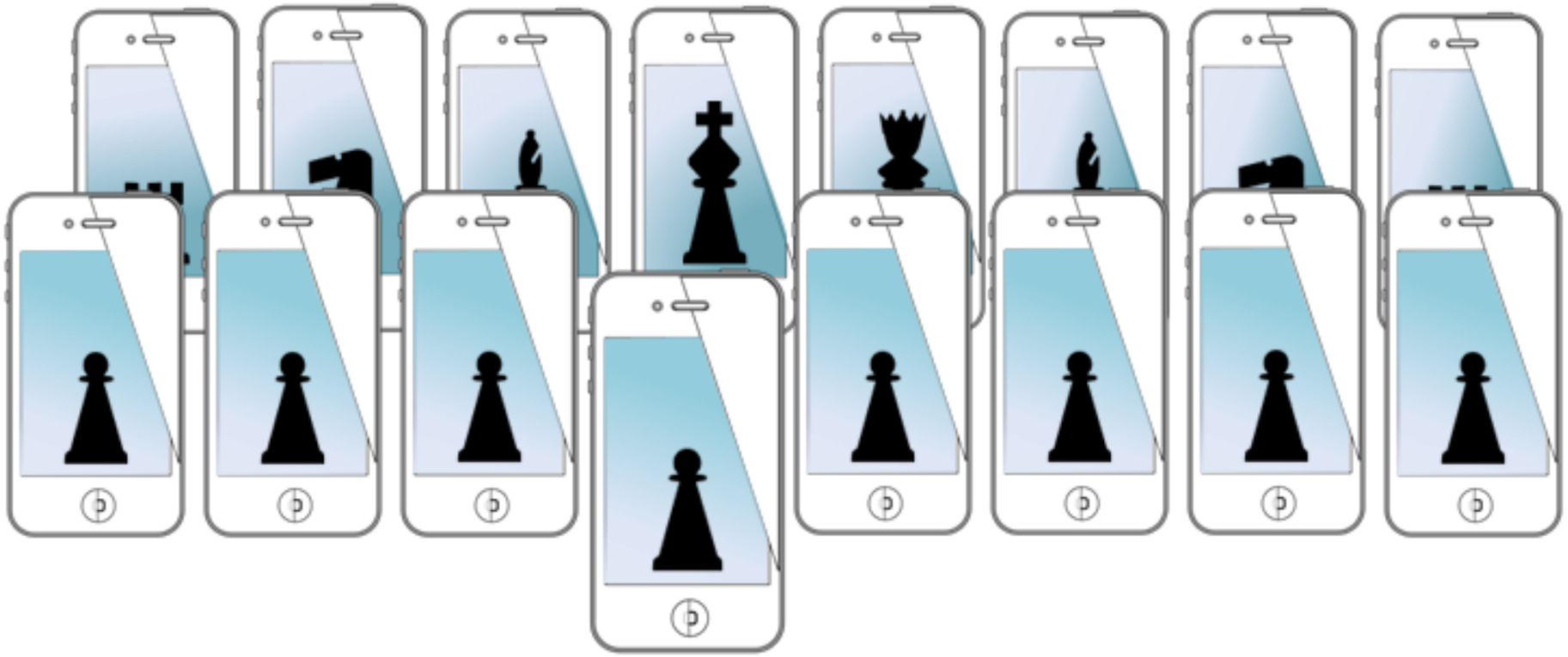
MOBILE SENSING LEARNING



CSE5323 & 7323

Mobile Sensing and Learning

week 3, lecture a: queues, blocks, c++, audio session

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

agenda

- blocks and multi-threading
- objective c++ (no longer needed!)
- core audio intro

blocks

- not callback functions (but similar)
  - created at runtime
  - can access data from scope when defined
  - syntax is ^(…)
- not exactly a lambda (*but similar*)
  - but it acts like an object that can be passed as an argument or created on the fly
- also used in swift (called closures)

block syntax

return type

block name

param types

```
//create a block on the fly
float (^onTheFlyBlockThatAddsTwoInts)(int,int); // declare the block, try not to make unclear
// define the behavior of the block
onTheFlyBlockThatAddsTwoInts =^(int a, int b){
    return (float)(a+b);
};
// use the block
NSLog(@" On the fly value: %.4f",onTheFlyBlockThatAddsTwoInts(5,6));
```

define code that will execute

```
typedef float(^TypeDefinedBlock)(float,float);

TypeDefinedBlock blockAsObject = ^(float arg1, float arg2){
    return arg1 / arg2;
};

    //-------------------------------
    //execute the block from typedef
    float value = blockAsObject(22.0,44.0);
    NSLog(@" Val = %.4f",value);
```

type define, more like callback

syntax to call block

```
    //-------------------------------
    //enumerate an Array with a block
    NSArray *myArray = @[@34.5,@56.4567,@(M_PI)];

    // 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]);
    }];
```

enumerate with block

some semantics

- variables from same scope where block is defined are **read only**

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

- Unless you use keyword:

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

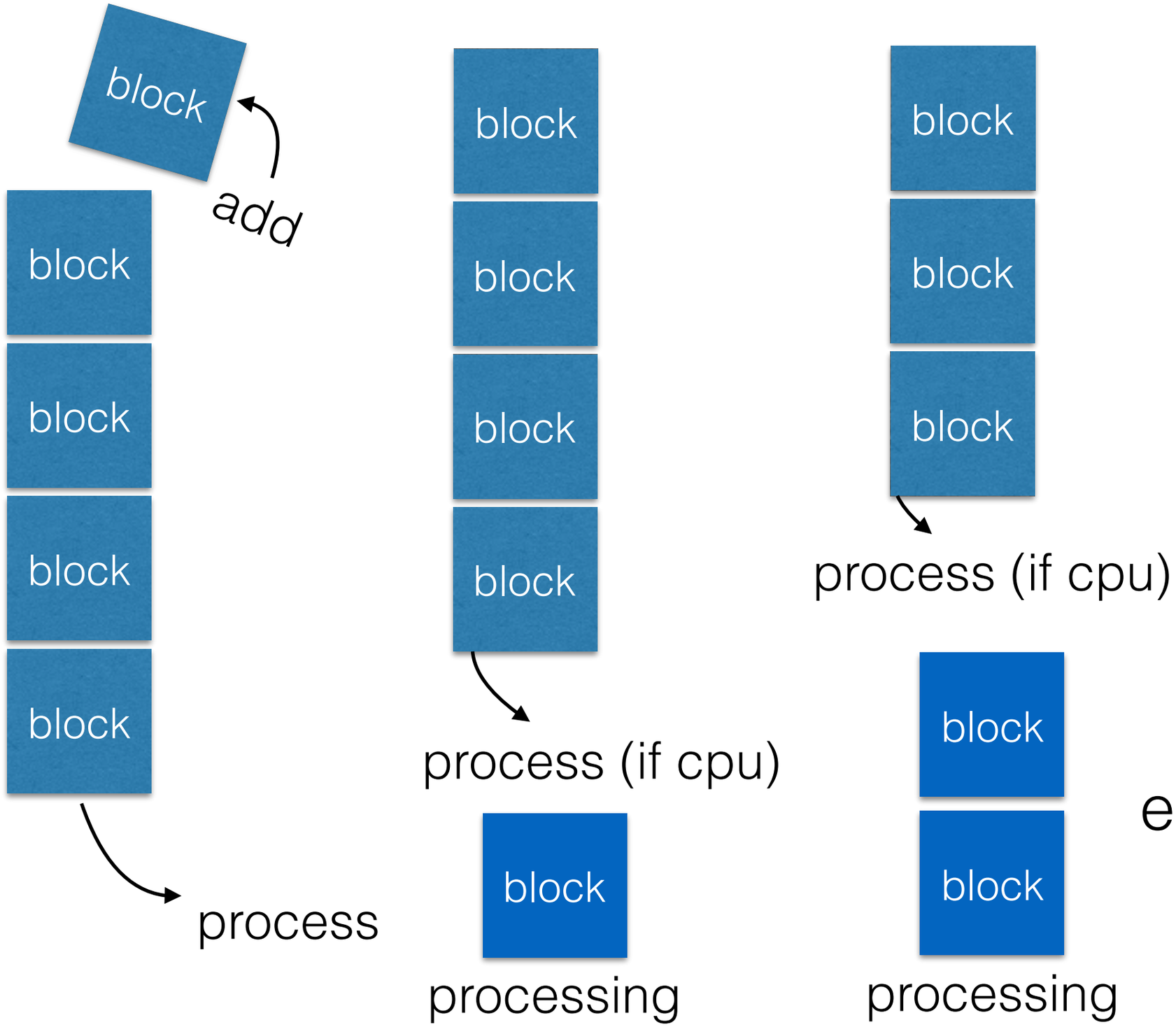
- classes hold a **strong** pointer to blocks they use
- blocks hold a **strong** pointer to __block variables
  - so using “self” would create a retain cycle

```
self.value = (some function in block)

__block ViewController * __weak  weakSelf = self;

weakSelf.value = (some function in block)
```

concurrency in iOS

- grand central dispatch (GCD) handles all operations
  - GCD looks at “queues” of **blocks** that need to be run
  - GCD and the Xcode compiler work deep inside the OS, actually in the kernel — they are optimized
  - for a **serial queue** each block is run sequentially
  - for **concurrent queues** the first block is dequeued
    - if CPU is available, then the next block is also dequeued, but could finish any time
- the **main queue handles all UI operations** (and no other queue should generate UI changes!!)
  - so, **no updating of** the views, labels, buttons, (image views*) **except from the main queue**

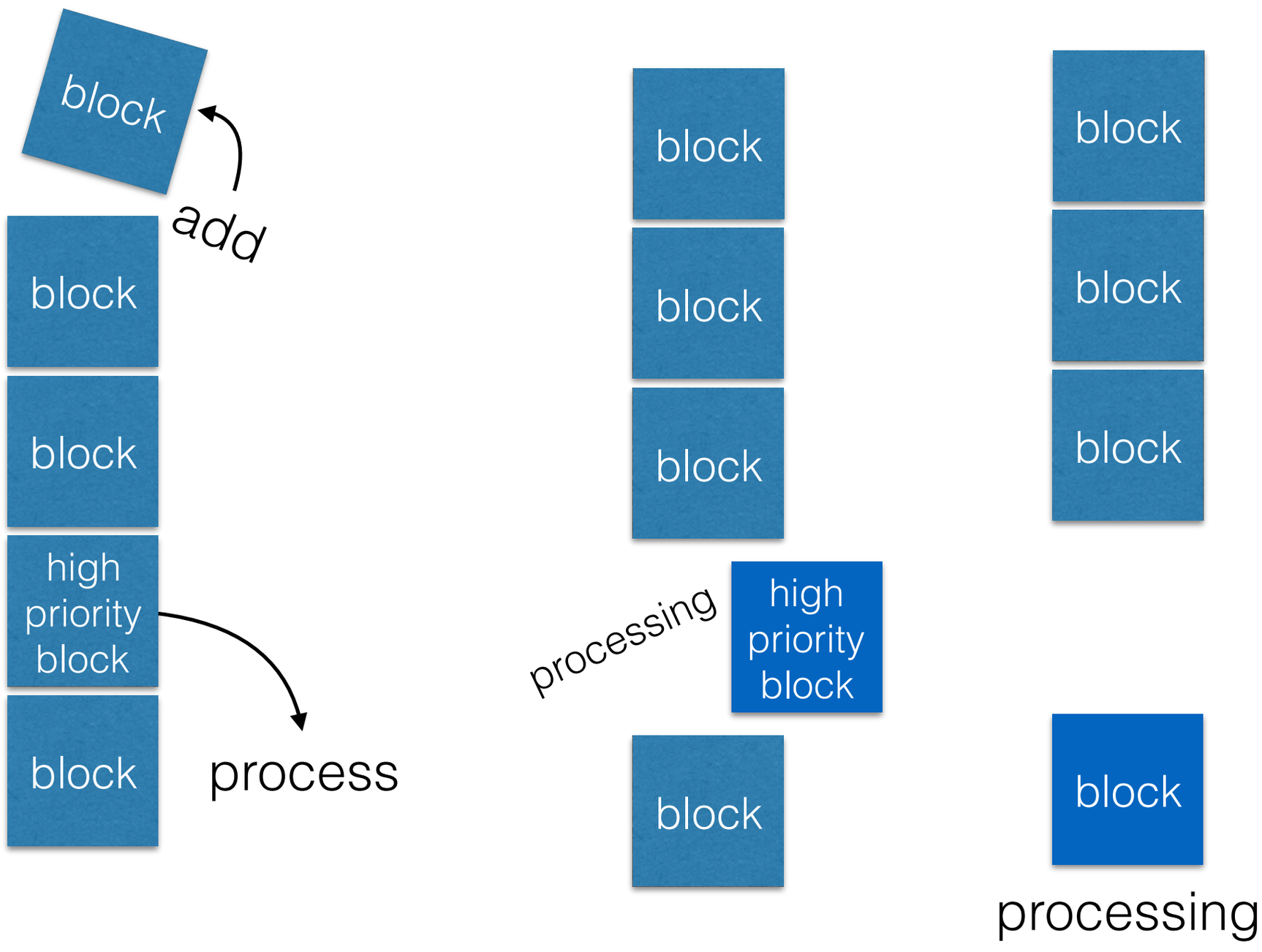
concurrent queues



threads?

either could finish

the main queue



one thread
UI is in sync

queue syntax

create new queue

```
// using c code:
 dispatch_queue_t someQueue = dispatch_queue_create("myCreatedQueue", DISPATCH_QUEUE_CONCURRENT );
 dispatch_async(someQueue, ^{
     // your code to execute
     for(int i=0;i<3;i++)
         NSLog(@"I am being executed from a dispatched queue");

     // now I need to set something in the UI, but I am not in the main thread!
     // call from main thread
     dispatch_async(dispatch_get_main_queue(), ^{
         self.label.text = [NSString stringWithFormat:@"Finished running %d times, Safe",3];
     });


 }); // this operation adds the block to the queue in a single clock cycle, then returns


NSOperationQueue *newQueue = [[NSOperationQueue alloc] init];
    newQueue.name = @"ObjCQueue";
    [newQueue addOperationWithBlock:^{
        // your code to execute
        for(int i=0;i<3;i++)
            NSLog(@"I am being executed from a dispatched queue, from objective-c");

        // now I need to set something in the UI, but I am not in the main thread!
        // call from main thread
        [self performSelectorOnMainThread:@selector(setMyLabel)
                                withObject:nil
                             waitUntilDone:NO];


    }];
```

define block

serial or concurrent

update UI, main thread

create new queue

define block

update UI, main thread

queue syntax

- using global queues

access a global queue

```
// An example of using already available queues from GCD
dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0), ^{
    // your code to execute
    for(int i=0;i<3;i++)
        NSLog(@"I am being executed from a global concurrent queue");

    // now I need to set something in the UI, but must get in the main thread!

    // call from main thread
    dispatch_async(dispatch_get_main_queue(), ^{
        self.label.text = @"Finished running from GCD global";
    });
```

not on main queue!!

main queue!

```
DISPATCH_QUEUE_PRIORITY_LOW
DISPATCH_QUEUE_PRIORITY_DEFAULT
DISPATCH_QUEUE_PRIORITY_HIGH
DISPATCH_QUEUE_PRIORITY_BACKGROUND
```

objective c++

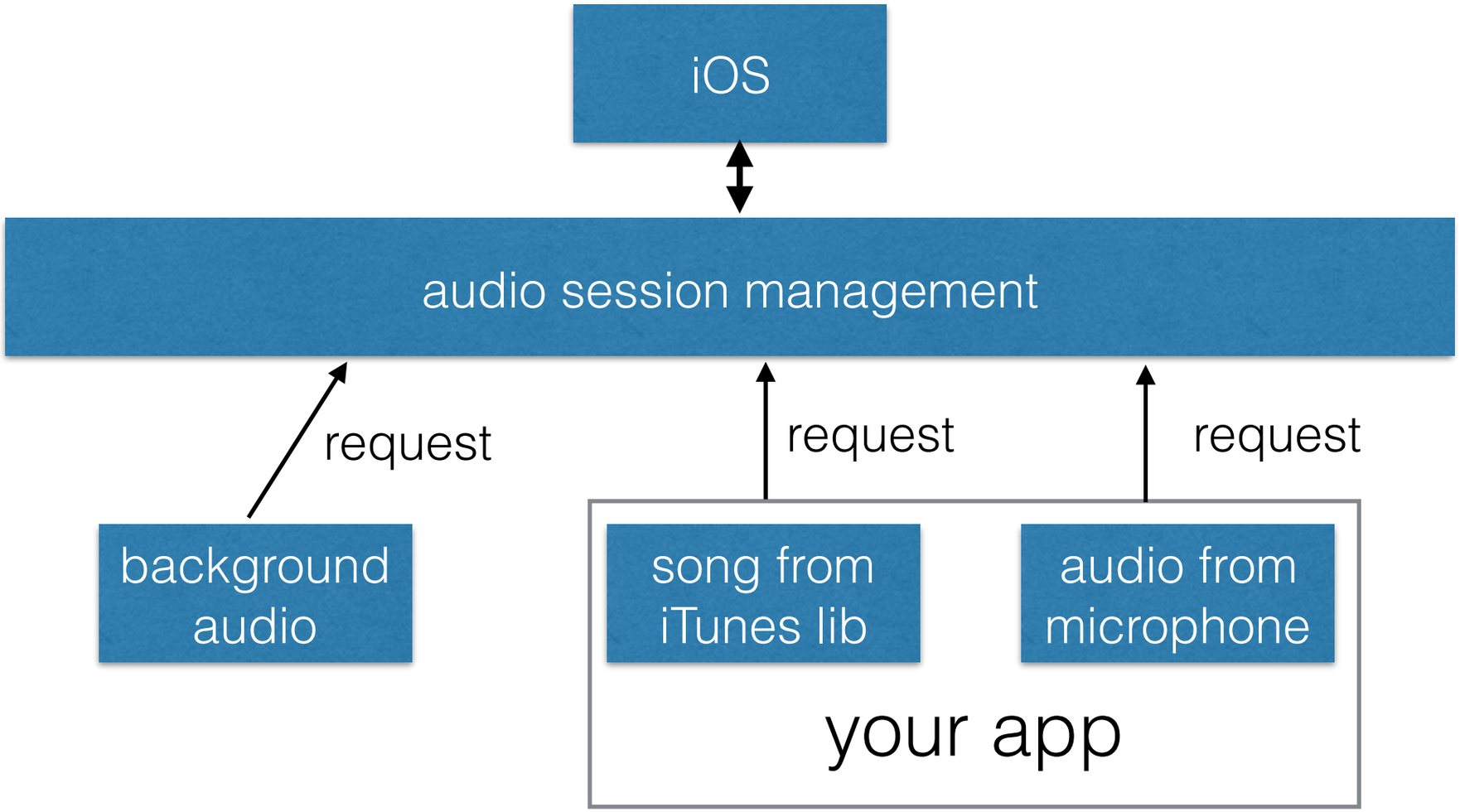
- this is no longer required for audio, but is good to know for later in the class…
- actually, its just c++
- …but need to tell compiler we are using c++

- add any **#include** statements
- change **extensions to .mm** where you use c++ class(es)
- ARC won't help you for *malloc*, *calloc*, or ***new***
  - so explicitly call dealloc and your class destructor
  - ***delete*** or *free*

Core Audio

- **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

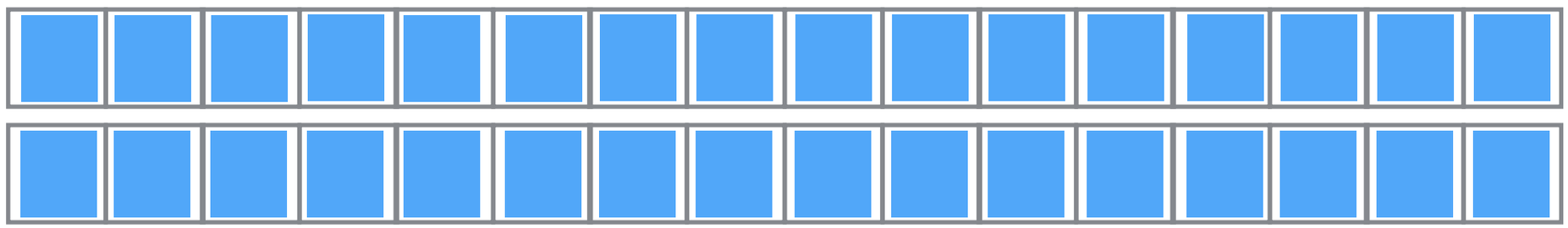


any request can alter management of other audio requests

mixing can be automatic — this is impressive

audio units

audio input buffer procedure, double buffer shown



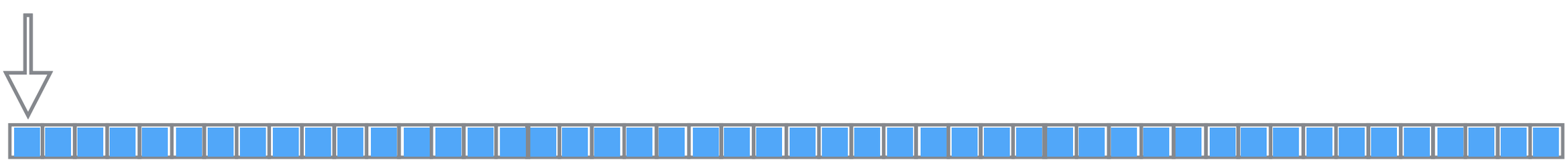
c

sent to audio session callback

copy over samples, convert

exit from call as soon as possible!

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



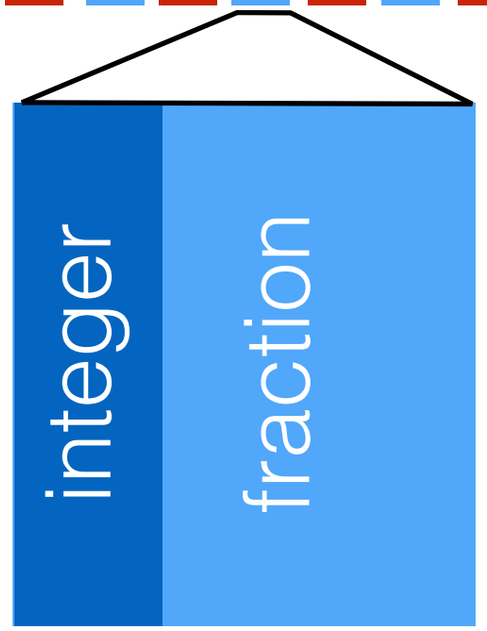
audio unit formats

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

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

214 Followers | 507 Starred | 327 Following

audio made easy

- Novocaine (I mean real easy!!)

declare properties

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

setup audio and init buffer

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

microphone samples as float array

```
-(void)viewWillAppear:(BOOL)animated{

    __block ViewController * __weak  weakSelf = self;
    [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];
    }];
```

novocaine setup demo

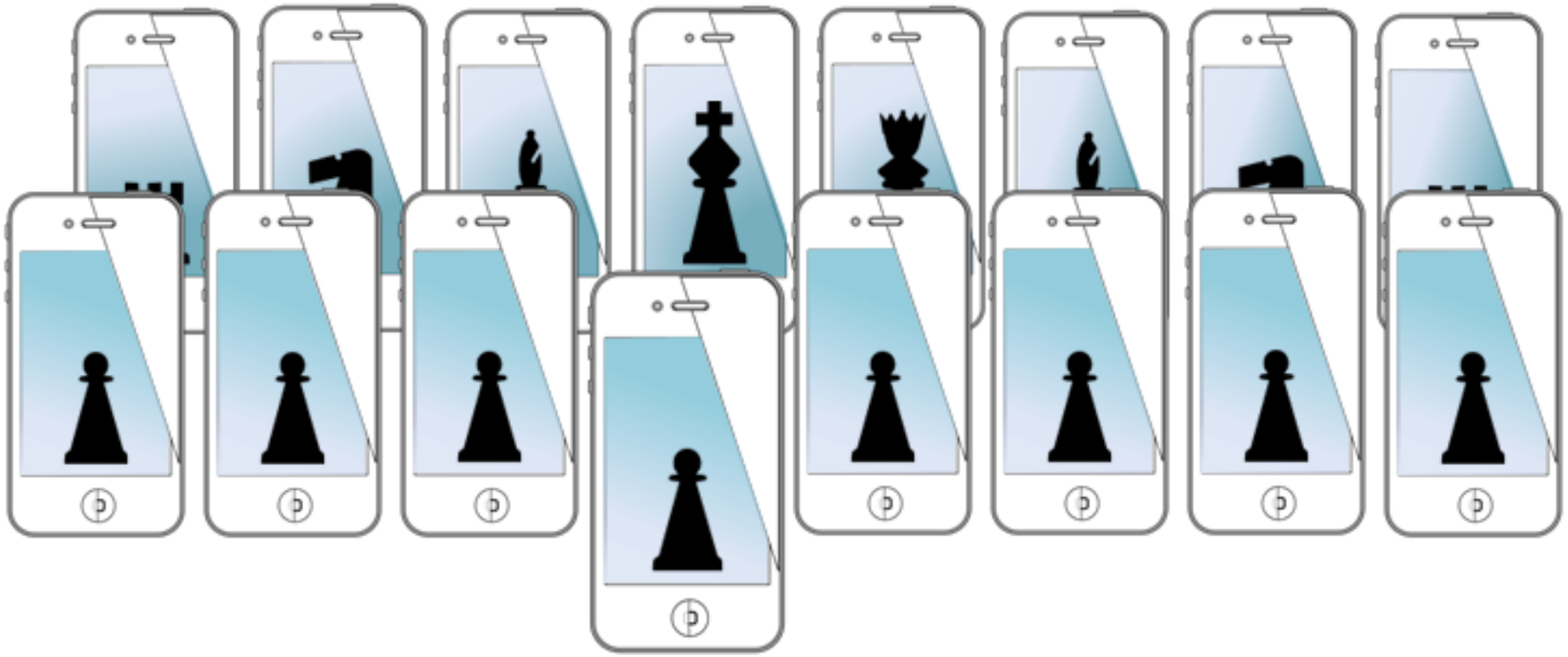
source code on GitHub, **uses submodules**

and rolling stones, if time

for next time...

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

MOBILE SENSING LEARNING



CSE5323 & 7323

Mobile Sensing and Learning

week 3, lecture a: queues, blocks, c++, audio session

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