AVAudioEngine in Practice

Session 502
Kapil Krishnamurthy
Core Audio Rock Star
Overview

Core Audio overview
AVAudioEngine
• Goals
• Features
• Building blocks
• Gaming and 3D audio
Core Audio Overview
iOS and Mac OS X Audio Stack

Multiple APIs for implementing audio features
Low latency, real-time audio
iOS and Mac OS X Audio Stack

Multiple APIs for implementing audio features
Low latency, real-time audio
Multiple APIs for implementing audio features

Low latency, real-time audio

AVAudioEngine—New to Mac OS X 10.10, iOS 8.0

AV Audio Utility classes—Refer to session 501 (What’s new in Core Audio)
Goals

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

Objective-C API set
Low latency, real-time audio

Features

• Read and write audio files
• Play audio using files and buffers
• Record audio
• Connect audio processing blocks
• Capture audio at any point in the processing chain
• Implement 3D audio for games
Sample Use Case One

Karaoke

- Microphone
- Delay
- Backing Track Player
- Mixer
- Speaker
Sample Use Case One
Karaoke

- Microphone
- Delay
- Backing Track Player
- Sound Effect Player
- Analyze Pitch
- Mixer
- Speaker
Sample Use Case Two

Streaming audio

- Player
- Buffer 1
- Buffer 2
- Buffer 3
- EQ
- Speaker
AVAudioEngine
Building blocks
Building Blocks

Engine (AVAudioEngine)
Node (AVAudioNode)
  • Output node (AVAudioOutputNode)
  • Mixer node (AVAudioMixerNode)
  • Player node (AVAudioPlayerNode)
Engine

AVAudioEngine class

Engine manages graph of audio nodes
Use the engine to set up connections between nodes
Start/stop the engine
Allows dynamic node configuration
Engine Workflow
AVAudioEngine class

Create an engine
Create nodes
Attach nodes to the engine
Connect the nodes together
Start the engine
Node

AVAudioNode class

Nodes are audio blocks

- Source—Player, microphone
- Process—Mixer, effect
- Destination—Speaker

N inputs/M outputs, specified as busses

Every bus has an audio data format
Active Chain

Node connections

Active chain => Source node to destination node
Active Chain
Node connections

Active chain => Source node to destination node
Inactive Chain
Node connections

Disconnected nodes are in an inactive state

Source Node (Player) → Processing Node → Destination Node
Inactive Chain
Node connections

Disconnected nodes are in an inactive state
Output Node

AVAudioOutputNode class

Engine has an implicit destination node called output node
Output node provides audio data to the output hardware
Standalone instance cannot be created
Mixer Node

AVAudioMixerNode class

Processing node that mixes N inputs to a single output
Mixer Node

AVAudioMixerNode class

Processing node that mixes N inputs to a single output
Volume control over each input bus
Mixer Node

AVAudioMixerNode class

Processing node that mixes N inputs to a single output
Volume control over each input bus
Volume control of output bus
Sub-Mixing

AVAudioMixerNode class
Mixer Node

AVAudioMixerNode class

Engine has an implicit mixer node
Additional mixer nodes can be created and attached to the engine
Mixer inputs can have different audio formats
AVAudioEngine Architecture

App

AVAudioEngine
AVAudioEngine Architecture
AVAudioEngine Architecture

App

AVAUDIO-engine

AVAUDIOPlayerNode (Source)

AVAUDIOEngine

AVAUDIOPlayerNode (Process)

AVAUDIOOutputNode (Destination)
AVAudioEngine Architecture

App

AVAudioEngine

 AVAudioPlayerNode (Source)

 AVAudioMixerNode (Process)

 AVAudioOutputNode (Destination)
Pushing Data on the Render Thread

Running engine = active render thread
Use player nodes to push data
Player Node

AVAudioPlayerNode class

Player nodes play data from files and buffers
Schedule events—Play data at a specified time
Schedule buffers
  • Multiple buffers with individual callbacks
  • Single buffer that loops
Schedule files and file segments
Player Node—Single Buffer

[Diagram of AVAudioPlayerNode, AVAudioMixerNode, and AVAudioOutputNode connected by arrows to form a process flow.]
Player Node—Single Buffer
Player Node—Single Buffer

AVAudioPlayerNode (Source) → AVAudioMixerNode (Process) → AVAudioOutputNode (Destination)

AVAudioBuffer

AVAudioEngine

App
Player Node—Single Buffer

AVAudioEngine

AVAudioPlayerNode (Source)

AVAudioMixerNode (Process)

AVAudioOutputNode (Destination)

AVAudioBuffer
Player Node—Single Buffer

AVAudioPlayerNode (Source) → AVAudioMixerNode (Process) → AVAudioOutputNode (Destination)

AVAudioEngine

App
Player Node—File

AVAudioPlayerNode (Source) -> AVAudioMixerNode (Process) -> AVAudioOutputNode (Destination)

AVAudioEngine

App

AVAudioFile
AVAudioEngine
Code example
AVAudioEngine *engine = [[AVAudioEngine alloc] init];

AVAudioPlayerNode *player = [[AVAudioPlayerNode alloc] init];

[engine attachNode:player];
Set up Player (File)

Code example

AVAudioFile *file = [[AVAudioFile alloc] initForReading:fileURL error:&error];

AVAudioMixerNode *mainMixer = [engine mainMixerNode];
[engine connect:player to:mainMixer format:file.processingFormat];

[player scheduleFile:file atTime:nil completionHandler:nil];
Set up Player (File)

Code example

AVAudioFile *file = [[AVAudioFile alloc] initForReading:fileURL error:&error];

AVAudioMixerNode *mainMixer = [engine mainMixerNode];
[engine connect:player to:mainMixer format:file.processingFormat];

[player scheduleFile:file atTime:nil completionHandler:nil];
Set up Player (File)

Code example

AVAudioFile *file = [[AVAudioFile alloc] initForReading:fileURL error:&error];

AVAudioMixerNode *mainMixer = [engine mainMixerNode];
[engine connect:player to:mainMixer format:file.processingFormat];

[player scheduleFile:file atTime:nil completionHandler:nil];
Set up Player (Buffer)

Code example

AVAudioPCMBuffer *buffer = ...

AVAudioMixerNode *mainMixer = [engine mainMixerNode];
[engine connect:player to:mainMixer format:buffer.format];

[player scheduleBuffer:buffer atTime:nil options:nil completionHandler:nil];
Set up Player (Buffer)

Code example

AVAudioPCMBuffer *buffer = ...

AVAudioMixerNode *mainMixer = [engine mainMixerNode];
[engine connect:player to:mainMixer format:buffer.format];

[player scheduleBuffer:buffer atTime:nil options:nil completionHandler:nil];
Start Engine and Play

Code example

NSError *error;
[engine startAndReturnError:&error]

[player play];
Start Engine and Play

Code example

NSError *error;
[engine startAndReturnError:&error]

[player play]
Buffer Schedule Options
AVAudioPlayerNode class

1. Play buffer now

[player scheduleBuffer:buffer atTime:nil options:nil completionHandler:nil];
[player play];
Buffer Schedule Options

AVAudioPlayerNode class

2. Append new buffer

[player scheduleBuffer:buffer atTime:nil options:nil completionHandler:nil];
[player play]
[player scheduleBuffer:buffer2 atTime:nil options:nil completionHandler:nil];
Buffer Schedule Options

AVAudioPlayerNode class

3. Interrupt with new buffer

[player scheduleBuffer:buffer atTime:nil options:nil completionHandler:nil];
[player play]
[player scheduleBuffer:buffer2 atTime:nil options:AVAudioPlayerNodeBufferInterrupts completionHandler:nil];
Buffer Schedule Options
AVAUDIOPLAYERNODE class

4. Loop single buffer

[player scheduleBuffer:buffer atTime:nil options:AVAUDIOPLAYERNODEBUFFERLOOPS completionHandler:nil];
[player play];
5. Interrupt looping buffer

[![player scheduleBuffer:buffer atTime:nil options:AVAudioPlayerNodeBufferLoops completionHandler:nil];
[player play];
[player scheduleBuffer:buffer atTime:nil
options:AVAudioPlayerNodeBufferInterrupts completionHandler:nil];
Buffer Schedule Options

AVAudioPlayerNode class

6. Interrupt looping buffer after current loop finishes

[player scheduleBuffer:buffer atTime:nil options:AVAudioPlayerNodeBufferLoops completionHandler:nil];
[player play];
[player scheduleBuffer:buffer atTime:nil options:AVAudioPlayerNodeBufferInterruptsAtLoop completionHandler:nil];
6. Interrupt looping buffer after current loop finishes

[player scheduleBuffer:buffer atTime: nil options: AVAudioPlayerNodeBufferLoops completionHandler: nil];
[player play];
[player scheduleBuffer:buffer atTime: nil options: AVAudioPlayerNodeBufferInterruptsAtLoop completionHandler: nil];
Buffer Looping Example
AVAudioPlayerNode class

Attack, Sustain, Release
Buffer Looping Example
AVAudioPlayerNode class

Attack, Sustain, Release

[player scheduleBuffer:attackBuffer atTime:nil options:nil completionHandler:nil];

[player scheduleBuffer:sustainBuffer atTime:nil options:AVAudioPlayerNodeBufferLoops completionHandler:nil];

[player play];

// after some time
[player scheduleBuffer:releaseBuffer atTime:nil options:AVAudioPlayerNodeBufferInterruptsAtLoop completionHandler:nil];
Buffer Looping Example

AVAudioPlayerNode class

Attack, Sustain, Release

[player scheduleBuffer:attackBuffer atTime:nil options:nil completionHandler:nil];

[player scheduleBuffer:sustainBuffer atTime:nil options:AVAudioPlayerNodeBufferLoops completionHandler:nil];

[player play];

// after some time
[player scheduleBuffer:releaseBuffer atTime:nil options:AVAudioPlayerNodeBufferInterruptsAtLoop completionHandler:nil];
Buffer Looping Example
AVAudioPlayerNode class

Attack, Sustain, Release

[player scheduleBuffer:attackBuffer atTime:nil options:nil completionHandler:nil];

[player scheduleBuffer:sustainBuffer atTime:nil options:AVAudioPlayerNodeBufferLoops completionHandler:nil];

[player play];

// after some time
[player scheduleBuffer:releaseBuffer atTime:nil options:AVAudioPlayerNodeBufferInterruptsAtLoop completionHandler:nil];
Buffer Looping Example

AVAudioPlayerNode class

Attack, Sustain, Release

[player scheduleBuffer:attackBuffer atTime:nil options:nil completionHandler:nil];

[player scheduleBuffer:sustainBuffer atTime:nil options:AVAudioPlayerNodeBufferLoops completionHandler:nil];

[player play];

// after some time
[player scheduleBuffer:releaseBuffer atTime:nil options:AVAudioPlayerNodeBufferInterruptsAtLoop completionHandler:nil];
Schedule to Play in the Future

AVAudioPlayerNode class

Buffers and files can be scheduled to play in the future

AVAudioTime *tenSecsInTheFuture = [AVAudioTime timeWithSampleTime:(10 * buffer.format.sampleRate) atRate:buffer.format.sampleRate];

[player scheduleBuffer:buffer atTime: tenSecsInTheFuture options:nil completionHandler:nil];

[player play];
Schedule to Play in the Future

AVAudioPlayerNode class

Buffers and files can be scheduled to play in the future

AVAudioTime *tenSecsInTheFuture = [AVAudioTime timeWithSampleTime:(10 * buffer.format.sampleRate) atRate:buffer.format.sampleRate];

[player scheduleBuffer:buffer atTime: tenSecsInTheFuture options:nil completionHandler:nil];

[player play];
Node Tap

AVAudioNode class

Node tap captures the output of a node

• Record microphone data
• Record a live performance in a music application
• Capture the output mix of a game

One tap per node’s output bus

Captured data is returned in a callback block
Node Tap

AVAudioPlayerNode → AVAudioMixerNode (Process) → AVAudioOutputNode (Destination)

AVAudioPlayerNode

AVAudioPlayerNode
Node Tap

App

AVAudioPlayerNode

AVAudioMixerNode (Process)

AVAudioOutputNode (Destination)

Tap
Node Tap

AVAudioPlayerNode

AVAudioMixerNode (Process)

AVAudioOutputNode (Destination)

Tap

AVAudioBuffer
Install Tap

Code example

[mixer installTapOnBus:0 bufferSize:4096 format:[mixer outputFormatForBus:0] block:^(_buffer, _when) {
    // perform operation using data
}]
Audio Data and the Render Thread

Players push data onto the render thread
Node taps pull data from the render thread
Getting the Mic in the Mix

AVAudioInputNode class

Engine has an implicit input node
Input node receives audio data from the input hardware
Data is pulled in an active connection chain
Standalone instance cannot be created
Connect Input Node

[Diagram showing AVAudioEngine with AVAudioInputNode (Source), AVAudioMixerNode (Process), and AVAudioOutputNode (Destination)]
Engine is running, input node is active
Data is pulled from the input node
Connect Input Node

Code example

AVAudioInputNode *input = [engine inputNode];

[engine connect:input to:mixer format:[input inputFormatForBus:0]];

[engine startAndReturnError:&error];
Connect Input Node

Code example

AVAudioInputNode *input = [engine inputNode];

[engine connect:input to:mixer format:[input inputFormatForBus:0]];

[engine startAndReturnError:&error];
Connect Input Node

Code example

AVAudioInputNode *input = [engine inputNode];

[engine connect:input to:mixer format:[input inputFormatForBus:0]];  

[engine startAndReturnError:&error];
Disconnect Input Node

Engine is running but input node is inactive
Data will not be pulled from the input node
Disconnect Input Node

Code example

[engine disconnectNodeOutput:input]
Capturing Input

AVAudioInputNode class

Use a node tap directly with the input node
Data is pulled when engine is running
Effect Nodes

AVAudioUnitEffect, AVAudioUnitTimeEffect

Effects are nodes that process data
Two main categories
• AVAudioUnitEffect
• AVAudioUnitTimeEffect
# Effect Nodes

<table>
<thead>
<tr>
<th>AVAudioUnitEffect</th>
<th>AVAudioUnitTimeEffect</th>
</tr>
</thead>
<tbody>
<tr>
<td>N frames in, N frames out</td>
<td>X frames in, Y frames out</td>
</tr>
<tr>
<td>Can be connected to input node</td>
<td>Cannot be connected to input node</td>
</tr>
</tbody>
</table>
Currently Available Effects

<table>
<thead>
<tr>
<th>AVAudioUnitEffect</th>
<th>AVAudioUnitTimeEffect</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVAudioUnitDelay</td>
<td>AVAudioUnitVarispeed</td>
</tr>
<tr>
<td>AVAudioUnitDistortion</td>
<td>AVAudioUnitTimePitch</td>
</tr>
<tr>
<td>AVAudioUnitEQ</td>
<td></td>
</tr>
<tr>
<td>AVAudioUnitReverb</td>
<td></td>
</tr>
</tbody>
</table>
Effect Node Example

App

AVAudioEngine

AVAudioPlayerNode (Source)

AVAudioUnitEQ (Process)

AVAudioOutputNode (Destination)

AVAudioFile
EQ Node Setup

Effect node code example

```objective-c
AVAudioUnitEQ *eq = [[AVAudioUnitEQ alloc] initWithNumberOfBands:2];

AVAudioUnitEQFilterParameters *filterParameters = eq.bands[0];
filterParameters.filterType = AVAudioUnitEQFilterTypeHighPass;
filterParameters.frequency = 80;

filterParameters = eq.bands[1];
filterParameters.filterType = AVAudioUnitEQFilterTypeParametric;
filterParameters.frequency = 500;
filterParameters.bandwidth = 2.0;
filterParameters.gain = 4.0;
```
EQ Node Setup

Effect node code example

AVAudioUnitEQ *eq = [[AVAudioUnitEQ alloc] initWithNumberOfBands:2];

AVAudioUnitEQFilterParameters *filterParameters = eq.bands[0];
filterParameters.filterType = AVAudioUnitEQFilterTypeHighPass;
filterParameters.frequency = 80;

filterParameters = eq.bands[1];
filterParameters.filterType = AVAudioUnitEQFilterTypeParametric;
filterParameters.frequency = 500;
filterParameters.bandwidth = 2.0;
filterParameters.gain = 4.0;
Connect and Play

Effect node code example

AVAudioFile *file = ...
AVAudioPlayerNode *player = ...

[engine connect:player to:eq format:file.processingFormat];
[engine connect:eq to:[engine outputNode] format:file.processingFormat];
Demo
Channel strip

Kapil Krishnamurthy
Core Audio Rock Star
Mixer Input Bus Settings

AVAudioMixing protocol

AVAudioMixing protocol defines mixer input bus settings
Source nodes conform to this protocol
- Player node
- Input node

Protocol properties take effect when connected to a mixer node
When not connected to a mixer
- Changes to protocol property values are cached
Mixer Input Bus Settings

AVAudioMixing protocol

Common mixing protocol properties
• Volume
  \[ \text{player.volume} = 0.5 \]

Stereo mixing protocol properties
• Pan
  \[ \text{player.pan} = -1.0; \]

3D mixing protocol properties
• ?
Mixer Input Bus Settings

AVAudioEngine

Player 1

Player 2

Player 3

Mixer 1

Mixer 2

Main Mixer

Output
Mixer Input Bus Settings

AVAudioEngine

Set pan to -1, volume to 0.5

Player 1 → Mixing Settings → Mixer 1 → Main Mixer → Output
Player 2 → Mixer 1
Player 3 → Mixer 2
Mixer Input Bus Settings

AVAudioEngine

Player 2 → Mixer 1 → Main Mixer → Output
Player 3 → Mixer 1 → Main Mixer → Output
Player 1 → Mixing Settings → Mixer 2 → Main Mixer → Output

Pan = -1, volume = 0.5
AVAudioEngine
Gaming and 3D audio
Using Existing API

- **AudioServices**: Play short sounds
- **AVAudioPlayer**: Play music, sounds from files
- **OpenAL**: Play sounds that get spatialized (3D)
Trade-Offs

Every API has different nomenclature

AudioServices
  • No real-time guarantee
  • Sounds < 30 secs

AVAudioPlayer
  • Cannot play buffers

OpenAL
  • Cannot play from file
  • Cannot play compressed data
Using AVAudioEngine

- AVAudioPlayerNode
  - AVAudioBuffer
    - Play short sounds

- AVAudioPlayerNode
  - AVAudioFile
    - Play music, sounds from files

- ?
  - Play sounds that get spatialized (3D)
Environment Node

AVAudioEnvironmentNode class

Mixer node that simulates a 3D space
Implicit listener
Source nodes act as sources in this space
Source node properties are set via AVAudioMixing protocol
Only mono inputs are spatialized
Environment Node

AVAudioEngine

AVAudioPlayerNode (Source)

Mixing Settings

AVAudioEnvironmentNode (Listener)

AVAudioPlayerNode (Source)

Mixing Settings
What Makes It Sound 3D?

Source
- Position
- Rendering algorithm
- Obstruction, occlusion

Environment
- Listener position, orientation
- Distance attenuation
- Reverberation
Source Position

AVAudioMixing protocol

Right-handed cartesian coordinate system

```swift
player.position = AVAudioMake3DPoint(-2.0, 0.0, 5.0);
```
Source Rendering Algorithm

AVAudioMixing protocol

Listener’s directional cues
- Inter-aural time difference
- Inter-aural level difference
- Effect of head and ear filtering

Available rendering algorithms
- Equal Power Panning
- Spherical Head
- HRTF
- Sound Field

player.renderingAlgorithm = AVAudio3DMixingRenderingAlgorithmEqualPowerPanning;
Source Obstruction

AVAudioMixing protocol

player.obstruction = -10.0;

Direct path is muffled

Reflections are clear
Source Occlusion
AVAudioMixing protocol

player.occlusion = -15.0;

Direct path is muffled
Reflections are muffled
Listener Position and Orientation

AVAudioEnvironmentNode class

Implicit listener

Listener coordinates
- Right handed cartesian coordinate system

Listener orientation
- Vector or angular based orientation

```javascript
environment.listenerPosition = AVAudioMake3DPoint(-5.0, 0.0, 0.0);
```

```javascript
environment.listenerAngularOrientation = AVAudioMake3DAngularOrientation(90.0, 0.0, 0.0);
```
Distance Attenuation

AVAudioEnvironmentNode class

Attenuation of sound as source moves away from listener

Multiple distance attenuation models available

Reference Distance  Maximum Distance

Volume

Distance from Listener

Inverse Curve
Exponential Curve
Linear Curve
Distance Attenuation

AVAudioEnvironmentNode class

Attenuation of sound as source moves away from listener
Multiple distance attenuation models available

AVAudioEnvironmentDistanceAttenuationParameters *dap =
environment.distanceAttenuationParameters;

dap.distanceAttenuationModel =
AVAudioEnvironmentDistanceAttenuationModelInverse;

dap.referenceDistance = 5.0;
dap.maximumDistance = 100.0;
dap.rolloffFactor = 1.0;
Reverberation

AVAudioEnvironmentNode class

Simulates sound reflections within a space
- Room size
- Decay time
- High frequency absorption

Built in reverb with factory presets

Blends individual amount of reverb for each source

Single post-reverb filter
Reverberation

AVAudioEnvironmentNode class

AVAudioEnvironmentReverbParameters *reverbParameters = environment.reverbParameters;

reverbParameters.enable = YES;

[reverbParameters loadFactoryReverbPreset:AVAudioUnitReverbPresetLargeHall];

player.reverbBlend = 0.2;
Moving Between Mixer Nodes

AVAudioMixing protocol

Protocol defines common, stereo and 3D properties

- AVAudioMixerNode responds to stereo properties
- AVAudioEnvironmentNode responds to 3D properties

Property changes are cached when not applicable
Moving Between Mixer Nodes

Pan and volume properties take effect

Set pan to -1, volume to 0.5

AVAudioEngine

Output

Player 1 -> Mixing Settings -> Mixer Node

Player 2 -> Environment Node

Player 3 -> Environment Node

Main Mixer
Moving Between Mixer Nodes

Pan property is cached but has no effect on environment node
Volume property takes effect
Sample Gaming Setup

AVAudioEngine

Player
Input

Environment (3D)

Main Mixer (2D)

Output
Sample Gaming Setup

AVAudioEngine

Player ➔ EQ ➔ Main Mixer (2D) ➔ Output
Sample Gaming Setup

AVAudioEngine

Player

Main Mixer (2D)

Output
Sample Gaming Setup

AVAudioEngine

Player

Input

Player

Environment (3D)

Main Mixer (2D)

Output

EQ
Demo
Gaming
Kapil Krishnamurthy
3D Audio Ninja
Summary

Engine and connections
Output node
Source nodes—Player, input
Mixer nodes, AVAudioMixing protocol
Effect nodes
Node taps
AVAudioEngine v.1
More Information

Filip Iliescu
Graphics and Games Technologies Evangelist
filiescu@apple.com

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

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

<table>
<thead>
<tr>
<th>Audio Lab</th>
<th>Media Lab A</th>
<th>Tuesday 11:30AM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audio Lab</td>
<td>Media Lab B</td>
<td>Wednesday 12:45PM</td>
</tr>
</tbody>
</table>
