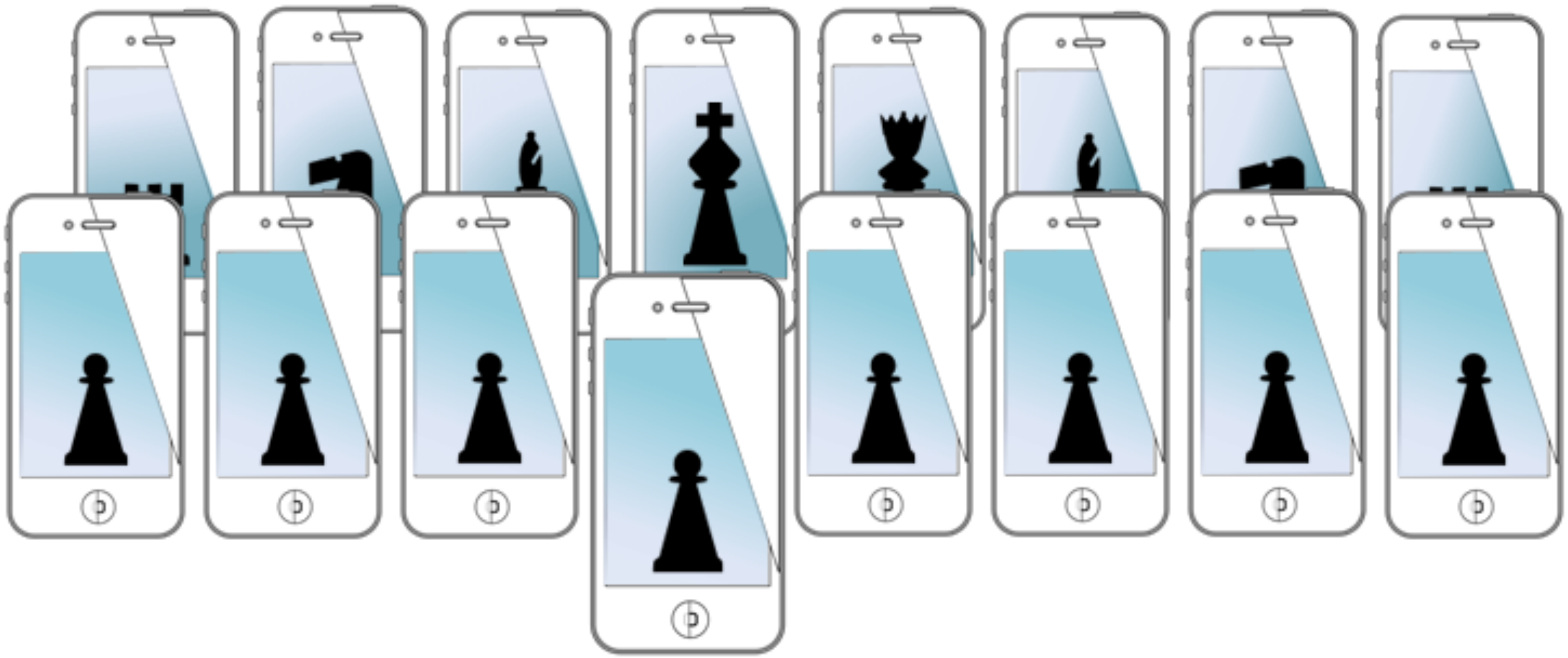
MOBILE SENSING LEARNING



CS5323 & 7323

Mobile Sensing and Learning

core audio

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

agenda

- core audio intro
- next lecture: more audio and graphing audio
- **next week**: second flipped module, the FFT

Core Audio

- many **audio packages** exist, but we want **low level signals**
- **Audio Sessions** (high level, setup audio hardware)
  - access the shared instance (for all applications)
  - access/set category (play, record, both)
      - choose options: like mixing with ambient sources
  - access/set audio route (mostly we will ignore this)
    - set specific hardware within audio route
  - **Audio Units** (once hardware is ready handle the input/output)
    - set stream format, buffer sizes, sampling rate,
    - initialize memory for audio buffers
    - set callback rendering procedure

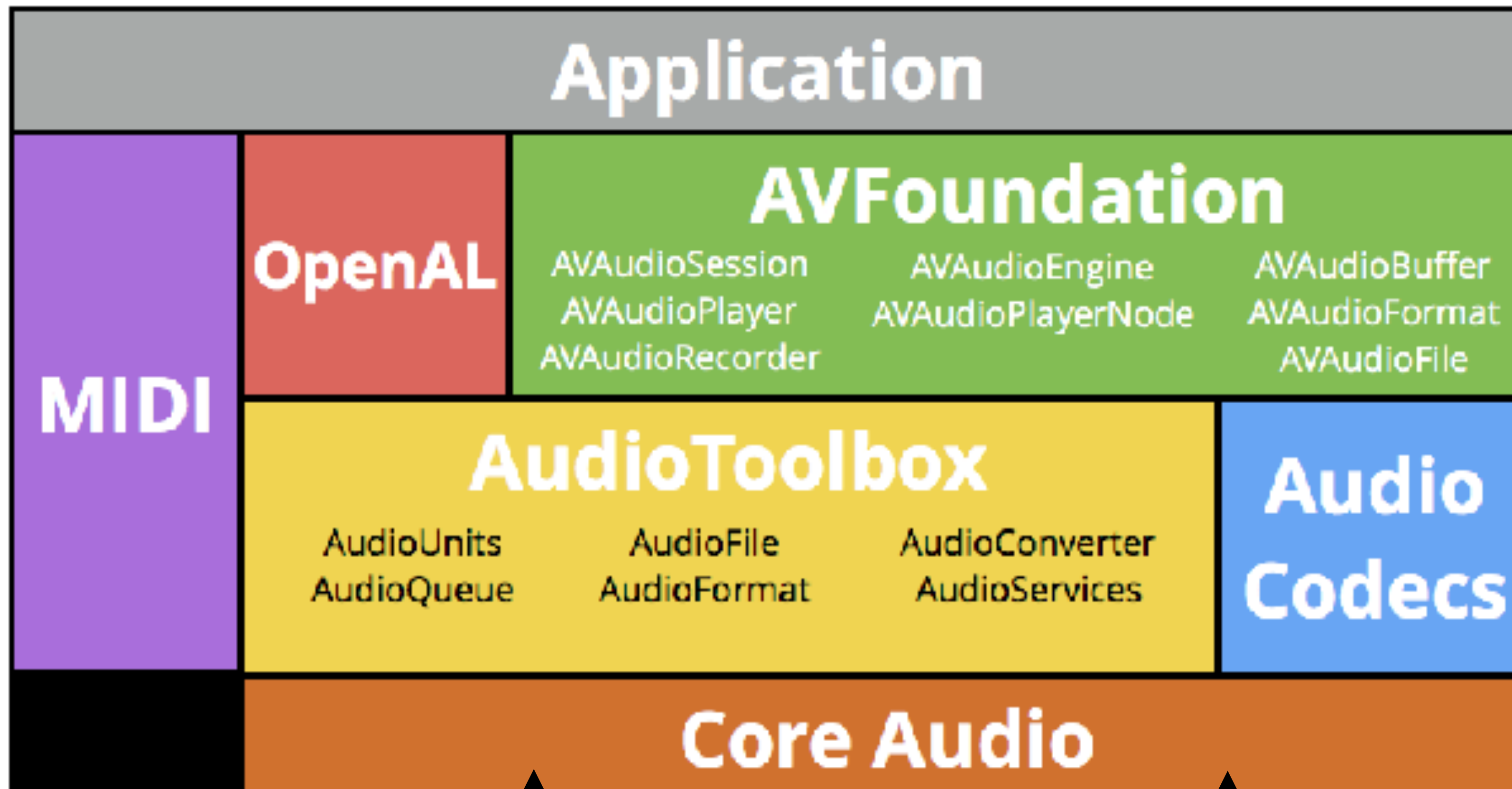
audio sessions

iOS

audio session management

request

background audio



request

request

song from iTunes lib

audio from microphone

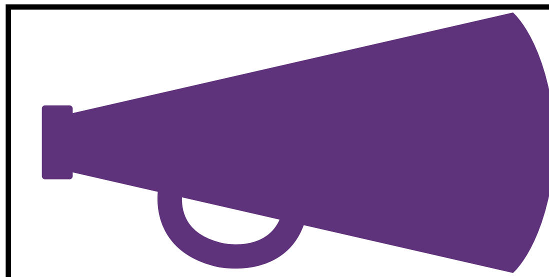
any request can alter management of other audio requests

mixing can be automatic — impressive!

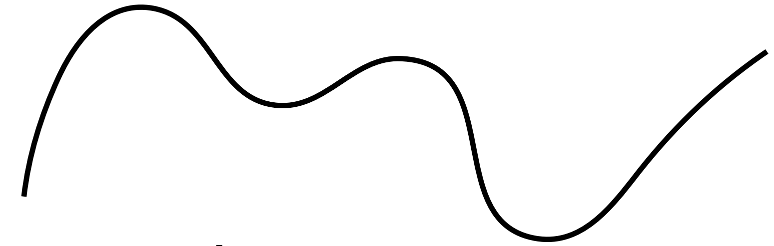
audio hardware

sound

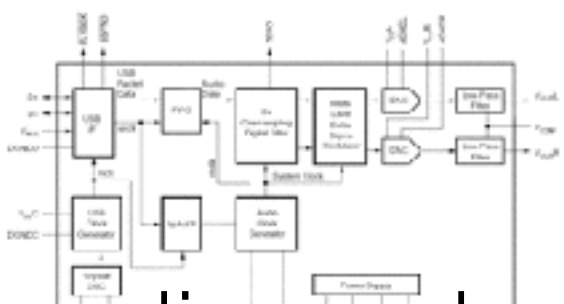
mechanical movement



amplifier

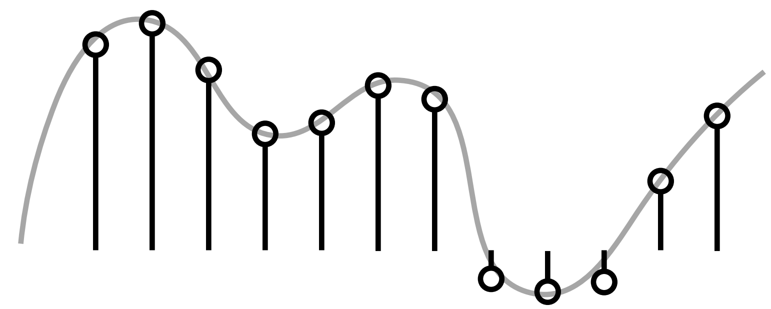


voltage



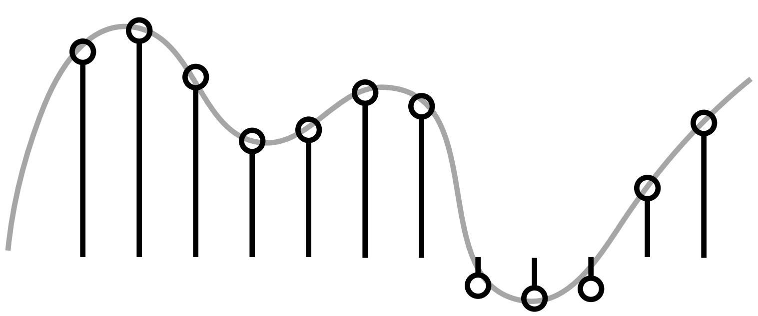
audio card

buffer and ADC

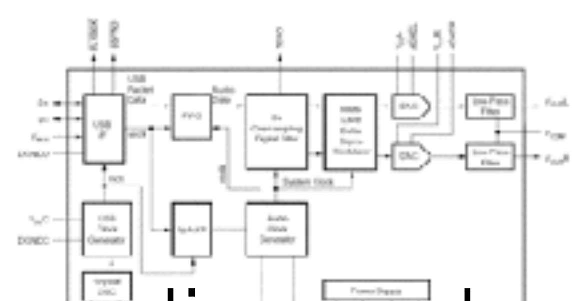


array of audio

notify software a buffer is ready

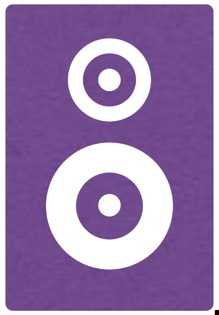


array of audio to play



audio card

buffer and DAC

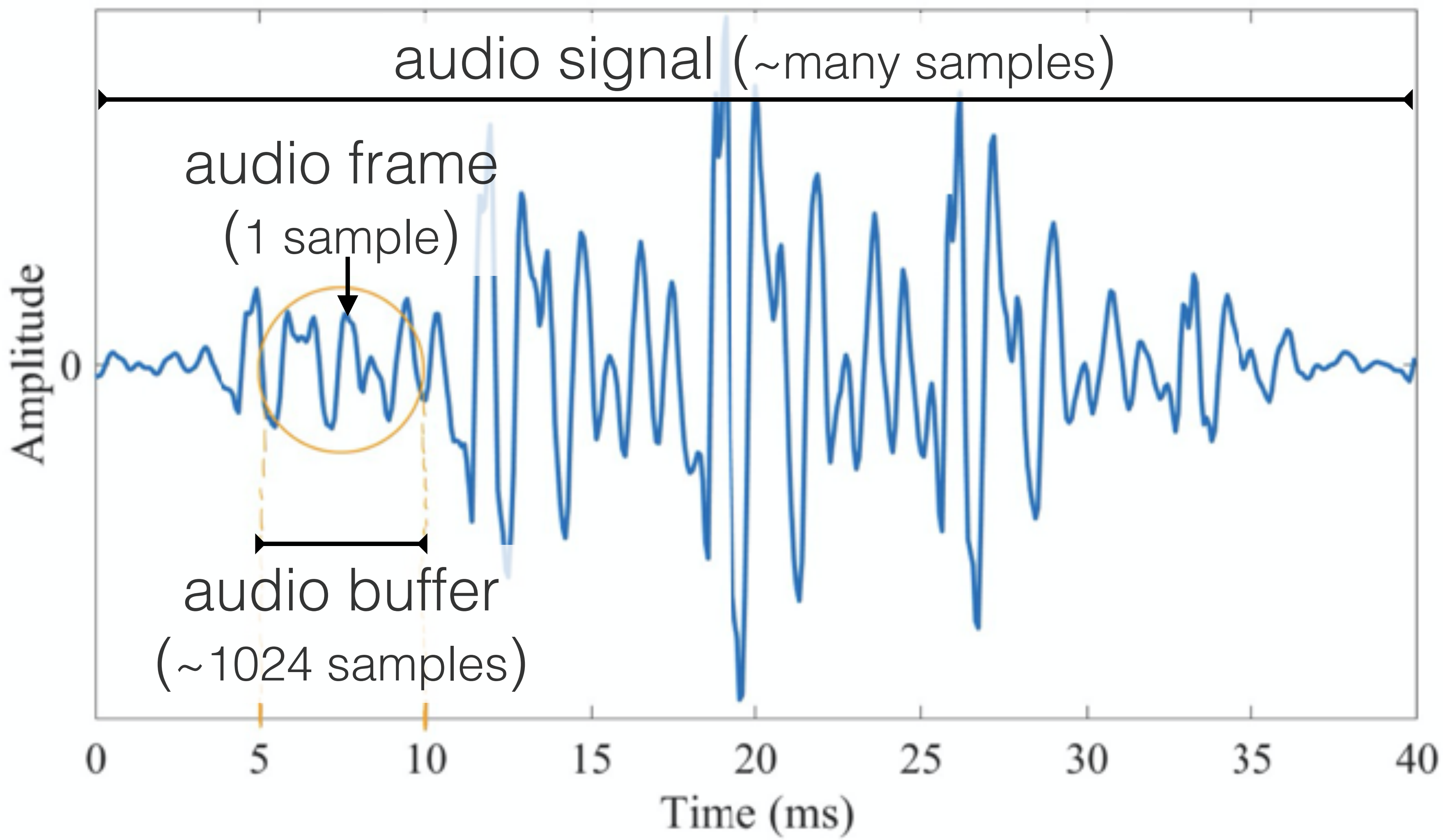


speaker

sound

audio buffering

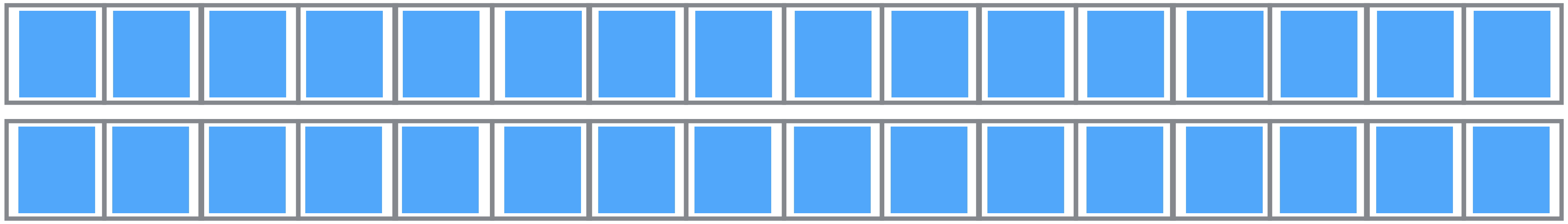
- audio card buffers up audio samples before sending to the main processor



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

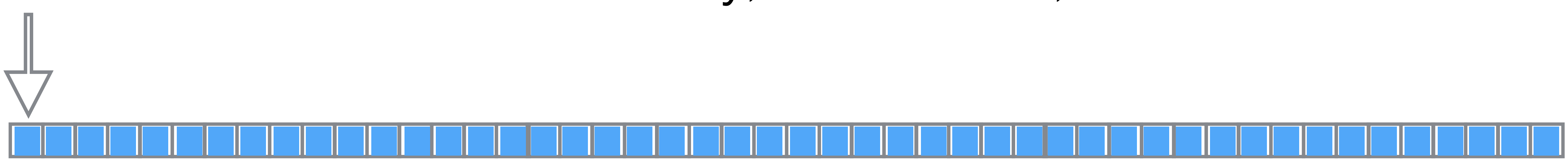
CPU

(memory in RAM)

copy over samples, convert

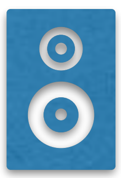
exit from call as soon as possible!

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

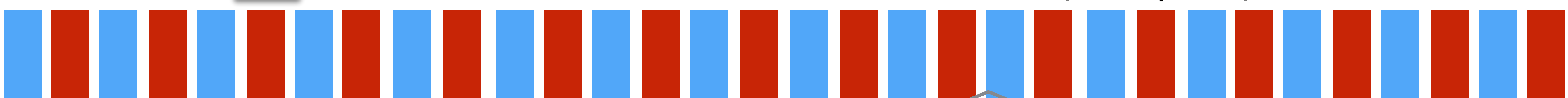


audio unit formats

microphone (input)

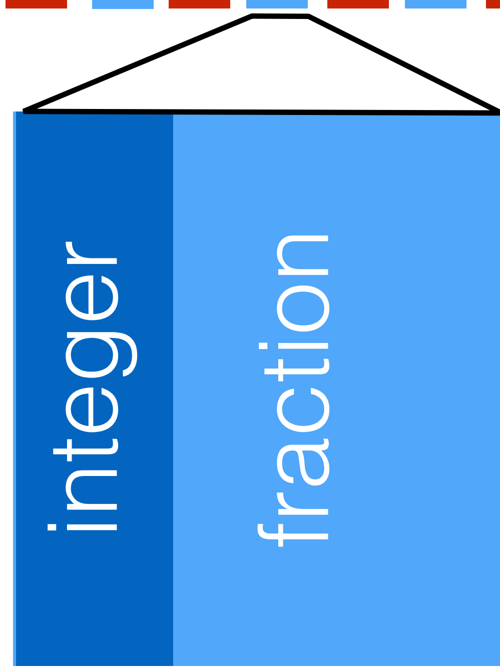


stereo could be interleaved (output)



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 nodes 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
```

…some APIs need the AVAudioBuffer Object…

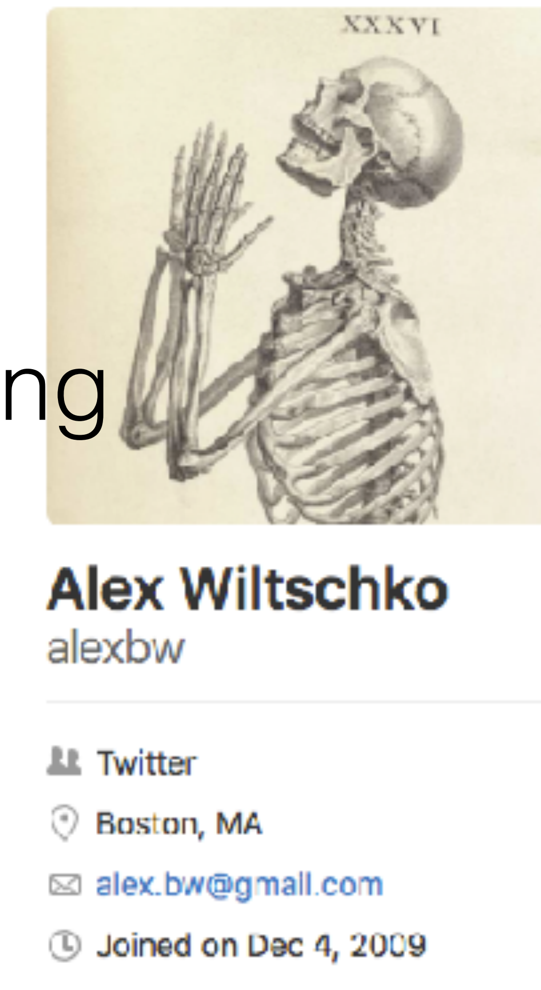
```
}

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

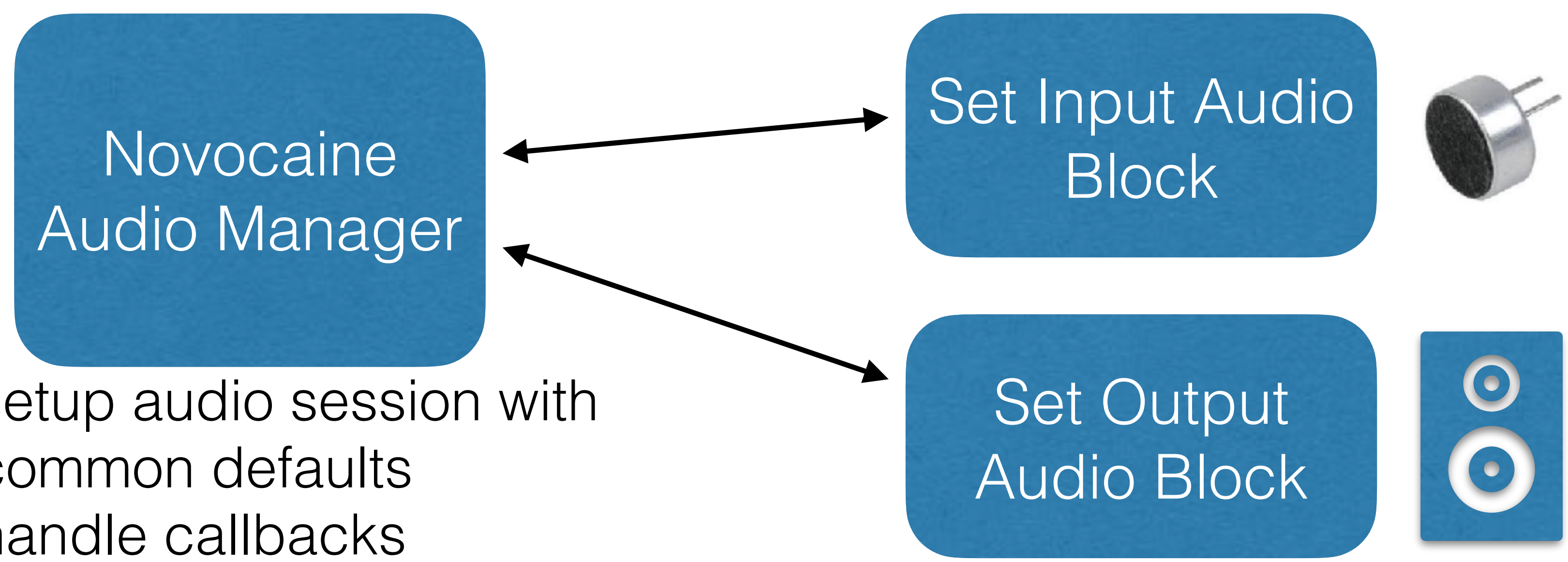
Get Samples

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**





- setup audio session with common defaults
- handle callbacks
- handle data conversion

novocaine

- Novocaine needs callbacks

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

declare properties and setup manager

init circular buffer (written for you)

```
@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

input

```
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.circBuffer?.addNewFloatData(data, withNumSamples: Int64(numFrames))
    }
```

setup function to give novocaine for callback

microphone samples as float array

output

```
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

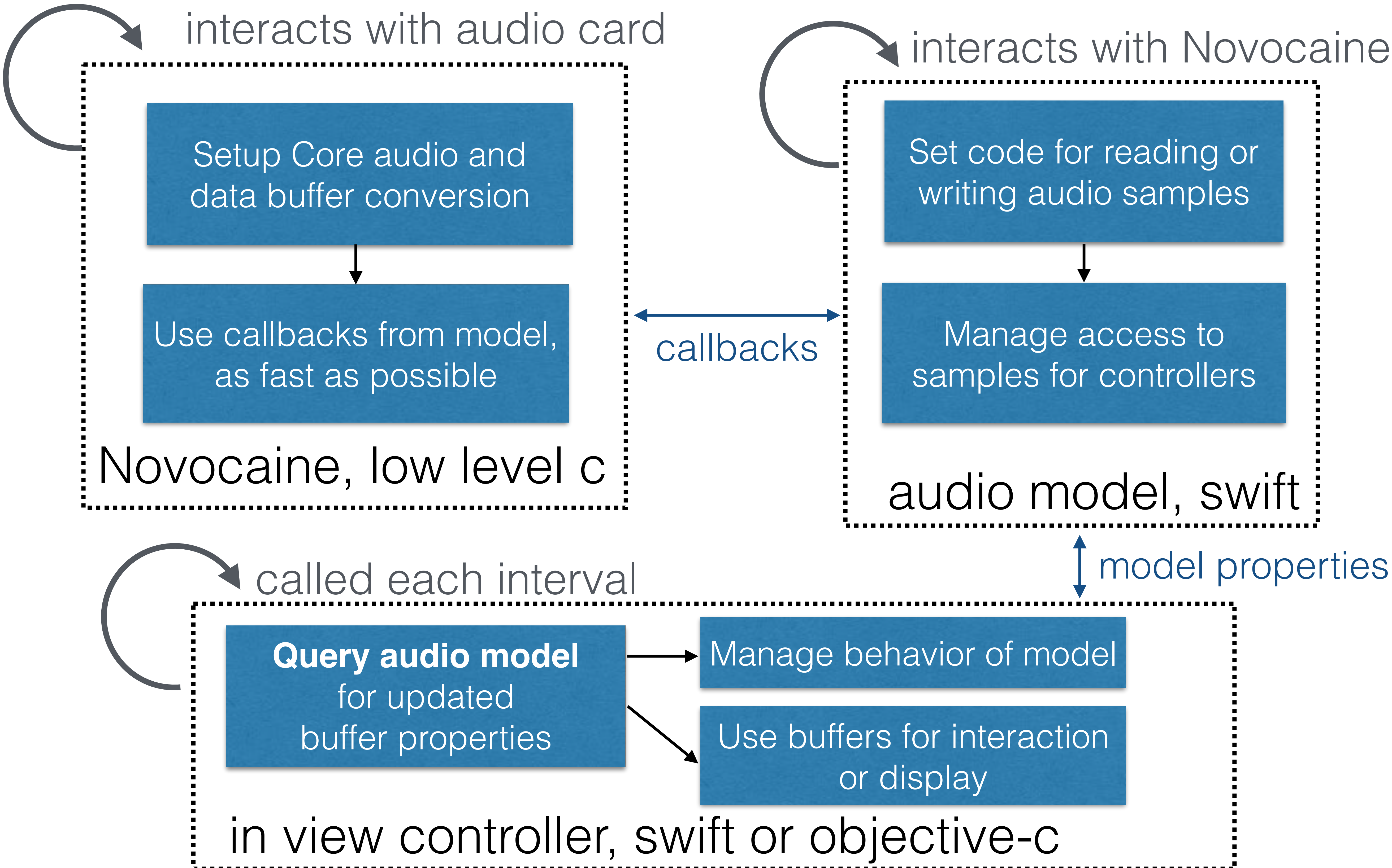
microphone samples as float array

```
[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];

 }];
```

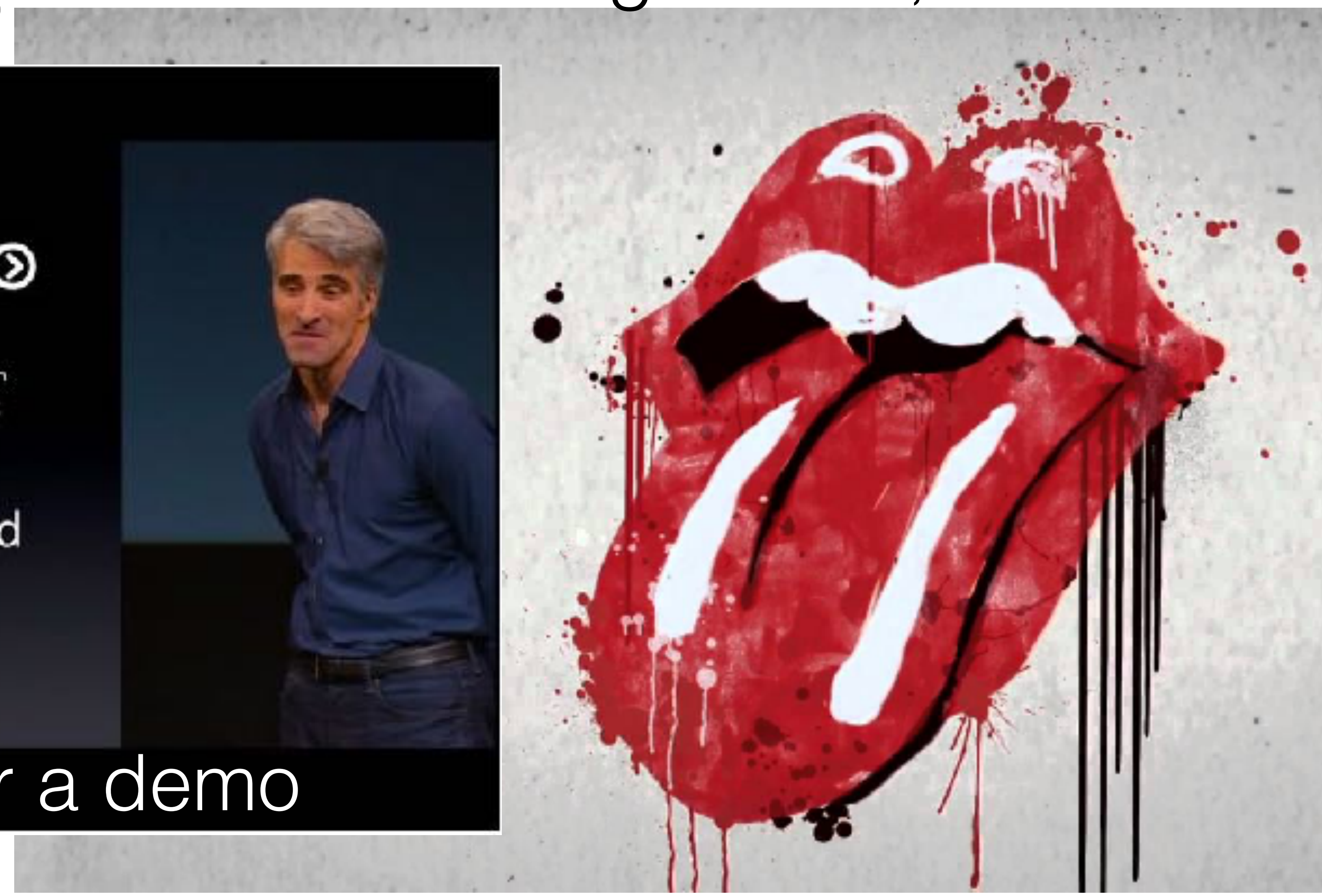
The program, using MVC



novocaine setup demo

source code on GitHub

rolling stones, if time



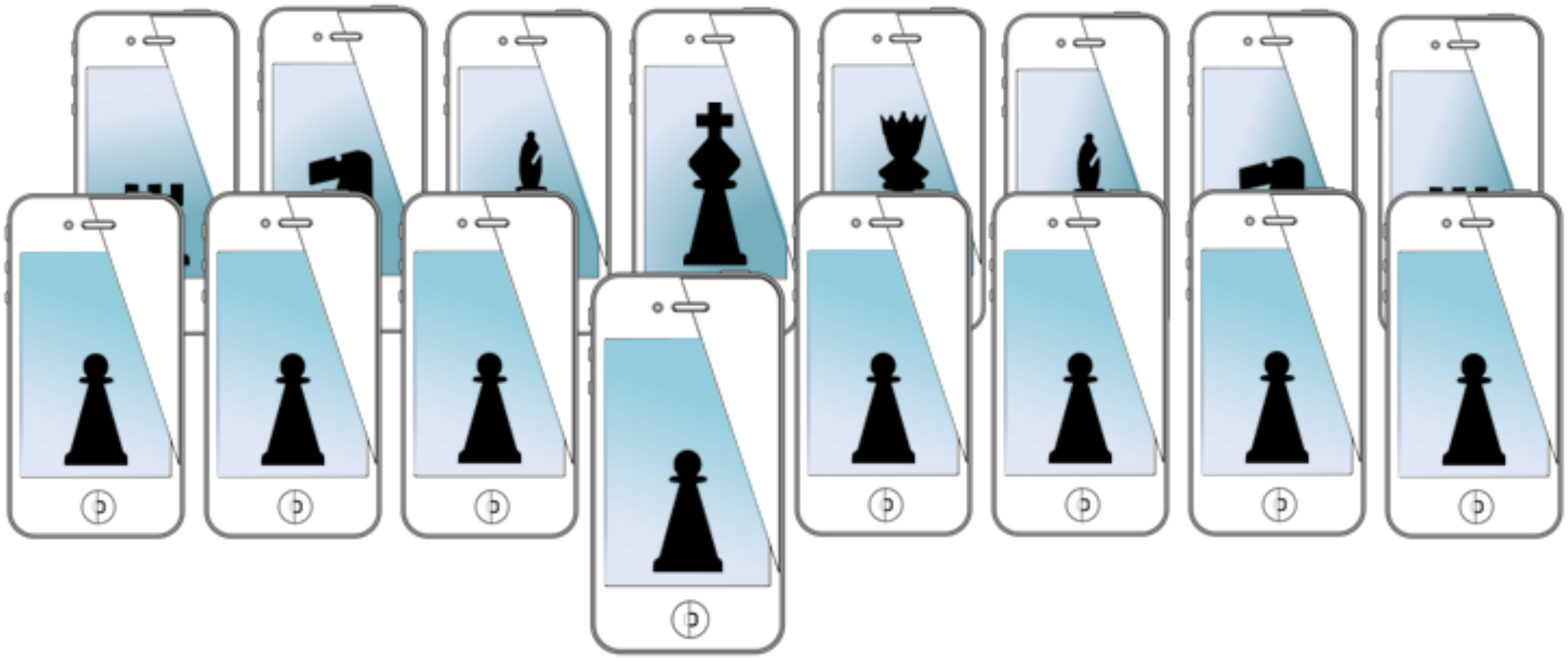
Declare in info.plist

Privacy - Microphone Usage Description	String	This App requires microphone access.

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