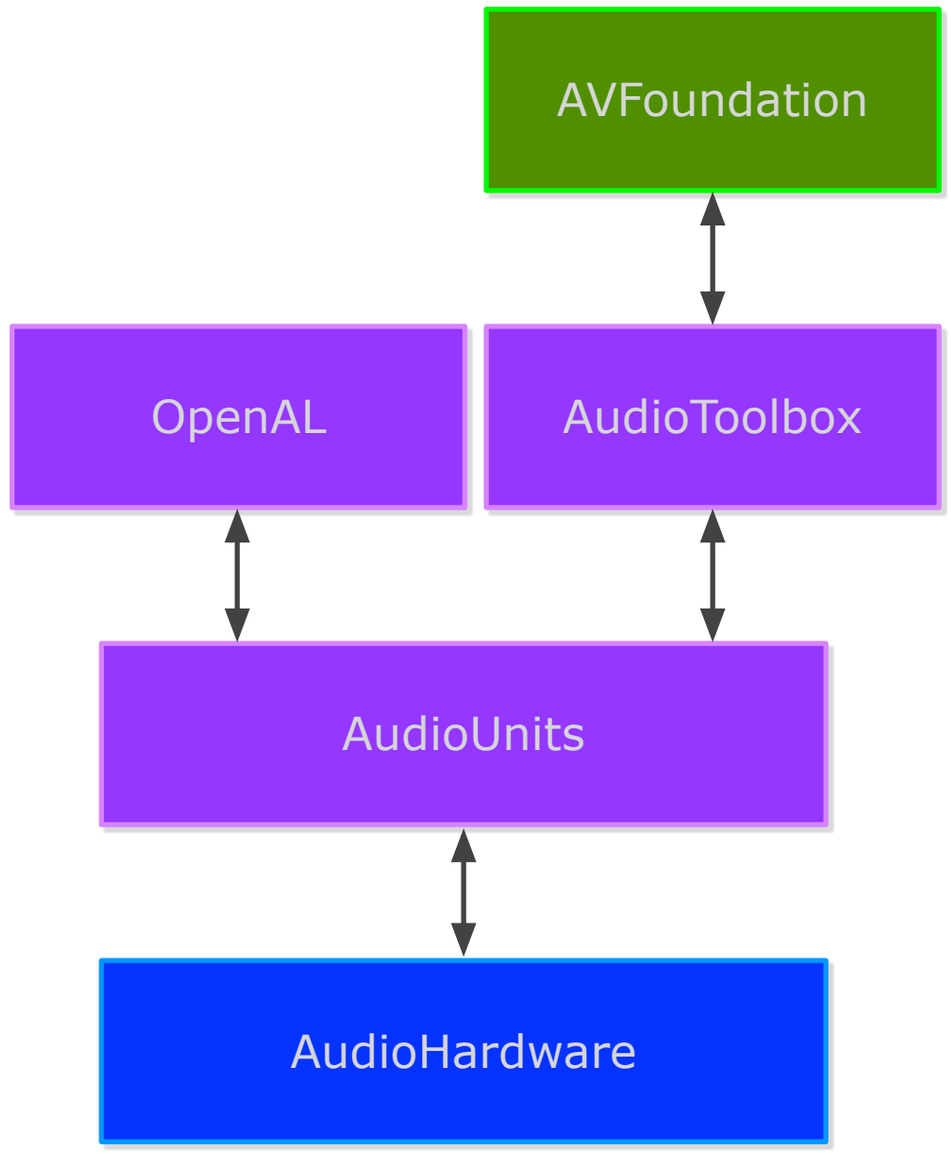
# Audio Unit Graphs

Dave Dribin – @ddribin 

# Playing Audio on iPhone OS

- ~~Media Player Framework~~
- ~~AV Foundation Framework~~
- ~~OpenAL Library~~
- Audio Toolbox Framework
- Audio Unit Framework

# Audio Architecture



# Disadvantage of Audio Queues

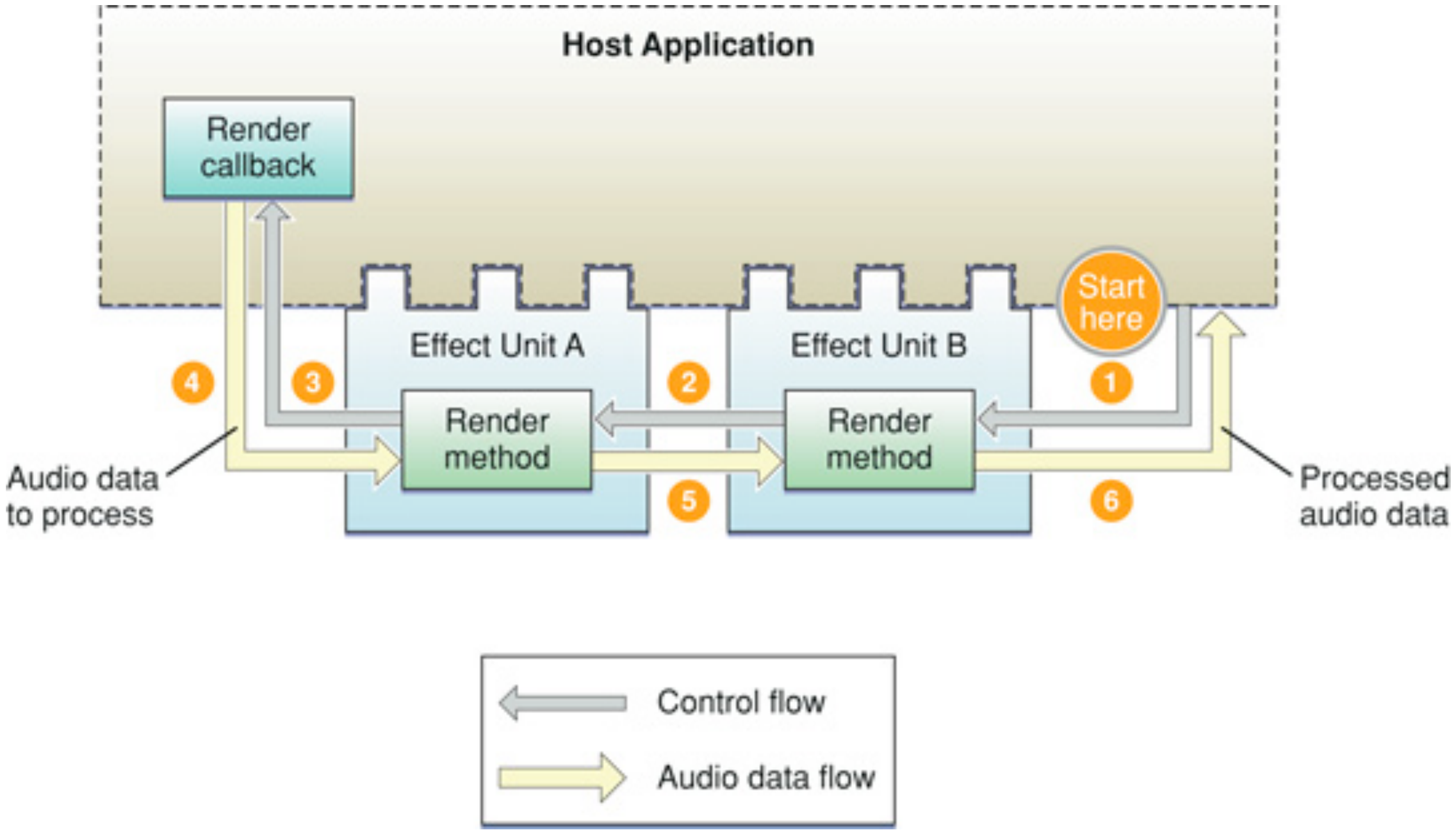
- High Latency
- Cannot Access iPod Equalizer
- No way to get access to samples “as they play”

# Audio Units

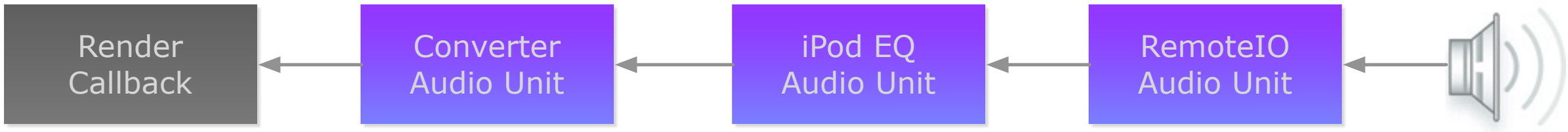
- Work only with uncompressed audio
- Work in constrained environment

# What are Audio Units?

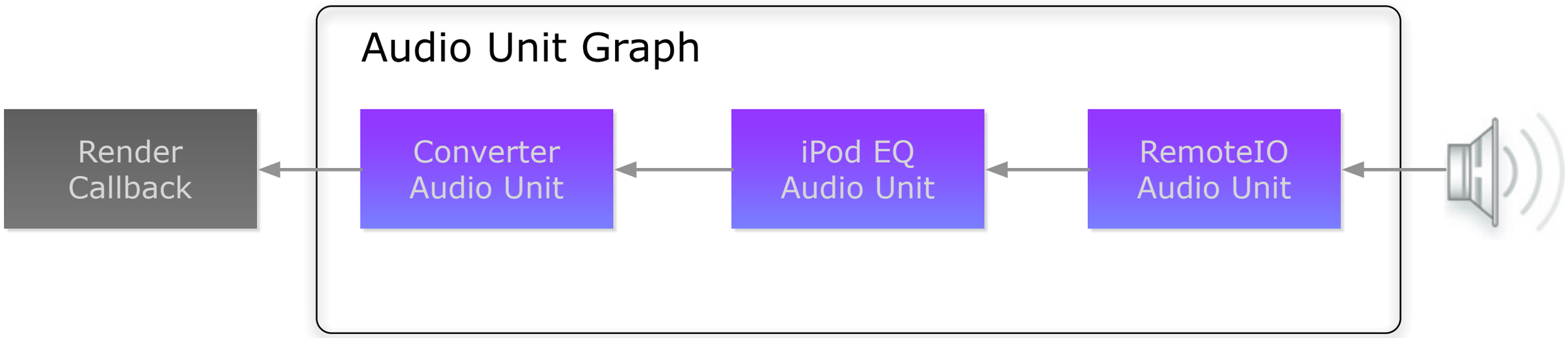
# Audio Unit Pull Model



# Core Audio Pull Model



# Audio Unit Graph



# Audio Unit Graph Data Types

- AUGraph
- AUNode
- AudioUnit
- AudioComponentDescription

```
typedef struct AudioComponentDescription {
    OSType              componentType;
    OSType              componentSubType;
    OSType              componentManufacturer;
    UInt32              componentFlags;
    UInt32              componentFlagsMask;
} AudioComponentDescription;
```

# Audio Unit “Hello World”

- Play 440 Hz sound
- A440 Project on BitBucket:
  - http://bitbucket.org/ddribin/a440

# Demo

```objc
@interface A440AUGraph : NSObject <A440Player>
{
    AUGraph _graph;
    AUNode _outputNode;
    AUNode _converterNode;
    AudioStreamBasicDescription _dataFormat;

    A440SineWaveGenerator _sineWaveGenerator;
}

- (BOOL)play:(NSError **)error;
- (BOOL)stop:(NSError **)error;

@end
```

```
- (BOOL)play:(NSError **)error;
{
    NSAssert(_graph == NULL, @"Graph is already started");

    OSStatus status;

    FAIL_ON_ERR(NewAUGraph(&_graph));
    FAIL_ON_ERR([self addOutputNode]);
    FAIL_ON_ERR([self addConverterNode]);
    FAIL_ON_ERR(AUGraphConnectNodeInput(_graph, _converterNode, 0,
                                        _outputNode, 0));
    FAIL_ON_ERR(AUGraphOpen(_graph));
    [self setupDataFormat];
    FAIL_ON_ERR([self setDataFormatOfConverterAudioUnit]);
    FAIL_ON_ERR([self setMaximumFramesPerSlice]);
    FAIL_ON_ERR([self setRenderCallbackOfConverterNode]);
    FAIL_ON_ERR(AUGraphInitialize(_graph));
    A440SineWaveGeneratorInitWithFrequency(&_sineWaveGenerator,
                                           440.0);
    FAIL_ON_ERR(AUGraphStart(_graph));

    return YES;

failed:
    // Error handling...
    return NO;
}
```

```
- (OSStatus)addOutputNode;
{
    AudioComponentDescription description = {
        .componentType = kAudioUnitType_Output,
#if TARGET_OS_IPHONE
        .componentSubType = kAudioUnitSubType_RemoteIO,
#else
        .componentSubType = kAudioUnitSubType_DefaultOutput,
#endif
        .componentManufacturer = kAudioUnitManufacturer_Apple,
    };
    return AUGraphAddNode(_graph, &description, &_outputNode);
}
```

```
- (OSStatus)addConverterNode;
{
    AudioComponentDescription description = {
        .componentType = kAudioUnitType_FormatConverter,
        .componentSubType = kAudioUnitSubType_AUConverter,
        .componentManufacturer = kAudioUnitManufacturer_Apple,
    };
    return AUGraphAddNode(_graph, &description, &_converterNode);
}
```

```
- (BOOL)play:(NSError **)error;
{
    NSAssert(_graph == NULL, @"Graph is already started");

    OSStatus status;

    FAIL_ON_ERR(NewAUGraph(&_graph));
    FAIL_ON_ERR([self addOutputNode]);
    FAIL_ON_ERR([self addConverterNode]);
    FAIL_ON_ERR(AUGraphConnectNodeInput(_graph, _converterNode, 0,
                                        _outputNode, 0));
    FAIL_ON_ERR(AUGraphOpen(_graph));
    [self setupDataFormat];
    FAIL_ON_ERR([self setDataFormatOfConverterAudioUnit]);
    FAIL_ON_ERR([self setMaximumFramesPerSlice]);
    FAIL_ON_ERR([self setRenderCallbackOfConverterNode]);
    FAIL_ON_ERR(AUGraphInitialize(_graph));
    A440SineWaveGeneratorInitWithFrequency(&_sineWaveGenerator,
                                           440.0);
    FAIL_ON_ERR(AUGraphStart(_graph));

    return YES;

failed:
    // Error handling...
    return NO;
}
```

```
- (OSStatus)setDataFormatOfConverterAudioUnit;
{
    AudioUnit converterAudioUnit;
    OSStatus status;
    status = AUGraphNodeInfo(_graph, _converterNode,
                             NULL, &converterAudioUnit);
    if (status != noErr) {
        return status;
    }

    status = AudioUnitSetProperty(converterAudioUnit,
                                  kAudioUnitProperty_StreamFormat,
                                  kAudioUnitScope_Input,
                                  0,
                                  &_dataFormat,
                                  sizeof(_dataFormat));
    return status;
}
```

# Technical Q&A QA1606

```
- (OSStatus)setMaximumFramesPerSlice;
{

    AudioUnit converterAudioUnit;
    OSStatus status;
    status = AUGraphNodeInfo(_graph, _converterNode,
                             NULL, &converterAudioUnit);
    if (status != noErr) {
        return status;
    }

    UInt32 maxFramesPerSlice = 4096;
    status =
        AudioUnitSetProperty(converterAudioUnit,
                             kAudioUnitProperty_MaximumFramesPerSlice,
                             kAudioUnitScope_Global,
                             0,
                             &maxFramesPerSlice,
                             sizeof(maxFramesPerSlice));
    return status;
}
```

```objc
- (BOOL)play:(NSError **)error;
{
    NSAssert(_graph == NULL, @"Graph is already started");

    OSStatus status;

    FAIL_ON_ERR(NewAUGraph(&_graph));
    FAIL_ON_ERR([self addOutputNode]);
    FAIL_ON_ERR([self addConverterNode]);
    FAIL_ON_ERR(AUGraphConnectNodeInput(_graph, _converterNode, 0,
                                        _outputNode, 0));
    FAIL_ON_ERR(AUGraphOpen(_graph));
    [self setupDataFormat];
    FAIL_ON_ERR([self setDataFormatOfConverterAudioUnit]);
    FAIL_ON_ERR([self setMaximumFramesPerSlice]);
    FAIL_ON_ERR([self setRenderCallbackOfConverterNode]);
    FAIL_ON_ERR(AUGraphInitialize(_graph));
    A440SineWaveGeneratorInitWithFrequency(&_sineWaveGenerator,
                                           440.0);
    FAIL_ON_ERR(AUGraphStart(_graph));

    return YES;

failed:
    // Error handling...
    return NO;
}
```

```
static OSStatus MyRenderer(
    void *                          inRefCon,
    AudioUnitRenderActionFlags * ioActionFlags,
    const AudioTimeStamp *          inTimeStamp,
    UInt32                          inBusNumber,
    UInt32                          inNumberFrames,
    AudioBufferList *               ioData)
{
    A440AUGraph * self = inRefCon;

    int16_t * sample = ioData->mBuffers[0].mData;
    UInt32 channelsPerFrame = self->_dataFormat.mChannelsPerFrame;

    for (UInt32 i = 0; i < inNumberFrames; i++) {
        FillFrame(self, sample);
        sample += channelsPerFrame;
    }

    ioData->mBuffers[0].mDataByteSize =
        inNumberFrames*self->_dataFormat.mBytesPerFrame;

    return noErr;
}
```

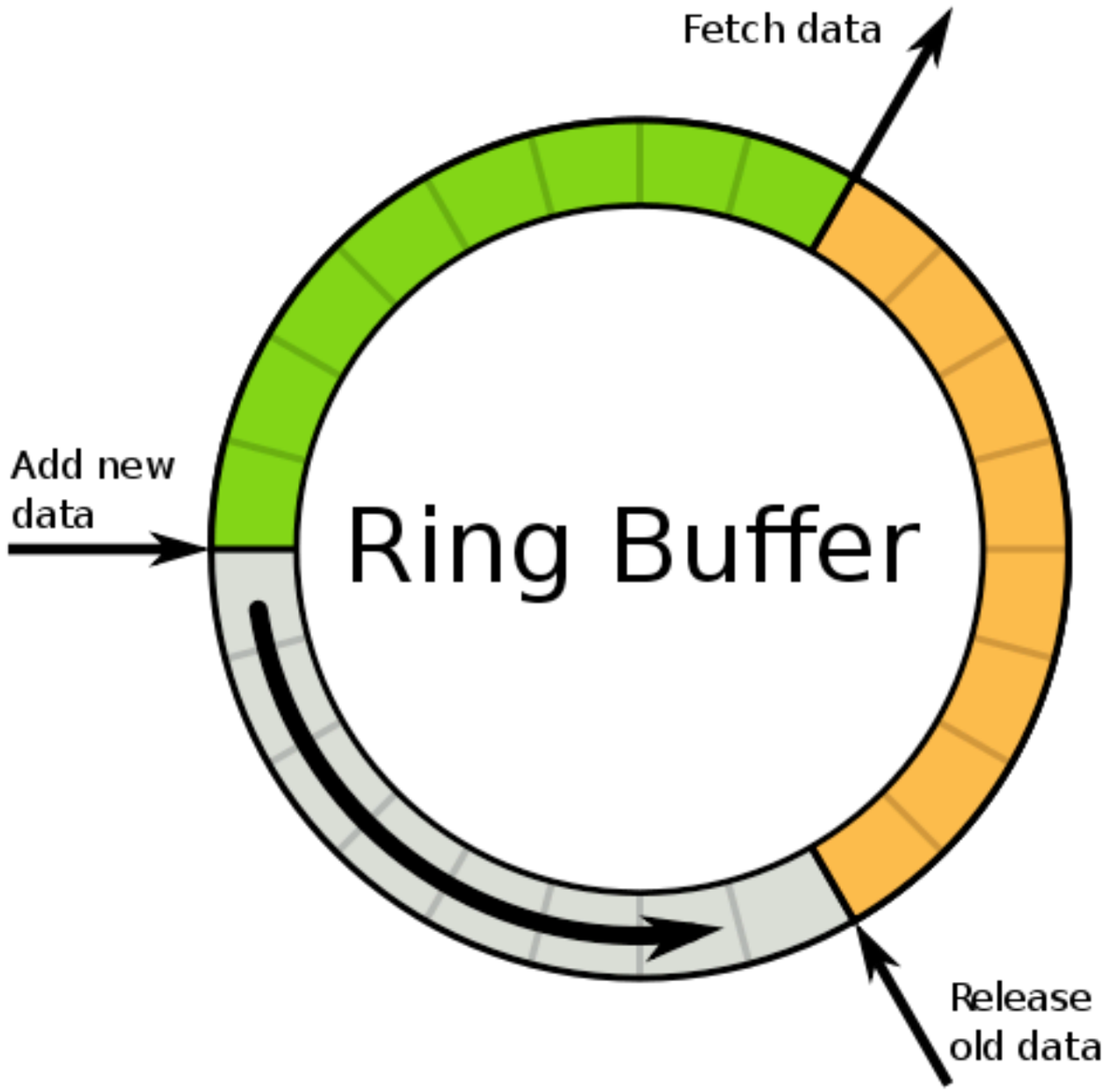
# AURenderCallback Dangers

- Don’t allocate any memory
- Don’t take any locks
- Don’t do any I/O
- Don’t waste time
- Don’t use any Objective-C?

# AURenderCallback Solutions

- Lockless Ring Buffers
    - CARingBuffer in “Public Utility”
- Lockless Buffer Queues
- Hope

# Ring Buffer

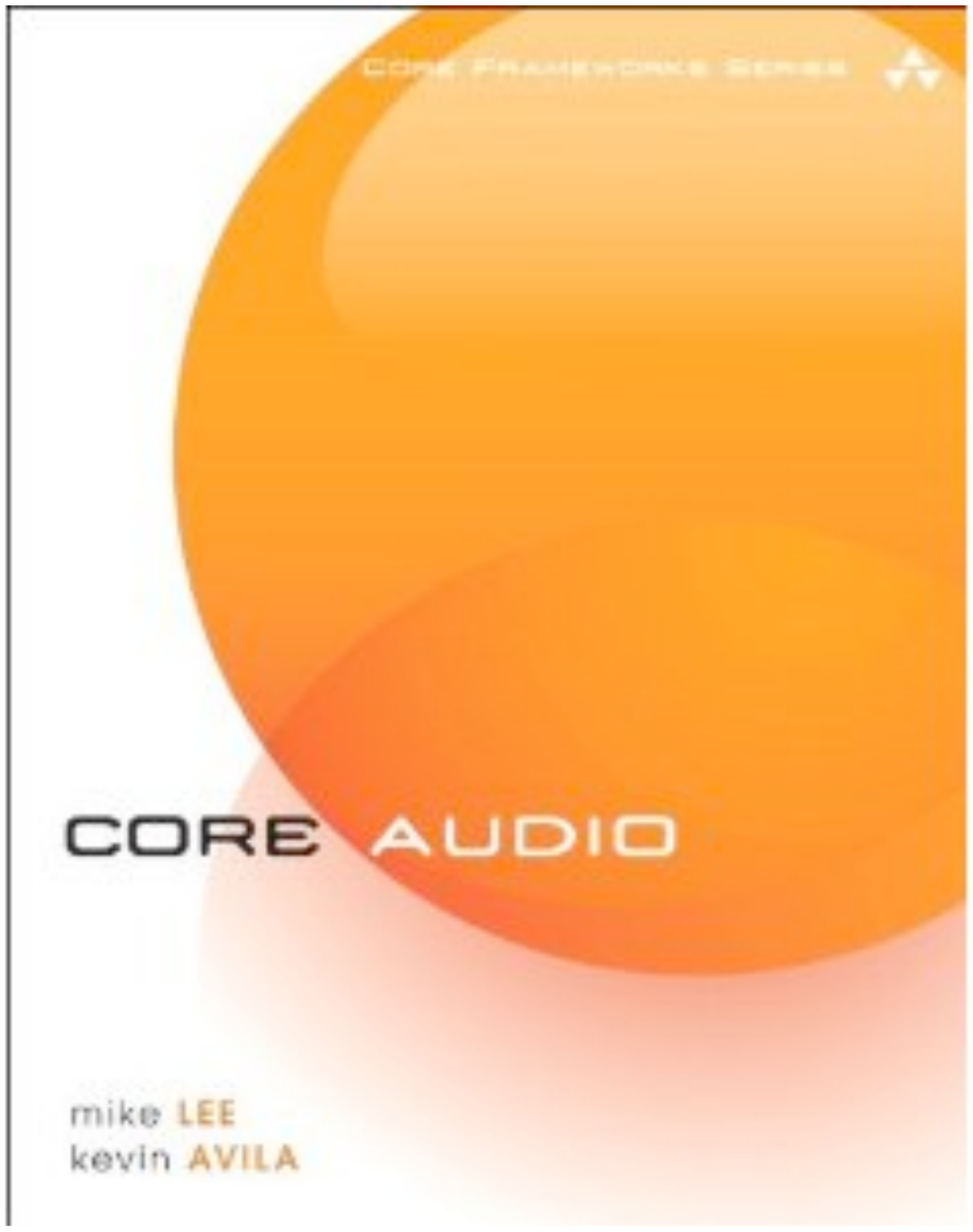


# Lockless Buffer Queue

- Dequeue buffers without locking
    - Very carefully using OSAtomicCompareAndSwapPtrBarrier()
- Still need a way to signal render callback
    - CFRunLoopSource
    - CFRunLoopSignal
    - CFRunLoopWakeup

# Chip Player Sample

- Two variations:
  - Everything in render callback
  - Using lockless buffer queues


CORE FRAMEWORKS SERIES
CORE AUDIO
mike LEE
kevin AVILA

# MobileSynth

- Open Source synthesizer
- http://code.google.com/p/mobilesynth/