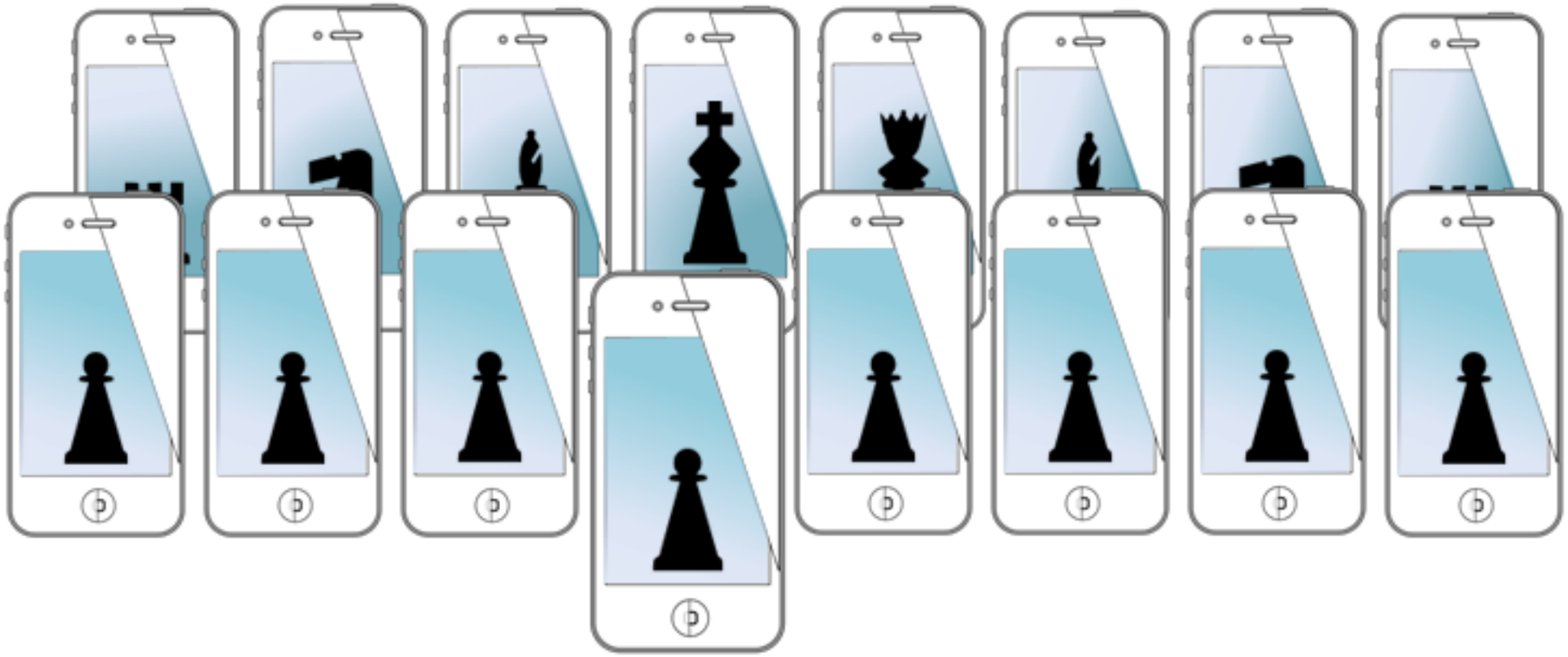
MOBILE SENSING LEARNING



CS5323 & 7323

Mobile Sensing and Learning

core audio

Eric C. Larson, Lyle School of Engineering,
Computer Science, Southern Methodist University

agenda

- blocks and multi-threading review
- core audio intro

blocks and closures

- not callback functions (but similar)
  - created at runtime
  - once created, can be called multiple times
  - can access data from scope when defined
  - syntax is different in swift and objective-c (also slightly different behavior)
- not exactly a lambda (*but similar*)
  - but it acts like an object that can be passed as an argument or created on the fly
- swift uses closures, objective-c uses blocks

block/closure syntax

most common usage is as input into a function

```
^(Parameters) {
    // code
}
```

enumerate with block

```
// here the block is created on the fly for the enumeration
[myArray enumerateObjectsUsingBlock:^(NSNumber *obj, NSUInteger idx, BOOL *stop) {
    // print the value of the NSNumber in a variety of ways
    NSLog(@"Float Value = %.2f, Int Value = %d",[obj floatValue],[obj integerValue]);
}];
```

swift syntax

```
myArray.enumerateObjects({obj, idx, ptr in
    print("\(obj) is at index \(idx)")
})
```

```
{ (parameters) -> return type in
    statements
}
```

some semantics

- variables from same scope where block is defined are **read only**, unless you use keyword:

```
__block NSNumber * valForBlock = @5.0;
```

- classes hold a **strong** pointer to blocks and blocks hold a **strong** pointer to __block variables (retain cycle)

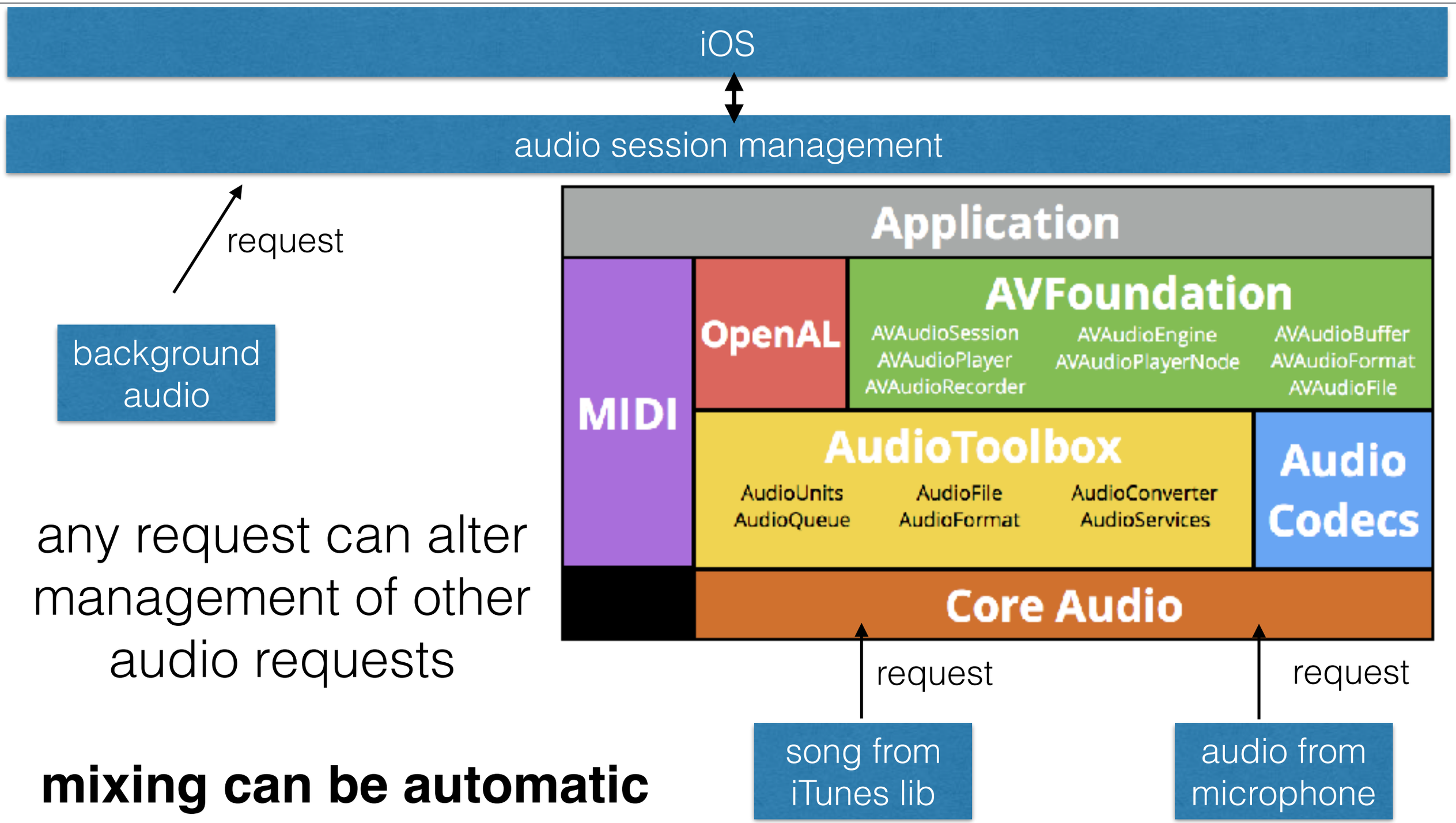
```
__block ViewController * __weak  weakSelf = self;
weakSelf.value = (some function in block)
```

```
DispatchQueue.main.async{ [weak self] in
    self.value = (some closure function)
}
```

Core Audio

- many **audio packages** exist, but we want **low level signals**
- **Audio Sessions** (high level, completely overhauled starting iOS7)
  - shared instance (for all applications)
  - set category (play, record, both)
    - choose options: like mixing with ambient sources
  - set audio route (new starting in iOS7)
    - set specific hardware within audio route
- **Audio Units** (more low level, output, input)
  - set stream format, buffer sizes, sampling rate,
  - initialize memory for audio buffers
  - set callback rendering procedure

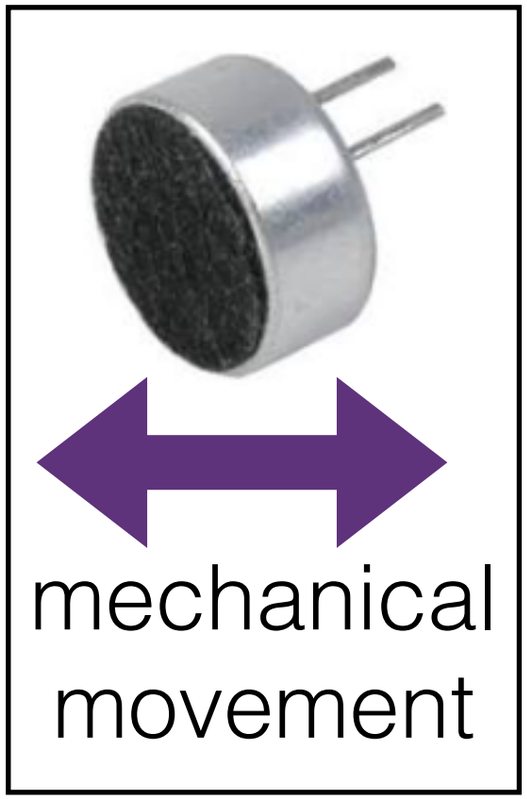
audio sessions

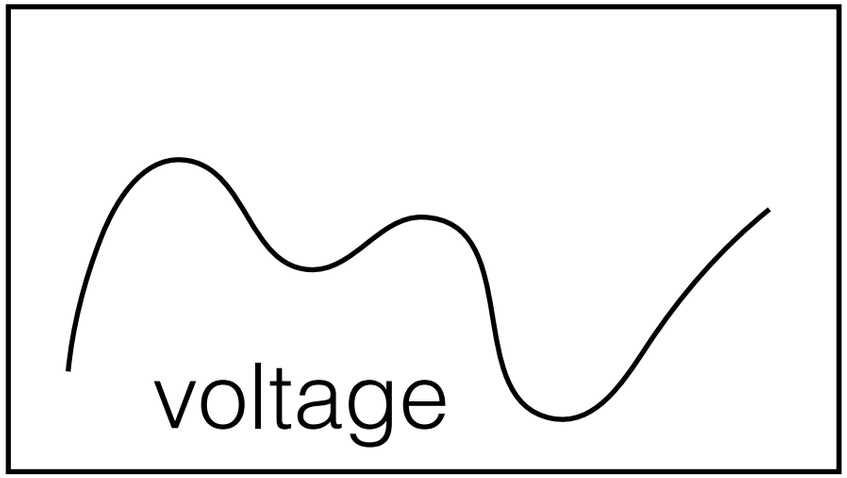


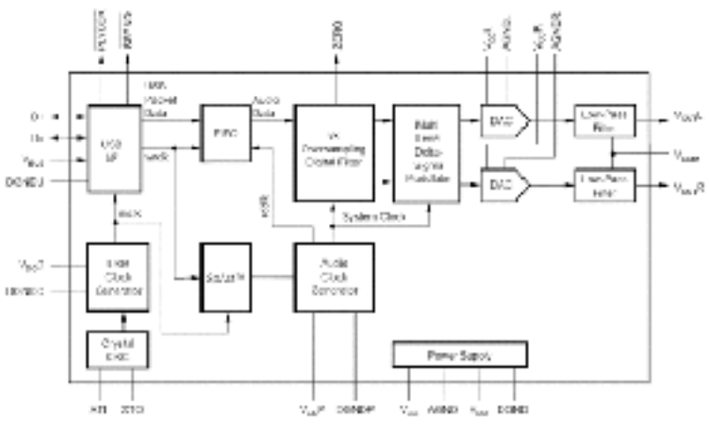
any request can alter management of other audio requests

mixing can be automatic — impressive!

audio hardware

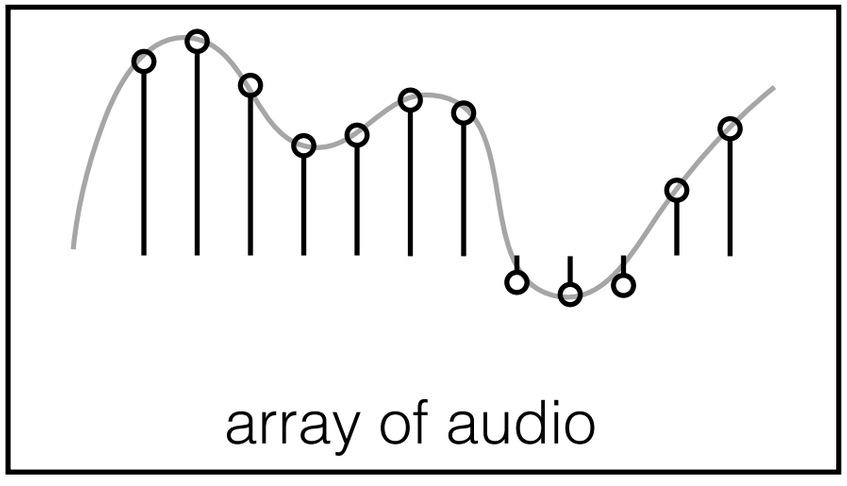






buffer and ADC

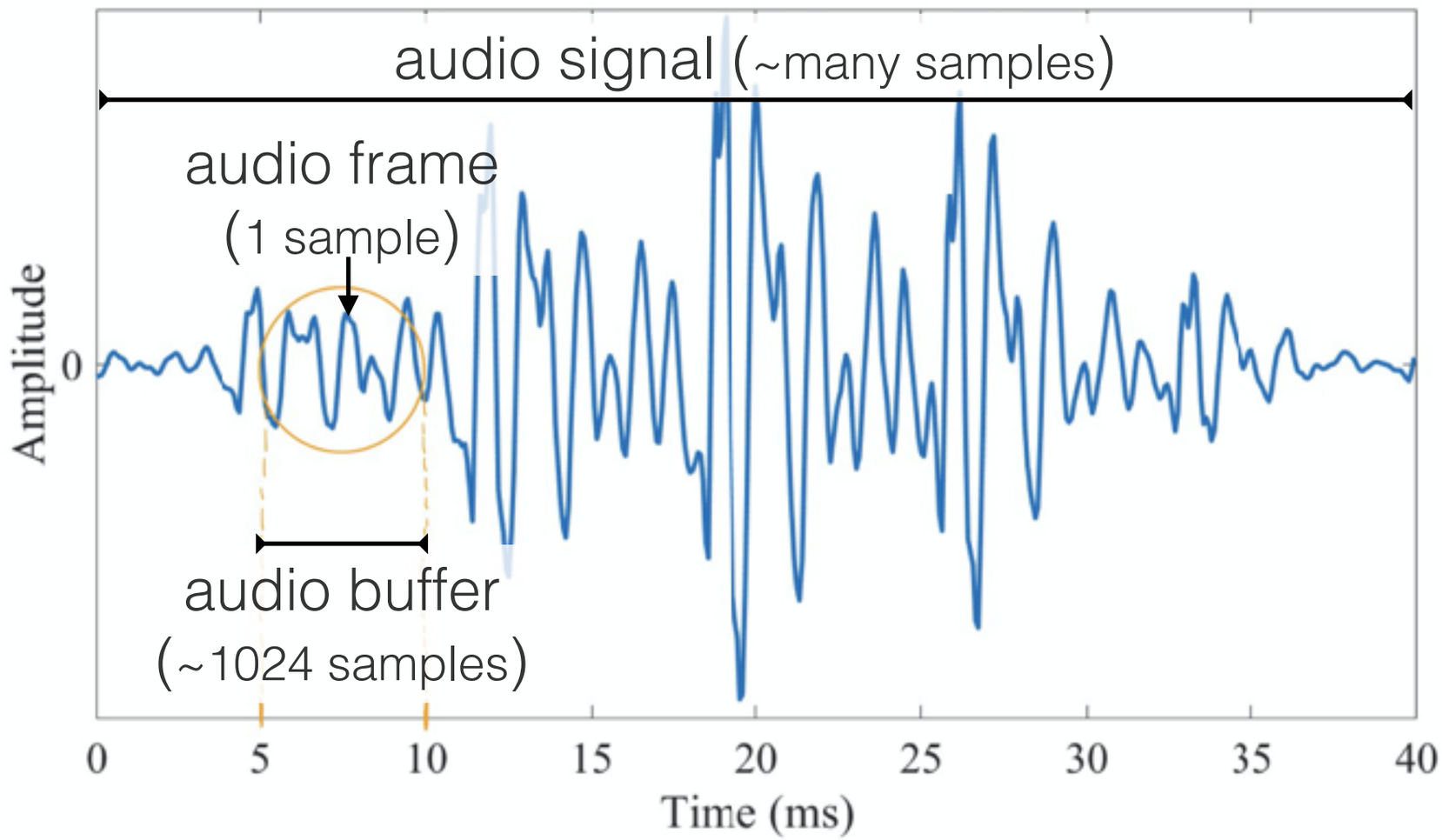
audio card



notify software a buffer is ready

audio buffering

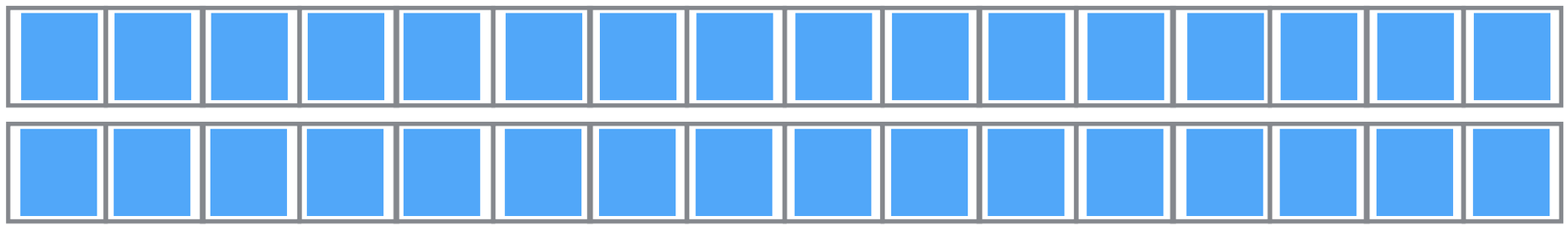
- audio card buffers up audio samples before sending to the CPU



https://medium.com/better-programming/audio-visualization-in-swift-using-metal-accelerate-part-1-390965c095d7

audio units

audio input buffer procedure, double buffer shown



Audio Card (memory allocated on card)

common language

C

sent to audio session callback

CPU

(memory in RAM)

copy over samples, convert

exit from call as soon as possible!

do not allocate memory, take locks, or waste time!!

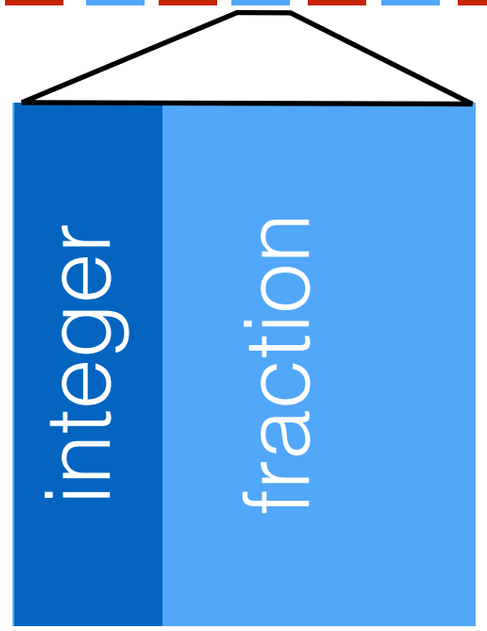
audio unit formats

microphone (input)

stereo could be interleaved (output)

right speaker

left speaker



32 bits

callback preallocates buffers
developer fills the output buffer
OS handles playing the buffer

if you don't fill fast enough, audio is choppy

audio units solution…

```
// The audio engine used to record input from the microphone.
private let audioEngine = AVAudioEngine()

// setup audio
let audioSession = AVAudioSession.sharedInstance()
do{
    try audioSession.setCategory(AVAudioSession.Category.record)
    try audioSession.setMode(AVAudioSession.Mode.measurement)
    try audioSession.setActive(true, options: .notifyOthersOnDeactivation)
}
catch { fatalError("Audio engine could not be setup”) }

let inputNode = audioEngine.inputNode
let recordingFormat = inputNode.outputFormat(forBus: 0)
```

Setup Audio

```
inputNode.installTap(onBus: 0, bufferSize: 1024, format: recordingFormat)
                        { (buffer: AVAudioPCMBuffer, when: AVAudioTime) in

}

audioEngine.prepare()
do{ try audioEngine.start() }
catch { fatalError("Audio engine could not start”) }
```

…some APIs need the AVAudioBuffer Object…

Get Samples

but, audio unit taps are slower than using core audio…

wouldn't it be **great** if there was a module that **handled** all the specifics of **audio units for us**?

Novocaine: takes the pain out of audio processing

Originally developed by **Alex Wiltschko**

Heavily manipulated by **eclarson**

Alex Wiltschko
alexbw

Twitter
Boston, MA
alex.bw@gmail.com
Joined on Dec 4, 2009

novocaine

- Novocaine needs callbacks

```
private lazy var audioManager:Novocaine? = {
        return Novocaine.audioManager()
    }()
private lazy var inputBuffer:CircularBuffer? = {
        return CircularBuffer.init(
                                numChannels: Int64(self.audioManager!.numInputChannels),
                                  andBufferSize: Int64(BUFFER_SIZE))
    }()
```

declare properties

setup manager and init buffer

```
@property (strong, nonatomic) Novocaine *audioManager;
@property (strong, nonatomic) CircularBuffer *buffer;

_audioManager = [Novocaine audioManager];
_buffer = [[CircularBuffer alloc]initWithNumChannels:1 andBufferSize:BUFFER_SIZE];
```

declare properties

setup audio and init buffer

the novocaine in/out block

microphone samples as float array

```
self.audioManager?.inputBlock = self.handleMicrophone

private func handleMicrophone (data:Optional<UnsafeMutablePointer<Float>>,
                               numFrames:UInt32,
                             numChannels: UInt32) {
        // copy samples from the microphone into circular buffer
        self.inputBuffer?.addNewFloatData(data, withNumSamples: Int64(numFrames))
    }

self.audioManager?.outputBlock = self.handleSpeakerQueryWithAudioFile

private func handleSpeakerQueryWithAudioFile(data:Optional<UnsafeMutablePointer<Float>>,
                                              numFrames:UInt32,
                                            numChannels: UInt32){
    self.outputBuffer?.fetchInterleavedData(data, withNumSamples:Int64(numFrames))
    }
```

data to write to speakers

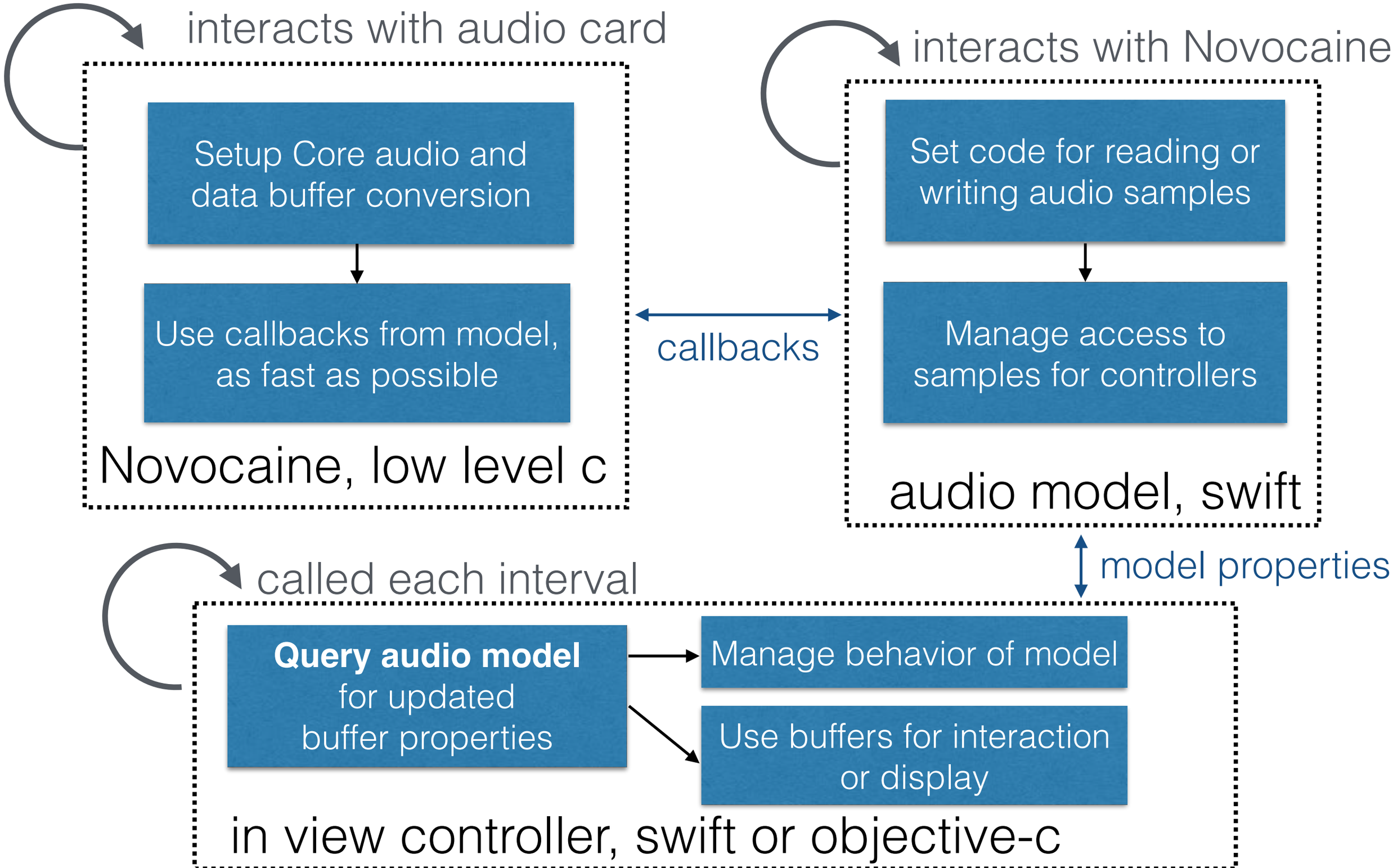
microphone samples as float array

```
[self.audioManager setInputBlock:^(float *data, UInt32 numFrames, UInt32 numChannels){

}]; [weakSelf.buffer addNewFloatData:data withNumSamples:numFrames];
```

data to write to speakers

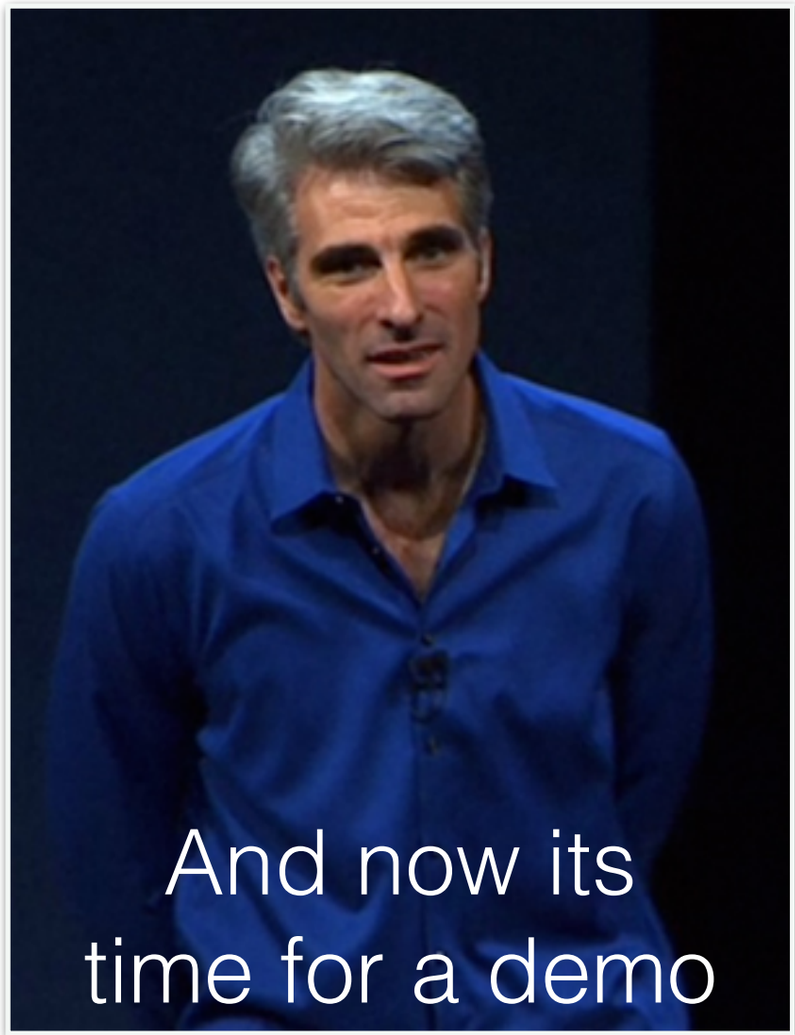
```
[self.audioManager setOutputBlock:^(float *data, UInt32 numFrames, UInt32 numChannels)
{

   [weakSelf.buffer fetchInterleavedData:data withNumSamples:numFrames];

}];
```

The program, using MVC



novocaine setup demo

source code on GitHub

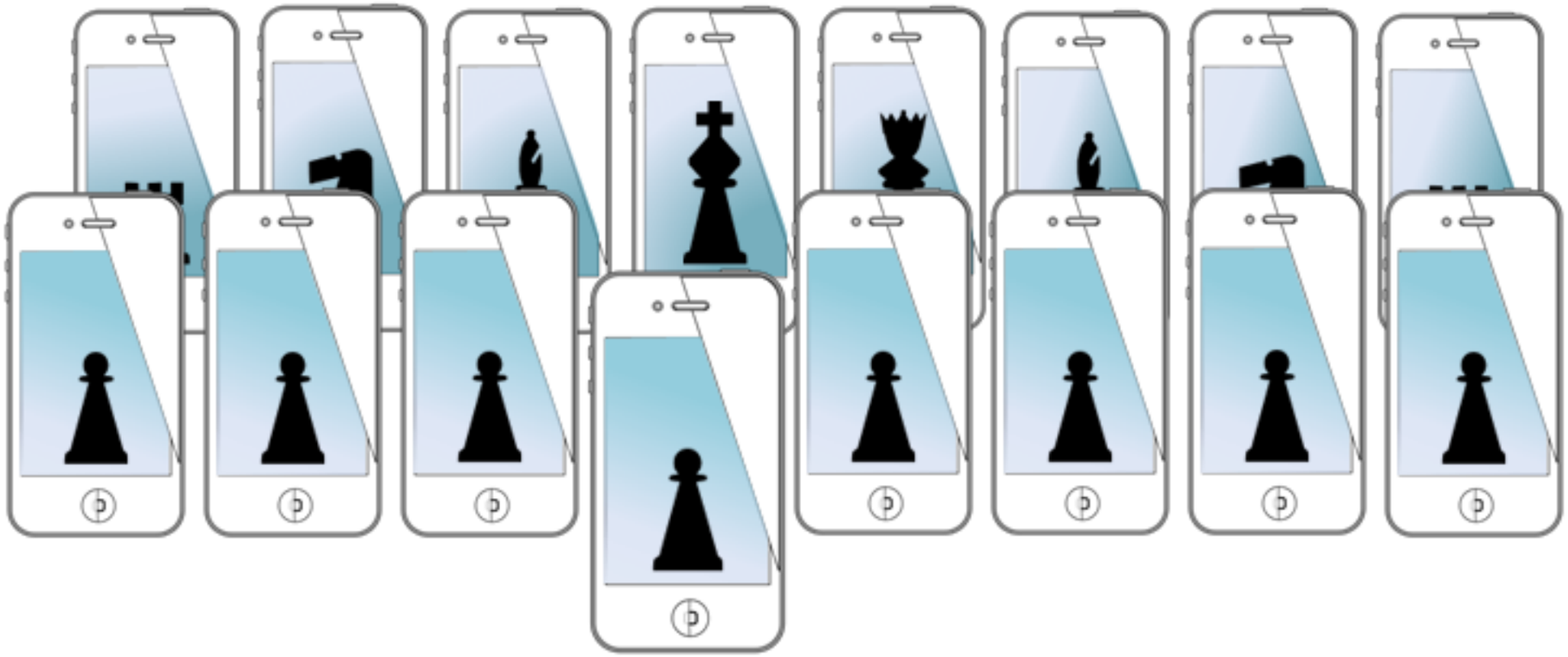


and rolling stones, if time

for next time…

- more core audio
  - playing songs (if not covered today)
  - getting samples from microphone
    - showing samples with Metal
  - working with sampled data
  - the accelerate framework

MOBILE SENSING LEARNING



CS5323 & 7323

Mobile Sensing and Learning

audio session

Eric C. Larson, Lyle School of Engineering,
Computer Science, Southern Methodist University