Audio Development for iPhone OS

Part 2: Audio unit input, output, and mixing

Murray Jason
Apple Developer Publications

Audio Unit Input, Output, and Mixing

1 Audio units in context

2 Audio architecture

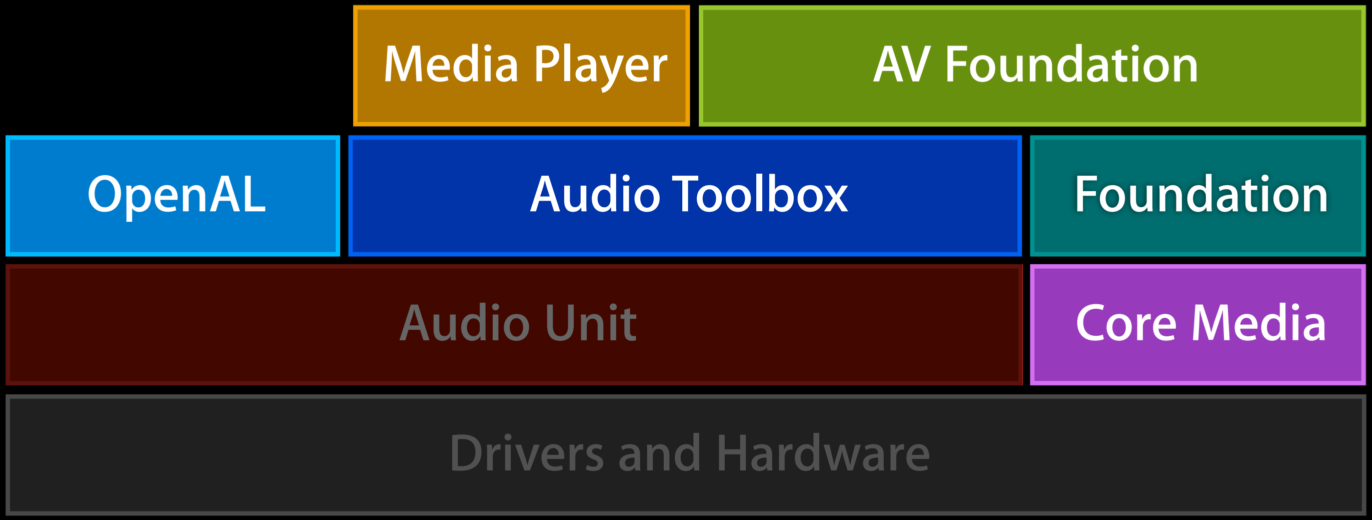
3 Let’s build an audio unit app

Audio Unit Primer

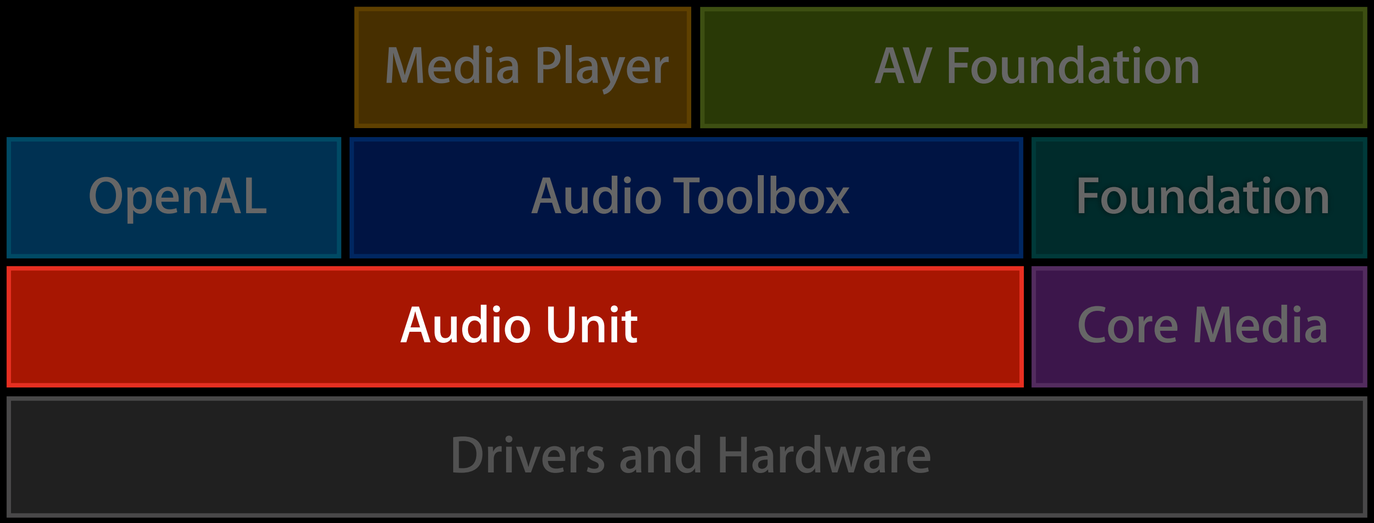
Audio units…

- Are the iPhone OS audio processing plug-in architecture
- Provide a flexible processing-chain facility
- Support real-time input, output, or simultaneous I/O
- Demand an informed approach

Audio Units in Context



Audio Units in Context



Media Player

AV Foundation

OpenAL

Audio Toolbox

Foundation

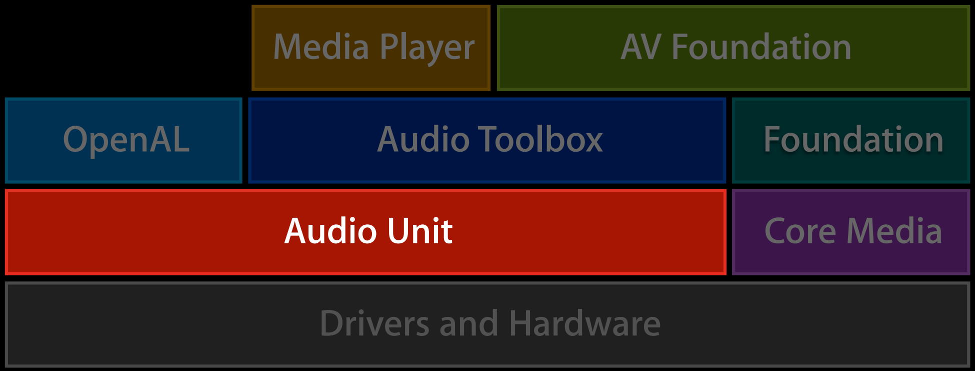
Audio Unit

Core Media

Drivers and Hardware

Audio Units Address Very Specific Needs

- Simultaneous I/O with low latency
- Responsive playback of synthesized sounds
- Use of a built-in feature: echo cancellation, mixing, panning…



Media Player

AV Foundation

OpenAL

Audio Toolbox

Foundation

Audio Unit

Core Media

Drivers and Hardware

Seven Audio Units in iPhone OS

Effect units	iPod Equalizer
Mixer units	3D Mixer Multichannel Mixer
I/O units	Remote I/O Voice-Processing I/O Generic Output
Format Converter units	Format Converter

Where to Use Audio Units

- In a VoIP app: Use the Voice Processing I/O unit
- In an interactive music app: Use a mixer unit
- For real-time audio I/O processing: Use the Remote I/O unit

Audio Unit Host Application Architecture

Pieces of the puzzle and how they fit together

Audio Unit Host Application Architecture

What you’ll see in this section

- Demo—I/O unit “hello world”
- App audio design at the black box level
- Inside the box
    - Functional pieces
    - API pieces

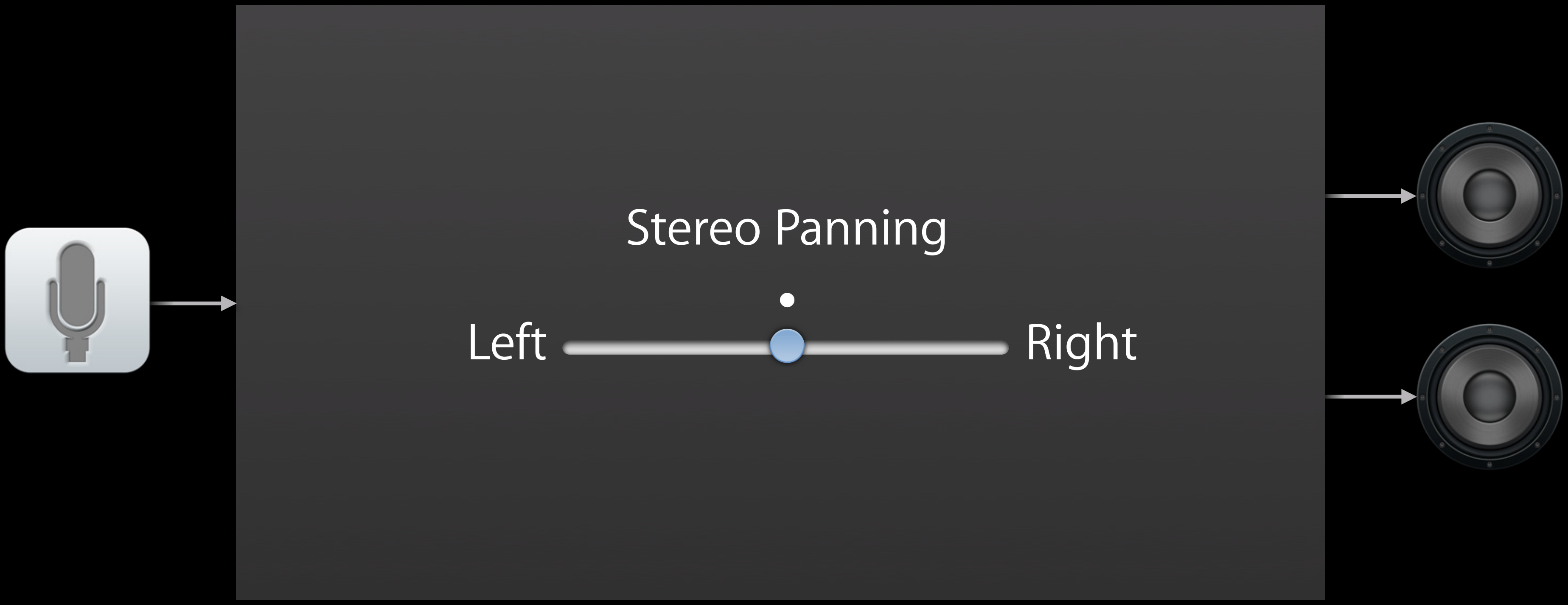
Demo

IOHost Sample Code

William Stewart
Core Audio Engineering

Audio Unit Host App Architecture

Black box design



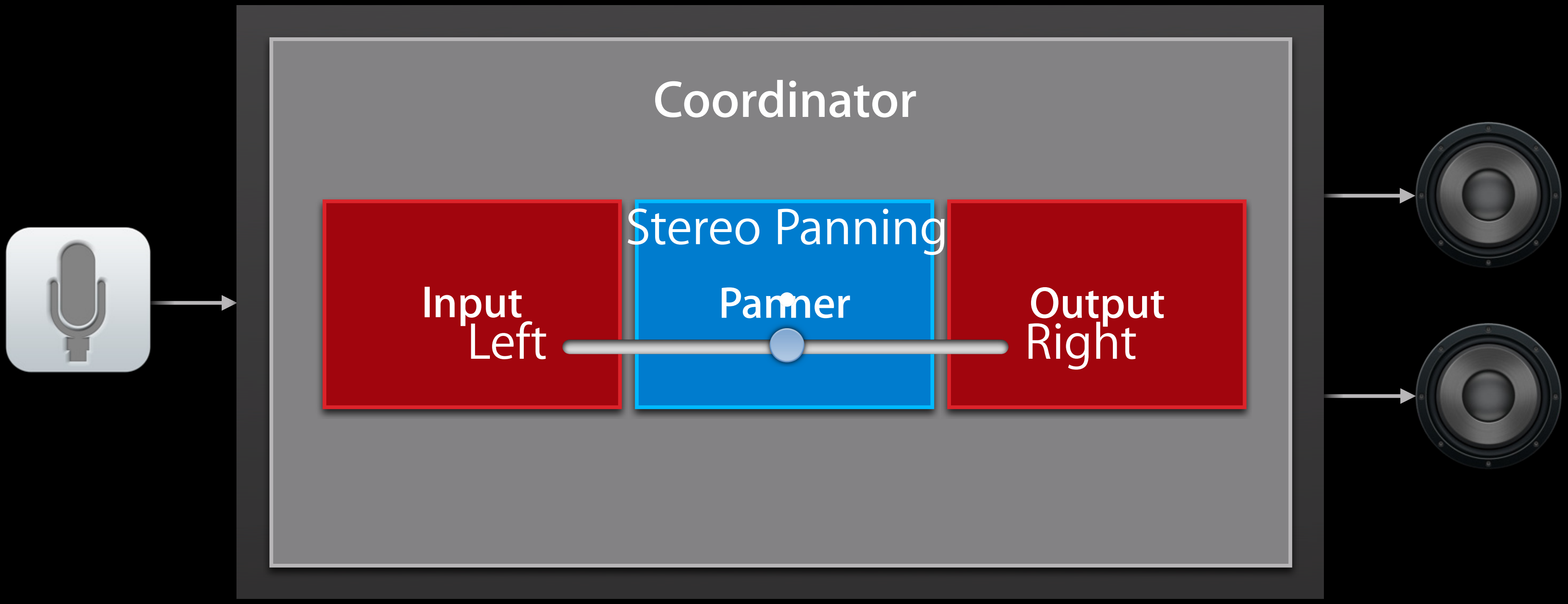
Stereo Panning

Left

Right

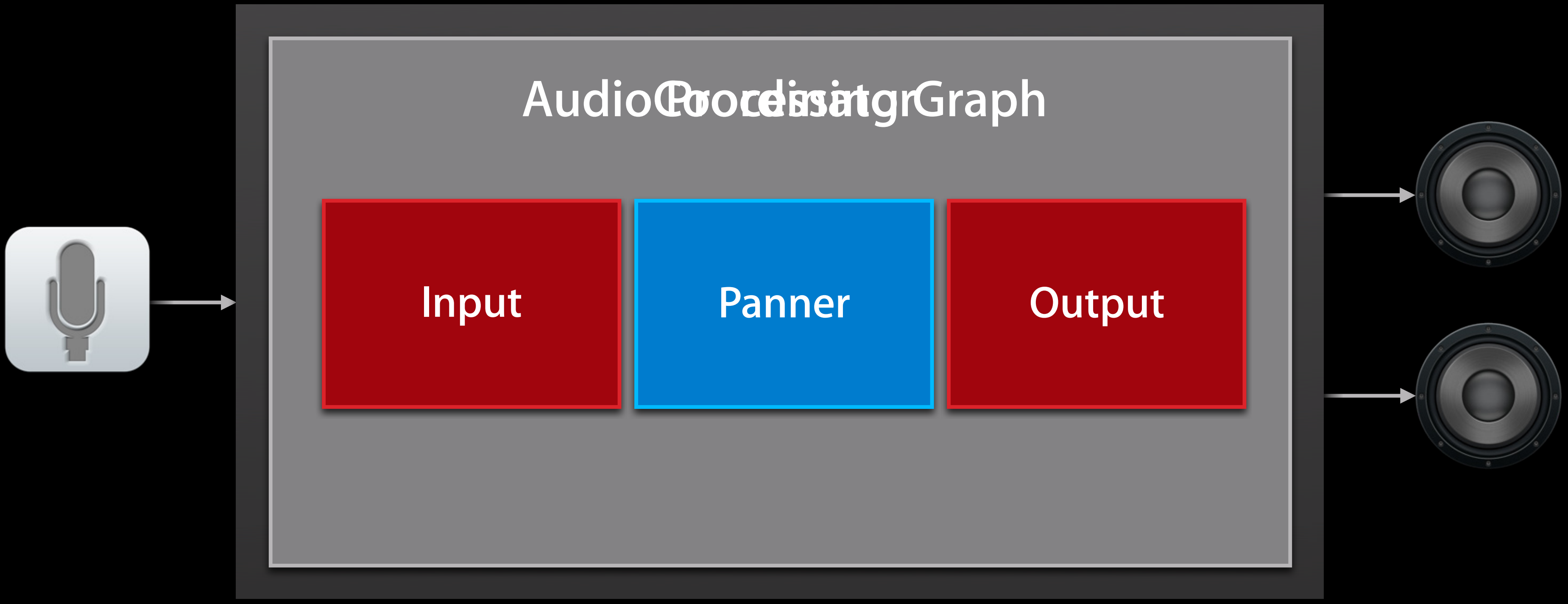
Audio Unit Host App Architecture

Functional representation



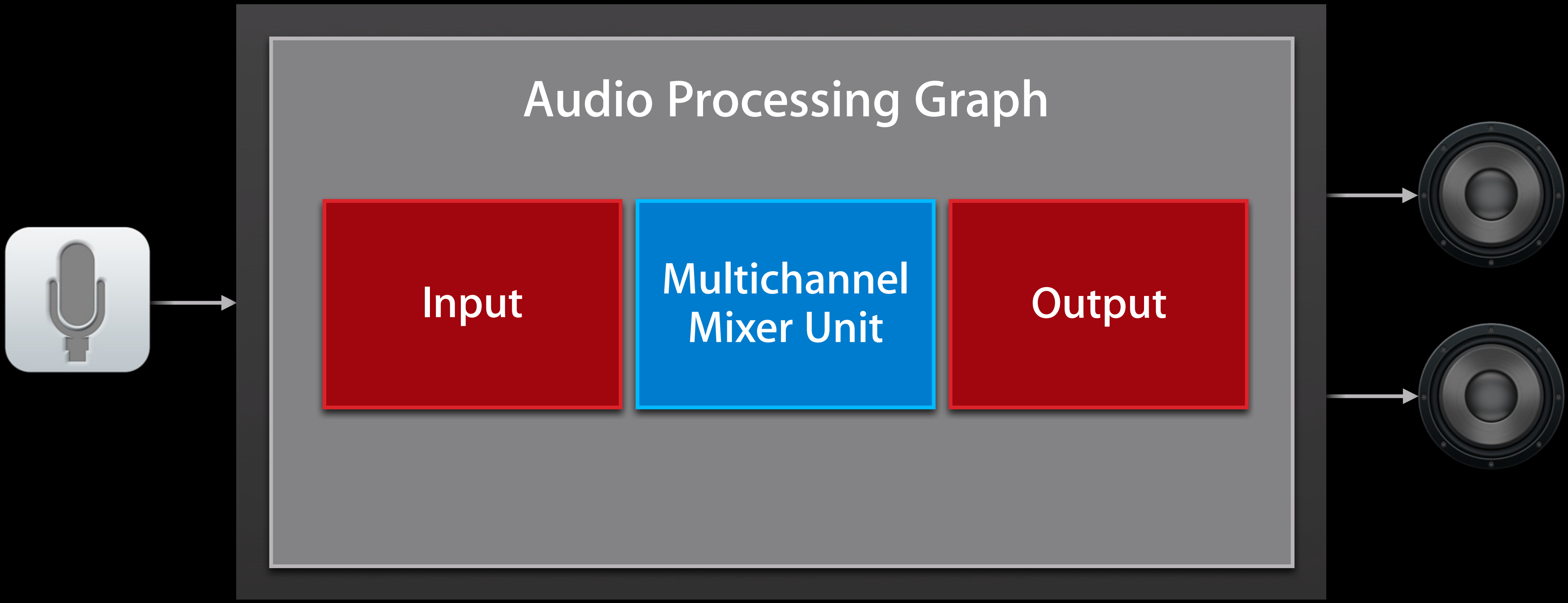
Audio Unit Host App Architecture

API pieces



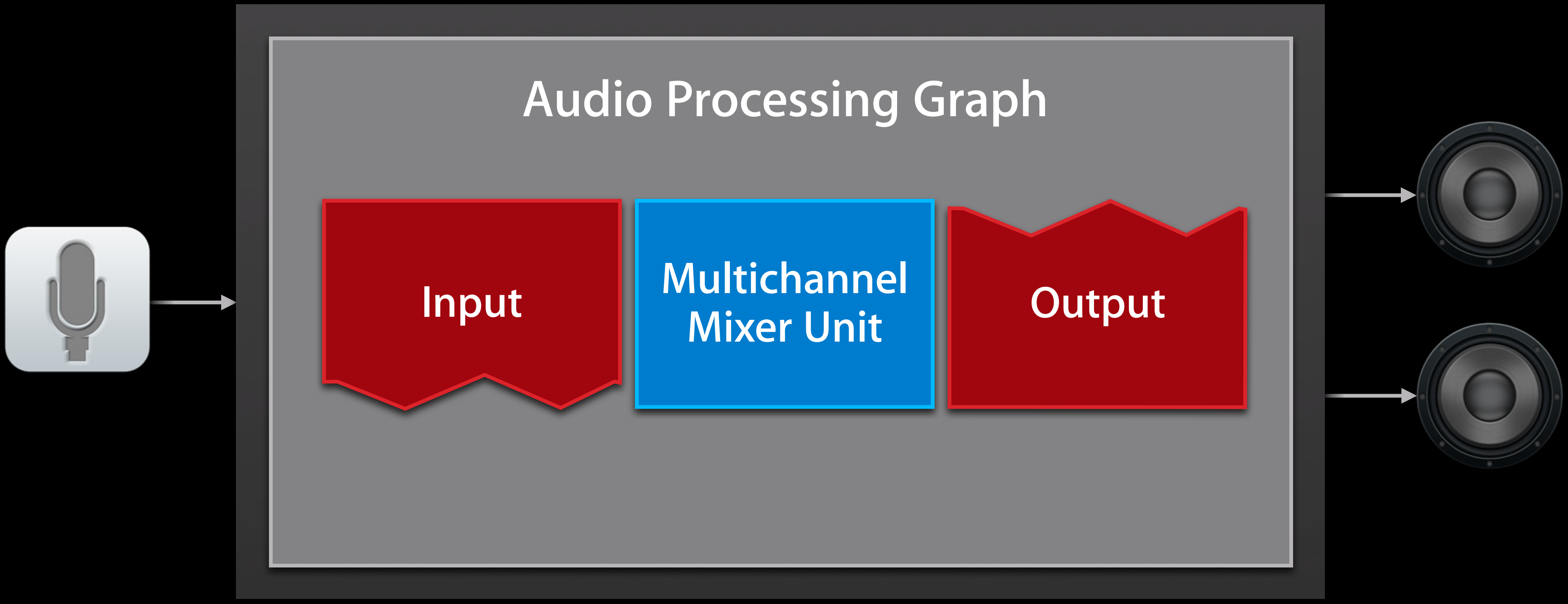
What About Input and Output?

API pieces



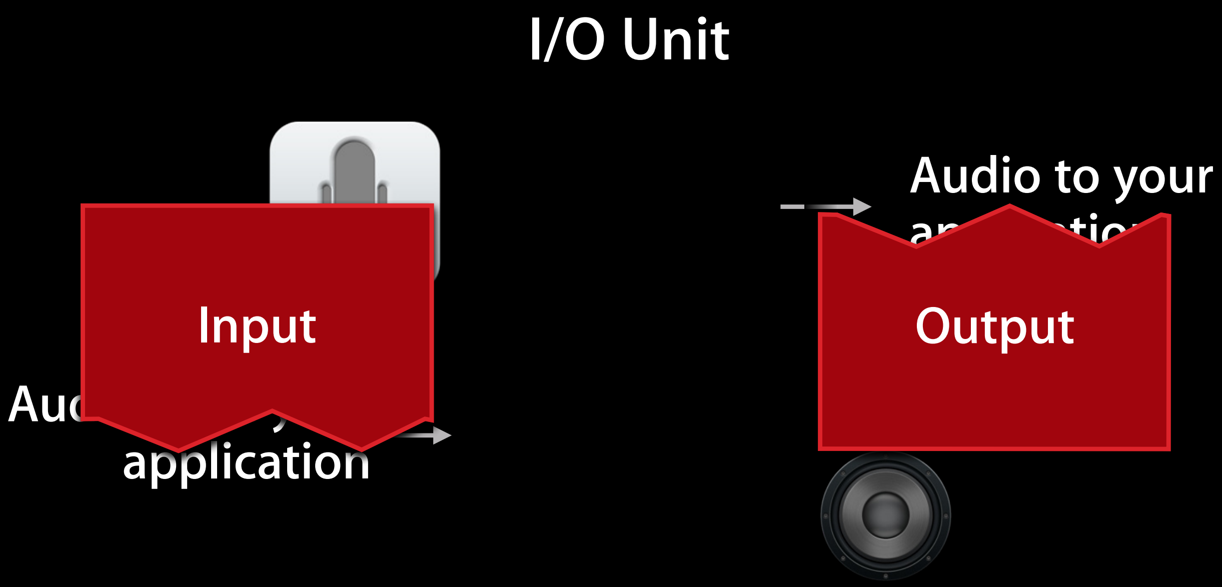
Input and Output: Two Parts of One Object

API pieces



Input and Output: Two Parts of One Object

API pieces

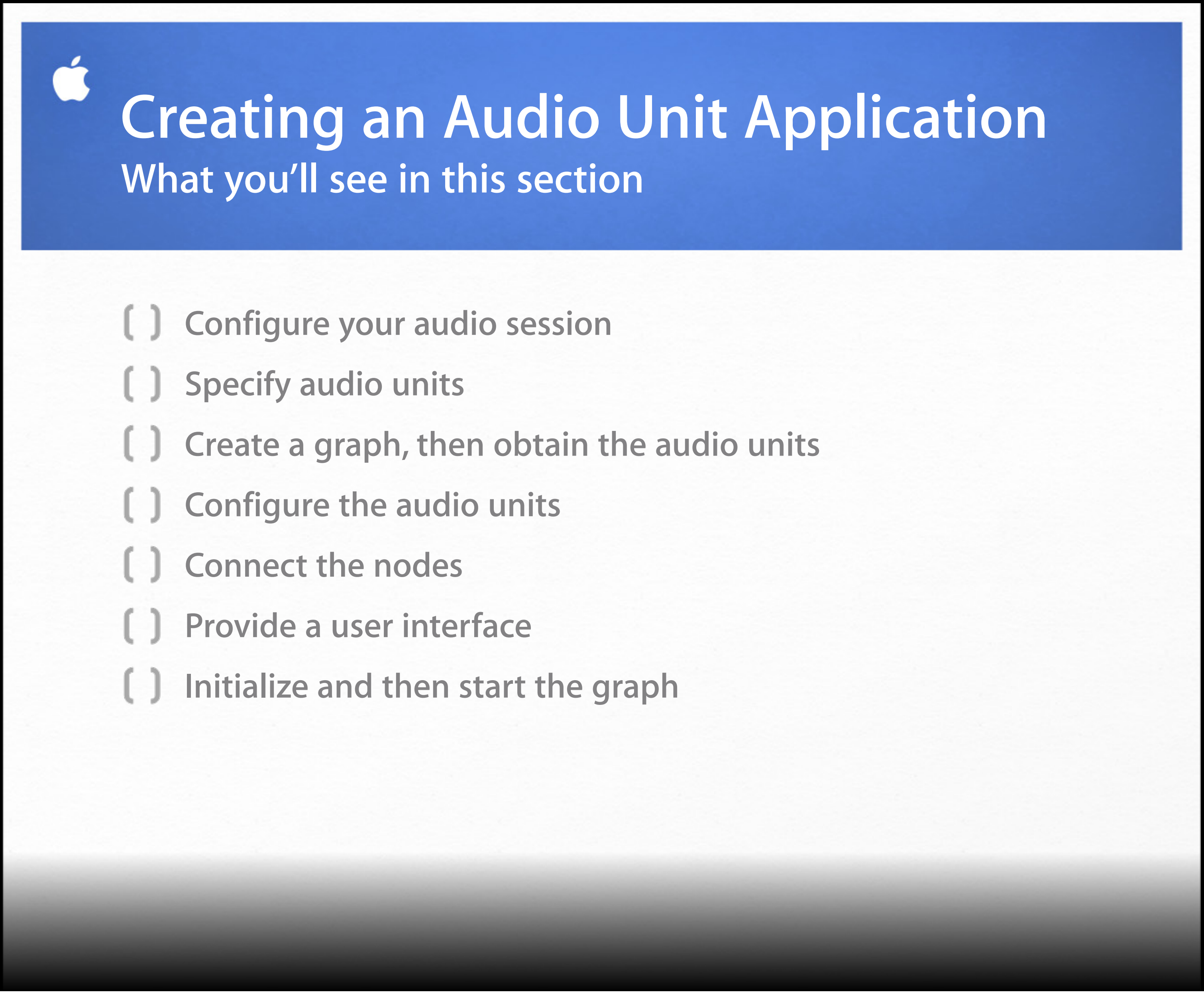


Some Definitions Before We Move On

Audio unit	An audio processing plug-in component that you find at run time
Audio unit node	A representation of an audio unit in the context of an audio processing graph
Audio processing graph	An object that manages a network of audio unit nodes

Creating an Audio Unit Application

Accessing and connecting audio units



Creating an Audio Unit Application

What you'll see in this section

- ( ) Configure your audio session
- ( ) Specify audio units
- ( ) Create a graph, then obtain the audio units
- ( ) Configure the audio units
- ( ) Connect the nodes
- ( ) Provide a user interface
- ( ) Initialize and then start the graph

Configure Your Audio Session

Notably, obtain the hardware sample rate

```
self.graphSampleRate = 44100.0; // Hertz
```

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

```
[mySession setPreferredHardwareSampleRate: graphSampleRate error: nil];
```

```
[mySession setCategory: AVAudioSessionCategoryPlayAndRecord error: nil];
```

```
[mySession setActive: YES error: nil];
```

```
self.graphSampleRate = [mySession currentHardwareSampleRate];
```

Specify Audio Units

```
AudioComponentDescription ioUnitDesc;
```

```
ioUnitDesc.componentType         = kAudioUnitType_Output;
ioUnitDesc.componentSubType      = kAudioUnitSubType_RemoteIO;
ioUnitDesc.componentManufacturer = kAudioUnitManufacturer_Apple;
ioUnitDesc.componentFlags        = 0;
ioUnitDesc.componentFlagsMask    = 0;
```

```
AudioComponentDescription mixerDesc;
```

```
mixerDesc.componentType         = kAudioUnitType_Mixer;
mixerDesc.componentSubType      = kAudioUnitSubType_MultiChannelMixer;
mixerDesc.componentManufacturer = kAudioUnitManufacturer_Apple;
mixerDesc.componentFlags        = 0;
mixerDesc.componentFlagsMask    = 0;
```

Create a Graph

```
AUGraph processingGraph;
NewAUGraph (&processingGraph);
```

```
AUNode ioNode;
AUNode mixerNode;
```

```
AUGraphAddNode (processingGraph, &ioUnitDesc, &ioNode);
AUGraphAddNode (processingGraph, &mixerDesc, &mixerNode);
```

Instantiate and Obtain Audio Units

```
AUGraphOpen (processingGraph);  // performs audio unit instantiation
```

```
AudioUnit ioUnit;

AudioUnit mixerUnit;
```

```
AUGraphNodeInfo (processingGraph, ioNode, NULL, &ioUnit);

AUGraphNodeInfo (processingGraph, mixerNode, NULL, &mixerUnit);
```

Creating an Audio Unit Application

Ready to configure the audio units

[✓] Configure your audio session

[✓] Specify audio units

[✓] Create a graph, then obtain the audio units

[ ] Configure the audio units

[ ] Connect the nodes

[ ] Provide a user interface

[ ] Initialize and then start the graph

Audio Unit Property Primer

Overview

- Properties are key-value pairs
- Typically, properties do not change over time
  - Audio stream format
  - Connections
  - Number of input buses on a mixer
- In general, you set properties only when an audio unit is uninitialized

Audio Unit Property Primer

Definitions

Property key	A unique constant
Property value	A designated type with particular read/write access and target scope(s)

Audio Unit Property Primer

Documentation example

`kAudioOutputUnitProperty_SetInputCallback`

Specifies the input callback and processing context for an I/O unit

A read/write `AURenderCallbackStruct` data structure valid on the audio unit global scope

See Audio Unit Properties Reference

Stream Formats

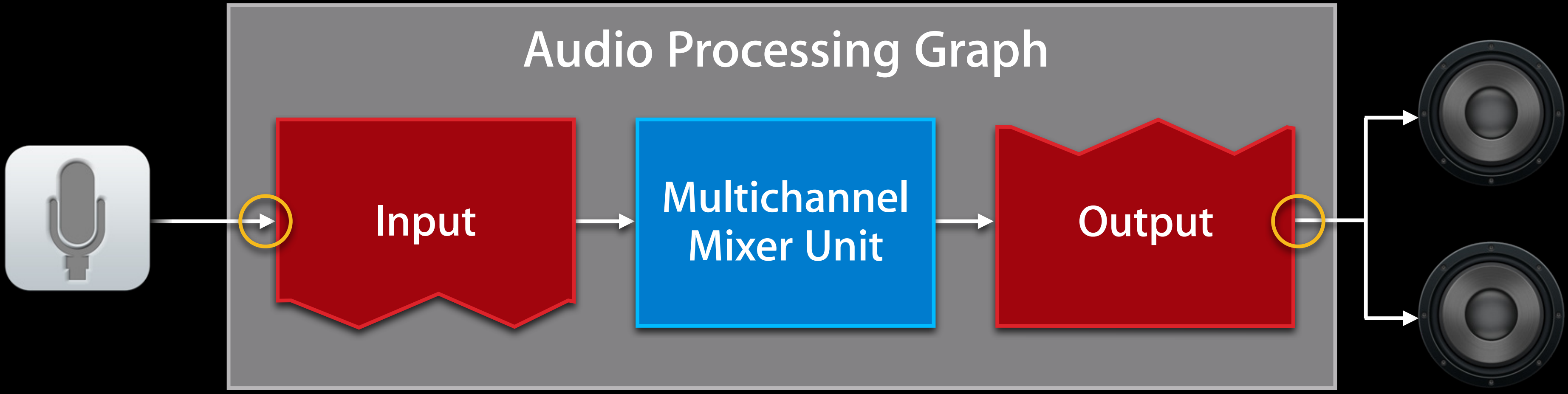
Stream Formats Primer

Overview

- Hardware imposes its stream formats
- You specify stream format(s) for the graph
- I/O units perform conversion
- Use the struct: `AudioStreamBasicDescription` (a.k.a. ASBD)
    - Refer to Core Audio Data Types Reference
    - View our sample code
    - Study: `/Developer/Extras/CoreAudio/PublicUtility/`
      `CAStreamBasicDescription.h`

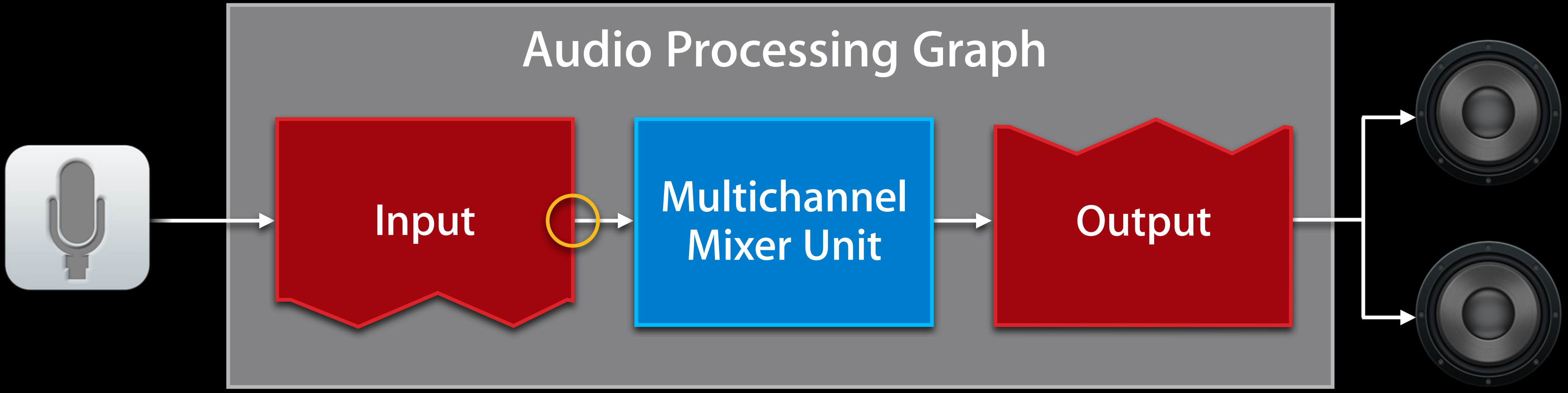
Stream Formats Primer

Hardware imposes its stream formats



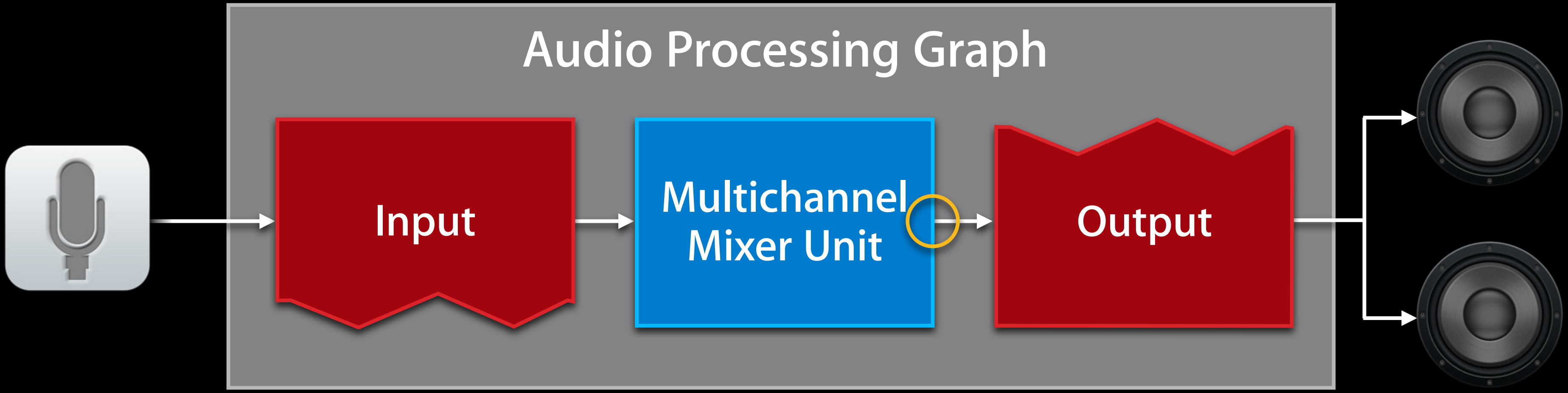
Stream Formats Primer

Set the application stream format on input



Stream Formats Primer

Set the output stream format where needed



Stream Formats Primer

Linear PCM, mono, noninterleaved, at hardware sample rate

```
int bytesPerSample = sizeof (AudioUnitSampleType);
AudioStreamBasicDescription inputStreamFormat = {0};
```

```
inputStreamFormat.mFormatID         = kAudioFormatLinearPCM;
inputStreamFormat.mFormatFlags      = kAudioFormatFlagsAudioUnitCanonical;
inputStreamFormat.mBytesPerPacket   = bytesPerSample;
inputStreamFormat.mBytesPerFrame    = bytesPerSample;
inputStreamFormat.mFramesPerPacket  = 1;
inputStreamFormat.mBitsPerChannel   = 8 * bytesPerSample;
inputStreamFormat.mChannelsPerFrame = 1;
inputStreamFormat.mSampleRate       = graphSampleRate;
```

Configure the I/O Unit

Set the application stream format

```
AudioUnitElement ioUnitInputElement = 1;
```

```
AudioUnitSetProperty (
    ioUnit,
    kAudioUnitProperty_StreamFormat,
    kAudioUnitScope_Output,
    ioUnitInputElement,
    &inputStreamFormat,
    sizeof (inputStreamFormat)
);
```

Configure the I/O Unit

Enable input

```
// From previous slide
// AudioUnitElement ioUnitInputElement = 1;
```

```
UInt32 enableInput = 1;
```

```
AudioUnitSetProperty (
    ioUnit,
    kAudioOutputUnitProperty_EnableIO,
    kAudioUnitScope_Input,
    ioUnitInputElement,
    &enableInput,
    sizeof (enableInput)
);
```

Configure the Multichannel Mixer Unit

Set the input bus count

```
UInt32 inputBusCount = 1;
```

```
AudioUnitSetProperty (
    mixerUnit,
    kAudioUnitProperty_ElementCount,
    kAudioUnitScope_Input,
    0, // always use 0 here
    &inputBusCount,
    sizeof (inputBusCount)
);
```

Configure the Multichannel Mixer Unit

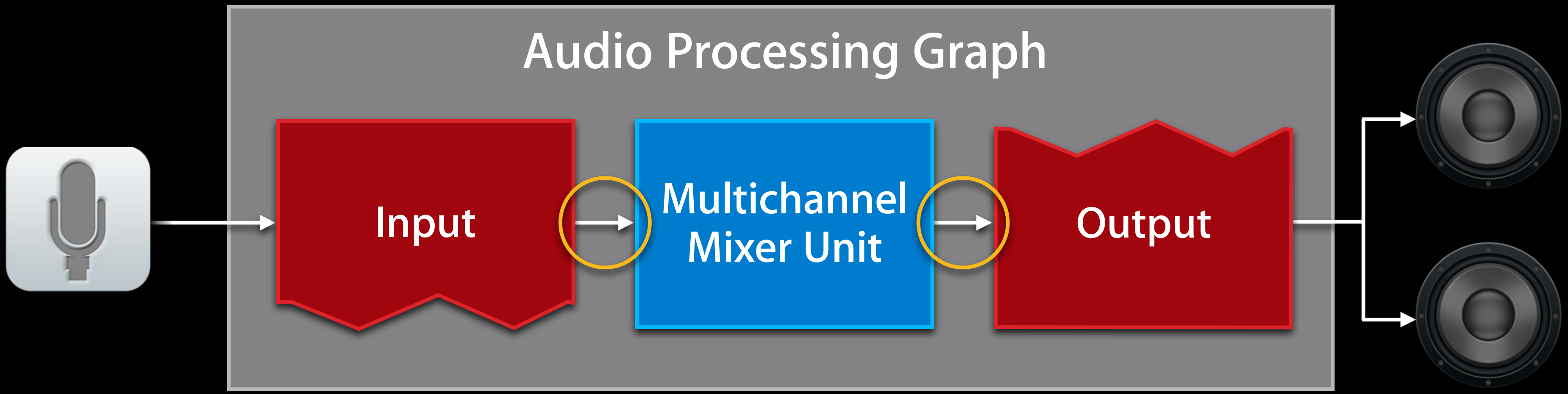
Set the output stream sample rate

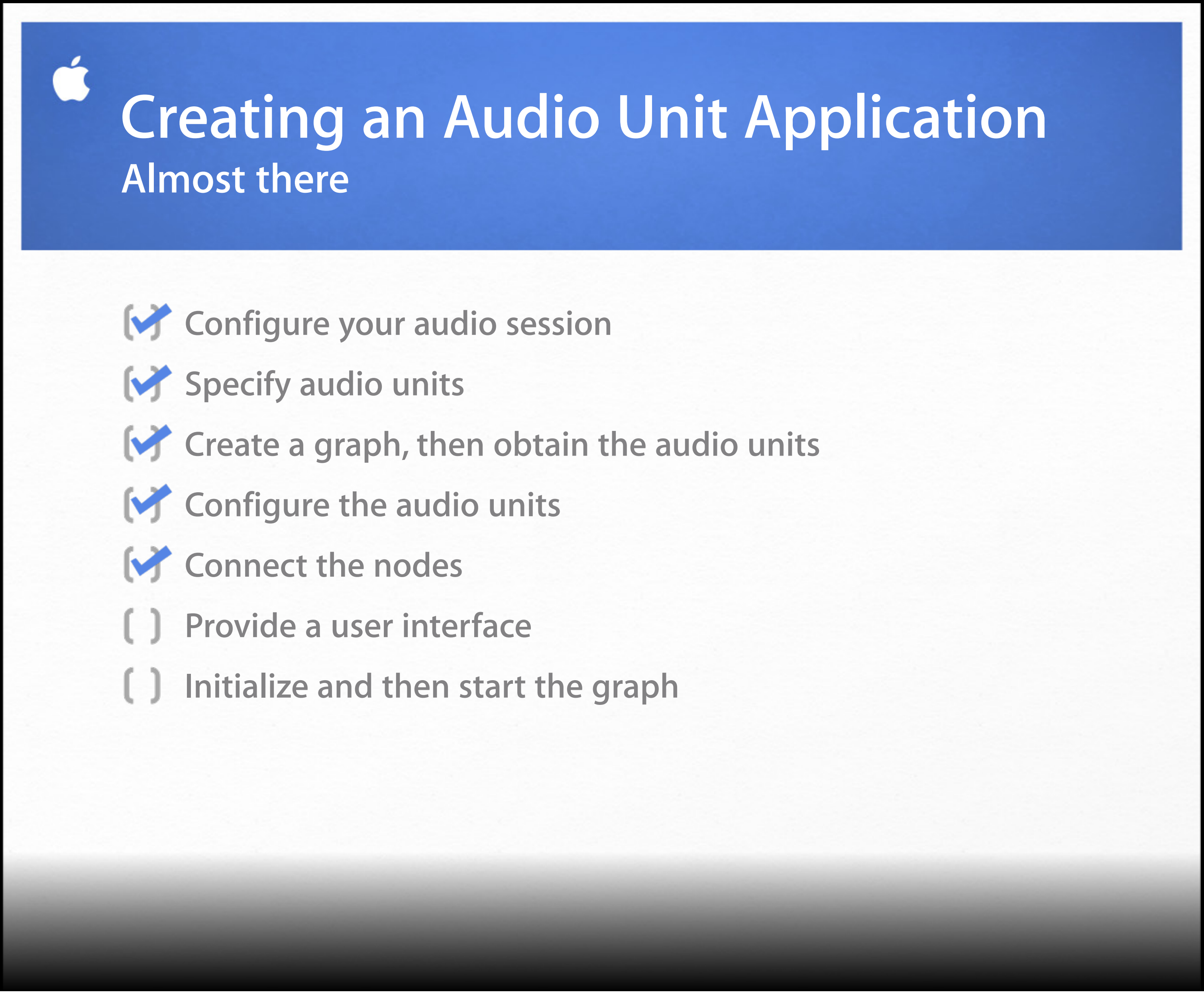
```
AudioUnitSetProperty (
    mixerUnit,
    kAudioUnitProperty_SampleRate,
    kAudioUnitScope_Output,
    0,  // there’s only one output bus on this audio unit
    &graphSampleRate,
    sizeof (graphSampleRate)
);
```

Connect the Audio Unit Nodes

```
// Connect output side of I/O unit input element to mixer input
AUGraphConnectNodeInput (processingGraph, ioNode, 1, mixerNode, 0);

// Connect mixer output to input side of I/O unit output element
AUGraphConnectNodeInput (processingGraph, mixerNode, 0, ioNode, 0);
```





Creating an Audio Unit Application

Almost there

- ☑ Configure your audio session
- ☑ Specify audio units
- ☑ Create a graph, then obtain the audio units
- ☑ Configure the audio units
- ☑ Connect the nodes
- □ Provide a user interface
- □ Initialize and then start the graph

Audio Unit Parameter Primer

Overview

- Parameters, like properties, are key-value pairs
- They’re intended to be varied during processing
  - Volume
  - Muting
  - Stereo panning position
- In general, users control parameters through a UI

Audio Unit Parameter Primer

Definitions

Parameter key	An identifier defined by an audio unit
Parameter value	32-bit floating point The audio unit defines meaning and permissible range

Audio Unit Parameter Primer

Documentation example

`kMultiChannelMixerParam_Pan`

Sets the stereo panning position for a mixer input.

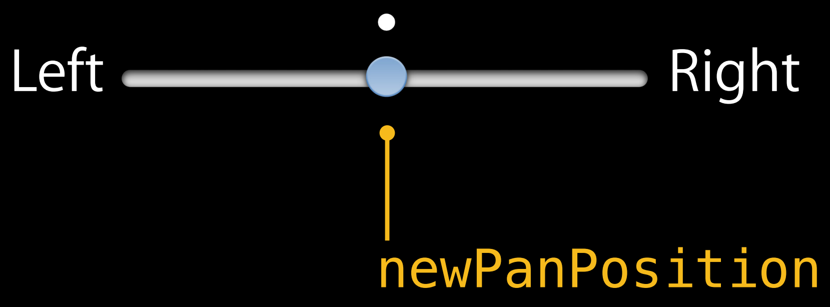
Range is –1 through +1. Default value is 0.

See Audio Unit Parameters Reference

Create a User Interface

- Use a `UISlider` object
- Use the `kMultiChannelMixerParam_Pan` parameter
- New in iOS 4

Stereo Panning



Left

Right

`newPanPosition`

Control the Stereo Panning Position

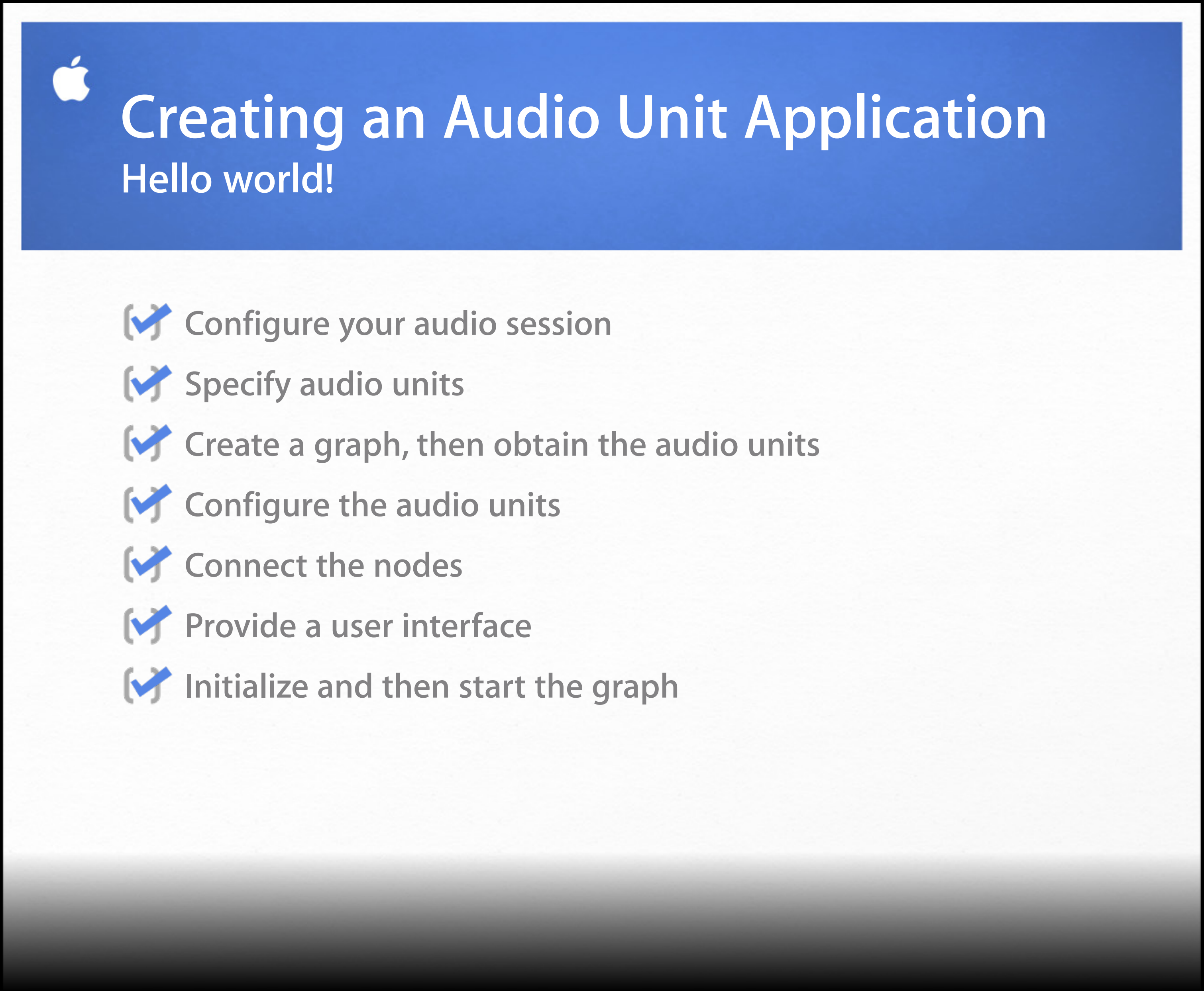
```
AudioUnitSetParameter (
    mixerUnit,
    kMultiChannelMixerParam_Pan,
    kAudioUnitScope_Input,
    0,  // bus number
    newPanPosition,
    0
);
```

Initialize and Start the Graph

```
AUGraphInitialize (processingGraph);
```

```
AUGraphStart (processingGraph);
```

```
// Some time later
AUGraphStop (processingGraph);
```



Creating an Audio Unit Application

Hello world!

- ☑ Configure your audio session
- ☑ Specify audio units
- ☑ Create a graph, then obtain the audio units
- ☑ Configure the audio units
- ☑ Connect the nodes
- ☑ Provide a user interface
- ☑ Initialize and then start the graph

Music Output with Audio Units

Music Output with Audio Units

What you’ll see in this section

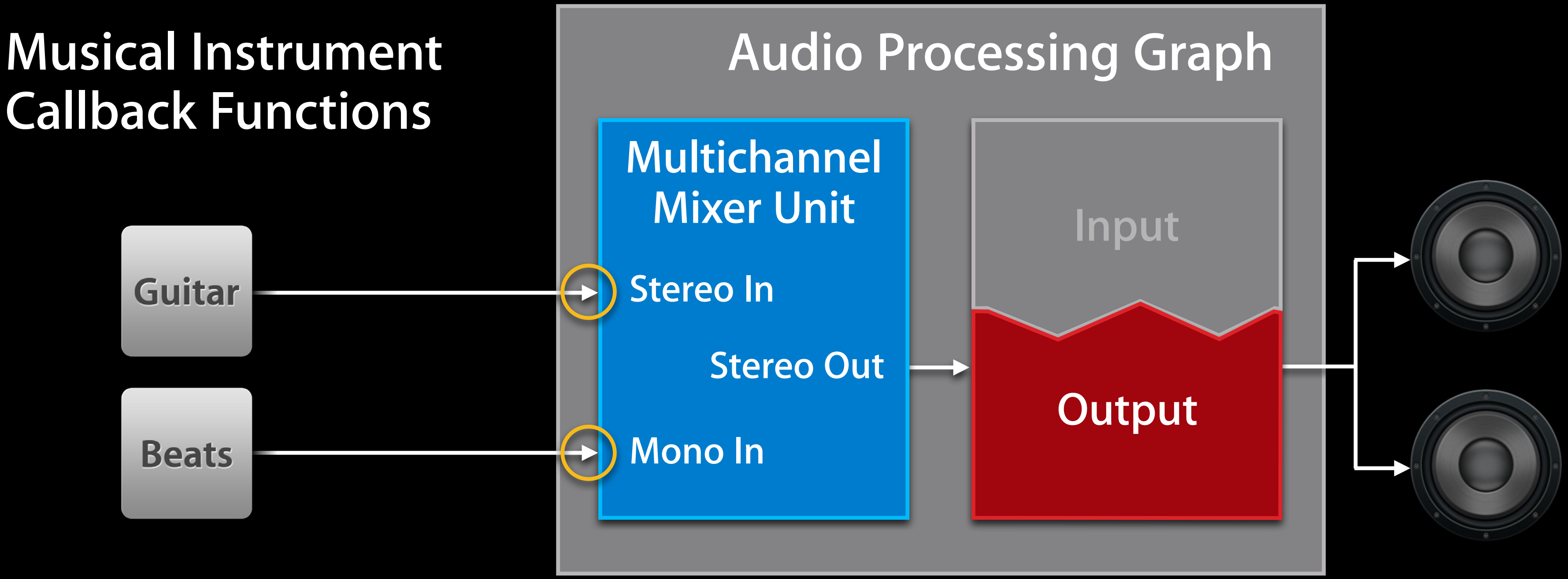
- Demo—MixerHost sample application
- Architecture of a music output app
- Building a music output app

Demo

MixerHost sample code

William Stewart
Core Audio Engineering

A Music Output Example



First Steps to Build the Music Output App

Similar to the I/O example

1	Configure the audio session	`AVAudioSession`
2	Specify audio units	`AudioComponentDescription`
3	Create a graph	`NewAUGraph, AUGraphAddNode`
4	Open the graph	`AUGraphOpen`
5	Obtain the audio units	`AUGraphNodeInfo`

Completing the Music Output Graph

Additional configuration

- Mixer requires two inputs
- Stream format on each mixer input
- Audio for each mixer input bus
    - Write two render callback functions
    - Attach the callbacks to the mixer inputs

Configure the Multichannel Mixer Unit

Set the input bus count to 2

```
UInt32 inputBusCount = 2;
```

```
AudioUnitSetProperty (
    mixerUnit,
    kAudioUnitProperty_ElementCount,
    kAudioUnitScope_Input,
    0,  // always use 0 here
    &inputBusCount,
    sizeof (inputBusCount)
);
```

Configure the Multichannel Mixer Unit

Set two mixer input stream formats

```
AudioUnitElement mixerGuitarBus = 0; // stereo
```

```
AudioUnitSetProperty (mixerUnit, kAudioUnitProperty_StreamFormat,
    kAudioUnitScope_Input, mixerGuitarBus,
    &stereoInputStreamFormat, sizeof (stereoInputStreamFormat));
```

```
AudioUnitElement mixerBeatsBus  = 1; // mono
```

```
AudioUnitSetProperty (mixerUnit, kAudioUnitProperty_StreamFormat,
    kAudioUnitScope_Input, mixerBeatsBus,
    &monoInputStreamFormat, sizeof (monoInputStreamFormat));
```

Configure the Multichannel Mixer Unit

Set the output stream sample rate

```
AudioUnitSetProperty (
    mixerUnit,
    kAudioUnitProperty_SampleRate,
    kAudioUnitScope_Output,
    0,  // there’s only one output bus
    &graphSampleRate,
    sizeof (graphSampleRate)
);
```

MaximumFramesPerSlice Primer

Definitions

Slice	The set of audio sample frames requested of an audio unit in one render cycle
Render cycle	One invocation of an audio unit’s render callback
I/O buffer duration	A read/write audio session property that determines the system slice size

MaximumFramesPerSlice Primer

Common slice sizes

	Frame count	Milliseconds at 44.1 kHz (approximate)
Default	1024	23
Screen sleep	4096	93
Low latency	256	5

MaximumFramesPerSlice Primer

When and where you must set the frames-per-slice property

	Input Running	Input Stopped
I/O units	✗	✗
All other audio units	✗	✓

Configure the Multichannel Mixer Unit

Set the maximum frames per slice

```
UInt32 maximumFramesPerSlice = 4096;
```

```
AudioUnitSetProperty (
    mixerUnit,
    kAudioUnitProperty_MaximumFramesPerSlice,
    kAudioUnitScope_Global,
    0,  // global scope has only one element
    &maximumFramesPerSlice,
    sizeof (maximumFramesPerSlice)
);
```

Attach Render Callback Functions

For audio playback

```
AURenderCallbackStruct myGuitarCallbackStruct;
myRenderCallbackStruct.inputProc = &myGuitarCallback;
myRenderCallbackStruct.inputProcRefCon = self;
```

```
AUGraphSetNodeInputCallback (processingGraph, mixerNode,
    guitarBus, &myGuitarCallbackStruct);
```

```
AURenderCallbackStruct myBeatsCallbackStruct;
myRenderCallbackStruct.inputProc = &myBeatsCallback;
myRenderCallbackStruct.inputProcRefCon = self;
```

```
AUGraphSetNodeInputCallback (processingGraph, mixerNode,
    beatsBus, &myBeatsCallbackStruct);
```

Render Callback Function Primer

Render callback functions…

- Generate or otherwise obtain audio to play
- Convey that audio to an audio unit
- Are invoked (or pulled) from upstream when output needs more data
- Live on a real-time, priority thread
    - Your work is time-constrained
    - If you miss the deadline, you get a gap in the sound

Render Callback Function Primer

Function prototype

```
OSStatus MyMusicCallbackFunction (
   void                         *inRefCon,
   AudioUnitRenderActionFlags  *ioActionFlags,
   const AudioTimeStamp         *inTimeStamp,
   UInt32                        inBusNumber,
   UInt32                        inNumberFrames,
   AudioBufferList              *ioData
);
```

See `AURenderCallback` in Audio Unit Component Services Reference

Render Callback Function Primer

The inRefCon parameter

```
OSStatus MyMusicCallbackFunction (
   void                        *inRefCon,
   AudioUnitRenderActionFlags  *ioActionFlags,
   const AudioTimeStamp        *inTimeStamp,
   UInt32                      inBusNumber,
   UInt32                      inNumberFrames,
   AudioBufferList             *ioData
);
```

- Points to context you need to generate the audio to play
- Includes any input audio needed to calculate output audio
- Specified when you attach callback to a particular bus

Render Callback Function Primer

The ioActionFlags parameter

```
OSStatus MyMusicCallbackFunction (
   void                           *inRefCon,
   AudioUnitRenderActionFlags     *ioActionFlags,
   const AudioTimeStamp           *inTimeStamp,
   UInt32                         inBusNumber,
   UInt32                         inNumberFrames,
   AudioBufferList                *ioData
);
```

- Typically, there are no flags for you on function input
- Use on output to indicate silence:

  `kAudioUnitRenderAction_OutputIsSilence`
- If playing silence, explicitly `memset` the `ioData` buffers to `0`

Render Callback Function Primer

The inTimeStamp parameter

```
OSStatus MyMusicCallbackFunction (
   void                        *inRefCon,
   AudioUnitRenderActionFlags  *ioActionFlags,
   const AudioTimeStamp        *inTimeStamp,
   UInt32                      inBusNumber,
   UInt32                      inNumberFrames,
   AudioBufferList             *ioData
);
```

- Parameter’s mSampleTime field is a sample-frame counter
- On each invocation, mSampleTime increases by inNumberFrames
- You can use it for scheduling

Render Callback Function Primer

The inBusNumber parameter

```
OSStatus MyMusicCallbackFunction (
   void                         *inRefCon,
   AudioUnitRenderActionFlags   *ioActionFlags,
   const AudioTimeStamp         *inTimeStamp,
   UInt32                        inBusNumber,
   UInt32                        inNumberFrames,
   AudioBufferList              *ioData
);
```

- Indicates the audio unit bus that invoked the callback
- When attaching callback, you specify inRefCon explicitly per bus

Render Callback Function Primer

The inNumberFrames parameter

```
OSStatus MyMusicCallbackFunction (
   void                         *inRefCon,
   AudioUnitRenderActionFlags   *ioActionFlags,
   const AudioTimeStamp         *inTimeStamp,
   UInt32                       inBusNumber,
   UInt32                       inNumberFrames,
   AudioBufferList              *ioData
);
```

- The number of audio sample frames the callback must provide
- Increments the `inTimeStamp.mSampleTime` field

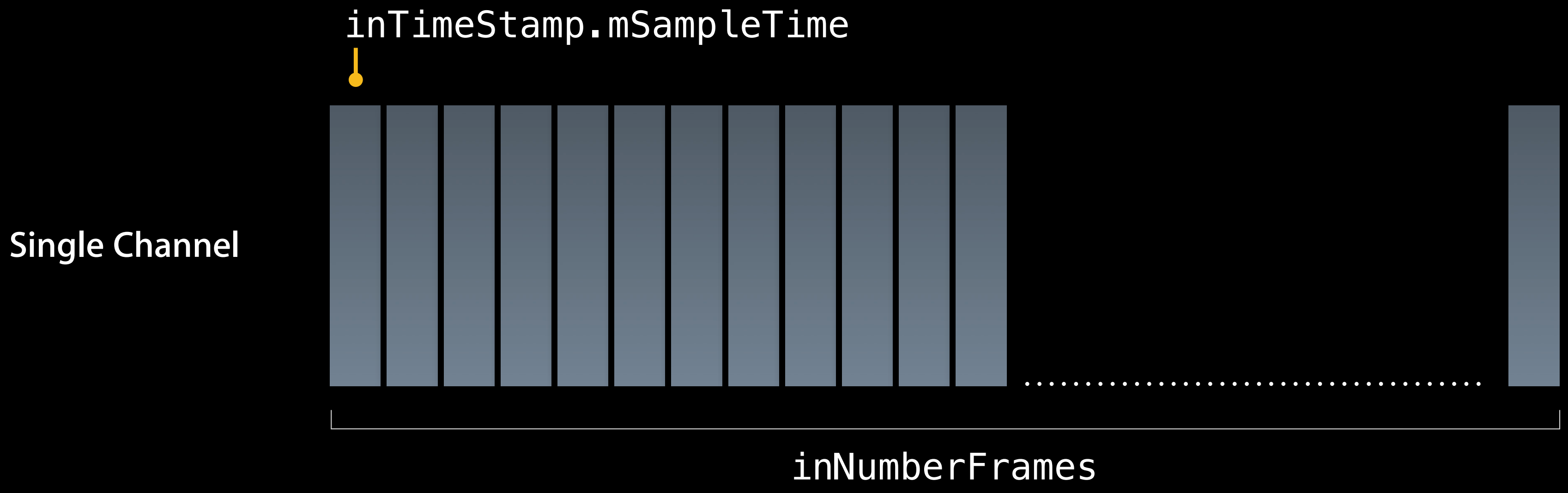
Render Callback Function Primer

The ioData parameter

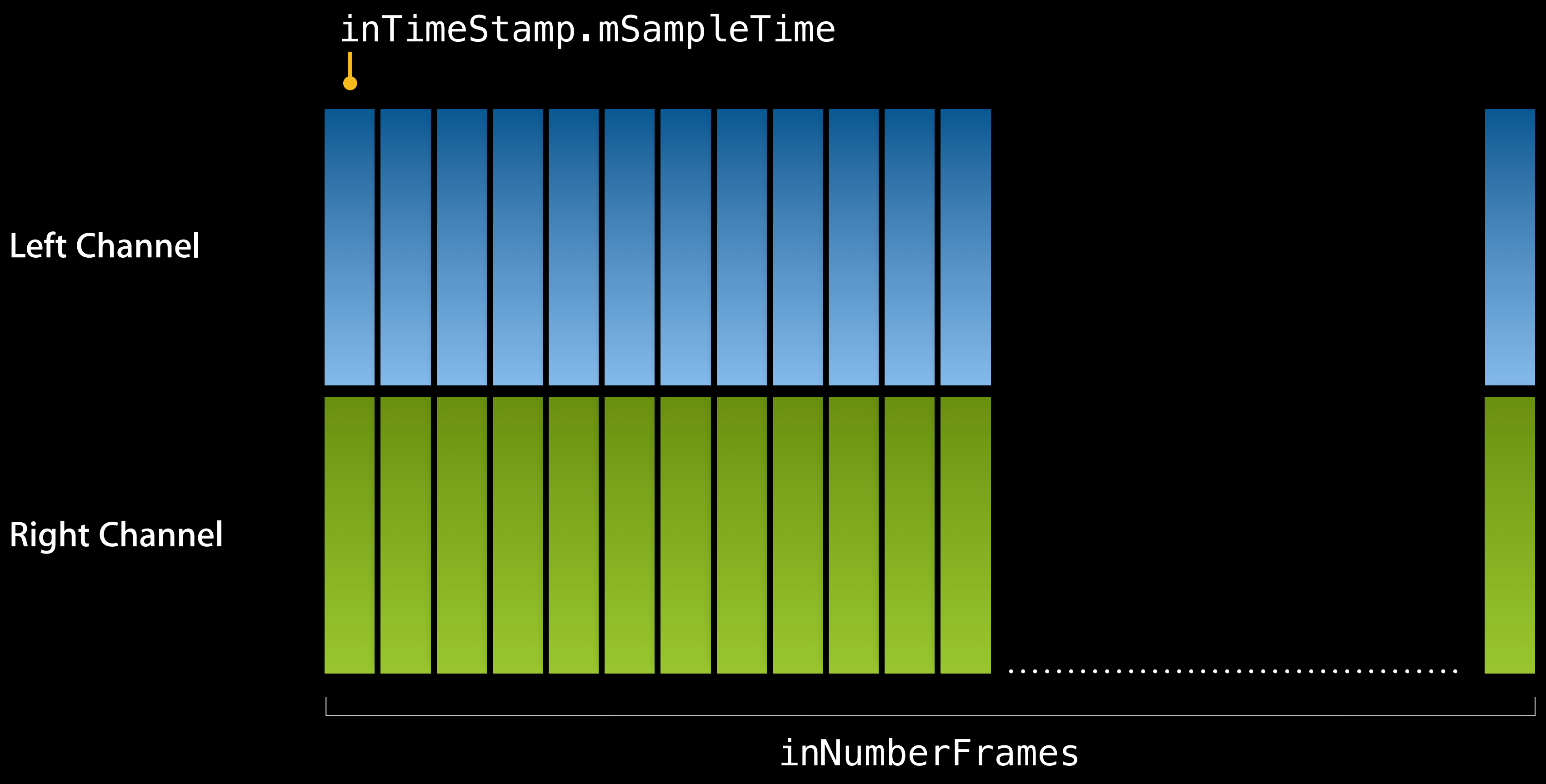
```
OSStatus MyMusicCallbackFunction (
   void                           *inRefCon,
   AudioUnitRenderActionFlags     *ioActionFlags,
   const AudioTimeStamp           *inTimeStamp,
   UInt32                         inBusNumber,
   UInt32                         inNumberFrames,
   AudioBufferList                *ioData
);
```

- You must fill this parameter with your rendered audio
- Your audio must conform to invoking bus's audio stream format
- If playing silence, explicitly `memset` the `ioData` buffers to `0`

The ioData Buffer for a Mono Callback



The ioData Buffers for a Stereo Callback



inTimeStamp.mSampleTime

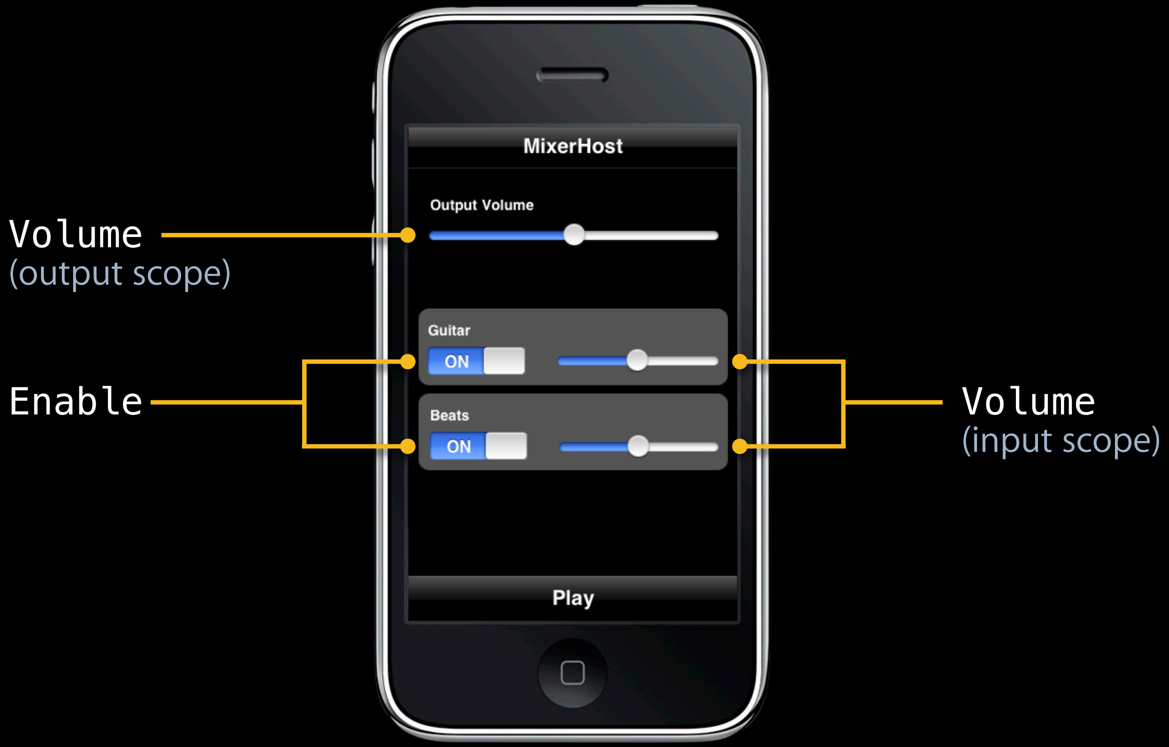
Left Channel

Right Channel

inNumberFrames

Adding a User Interface

kMultiChannelMixerParam_* parameters



Initialize and Start the Graph

```
AUGraphInitialize (processingGraph);

AUGraphStart (processingGraph);

// Some time later
AUGraphStop (processingGraph);
```

Fun with Audio Processing Graphs

Let’s get dynamic

Fun with Audio Processing Graphs

What you’ll see in this section

- Audio processing graphs add thread safety
- Architecture of a dynamic app
- Let’s change a graph—while it’s running

Audio Units Are Not Thread Safe

While processing audio…

- Do not uninitialize/reconfigure/reinitialize
- Do not make or break connections
- Do not attach or remove callbacks

AUGraph Adds Thread Safety

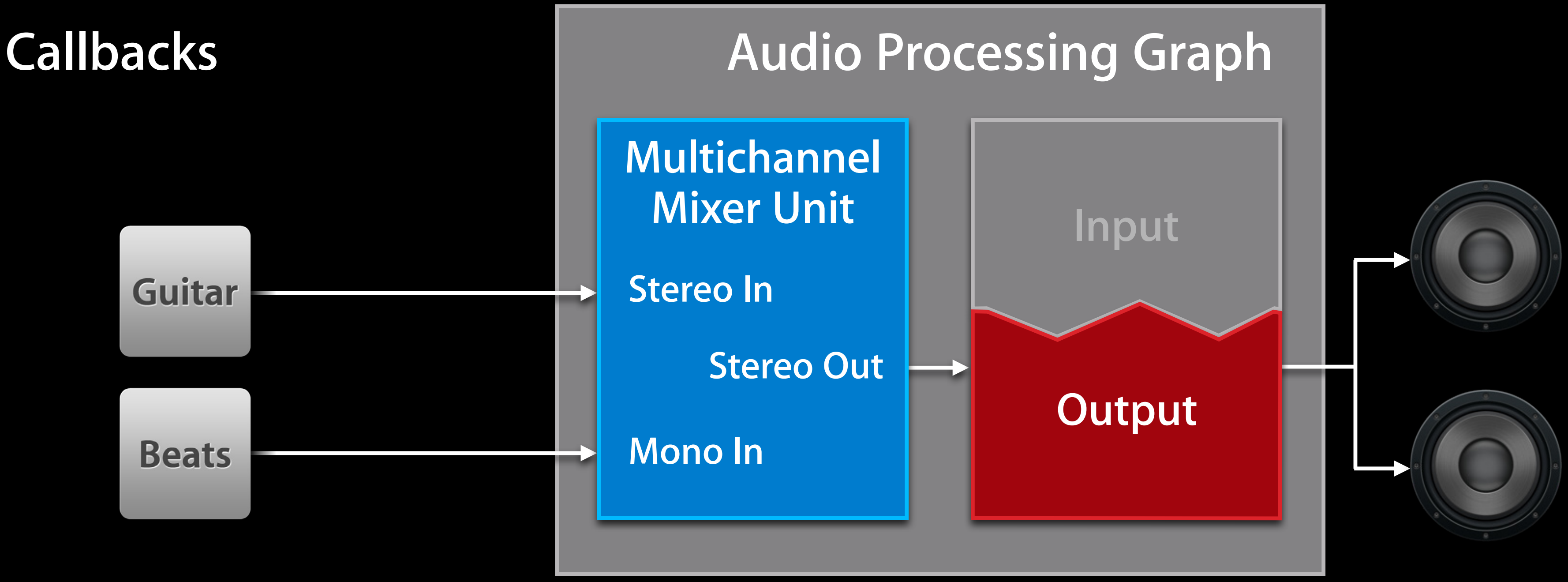
Just two steps for dynamic reconfiguration

- Specify the changes you want
    - Add/remove audio units
    - Make/break connections
    - Attach/remove callback functions
- Call `AUGraphUpdate` to implement the changes
- There is no step three

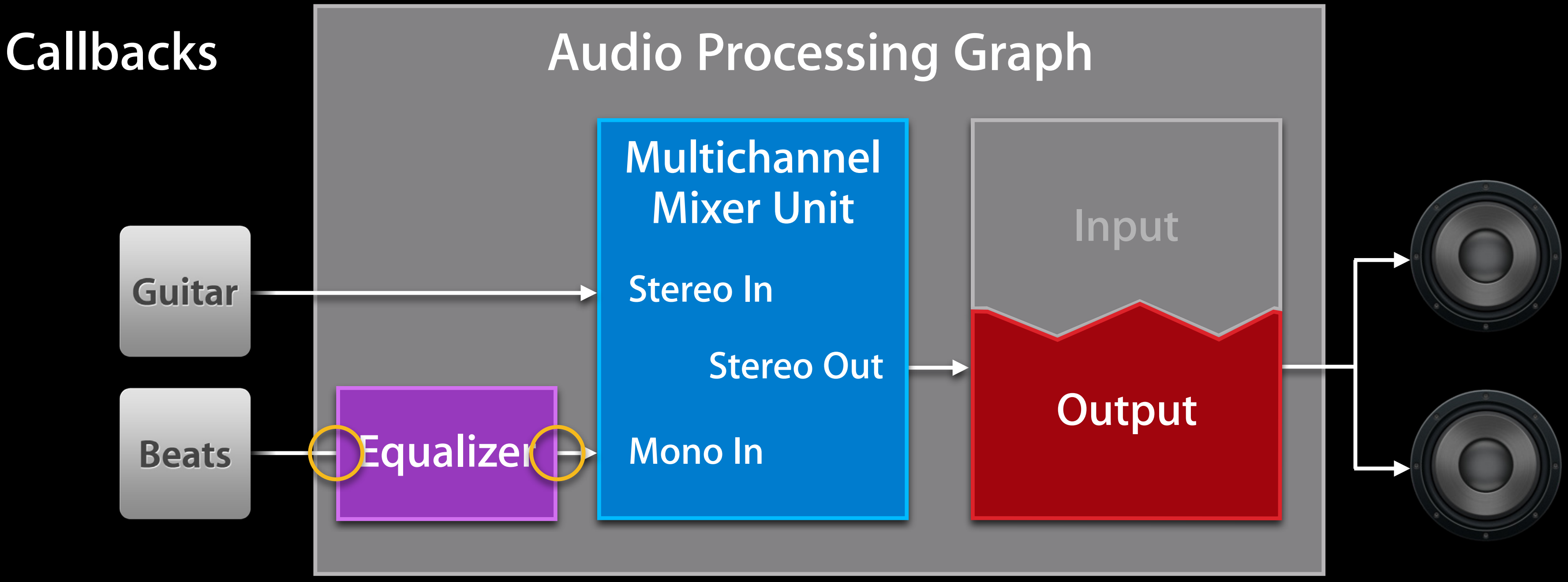
The AUGraph To-Do List Metaphor

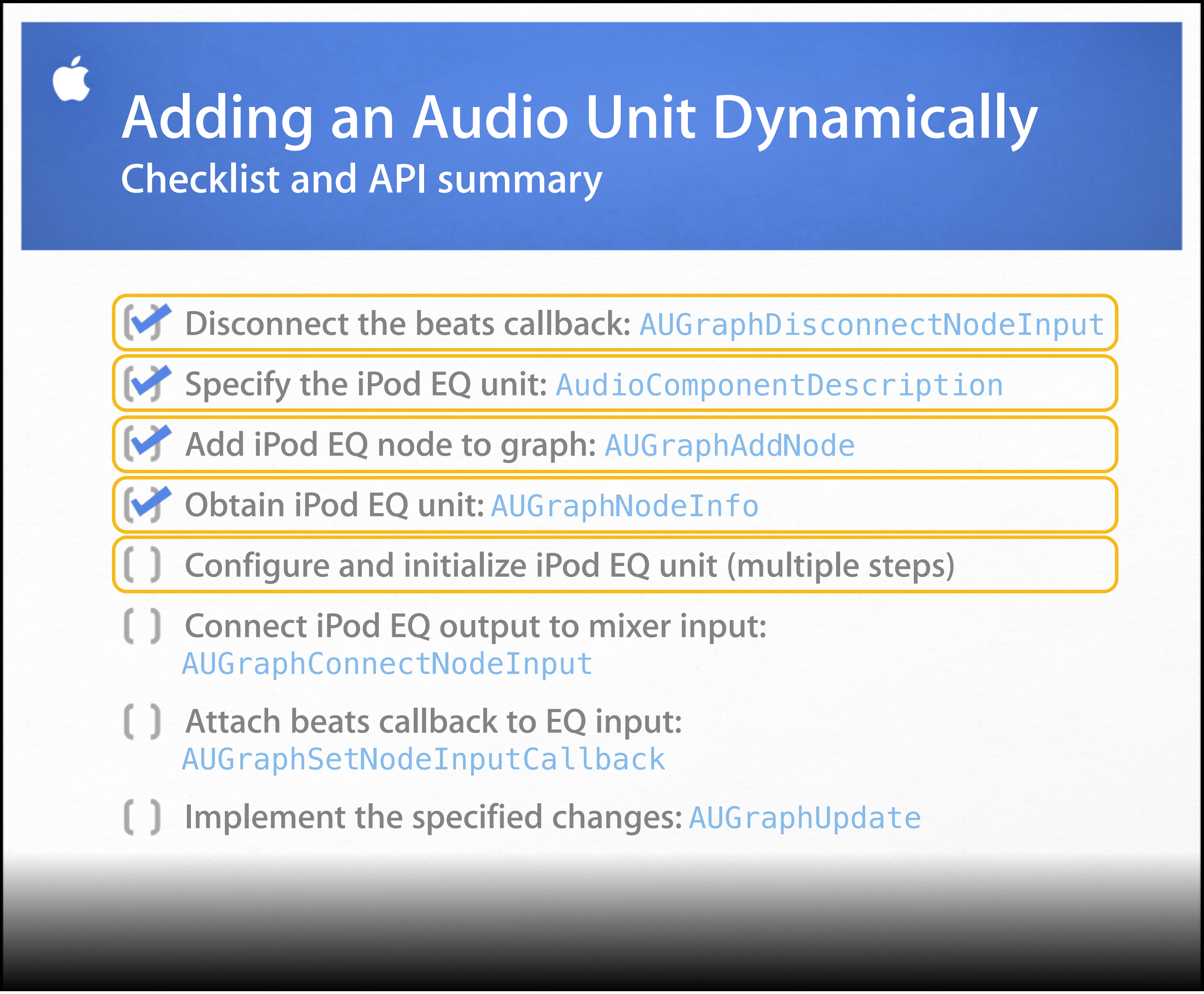
	Typical Time to Perform Call	Execution Semantic
Most AUGraph* Calls	Any time	Adds to the graph’s to-do list
AUGraphInitialize	Not running	Executes the to-do list
AUGraphUpdate	Running	

Adding an Audio Unit Dynamically



Adding an Audio Unit Dynamically





Adding an Audio Unit Dynamically

Checklist and API summary

- ☑ Disconnect the beats callback: `AUGraphDisconnectNodeInput`
- ☑ Specify the iPod EQ unit: `AudioComponentDescription`
- ☑ Add iPod EQ node to graph: `AUGraphAddNode`
- ☑ Obtain iPod EQ unit: `AUGraphNodeInfo`
- ☐ Configure and initialize iPod EQ unit (multiple steps)
- ☐ Connect iPod EQ output to mixer input: `AUGraphConnectNodeInput`
- ☐ Attach beats callback to EQ input: `AUGraphSetNodeInputCallback`
- ☐ Implement the specified changes: `AUGraphUpdate`

Configure the iPod EQ Unit

Retrieve the stream format from mixer’s beats input

```
AudioUnitGetProperty (
    mixerUnit,
    kAudioUnitProperty_StreamFormat,
    kAudioUnitScope_Input,
    beatsBus,
    &beatsStreamFormat,
    sizeof (beatsStreamFormat)
);
```

Configure the iPod EQ Unit

Apply the stream format to iPod EQ input and output

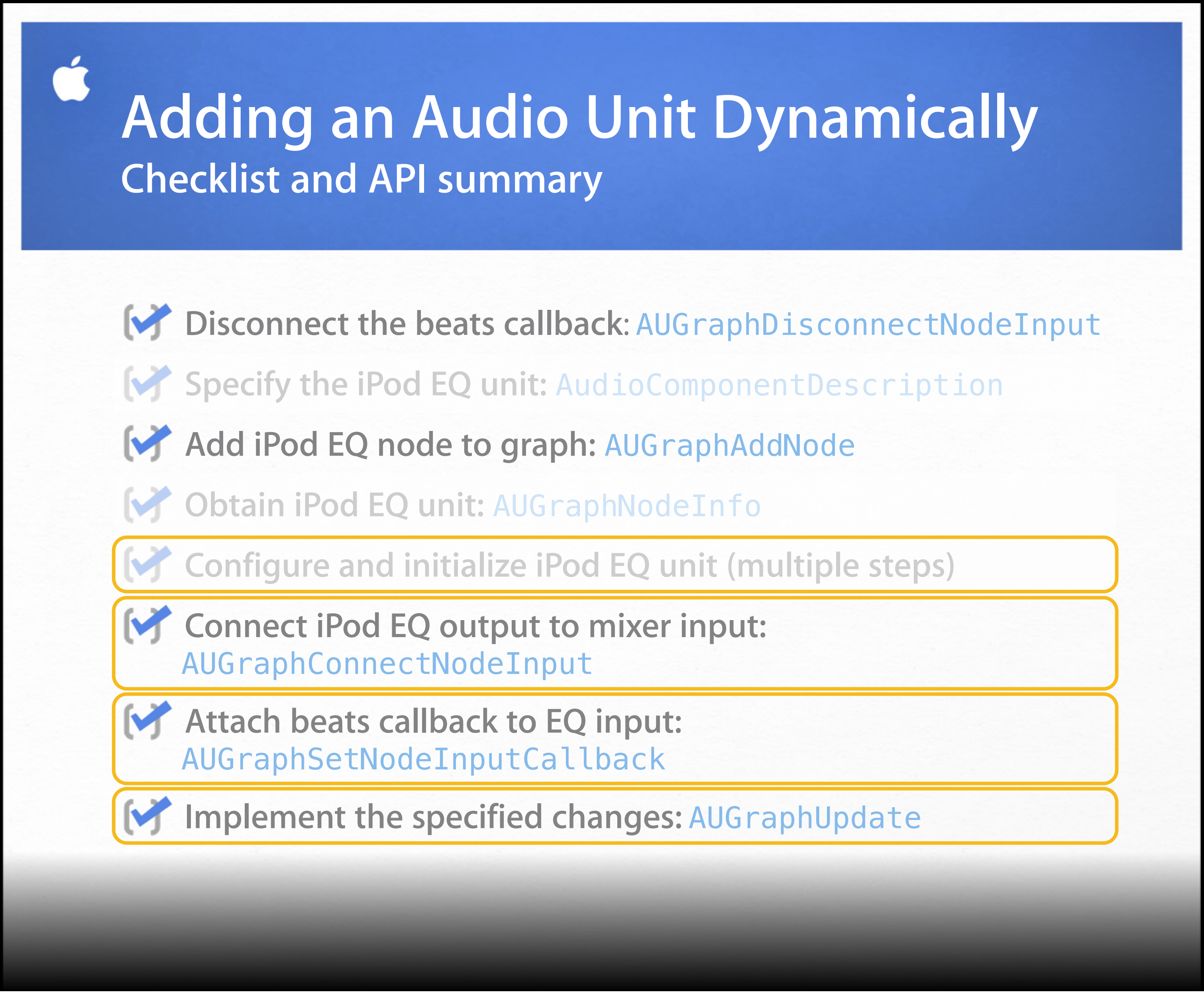
```
AudioUnitSetProperty (eqUnit, kAudioUnitProperty_StreamFormat,
    kAudioUnitScope_Input, 0, &beatsStreamFormat,
    sizeof (beatsStreamFormat));
```

```
AudioUnitSetProperty (eqUnit, kAudioUnitProperty_StreamFormat,
    kAudioUnitScope_Output, 0, &beatsStreamFormat,
    sizeof (beatsStreamFormat));
```

Explicitly Initialize the iPod EQ Unit

Allocate resources before calling AUGraphUpdate

```
AudioUnitInitialize (eqUnit);
```



Adding an Audio Unit Dynamically

Checklist and API summary

- ☑ Disconnect the beats callback: `AUGraphDisconnectNodeInput`
- ☑ Specify the iPod EQ unit: `AudioComponentDescription`
- ☑ Add iPod EQ node to graph: `AUGraphAddNode`
- ☑ Obtain iPod EQ unit: `AUGraphNodeInfo`
- ☑ Configure and initialize iPod EQ unit (multiple steps)
- ☑ Connect iPod EQ output to mixer input: `AUGraphConnectNodeInput`
- ☑ Attach beats callback to EQ input: `AUGraphSetNodeInputCallback`
- ☑ Implement the specified changes: `AUGraphUpdate`

Audio Processing Graph Wrap-Up

Audio processing graphs…

- Always include exactly one I/O unit
- Add thread safety to audio units
- Use a to-do list metaphor

More Information

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

Eryk Vershen
Media Technologies Evangelist
evershen@apple.com

Documentation and Sample Code
iPhone Dev Center
http://developer.apple.com/iphone

Audio Unit Hosting Guide for iPhone OS
WWDC attendee website

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

Related Sessions

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

Labs

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

Summary

- Use audio units for real-time, highest-performance sound
- Use I/O units to gain access to audio hardware
- Configure and customize using properties
- Control using UI and parameters
- Understand audio unit life cycle: access, instantiation, configuration, initialization, rendering
- Use render callbacks to feed your own audio to an audio unit
- Use audio processing graphs to manage audio units while they are producing sound

WWDC10

The last slide after the logo is intentionally left blank for all