Inherits your category setting
    - Behavior consistent with your app
    - Doesn't interrupt you, be ready!
  - See `@property BOOL useApplicationAudioSession;`

Using AVAudioSession

Five basic tasks

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

Interruptions

Responding to Interruptions

- Session may be interrupted by higher-priority audio
  - Phone call, clock alarm, foreground app
- Interruption makes your session inactive
  - Currently playing audio is stopped
- After the interruption is over
  - Reactivate certain state (API-specific)
  - Then become active again (if appropriate)

Responding to Interruptions

AVAudioSession delegates

- `- (void)beginInterruption`
  - Playback has stopped. Already inactive
  - Change state of UI, etc., to reflect non-playing state
    - For instance, change play button to its stopped state
- `- (void)endInterruptionWithFlags:`
  - `AVAudioSessionInterruptionFlags_ShouldResume`
    - Make session active
    - Update user interface
    - Resume playback

Responding to Interruptions

OpenAL and AVAudioSession

- Invalidate context when interrupted
- Make context current again upon interruption end

```
- (void)beginInterruption {
    alcMakeContextCurrent (NULL);
}

- (void)endInterruptionWithFlags:(NSUInteger)flags
{
    if (flags == AVAudioSessionInterruptionFlags_ShouldResume)
    {
        [session setActive:YES error:nil];
        alcMakeContextCurrent (myContext);
    }
    ...
}
```

Responding to Interruptions

AVAudioSession and AudioQueue

- Save playback or recording position
- If using a hardware codec
  - Dispose currently playing AudioQueue
  - Create and start a new AudioQueue when ready
- If using a software codec
  - No need to dispose
  - Just restart the queue when ready
    - `AudioQueueStart()`

Routing

Audio Routing

What users expect

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

Audio Routing

Topics

- Querying the route
- Listening for changes
- Redirecting the audio output route

Querying the Route

What is the current route?

Audio Session Services API

- kAudioSessionProperty_AudioRoute
  - CFStringRef with name of current route

    e.g. @“Speaker”, @”Headphone”, @“Receiver”
    - @“” for none

```
CFStringRef route;
UInt32 size = sizeof(route);

AudioSessionGetProperty( kAudioSessionProperty_AudioRoute,
     &size, &currentRoute);

NSLog(@“Route is %@”, currentRoute);
```

Listening for Changes

Where'd the route go?

- `AudioSessionAddPropertyListener()`
  - Register for notifications for when a route changes
- `kAudioSessionProperty_AudioRouteChange`
  - Notification for when the route changes
    - Reason why route changed
    - What was the old route

```
AudioSessionAddPropertyListener (
    kAudioSessionProperty_AudioRouteChange,
    MyPropListener, &clientData );
```

Audio Route Change Notification

```
void MyPropListener (void* clientData,
                     AudioSessionPropertyID inID,
                     UInt32 dataSize,
                     const void* inData)
{
    CFDictionaryRef dict = (CFDictionaryRef)inData;

    CFNumberRef reason = CFDictionaryGetValue(dict,
        CFSTR(kAudioSession_AudioRouteChangeKey_Reason));

    CFStringRef oldRoute = CFDictionaryGetValue(dict,
        CFSTR(kAudioSession_AudioRouteChangeKey_OldRoute));
}
```

Redirecting Output Audio

Overriding the output audio route

- kAudioSessionProperty_OverrideAudioRoute
  - "Speaker"
    - Routes output to the speaker
  - "None"
    - Go back to the receiver
  - Only valid with category PlayAndRecord

```
UInt32 override = kAudioSessionOverrideAudioRoute_Speaker;
AudioSessionSetProperty (
    kAudioSessionProperty_OverrideAudioRoute,
    sizeof(override), &override );
```

Using AVAudioSession

Summary

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

Related Sessions

Fundamentals of Digital Audio for Mac OS X and iPhone OS	Mission Wednesday 10:15AM
Audio Development for iPhone OS, Part 2	Mission Wednesday 11:30AM

Labs

Audio Lab	Graphics & Media Lab C Wednesday 2:00PM
Audio Lab	Graphics & Media Lab B Thursday 9:00AM

More Information

Allan Schaffer
Graphics and Game Technologies Evangelist
aschaffer@apple.com

Eryk Vershen
Media Technologies Evangelist
evershen@apple.com

Audio Session Programming Guide
iPhone Dev Center
http://developer.apple.com/iphone

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


WWDC10