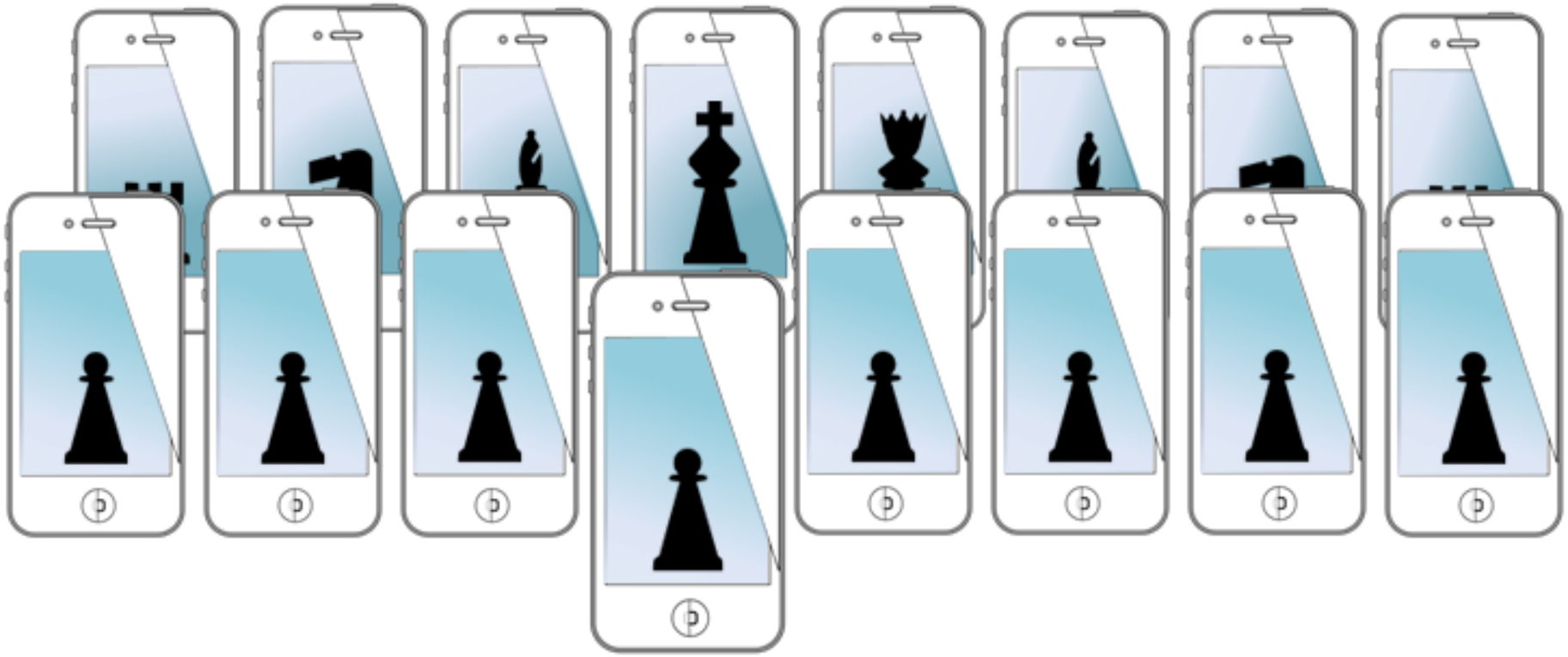
MOBILE SENSING LEARNING & CONTROL



CSE5323 & 7323

Mobile Sensing, Learning, and Control

lecture four: page controllers & core data

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

course logistics

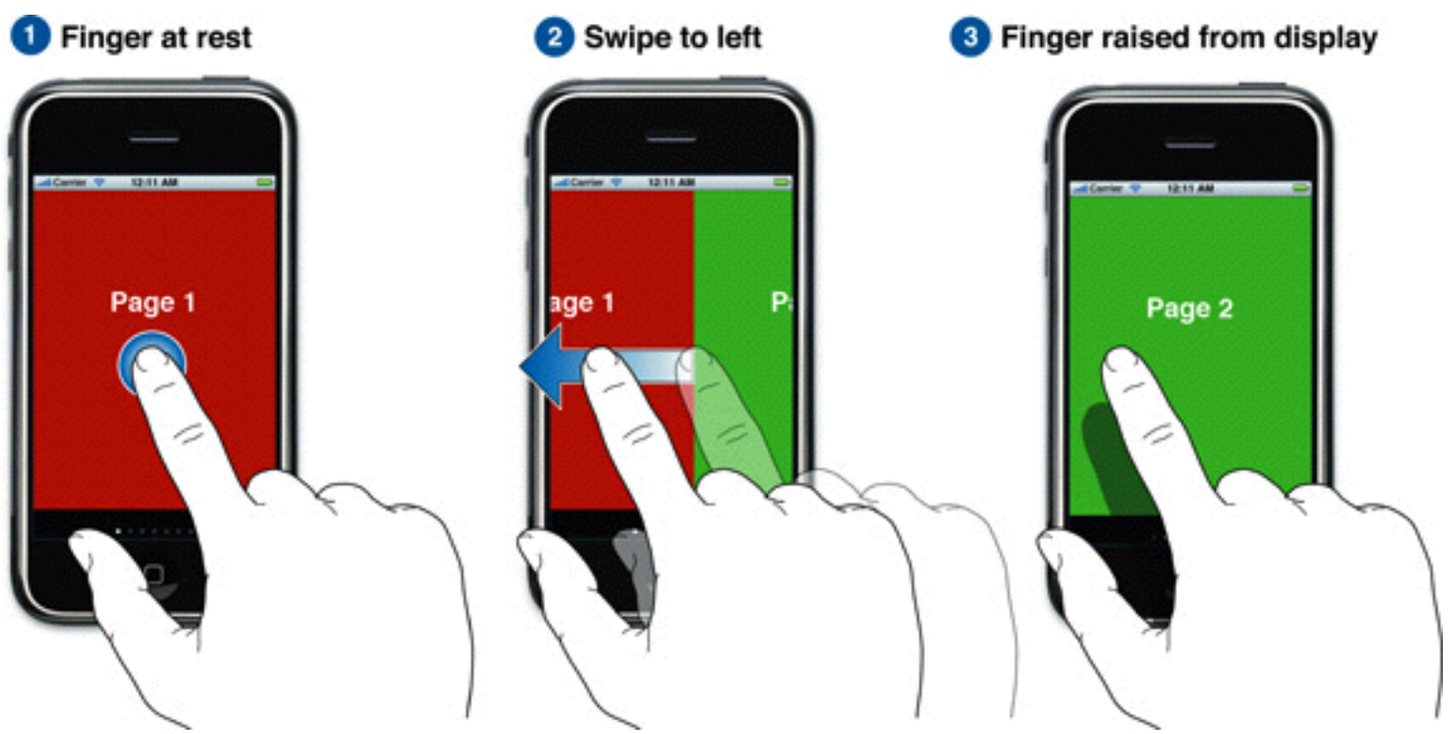
- A1 due this Friday
- I have 20 people on teams, and 2 unassigned
    - correct?

agenda

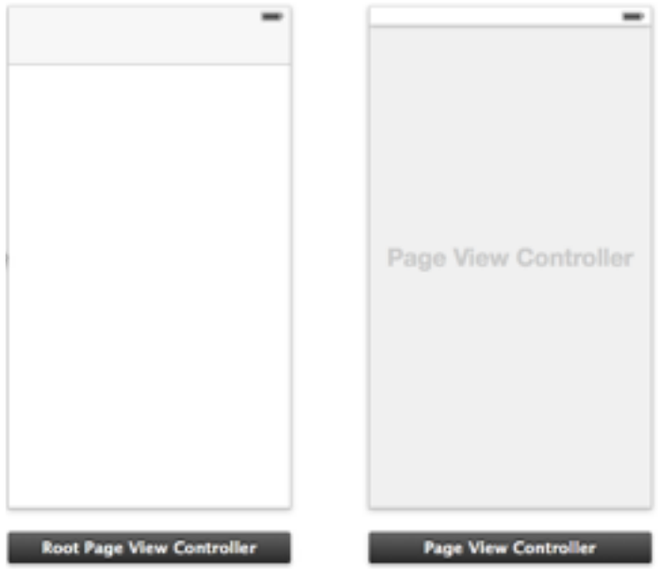
- page view controllers
- timers / segmented control
- persistent storage
  - core data for creating and using database schema
- blocks and multi-threading
- objective c++

page view controller

- place UIPageViewController in storyboard
- place a “root controller” for the page
  - adopt <UIPageViewControllerDataSource>
  - instantiate pageViewController from “root”
  - instantiate views to be paged in “root”

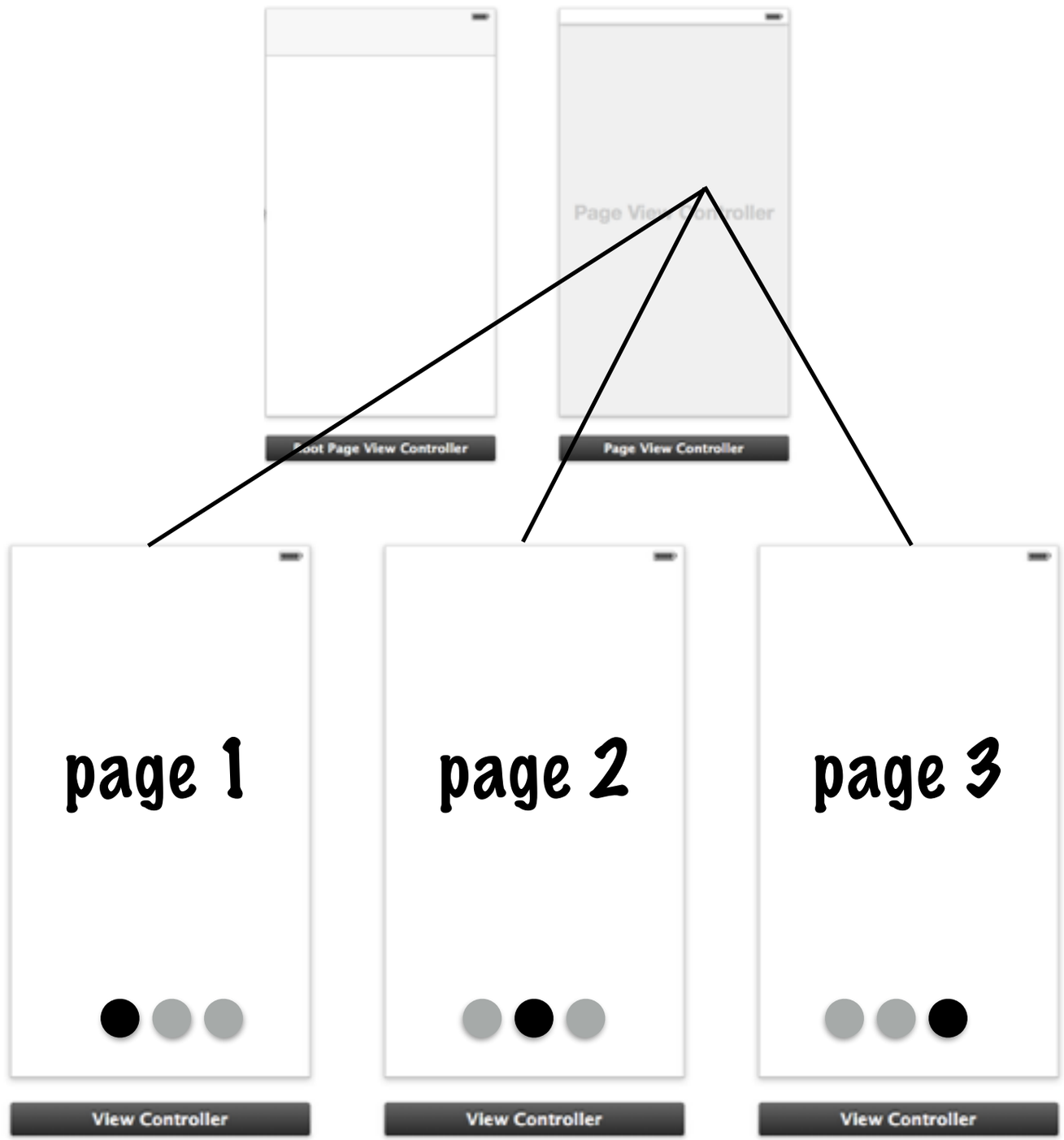


page view controller



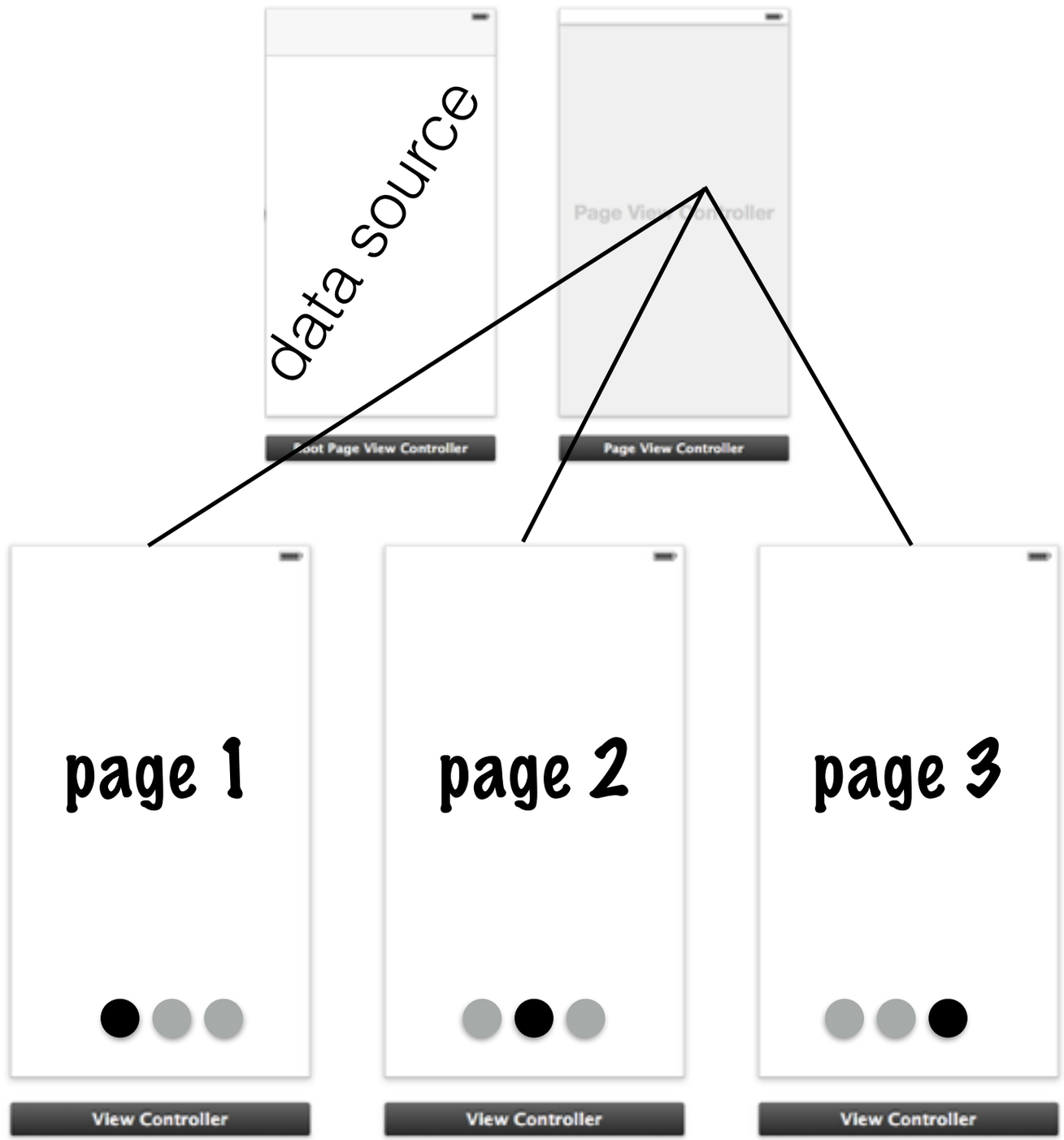
different instantiations of view controller

page view controller



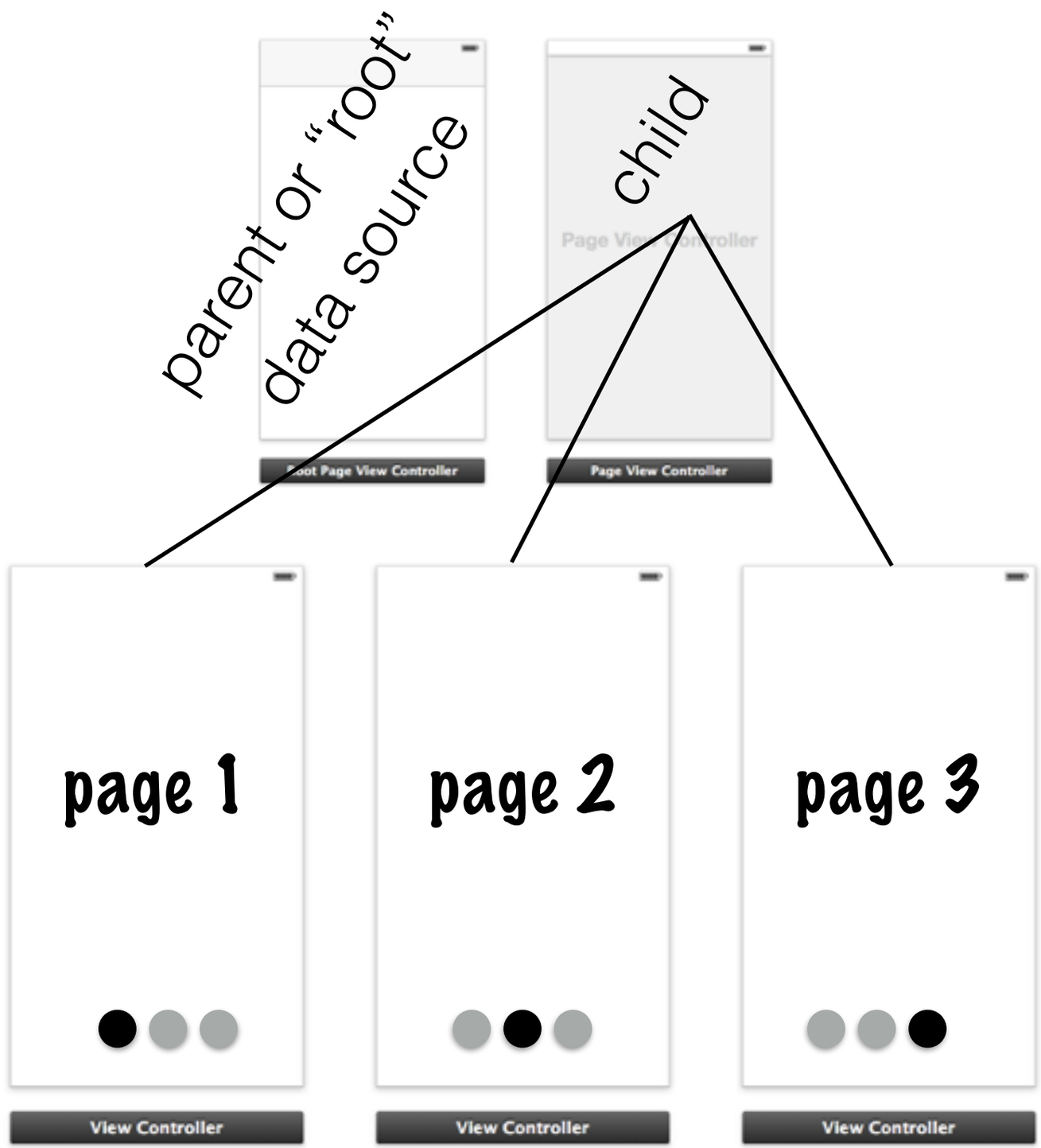
different instantiations of view controller

page view controller



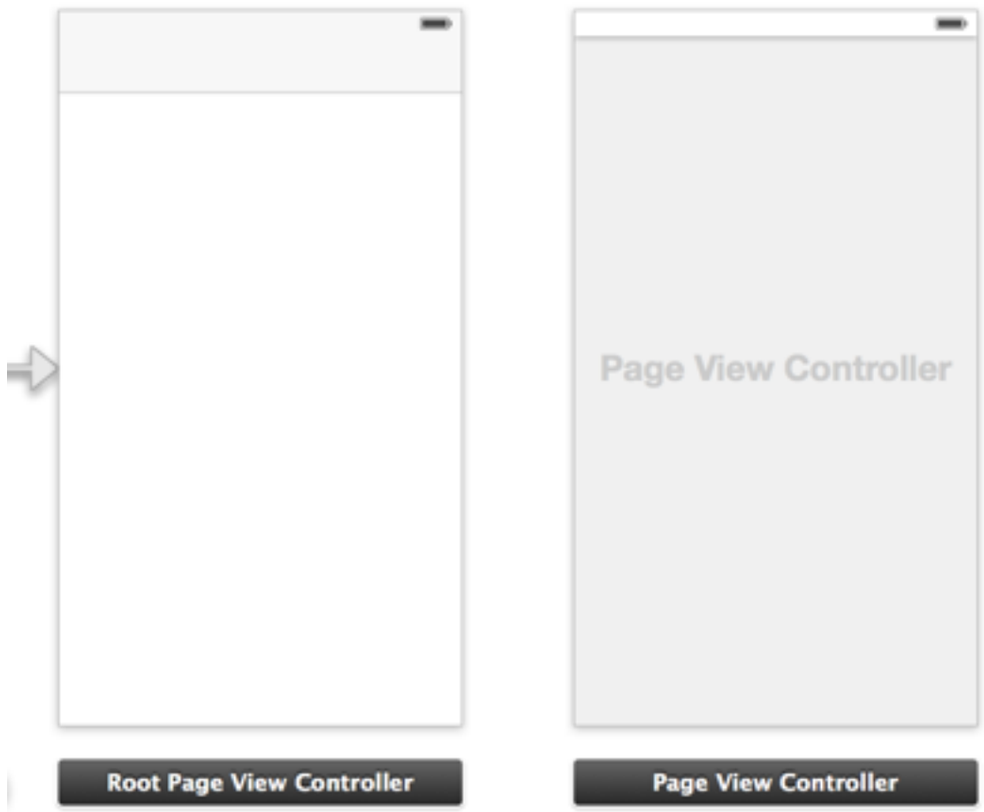
different instantiations of view controller

page view controller

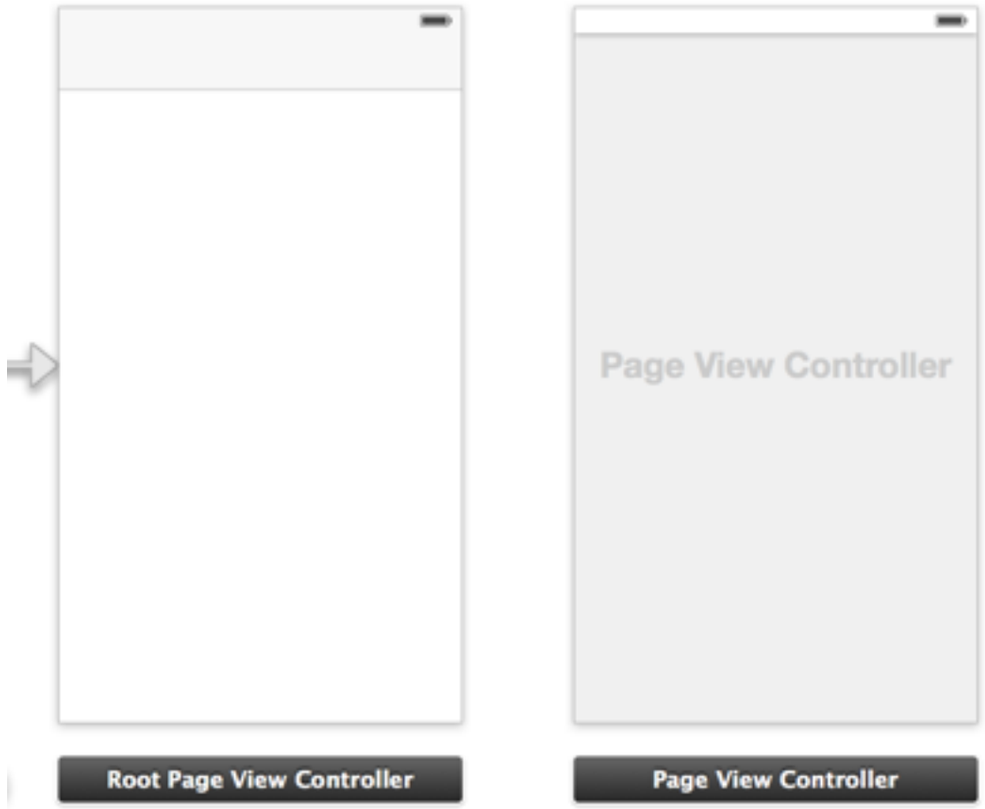


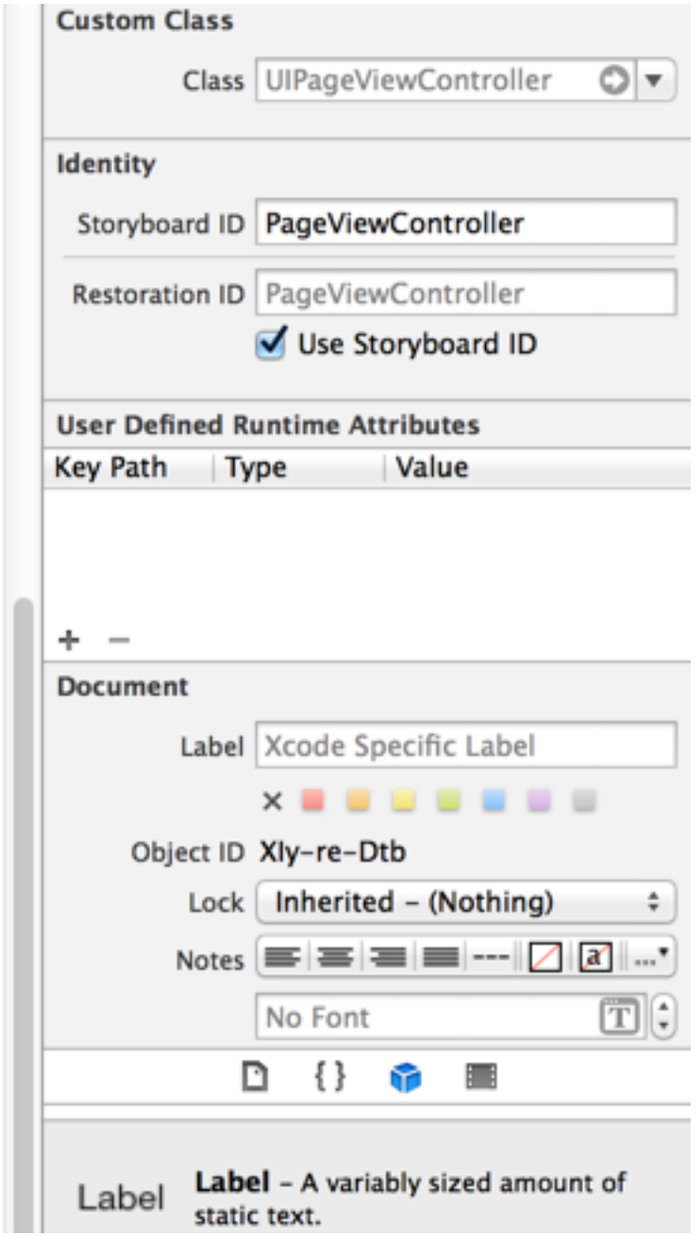
different instantiations of view controller

page view controller

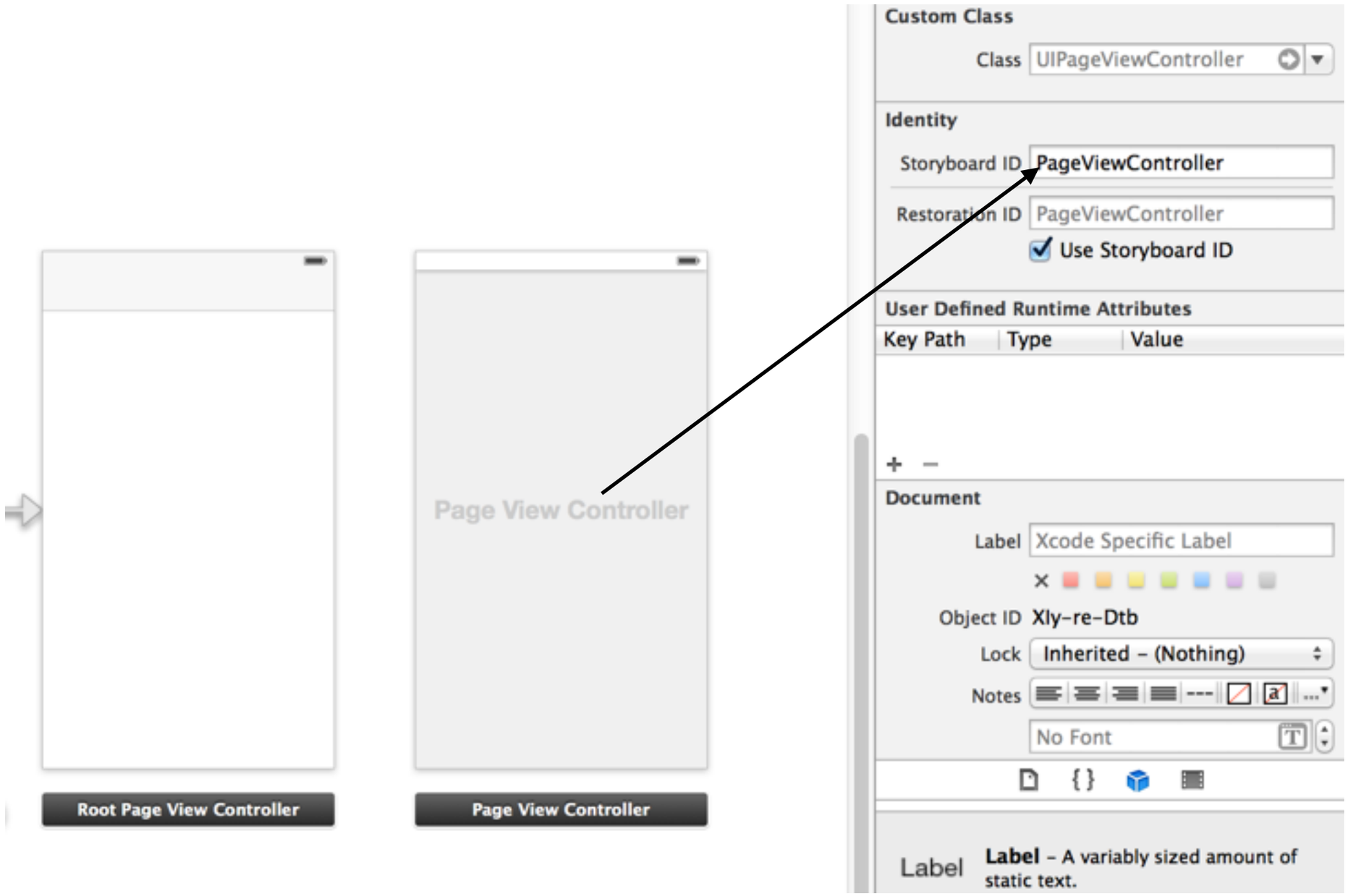


page view controller



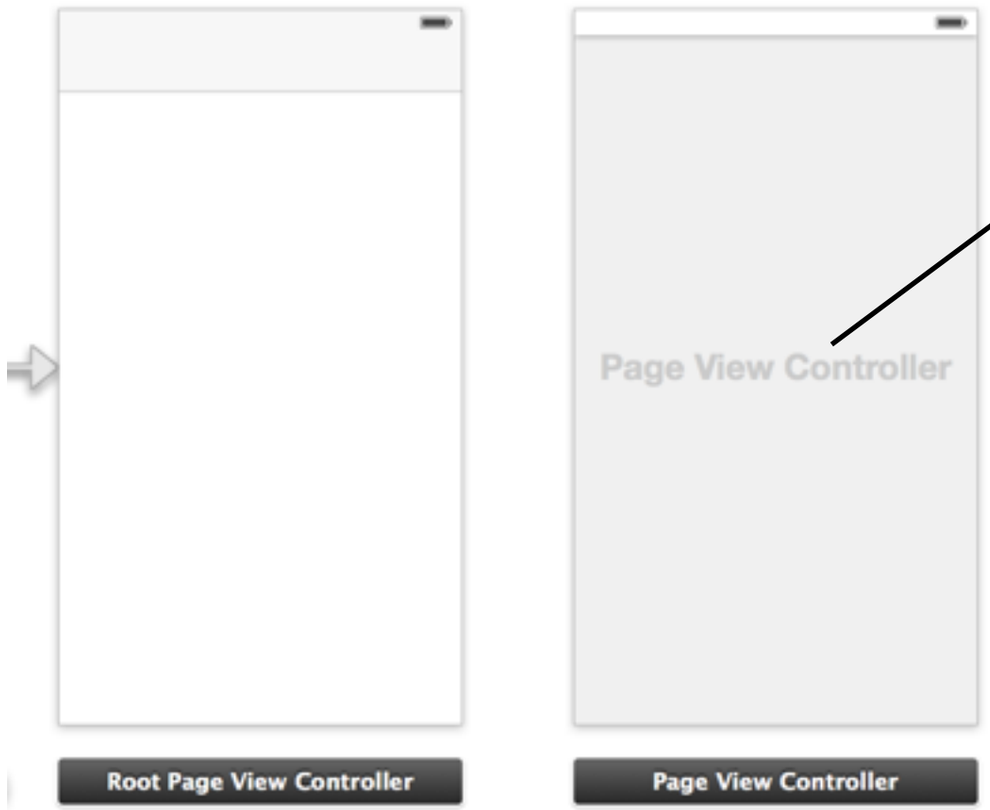


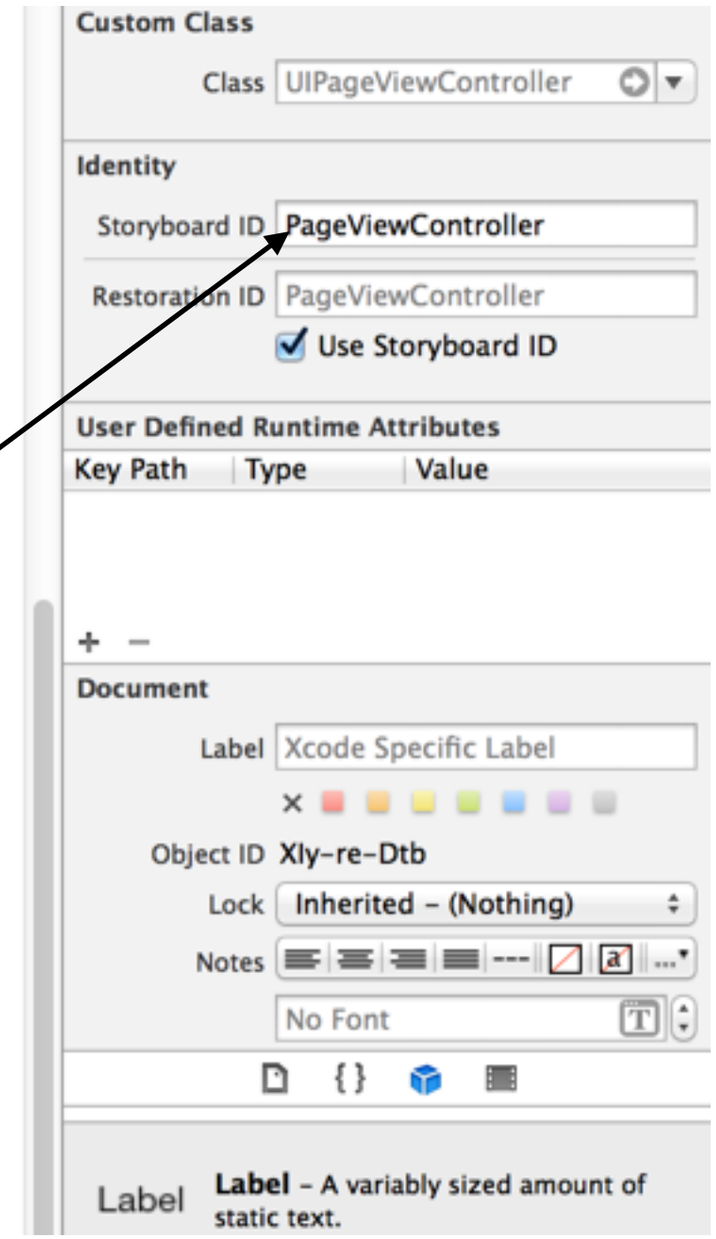
page view controller



page view controller

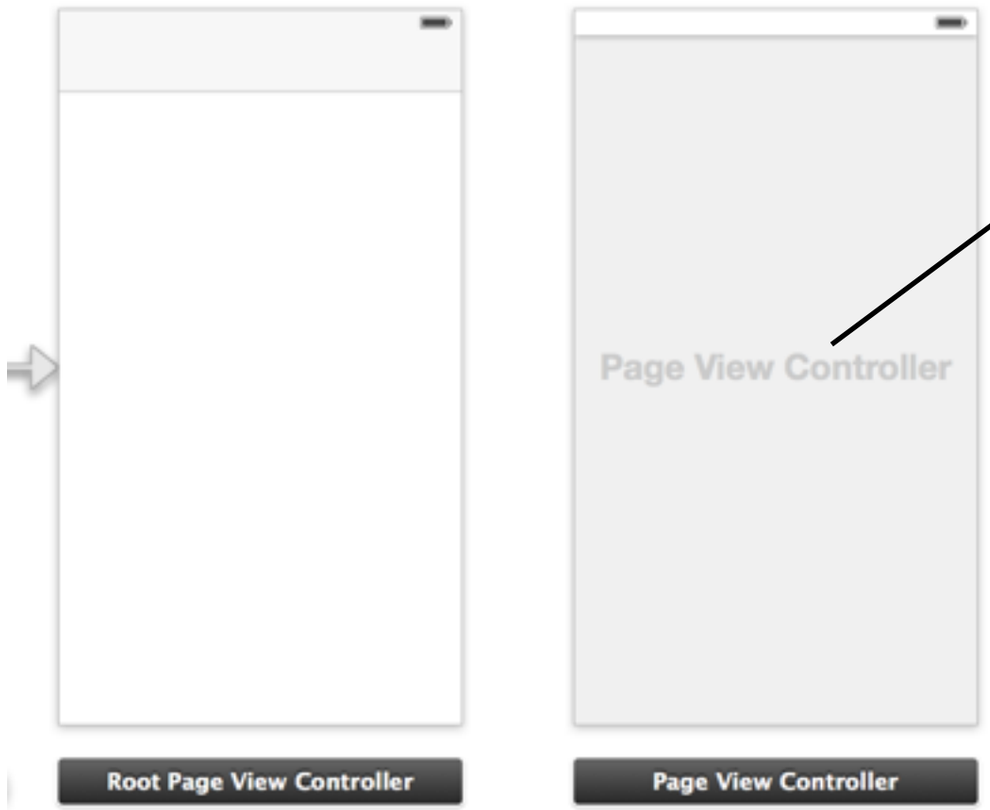
no need to subclass the page controller!

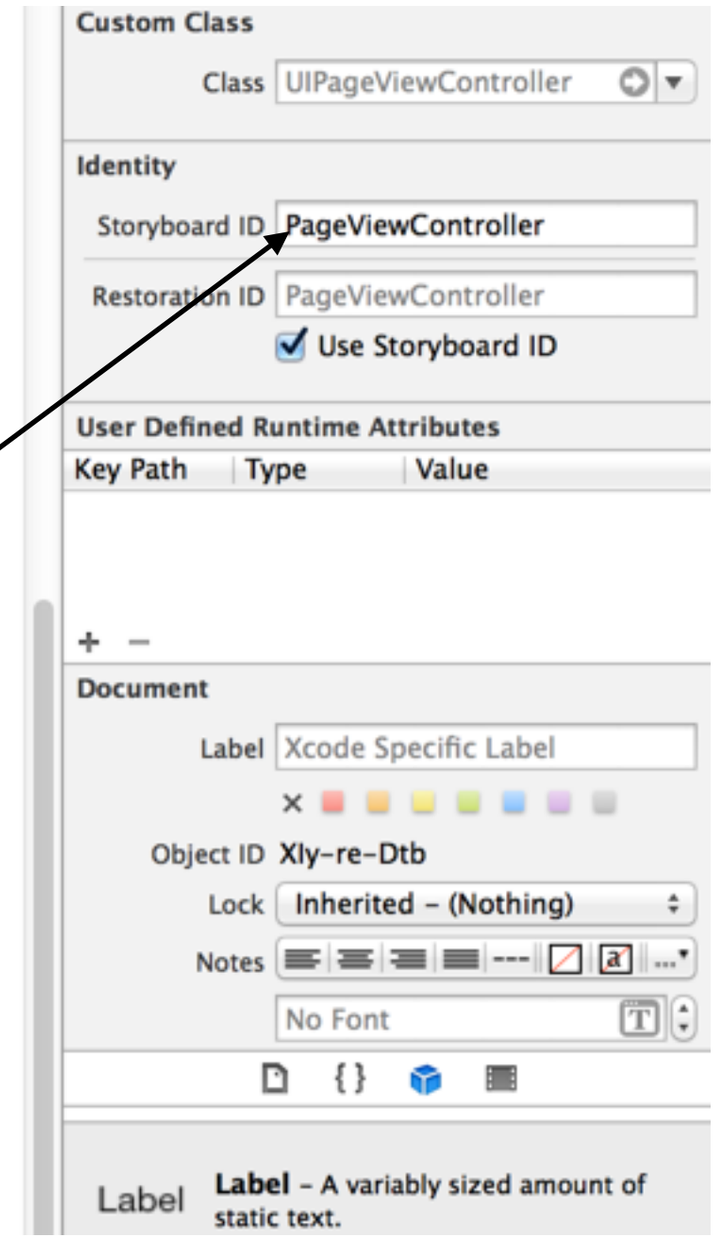




page view controller

no need to subclass the page controller!





but root of the page controller must be the data source…

root page view controller

root page view controller

instantiation in root view controller

```
@property (strong, nonatomic) UIPageViewController * pageViewController;
@property (strong, nonatomic) NSArray *pageContent;

_pageViewController = [self.storyboard instantiateViewControllerWithIdentifier:@"PageViewController"];
_pageViewController.dataSource = self;
```

root page view controller

instantiation in root view controller

```
@property (strong, nonatomic) UIPageViewController * pageViewController;
@property (strong, nonatomic) NSArray *pageContent;

_pageViewController = [self.storyboard instantiateViewControllerWithIdentifier:@"PageViewController"];
_pageViewController.dataSource = self;
```

instantiate!

root page view controller

instantiation in root view controller

```
@property (strong, nonatomic) UIPageViewController * pageViewController;
@property (strong, nonatomic) NSArray *pageContent;

_pageViewController = [self.storyboard instantiateViewControllerWithIdentifier:@"PageViewController"];
_pageViewController.dataSource = self;
```

instantiate!

in viewDidLoad

```
[self.pageViewController setViewControllers:firstPageToDisplay // the page is a view controller!
                                  direction:UIPageViewControllerNavigationDirectionForward
                                   animated:NO
                                 completion:nil];

[self addChildViewController:_pageViewController];
[self.view addSubview:_pageViewController.view];
[self.pageViewController didMoveToParentViewController:self];
```

root page view controller

instantiation in root view controller

```
@property (strong, nonatomic) UIPageViewController * pageViewController;
@property (strong, nonatomic) NSArray *pageContent;

_pageViewController = [self.storyboard instantiateViewControllerWithIdentifier:@"PageViewController"];
_pageViewController.dataSource = self;
```

set first page

instantiate!

in viewDidLoad

```
[self.pageViewController setViewControllers:firstPageToDisplay // the page is a view controller!
                                  direction:UIPageViewControllerNavigationDirectionForward
                                   animated:NO
                                 completion:nil];

[self addChildViewController:_pageViewController];
[self.view addSubview:_pageViewController.view];
[self.pageViewController didMoveToParentViewController:self];
```

root page view controller

instantiation in root view controller

```
@property (strong, nonatomic) UIPageViewController * pageViewController;
@property (strong, nonatomic) NSArray *pageContent;

_pageViewController = [self.storyboard instantiateViewControllerWithIdentifier:@"PageViewController"];
_pageViewController.dataSource = self;
```

set first page

instantiate!

in viewDidLoad

```
[self.pageViewController setViewControllers:firstPageToDisplay // the page is a view controller!
                                  direction:UIPageViewControllerNavigationDirectionForward
                                   animated:NO
                                 completion:nil];

[self addChildViewController:_pageViewController];
[self.view addSubview:_pageViewController.view];
[self.pageViewController didMoveToParentViewController:self];
```

apple says do this, in order

root page view controller

instantiation in root view controller

```
@property (strong, nonatomic) UIPageViewController * pageViewController;
@property (strong, nonatomic) NSArray *pageContent;

_pageViewController = [self.storyboard instantiateViewControllerWithIdentifier:@"PageViewController"];
_pageViewController.dataSource = self;
```

set first page

instantiate!

in viewDidLoad

```
[self.pageViewController setViewControllers:firstPageToDisplay // the page is a view controller!
                                  direction:UIPageViewControllerNavigationDirectionForward
                                   animated:NO
                                 completion:nil];

[self addChildViewController:_pageViewController];
[self.view addSubview:_pageViewController.view];
[self.pageViewController didMoveToParentViewController:self];
```

apple says do this, in order

some datasource protocol methods

```
- (NSInteger)presentationCountForPageViewController:(UIPageViewController *)pageViewController
{
    return [self.pageContent count];
}

- (NSInteger)presentationIndexForPageViewController:(UIPageViewController *)pageViewController
{
    return 0;
}
```

root page view controller

some datasource protocol methods (cont.)

```
- (NSInteger)presentationCountForPageViewController:(UIPageViewController *)pageViewController
{
    return [self.pageContent count];
}

- (NSInteger)presentationIndexForPageViewController:(UIPageViewController *)pageViewController
{
    return 0;
}
```

root page view controller

some datasource protocol methods (cont.)

```
- (NSInteger)presentationCountForPageViewController:(UIPageViewController *)pageViewController
{
    return [self.pageContent count];
}

- (NSInteger)presentationIndexForPageViewController:(UIPageViewController *)pageViewController
{
    return 0;
}

-(UIViewController*)pageViewController:(UIPageViewController *)pageViewController
viewControllerBeforeViewController:(UIViewController *)viewController
{}
```

root page view controller

some datasource protocol methods (cont.)

```
- (NSInteger)presentationCountForPageViewController:(UIPageViewController *)pageViewController
{
    return [self.pageContent count];
}

- (NSInteger)presentationIndexForPageViewController:(UIPageViewController *)pageViewController
{
    return 0;
}

-(UIViewController*)pageViewController:(UIPageViewController *)pageViewController
viewControllerBeforeViewController:(UIViewController *)viewController
{}

-(UIViewController*)pageViewController:(UIPageViewController *)pageViewController
viewControllerAfterViewController:(UIViewController *)viewController
{}
```

root page view controller

some datasource protocol methods (cont.)

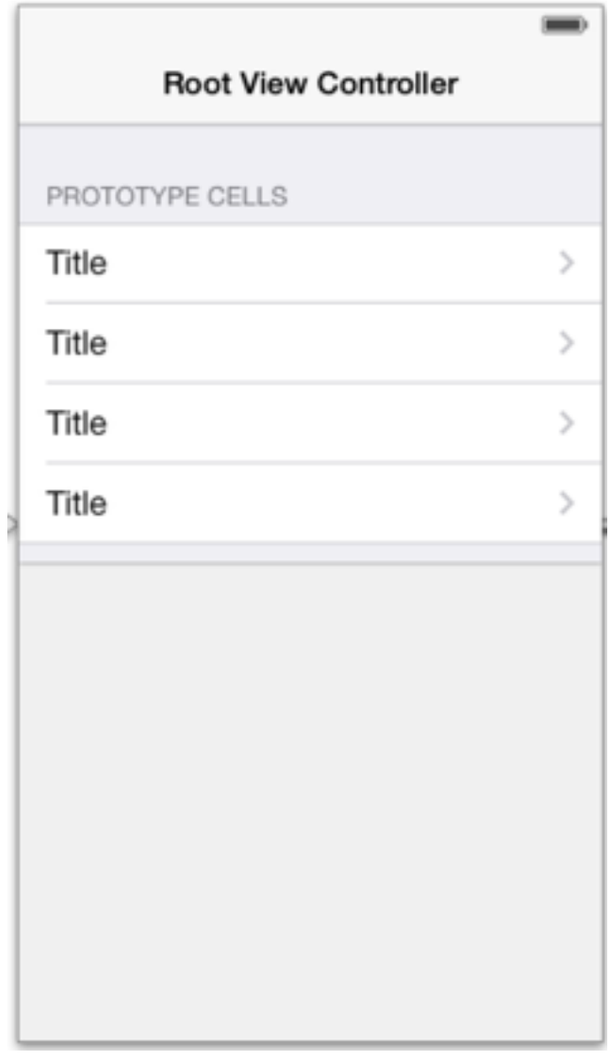
```
- (NSInteger)presentationCountForPageViewController:(UIPageViewController *)pageViewController
{
    return [self.pageContent count];
}

- (NSInteger)presentationIndexForPageViewController:(UIPageViewController *)pageViewController
{
    return 0;
}

-(UIViewController*)pageViewController:(UIPageViewController *)pageViewController
viewControllerBeforeViewController:(UIViewController *)viewController
{}

-(UIViewController*)pageViewController:(UIPageViewController *)pageViewController
viewControllerAfterViewController:(UIViewController *)viewController
{}
```

1. create pages (VCs)
2. set any information for loading
3. return the instantiated VC

page view demo

assignment one

- Automatic Layout (storyboard and programmatically)
- UIButtons (created in storyboard and programmatically)
- Sliders (created in storyboard and programmatically)
- Labels (created in storyboard and programmatically)
- Stepper
- Switch
- Picker (Date or otherwise)
- UINavigationController
- **UISegmentedControl**
- **NSTimer** (which should repeat and somehow update the UIView)
- UIScrollView (with scrollable, zoomable content)
- UIPageViewController
- UIImageView
- **(optional) Persistent storage via CoreData**



due Friday, Feb. 7

timers, segmented control

```
- (IBAction)updateFromSegmentedControl:(UISegmentedControl *)sender {

    NSString *selectedText = [sender titleForSegmentAtIndex: [sender selectedSegmentIndex]];

    YOUR_CODE
}
```

timers, segmented control

```
- (IBAction)updateFromSegmentedControl:(UISegmentedControl *)sender {

    NSString *selectedText = [sender titleForSegmentAtIndex: [sender selectedSegmentIndex]];

    YOUR_CODE
}
```

timers, segmented control

```
- (IBAction)updateFromSegmentedControl:(UISegmentedControl *)sender {

    NSString *selectedText = [sender titleForSegmentAtIndex: [sender selectedSegmentIndex]];

    YOUR_CODE
}
```

get title from control

timers, segmented control

```
- (IBAction)updateFromSegmentedControl:(UISegmentedControl *)sender {

    NSString *selectedText = [sender titleForSegmentAtIndex: [sender selectedSegmentIndex]];

    YOUR_CODE
}
```

get title from control


```
NSTimer *timer = [NSTimer scheduledTimerWithTimeInterval:someIntervalInSeconds
                                                  target:self
                                                selector:@selector(someFunction:)
                                                userInfo:nil
                                                 repeats:YES];

// don't get blocked by the main thread
[[NSRunLoop mainRunLoop] addTimer:timer forMode:NSRunLoopCommonModes];
```

core data databases

core data databases

- allows access to SQLite database

core data databases

- allows access to SQLite database
- integrated deeply into Xcode and into iOS

core data databases

- allows access to SQLite database
- integrated deeply into Xcode and into iOS
- highly optimized

core data databases

- allows access to SQLite database
- integrated deeply into Xcode and into iOS
- highly optimized
- excellent for storing persistent table data

core data databases

- allows access to SQLite database
- integrated deeply into Xcode and into iOS
- highly optimized
- excellent for storing persistent table data
  - but usable for most anything

core data schema

ENTITIES
- Student
- Teams

FETCH REQUESTS

CONFIGURATIONS
- Default

Attributes

Attribute	Type
hardware	String
name	String

+ −

ENTITIES
- Student
- Teams

FETCH REQUESTS

CONFIGURATIONS
- Default

Attributes

Attribute	Type
major	String
name	String

+ −

core data schema

ENTITIES
- Student
- Teams

FETCH REQUESTS

CONFIGURATIONS
- Default

Attributes

Attribute	Type
hardware	String
name	String

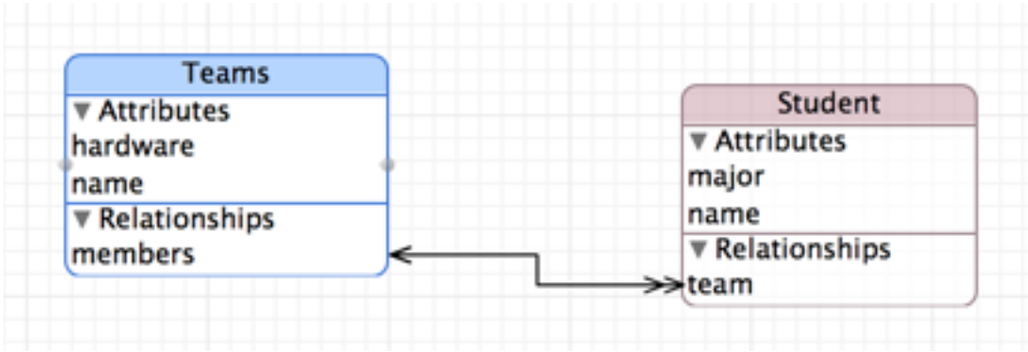
ENTITIES
- Student
- Teams

FETCH REQUESTS

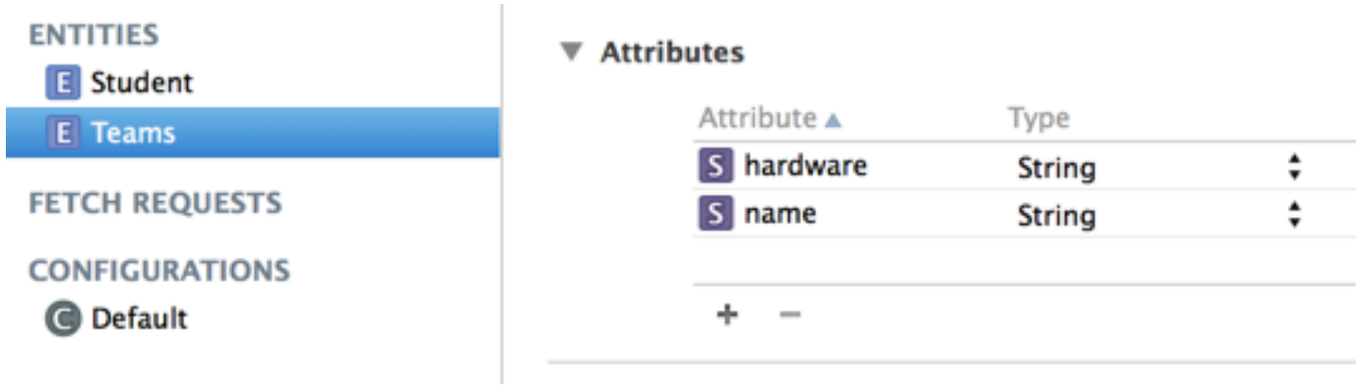
CONFIGURATIONS
- Default

Attributes

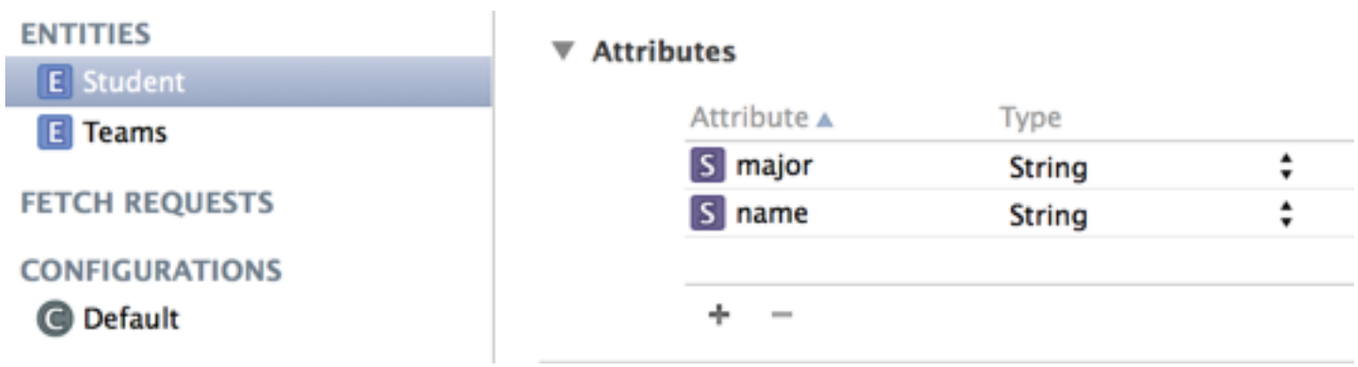
Attribute	Type
major	String
name	String

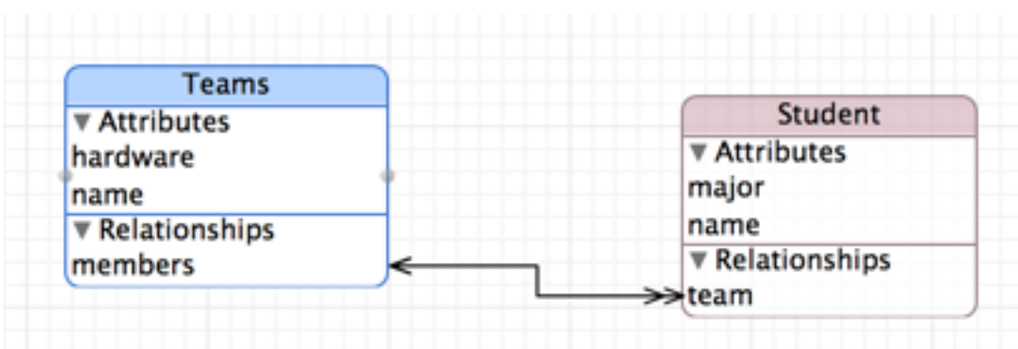


core data schema




```
@interface Teams : NSManagedObject

@property (nonatomic, retain) NSString * name;
@property (nonatomic, retain) NSString * hardware;
@property (nonatomic, retain) NSSet *members;

@end
```





core data schema

ENTITIES
- Student
- Teams

FETCH REQUESTS

CONFIGURATIONS
- Default

Attributes

Attribute	Type
hardware	String
name	String

```
@interface Teams : NSManagedObject

@property (nonatomic, retain) NSString * name;
@property (nonatomic, retain) NSString * hardware;
@property (nonatomic, retain) NSSet *members;

@end
```

ENTITIES
- Student
- Teams

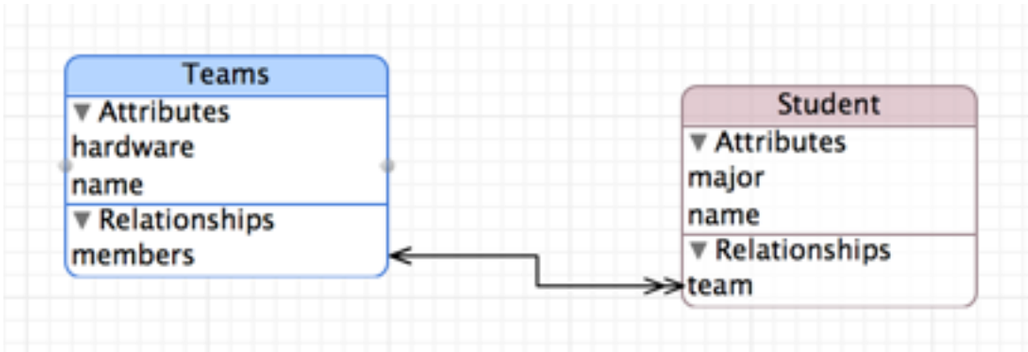
FETCH REQUESTS

CONFIGURATIONS
- Default

Attributes

Attribute	Type
major	String
name	String

```
@interface Student : NSManagedObject

@property (nonatomic, retain) NSString * name;
@property (nonatomic, retain) NSString * major;
@property (nonatomic, retain) Teams *team;

@end
```



core data

- schema creation
  - automatic subclassing
- NSManagedObject
- NSManagedObjectContext
- NSPersistentStore
- NSFetchRequest

core data

- schema creation
  - automatic subclassing
- NSManagedObject
- NSManagedObjectContext
- NSPersistentStore
- NSFetchRequest

create SQLite Database on phone

core data

- schema creation
  - automatic subclassing
- NSManagedObject
- NSManagedObjectContext
- NSPersistentStore
- NSFetchRequest

create SQLite Database on phone

enable access through properties

core data

- schema creation
  - automatic subclassing
- NSManagedObject
- NSManagedObjectContext
- NSPersistentStore
- NSFetchRequest

create SQLite Database on phone

enable access through properties

bundle “data models”

core data

- schema creation
  - automatic subclassing
- NSManagedObject
- NSManagedObjectContext
- NSPersistentStore
- NSFetchRequest

create SQLite Database on phone

enable access through properties

bundle “data models”

get “context” for using data model

core data

- schema creation
  - automatic subclassing
- NSManagedObject
- NSManagedObjectContext
- NSPersistentStore
- NSFetchRequest

create SQLite Database on phone

enable access through properties

bundle “data models”

get “context” for using data model

coordinate access to the data model

core data

- schema creation
  - automatic subclassing
- NSManagedObject
- NSManagedObjectContext
- NSPersistentStore
- NSFetchRequest

create SQLite Database on phone

enable access through properties

bundle “data models”

get “context” for using data model

coordinate access to the data model

create and execute queries

core data setup

```
// Getter for managed context
- (NSManagedObjectContext *) managedObjectContext {

    if(!_managedObjectContext){
        // create the storage coordinator
        NSPersistentStoreCoordinator *coordinator = [self persistentStoreCoordinator];
        if (coordinator != nil) {
            _managedObjectContext = [[NSManagedObjectContext alloc] init];
            [_managedObjectContext setPersistentStoreCoordinator: coordinator];
        }
    }

    return _managedObjectContext;
}
```

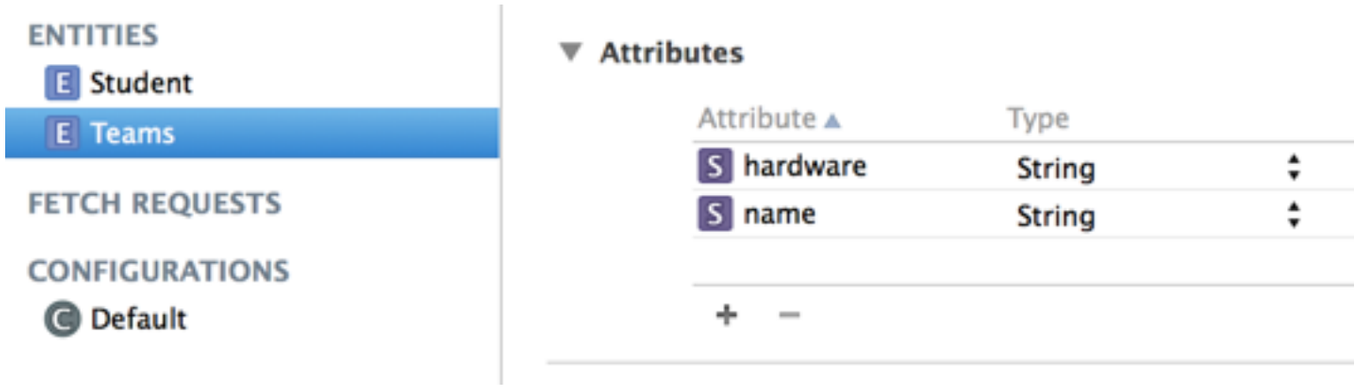
core data setup

```
// Getter for managed context
- (NSManagedObjectContext *) managedObjectContext {

    if(!_managedObjectContext){
        // create the storage coordinator
        NSPersistentStoreCoordinator *coordinator = [self persistentStoreCoordinator];
        if (coordinator != nil) {
            _managedObjectContext = [[NSManagedObjectContext alloc] init];
            [_managedObjectContext setPersistentStoreCoordinator: coordinator];
        }
    }

    return _managedObjectContext;
}

// getter for the storage coordinator
- (NSPersistentStoreCoordinator *)persistentStoreCoordinator {
    if (!_persistentStoreCoordinator) {

        // this points to our model
        NSURL *storeUrl = [NSURL fileURLWithPath: [[self applicationDocumentsDirectory]
                                                    stringByAppendingPathComponent: @"ModelName.sqlite"]];
        NSError *error = nil;
        _persistentStoreCoordinator = [[NSPersistentStoreCoordinator alloc]
                                        initWithManagedObjectModel:[self managedObjectModel]];

        if(![_persistentStoreCoordinator addPersistentStoreWithType:NSSQLiteStoreType
                                configuration:nil URL:storeUrl options:nil error:&error]) {
            // exit gracefully if you need the database to function in the UI
        }
    }
    return _persistentStoreCoordinator;
}
```

core data setup

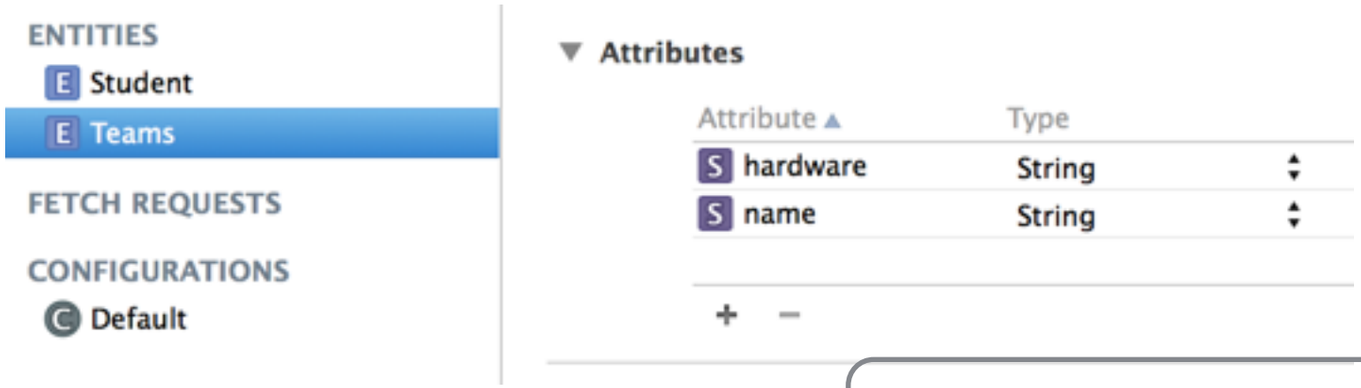
```
// getter for the storage coordinator
- (NSPersistentStoreCoordinator *)persistentStoreCoordinator {
    if (!_persistentStoreCoordinator) {

        // this points to our model
        NSURL *storeUrl = [NSURL fileURLWithPath: [[self applicationDocumentsDirectory]
                                                    stringByAppendingPathComponent: @"ModelName.sqlite"]];
        NSError *error = nil;
        _persistentStoreCoordinator = [[NSPersistentStoreCoordinator alloc]
                                        initWithManagedObjectModel:[self managedObjectModel]];

        if(![_persistentStoreCoordinator addPersistentStoreWithType:NSSQLiteStoreType
                                configuration:nil URL:storeUrl options:nil error:&error]) {
            // exit gracefully if you need the database to function in the UI
        }
    }
    return _persistentStoreCoordinator;
}


// getter for the object model, create if needed
- (NSManagedObjectModel *)managedObjectModel {
    if (!_managedObjectModel) {
        _managedObjectModel = [NSManagedObjectModel mergedModelFromBundles:nil];
    }
    return _managedObjectModel;
}
```

entering data




```
//  get a new entry
 team = [NSEntityDescription insertNewObjectForEntityForName:@"Teams"
                                        inManagedObjectContext:self.managedObjectContext];
 // save the attributes
 team.name = self.teamNameTextField.text;
 team.hardware = [self assignHardware];

 //  save into the database
 NSError *error;
 if (![self.managedObjectContext save:&error]) {
     NSLog(@"save database failed: %@", [error localizedDescription]);
 }
```

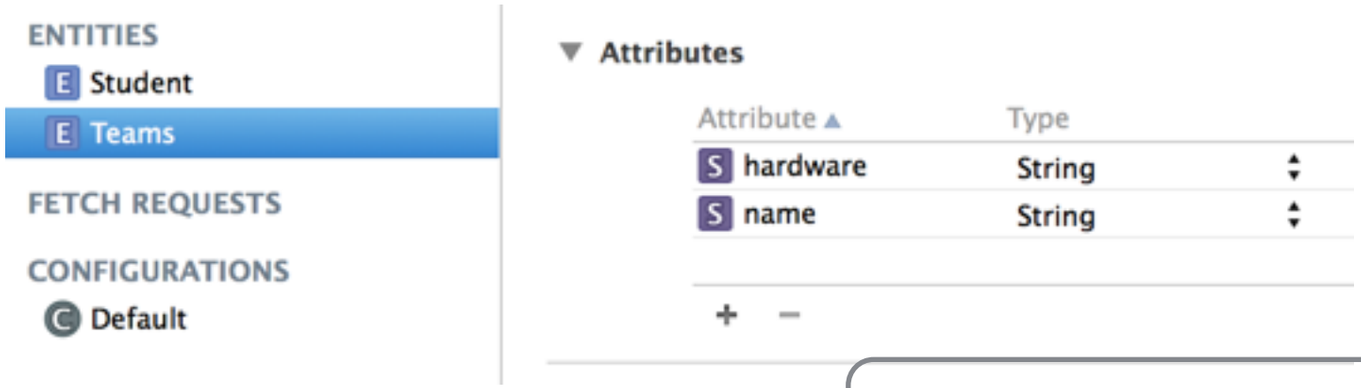
entering data



create a new entity from model

```
//  get a new entry
 team = [NSEntityDescription insertNewObjectForEntityForName:@"Teams"
                                        inManagedObjectContext:self.managedObjectContext];
 // save the attributes
 team.name = self.teamNameTextField.text;
 team.hardware = [self assignHardware];

 //  save into the database
 NSError *error;
 if (![self.managedObjectContext save:&error]) {
     NSLog(@"save database failed: %@", [error localizedDescription]);
 }
```

entering data

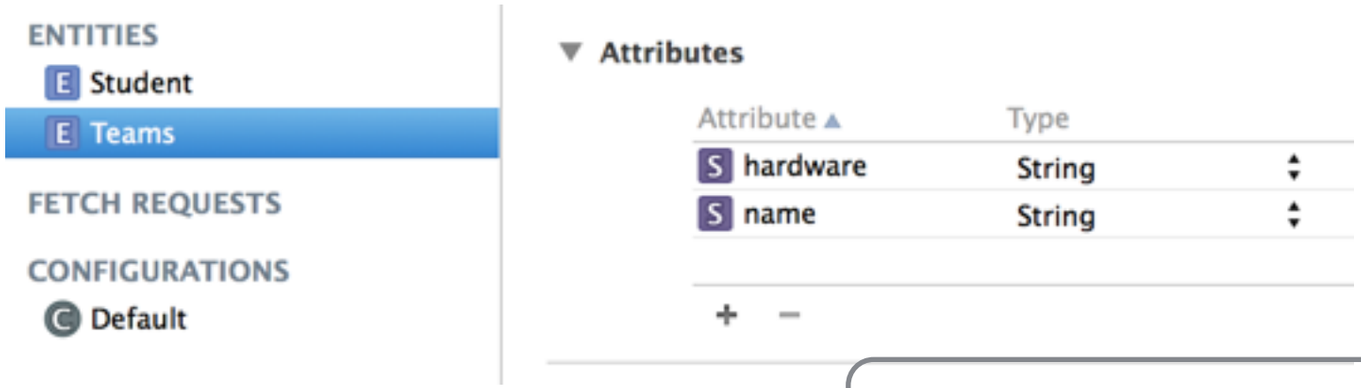


create a new entity from model

```
//  get a new entry
 team = [NSEntityDescription insertNewObjectForEntityForName:@"Teams"
                                                 inManagedObjectContext:self.managedObjectContext];
 // save the attributes
 team.name = self.teamNameTextField.text;
 team.hardware = [self assignHardware];

 //  save into the database
 NSError *error;
 if (![self.managedObjectContext save:&error]) {
     NSLog(@"save database failed: %@", [error localizedDescription]);
 }
```

set attributes

entering data



create a new entity from model

```
//  get a new entry
 team = [NSEntityDescription insertNewObjectForEntityForName:@"Teams"
                                                 inManagedObjectContext:self.managedObjectContext];
 // save the attributes
 team.name = self.teamNameTextField.text;
 team.hardware = [self assignHardware];

 //  save into the database
 NSError *error;
 if (![self.managedObjectContext save:&error]) {
     NSLog(@"save database failed: %@", [error localizedDescription]);
 }
```

set attributes

not saved in database until here

queries in core data

```
-(NSArray*)getAllTeamsFromDatabase
{
    // initializing NSFetchRequest
    NSFetchRequest *fetchRequest = [[NSFetchRequest alloc] init];

    //Setting Entity to be Queried
    NSEntityDescription *entity = [NSEntityDescription entityForName:@"Teams"
                                                inManagedObjectContext:self.managedObjectContext];
    [fetchRequest setEntity:entity];
    NSError* error;

    // Query on managedObjectContext With Generated fetchRequest
    NSArray *fetchedRecords = [self.managedObjectContext executeFetchRequest:fetchRequest error:&error];

    // Returning Fetched Records
    return fetchedRecords;
}
```

queries in core data

```
-(NSArray*)getAllTeamsFromDatabase
{
    // initializing NSFetchRequest
    NSFetchRequest *fetchRequest = [[NSFetchRequest alloc] init];

    //Setting Entity to be Queried
    NSEntityDescription *entity = [NSEntityDescription entityForName:@"Teams"
                                                 inManagedObjectContext:self.managedObjectContext];
    [fetchRequest setEntity:entity];
    NSError* error;

    // Query on managedObjectContext With Generated fetchRequest
    NSArray *fetchedRecords = [self.managedObjectContext executeFetchRequest:fetchRequest error:&error];

    // Returning Fetched Records
    return fetchedRecords;
}
```

request

queries in core data

```
-(NSArray*)getAllTeamsFromDatabase
{
    // initializing NSFetchRequest
    NSFetchRequest *fetchRequest = [[NSFetchRequest alloc] init];

    //Setting Entity to be Queried
    NSEntityDescription *entity = [NSEntityDescription entityForName:@"Teams"
                                                 inManagedObjectContext:self.managedObjectContext];
    [fetchRequest setEntity:entity];
    NSError* error;

    // Query on managedObjectContext With Generated fetchRequest
    NSArray *fetchedRecords = [self.managedObjectContext executeFetchRequest:fetchRequest error:&error];

    // Returning Fetched Records
    return fetchedRecords;
}
```

request

entity to request from

queries in core data

```
-(NSArray*)getAllTeamsFromDatabase
{
    // initializing NSFetchRequest
    NSFetchRequest *fetchRequest = [[NSFetchRequest alloc] init];

    //Setting Entity to be Queried
    NSEntityDescription *entity = [NSEntityDescription entityForName:@"Teams"
                                                 inManagedObjectContext:self.managedObjectContext];
    [fetchRequest setEntity:entity];
    NSError* error;

    // Query on managedObjectContext With Generated fetchRequest
    NSArray *fetchedRecords = [self.managedObjectContext executeFetchRequest:fetchRequest error:&error];

    // Returning Fetched Records
    return fetchedRecords;
}
```

request

fetch

entity to request from

queries in core data

```
-(NSArray*)getAllTeamsFromDatabase
{
    // initializing NSFetchRequest
    NSFetchRequest *fetchRequest = [[NSFetchRequest alloc] init];

    //Setting Entity to be Queried
    NSEntityDescription *entity = [NSEntityDescription entityForName:@"Teams"
                                                 inManagedObjectContext:self.managedObjectContext];
    [fetchRequest setEntity:entity];
    NSError* error;

    // Query on managedObjectContext With Generated fetchRequest
    NSArray *fetchedRecords = [self.managedObjectContext executeFetchRequest:fetchRequest error:&error];

    // Returning Fetched Records
    return fetchedRecords;
}
```

request

entity to request from

fetch

array of results, even if size=0

queries in core data

```
-(NSArray*)getAllTeamsFromDatabase
{
    // initializing NSFetchRequest
    NSFetchRequest *fetchRequest = [[NSFetchRequest alloc] init];

    //Setting Entity to be Queried
    NSEntityDescription *entity = [NSEntityDescription entityForName:@"Teams"
                                                 inManagedObjectContext:self.managedObjectContext];
    [fetchRequest setEntity:entity];
    NSError* error;

    // Query on managedObjectContext With Generated fetchRequest
    NSArray *fetchedRecords = [self.managedObjectContext executeFetchRequest:fetchRequest error:&error];

    // Returning Fetched Records
    return fetchedRecords;
}
```

request

fetch

entity to request from

array of results, even if size=0

```
-(NSArray*)getTeamFromDatabase:(NSString*)teamName
{
    // initializing NSFetchRequest

    …

   fetchRequest.predicate =
        [NSPredicate predicateWithFormat:@"name = %@",teamName];

    …

    // Returning Fetched Records
    return [self.managedObjectContext executeFetchRequest:fetchRequest error:&error];
}
```

queries in core data

```
-(NSArray*)getAllTeamsFromDatabase
{
    // initializing NSFetchRequest
    NSFetchRequest *fetchRequest = [[NSFetchRequest alloc] init];

    //Setting Entity to be Queried
    NSEntityDescription *entity = [NSEntityDescription entityForName:@"Teams"
                                                 inManagedObjectContext:self.managedObjectContext];
    [fetchRequest setEntity:entity];
    NSError* error;

    // Query on managedObjectContext With Generated fetchRequest
    NSArray *fetchedRecords = [self.managedObjectContext executeFetchRequest:fetchRequest error:&error];

    // Returning Fetched Records
    return fetchedRecords;
}
```

request

entity to request from

fetch

array of results, even if size=0

```
-(NSArray*)getTeamFromDatabase:(NSString*)teamName
{
    // initializing NSFetchRequest
    …

   fetchRequest.predicate =
        [NSPredicate predicateWithFormat:@"name = %@",teamName];

    …

    // Returning Fetched Records
    return [self.managedObjectContext executeFetchRequest:fetchRequest error:&error];
}
```

set predicate

queries in core data

```
-(NSArray*)getAllTeamsFromDatabase
{
    // initializing NSFetchRequest
    NSFetchRequest *fetchRequest = [[NSFetchRequest alloc] init];

    //Setting Entity to be Queried
    NSEntityDescription *entity = [NSEntityDescription entityForName:@"Teams"
                                                 inManagedObjectContext:self.managedObjectContext];
    [fetchRequest setEntity:entity];
    NSError* error;

    // Query on managedObjectContext With Generated fetchRequest
    NSArray *fetchedRecords = [self.managedObjectContext executeFetchRequest:fetchRequest error:&error];

    // Returning Fetched Records
    return fetchedRecords;
}
```

request

fetch

entity to request from

array of results, even if size=0

```
-(NSArray*)getTeamFromDatabase:(NSString*)teamName
{
    // initializing NSFetchRequest
    …

   fetchRequest.predicate =
        [NSPredicate predicateWithFormat:@"name = %@",teamName];

    …

    // Returning Fetched Records
    return [self.managedObjectContext executeFetchRequest:fetchRequest error:&error];
}
```

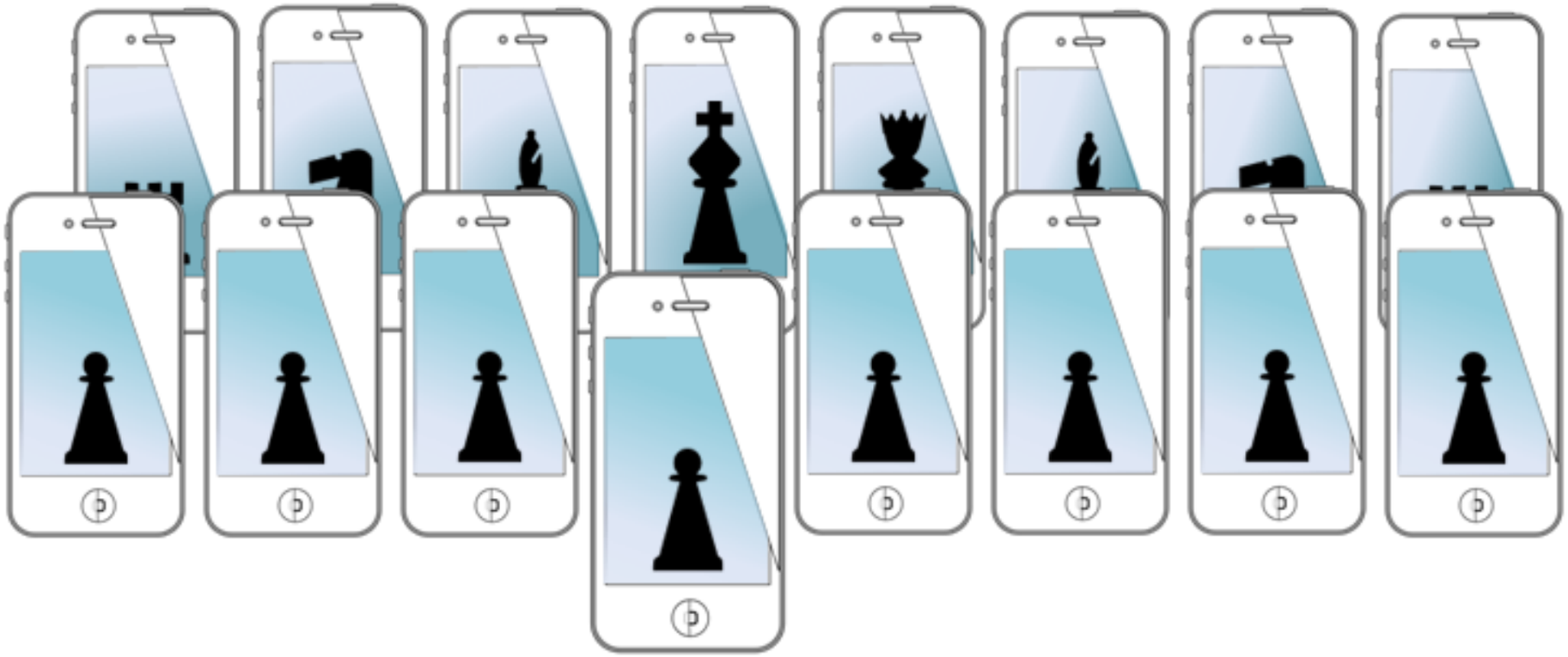
set predicate

```
@"name = %@“
@"name contains[c] %@“
@"value > 7“
@“team.name = %@“
@“any student.name contains %@“
```

core data demo

- Who Was In That!
- Class Teams! will make available on website

MOBILE SENSING LEARNING & CONTROL



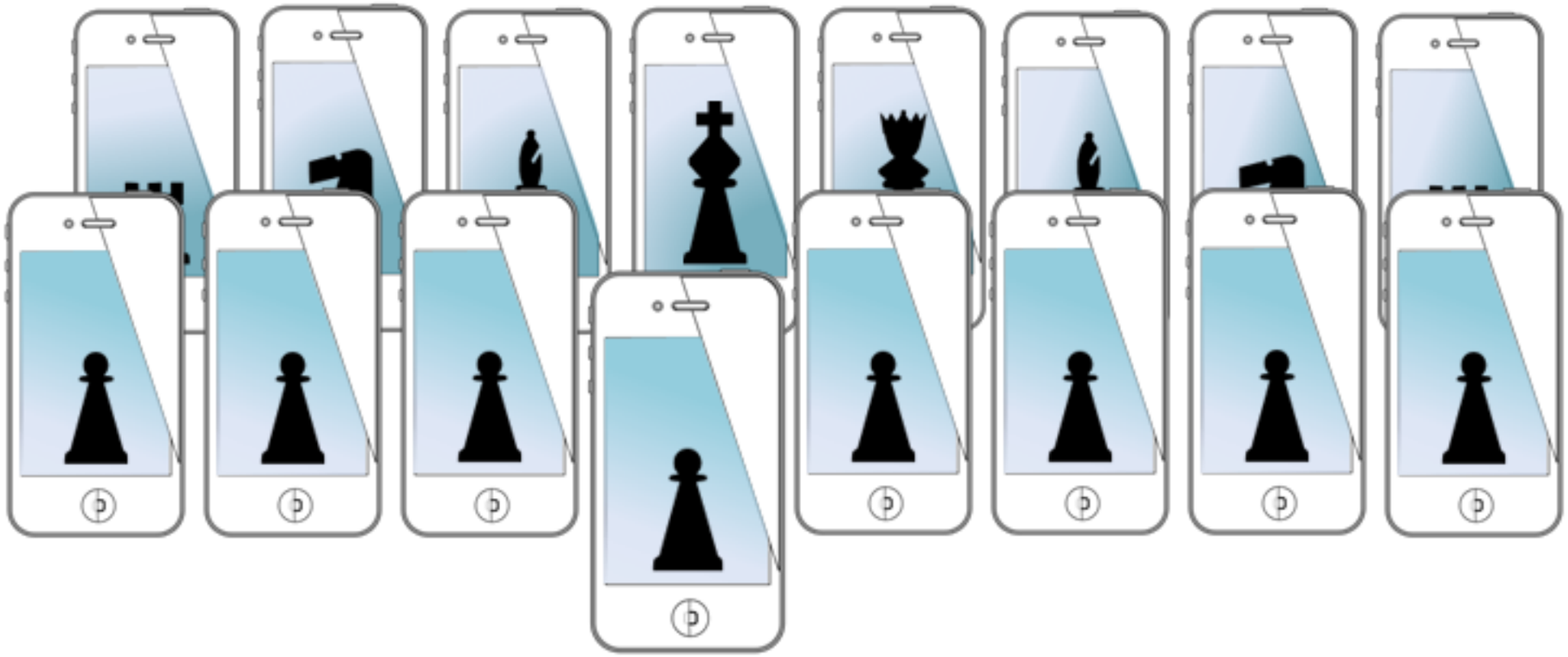
CSE5323 & 7323

Mobile Sensing, Learning, and Control

lecture four: page controllers & core data

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

MOBILE SENSING LEARNING & CONTROL



CSE5323 & 7323

Mobile Sensing, Learning, and Control

lecture five: queues, blocks, c++, audio session

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

agenda

- questions about core data?
- notifications
- blocks and multi-threading
- objective c++
- core audio intro
  - demo if time

notifications

- NotificationCenter - a radio station for which any method can tune in on

```
MCDAppDelegate* appDelegate = [UIApplication sharedApplication].delegate;

    [[NSNotificationCenter defaultCenter] addObserver:self
                                              selector:@selector(tableDataDidChange)
                                                  name:NSManagedObjectContextDidSaveNotification
                                                object:appDelegate.managedObjectContext];
```

notifications

- NotificationCenter - a radio station for which any method can tune in on

access notification center

```
MCDAppDelegate* appDelegate =                                    delegate;

    [[NSNotificationCenter defaultCenter] addObserver:self
                                             selector:@selector(tableDataDidChange)
                                                 name:NSManagedObjectContextDidSaveNotification
                                               object:appDelegate.managedObjectContext];
```

notifications

- NotificationCenter - a radio station for which any method can tune in on

access notification center

```
MCDAppDelegate* appDelegate =                                    delegate;

    [[NSNotificationCenter defaultCenter] addObserver:self
                                             selector:@selector(tableDataDidChange)
                                                 name:NSManagedObjectContextDidSaveNotification
                                               object:appDelegate.managedObjectContext];
```

when this happens

notifications

- NotificationCenter - a radio station for which any method can tune in on

access notification center

```
MCDAppDelegate* appDelegate =                                    delegate;

    [[NSNotificationCenter defaultCenter] addObserver:self
                                             selector:@selector(tableDataDidChange)
                                                 name:NSManagedObjectContextDidSaveNotification
                                               object:appDelegate.managedObjectContext];
```

when this happens

from this context

notifications

- NotificationCenter - a radio station for which any method can tune in on

access notification center

do this

```
MCDAppDelegate* appDelegate = [[UIApplication sharedApplication].delegate];

    [[NSNotificationCenter defaultCenter] addObserver:self
                                             selector:@selector(tableDataDidChange)
                                                 name:NSManagedObjectContextDidSaveNotification
                                               object:appDelegate.managedObjectContext];
```

when this happens

from this context

notifications

- NotificationCenter - a radio station for which any method can tune in on

access notification center

do this

```
MCDAppDelegate* appDelegate = [[UIApplication sharedApplication].delegate;

    [[NSNotificationCenter defaultCenter] addObserver:self
                                             selector:@selector(tableDataDidChange)
                                                 name:NSManagedObjectContextDidSaveNotification
                                               object:appDelegate.managedObjectContext];
```

when this happens

from this context

lets add notifications to WhoWasInThat!

blocks

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

block syntax

```
//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));
```

block syntax

block name

```
//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));
```

block syntax

return type

block name

```
// 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));
```

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));
```

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

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);
```

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

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;
};
```

type define, more like callback

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

syntax to call block

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;
};
```

type define, more like callback

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

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]);
}];
```

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;
};
```

type define, more like callback

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

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
  - __block float someVariable; // this is now readwrite
- blocks hold a **strong** pointer from where there defined
  - so using “self” would create a retain cycle

concurrency in iOS

concurrency in iOS

- grand central dispatch (GCD) handles all operations

concurrency in iOS

- grand central dispatch (GCD) handles all operations
  - GCD looks at “queues” of **blocks** that need to be run

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

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

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

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

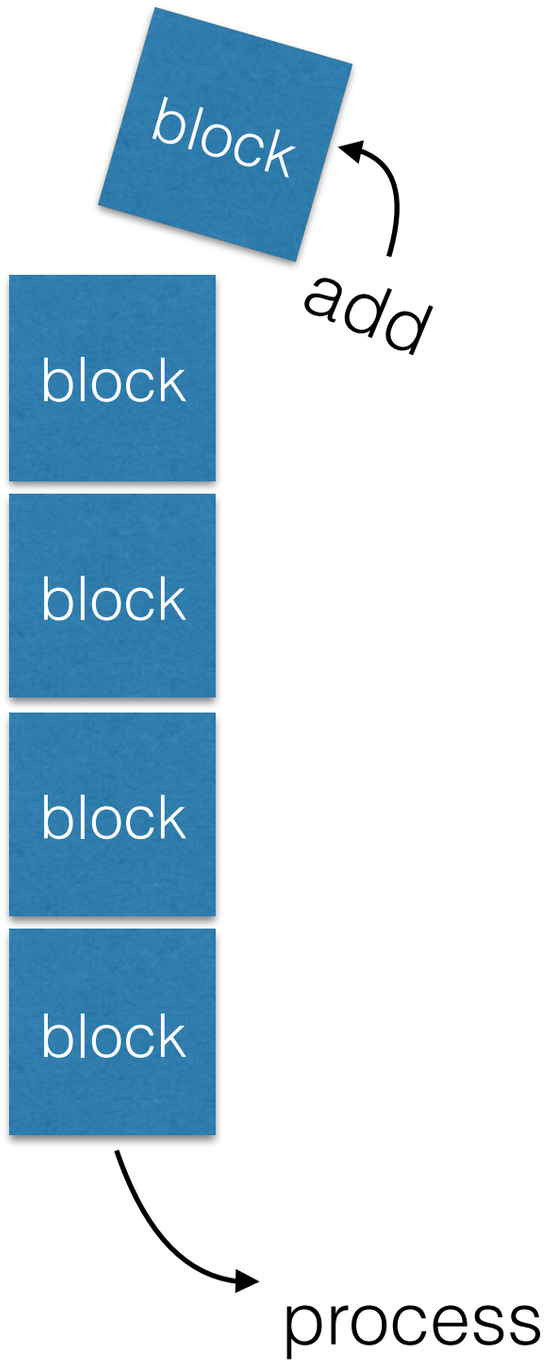
concurrency in iOS

- 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!!)

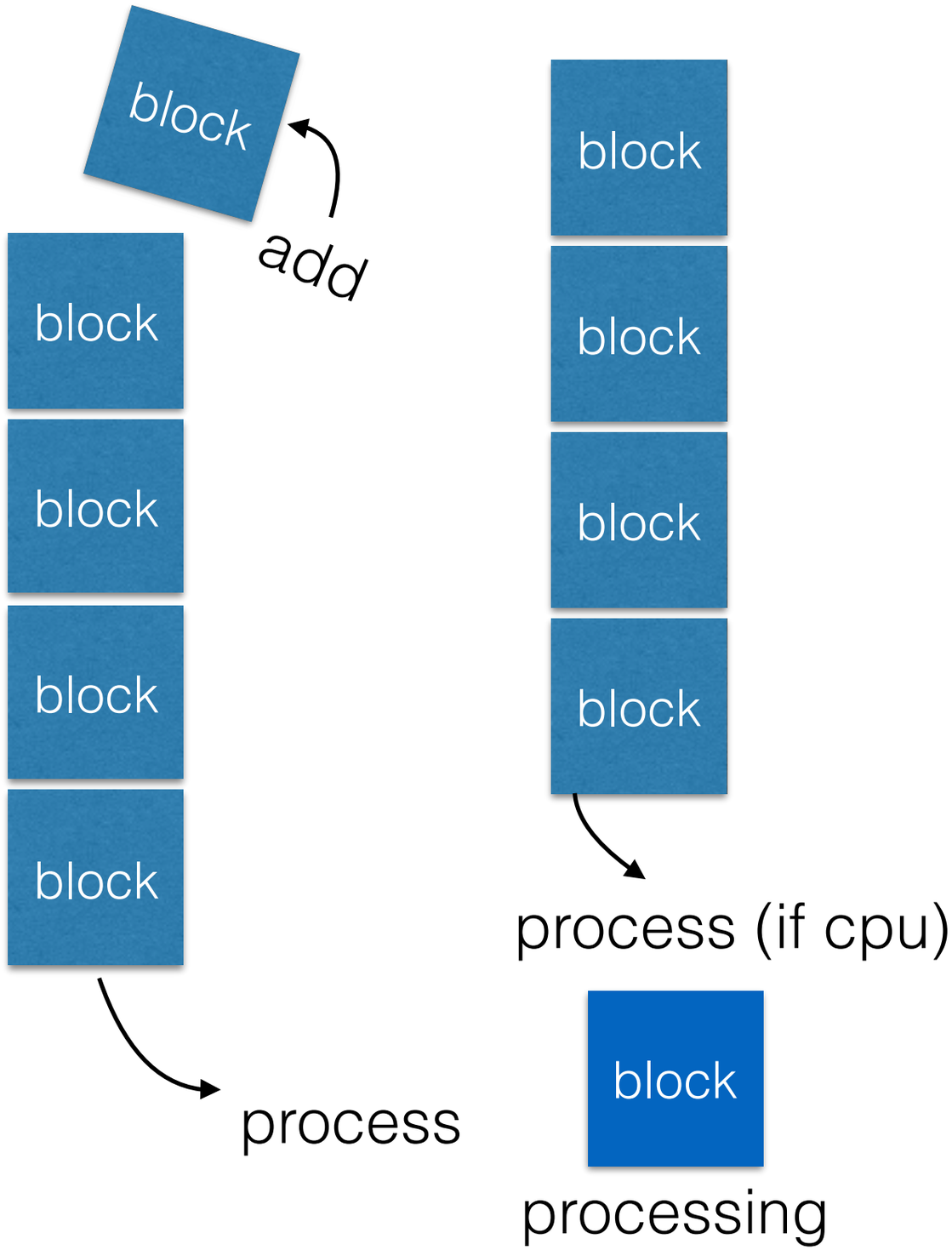
concurrency in iOS

- 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

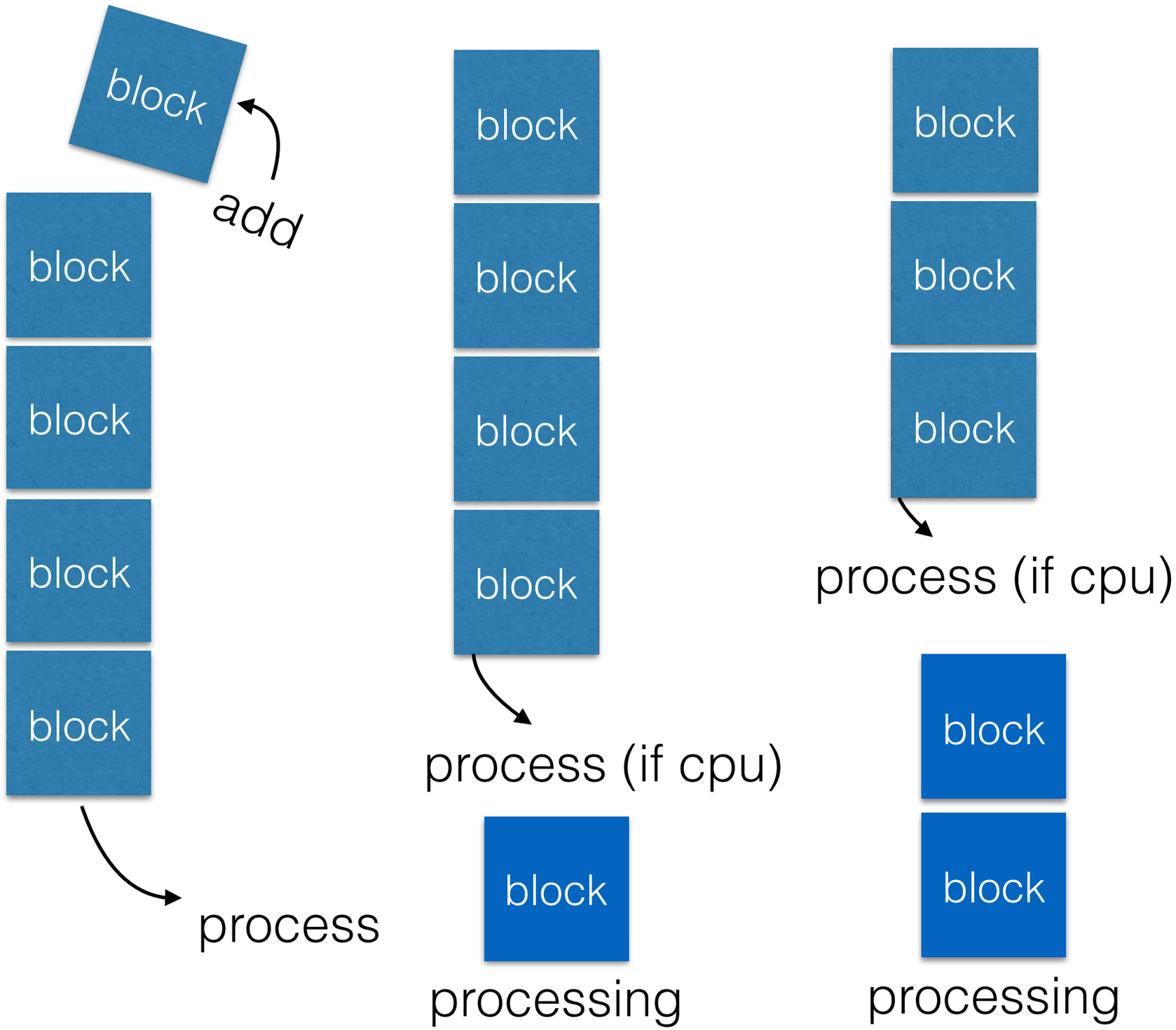
concurrent queues



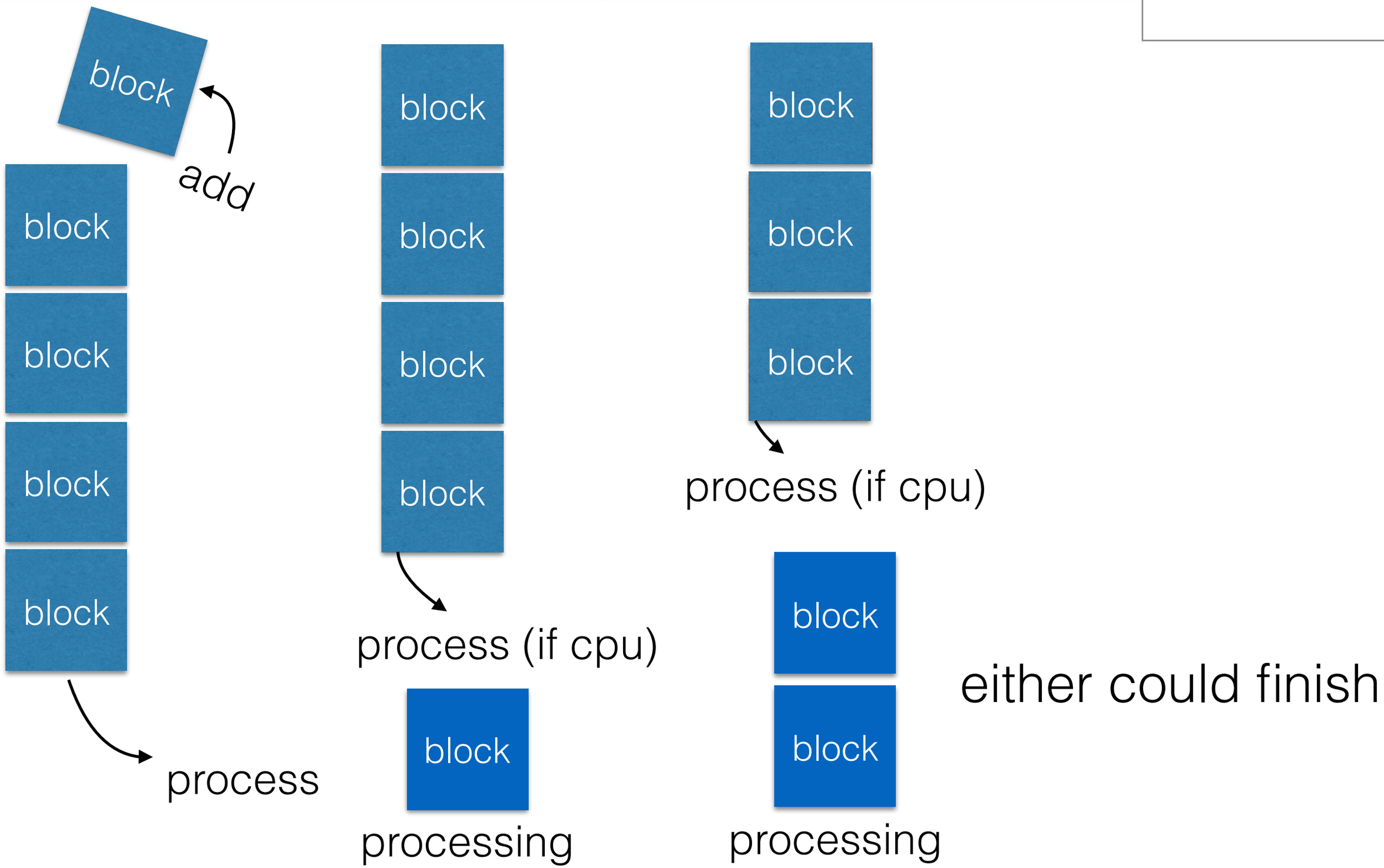
concurrent queues



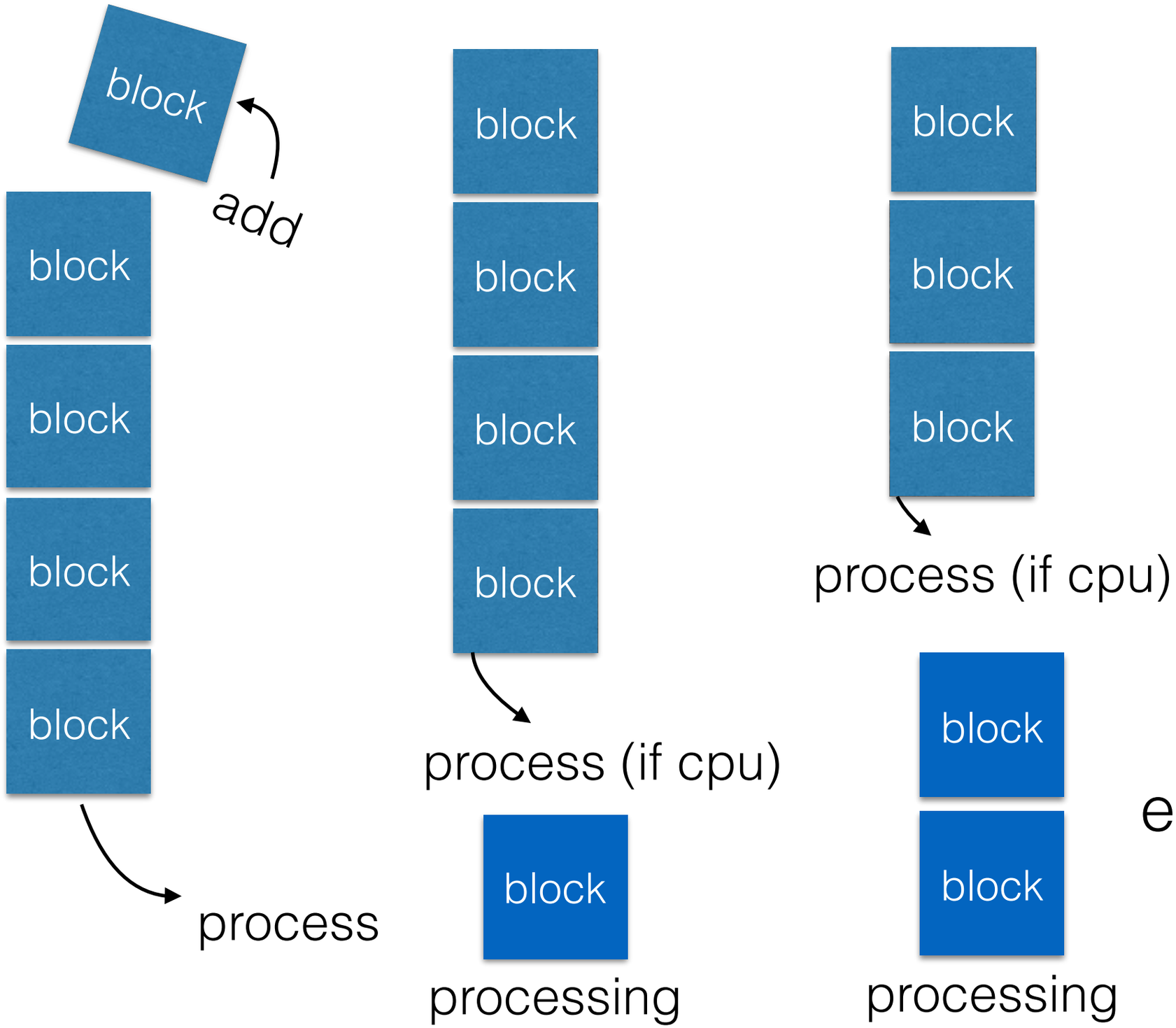
concurrent queues



concurrent queues



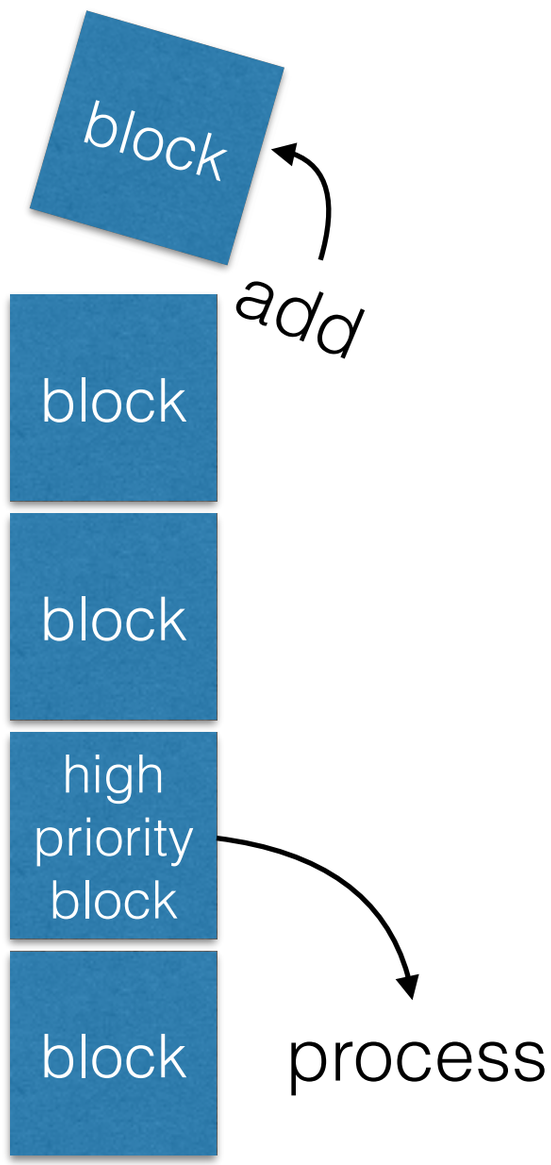
concurrent queues



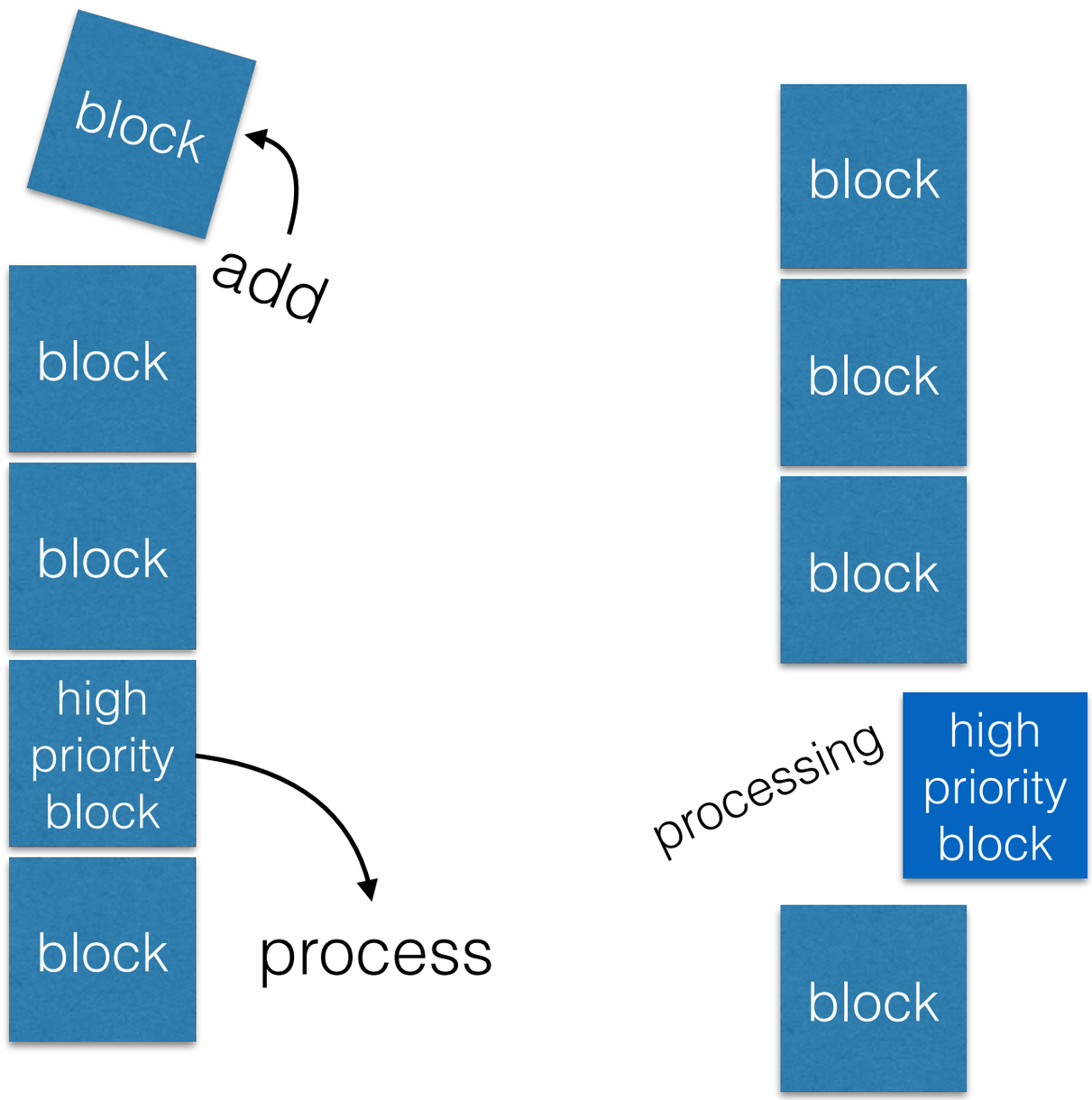
threads?

either could finish

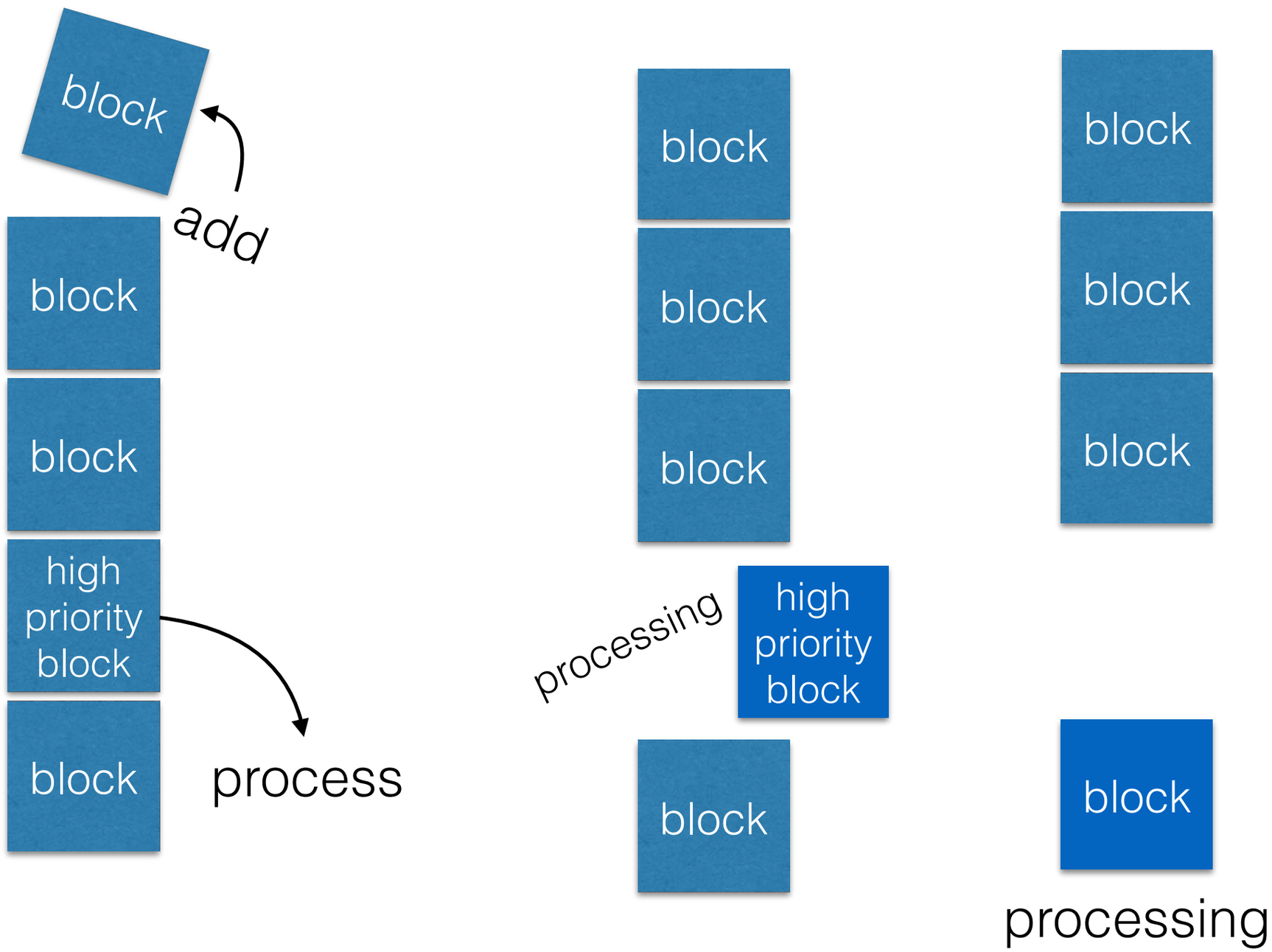
the main queue



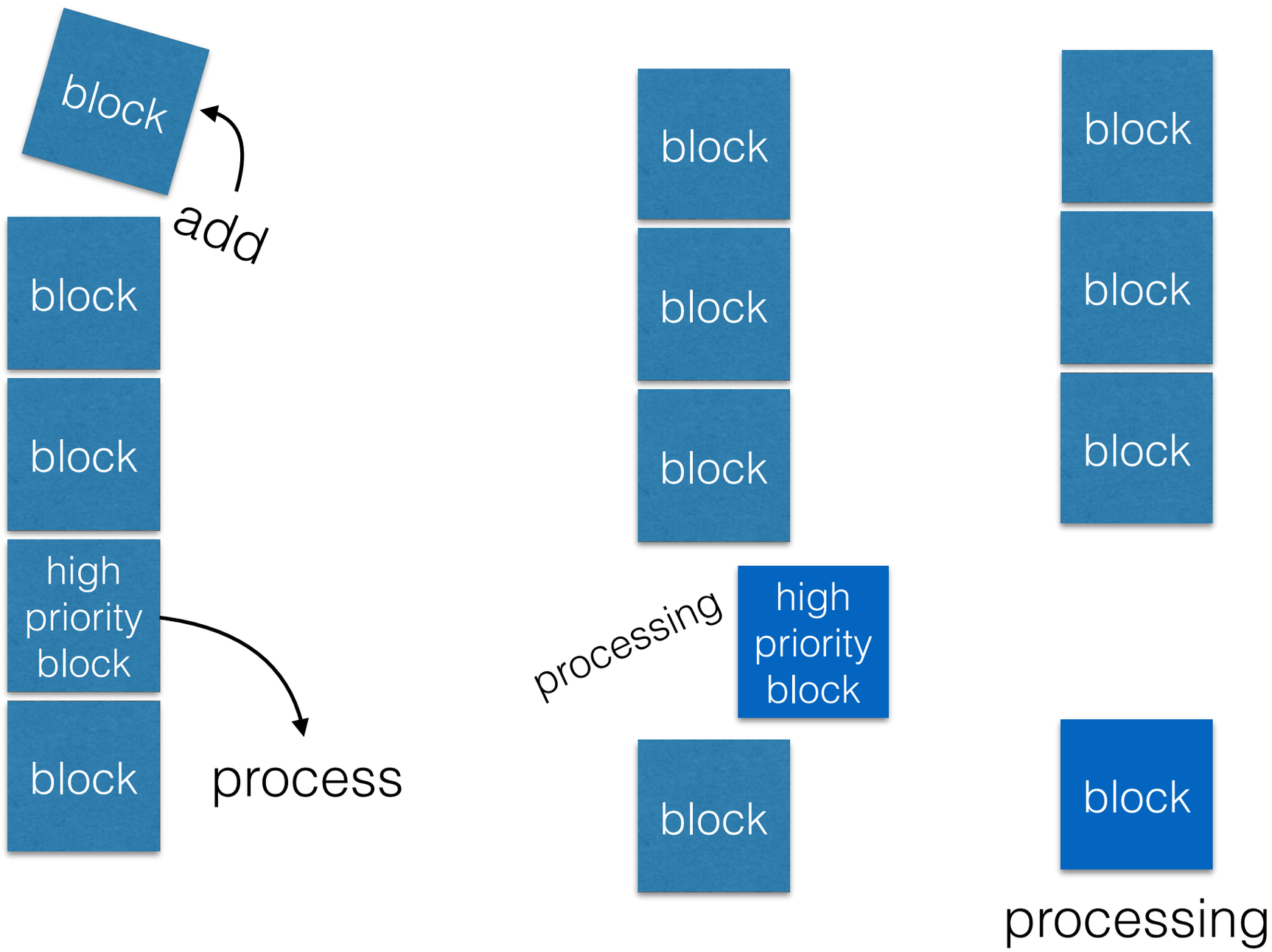
the main queue



the main queue



the main queue



one thread
UI is in sync

queue syntax

```
// 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
```

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
```

queue syntax

create new queue

define block

```
// 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
```

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
```

define block

update UI, main thread

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
```

define block

update UI, main thread

```
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];


    }];
```

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
```

define block

update UI, main thread

```
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];


    }];
```

create new queue

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
```

define block

update UI, main thread

```
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];


    }];
```

create new queue

define block

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
```

define block

update UI, main thread

```
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];


    }];
```

create new queue

define block

update UI, main thread

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
```

define block

serial or concurrent

update UI, main thread

```
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];


    }];
```

create new queue

define block

update UI, main thread

queue syntax

- using global queues

queue syntax

- using global queues

```
// 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 I am not in the main thread!

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

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 I am not in the main thread!

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

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 I am not in the main thread!

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

```
DISPATCH_QUEUE_PRIORITY_LOW
DISPATCH_QUEUE_PRIORITY_DEFAULT
DISPATCH_QUEUE_PRIORITY_HIGH
DISPATCH_QUEUE_PRIORITY_BACKGROUND
```

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 not 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!!

```
DISPATCH_QUEUE_PRIORITY_LOW
DISPATCH_QUEUE_PRIORITY_DEFAULT
DISPATCH_QUEUE_PRIORITY_HIGH
DISPATCH_QUEUE_PRIORITY_BACKGROUND
```

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 not 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++

objective c++

- actually, its just c++

objective c++

- actually, its just c++
- …but need to tell compiler we are using c++

objective c++

- actually, its just c++
- …but need to tell compiler we are using c++

objective c++

- actually, its just c++
- …but need to tell compiler we are using c++

- add any #include statements

objective c++

- 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)

objective c++

- 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

objective c++

- 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

objective c++

- 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

so let's add a c++ class to our multi-tasking code

Core Audio

Core Audio

- Audio Sessions (completely overhauled for iOS7)

Core Audio

- Audio Sessions (completely overhauled for iOS7)
  - shared instance (for all applications)

Core Audio

- Audio Sessions (completely overhauled for iOS7)
  - shared instance (for all applications)
  - set category (play, record, both)

Core Audio

- Audio Sessions (completely overhauled for iOS7)
  - shared instance (for all applications)
  - set category (play, record, both)
    - choose options: like mixing with ambient sources

Core Audio

- Audio Sessions (completely overhauled for iOS7)
  - shared instance (for all applications)
  - set category (play, record, both)
    - choose options: like mixing with ambient sources
  - set audio route (new in iOS7)

Core Audio

- Audio Sessions (completely overhauled for iOS7)
  - shared instance (for all applications)
  - set category (play, record, both)
    - choose options: like mixing with ambient sources
  - set audio route (new in iOS7)
    - set specific hardware within audio route

Core Audio

- Audio Sessions (completely overhauled for iOS7)
  - shared instance (for all applications)
  - set category (play, record, both)
    - choose options: like mixing with ambient sources
  - set audio route (new in iOS7)
    - set specific hardware within audio route
- Audio Units (output, input)

Core Audio

- Audio Sessions (completely overhauled for iOS7)
  - shared instance (for all applications)
  - set category (play, record, both)
    - choose options: like mixing with ambient sources
  - set audio route (new in iOS7)
    - set specific hardware within audio route
- Audio Units (output, input)
  - set stream format, buffer sizes, sampling rate,

Core Audio

- Audio Sessions (completely overhauled for iOS7)
  - shared instance (for all applications)
  - set category (play, record, both)
    - choose options: like mixing with ambient sources
  - set audio route (new in iOS7)
    - set specific hardware within audio route
- Audio Units (output, input)
  - set stream format, buffer sizes, sampling rate,
  - initialize memory for audio buffers

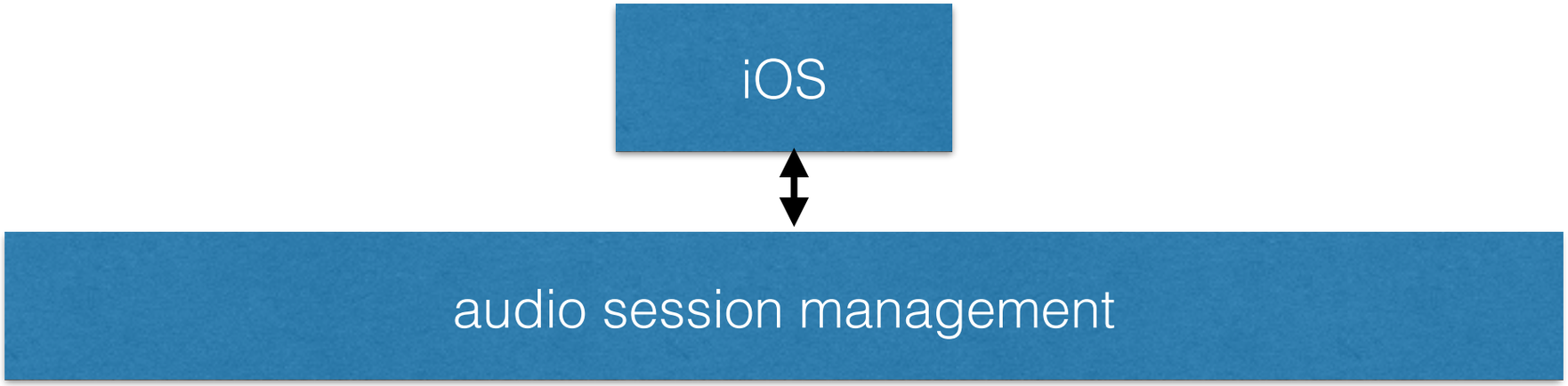
Core Audio

- Audio Sessions (completely overhauled for iOS7)
  - shared instance (for all applications)
  - set category (play, record, both)
    - choose options: like mixing with ambient sources
  - set audio route (new in iOS7)
    - set specific hardware within audio route
- Audio Units (output, input)
  - set stream format, buffer sizes, sampling rate,
  - initialize memory for audio buffers
  - set callback rendering procedure

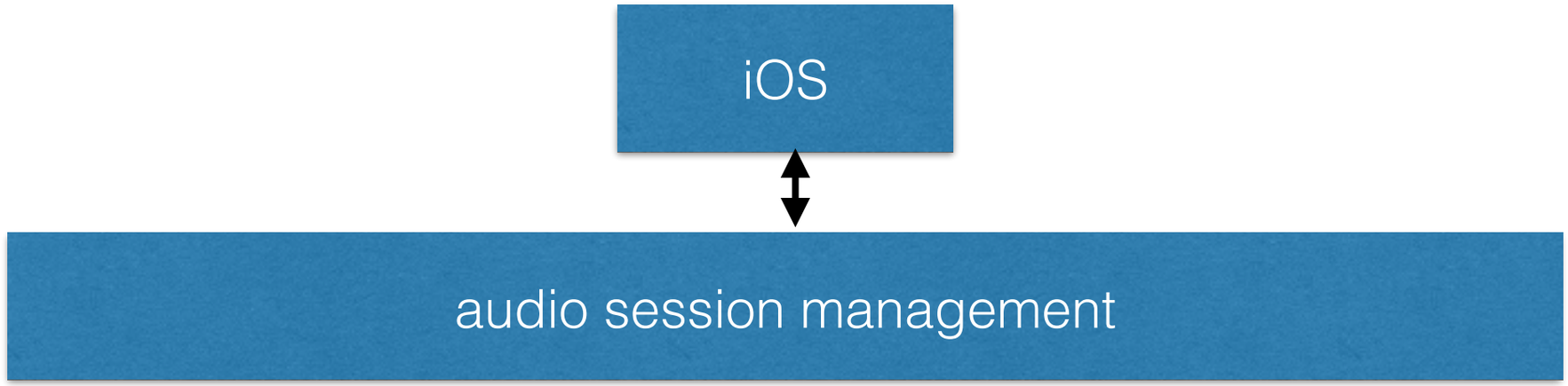
audio sessions

iOS

audio sessions

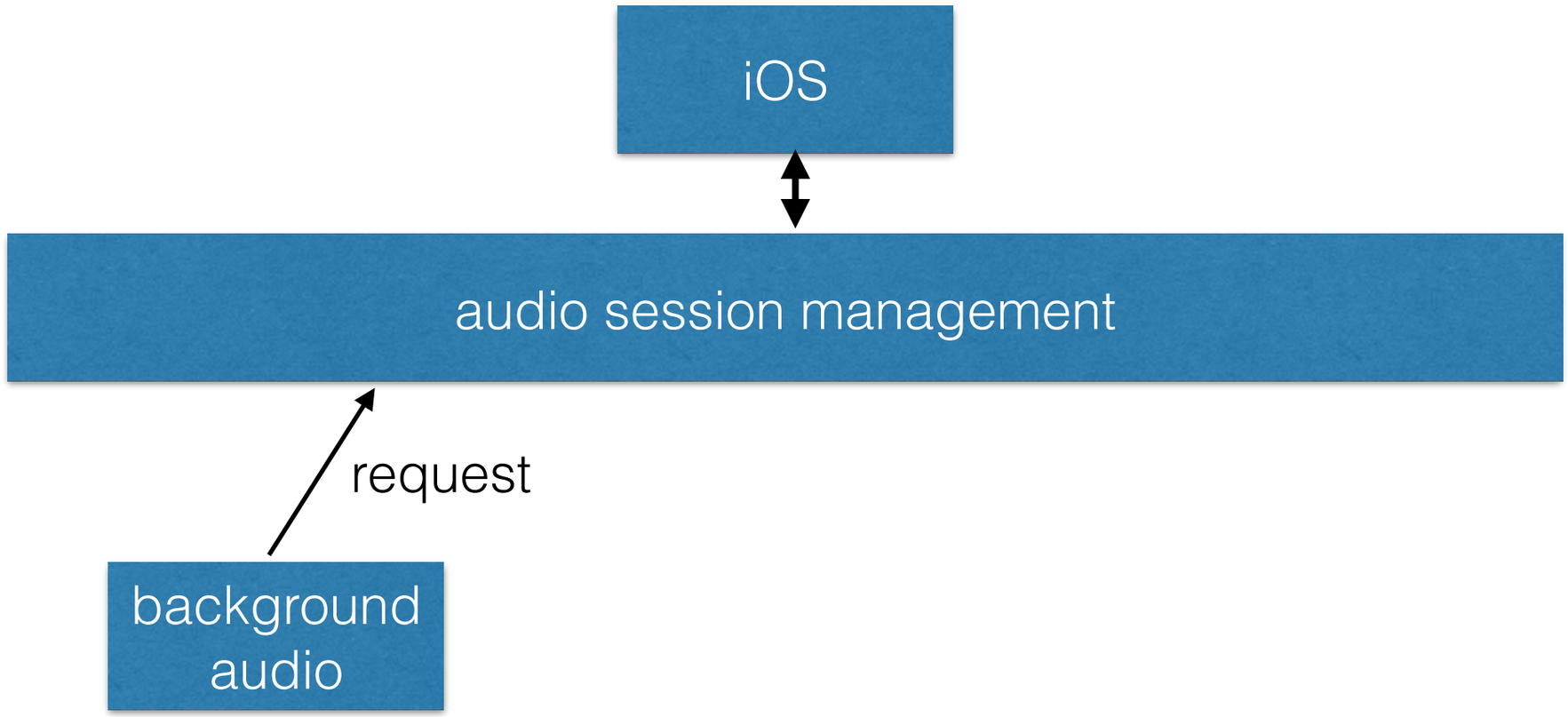


audio sessions

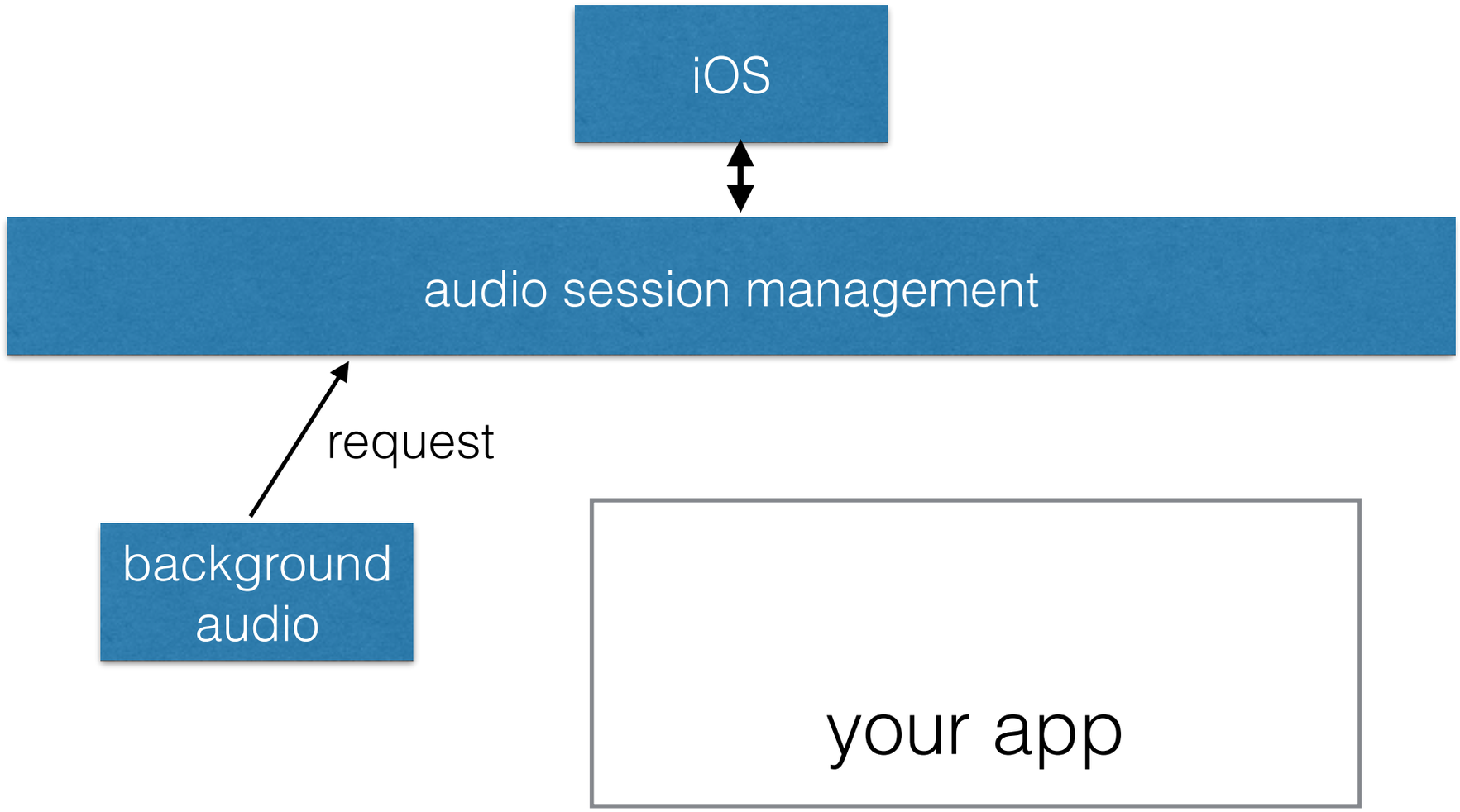


background audio

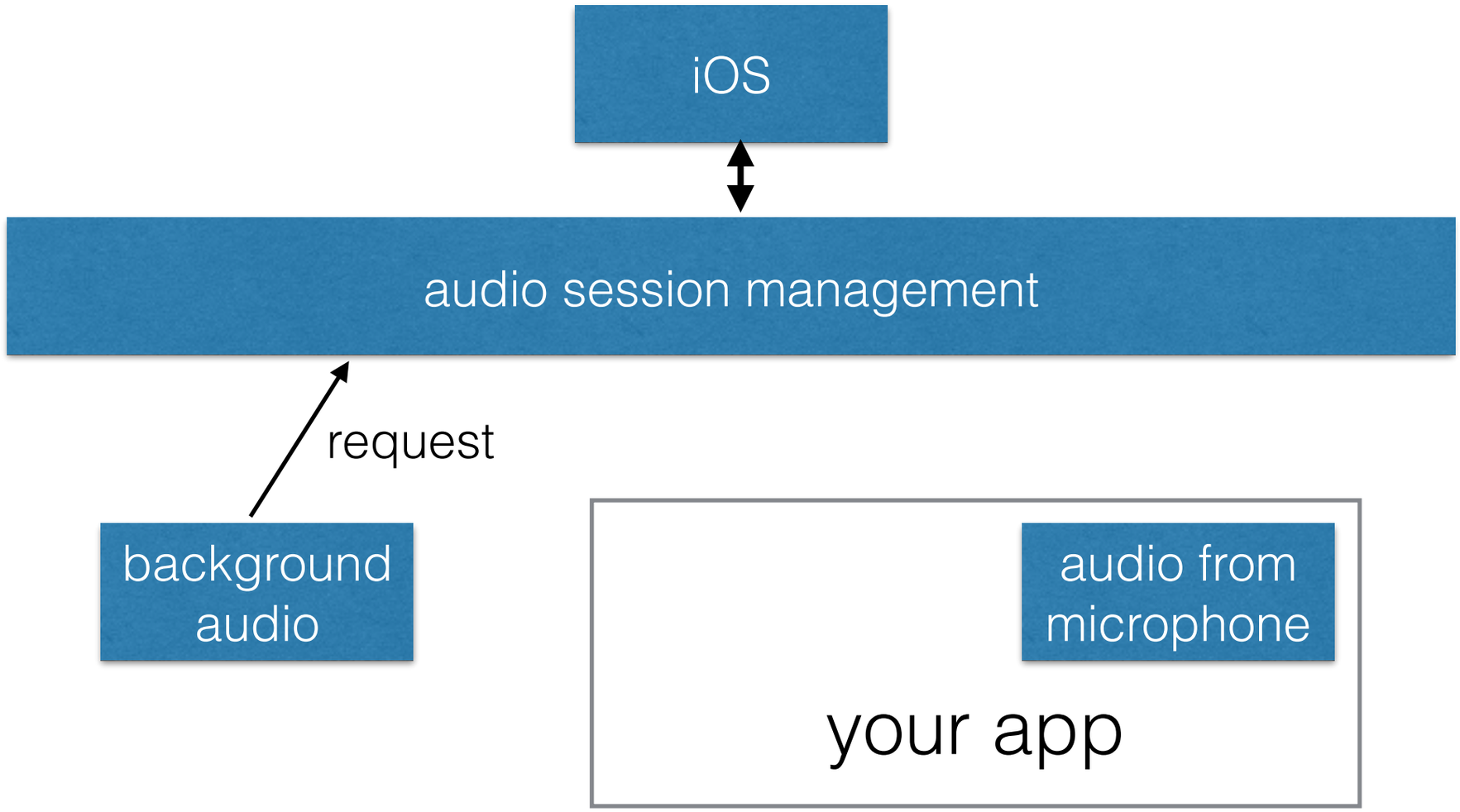
audio sessions



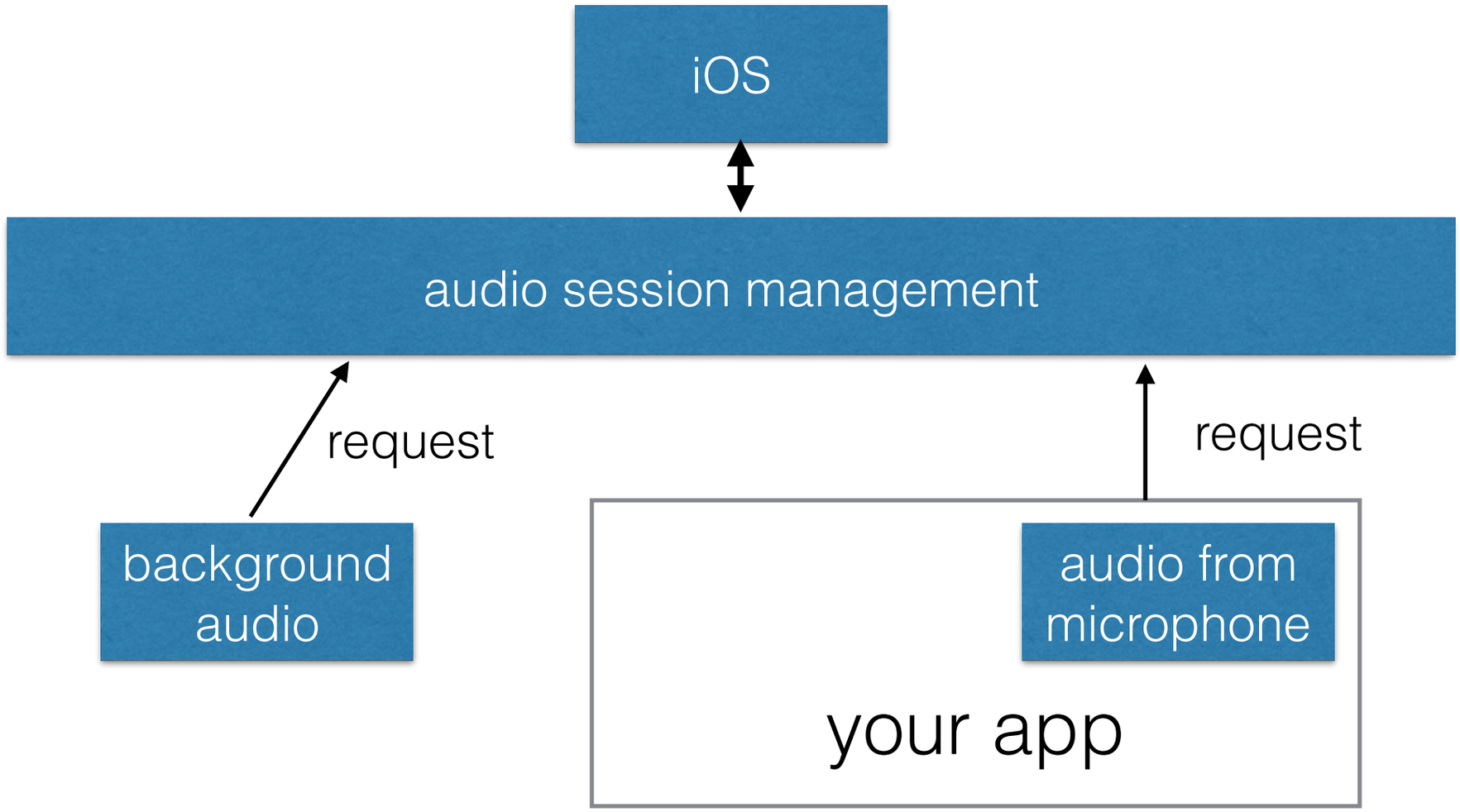
audio sessions



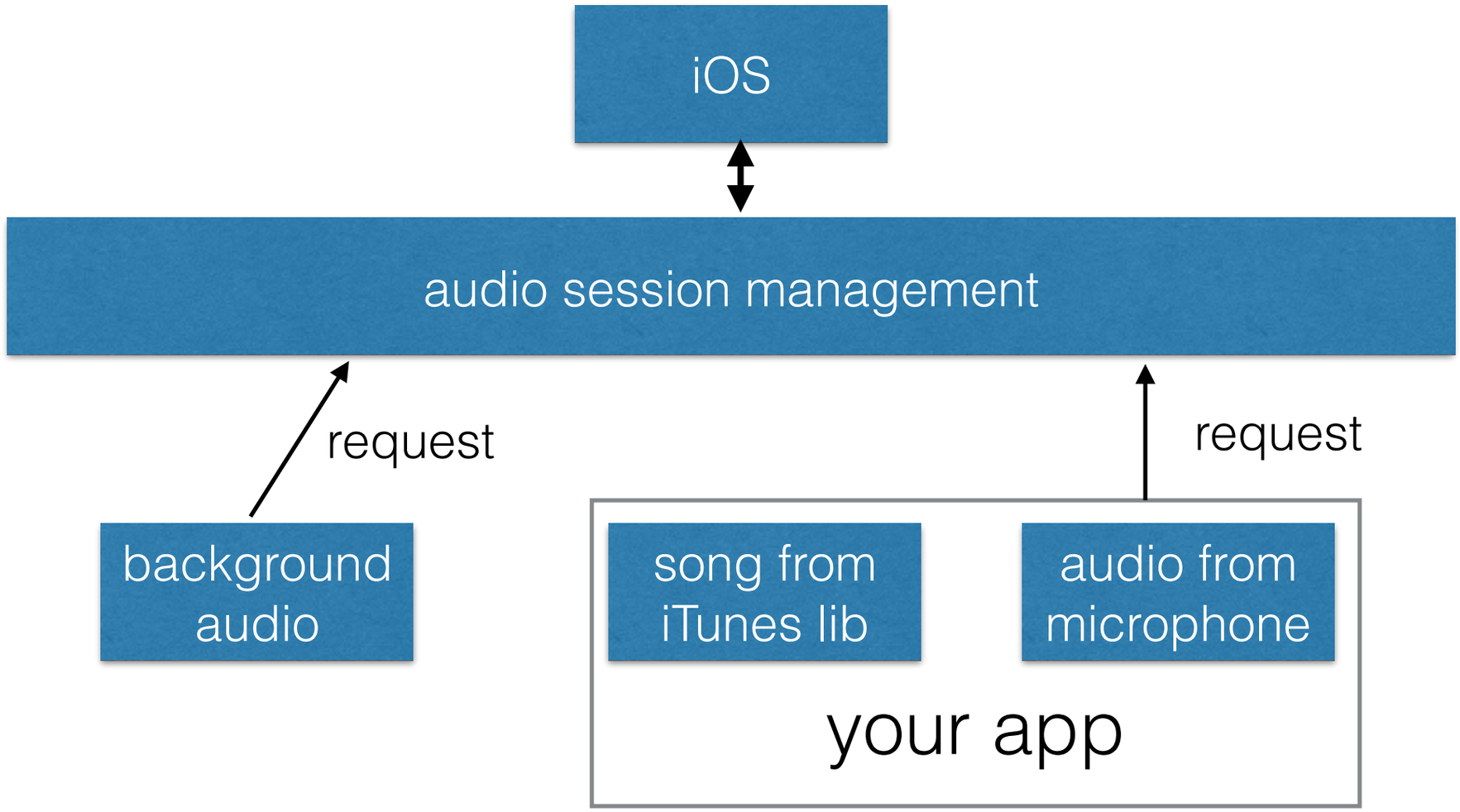
audio sessions



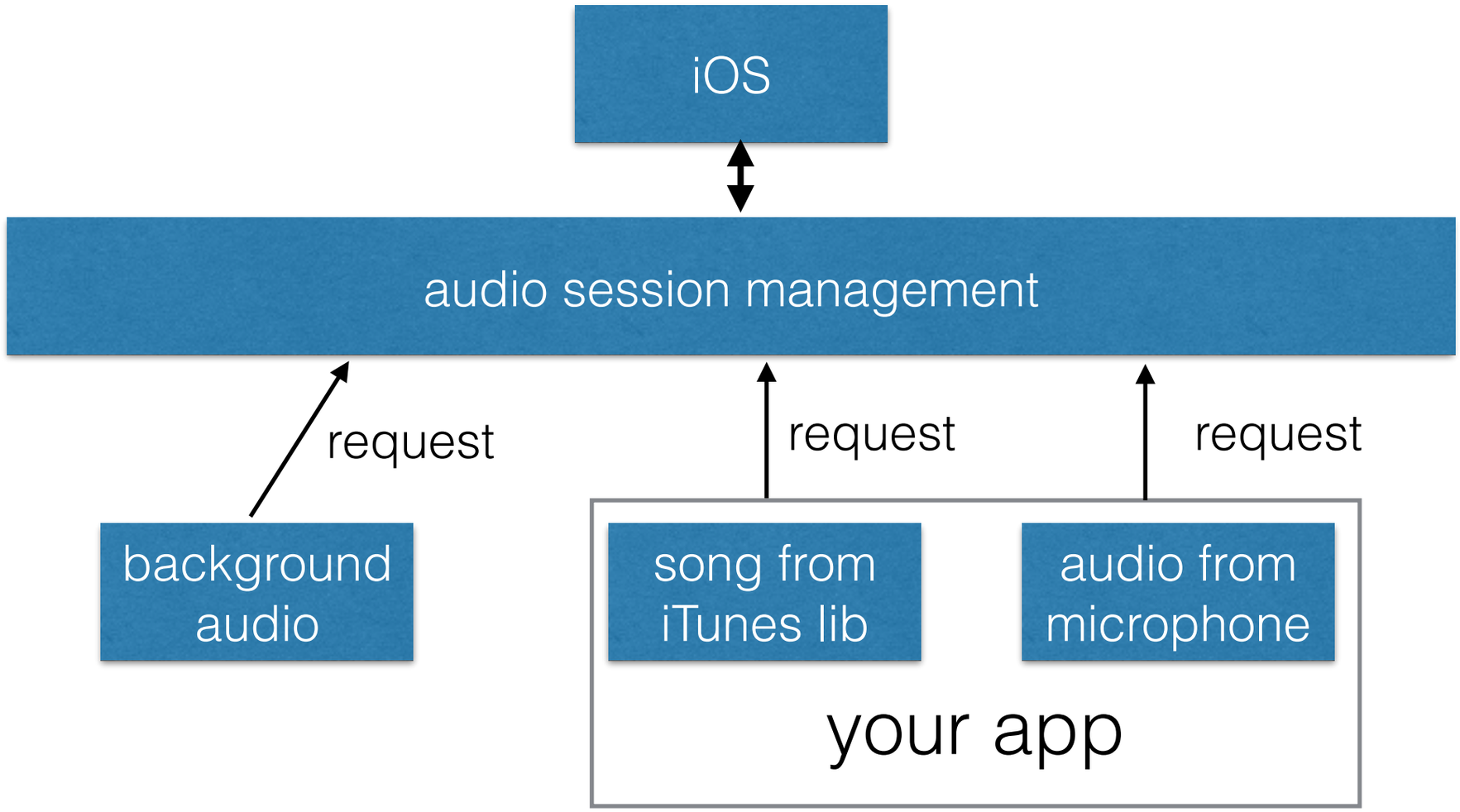
audio sessions



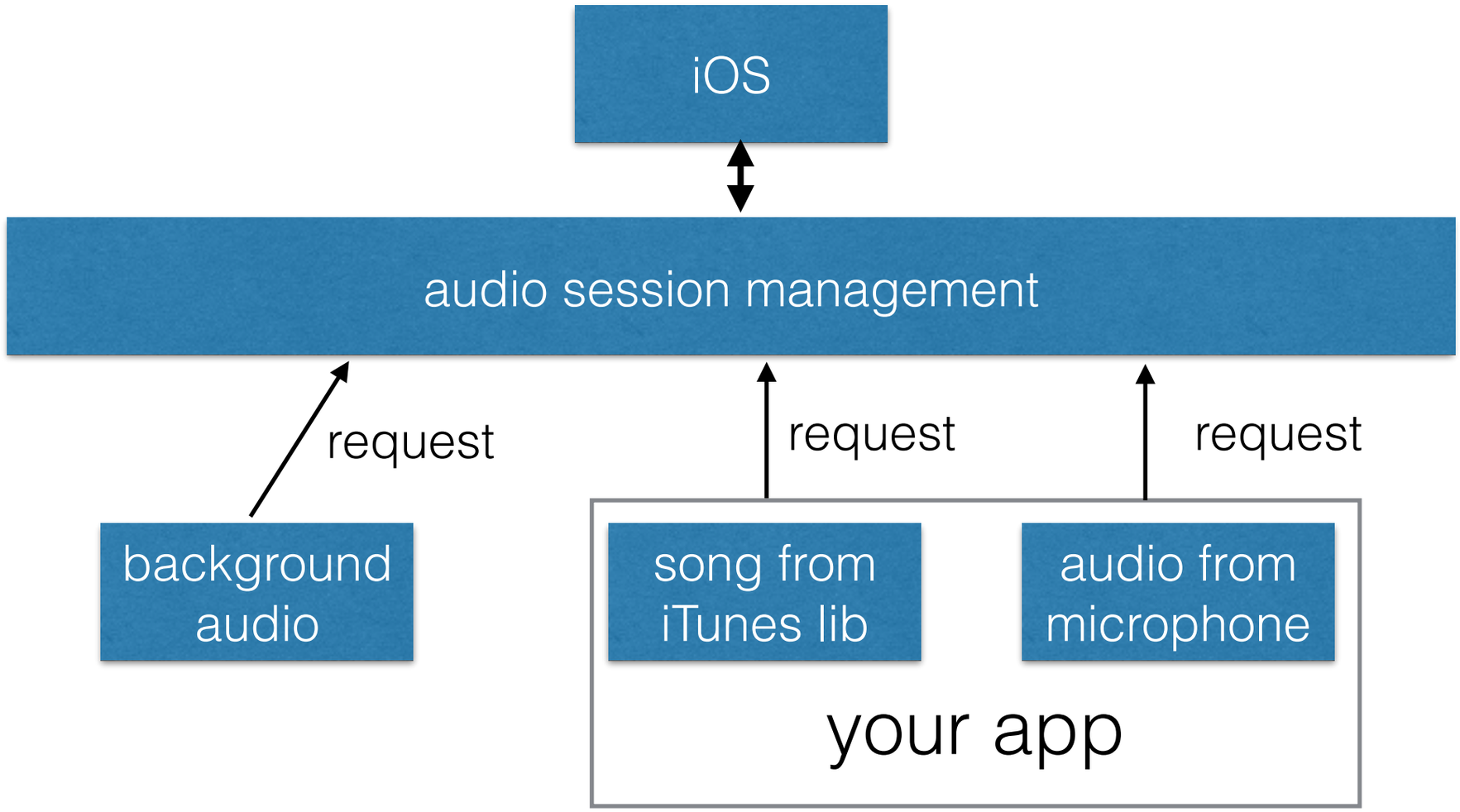
audio sessions



audio sessions

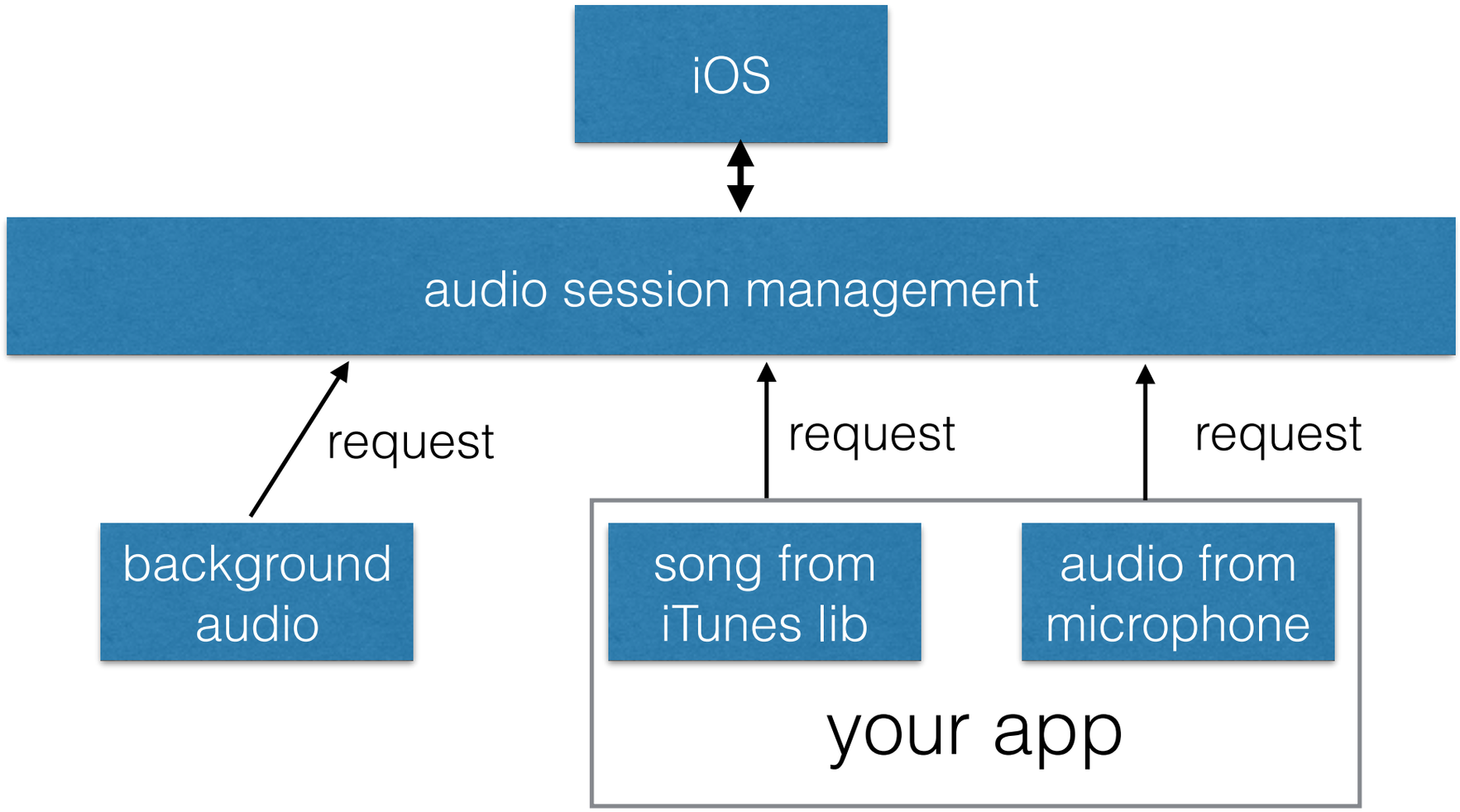


audio sessions



any request can alter management of other audio requests

audio sessions



any request can alter management of other audio requests

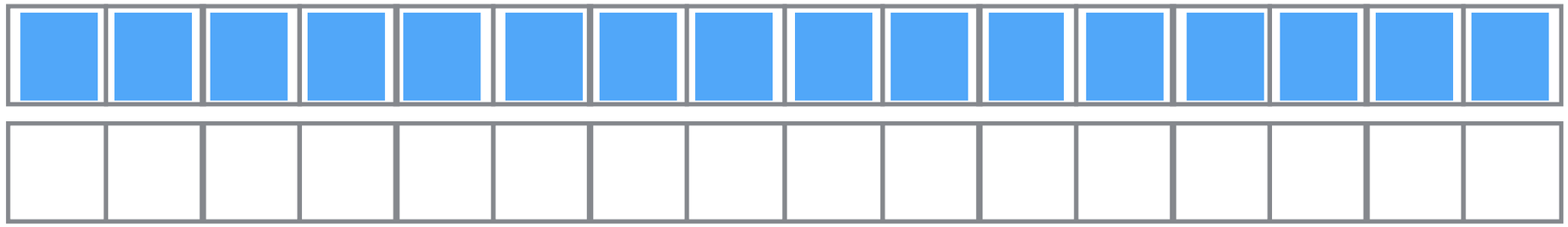
mixing can be automatic — this is impressive

audio units

audio input buffer

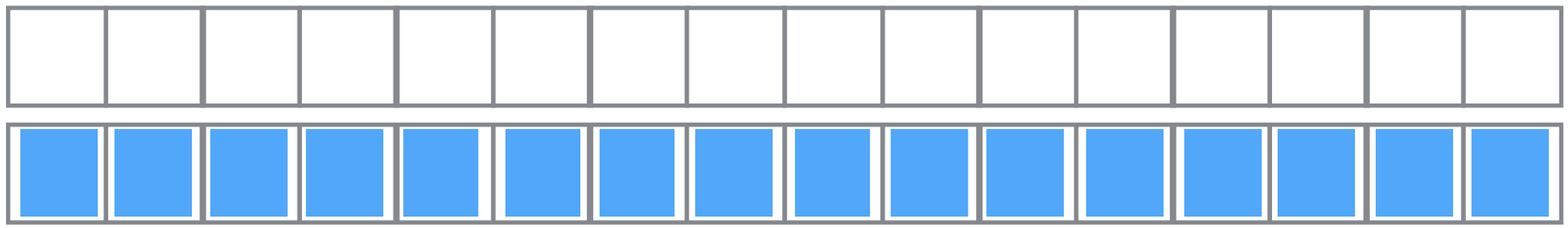
audio units

audio input buffer



audio units

audio input buffer



sent to audio session callback

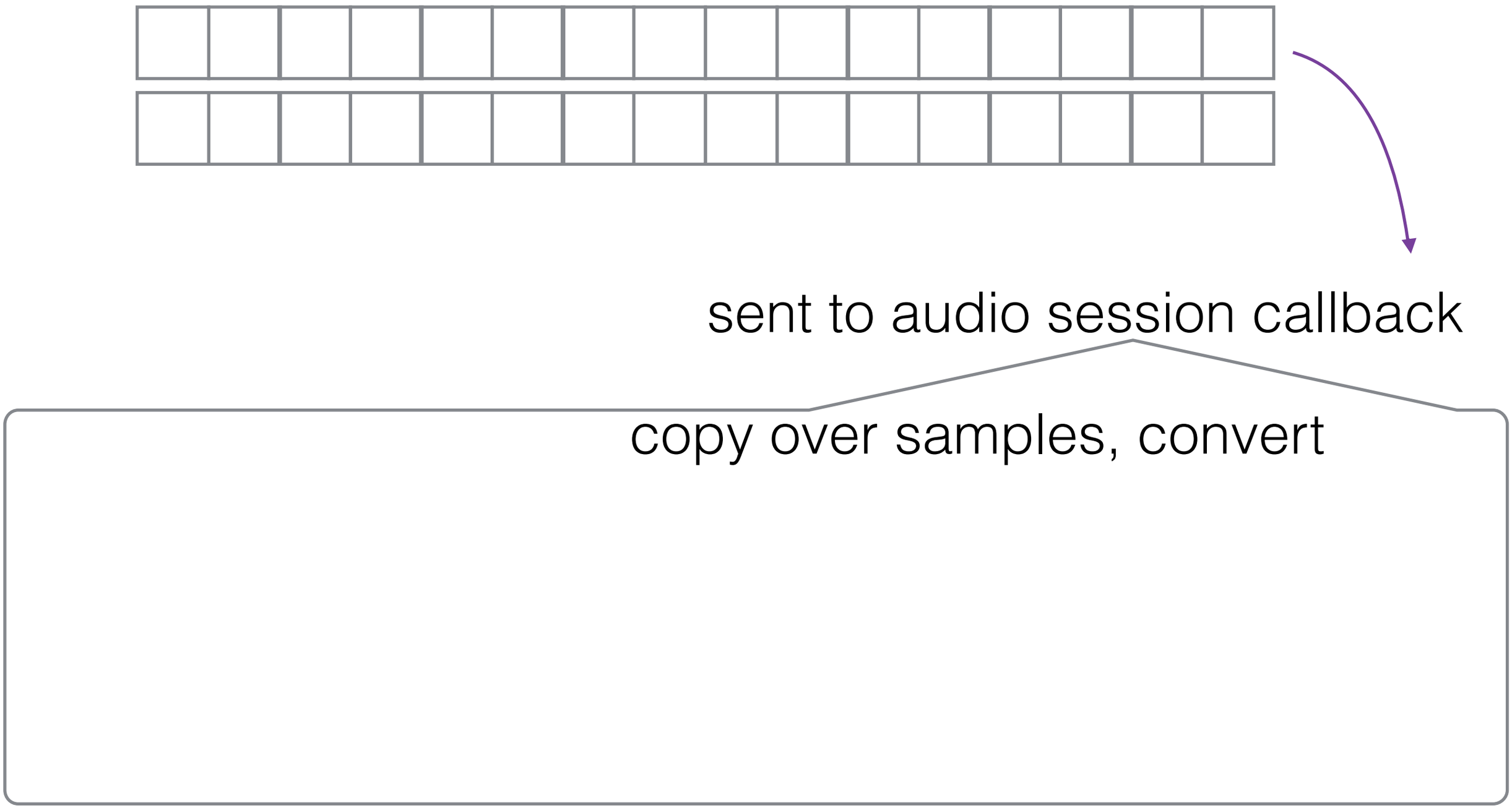
audio units

audio input buffer

sent to audio session callback

audio units

audio input buffer



audio units

audio input buffer

sent to audio session callback

copy over samples, convert

exit from call as soon as possible!

audio units

audio input buffer

sent to audio session callback

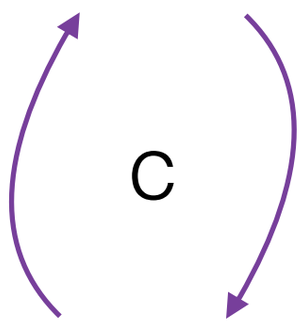
copy over samples, convert

exit from call as soon as possible!

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

audio units

audio input buffer



sent to audio session callback

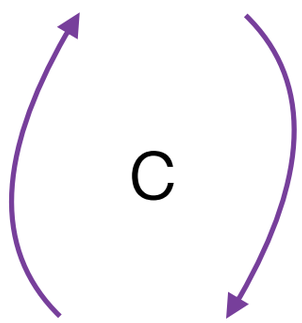
copy over samples, convert

exit from call as soon as possible!

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

audio units

audio input buffer



sent to audio session callback

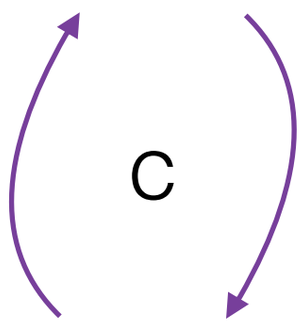
copy over samples, convert

exit from call as soon as possible!

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

audio units

audio input buffer

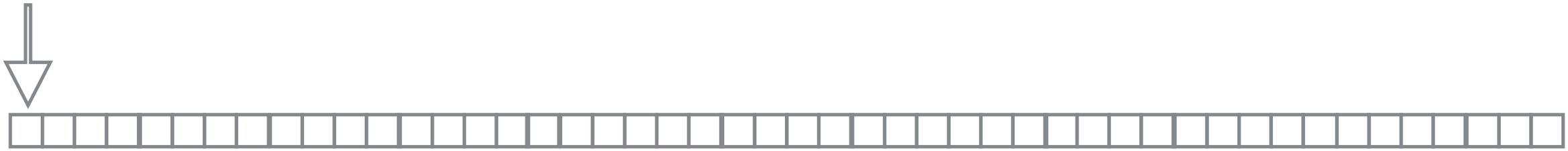


sent to audio session callback

copy over samples, convert

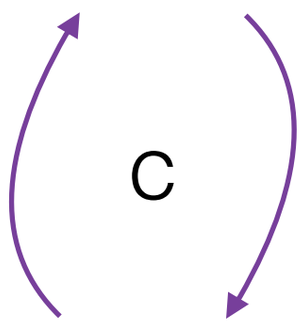
exit from call as soon as possible!

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



audio units

audio input buffer

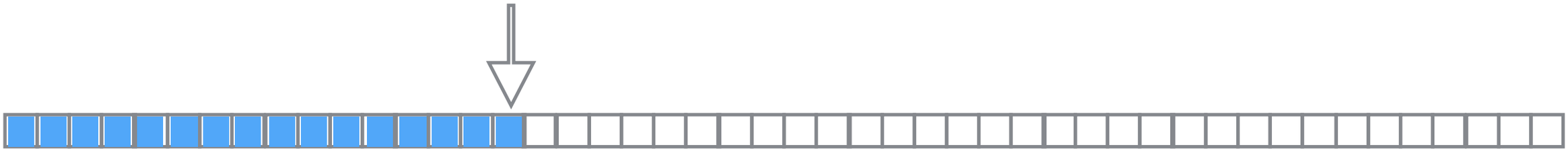


sent to audio session callback

copy over samples, convert

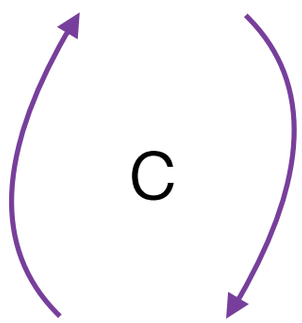
exit from call as soon as possible!

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



audio units

audio input buffer



sent to audio session callback

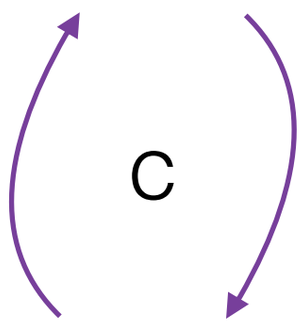
copy over samples, convert

exit from call as soon as possible!

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

audio units

audio input buffer



sent to audio session callback

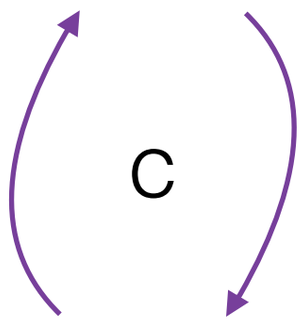
copy over samples, convert

exit from call as soon as possible!

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

audio units

audio input buffer



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

audio unit formats

microphone

stereo is interleaved

audio unit formats

microphone

stereo is interleaved

audio unit formats

microphone

stereo is interleaved

32 bits

audio made easy

- Novocaine (I mean real easy!!)

audio made easy

- Novocaine (I mean real easy!!)

```
Novocaine        *audioManager;
RingBuffer       *ringBuffer;

- (void)viewDidLoad

   audioManager = [Novocaine audioManager];
   ringBuffer = new RingBuffer(kBufferLength,1);
```

audio made easy

- Novocaine (I mean real easy!!)

```
Novocaine        *audioManager;
RingBuffer       *ringBuffer;

- (void)viewDidLoad

   audioManager = [Novocaine audioManager];
   ringBuffer = new RingBuffer(kBufferLength,1);
```

declare class

audio made easy

- Novocaine (I mean real easy!!)

```
Novocaine        *audioManager;
RingBuffer       *ringBuffer;
```

declare class

```
- (void)viewDidLoad

   audioManager = [Novocaine audioManager];
   ringBuffer = new RingBuffer(kBufferLength,1);
```

setup audio and init buffer

audio made easy

- Novocaine (I mean real easy!!)

```
Novocaine        *audioManager;
RingBuffer       *ringBuffer;
```

declare class

```
- (void)viewDidLoad

   audioManager = [Novocaine audioManager];
   ringBuffer = new RingBuffer(kBufferLength,1);
```

setup audio and init buffer

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

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



      }];



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



       }];
```

audio made easy

- Novocaine (I mean real easy!!)

```
Novocaine        *audioManager;
RingBuffer       *ringBuffer;
```

declare class

```
- (void)viewDidLoad

   audioManager = [Novocaine audioManager];
   ringBuffer = new RingBuffer(kBufferLength,1);
```

setup audio and init buffer

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

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



      }];



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



       }];
```

microphone samples as float array

audio made easy

- Novocaine (I mean real easy!!)

```
Novocaine        *audioManager;
RingBuffer       *ringBuffer;
```

declare class

```
- (void)viewDidLoad

   audioManager = [Novocaine audioManager];
   ringBuffer = new RingBuffer(kBufferLength,1);
```

setup audio and init buffer

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

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



      }];
```

microphone samples as float array

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



       }];
```

data to write to speakers

audio made easy

- Novocaine (I mean real easy!!)

```
Novocaine        *audioManager;
RingBuffer       *ringBuffer;
```

declare class

```
- (void)viewDidLoad

   audioManager = [Novocaine audioManager];
   ringBuffer = new RingBuffer(kBufferLength,1);
```

setup audio and init buffer

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

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

          ringBuffer->AddNewInterleavedFloatData(data, numFrames, numChannels);

      }];
```

microphone samples as float array

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




       }];
```

data to write to speakers

audio made easy

- Novocaine (I mean real easy!!)

```
Novocaine        *audioManager;
RingBuffer       *ringBuffer;
```

declare class

```
- (void)viewDidLoad

   audioManager = [Novocaine audioManager];
   ringBuffer = new RingBuffer(kBufferLength,1);
```

setup audio and init buffer

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

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

          ringBuffer->AddNewInterleavedFloatData(data, numFrames, numChannels);

      }];
```

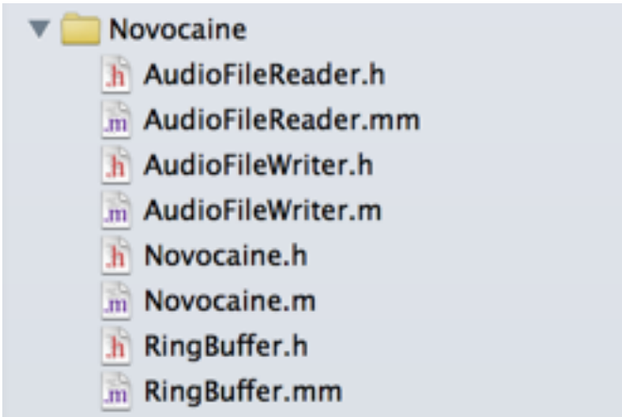
microphone samples as float array

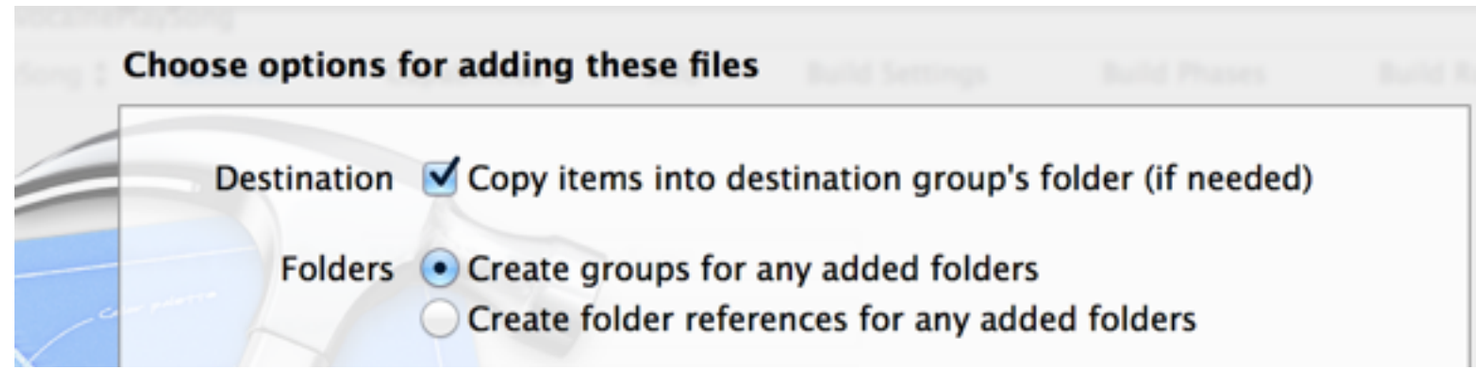
data to write to speakers

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

          ringBuffer->FetchInterleavedData(data, numFrames, numChannels);

       }];
```

novocaine uses c++

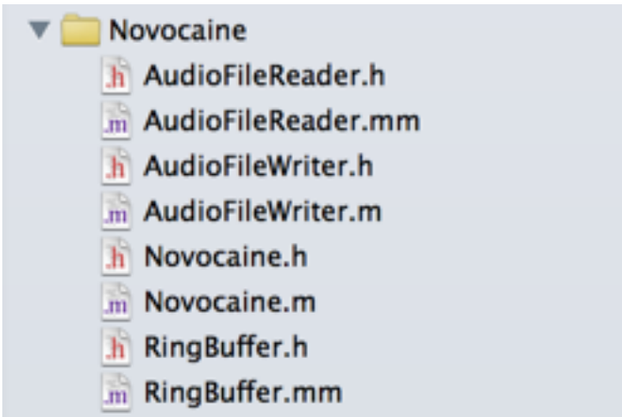
drag into project

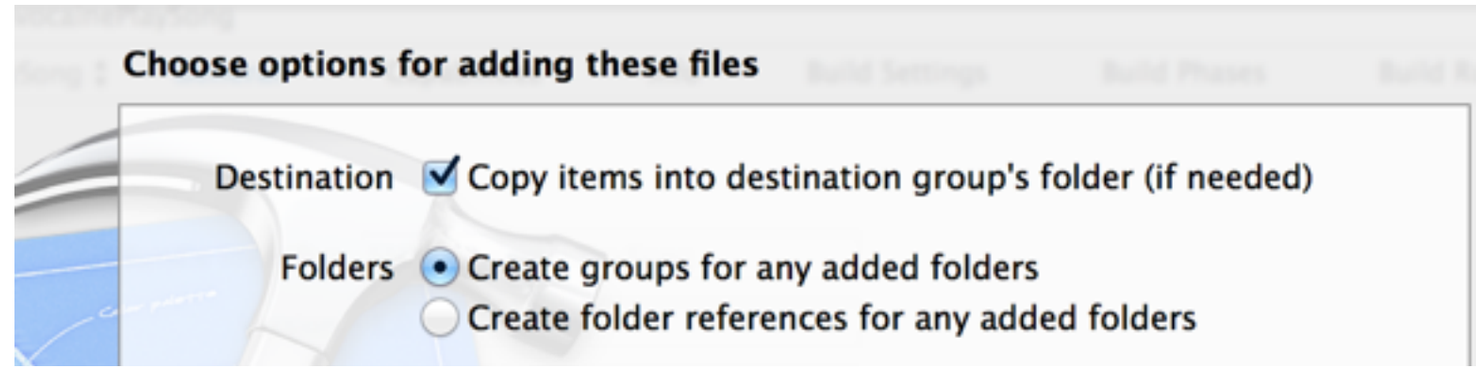


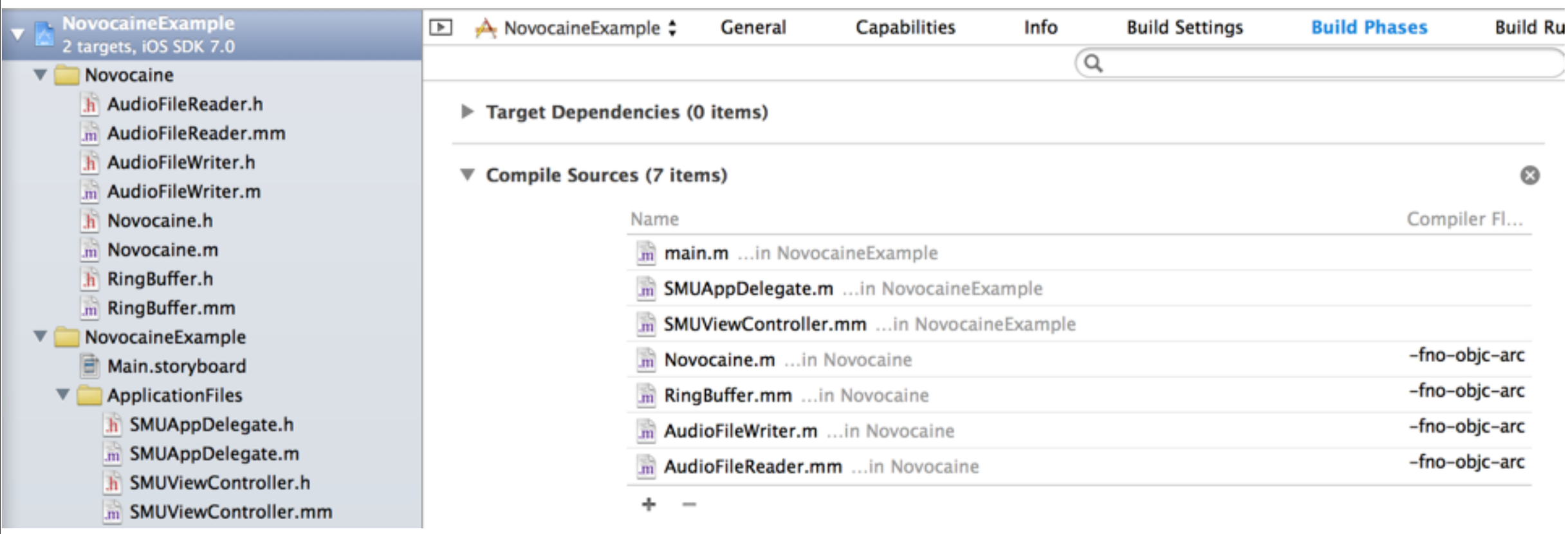


novocaine uses c++

drag into project

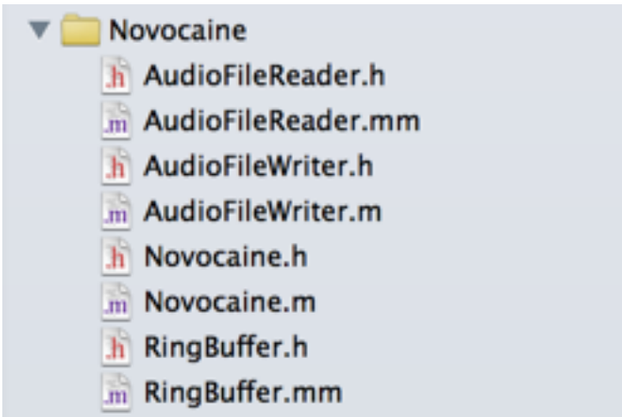


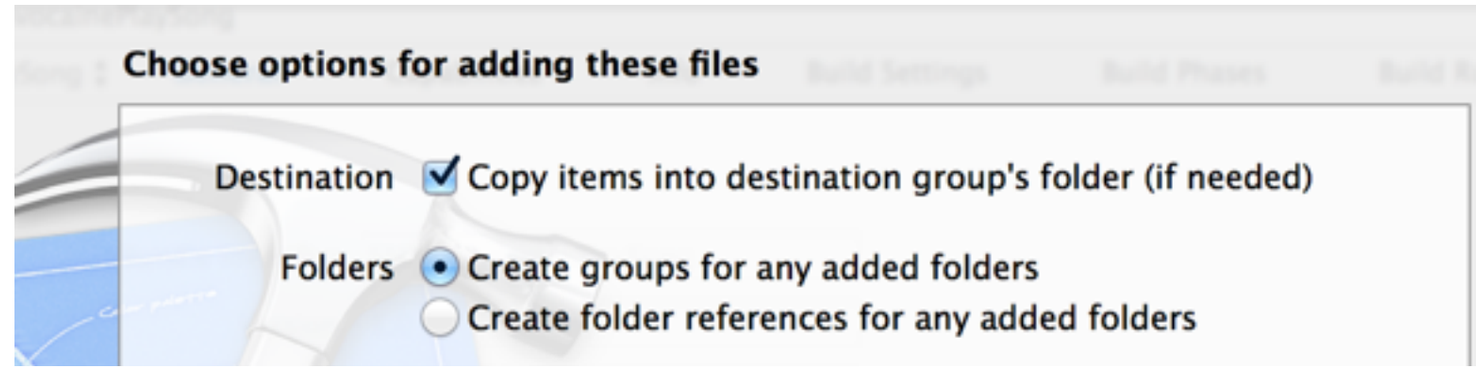




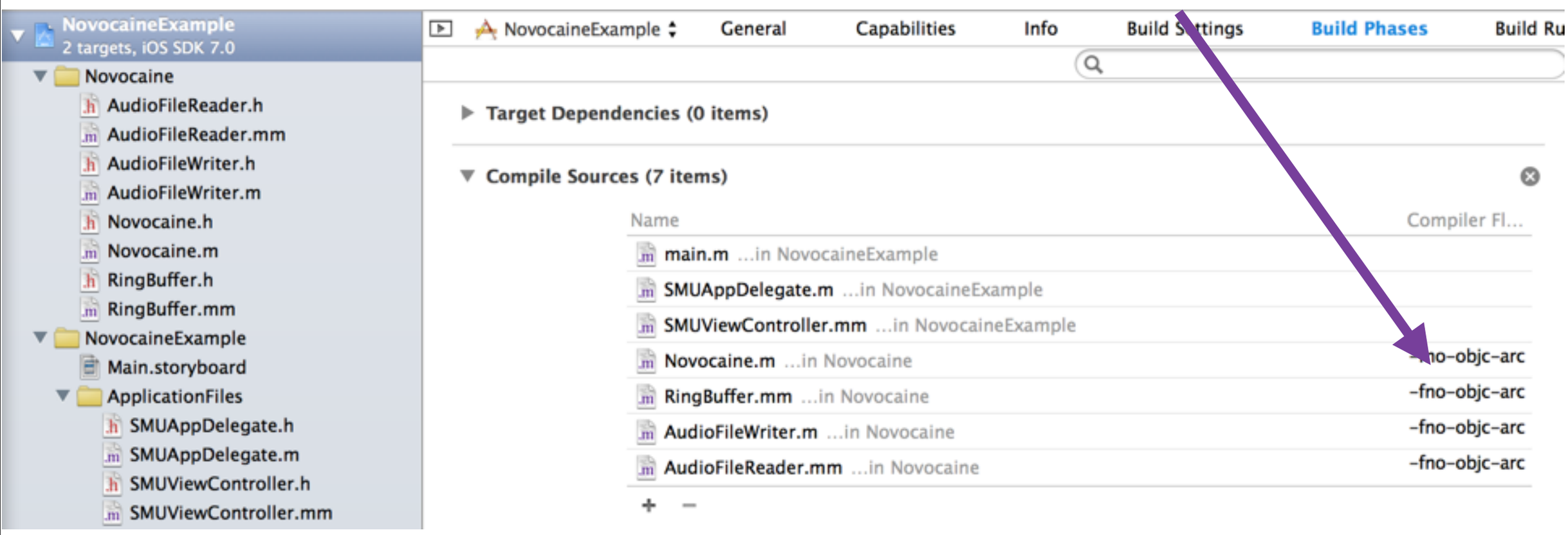
novocaine uses c++

drag into project



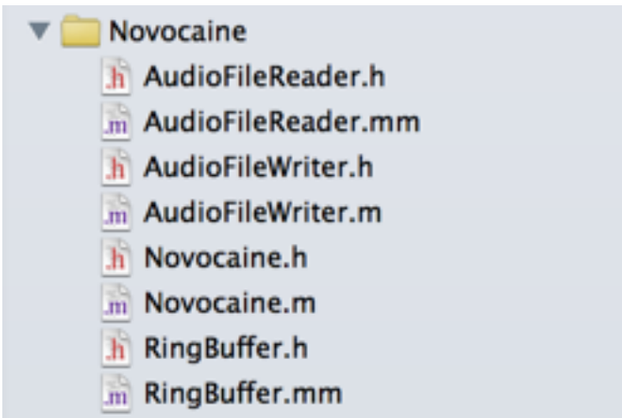


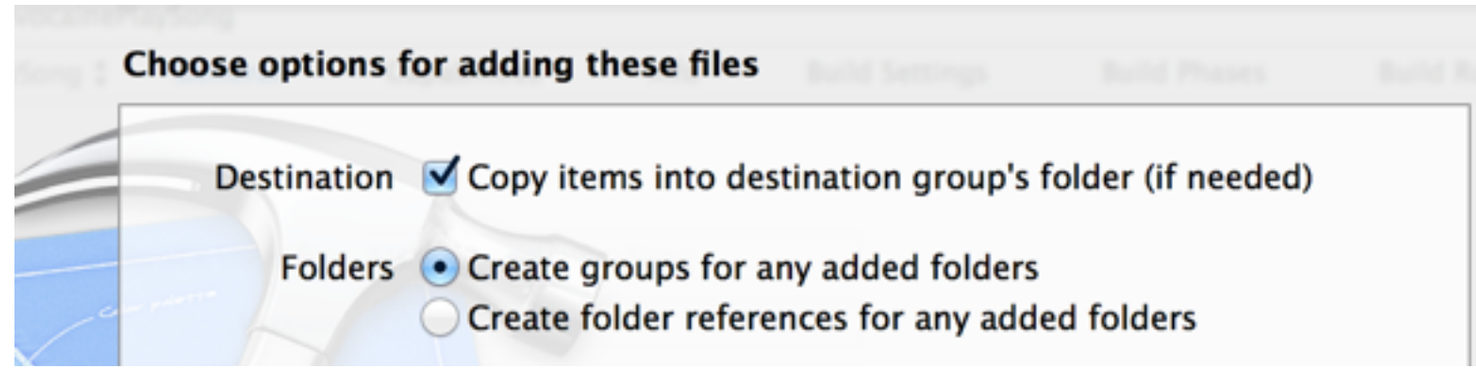
disable arc for these files



novocaine uses c++

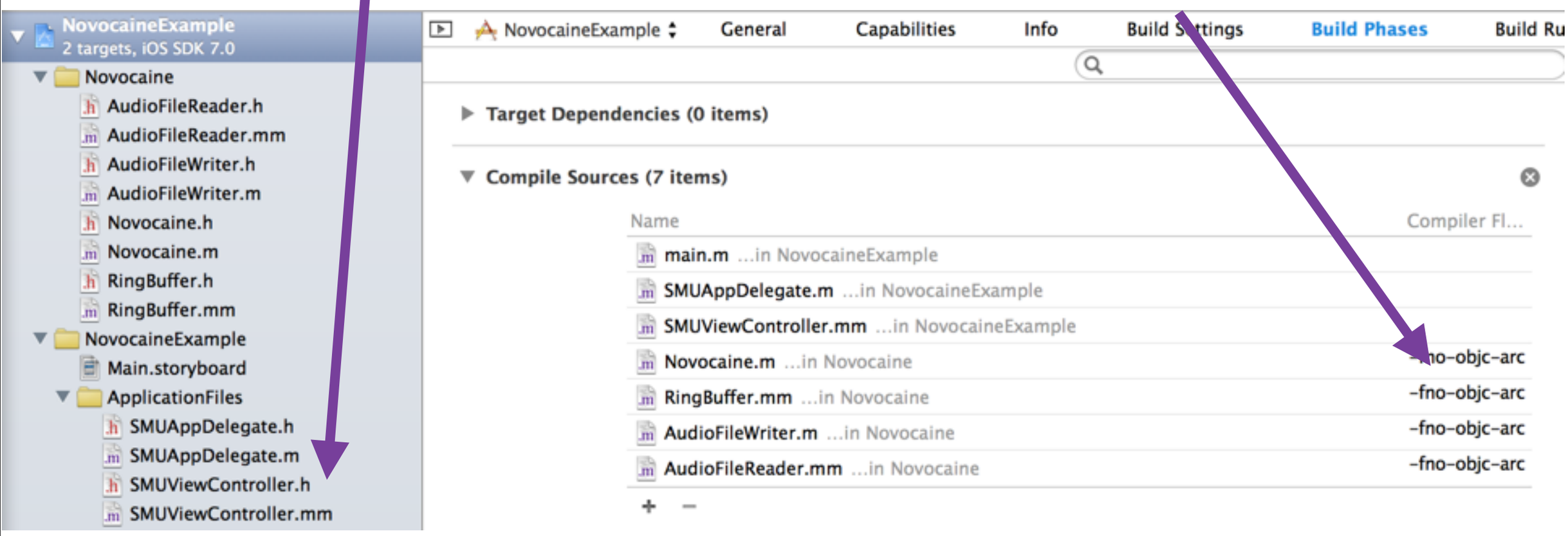
drag into project





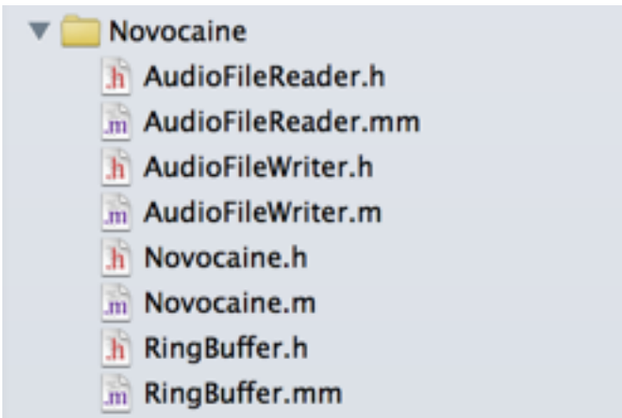
rename method to use c++

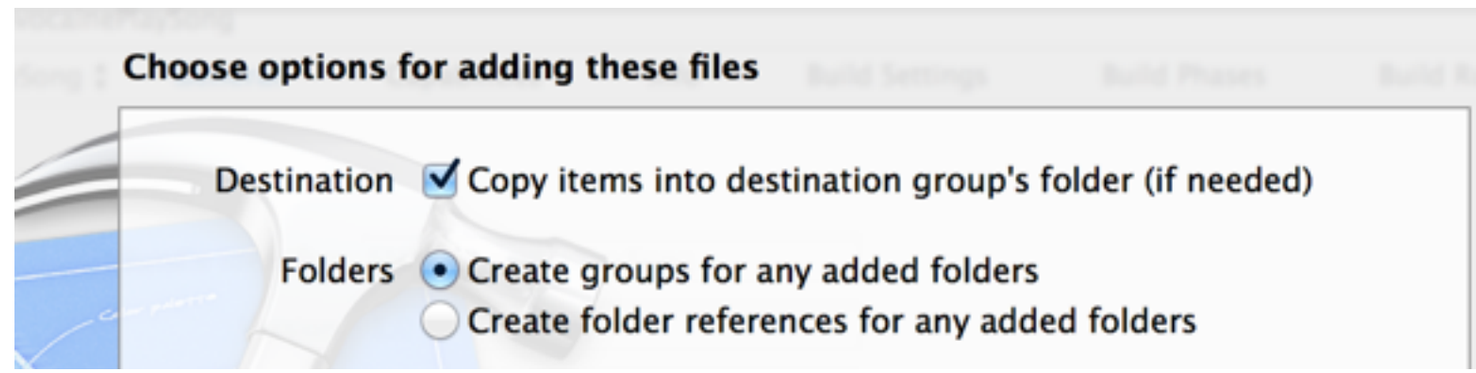
disable arc for these files



novocaine uses c++

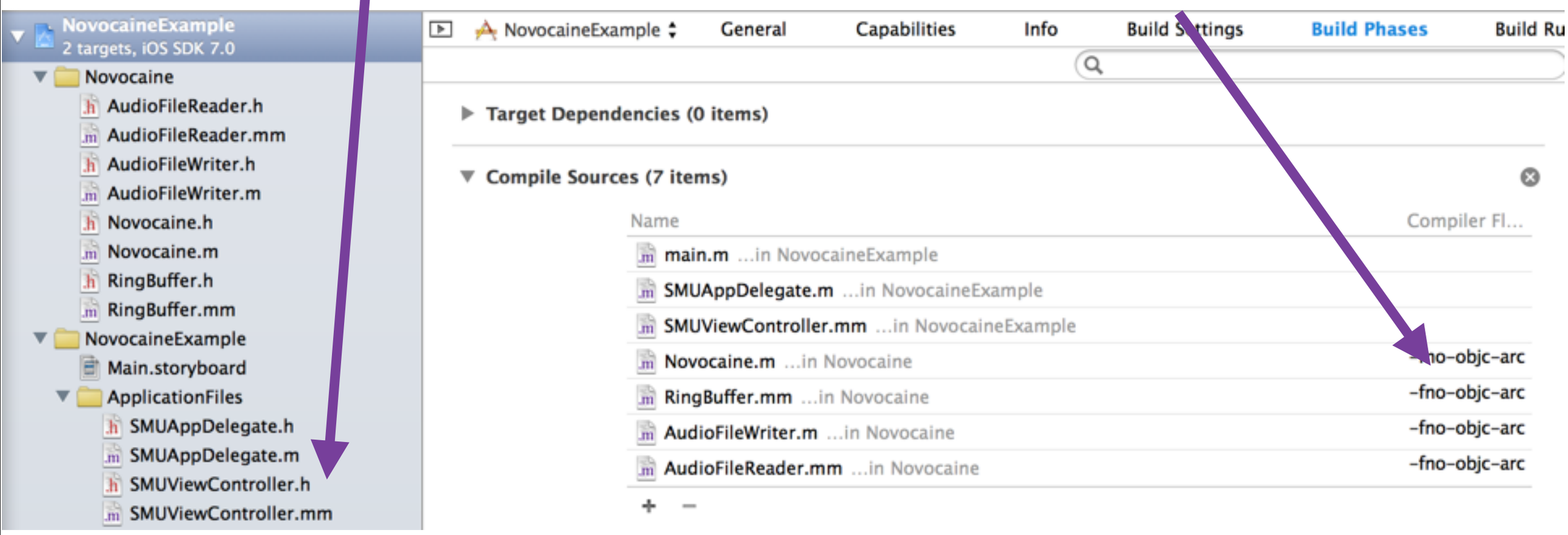
drag into project





rename method to use c++

disable arc for these files



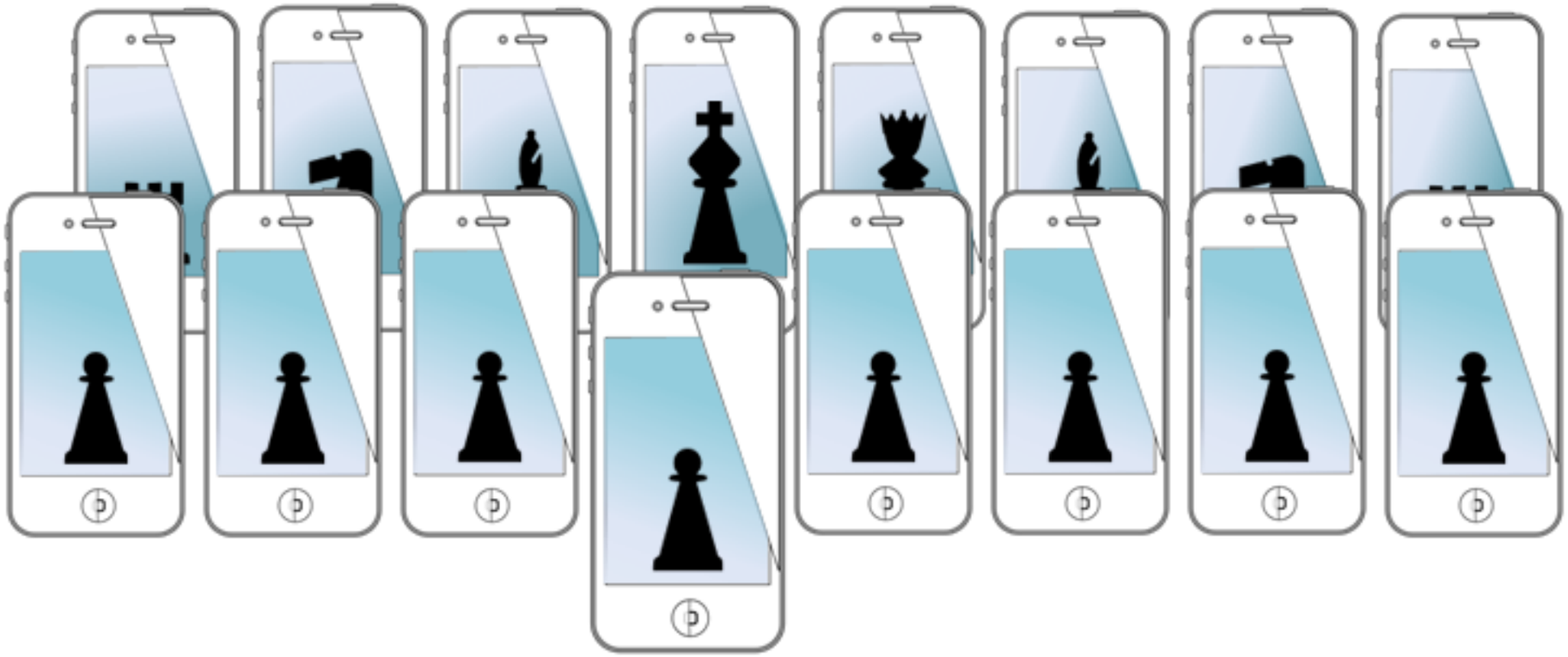
get source from blackboard!!

novocaine setup demo

for next time...

- more core audio
  - playing songs
  - getting samples from microphone
    - showing samples with OpenGL
  - working with sampled data
  - the accelerate framework

MOBILE SENSING LEARNING & CONTROL



CSE5323 & 7323

Mobile Sensing, Learning, and Control

lecture five: queues, blocks, c++, audio session

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