Audio Development for Games

Session 404

Kapil Krishnamurthy
James McCartney
Core Audio Engineering

Agenda

- A “simple” game
  - AVAudioPlayer
- Understanding your audio assets
- A “complex” game
  - Spatial audio and OpenAL
- AudioSession

A “simple” game

- Background score
- Sound effects
- Basic control: volume, pan, looping
- Recommended API: AVAudioPlayer

AVAudioPlayer

- caf, m4a, mp3, aif, wav, au, snd, aac
- Play, pause, seek, stop
- Multiple sounds?
  - Use multiple AVAudioPlayer objects
- Volume, panning, looping

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

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 = 3;      // play once, repeat 3 times
player.currentTime = 5.0;      // playback position starts 5 seconds from start of file
player.delegate = myDelegate;  // delegate
```

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

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 0 to reset playback position

AVAudioPlayer

Playback controls

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


- Pauses playback
- Player remains prepared to play
  - Resources 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”
  - Resources are disposed
- To resume:
  - Need to “prepare” again

AVAudioPlayer

Current Time

- Current Time sets the time in seconds
- Pause and Stop leave play head at current position
- Set Current Time to 0 to reset play head

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

AVAudioPlayer

Prepare sound for playback

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

  // Set properties
  player.delegate = myDelegate;

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

AVAudioPlayer

Prepare sound for playback

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

  // Set properties
  player.delegate = myDelegate;

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

AVAudioPlayer

Prepare sound for playback

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

  // Set properties
  player.delegate = myDelegate;

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

Tuning Your Assets

Tuning Your Assets

Overview

- Create assets at the same sample rate
  - Multiple sample rate conversions are expensive

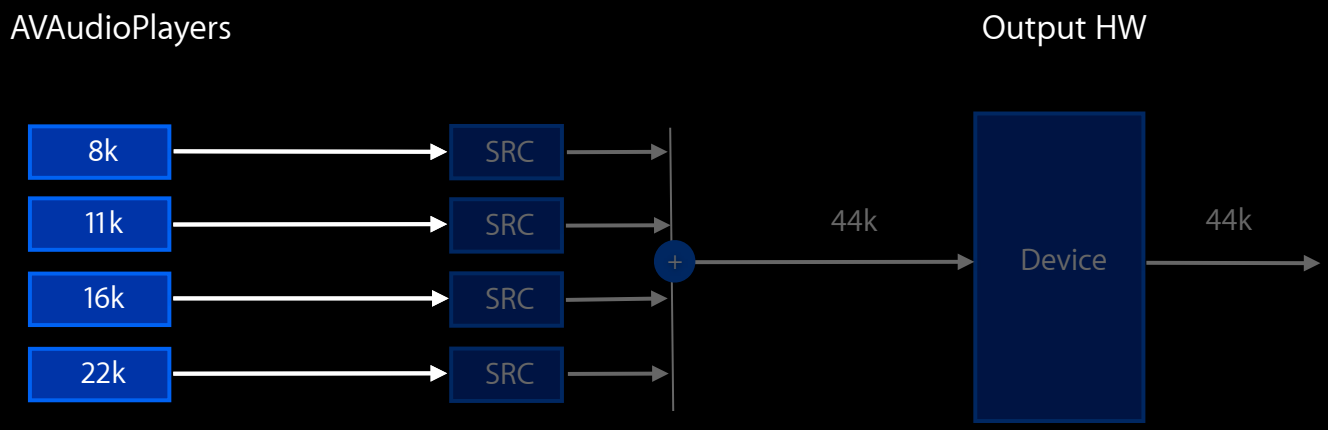
Tuning Your Assets

Overview

- Create assets at the same sample rate
  - Multiple sample rate conversions are expensive
- AAC vs. MP3
  - Better quality at same asset size
  - Similar quality with smaller asset size
  - Cheaper to decode

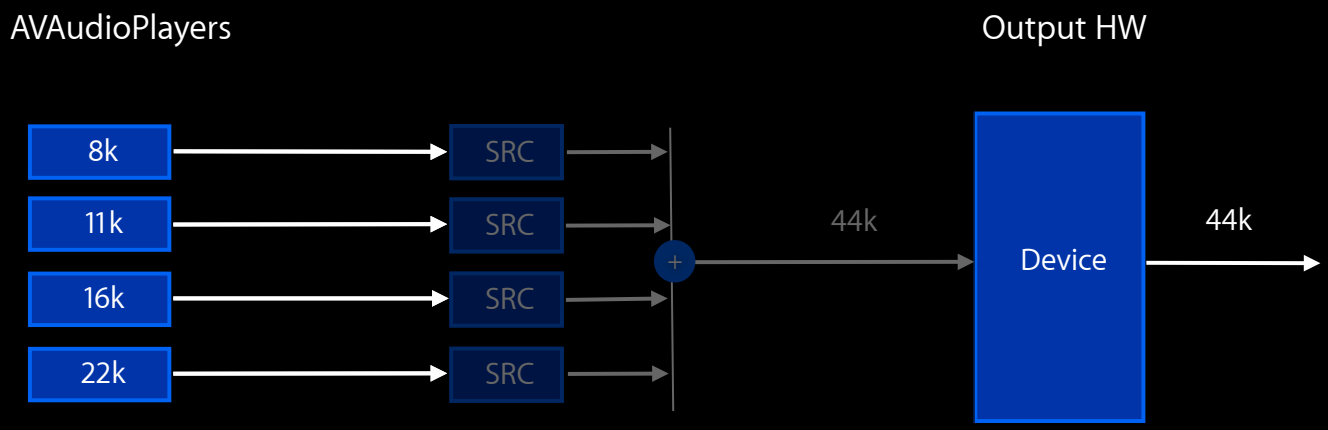
Tuning Your Assets

Sample rate conversions



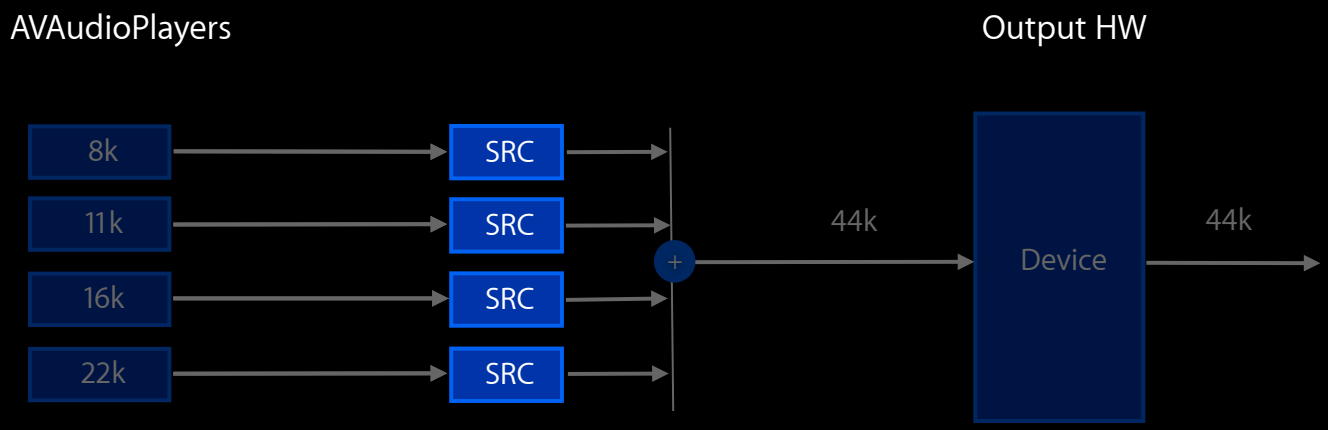
Tuning Your Assets

Sample rate conversions



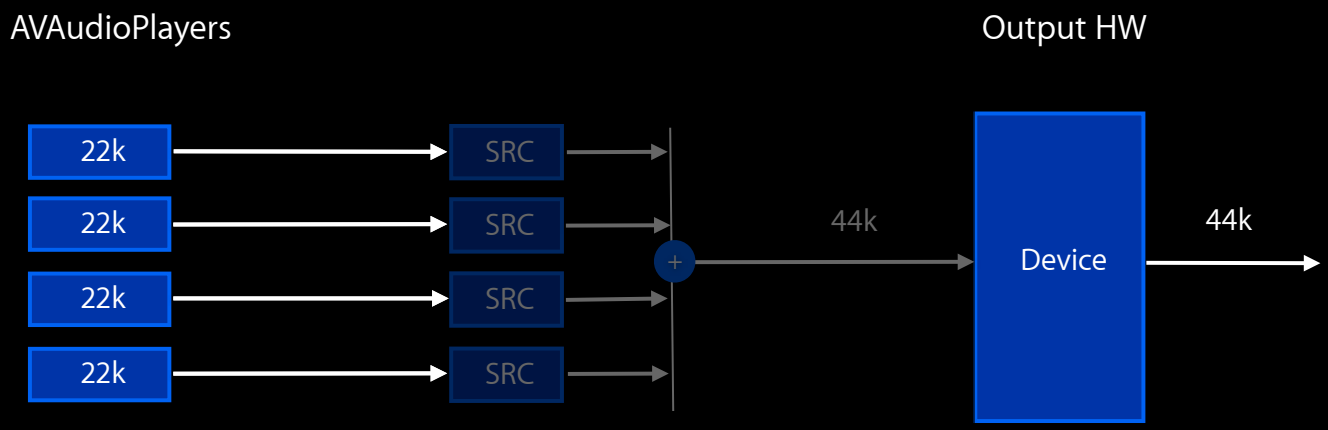
Tuning Your Assets

Sample rate conversions



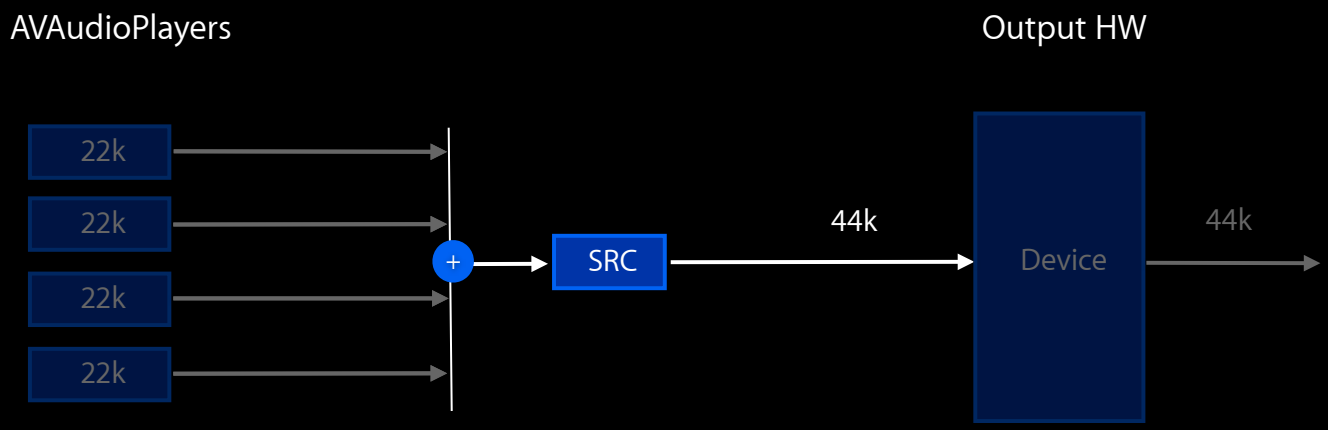
Tuning Your Assets

Sample rate conversions



Tuning Your Assets

Sample rate conversions



Tuning Your Assets

Picking a sample rate

- Pick the highest sample rate needed to capture fidelity of the sounds in your game

Tuning Your Assets

Picking a sample rate

- Pick the highest sample rate needed to capture fidelity of the sounds in your game
- AAC decoding is inexpensive

Audio Format Tools

afconvert

- Desktop command-line tool
- Converts between data formats, file formats, and sample rates

```
> afconvert sourcePCM.aif destAAC.m4a –f 'm4af' –d ‘aac ‘
```

Audio Format Tools

afinfo

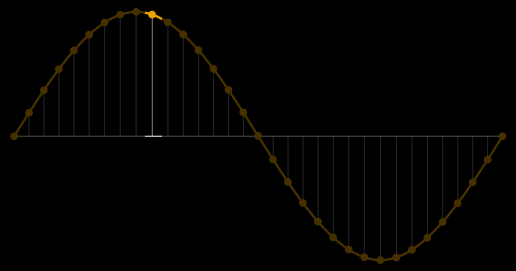
- Tool that displays file metadata

```
> afinfo destAAC.m4a
File:           destAAC.m4a
File type ID:   m4af
Data format:     1 ch,  44100 Hz, 'aac ' (0x00000000) 0 bits/channel, 0
bytes/packet, 1024 frames/packet, 0 bytes/frame
Channel layout: Mono
estimated duration: 0.915692 sec
audio bytes: 9749
audio packets: 42
audio 40382 valid frames + 2112 priming + 514 remainder = 43008
bit rate: 79972 bits per second
packet size upper bound: 313
audio data file offset: 4096
optimized
```

Working with Compressed Audio

James McCartney
Core Audio Engineering

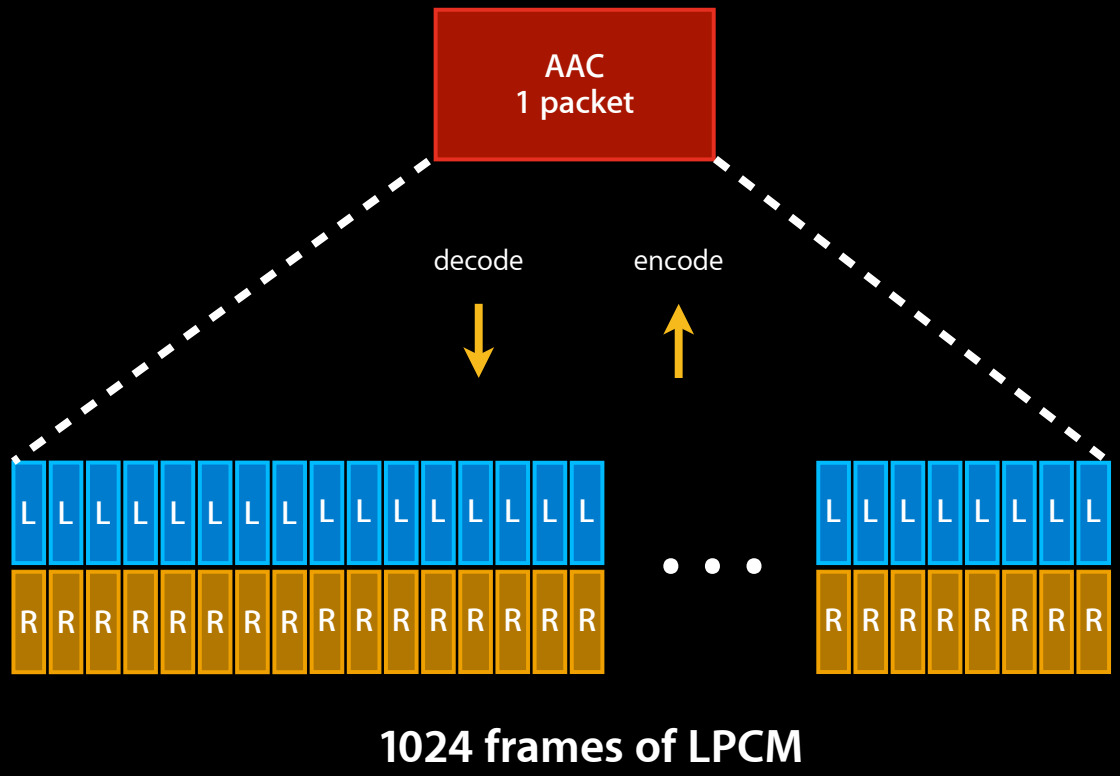
Samples, Frames, and Packets



- Sample
  - One sample of a waveform
- Frame
  - A collection of samples for each channel
- Packet
  - The smallest cohesive unit of data for a format
    - For LPCM, one packet equals one frame
    - For compressed formats, one packet is a group of bytes that decompress to some number of frames of LPCM

AAC
1 packet

Packets and Frames



1024 frames of LPCM

Working with Compressed Audio

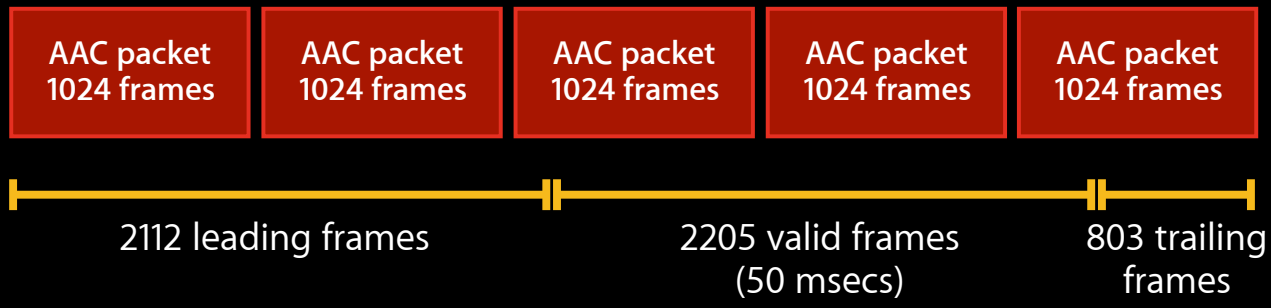
- Compressed audio has "leading" and "trailing" frames
  a.k.a. "priming" and "remainder"

Working with Compressed Audio

- Compressed audio has “leading” and “trailing” frames
- Leading frames express the processing latency of the codec
- Trailing frames are excess frames within the last packet that are not part of the program material

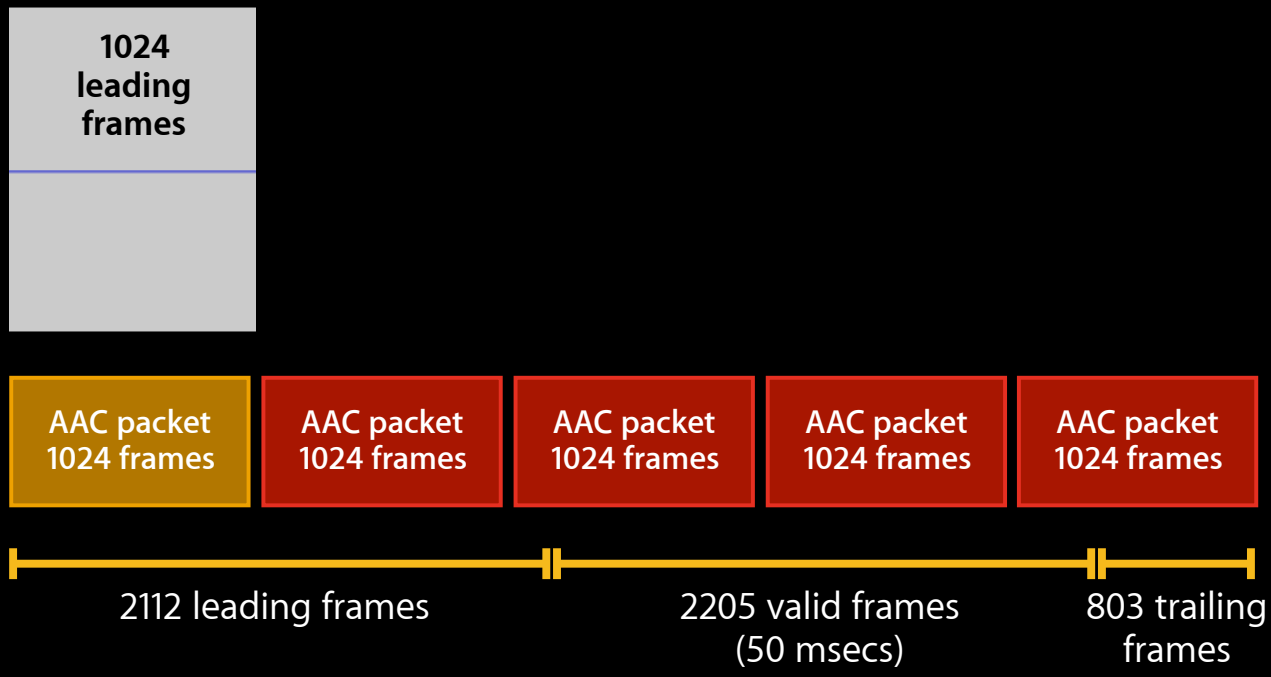
Working with Compressed Audio

- Compressed audio has "leading" and "trailing" frames
- Leading frames express the processing latency of the codec
- Trailing frames are excess frames within the last packet that are not part of the program material



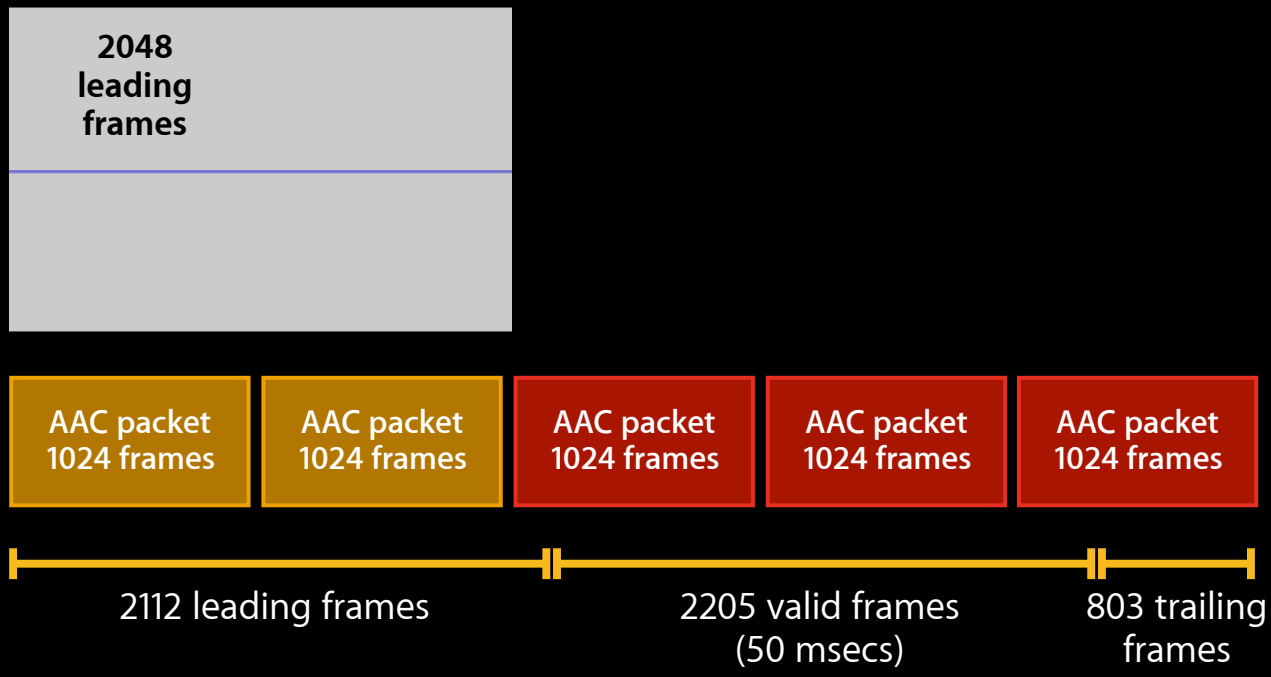
5 packets * 1024 frames per packet - 2112 leading - 803 trailing = 2205 valid frames

Decoding Compressed Audio



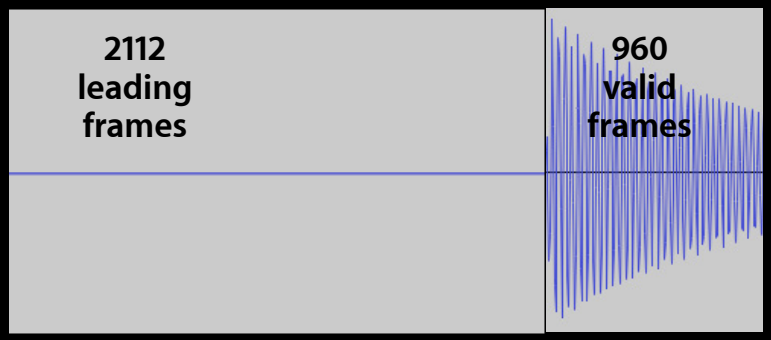
5 packets * 1024 frames per packet - 2112 leading - 803 trailing = 2205 valid frames

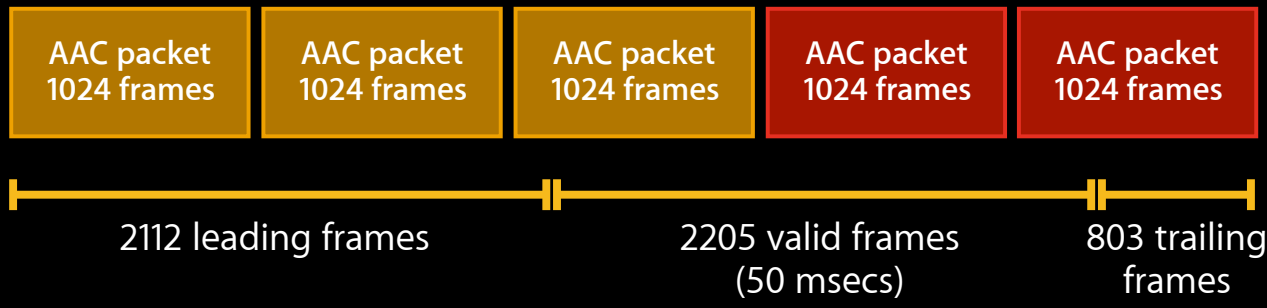
Decoding Compressed Audio



5 packets * 1024 frames per packet - 2112 leading - 803 trailing = 2205 valid frames

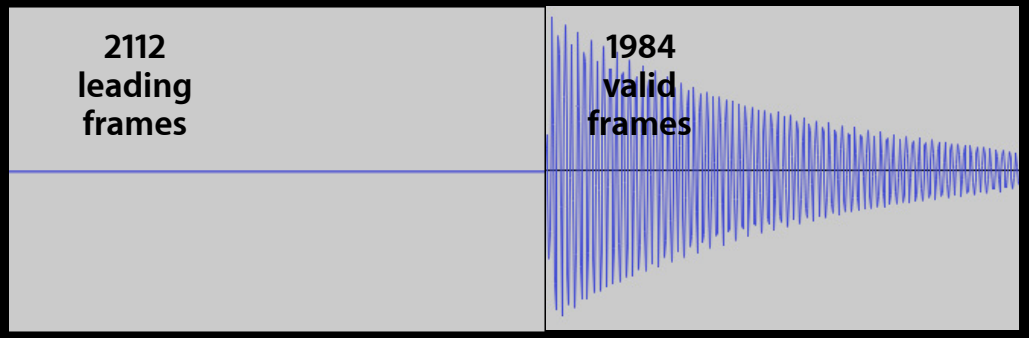
Decoding Compressed Audio

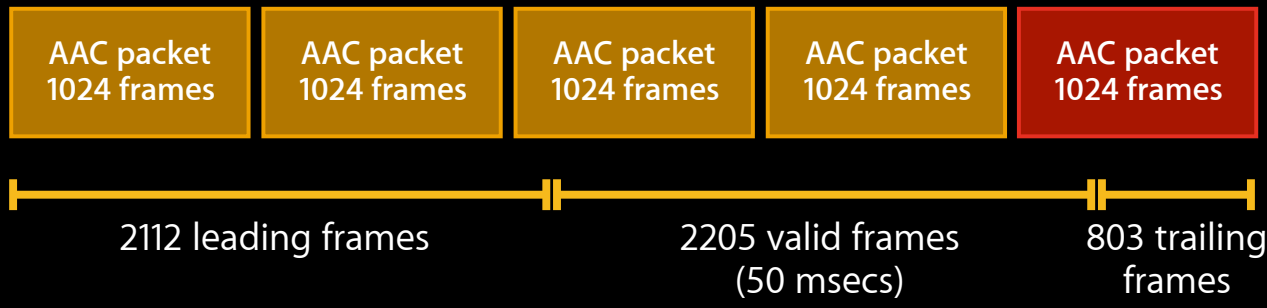




5 packets * 1024 frames per packet - 2112 leading - 803 trailing = 2205 valid frames

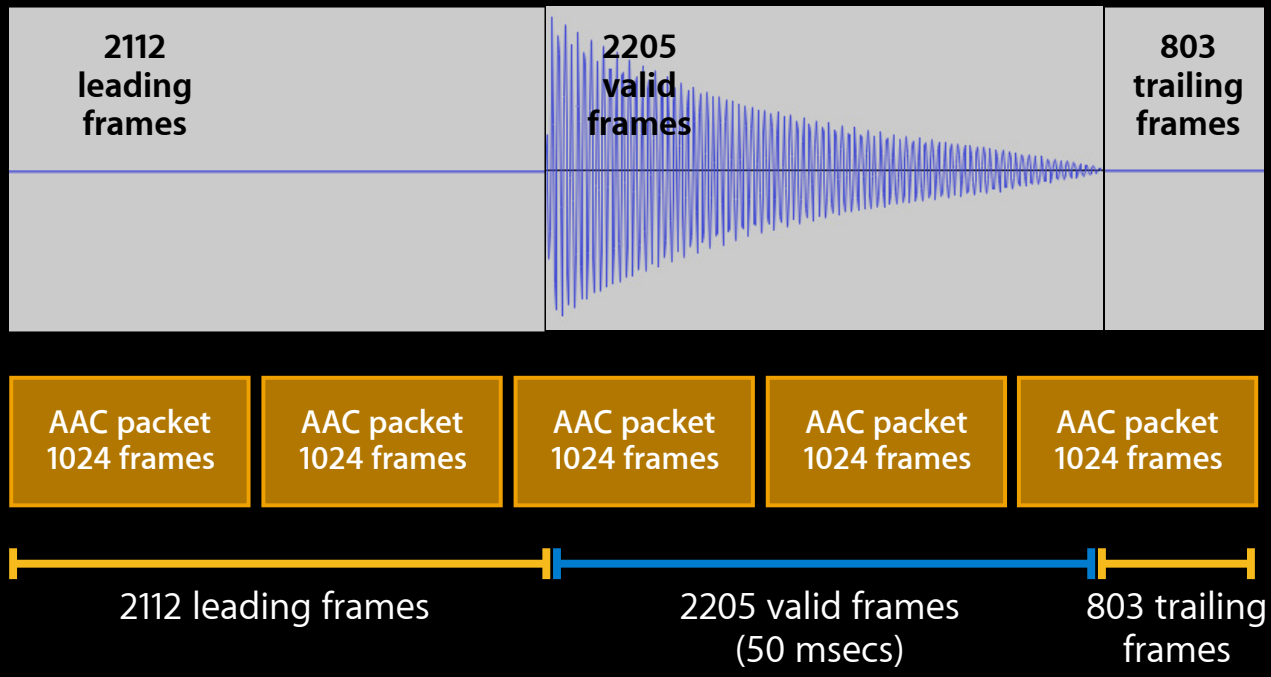
Decoding Compressed Audio





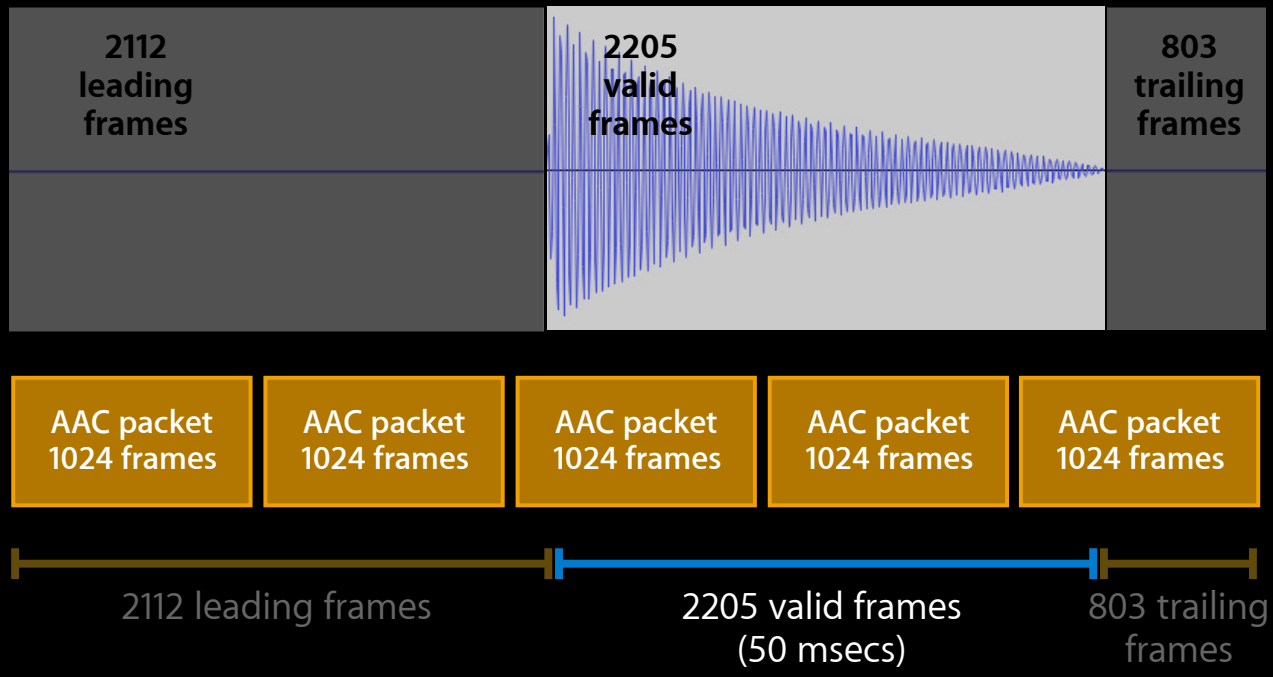
5 packets * 1024 frames per packet - 2112 leading - 803 trailing = 2205 valid frames

Decoding Compressed Audio



5 packets * 1024 frames per packet - 2112 leading - 803 trailing = 2205 valid frames

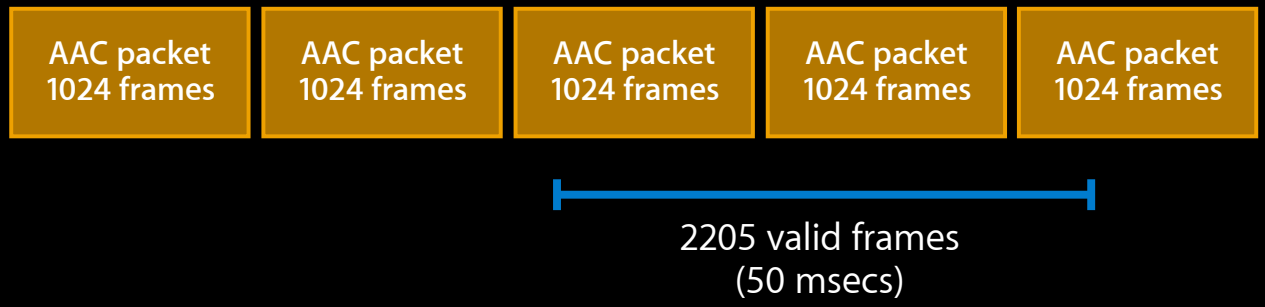
Decoding Compressed Audio



5 packets * 1024 frames per packet - 2112 leading - 803 trailing = 2205 valid frames

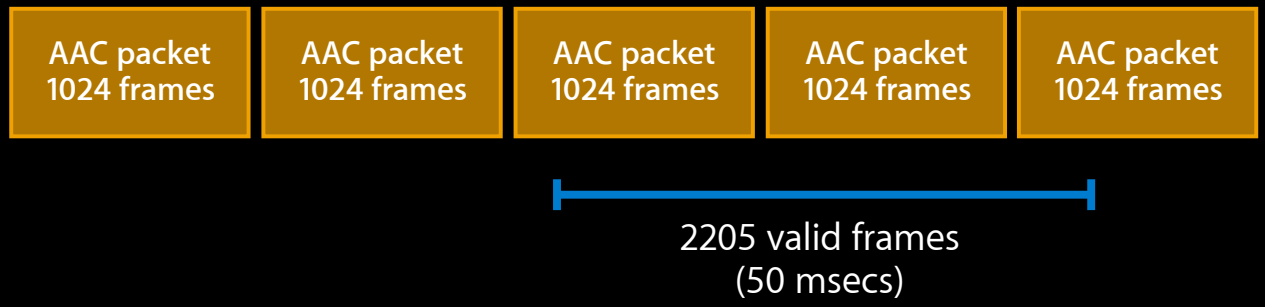
Working with Compressed Audio

- AVAudioPlayer handles this for you



Working with Compressed Audio

- AVAudioPlayer handles this for you
- Extended Audio File allows you to ignore it and deal with PCM



afinfo

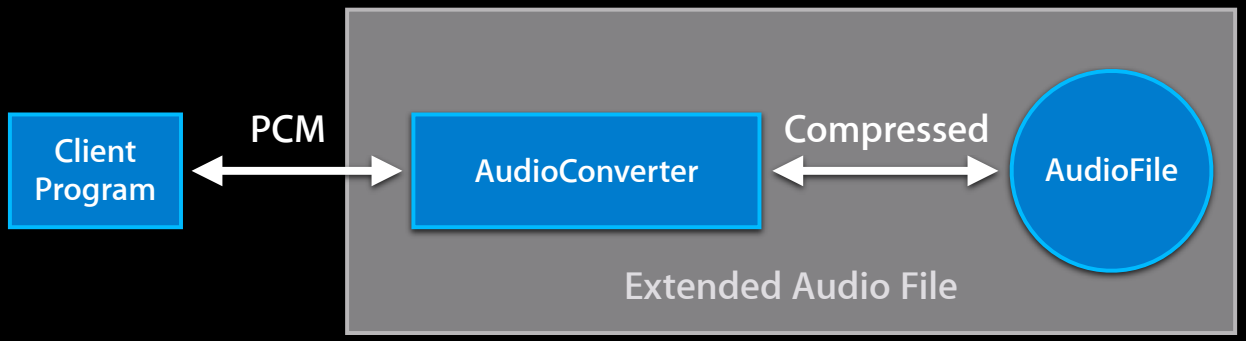
Leading and trailing frames

```
> afinfo tic.caf
File:           tic.caf
File type ID:   caff
Data format:     1 ch,  44100 Hz, 'aac ' (0x00000000) 0 bits/channel, 0
bytes/packet, 1024 frames/
packet, 0 bytes/frame
                no channel layout.
estimated duration: 0.05 sec
audio bytes: 488
audio packets: 5
audio 2205 valid frames + 2112 priming + 803 remainder = 5120
bit rate: 18604 bits per second
packet size upper bound: 224
audio data file offset: 4096
optimized
```

The Extended Audio File API

And compressed data

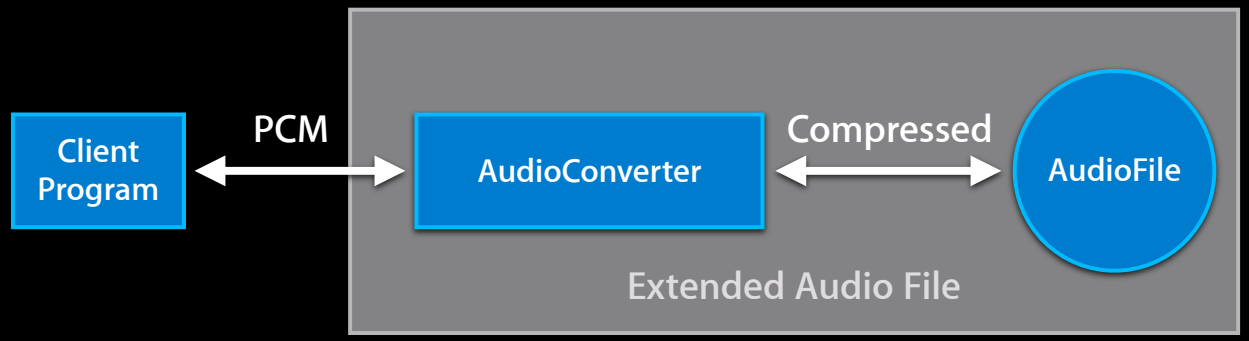
- Combines an AudioFile with an AudioConverter
  - Allows you to more easily read and write files in any supported format while treating the data like it was linear PCM



The Extended Audio File API

And compressed data

- Combines an AudioFile with an AudioConverter
  - Allows you to more easily read and write files in any supported format while treating the data like it was linear PCM
- Client side can just deal with the uncompressed data
  - All reading, writing, and file positions are handled in sample frames



Spatial Audio

A More "Complex" Game

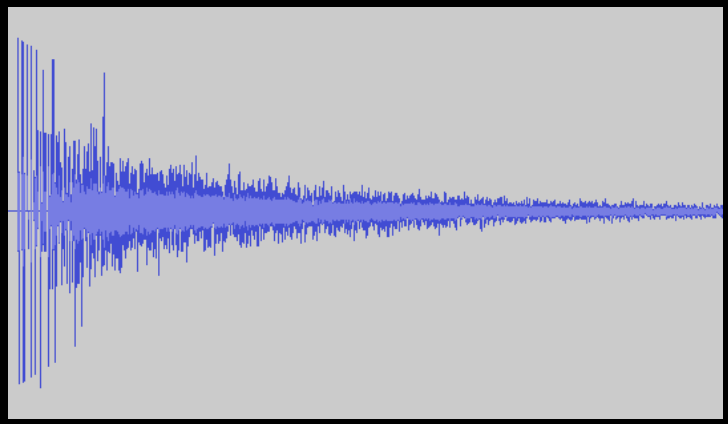
- A listener and multiple sources
- 3D audio
  - Panning, directional cues, reverberation, obstruction, occlusion
- Low latency

Spatial Audio Cues

- Interaural intensity difference
- Interaural time delay
- Head filtering
- Distance filtering

Reverberation

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

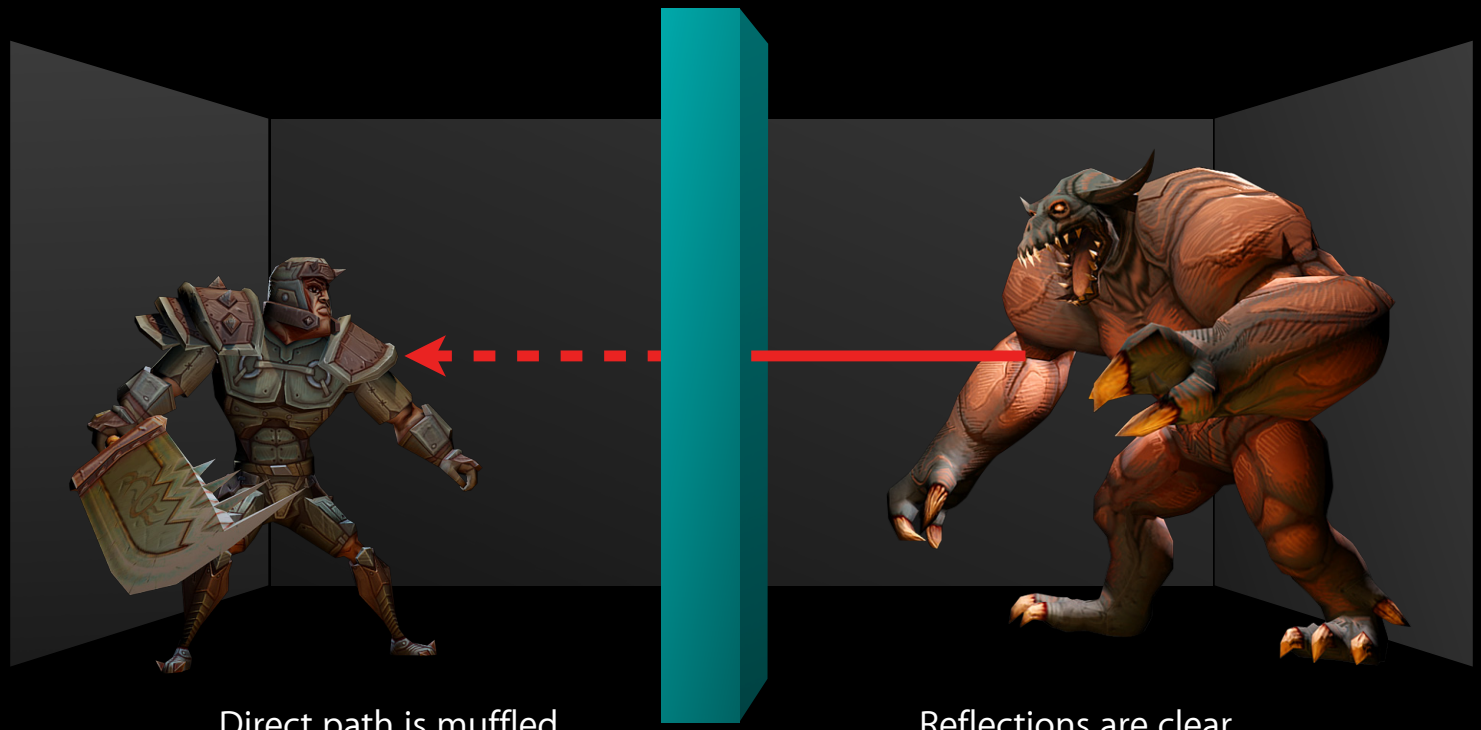


Obstruction and Occlusion

- Simulate filtering of sound due to objects in the environment that block propagation paths

Obstruction

Direct path is blocked

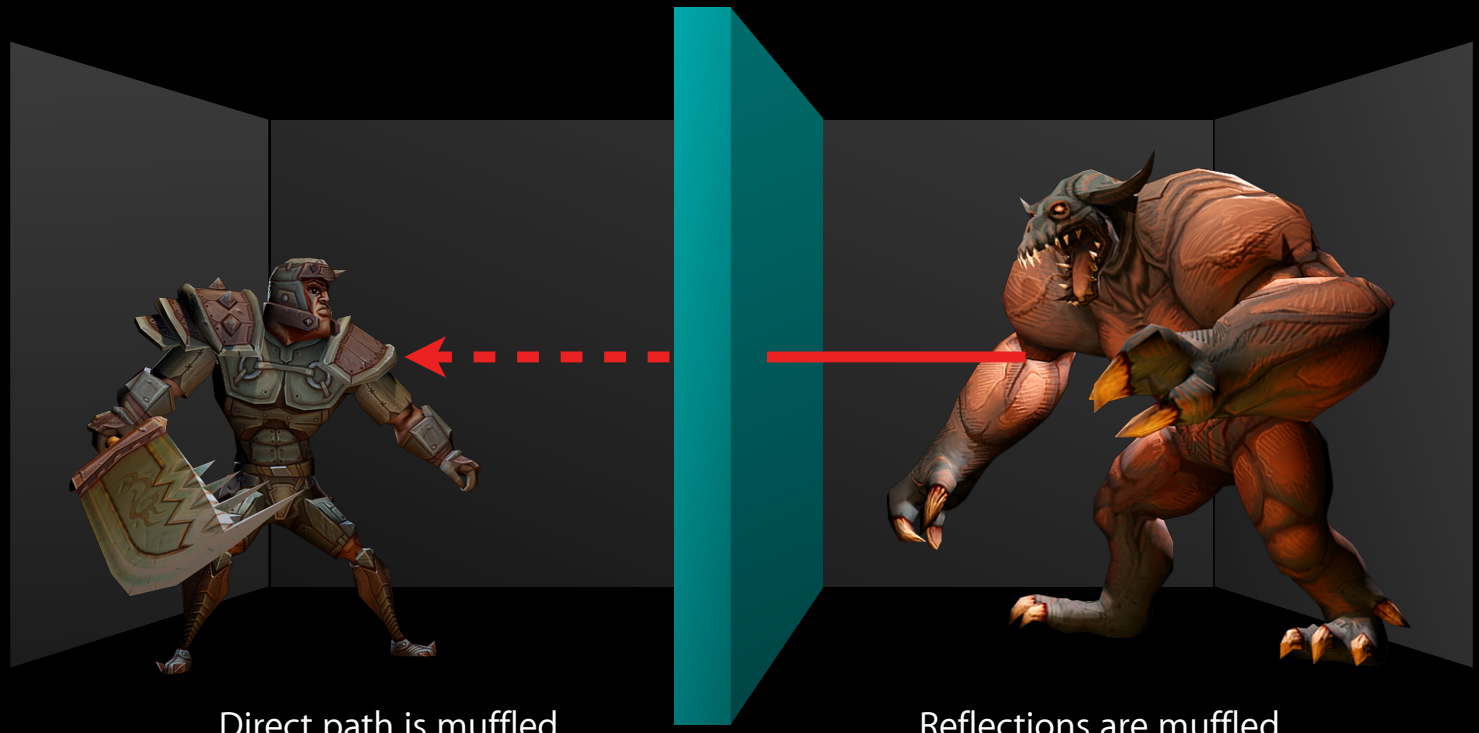


Direct path is muffled

Reflections are clear

Occlusion

Both direct and reverb paths are blocked



Direct path is muffled

Reflections are muffled

3D Mixer

- The 3D mixer is how spatial audio is supported on Mac OS X and iOS
- Several spatialization modes supported
- Reverb
- Filters for occlusion and obstruction

iOS 3D Mixer

Spatialization modes

- Equal power
- Spherical head
  - Interaural intensity difference
  - Interaural time-delay cue
  - Filtering due to head
  - Distance filtering

Mac OS X 3D Mixer

Spatialization modes

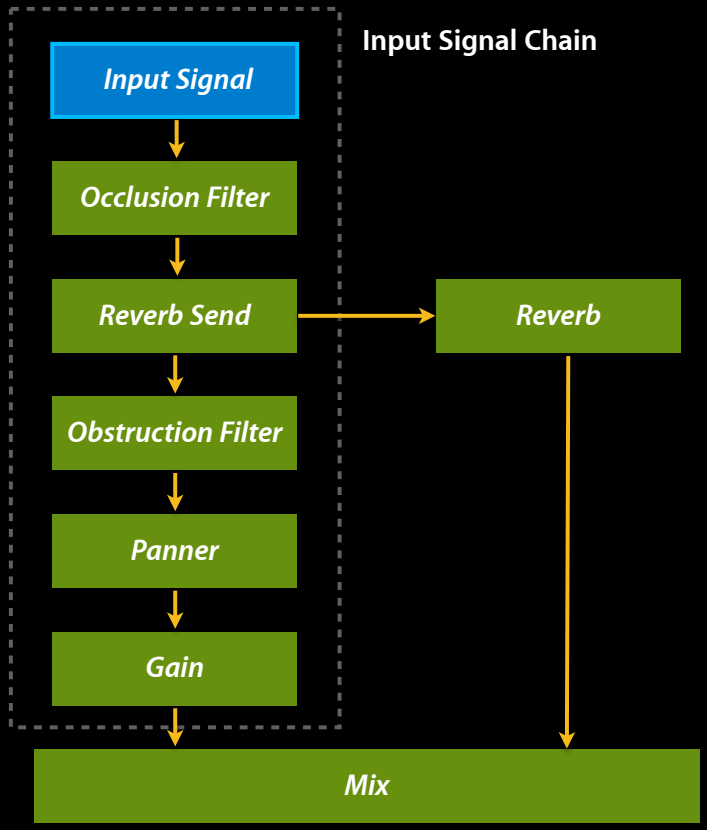
- Equal power
- Spherical head
- Head Related Transfer Function (HRTF)
- Sound field (multichannel)
- Vector-based panning (multichannel)

iOS 3D Mixer

New features for iOS 5

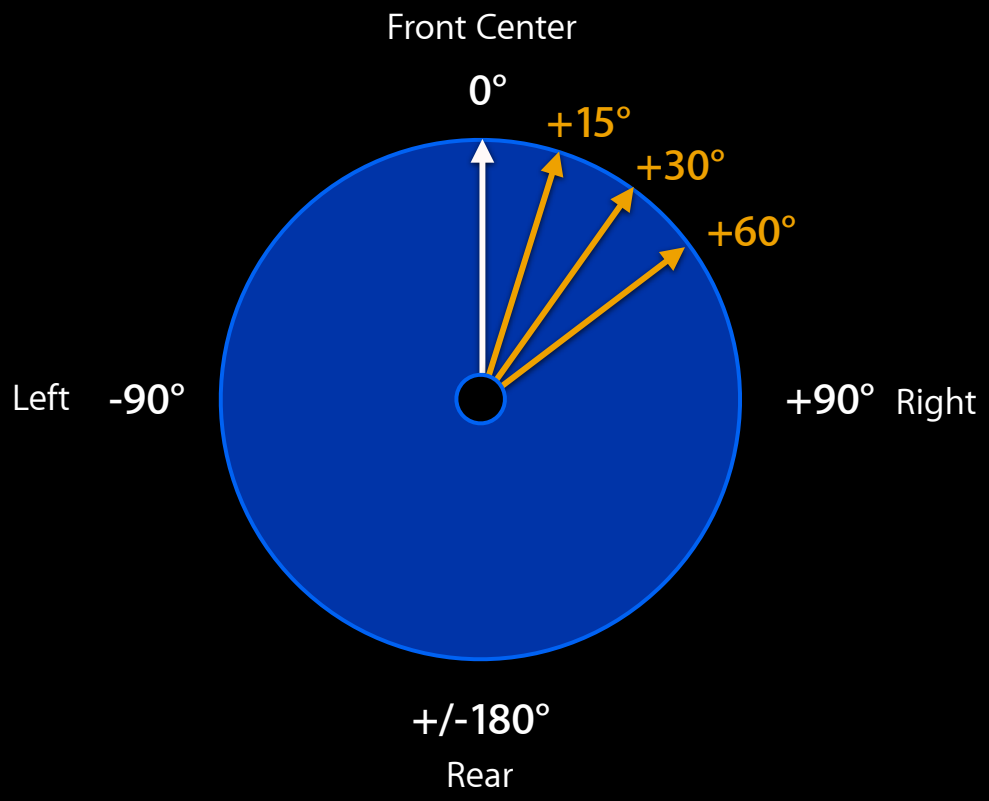
- Reverb
- Occlusion
  - Sound source is in an adjacent environment (room)
  - Both direct and reverb path are filtered
- Obstruction
  - Sound source is in the same environment but obstructed
  - Only direct path is filtered

3D Mixer Signal Path



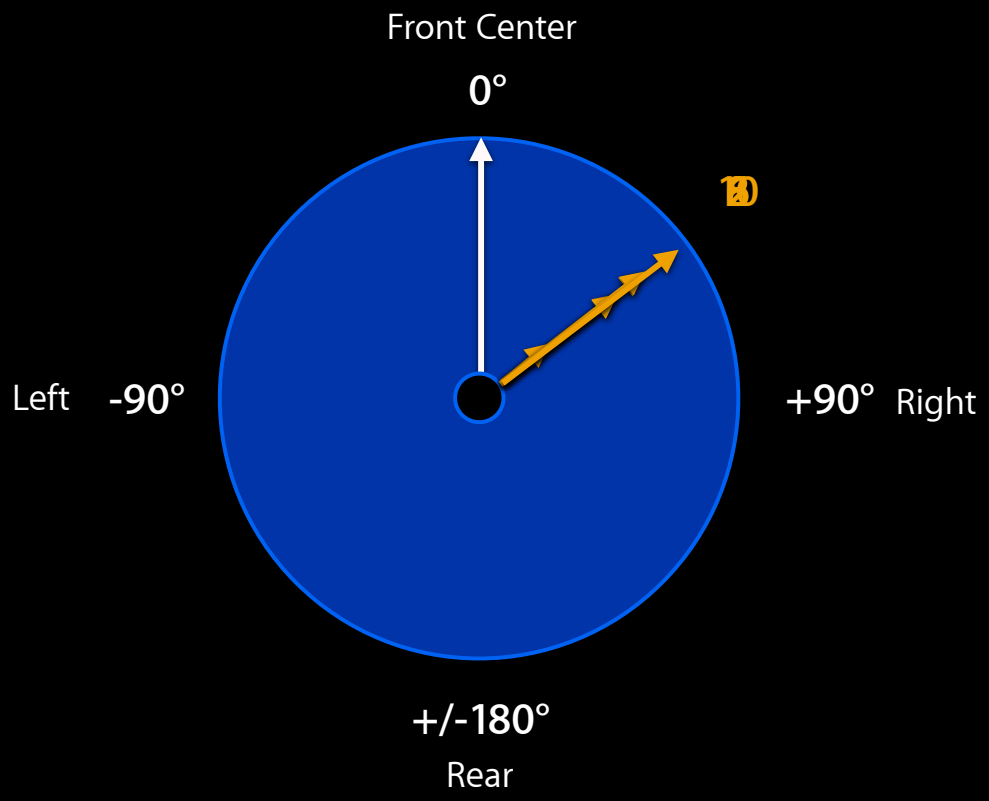
3D Mixer Parameters

Azimuth



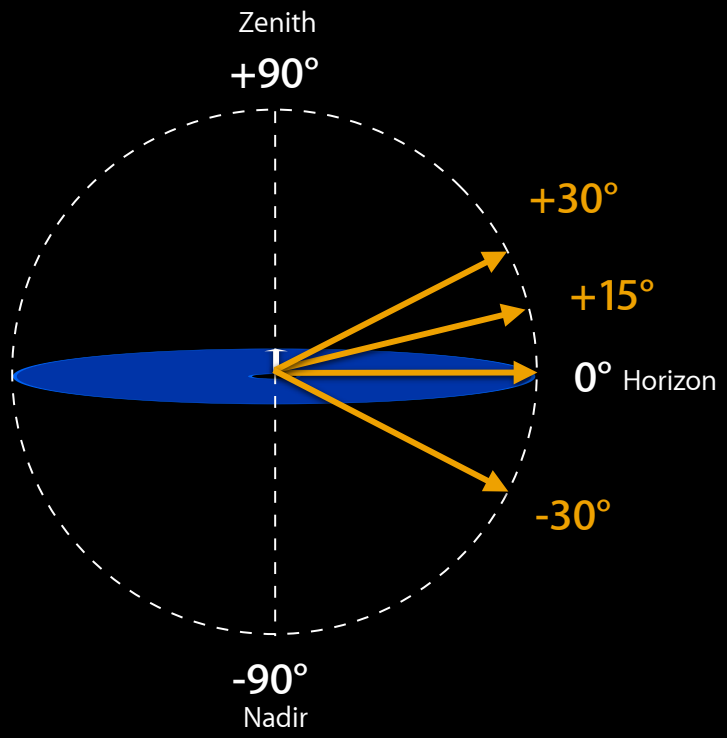
3D Mixer Parameters

Distance



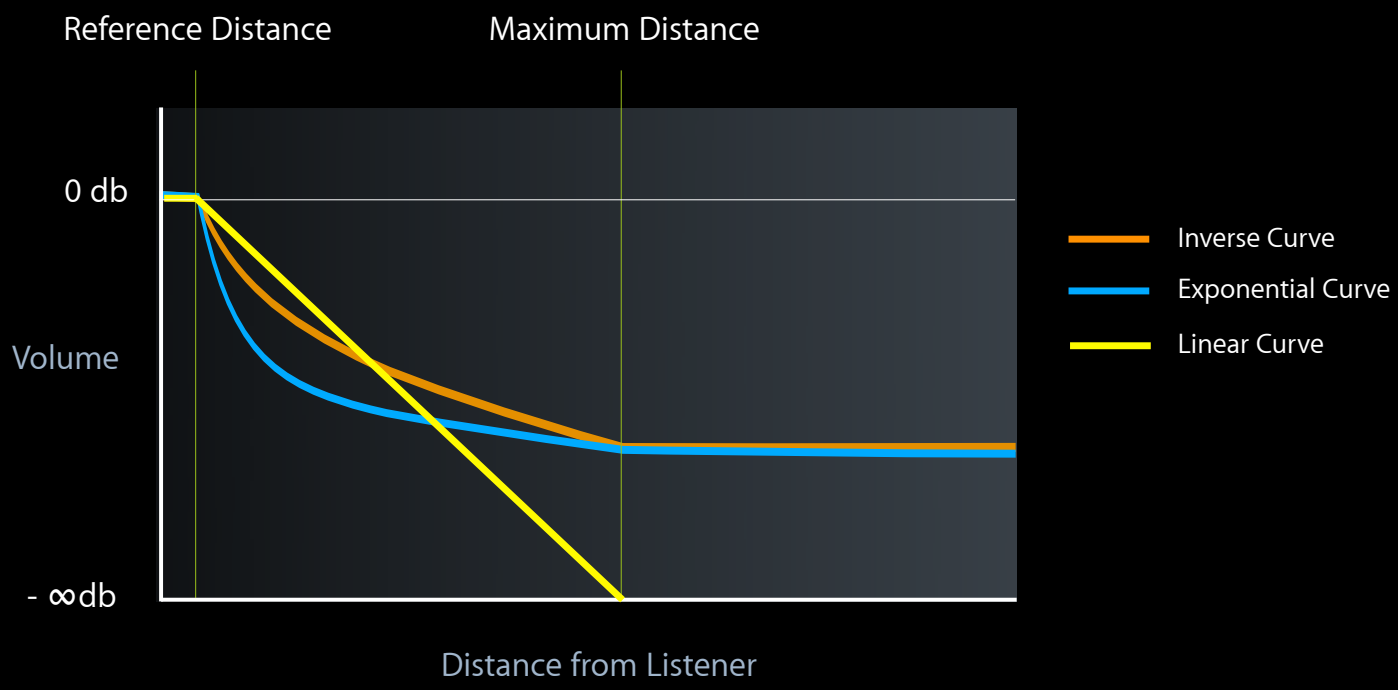
3D Mixer Parameters

Elevation



3D Mixer Property

Distance attenuation



Use This Stuff

- It will give depth to your game and make it more engaging
- The more you use this and the more feedback we get, the better this will be

OpenAL

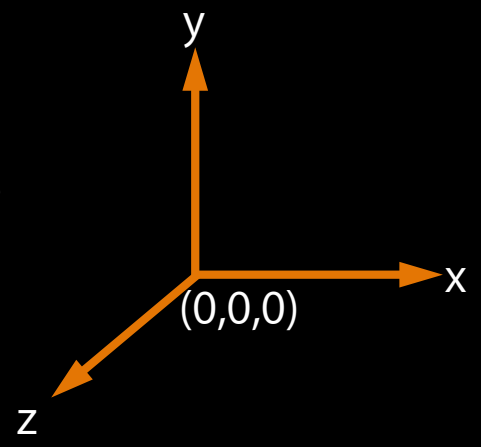
Kapil Krishnamurthy
Core Audio Engineering

What Is OpenAL?

- Open-standard audio API for spatial (3D) audio
- Designed to complement OpenGL
  - OpenGL Coordinate system
- Available on Mac OS X and iOS

OpenAL Coordinate System

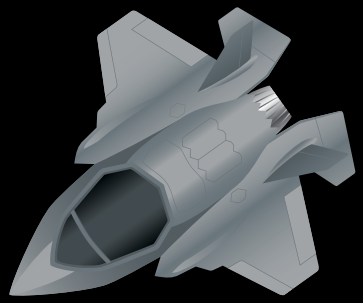
- Right-handed Cartesian coordinate system
  - Thumb : x axis (pointing right)
  - Index finger : y axis (pointing up)
  - Middle finger : z axis (pointing toward you)

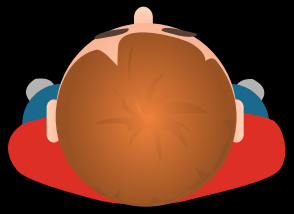


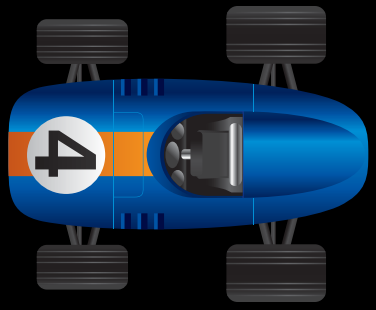
Setting Up OpenAL

Listener orientation

- Direction and rotation of listener's head
- Expressed as "up" and "at" vectors
- Objects are panned correctly



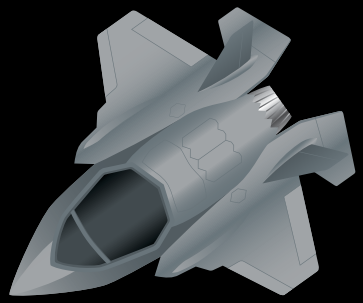


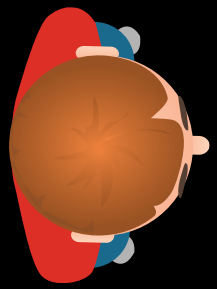


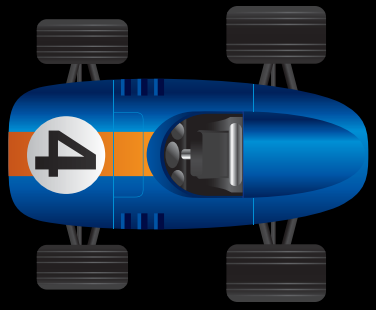
Setting Up OpenAL

Listener orientation

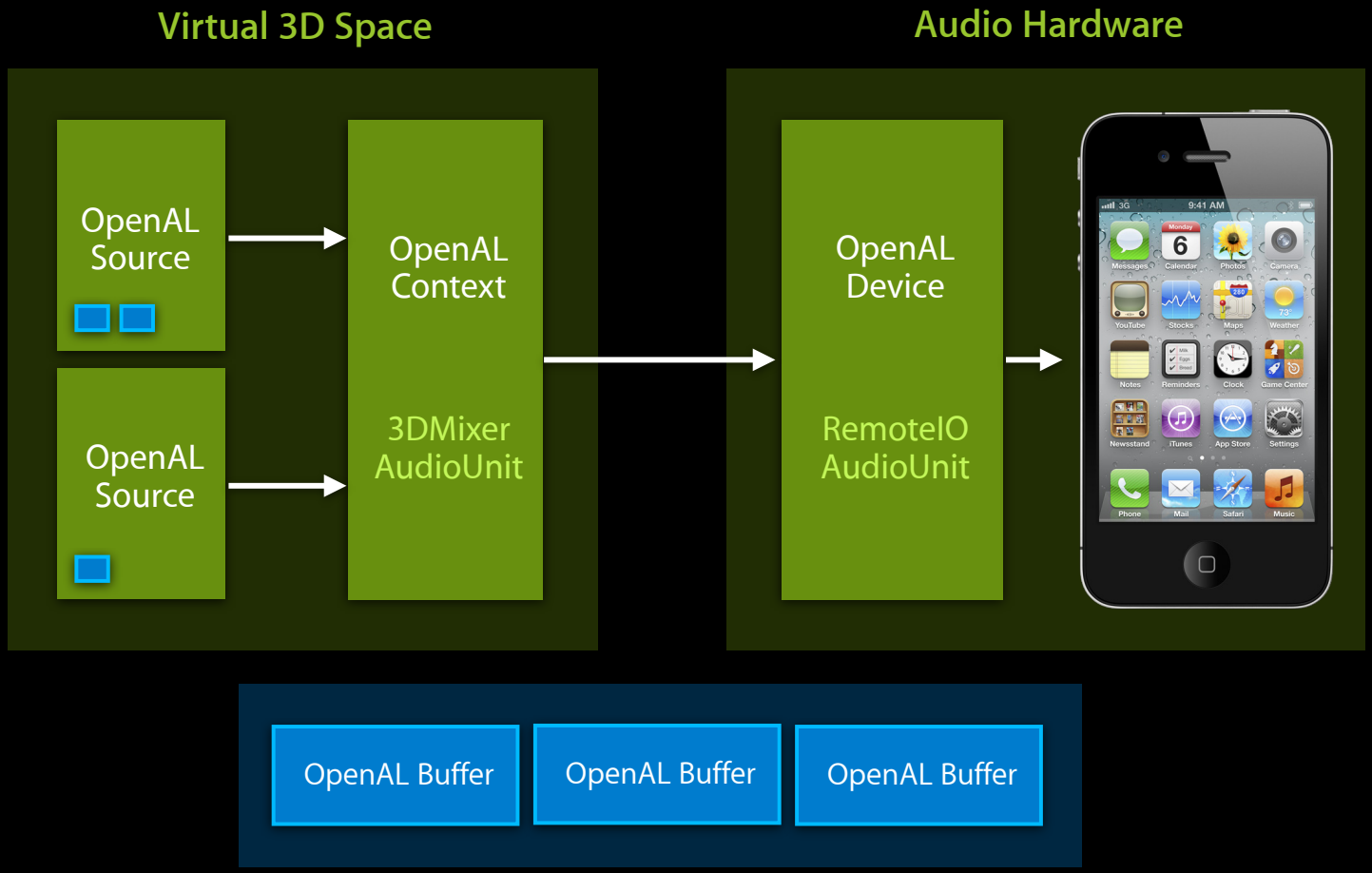
- Direction and rotation of listener's head
- Expressed as "up" and "at" vectors
- Objects are panned correctly



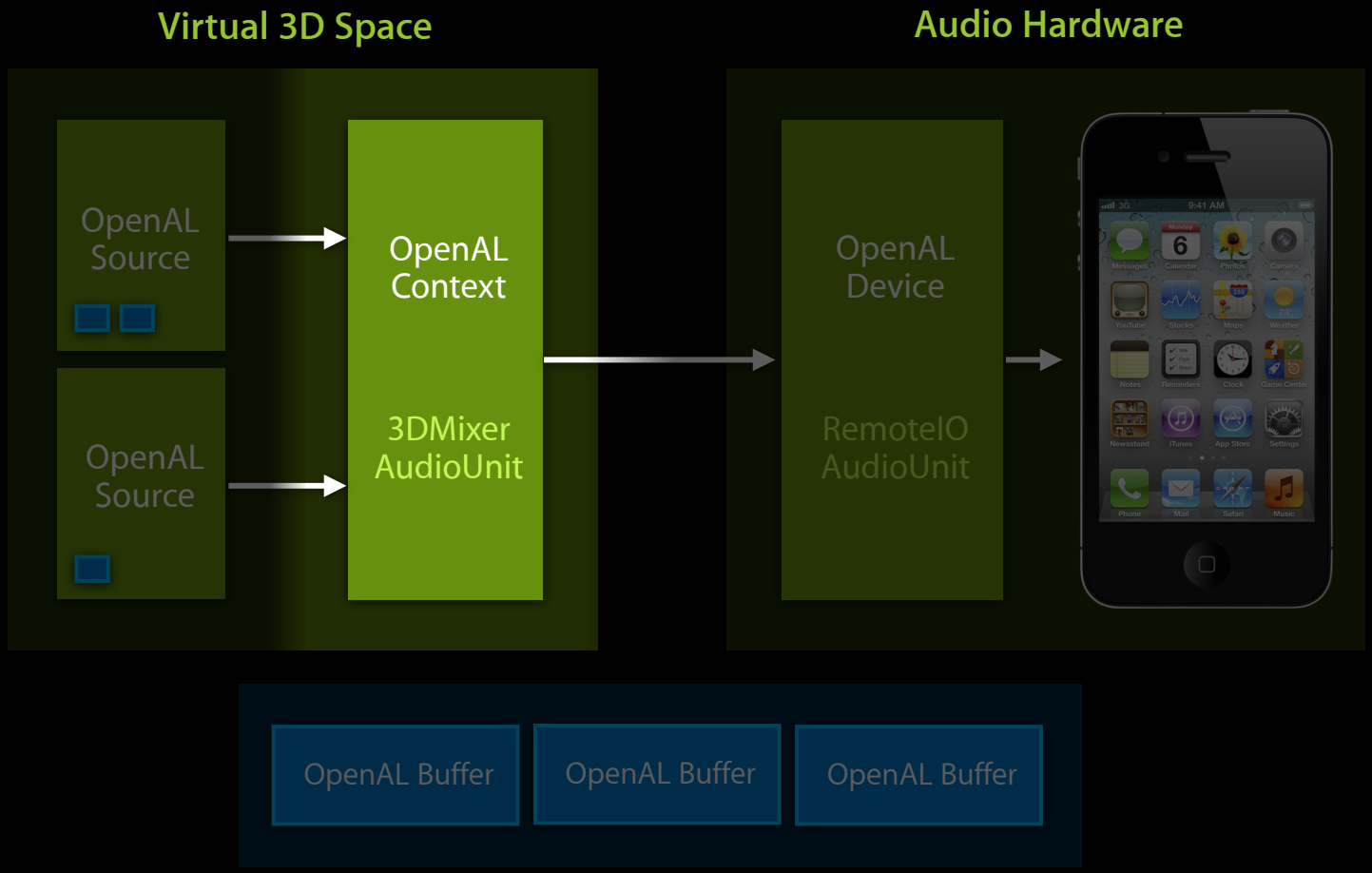




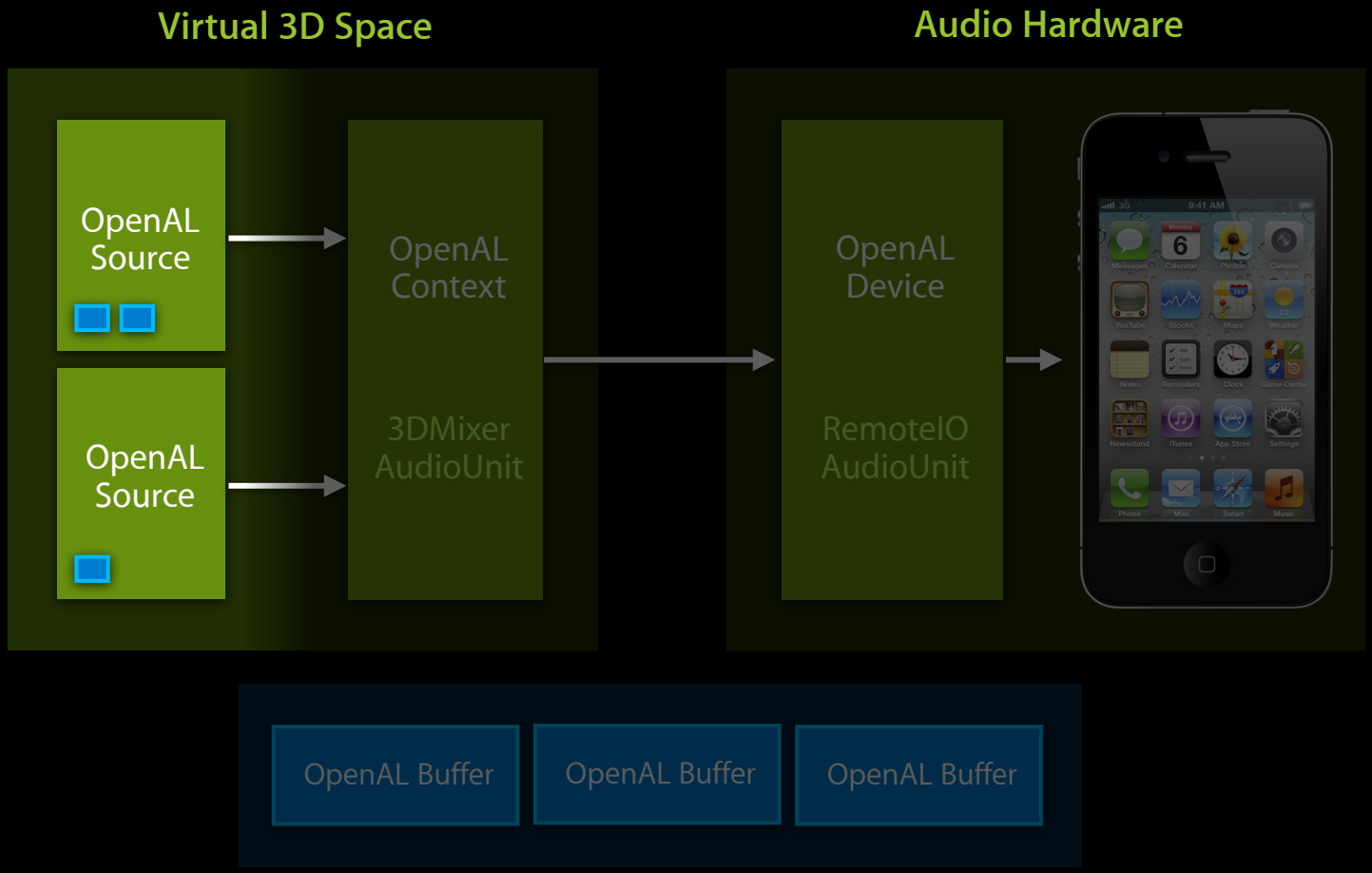
OpenAL—Review



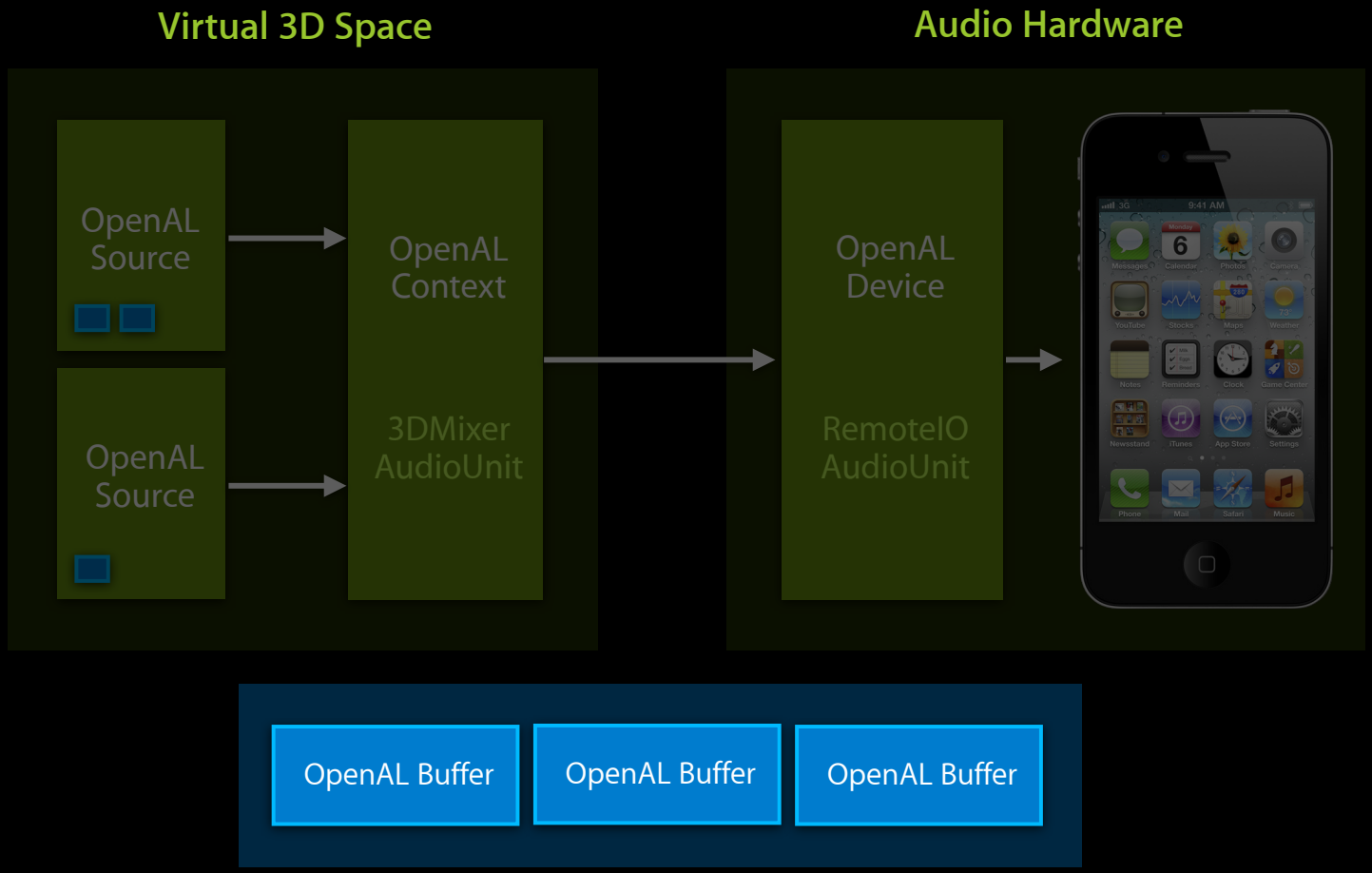
OpenAL—Review



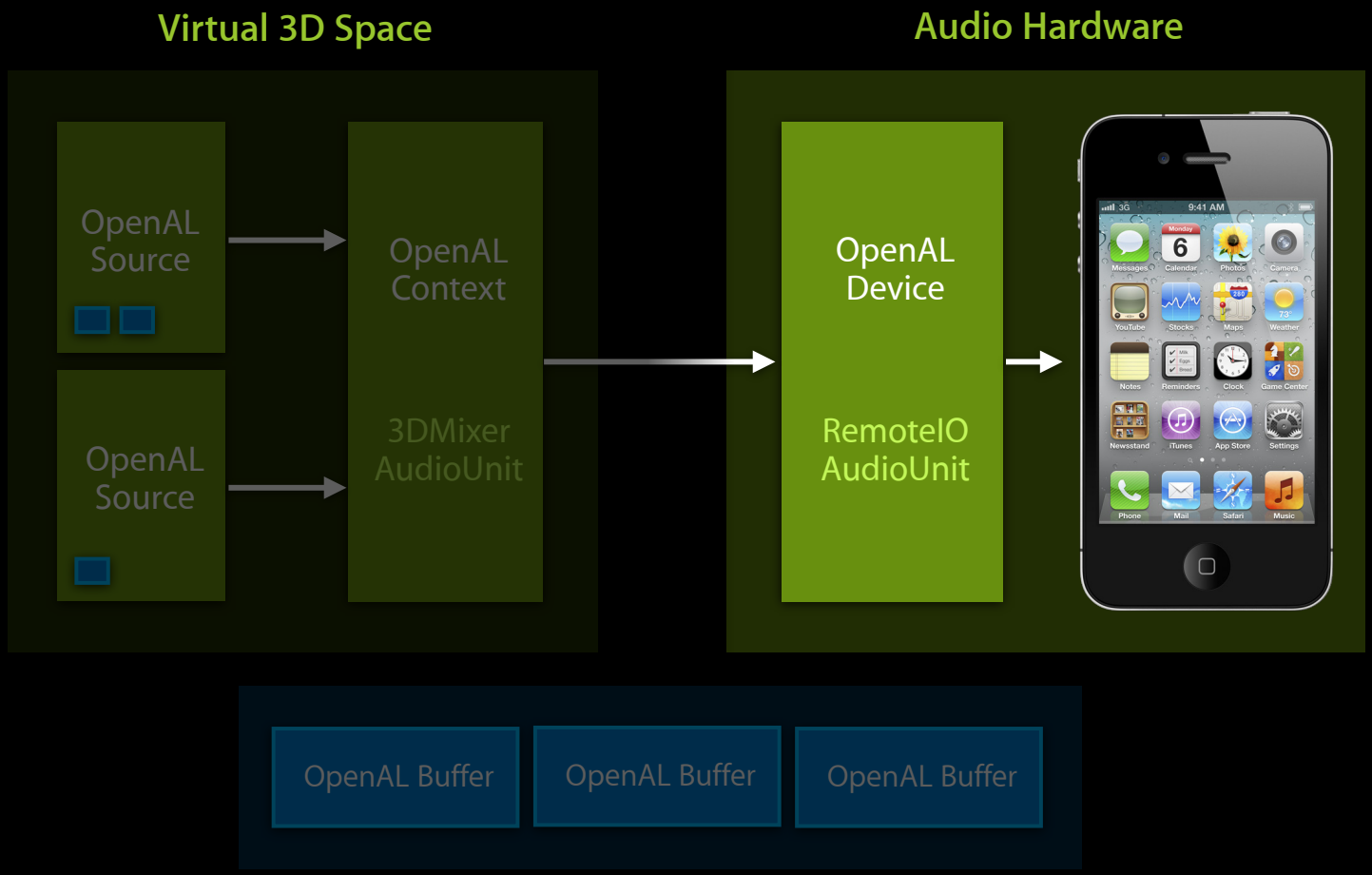
OpenAL—Review



OpenAL—Review



OpenAL—Review



Setting Up OpenAL

Creating the context (listener)

```
// Open an OpenAL Device
// Uses default system output device
device = alcOpenDevice(NULL);
```

```
// Create a new OpenAL Context (the mixer) for rendering
// Listener is implicit within the context
context = alcCreateContext(device, 0);
```

```
// Make the new context the Current OpenAL Context
alcMakeContextCurrent(context);
```

Setting Up OpenAL

Creating a source

```
// Create an OpenAL Source Object
alGenSources(1, &source);
```

Setting Up OpenAL

Creating a buffer

```
// Create an OpenAL Buffer Object to store audio data
alGenBuffers(1, &buffer);


// Get Some Audio Data with ExtAudioFile...


// Fill the buffer with audio data
alBufferDataStatic(buffer, AL_FORMAT_MONO16, dataPtr, dataSize, 22050);
```

Setting Up OpenAL

Creating a buffer

```
// Create an OpenAL Buffer Object to store audio data
alGenBuffers(1, &buffer);


// Get Some Audio Data with ExtAudioFile...


// Fill the buffer with audio data
alBufferDataStatic(buffer, AL_FORMAT_MONO16, dataPtr, dataSize, 22050);
```

Setting Up OpenAL

Using ExtAudioFile to load data into OpenAL buffer

```
//Open the file
err = ExtAudioFileOpenURL(url, &xaf);
```

Setting Up OpenAL

Using ExtAudioFile to load data into OpenAL buffer

```
// Set the client format to the OpenAL format: AL_FORMAT_MONO16
UInt32 flags = kAudioFormatFlagIsSignedInteger |
               kAudioFormatFlagIsPacked        |
               kAudioFormatFlagIsNativeEndian;

AudioStreamBasicDescription format = { 22050., kAudioFormatLinearPCM,
                                       flags, 2, 1, 2, 1, 16, 0 };
err = ExtAudioFileSetProperty(xaf, kExtAudioFileProperty_ClientDataFormat,
                              size, &format);
```

Setting Up OpenAL

Using ExtAudioFile to load data into OpenAL buffer

```
//Allocate buffers and read data
UInt32 dataSize = numFrames * sizeof(SInt16);
SInt16* dataPtr = (SInt16*) malloc(dataSize);

AudioBufferList abl;
abl.mNumberBuffers = 1;
abl.mBuffers[0].mDataByteSize = dataSize;
abl.mBuffers[0].mData = dataPtr;

//Read data into buffers using ExtAudioFileRead
err = ExtAudioFileRead(xaf, &numFrames, abl);
```

Setting Up OpenAL

Creating a buffer

```
// Create an OpenAL Buffer Object to store audio data
alGenBuffers(1, &buffer);


// Get Some Audio Data with ExtAudioFile...


// Fill the buffer with audio data
alBufferDataStatic(buffer, AL_FORMAT_MONO16, dataPtr, dataSize, 22050);
```

Setting Up OpenAL

Creating a buffer

```
// Create an OpenAL Buffer Object to store audio data
alGenBuffers(1, &buffer);


// Get Some Audio Data with ExtAudioFile...

// Fill the buffer with audio data
alBufferDataStatic(buffer, AL_FORMAT_MONO16, dataPtr, dataSize, 22050);


// attach OpenAL Buffer to OpenAL Source
alSourcei(source, AL_BUFFER, buffer);
```

Setting Up OpenAL

Set source and listener attributes

```
//Set some source attributes
alSourcefv(source, AL_POSITION, source_position);
alSourcef (source, AL_REFERENCE_DISTANCE, 5.0f);
alSourcei (source, AL_LOOPING, AL_TRUE);
```

Setting Up OpenAL

Set source and listener attributes

```
//Set some listener attributes
alListenerfv(AL_POSITION, listener_position);
alListenerfv(AL_ORIENTATION, listener_orientation);
```

Setting Up OpenAL

Play your sound

```
// Begin playing our audio source
alSourcePlay(source);
```

Setting Up OpenAL

Play your sound

```
// Begin playing our audio source
alSourcePlay(source);

// then during gameplay...
// move source position
alSourcefv(source, AL_POSITION, source_position);

// move listener position
alListenerfv(AL_POSITION, listener_position);
```

OpenAL Extensions

OpenAL—Extensions

- What are OpenAL extensions?
  - Mechanism for augmenting API set
- Determine if extension is present at runtime
- Get pointers for extension functions

OpenAL—Extensions

- ASA Extension (Apple Spatial Audio)
  - `ALC_EXT_ASA`
  - Reverb, Occlusion, and Obstruction
  - New in iOS 5
  - Already available on Mac OS X

OpenAL—Extensions

- ASA Extension (Apple Spatial Audio)
  - `ALC_EXT_ASA`
  - Reverb, Occlusion, and Obstruction
  - New in iOS 5
  - Already available on Mac OS X
- Source Notifications Extension
  - `AL_EXT_SOURCE_NOTIFICATIONS`
  - Callback Mechanism for Source State
  - New in iOS 5 and Mac OS 10.7

OpenAL—Extensions

ASA extension

- Spatial effects
  - Reverb
    - Stock room presets
  - Occlusion
  - Obstruction
- Set/Get listener/source properties
  - `alcASAGetListener()` and `alcASAGetSource()`
  - `alcASASetListener()` and `alcASASetSource()`

OpenAL—Extensions

ASA extension: listener reverb properties

`ALC_ASA_REVERB_ON`

`ALC_ASA_REVERB_GLOBAL_LEVEL`

- Overall reverb level (-40 dB to 40 dB)

OpenAL—Extensions

ASA extension: listener reverb type properties

`ALC_ASA_REVERB_ROOM_TYPE`

- Predefined room types

`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`

OpenAL—Extensions

ASA extension: listener reverb EQ properties

- Parametric EQ settings for reverb
- Settable in real time

`ALC_ASA_EQ_GAIN`

  - Cut/boost level of EQ

`ALC_ASA_EQ_BANDWIDTH`

  - Frequency bandwidth in octaves

`ALC_ASA_EQ_FREQ`

  - Center frequency of EQ band

OpenAL—Extensions

ASA extension: source properties

`ALC_ASA_REVERB_SEND_LEVEL`

- Wet/dry reverb mix
- 0.0 = no reverb – 1.0 = only reverb

OpenAL—Extensions

ASA extension: source properties

`ALC_ASA_OCCLUSION`

- Low pass filter applied to source
- 0.0 dB (no effect) to -100.0 dB (most occlusion)
- Applied to pre-reverb send

OpenAL—Extensions

ASA extension: source properties

`ALC_ASA_OBSTRUCTION`

- Low pass filter applied to source
- 0.0 dB (no effect) to -100.0 dB (most obstruction)
- Applied to post-reverb send

OpenAL—Extensions

Using the ASA extension

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

//Set a source property
ALfloat level = 0.4;
alcASASetSourceProc(alcGetEnumValue(NULL, "ALC_ASA_REVERB_SEND_LEVEL"),
source, &level, sizeof(level));
```

Demo

OpenAL—Extensions

Source-state notifications

- Eliminates the need for polling to check for:
  - Source-state change
  - Number of buffers processed
- Notifies the user via a callback mechanism

OpenAL—Extensions

Source-state notifications

- Notification types
  - AL_SOURCE_STATE
    - AL_INITIAL
    - AL_PLAYING
    - AL_PAUSED
    - AL_STOPPED
  - AL_BUFFERS_PROCESSED
  - AL_QUEUE_HAS_LOOPED

OpenAL—Extensions

Polling for changes—native mechanism

```
ALint   numBuffersProcessed  = 0;
while (numBuffersProcessed < 1)
{
  alGetSourcei(mySourceID, AL_BUFFERS_PROCESSED, numBuffersProcessed);
  usleep(1000);
}
```

OpenAL—Extensions

Using source-state notifications

- Register source for notifications

```
//Register for a notification
error = alSourceAddNotificationProc(source, AL_BUFFERS_PROCESSED,
(alSourceNotificationProc) HandleNotification, NULL);
```

- Handle notification calls in application

```
void HandleNotification(ALuint source, ALuint notificationID,
alSourceNotificationProc notifyProc, ALvoid* userData)
{
  //Do something based on the source and notificationID
}
```

OpenAL Summary

- Coordinate system/listener orientation
- Objects—context (listener), sources, buffers, device
- OpenAL Extensions
  - Apple Spatial Audio Extension (ASA)
  - Source Notifications Extension

AudioSession and Games

AudioSession

- AudioSession category
- Detecting background audio
- Responding to interruptions

AudioSession Category

Ambient

- Audio obeys ringer switch
- Audio obeys screen lock
- Solo or not?

Is the Game's Audio Primary?

Detecting background audio


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);
```

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)

Responding to Interruptions

AVAudioPlayer

- AVAudioPlayers are stopped
- Override delegate methods if:
  - UI needs to be updated
  - Sounds need to be restarted

Responding to Interruptions

AVAudioPlayer delegate

- `- (void)audioPlayerBeginInterruption:(AVAudioPlayer*) player`
  - Playback automatically stops
  - Update UI if needed
- `- (void)audioPlayerEndInterruption:(AVAudioPlayer*)player`
  `withFlags:(NSUInteger)flags`
  - Resume playback of sounds if needed
  - Update UI if needed

Responding to Interruptions

OpenAL

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

Responding to Interruptions

OpenAL and AVAudioSession

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

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

AudioSession Summary

- AudioSession category
- Detecting background audio
- Handling interruptions

To Sum It All Up…

- Simple game
- Understanding your assets
- Complex game
- AudioSession

Related Sessions

Audio Session Management for iOS	Marina Wednesday 11:30AM

Labs

Audio Lab	Graphics, Media and Games Lab B Tuesday 2:00PM
Audio Lab	Graphics, Media and Games Lab C Wednesday 2:00PM

More Information

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

Eryk Vershen
Media Technologies Evangelist
evershen@apple.com

Audio Programming Guides
iPhone Dev Center
http://developer.apple.com/devcenter/ios/

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


WWDC2011