Audio Recipes for iOS

AVFoundation / MPMediaPlayer / CoreAudio / OpenAL

Javier Sánchez
@jsanchezsierra
http://ccrma.stanford.edu/~jsanchez

NSSpain, September 17-19, Logroño, Spain
Agenda

- SystemSound (Audio Toolbox)
- Media Player (MPMusicPlayerController)
- AVFoundation (AVAudioPlayer)
- Audio Sessions

- CoreAudio / Audio Units
- Open AL

@jsanchezsierra
http://ccrma.stanford.edu/~jsanchez
Introduction

- Media Player
- AV Foundation
- OpenAL
- Audio Toolbox
- Foundation
- Audio Unit
- Core Media
- Drivers and Hardware

@jsanchezsierra  
http://ccrma.stanford.edu/~jsanchez
Introduction

- Media Player
- AV Foundation
- OpenAL
- Audio Toolbox
- Foundation
- Audio Unit
- Core Media
- Drivers and Hardware

@jsanchezsierra
http://ccrma.stanford.edu/~jsanchez
SystemSound
Audio Toolbox
SystemSound
Audio Toolbox
- Add `AudioToolbox` framework

```c
#import <AudioToolBox/AudioToolBox.h>
```

- Create the URL to your sound file (ex: `sound.mp3`)

```c
NSString *soundPath = [[NSBundle mainBundle] pathForResource:@"sound" ofType:@"mp3" inDirectory:@"/"];
CFURLRef SoundPathPathURL = (CFURLRef) [[NSURL alloc] initFileURLWithPath: soundPath];
```

- Create the sound and play

```c
SystemSoundID systemSound;
AudioServicesCreateSystemSoundID ( SoundPathPathURL ,&systemSound);
AudioServicesPlaySystemSound(systemSound);
```

- No longer than 30 seconds in duration
- In linear PCM or IMA4 (IMA/ADPCM) format
- Packaged in a `.caf`, `.aif`, or `.wav` file
Media Player
MPMediaPickerController / MPMusicPlayerController
MPMediaPickerController           MPMusicPlayerController
• Add **MediaPlayer** framework

```c
#import <MediaPlayer/MediaPlayer.h>
```

• **MPMediaPickerController** allows you to access your music Library

```c
MPMediaPickerController *picker = [[MPMediaPickerController alloc] initWithMediaTypes: MPMediaTypeMusic];
picker.delegate = self; // MPMediaPickerControllerDelegate
picker.allowsPickingMultipleItems = YES;
picker.prompt = @"Añade las canciones a reproducir";
```

• Present the **MPMediaPickerController** in a modal ViewController

```c
UINavigationController *theNavController = [[[UINavigationController alloc] initWithRootViewController:picker];
[self presentModalViewController:theNavController animated: YES];
```

• Delegate Methods --> **MPMediaPickerControllerDelegate**.

```c
- (void) mediaPicker: (MPMediaPickerController *) mediaPicker
didPickMediaItems: (MPMediaItemCollection *) mediaItemCollection
{
...
[myPlayer setQueueWithItemCollection: mediaItemCollection];
...
}
```
MPMusicPlayerController
Media Player

• Add MediaPlayer framework

#import <MediaPlayer/MediaPlayer.h>

• Create an instance of MPMusicPlayerController

MPMusicPlayerController *myPlayer = [[MPMusicPlayerController applicationMusicPlayer] autorelease];
[myPlayer setQueueWithItemCollection: mediaItemCollection];
[myPlayer play];

• Get metadata (título, artista, album, imagen...)

MPMediaItem *myTrack = [myPlayer nowPlayingItem];
NSString *title = [myTrack valueForProperty: MPMediaItemPropertyTitle];
NSString *artist = [myTrack valueForProperty: MPMediaItemPropertyArtist];
UIImage *artworkImage = [[myTrack valueForProperty: MPMediaItemPropertyArtwork] imageWithSize: CGSizeMake (120, 120)];

• Methods and properties

[myPlayer play];
[myPlayer pause];
[myPlayer stop];
[myPlayer skipToNextItem];
[myPlayer skipToPreviousItem];
[myPlayer skipToBeginning];
[myPlayer nowPlayingItem];
[myPlayer currentPlaybackTime];
[myPlayer repeatMode];
[myPlayer shuffleMode];
[myPlayer playbackState];
[myPlayer volumen];
MPVolumeView
MPVolumeView
```objective-c
MPVolumeView *myVolumeView = [[MPVolumeView alloc] initWithFrame: CGRectMake(20, 450, 280, 20)];
[self.view addSubview: myVolumeView];

UIImage* knobImage = ...;
UIImage* volumeViewMinImage = ...;
UIImage* volumeViewMaxImage = ...

[myVolumeView setVolumeThumbImage: KnobImage forState: UIControlStateNormal];
[myVolumeView setMaximumVolumeSliderImage: volumeViewMaxImage forState: UIControlStateNormal];
```

@jsanchezierra
http://ccrma.stanford.edu/~jsanchez
VolumeView
My Custom Volume View Class (colors, number of lines)
https://github.com/jsanchezsierra/VolumeView

```
VolumeView *volumeView = [[VolumeView alloc] initWithFrame:CGRectMake(175, 425, 110, 55)];
[volumeView setNumberOfBars:7];
[volumeView setBarsColorMin:[UIColor greenColor]];
[volumeView setBarsColorMax:[UIColor redColor]];
[self.view addSubview: volumeView];
```
AVFoundation
AVAudioPlayer

@jsanchezsierra
http://ccrma.stanford.edu/~jsanchez
AVAudioPlayer
AVFoundation
AVAudioPlayer

• Add AVFoundation framework

#include <AVFoundation/AVFoundation.h>

• Create a AVAudioPlayer from a local file

NSURL * urlTrack = [[NSURL alloc] initWithPath:
                    [[NSBundle mainBundle] pathForResource:@"track" ofType:@"mp3"]];  
AVAudioPlayer *track = [[AVAudioPlayer alloc] initWithContentsOfURL: urlTrack error: nil];

• Create a AVAudioPlayer from NSData

AVAudioPlayer *track = [[AVAudioPlayer alloc] initWithData: [NSData ...] error: nil];

• Methods and properties

track duration; (read only) track numberOfLoops;
track numberOfChannels; (read only) track rate;
track isPlaying; (read only) track enableRate;
track prepareToPlay; track currentTime;
track play; track averagePowerForChannel;
track pause; track peakPowerForChannel;
track stop; track url;
track playAtTime; track data;
track volumen; track settings;
track pan;
AVFoundation
AVAudioPlayer Delegates

-(void) audioPlayerDidFinishPlaying:(AVAudioPlayer *)player successfully:(BOOL)flag {
    //Update UI
    /...
}

-(void) audioPlayerBeginInterruption:(AVAudioPlayer *)player {
    [player pause];
    //Update UI
    /...
}

-(void) audioPlayerEndInterruption:(AVAudioPlayer *)player withOptions:(NSUInteger)flags {
    [player play];
    //Update UI
    /...
}
Audio Session
AVAudioSession
Audio Session
AVAudioSession

- Categorize the App audio type
- Mix with background audio
- Handle interruptions and routing changes

- Objective-C alternative to many features from the C-based Audio Session services

AVAudioSession

<AVFoundation/AVAudioSession.h>
High-level wrapper with most common functionalities

Audio Session Services

<AudioToolbox/AudioSession.h>
C-Based, lower-level, all the implementation
#import <AVFoundation/AVFoundation.h>

//Create an instance of AVAudioSession
AVAudioSession *session = [AVAudioSession sharedInstance];

//Registers the delegate of the audio session.
[session setDelegate: self];

//set the playback category for the session
[session setCategory: AVAudioSessionCategoryPlayback error: nil];

// Activates the audio session.
[session setActive: YES error: nil];
Audio Session

2.- Choose a category

Based on the role of audio in your app

<table>
<thead>
<tr>
<th>Category Name</th>
<th>Intended Usage</th>
<th>Obey Ringer Switch</th>
<th>Obey Screen Lock</th>
<th>Mix with Others</th>
<th>Audio Input</th>
<th>Audio Output</th>
<th>Allowed In Background</th>
</tr>
</thead>
<tbody>
<tr>
<td>Playback</td>
<td>AudioPlayers, Video Players</td>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Record</td>
<td>AudioRecorders, Voice Capture</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Play and Record</td>
<td>VOIP, Voice Chat</td>
<td>Optional</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Audio Processing</td>
<td>Offline Conversion, Offline Processing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ambient</td>
<td>Games, Productivity Apps</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Solo Ambient</td>
<td>Games, Productivity Apps</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

WWDC 2010 - Session 412 - Audio Development for iOS, Part I

@jsanchezsierra
http://ccrma.stanford.edu/~jsanchez
Audio Session
Handle interruptions

AVAudioSessionDelegate

// Called after your audio session is interrupted
-(void) beginInterruption
{
    // Update UI ...
    // Playback stopped
}

// Called after your audio session interruption ends
// flags indicate the state of the audio session
-(void) endInterruptionWithFlags:(NSUInteger)flags
{
    // Update UI ...
    // Make session active
    // Resume playback
}
Audio Session
Handle route changes

What is the user experience?

Pluggin in the headphone
- routed to headphone
- audio continues playing, no pause

Unpluggin the headphone
- routed to output
- audio pause
Querying the route --> kAudioSessionProperty_AudioRoute

```swift
CFStringRef currentRoute;
UInt32 size = sizeof(currentRoute);
AudioSessionGetProperty( kAudioSessionProperty_AudioRoute, &size, &currentRoute);
NSLog (@"Current route is %@", currentRoute);
```

route values --> @“speaker”, @“Headphone”, @“receiver”, @“”...

Overrinding the output audio route --> kAudioSessionProperty_OverrideAudioRoute

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

routes output to speaker
Audio Session
Handle route changes

listening to route changes --> kAudioSessionAddPropertyListener

AudioSessionAddPropertyListener(kAudioSessionProperty_AudioRouteChange, MyPropListenerCallback, &clientData);

Register for notifications when route changes (reason/old route).

AudioRoute change CallBack --> kAudioSessionProperty_OVERRIDE_AUDIOROUTE

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));
}

@jsanchezsierra
http://ccrma.stanford.edu/~jsanchez
Playing in Background
info.plist file --> set Required background modes

@jsanchezsierra
http://ccrma.stanford.edu/~jsanchez
Core Audio
Core Audio
Audio Units

- Media Player
- AV Foundation
- OpenAL
- Audio Toolbox
- Foundation
- Audio Unit
- Core Media
- Drivers and Hardware

@jsanchezsierra
http://ccrma.stanford.edu/~jsanchez
Audio Units
Core Audio

When to use the Audio Units

• Very specific needs
• Low Latency I/O
• Responsive playback of synthesized sounds
• Use of built-in features (echo cancelation, mixing, paning...)

Where to use Audio Units

• VoIP Apps (using Voice Processing I/O unit)
• Interactive music apps (mixer unit)
• For real time I/O processing
Digital audio basic concepts
Digital Audio Basics

Digital audio representation of a sound

An analog waveform is represented digitally using sines and cosines expressions. Discrete representation -> sampling rate, bit sample.
Digital Audio

Sample: smaller value of digitized audio data
Sampling rate: number of samples per second

44.100 samples/second

44.1 KHz

t=0

t=1

time

@jsanchezsierra
http://ccrma.stanford.edu/~jsanchez
### Digital Audio

**Sampling rate**

<table>
<thead>
<tr>
<th>Sampling rate</th>
<th>Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 KHz</td>
<td>Narrow-band speech</td>
</tr>
<tr>
<td>16 KHz</td>
<td>Narrow-band speech</td>
</tr>
<tr>
<td>44.1 KHz</td>
<td>CD quality</td>
</tr>
<tr>
<td>48 KHz</td>
<td>Digital Audio Tape</td>
</tr>
<tr>
<td>96 KHz</td>
<td>Pro Quality</td>
</tr>
<tr>
<td>192 KHz</td>
<td>Ultimate marketing quality</td>
</tr>
</tbody>
</table>
Digital Audio
Channels, frames

Stereo sound: 2 channels: R & L
Frames: A collection of samples for each channel. Stereo has 2 frames

One frame of interleaved stereo LPCM

same buffer for both channel

One frame of non-interleaved stereo LPCM

two buffers, one per channel

http://ccrma.stanford.edu/~jsanchez

@jsanchezsierra
Digital Audio

bit depth (bits/sample)

- Each sample represents the amplitude of the wave
- Bit depth: sample resolution (8 bits, 16 bits...)

If the difference between two consecutive sounds is smaller than the sample resolution, the difference is lost.

bit rate (bits/second) = number of channels × bit depth (bits/sample) × sampling rate (samples/second)

bit rate = 2 × 16 bits/sample × 44100 samples/second = 1,411,200 bits/second = 1,411 Kbits/second

80 minutes of audio, stereo 16 bits, 44.1 KHz, CD Quality

bit rate = 80 minutes × 60 seconds/minute × 2 channels × 16 bits/sample × 44100 samples/second / 8 bits/byte = 846,720,000 bytes = 847 Mb

80 minutes = 847 Mb  Hardware limitations!!
# Audio Formats

## File Format / Data Formats

<table>
<thead>
<tr>
<th>File Format</th>
<th>Data Formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAC (.aac, .adts)</td>
<td>'aac '</td>
</tr>
<tr>
<td>AC3 (.ac3)</td>
<td>'ac-3'</td>
</tr>
<tr>
<td>AIFF (.aiff, .aiff, .aifc)</td>
<td>BE18, BE116, BE124, BE132, BEF32, BEF64, 'ulaw', 'alaw', 'MAC3', 'MAC6', 'ima4', 'QDMC', 'QDM2', 'Qclp', 'agsm'</td>
</tr>
<tr>
<td>AIFF (.aiff)</td>
<td>BE18, BE116, BE124, BE132</td>
</tr>
<tr>
<td>Apple Core Audio Format</td>
<td>(.caf) 'mp3', 'MAC3', 'MAC6', 'QDM2', 'QDMC', 'Qclp', 'Qclq', 'aac ', 'agsm', 'alac', 'alaw', 'drms', 'dvi ', 'ima4', 'lpc ', 'BE18, BE116, BE124, BE132, BEF32, BEF64, LEI16, LEI24, LEI32, LEF32, LEF64, 'ms\x00\x02', 'ms\x00\x11', 'ms\x00\x11', 'ms\x00\x11', 'ms\x00\x00', 'ms\x00', 'samr', 'ulaw'</td>
</tr>
<tr>
<td>MPEG Layer 3 (.mp3)</td>
<td>'.mp3'</td>
</tr>
<tr>
<td>MPEG 4 Audio (.mp4)</td>
<td>'aac '</td>
</tr>
<tr>
<td>MPEG 4 Audio (.m4a)</td>
<td>'aac ', 'alac'</td>
</tr>
<tr>
<td>NeXT/Sun Audio (.snd, .au)</td>
<td>BE18, BE116, BE124, BE132, BEF32, BEF64, 'ulaw'</td>
</tr>
<tr>
<td>Sound Designer II (.sd2)</td>
<td>BE18, BE116, BE124, BE132</td>
</tr>
<tr>
<td>WAVE (.wav)</td>
<td>LEUI8, LEI16, LEI24, LEI32, LEF32, LEF64, 'ulaw', 'alaw'</td>
</tr>
</tbody>
</table>
Audio Formats
LPCM / compressed formats

-LPCM (Linear Pulse Code Modulation)
  Uncompressed format
  One packet = one frame
  Constant bit rate (CBR)
  number of frames per packet = 1

-Packetize compressed formats
  A group of frames of LPCM is compressed into a packet. Packets have dependencies on preceding packets.
  Variable bit rate (VBR)
  number of frames per packet (AAC has 1024 frames/packet)
Audio Formats
Get audio file information

> afinfo mySong.mp3

File: mySong.mp3
File type ID: MPG3
Num Tracks: 1
----
Data format: 2 ch, 44100 Hz, '.mp3' (0x00000000) 0 bits/channel, 0 bytes/packet, 1152 frames/packet, 0 bytes/frame no channel layout.
estimated duration: 274.745600 sec
audio bytes: 5494912
audio packets: 10517
bit rate: 160000 bits per second
packet size upper bound: 1052
maximum packet size: 523
audio data file offset: 0
optimized

> afinfo Sound2.caf

File: Sound2.caf
File type ID: caff
Num Tracks: 1
----
Data format: 1 ch, 44100 Hz, 'lpcm' (0x0000000C) 16-bit little-endian signed integer no channel layout.
estimated duration: 0.328345 sec
audio bytes: 28960
audio packets: 14480
bit rate: 705600 bits per second
packet size upper bound: 2
maximum packet size: 2
audio data file offset: 4096
optimized
audio 14480 valid frames + 0 priming + 0 remainder = 14480
source bit depth: I16
sound check:
  approximate duration in seconds 0.328
sound check volume normalization gain: 0.00 dB
----

@jsanchezsierra
http://ccrma.stanford.edu/~jsanchez
Audio Stream Basic Description

format properties of a stream of audio data

```c
struct AudioStreamBasicDescription {
    Float64 mSampleRate;
    UInt32 mFormatID;
    UInt32 mFormatFlags;
    UInt32 mBytesPerPacket;
    UInt32 mFramesPerPacket;
    UInt32 mBytesPerFrame;
    UInt32 mChannelsPerFrame;
    UInt32 mBitsPerChannel;
    UInt32 mReserved;
};
```

Compressed audio (VBR)
- `mBytesPerPacket = 0;`
- `mBytesPerFrame = 0;`
- `mBitsPerChannel = 0;`
Audio Units
Core Audio
Audio Units

Some definitions

• Audio Processing Graph: An object that manages a network of audio unit nodes
• Audio Unit: audio processing plug-in component
• Audio Node: representation of Audio Unit in the context of an Audio Processing Graph
Audio Units
Some definitions

• **Audio Processing Graph**: An object that manages a network of audio unit nodes
• **Audio Unit**: audio processing plug-in component
• **Audio Node**: representation of Audio Unit in the context of an Audio Processing Graph
Audio Units

Available Units for OSX

• **Generator units:** stream of audio from files, network, memory
• **Instrument units:** stream of synthesized audio from MIDI data.
• **Mixer units:** Combine multiple streams into one or more streams.
• **Effect units:** digital signal processing, reverb, pitch change, noise filtering...
• **Converter units:** Perform transformations (change sample rate, bit depth, adjust playback speed...)
• **Output units:** Interface with audio input and/or output hardware. Input/output

Available Units for iOS

• **Effect units:** iPod Equalizer
• **Mixing units:** 3D Mixer / Multichannel Mixer
• **I/O Unit:** Remote I/O, Voice-Processing I/O, Generic Output
• **Format conversion:** Format converter

@jsanchezsierra
http://ccrma.stanford.edu/~jsanchez
Filter Units

OSX

@jsanchezsierra
http://ccrma.stanford.edu/~jsanchez
Audio Units

other examples
Demo 2

iPhoneMixerEQGraphTest (Apple code sample)
Audio Unit example

iPhoneMixerEQGraphTest (apple code sample)
Audio Units
Creating an Audio Unit App

1. Create audio session
2. Specify audio units
3. Create a graph, then obtain the audio units
4. Configure the audio units
5. Connect the nodes
6. Provide a user interface
7. Initialize and then start the graph
Audio Units

I.- Configure Audio Session

```swift
self.graphSampleRate = 44100.0;

AVAudioSession *mySession = [AVAudioSession sharedInstance];
[mySession setPrederredHardwareSampleRate: graphSampleRate error: nil];
[mySession setCategory: AVAudioSessionCategoryPlayAnddRecord error: nil];
[mySession setActive: YES error: nil];
self.graphSampleRate = [mySession currentHardwareSampleRate];
```

Sets the preferred hardware sample rate for input and output.
Audio Units
2.- Specify Audio Units

// multichannel mixer unit
AudioComponentDescription mixer_desc;
mixer_desc.componentType = kAudioUnitType_Mixer;
mixer_desc.componentSubyype = kAudioUnitSubType_MultiChannelMixer;
mixer_desc.componentManufacturer = kAudioUnitManufacturer_Apple;
mixer_desc.componentFlags = 0;
mixer_desc.componentFlashsMask = 0;

// iPodEQ unit
AudioComponentDescription eq_desc;
eq_desc.componentType = kAudioUnitType_Effect;
eq_desc.componentSubyype = kAudioUnitSubType_AUiPodEQ;
eq_desc.componentManufacturer = kAudioUnitManufacturer_Apple;
eq_desc.componentFlags = 0;
eq_desc.componentFlashsMask = 0;

// output unit
AudioComponentDescription output_desc;
output_desc.componentType = kAudioUnitType_Output;
output_desc.componentSubyype = kAudioUnitSubtype_RemoteIO;
output_desc.componentManufacturer = kAudioUnitManufacturer_Apple;
output_desc.componentFlags = 0;
output_desc.componentFlashsMask = 0;

@jsanchezsierra
http://ccrma.stanford.edu/~jsanchez
Audio Units

3.- Create a graph, then obtain the audio units

// create a new AUGraph
AUGraph  mGraph;
result = NewAUGraph(&mGraph);

// Add Audio Nodes to graph
AUNode  outputNode;
AUNode  eqNode;
AUNode  mixerNode;
AUGraphAddNode(mGraph, & mixer_desc, &mixerNode);
AUGraphAddNode(mGraph, & eq_desc, &eqNode);
AUGraphAddNode(mGraph, & output_desc, &outputNode);

// open the graph AudioUnits (but not initialized)
result = AUGraphOpen(mGraph);

// grab the audio unit instances from the nodes
AudioUnit  mEQ;
AudioUnit  mMixer;
result = AUGraphNodeInfo(mGraph, mixerNode, NULL, &mMixer);
result = AUGraphNodeInfo(mGraph, eqNode, NULL, &mEQ);
Audio Units

4.- Configure the audio units (AudioUnitSetProperty / AudioUnitGetProperty)

// set number of input buses for the mixer Audio Unit
UInt32 numbuses = 2;
AudioUnitSetProperty ( mMixer, kAudioUnitProperty_ElementCount,
    kAudioUnitScope_Input, 0, &numbuses, sizeof(numbuses));

// set a callback for the specified node's specified input
AURenderCallbackStruct rcbs=...;
AUGraphSetNodeInputCallback(mGraph, mixerNode, busNumber, &rcbs);

//Set mixer input & output format
CAStreamBasicDescription mClientFormat= ...;
CAStreamBasicDescription mOutputFormat= ...;
AudioUnitSetProperty(mMixer, kAudioUnitProperty_StreamFormat,
    kAudioUnitScope_Input, i, &mClientFormat, sizeof(mClientFormat));
AudioUnitSetProperty(mMixer, kAudioUnitProperty_StreamFormat,
    kAudioUnitScope_Output, 0, &mOutputFormat, sizeof(mOutputFormat));

// get the equalizer factory presets list
CFArrayRef mEQPresetsArray;
UInt32 sizeof = sizeof(mEQPresetsArray);
AudioUnitGetProperty(mEQ, kAudioUnitProperty_FactoryPresets,
    kAudioUnitScope_Global, 0, &mEQPresetsArray, &size);
Audio Units

5.- Connect the nodes

// connect a node's output to a node's input
// mixer -> eq -> output
result = AUGraphConnectNodeInput(mGraph, mixerNode, 0, eqNode, 0);
result = AUGraphConnectNodeInput(mGraph, eqNode, 0, outputNode, 0);
Audio Units
6.- Provide a user interface

```c
// Mixer Unit - Change input volumen for inputNum bus
AudioUnitSetParameter(mMixer, kMultiChannelMixerParam_Volume,
                        kAudioUnitScope_Input, busNumber, value, 0);

// Mixer Unit - Change output volumen for bus 0
AudioUnitSetParameter(mMixer, kMultiChannelMixerParam_Volume,
                        kAudioUnitScope_Output, 0, value, 0);

// Mixer Unit - Enable bus
AudioUnitSetParameter(mMixer, kMultiChannelMixerParam_Enable,
                       kAudioUnitScope_Input, busNumber, isONValue, 0);

// Equalizer Unit - Change equalizer preset from mEQPresetArray
AUPreset *aPreset = (AUPreset*)CFArrayGetValueAtIndex(mEQPresetsArray, presetIndex);
AudioUnitSetProperty(mEQ, kAudioUnitProperty_PresentPreset,
                      kAudioUnitScope_Global, 0, aPreset, sizeof(AUPreset));
```

@jsanchezsierra
http://ccrma.stanford.edu/~jsanchez
Audio Units

7.- Initialize and start the graph

```c
AUGraphInitialize(mGraph);

AUGraphStart(mGraph);

...

AUGraphStop(mGraph);

...

Boolean isRunning;
AUGraphIsRunning(mGraph, &isRunning);
```
Audio Unit example

iPhoneMixerEQGraphTest (apple code sample)
Spatial Audio
3DMixer Unit / OpenAL
Demo 3

oalTouch (Apple sample code)

3DMixer Audio Unit

Fundamentals

• One listener and multiple sound sources
• 3D Audio (panning, audio cues, reverb, obstruction, occlusion)
enum {
    k3DMixerParam_Azimuth                 = 0,
    k3DMixerParam_Elevation               = 1,
    k3DMixerParam_Distance                = 2,
    k3DMixerParam_Gain                    = 3,
    k3DMixerParam_PlaybackRate            = 4,
    k3DMixerParam_Enable                  = 5,
    k3DMixerParam_MinGain                 = 6,
    k3DMixerParam_MaxGain                 = 7,
    k3DMixerParam_ReverbBlend             = 8,
    k3DMixerParam_GlobalReverbGain        = 9,
    k3DMixerParam_OcclusionAttenuation    = 10,
    k3DMixerParam_ObstructionAttenuation  = 11
};

3DMixer Audio Unit
Parameters
• Open standard audio API for 3D spatial audio (complement to Open GL)
• Available on OSX and iOS.

// Device Management
alcOpenDevice(const ALCchar *devicename);
alcCloseDevice(ALCdevice *device);

// Create Open AL Context
alcCreateContext(ALCdevice *device, const ALCint *attrlist);
alcMakeContextCurrent(ALCcontext *context);

// Creating a source
alGenSources(ALsizei n, ALuint *sources);

// Creating a buffer and fill it
alGenBuffers(ALsizei n, ALuint *buffers);

// Attach OpenAL buffer to OpenAL source
alSourcei(source, AL__BUFFER, buffer);
// Set source attributes
alSourcefv(source, AL_POSITION, 0.8);
alSourcef(source, AL_REFERENCE_DISTANCE, 4);
alSourcei(source, AL__BUFFER, 3);

// Set listener attributes
alSourcefv(AL_POSITION ,listener_position);
alSourcefv(AL_POSITION ,listener_orientation);

// Play a sound
alSourcefv(AL_POSITION ,listener_position);

// Move source & listener position
alSourcefv(source, AL_POSITION ,2.0);
alSourcefv(AL_POSITION ,listener_position);
OpenAL Extensions
Reverb, Occlusion, Obstruction

ASA Extension (Apple Spatial Audio)

ALCASA_REVERB_ON
ALCASA_REVERB_GLOBAL_LEVEL
ALCASA_REVERB_ROOM_TYPE
  ALC_REVERB_ROOM_TYPE_SmallRoom
  ALC_REVERB_ROOM_TYPE_MediumRoom
  ALC_REVERB_ROOM_TYPE_LargeRoom
  ALC_REVERB_ROOM_TYPE_MediumHall
  ALC_REVERB_ROOM_TYPE_LargeHall
  ALC_REVERB_ROOM_TYPE_Cathedral
  ALC_REVERB_ROOM_TYPE_Plate
  ALC_REVERB_ROOM_TYPE_MediumChamber
  ALC_REVERB_ROOM_TYPE_LargeChamber
  ALC_REVERB_ROOM_TYPE_LargeRoom2
  ALC_REVERB_ROOM_TYPE_MediumHall2
  ALC_REVERB_ROOM_TYPE_MediumHall3
  ALC_REVERB_ROOM_TYPE_LargeHall2
ALCASA_EQ_GAIN
ALCASA_EQ_BANDWIDTH
ALCASA_EQ_FREQ
ALCASA_REVERB_SEND_LEVEL
ALCASA_OCCLUSION
ALCASA_OBSTRUCTION

Source Notifications Extension

AL_SOURCE_STATE
  AL_INITIAL
  AL_PLAYING
  AL_PAUSED
  AL_STOPPED
AL_BUFFERS_PROCESSED
AL_QUEUE_HAS_LOOPED

// Set a listener property
ALuint setting = 1;
alcASASetListenerProc(alcGetEnumValue(NULL, "ALCASA_REVERB_ON"), &setting, sizeof(setting));

// Set a source property
ALfloat level = 0.4;
alcASASetSourceProc(alcGetEnumValue(NULL, "ALCASA_REVERB_SEND_LEVEL"), source, &level, sizeof(level));
Core Audio
other interesting projects

AudioGraph
https://github.com/tkzic/audiograph

Novocaine
https://github.com/alexbw/novocaine

NVDSP (with novocaine)
https://github.com/bartolstoorn/NVDSP

AurioTouch2

DSP (Digital signal processing), FFT (Fast Fourier Transform), DFT (Discrete Fourier Transform)
Core Audio

Now you have the basics to start discovering the power of this framework...
Apple Documentation

AudioSession Programming Guide
AudioSessionProgrammingGuide.pdf

iPod Library Access Programming Guide

AudioQueue Programming Guide
AudioQueueProgrammingGuide.pdf

CoreAudio Overview
CoreAudioOverview.pdf

CoreAudio Framework Reference
CACoreAudioReference.pdf

Core Audio Format Specification

AudioUnitHostingGuideForiOS
AudioUnitHostingGuideForiOS.pdf

@jsanchezsierra
http://ccrma.stanford.edu/~jsanchez
Book

Learning Core Audio: A hands-on Guide to Audio Programming for Mac and iOS

Book Sample Code

WWDC 2010
Session 411 - Fundamentals of Digital Audio for Mac OS X and iPhone OS
Session 412 - Audio Development for iPhone OS, Part 1
Session 413 - Audio Development for iPhone OS, Part 2

WWDC 2011
Session 404 - Audio Development for Games
Session 411 - Music in iOS and Lion
Session 413 - Audio Session Management for iOS

WWDC 2012
Session 505 - Audio Session and Multiroute in iOS

WWDC 2013
Session 602-What’s New in Core Audio for iOS
MoMu Toolkit (Mobile Music at Stanford)
http://momu.stanford.edu/toolkit/

MoMu is a light-weight software toolkit for creating musical instruments and experiences on mobile device, and currently supports the iPhone platform (iPhone, iPad, iPod Touches).

MoMu provides API's for real-time full-duplex audio, accelerometer, location, multi-touch, networking (via OpenSoundControl), graphics, and utilities.

The Synthesis Toolkit (STK)
http://momu.stanford.edu stk/
Stanford Mobile Phone Orchestra (MoPhO) at CCRMA, Ge Wang

Core Audio, low-level, DSP

Funding: $25.5M since 2008
Ocarina for iPhone - Nov 2008

http://www.youtube.com/watch?v=kfrONZjakRY
Magic Piano

Lang Lang, San Francisco Symphony Hall, April 19th, 2010

http://www.youtube.com/watch?v=HvpIGbCBAiA
Double Company - smule / Kush

Hi. We're Smule

http://www.youtube.com/watch?v=B9AUad-HEIE
Thanks!
Audio Recipes for iOS
AVFoundation / MPMediaPlayer / CoreAudio / OpenAL

Javier Sánchez
@jsanchezsierra
http://ccrma.stanford.edu/~jsanchez


NSSpain, September 17-19, Logroño, Spain