Fundamentals of Digital Audio

For Mac and iPhone

James McCartney
Core Audio Engineer

Eric Allamanche
Core Audio Engineer

Agenda

Audio Processing Basics

Voice Processing Audio Unit and Audio Codecs

Audio Processing Basics

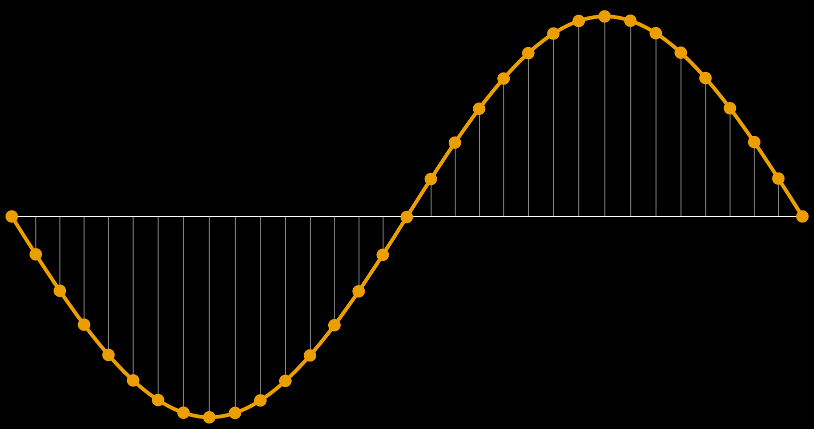
James McCartney
Core Audio Engineer

Audio Processing
Topics

- About audio representation formats
- Converting audio
- Processing audio

Representing Audio Digitally

- An analog waveform is sampled at the sampling rate
- Each sample is a number
- Two numbers per cycle of a sine wave required to reconstruct
- Highest reproducible frequency is half of the sampling rate



Audio Formats

- Linear PCM
- Nonlinear PCM
- Packetized compressed formats

LPCM

- The most direct way to represent the sampled audio
  - Linear Pulse Code Modulation or LPCM
    - Just store the sampled numbers in binary form

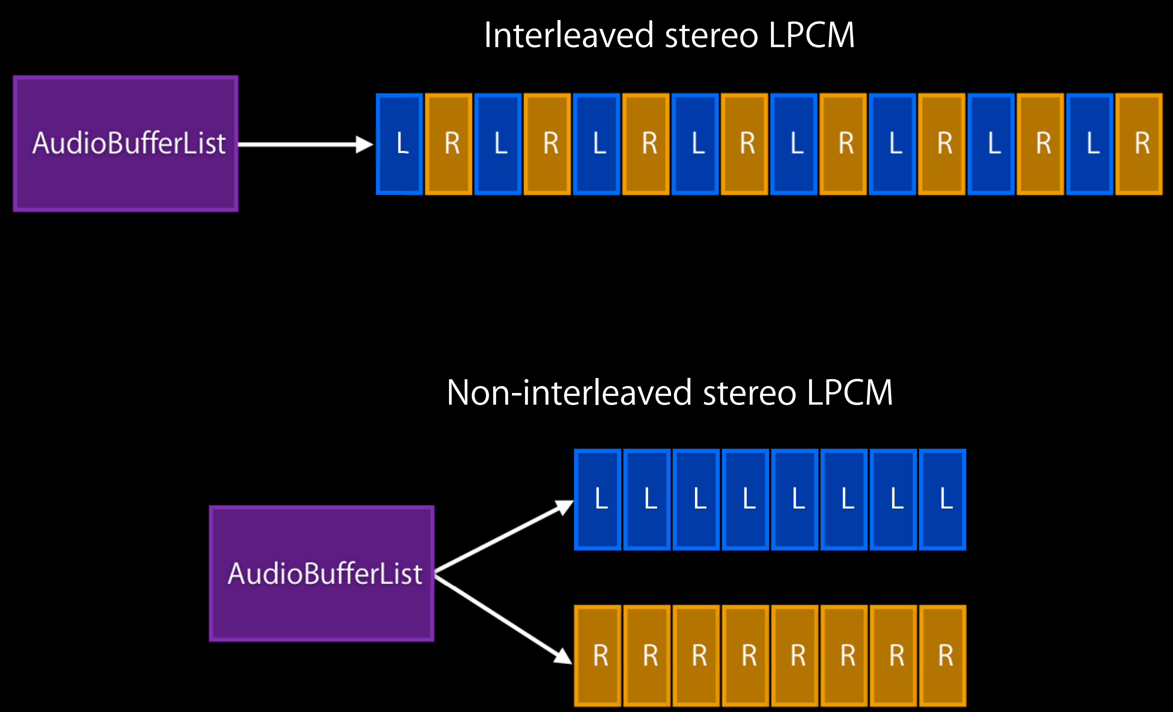
Attributes of LPCM

- Are the numbers integer or floating point?
- Are integers signed or unsigned?
- How many bits are in each number?
  - Bit depth
- What order are the bytes of the number stored in?
  - Big endian or little endian
- How many channels?
- Are channels stored together or separately?
  - Interleaved or non-interleaved
- Are the bytes packed or is there padding?

Samples, Frames, Packets

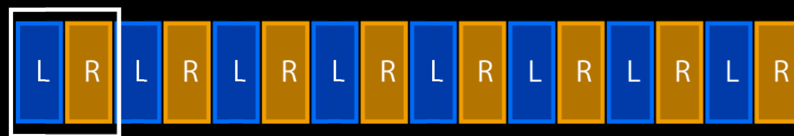
- Sample
  - One sample of a waveform
- Frame
  - A collection of samples for each channel
- Packet
  - The smallest cohesive unit of data for a format
    - For LPCM, one packet equals one frame
    - For compressed formats, one packet is a group of bytes that decompress to some number of frames of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

Interleaving

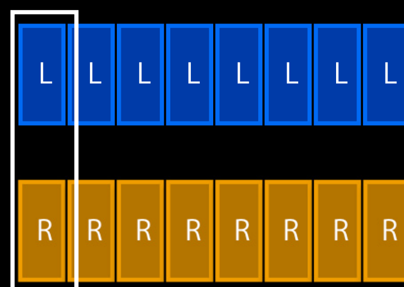


Frames of LPCM

One frame of interleaved stereo LPCM



One frame of non-interleaved stereo LPCM



Quality—Sample Rate

Common sample rates

Sample Rate	Quality
8 kHz	Narrow-band speech
16 kHz	Wide-band speech
44.1 kHz	CD quality
48 kHz	Digital audio tape, etc.
96 kHz	Pro quality
192 kHz	Ultimate marketing quality

Quality—Sample Rate

- Human hearing extends to ~20 kHz
  - Sampling rates above 48 kHz are not needed to improve audible quality
  - Reasons for rates above 48 kHz
    - Eliminating phase effects caused by steep filters
    - Eliminating the need for steep filters
    - Allowing frequencies that would otherwise alias to fall in upper octaves where they can be filtered out

Quality—Bit Depth

- Bit depth determines signal to noise ratio
- What is Signal to Noise Ratio (SNR)?
  - Amplitude of signal divided by amplitude of noise
  - The noise is quantization error
  - Every 6 decibels (dB) is roughly a factor of two
  - Every bit adds 6 dB to the SNR
  - In integer formats, SNR is amplitude dependent
    - Quieter signals have worse SNR
  - In floating point, SNR is independent of amplitude

Quality—Bit Depth

Bit depth and Signal to Noise Ratio

Format	Common Use	Signal to Noise Ratio
8 bit integer	Old games, 1980s gear	48 dB SNR *
16 bit integer	CD quality	96 dB SNR *
24 bit integer	Newer consumer format	144 dB SNR *
32 bit integer as 8.24 fixed point	AudioUnitSampleType iPhone OS	144 dB SNR * 42 dB headroom
32 bit floating point	AudioUnitSampleType Mac OS X	144 dB SNR for any amplitude unlimited dynamic range

** for full-scale signal*

Quality

- Quality, once lost, cannot be added back
- None of the following will increase quality:
  - Converting to a higher sample rate
  - Converting to a higher bit depth
  - Re-encoding compressed data to a higher bit rate
  - Re-encoding compressed data to a better codec
    - If you have the original uncompressed source then re-encoding is an option

Nonlinear PCM

- Instead of storing the number, the logarithm of the number is stored
- Increases the SNR of quiet signals at the expense of the SNR of loud signals
- Two common algorithms:
  - μ-law
  - A-law
  - Both encode audio in 8 bits per sample

Packetized Compressed Formats

- A group of frames is compressed into a packet
- Packets often have dependencies on preceding packets
  - Butt-splicing compressed packets will often cause a glitch
- More on compressed formats in next section

AudioStreamBasicDescription

- Struct used to describe audio formats throughout CoreAudio
- Has been covered extensively in previous WWDCs
  - How to fill one out
  - How to get one from other APIs
    - AudioFormat
    - AudioFile
    - AudioConverter
- If you use the AVFoundation classes you can avoid them

Converting Audio

Converting Between Formats

- AudioConverter API is used to convert between formats
  - LPCM ⟷ LPCM
    - Sample rate conversion
    - Bit depth change
    - Integer ⟷ floating point
    - Interleaved ⟷ non-interleaved
    - etc.
  - LPCM ⟶ compressed (encoding)
  - Compressed ⟶ LPCM (decoding)

Converting Between Formats

Some APIs have AudioConverters within them

API	Function	Converts Between
AudioQueue	Play/record buffers of audio to/from hardware	Client and hardware format
ExtendedAudioFile	Write/read audio to/from files	Client and file format
AVAudioPlayer	Play a file to hardware	File and hardware format
AVAudioRecorder	Record a file from hardware	Hardware and file format

Sample Rate Conversion Quality

- CoreAudio provides quality levels for sample rate conversion

`kAudioConverterSampleRateConverterComplexity`

  - Linear
  - Normal
  - Mastering (desktop)

`kAudioConverterSampleRateConverterQuality`

  - Minimum, low, medium, high, maximum
- Higher quality costs more CPU
- Linear is fast, but very low quality

Processing Audio

Signal Processing with AudioUnits

- AudioUnits
  - Components that process audio
  - Attributes
    - Inputs, outputs
    - Parameters that can be varied in real time

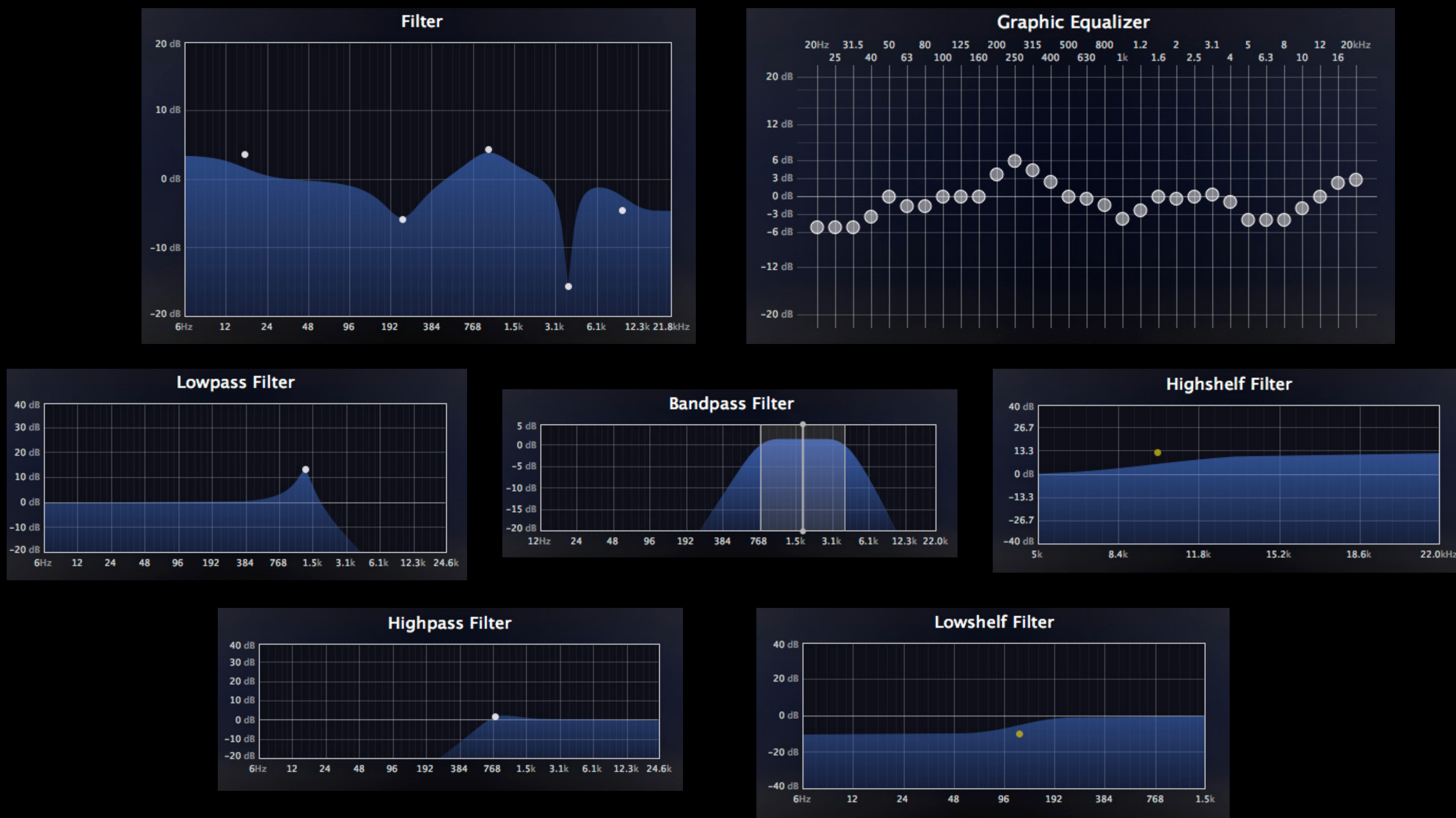
Signal Processing with AudioUnits

- I/O
- Effects
  - Filters
  - Compressors
  - Delays, reverbs
  - Time/pitch alterations
- Panners
- Mixing
  - 3D Mixer

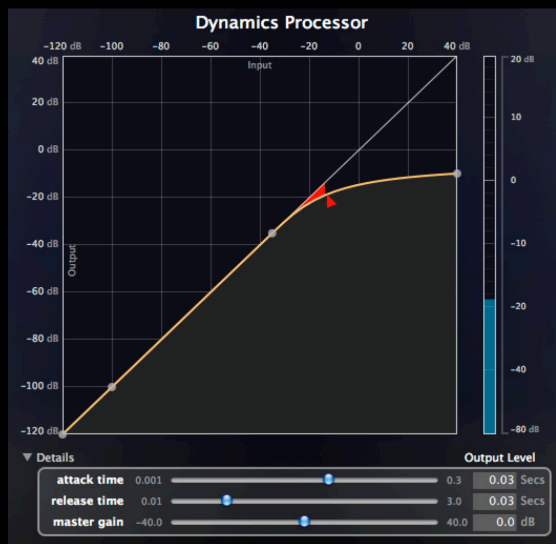
I/O AudioUnits

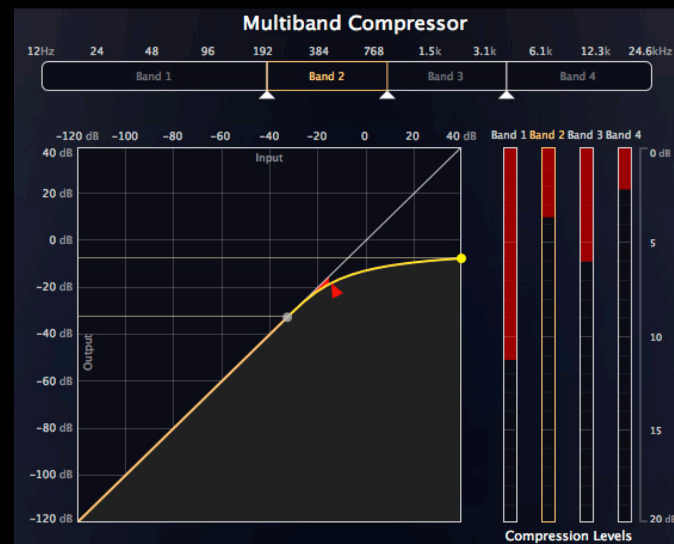
- Remote I/O—iPhone OS
- AUHAL—Mac OS X

Filters (Desktop)

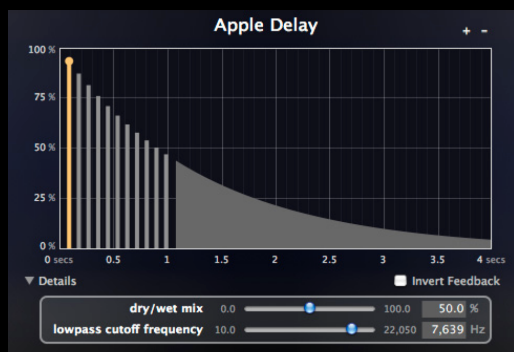


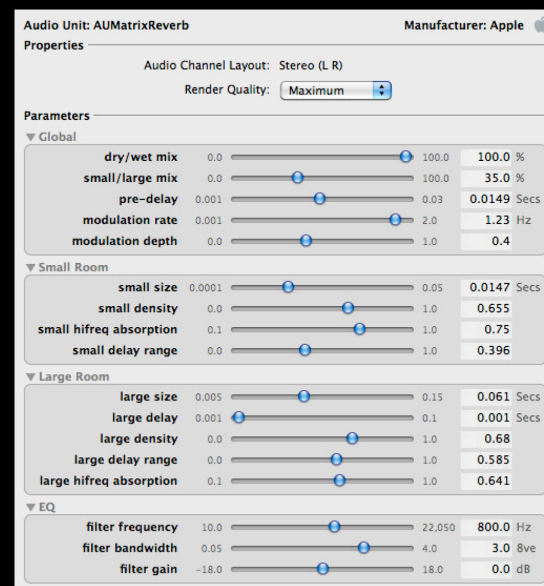
Compressors (Desktop)





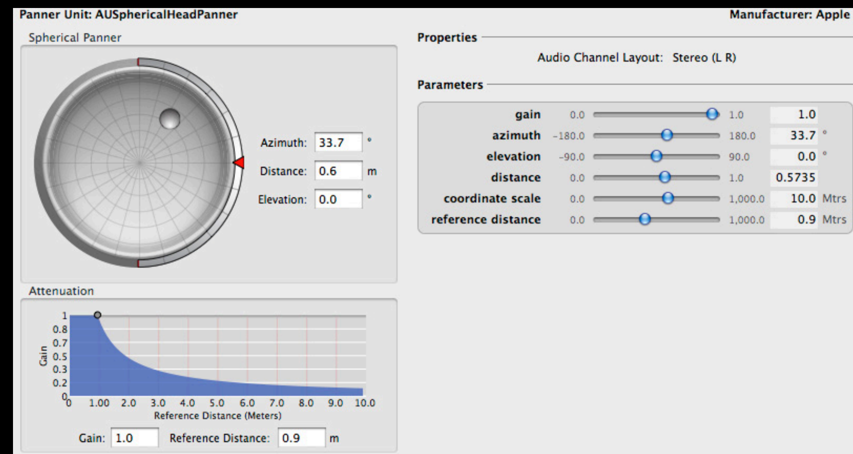
Delay, Reverb (Desktop)





Panners (Desktop)

- AUSoundFieldPanner
- AUSphericalHeadPanner
- AUVectorPanner
- HRTFPanner



Mixers

- iPhone
  - "Multichannel" Mixer
  - Embedded 3D Mixer
- Desktop
  - Multichannel Mixer
  - 3D Mixer
  - Stereo Mixer
  - Matrix Mixer

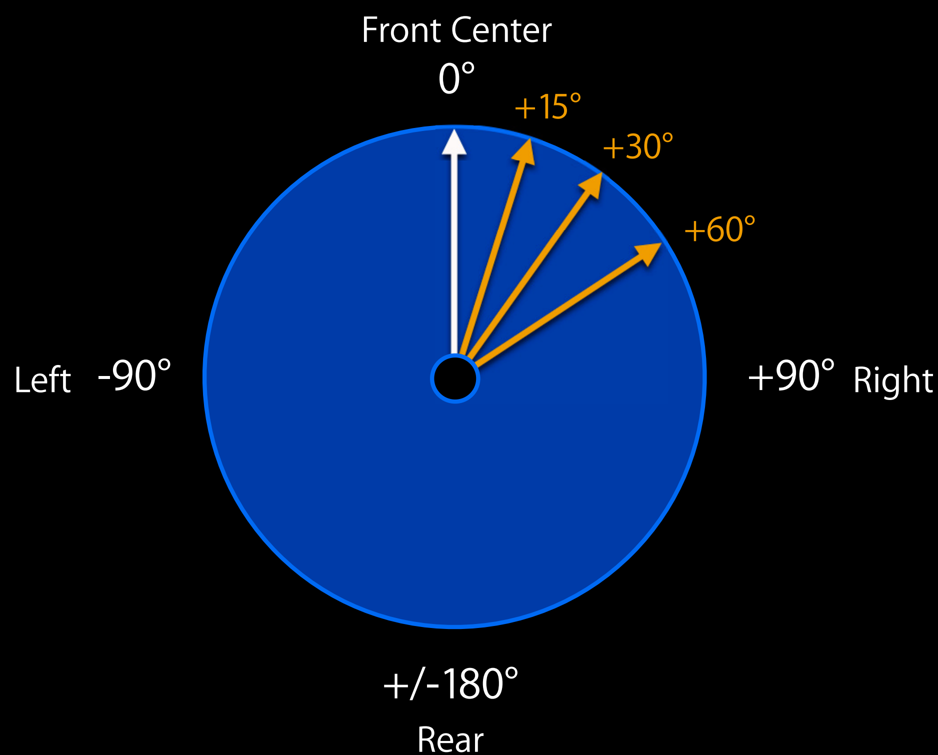
Embedded 3D Mixer

Spatialization modes

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

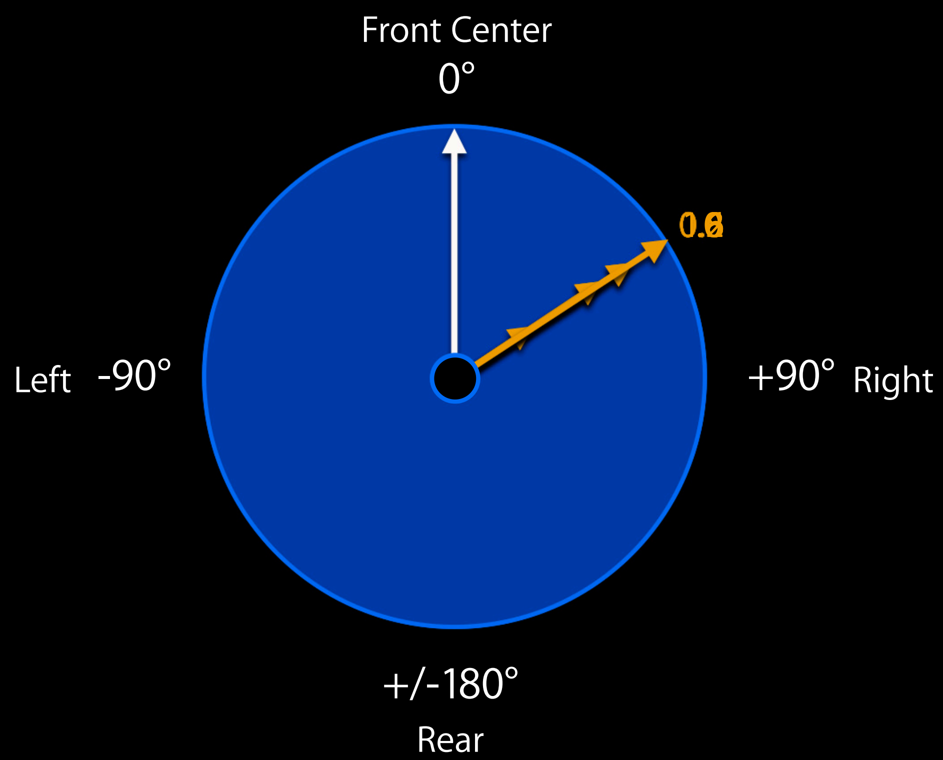
3D Mixer Parameters

Azimuth



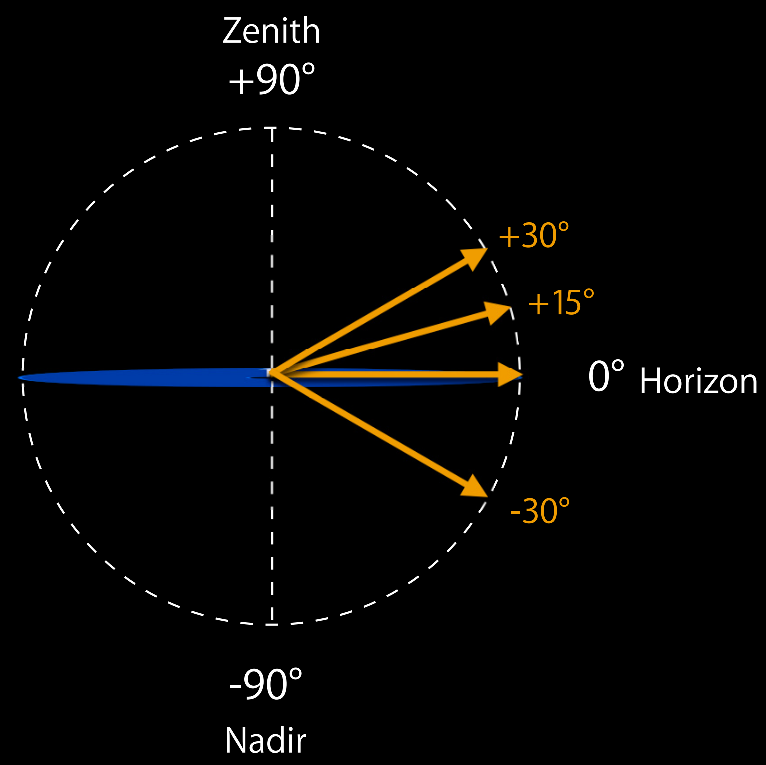
3D Mixer Parameters

Distance



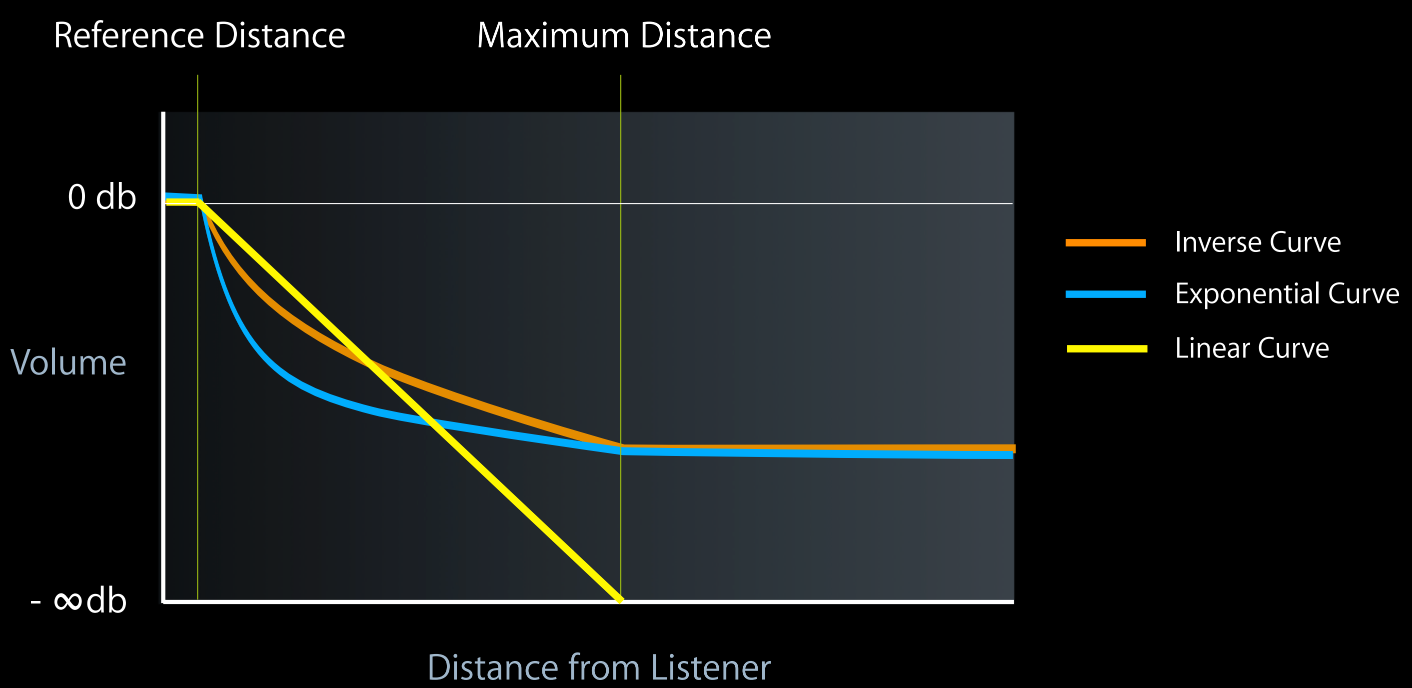
3D Mixer Parameters

Elevation



3D Mixer Property

Distance attenuation



OpenAL

- OpenGL—like library for 3D audio
- Cross-platform
- 3D spatialized source positioning
- Built on top of 3D Mixer
- Uses world coordinates (x, y, z)

Voice Processing Audio Unit and Audio Codecs

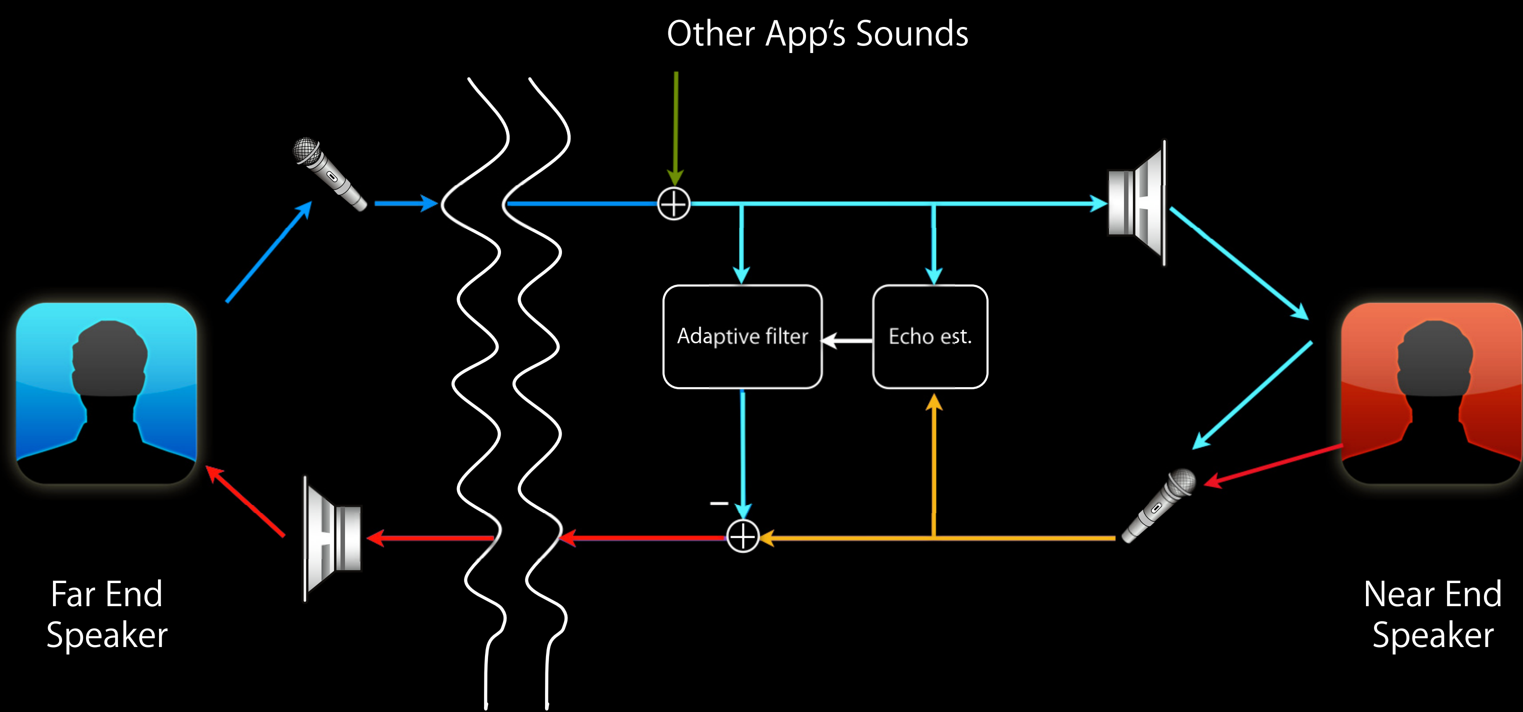
Eric Allamanche
Core Audio Engineer

Voice Processing Audio Unit

Voice Processing Audio Unit

- Dedicated AURemoteIO with built-in Acoustic Echo Canceler
- Instantiated and accessed the same way as the Output Audio Unit
- Significantly better quality, specially designed for high-quality chat
- Two algorithms available allowing quality versus complexity tradeoff

Functionality of an Acoustic Echo Canceler



Opening the Voice Processing Audio Unit

```
#include <AudioUnit/AudioUnit.h>

AudioUnit myIOUnit = NULL;
AudioComponentDescription desc;

desc.componentType = kAudioUnitType_Output;
desc.componentSubType = kAudioUnitSubType_VoiceProcessingIO;
desc.componentManufacturer = kAudioUnitManufacturer_Apple;
desc.componentFlags = 0;
desc.componentFlagsMask = 0;

AudioComponent comp = AudioComponentFindNext(NULL, &desc);

AudioComponentInstanceNew(comp, &myIOUnit);
```

Voice Processing Audio Unit Properties

- Defined in the `<AudioUnit/AudioUnitProperties.h>` header file

```
kAUVoiceIOProperty_BypassVoiceProcessing
kAUVoiceIOProperty_VoiceProcessingEnableAGC
kAUVoiceIOProperty_DuckNonVoiceAudio
```

Voice Processing Audio Unit Properties

New in iOS 4

- Defined in the `<AudioUnit/AudioUnitProperties.h>` header file

```
kAUVoiceIOProperty_BypassVoiceProcessing
kAUVoiceIOProperty_VoiceProcessingEnableAGC
kAUVoiceIOProperty_DuckNonVoiceAudio
kAUVoiceIOProperty_VoiceProcessingQuality
kAUVoiceIOProperty_MuteOutput
```

Audio Codecs

Audio Codecs

- CODEC = enCOder + DECoder
- Compresses and decompresses PCM audio signals
- Lossy and lossless codecs
- Core technology in digital audio nowadays

Lossy Versus Lossless Audio Codecs

- Lossless
  - No loss of information
  - After encoding and decoding resulting signal is bit identical to the original regardless of the bit depth
  - Typical compression factor: 1.5–2.0
- Lossy
  - Relies on a perceptual model of the human auditory system
  - Removes redundant and irrelevant (not perceived) information
  - Quality controlled by bit rate
  - Typical compression factor: 6–24

Available Audio Decoders

- ADPCM: IMA, IMA4, DVI, MS-ADPCM
- QDesign (version 1 and 2)
- GSM
- iLBC (Internet Low Bit rate Codec)
- MPEG-1/2 Layer 3 (MP3)
- Apple Lossless (ALAC)
- MPEG-4 AAC family

Available Audio Encoders

- ADPCM: IMA4
- iLBC (Internet Low Bit rate Codec)
- Apple Lossless (ALAC)
- MPEG-4 AAC
    - Low Complexity
    - Low Delay
    - Enhanced Low Delay
- Note: no MP3 encoder, provided through iTunes only

Specifications

Codec	Sample Rates	Bit Rates	Channels	Optimized For
iLBC	8 kHz	13.3, 15.2 kbps	1	Speech
MP3	16-48 kHz	8-320 kbps	1-2	General audio
ALAC	Any	not settable	1-2	General audio
AAC Low Complexity	8-48 kHz	8-768 kbps	1-2, 5.1*, 7.1*	General audio
High Efficiency AAC	16-48kHz	12-96 kbps	1-2	Streaming audio
AAC (Enhanced) Low Delay	16-48 kHz	16-256 kbps	1-2	AV chat

** Down mixed to stereo*

AAC Codecs

Why Advanced Audio Codec (AAC)?

- About MP3
  - Is almost 20 years old! (standardized in 1991)
  - Has limited bit rates, sample rates, channel configurations
  - Can't ensure transparent coding for certain signal classes
- AAC was designed to address these issues and allow transparent coding (standardized in MPEG-2 in 1997)
- Many enhancements and variations added since its first standardization (MPEG-4)

AAC Codecs

Low Complexity

High Efficiency
(v2)

Low Delay

AAC Codecs

Low Complexity

High Efficiency
(v2)

Enhanced
Low Delay

Low Complexity and High Efficiency

- Low Complexity (`'kAudioFormatMPEG4AAC'`)
  - Highest audio quality, multichannel support
- High Efficiency (`'kAudioFormatMPEG4AAC_HE'`)
  - Synthesizes upper frequency bands rather than encoding them
- High Efficiency v2 (`'kAudioFormatMPEG4AAC_HE_V2'`)
  - Expands mono signal to stereo using time-frequency parameters (Parametric Stereo)
  - Very low bit rate down to 20 kbps

Highest Quality

Lowest Bit Rate

AAC Sound Examples

AAC Sound Examples

- Low Complexity @ 128 kbps

AAC Sound Examples

- Low Complexity @ 128 kbps
- High Efficiency @ 64 kbps

AAC Sound Examples

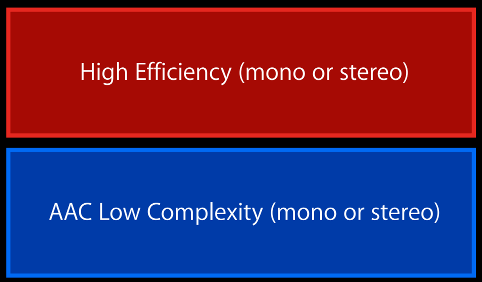
- Low Complexity @ 128 kbps
- High Efficiency @ 64 kbps
- High Efficiency v2 @ 32 kbps

AAC Sound Examples

- Low Complexity @ 128 kbps
- High Efficiency @ 64 kbps
- High Efficiency v2 @ 32 kbps
- Low Complexity @ 32 kbps

Low Complexity and High Efficiency

Adding layers



`kAudioFormatMPEG4AAC_HE`
(44.1 kHz, mono or stereo)

`kAudioFormatMPEG4AAC`: most compatible format (22.05 kHz, mono or stereo)

TN2236

Low Complexity and High Efficiency

Adding layers

High Efficiency v2 (stereo)

High Efficiency (mono)

AAC Low Complexity (mono)

`kAudioFormatMPEG4AAC_HE_V2`: richest format
(44.1 kHz, stereo)

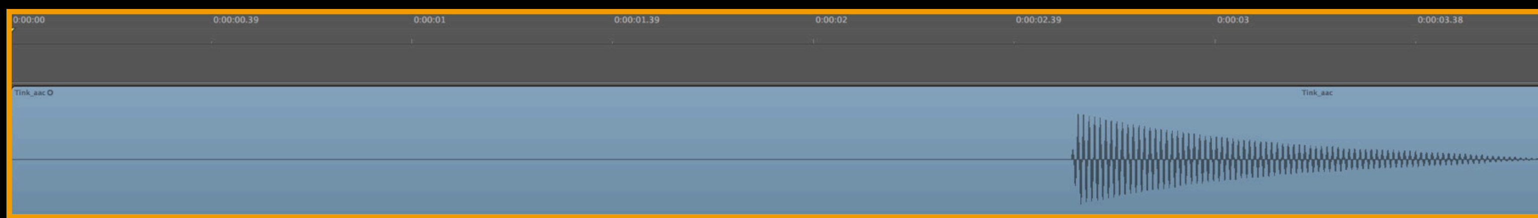
`kAudioFormatMPEG4AAC_HE`
(44.1 kHz, mono)

`kAudioFormatMPEG4AAC`: most compatible format
(22.05 kHz, mono)

TN2236

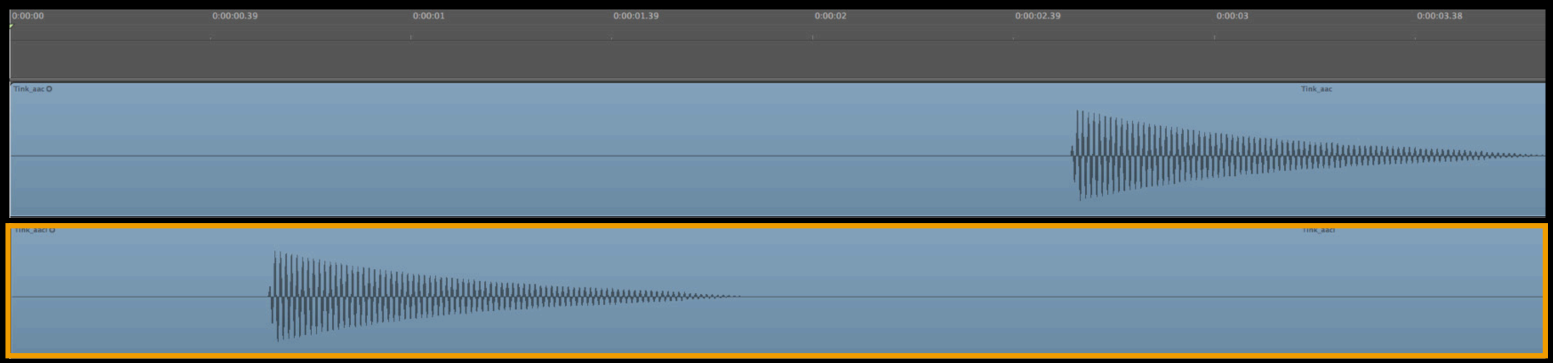
AAC Low Delay Codecs

Low Complexity



AAC Low Delay Codecs

Low Delay



AAC Low Delay Codecs

Enhanced Low Delay

AAC Low Delay Codecs

AAC Low Delay Codecs

- Use the same foundations as AAC but with much smaller delay (typically 15-40ms), designed for high-quality full duplex communication
- Low Delay (`‘kAudioFormatMPEG4AAC_LD‘`)
  - Minimum delay of 20ms
- Enhanced Low Delay (`‘kAudioFormatMPEG4AAC_ELD’`)
  - Minimum delay of 15ms
  - Optional high-efficiency extension for increased bandwidth
- Large bit rate range allows for transparent quality
- Delay is proportional to sample rate

TN2258

Hardware/Software Decoder Availability

Codec	Software	Hardware
MP3	✓	✓
ALAC	✓	✓
AAC Low Complexity	✓	✓
AAC High Efficiency	✗	✓
AAC High Efficiency v2	✗	✓
AAC (Enhanced) Low Delay	✓	✗

Key Parameters

- Sampling rate
- Number of channels
- Bit rate
- Bit rate mode
- Subjective quality

Bit Rate

- Determines the compression ratio
- Typically accounts for all channels
- Grows with the number of channels and sample rate
- AAC encoder will lower the output sample rate if the provided bit rate is deemed insufficient for the sample rate/number of channels combination

Bit Rate Modes

- Constant Bit Rate (CBR)
  - Simplest mode where a fixed number of bytes is allocated for each packet
  - Does not accommodate to the content, like silence or transients
- Average Bit Rate (ABR)
  - Allocates bit resources dynamically according to the content while maintaining an average target bit rate
- Variable Bit Rate (VBR)
  - Most flexible mode where the goal is to maintain a constant quality
  - Uses a quality value [0,127] instead of bit rate

Recommendations

- Choose the codec according to the use case and its limitations
  - High-quality audio: Low Complexity
  - Streaming on constrained channels: High Efficiency
  - High-quality voice chats: Enhanced Low Delay
- When possible, favor the highest possible quality
  - Best codec
  - Best encoding mode
  - Highest bit rate

Recommendations

- Avoid transcoding; e.g., converting MP3 to AAC, even if the bit rate is higher
  - Lost information can't be recovered!

Container and Streaming Formats for AAC

- .mp4: MPEG-4 native file format
- .m4a: MPEG-4 compatible, adds iTunes-specific data chunks, also used for ALAC
- .caf: preferred format
- .adts/.aac: self-framing format, used in SHOUTcast and HTTP live streaming

Related Sessions

Audio Development for iPhone OS, part 1	Mission Wednesday 9:00AM
Audio Development for iPhone OS, part 2	Mission Wednesday 11:30AM

Labs

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

WWDC10

[redacted]