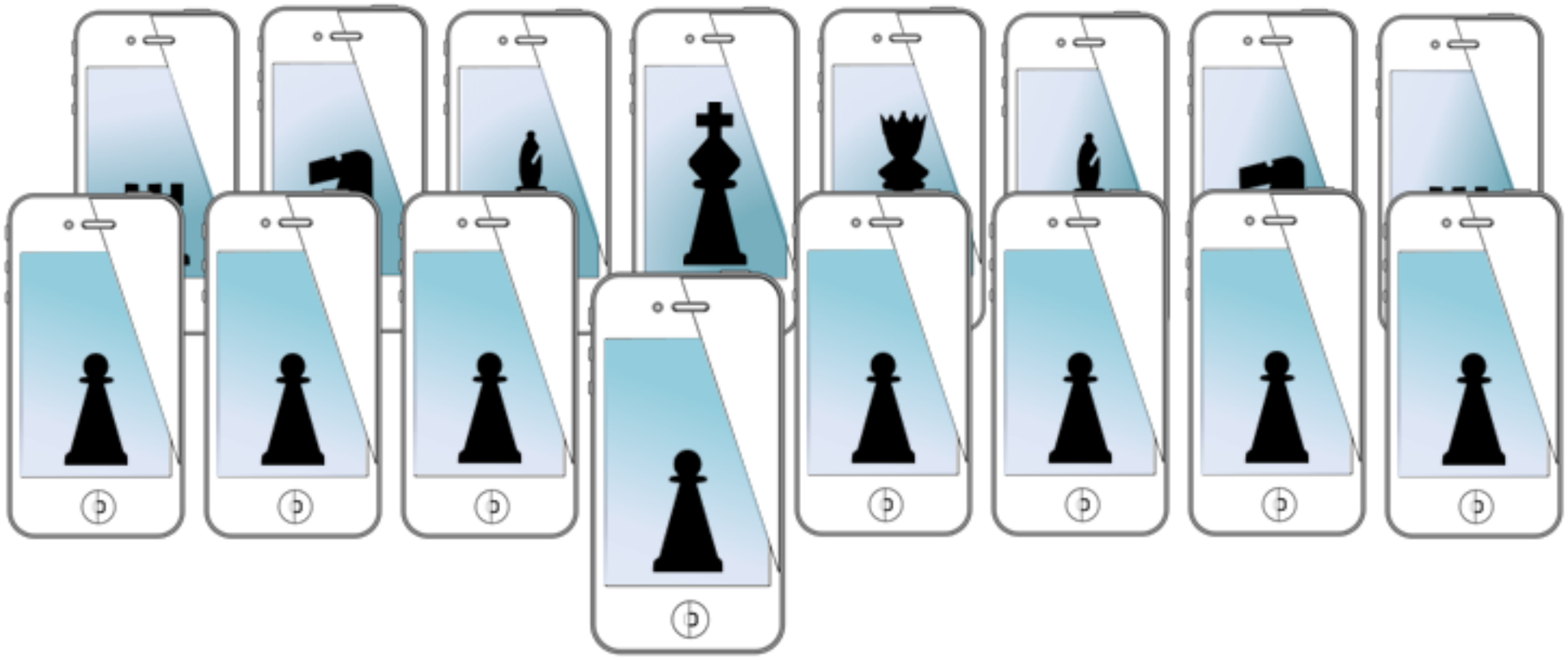
MOBILE SENSING & LEARNING



CS5323 & 7323

Mobile Sensing & Learning

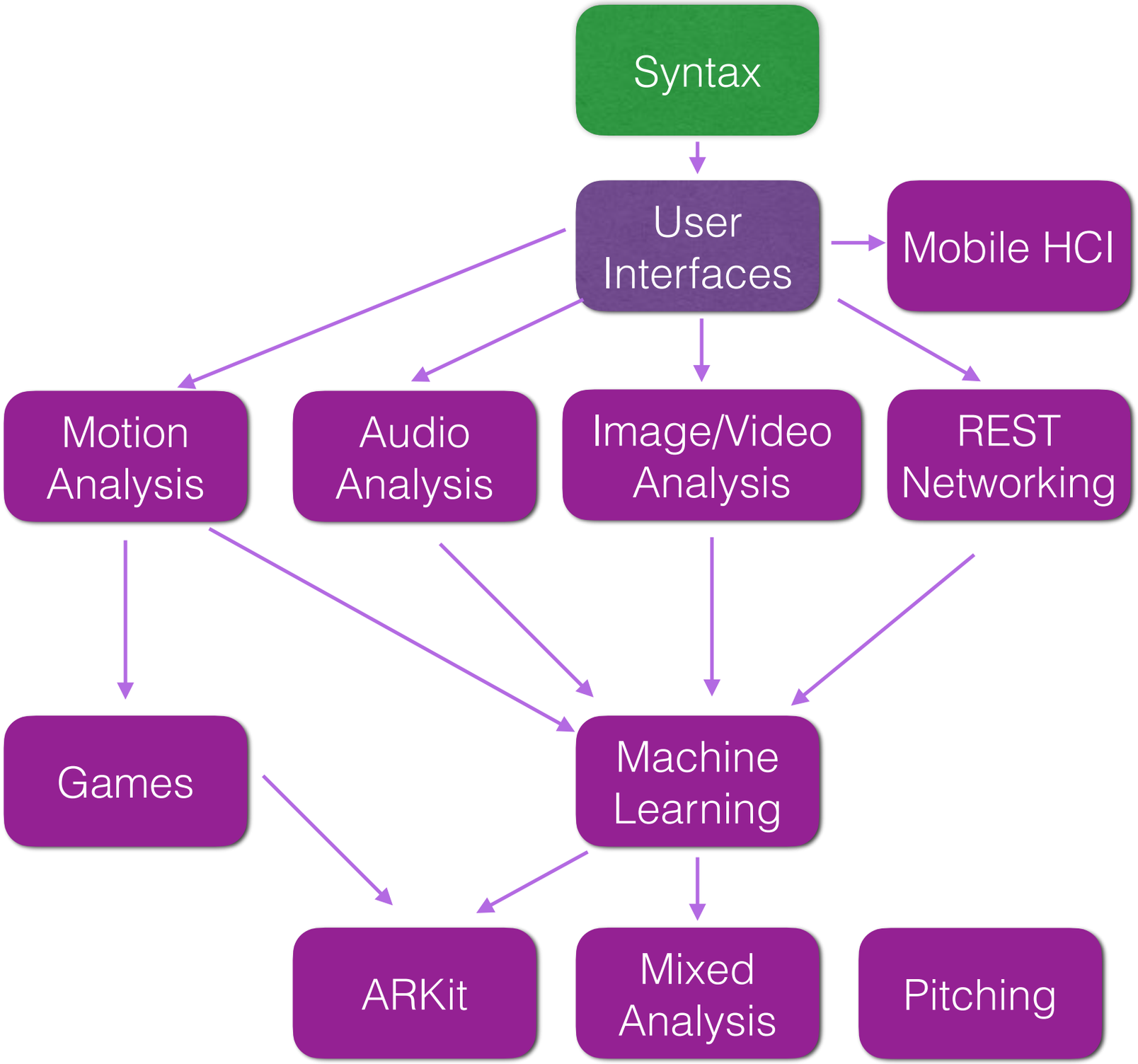
UI elements

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

course logistics

- reminder: developer program! (*re-send invitations*)
- next time: **flipped assignment**, in person/distance
- View Controllers in iOS via **flipped assignment**
  - Watch videos **before class**
  - download GitHub project and have running **before class**
  - come ready to work in teams on an in-class assignment
  - distance students: turn in within a few days of the assignment

class progression



agenda

- syntax review
- blocks and concurrency
- target action behavior
  - and constraints
- text fields
- gesture recognizers
- timers / segmented control
- **remainder of time**: demo!

review

```objc
@interface SomeViewController ()

@property (strong, nonatomic) NSString *aString;
@property (strong, nonatomic) NSDictionary *aDictionary;

@end

@implementation SomeViewController
@synthesize aString = _aString;

-(NSString *)aString{
    if(!_aString)
        _aString = [NSString stringWithFormat:
                    @“This is a string %d",3];
    return _aString;
}

-(void)setAString:(NSString *)aString{
    _aString = aString;
}

- (void)viewDidLoad
{
    [super viewDidLoad];

    self.aDictionary = @{@"key1":@3,@"key2":@"a string"};
    for(id key in _aDictionary)
        NSLog(@"key=%@, value=%@",key,_aDictionary[key]);


    NSArray *myArray = @[@32,@"a string", self.aString ];
    for(id obj in myArray)
        NSLog(@“Obj=%@",obj);


}
```

private properties

backing variable

getter

setter

call from super class

dictionary iteration

array iteration

[C]

```swift
class SomeViewController: UIViewController {

    private lazy var aString = {
       return "This is a string \(3)"
    }()

    private var aDictionary:[String : Any] = [:]



    override func viewDidLoad() {
        super.viewDidLoad()

        self.aDictionary = ["key1":3, “key2":
                    "String value"] as [String : Any]

        for (_,val) in self.aDictionary {
            print(val)
        }



        let myArray: [Any] = [32,"a string”,
                              self.aString]
        for val in myArray{
            print(val)
        }

    }
}
```

private properties

call from super class

dictionary iteration

array iteration

optional versus implicit

- optional values can be nil or a specific datatype
  - many properties of a swift class will therefore be optional
- lazily instantiated properties must be optional (no init needed), but `if let` syntax is cumbersome if used for all properties every time
- enter the world of implicitly unwrapped optionals!
- do not check if nil, go to the data … runtime error if nil…

```
@IBOutlet weak var implictUnwrapLabel: UILabel!
@IBOutlet weak var optionalLabel: UILabel?
@IBOutlet weak var label: UILabel

implicitUnwrapLabel.text = “No need to unwrap with ?”
optionalLabel?.text = “must unwrap”

if let label = optionalLabel {
       label.text = “safely unwrapped in if let”
}
```

why can we not do this?

adding to our project

- let's add to our project
  - an objective-c class
  - that uses lazy instantiation

blocks and closures

- a block of code that you want to run at another time and perhaps pass to other classes to run
  - created at runtime
  - acts like an object that can be passed as an argument or created on the fly
  - once created, can be called repeatedly
  - can access variables from scope where defined
  - syntax is slightly different in swift and objective-c
  - common to define when calling a method that uses block
- swift calls these **closures**, objective-c says **blocks**

block/closure syntax

most common usage is as input into a function

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

this variable is in scope of block!

```
NSNumber *objInScope = @(32)
// 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]);
    NSLog(@“Scope Variable = %.2f",[objInScope floatValue]);
}];
```

swift syntax

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

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

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

Also valid if closure is last input

some semantics

optional content

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

```
NSNumber * objInScope = @5.0;
```

- Unless you use keyword (now mutable):

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

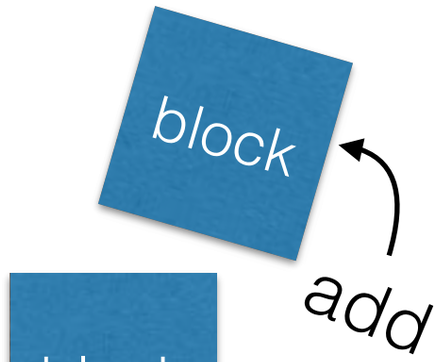
- classes hold a **strong** pointer to blocks they use
- blocks hold a **strong** pointer to `__block` variables
  - so using “self” would create a retain cycle

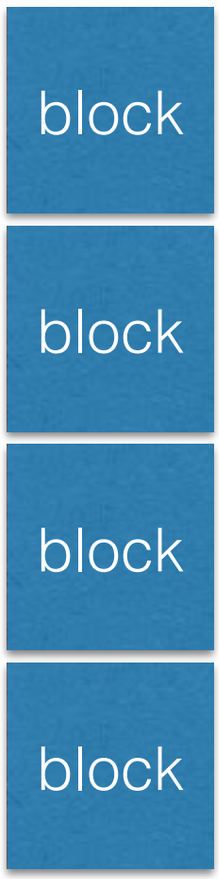
```
self.value = (some function in block)

__block ViewController * __weak  weakSelf = self;

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

concurrency in iOS *optional content*

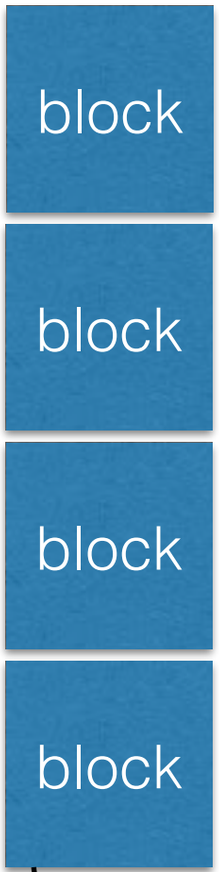
- grand central dispatch (GCD) handles all operations
  - GCD looks at “queues” of **blocks** that need to be run
  - GCD and the Xcode compiler work deep inside the OS, actually in the kernel — they are optimized
  - for a **serial queue** each block is run sequentially
  - for **concurrent queues** 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

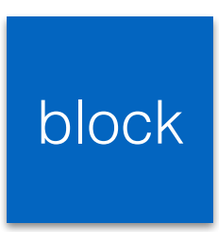




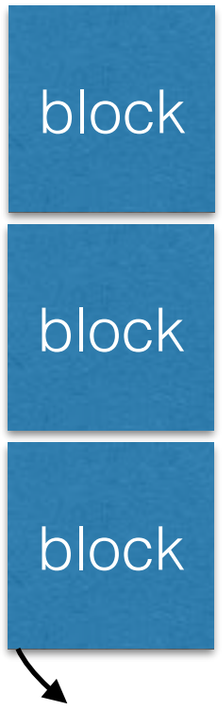
process



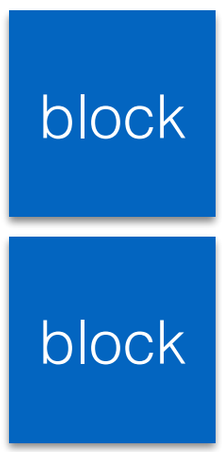
process (if cpu)



processing



process (if cpu)



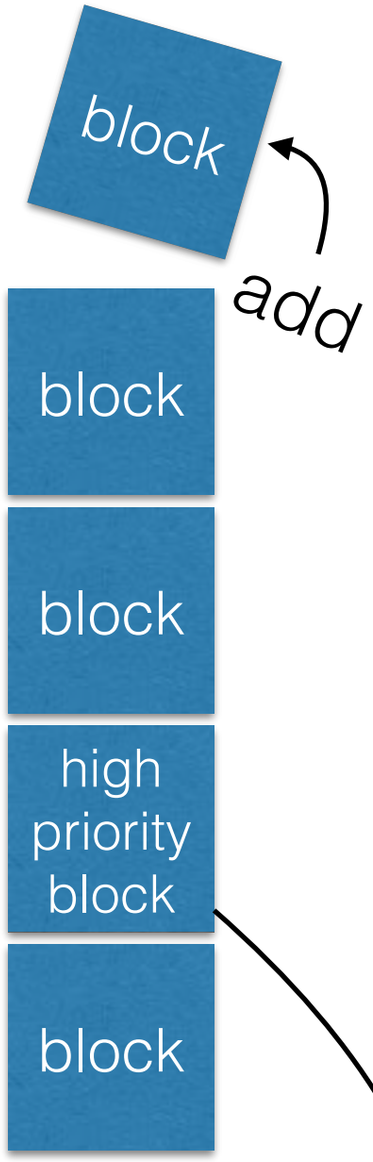
processing

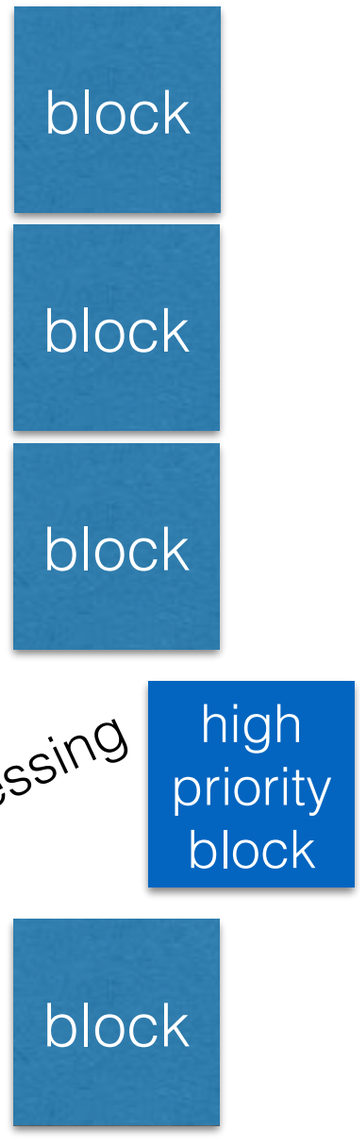
threads?

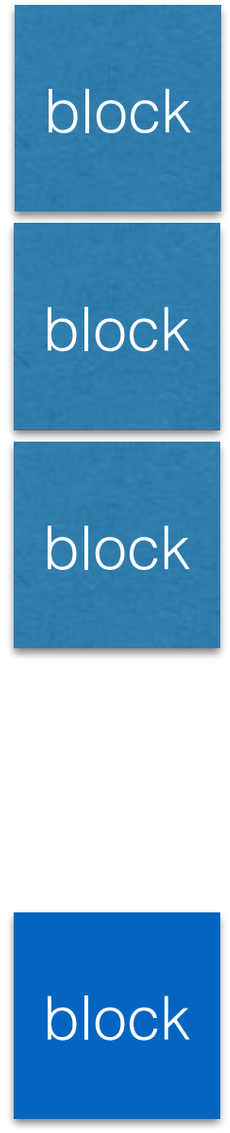
either could finish

GCD:

the main queue







one thread
UI is in sync

GCD:

process

process next block after completion

create your own queue!

```objc
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, just printing but
        imagine I am doing something time consuming, like loading something from the internet”);

    // 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];
    });

}];
```

create new queue

define block

update UI, another block

```swift
var queue:DispatchQueue = DispatchQueue(label: "mySwiftQueue")
queue.async {
    //code to execute in block
    for _ in 0..<3{
        print(" I am being executed from a custom queue")
    }
    // now we go to the main queue
    DispatchQueue.main.async {
        print("Running from main queue!")
    }
}
```

same functionality,
update UI, another block

common queues

- using global queues

access a global queue

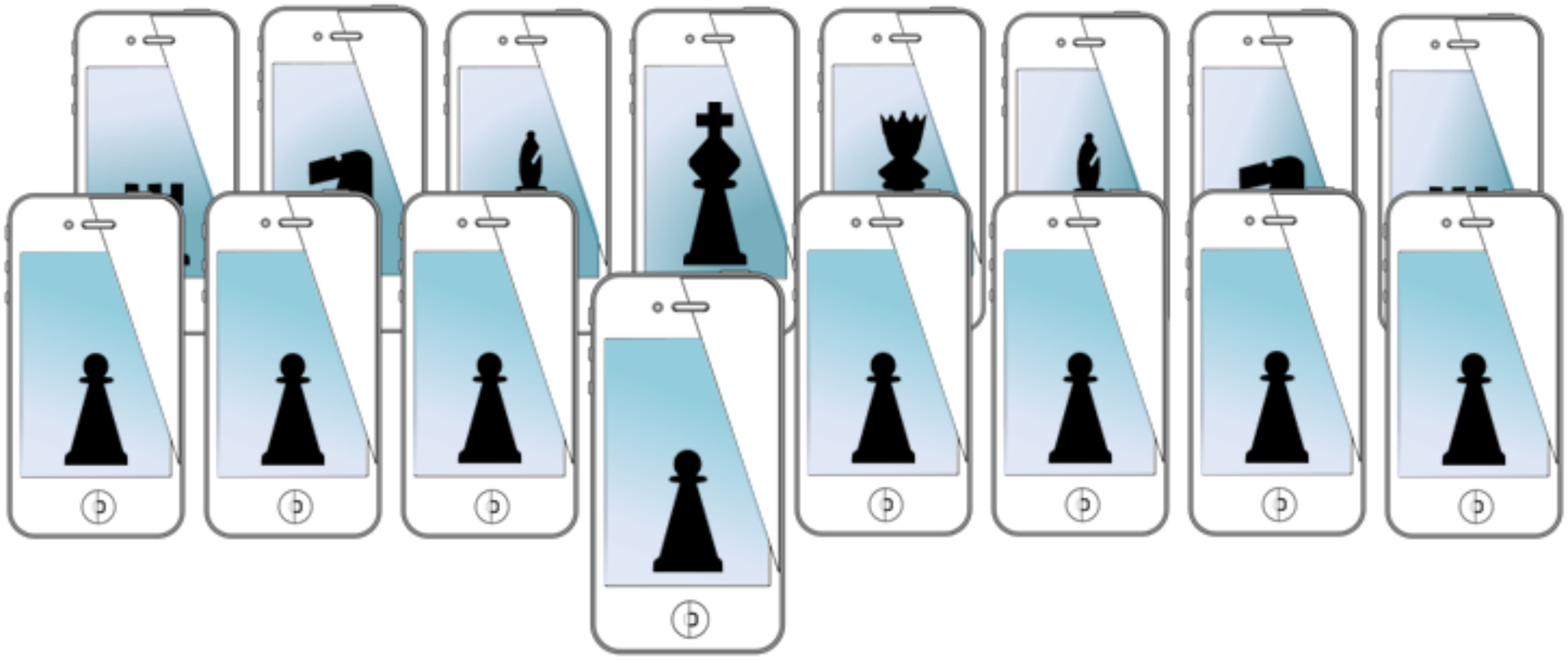
```
// An example of using already available queues from GCD
dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0), ^{
    // your code to execute
    for(int i=0;i<3;i++)
        NSLog(@"I am being executed from a global concurrent queue");

    // now I need to set something in the UI, but it must be set 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
```

MOBILE SENSING & LEARNING



CS5323 & 7323

Mobile Sensing & Learning

UI elements

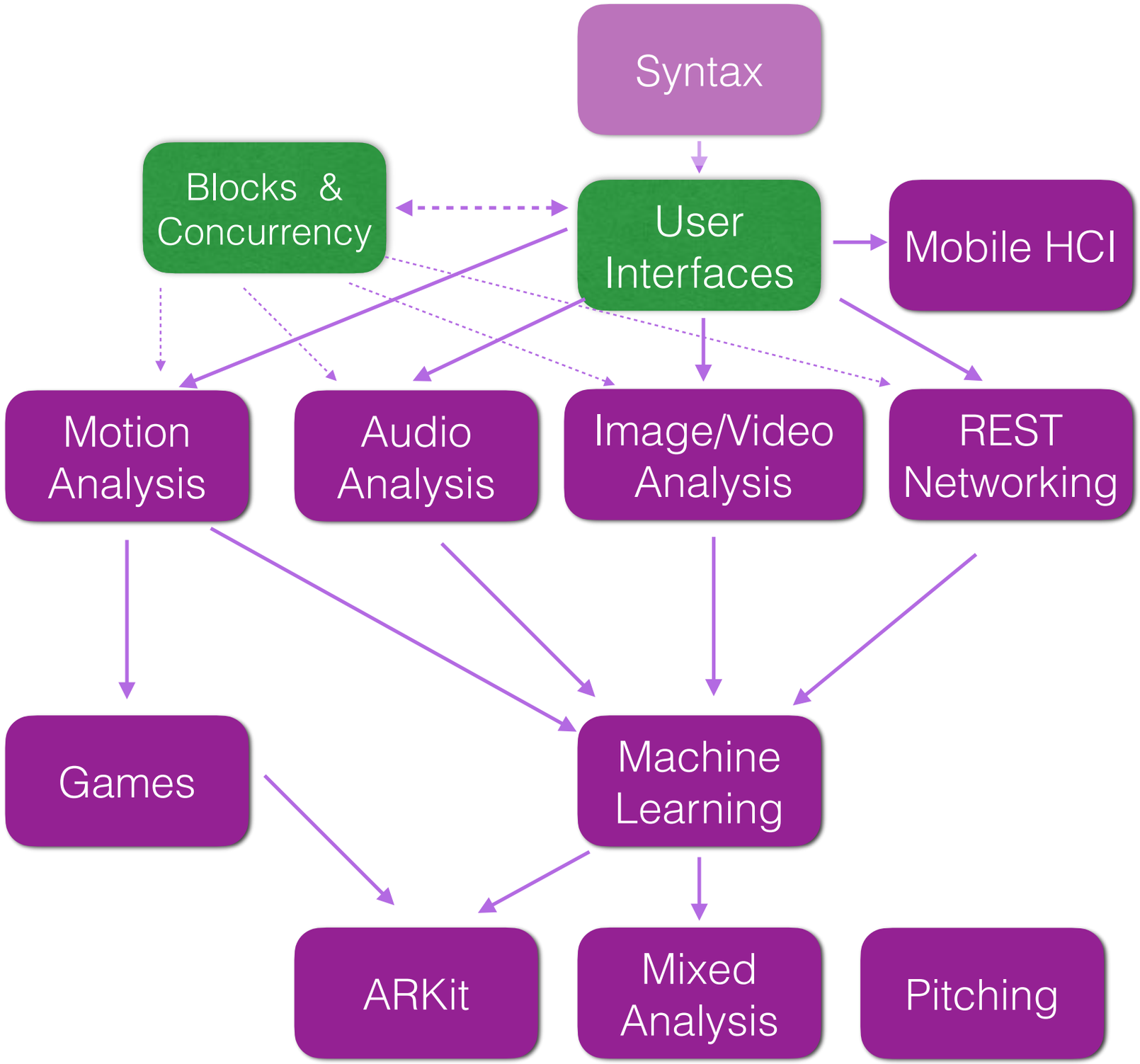
Eric C. Larson, Lyle School of Engineering,
Department of Computer Science, Southern Methodist University

course logistics and agenda

- **Lab Assignment One** rubric is posted
  - In addition to the app code, you need quick explanation of it
  - make a video of the app and submit it (YouTube, dropbox, direct upload to canvas, etc.)
  - use quicktime for video (if you don't know what to use)

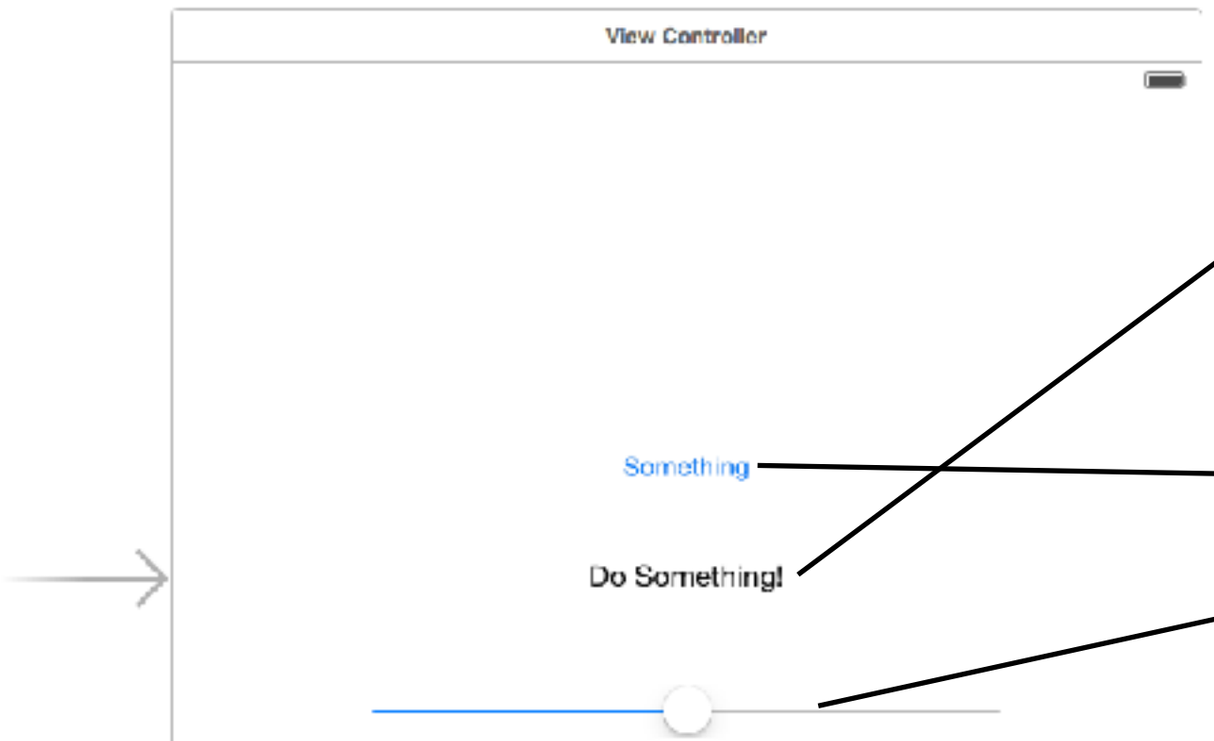
- agenda:
  - gesture recognizers
  - timers / segmented control
  - **remainder of time**: extended demo!

class progression



target and action review

- UI elements can have **outlets** and **actions**, UI actions are called from the **Main Queue**



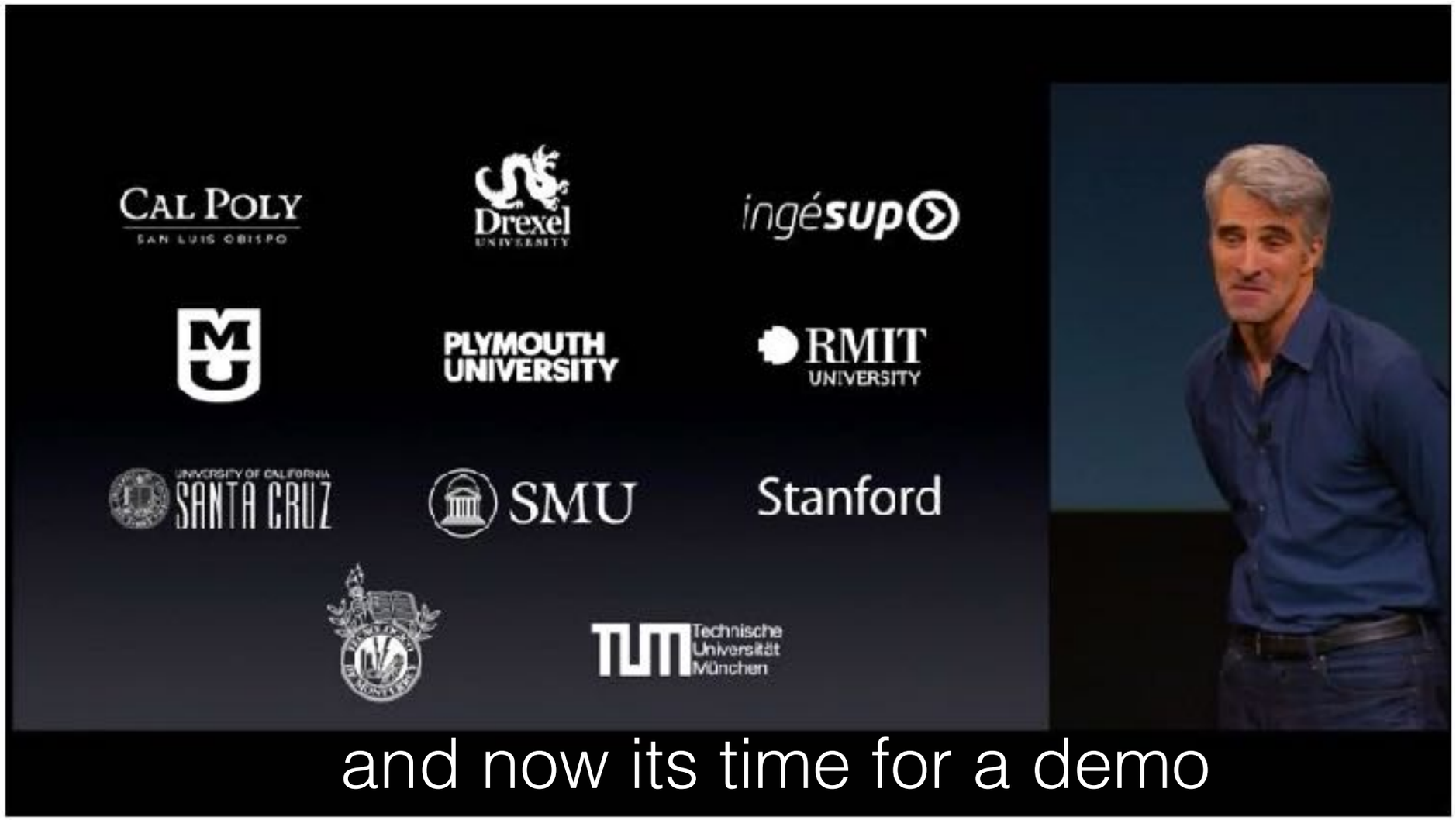
storyboard classes: the views

class: the controller

```
//
#import "ViewController.h"

@interface ViewController ()
@property (weak, nonatomic) IBOutlet UILabel *
    somethingLabel;

@end

@implementation ViewController
- (IBAction)buttonPressed:(UIButton *)sender {
    self.somethingLabel.text = @"Thanks!";
}
- (IBAction)sliderChanged:(UISlider *)sender {
    self.somethingLabel.text = [NSString
        stringWithFormat:@"Value of slider is %.
        2f",sender.value];
}

- (void)viewDidLoad {
    [super viewDidLoad];
    // Do any additional setup after loading the
        view, typically from a nib.
}
```

UI basics demo

Guess the Number…



adding a text field for guessing!

delegation and protocols

- delegation (alternative to virtual), avoids inheritance
- rather than inheriting functionality, **class declares that it adopts a protocol** (a collection of methods)
  - other classes can call these protocol methods using the class!
- delegation is user EVERYWHERE in iOS user interfaces

Class A

I Implement Protocol Larson

I am a delegate for Larson

```
self.classB.delegate = self
```

Protocol Larson:

```
allowsLateWork -> (Boolean)
hasOfficeHours: -> (Boolean)
```

Class B

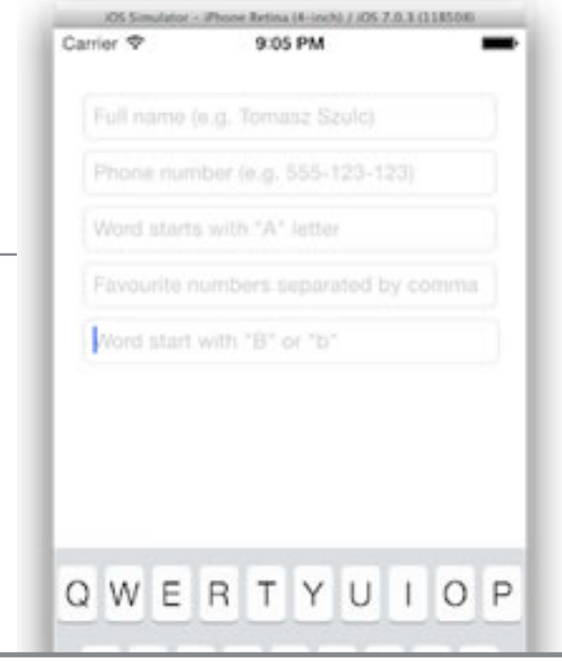
I need a delegate that implements protocol Larson

```
if self.delegate{
   self.delegate.allowsLateWork()
}
```

would be error if delegate did not implement protocol!!!!

text fields

- text fields require the use of the keyboard, which is done through delegation
- UITextField needs a **delegate**
  - so that it can dismiss keyboard
  - define what happens when user presets “Done”



outlet, setup from storyboard

```
@interface ViewController () <UITextFieldDelegate>
                    onatomic) IBOutlet UITextField *nameTextField;


                   ewController

- (    )viewDidLoad {
        per viewDidLoad];
      self.nameTextField.delegate = self;
}

-(BOOL)textFieldShouldReturn:(UITextField *)textField{
    [textField resignFirstResponder];
    return YES;
}
```

protocol method
text field can call this when needed

tell compiler we are delegate

I am your delegate

give up keyboard control

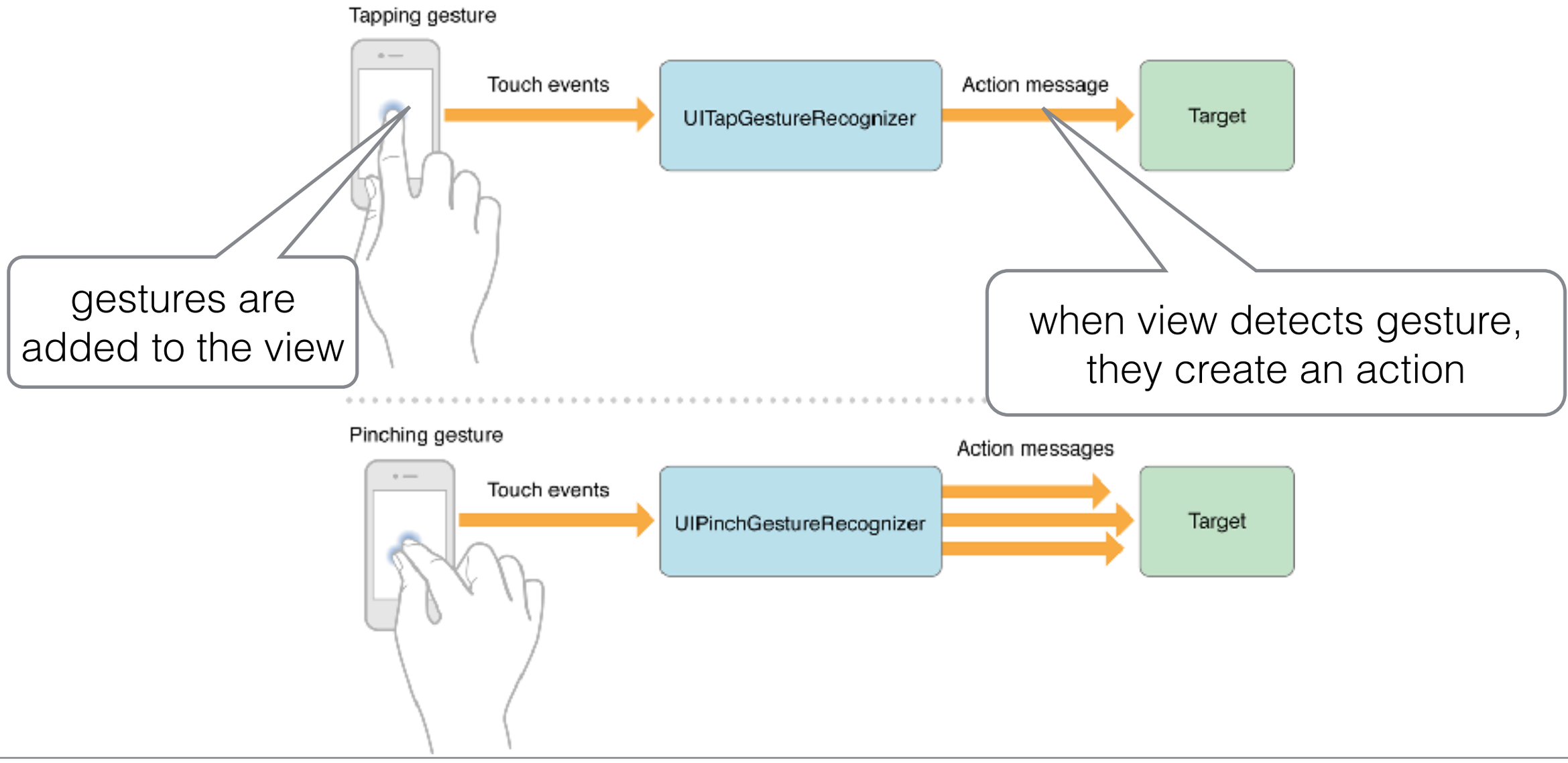
UI text field demo

gesture recognition

- the fun part about doing things on the iPhone!
- **the point**: recognize different gestures and then make something happen
- lots of ways to do this
  - **programmatically**: quick and versatile
  - **target-action**: easy
  - **delegation**: more feature rich
- complete documentation on developer.apple.com

gesture recognition

- need a UIGestureRecognizer
  - UITapGestureRecognizer, UIPinchGestureRecognizer, …



UI gesture demo

the coming weeks

- next lectures:
  - mobile HCI
  - moving on to audio

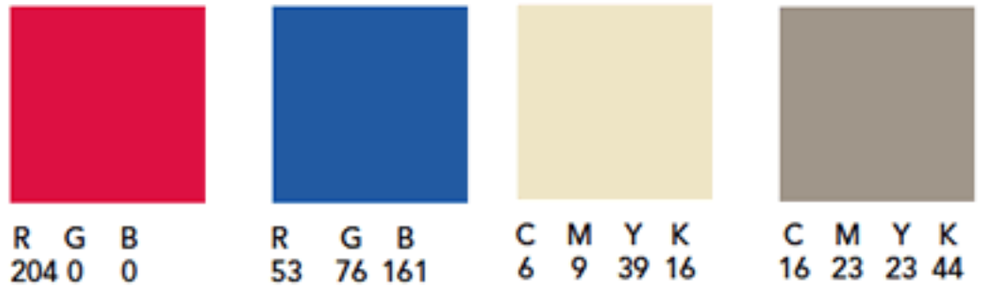
timers, segmented control

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

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

    YOUR_CODE
}
```

get title from control

get value of control

standard SMU colors




```
var elapsedTime:Int = 0

let timer:Timer = Timer(timeInterval: 1, repeats: true) { tmp in
            elapsedTime += 1
            someTextField.text = "Elapsed: \(elapsedTime)"
}

RunLoop.main.add(timer, forMode: .common)
```

pickers

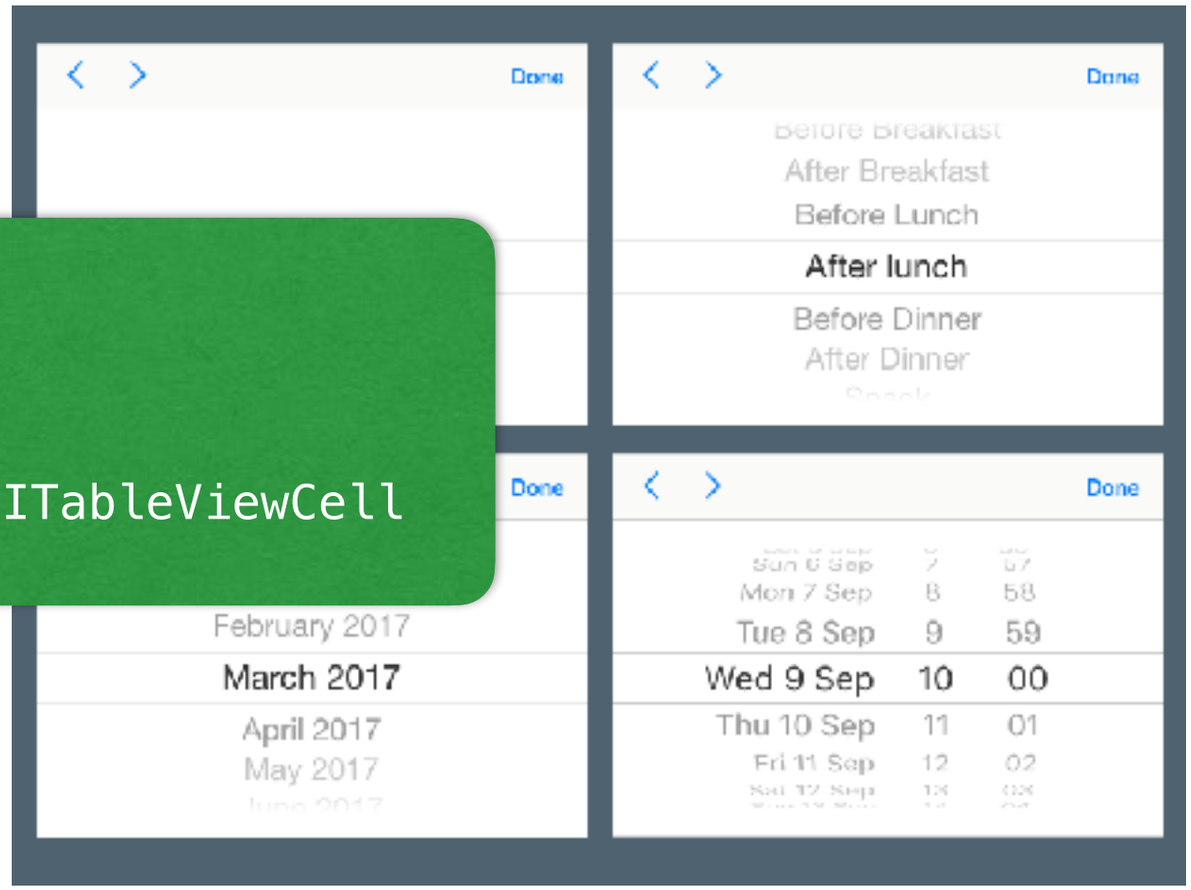
- **look at documentation**: find out how to use a picker view
- you will have all the tools to do it from working with table view controllers in flipped lecture

Class A
I Implement datasource methods
I am a datasource for SomeView
(like UITableView)

urce!

Datasource Methods:

```
numberOfSections -> (Int)
tableView(_ tableView: UITableView,
cellForRowAt indexPath: IndexPath) -> UITableViewCell
```



assignment one, discuss

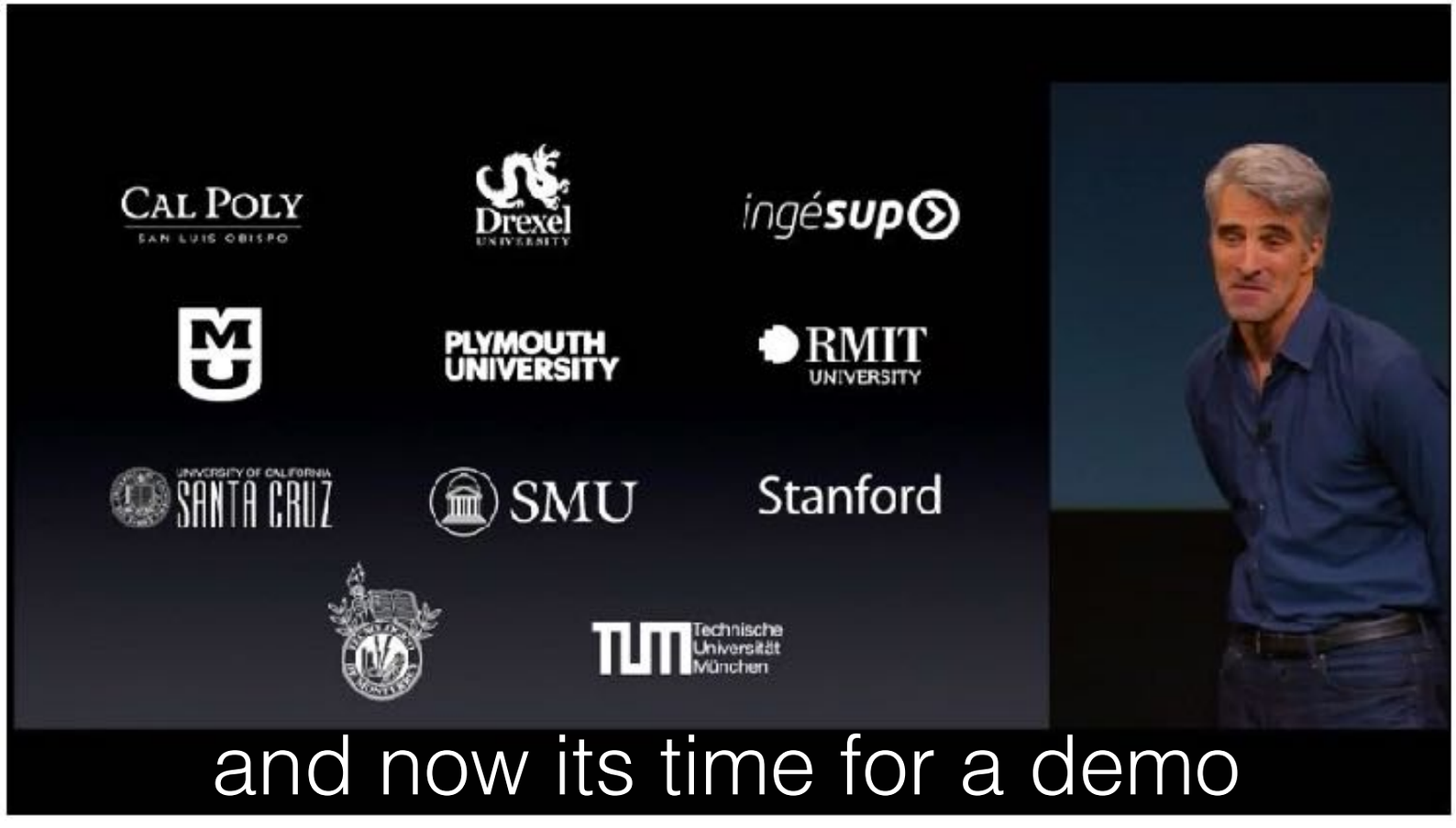
- Posted on Canvas!

before demo… don't forget

- next lectures:
  - mobile HCI
  - audio access

adding functionality

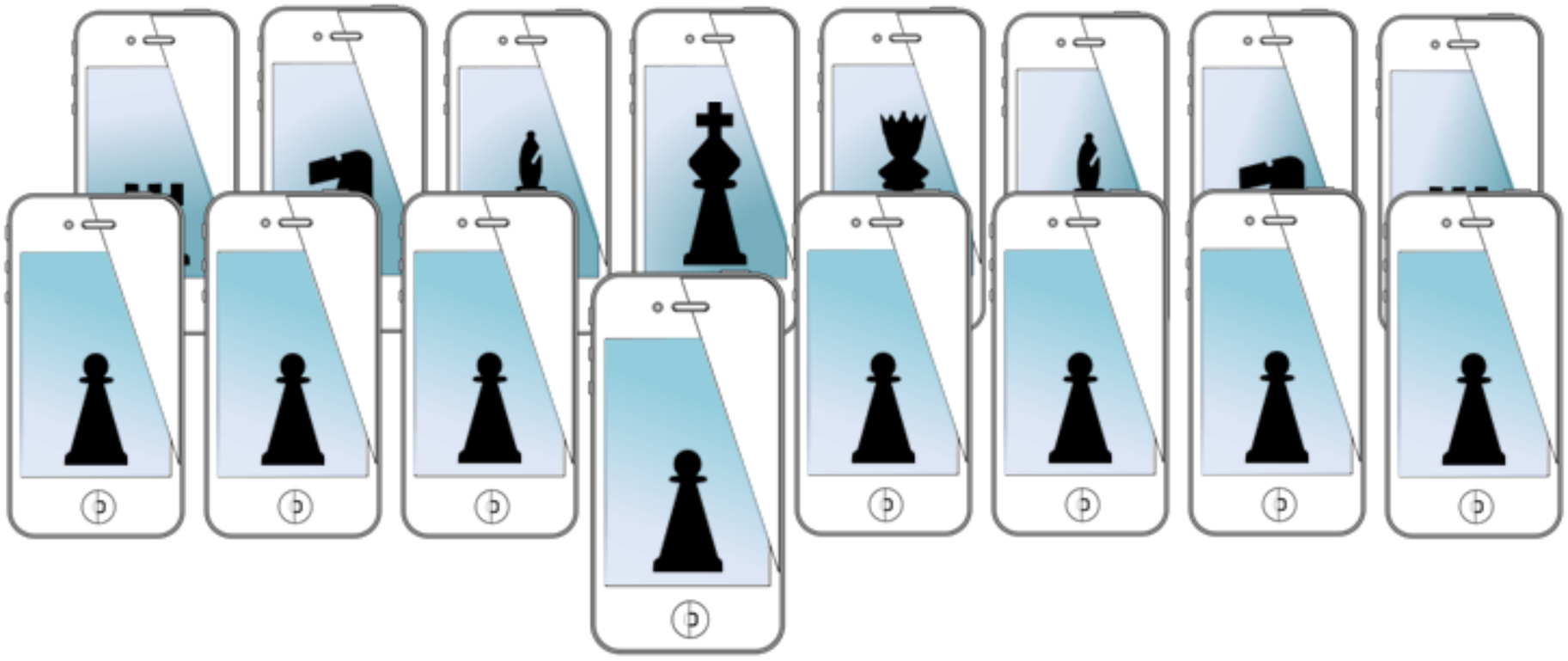
- additional functionality, feedback, etc.
- using the `switch` statement in swift (very powerful)
- `if let`
- mixing objective-c
- and more!



https://developer.apple.com/swift/

https://docs.swift.org/swift-book/GuidedTour/GuidedTour.html

MOBILE SENSING & LEARNING



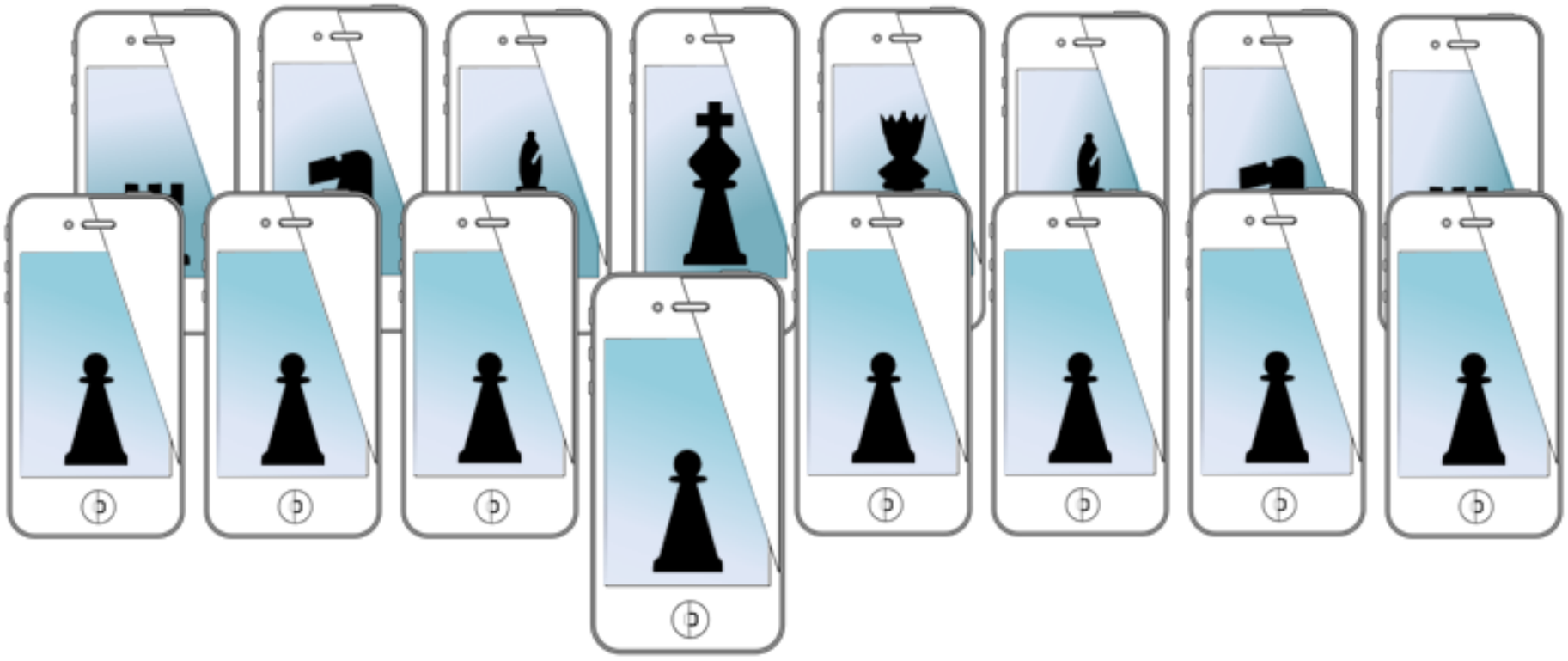
CS5323 & 7323

Mobile Sensing & Learning

UI elements

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

MOBILE SENSING & LEARNING



CS5323 & 7323

Mobile Sensing & Learning

Video Module One, model view controllers

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

agenda (video)

- MVC (potential review)
  - outlets, actions, delegates, protocols, data source
- ViewControllers in iOS
  - TableViewControllers, NavigationViewController, CollectionViewController, UIViewController
- storyboard (with UIViewController)
  - outlets, auto layout, programatic creation
  - timers, UIScrollView, image assets,

MVC's

controller has direct connection to view class

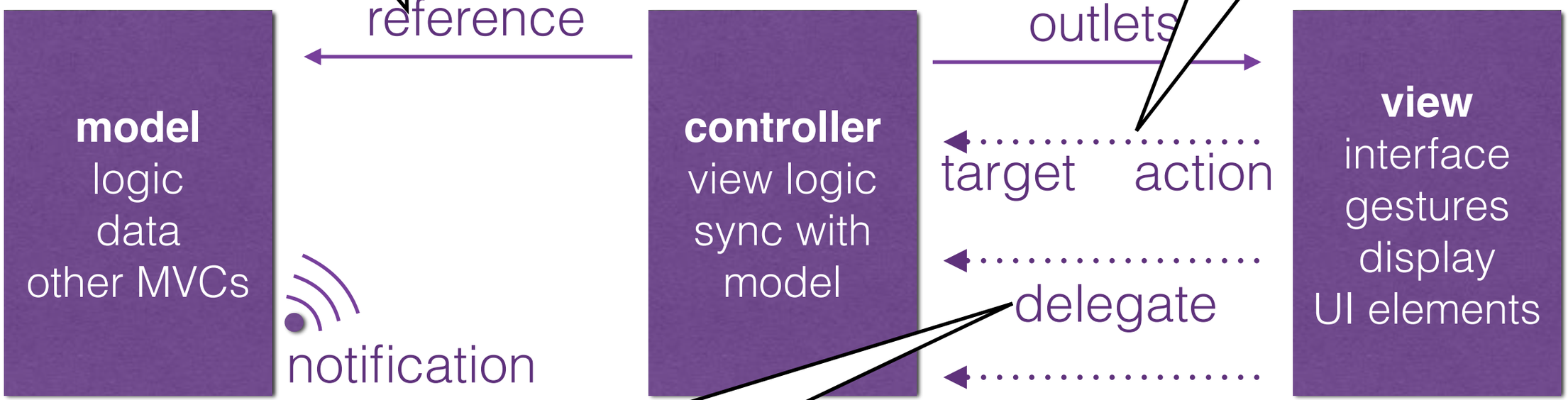
```
@property (weak, nonatomic) IBOutlet UITextField *firstName;
@property (weak, nonatomic) IBOutlet UITextField *lastName;
@property (weak, nonatomic) IBOutlet UITextField *phoneNumber;
```

controller has direct connection to model class

```
ModelClass *myModel = [get global handle to model]
PhoneNumberStruct * phNumber = [myModel getNumber];
self.phoneNumberLabel.text = phNumber.number;
```

view sends a targeted message

```
- (IBAction)buttonPressed:(id)sender;
- (IBAction)showPhBookPressed:(id)sender;
```




```
MainViewController ()<UITextFieldDelegate>
#pragma  mark - UITextfield Delegate
- (BOOL)textFieldShouldReturn:(UITextField *)textField { …  }
```

controller implements method for view class

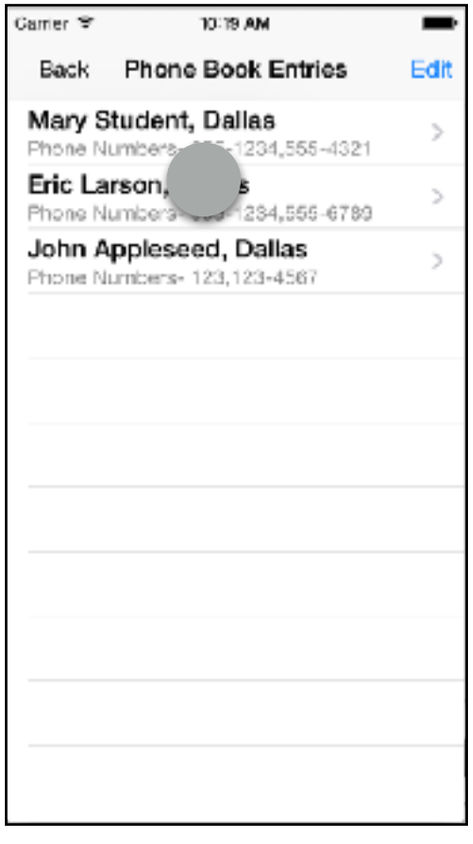
Legend

direct connection
indirect action
general broadcast

```
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
```

controller life cycle review

- problem: we need to handoff control of the screen to a new view
- the app itself is handling most of this transition
  - app will “unfreeze” the new view and its class properties
  - **you** need to send information from **source** ViewController to **destination** ViewController



controller life cycle review

Source Controller

`prepareForSegue`
prepare to leave the screen

set properties of destination, if needed

Destination Controller

view is unfrozen, property memory allocated

view outlets are ready for interaction

`viewDidLoad`

`viewWillAppear`

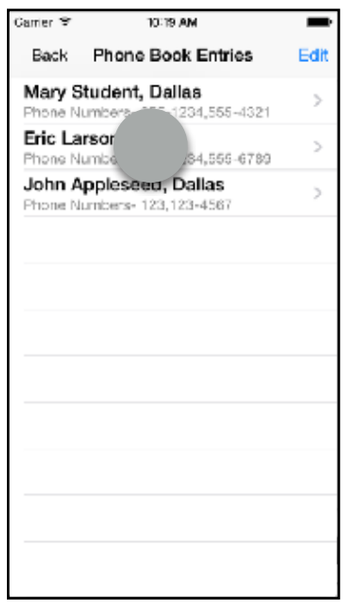
`viewDidAppear`

`viewWillDisappear`

`viewDidDisappear`

memory deallocated when app is ready

source



destination

user

the storyboard

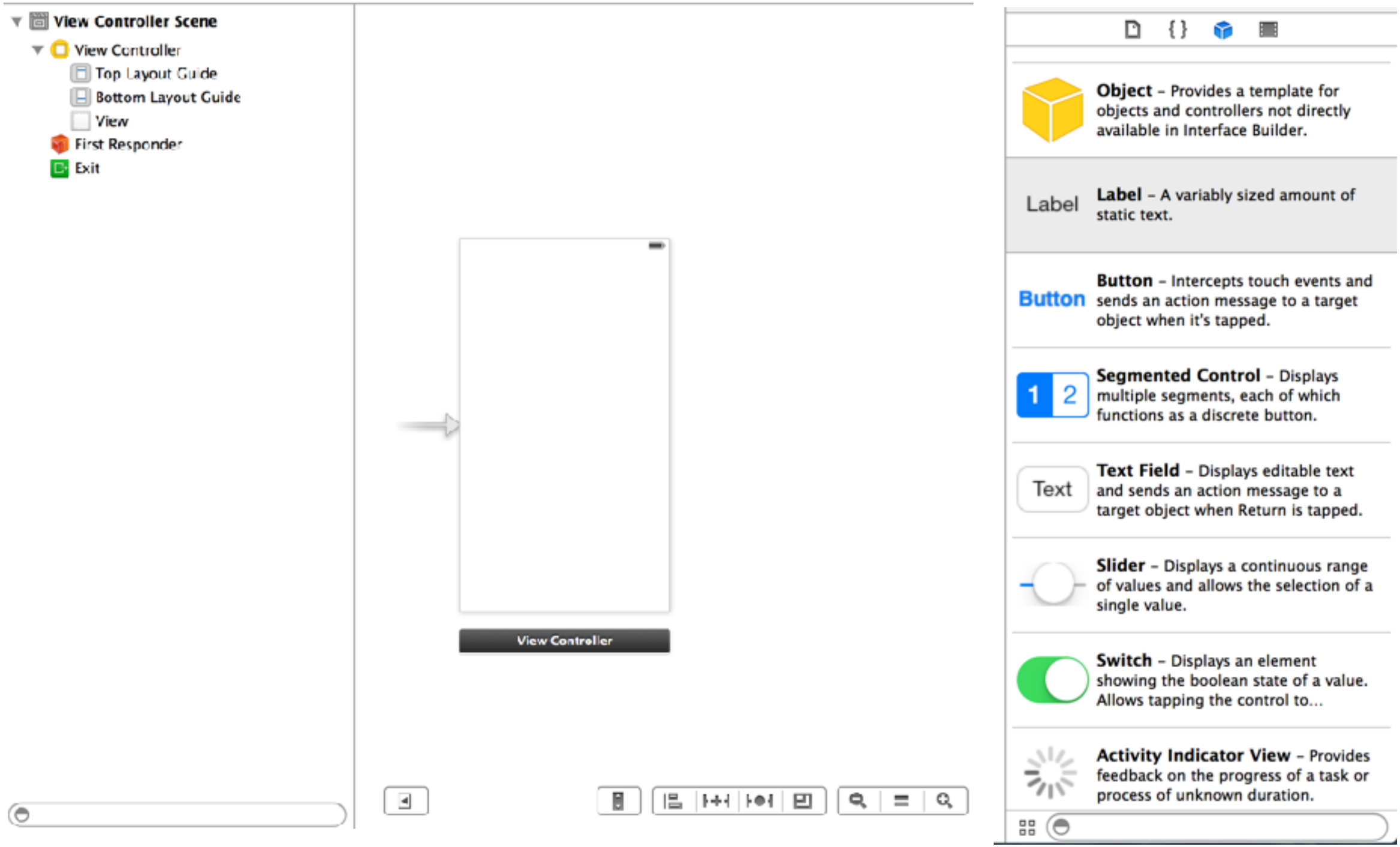
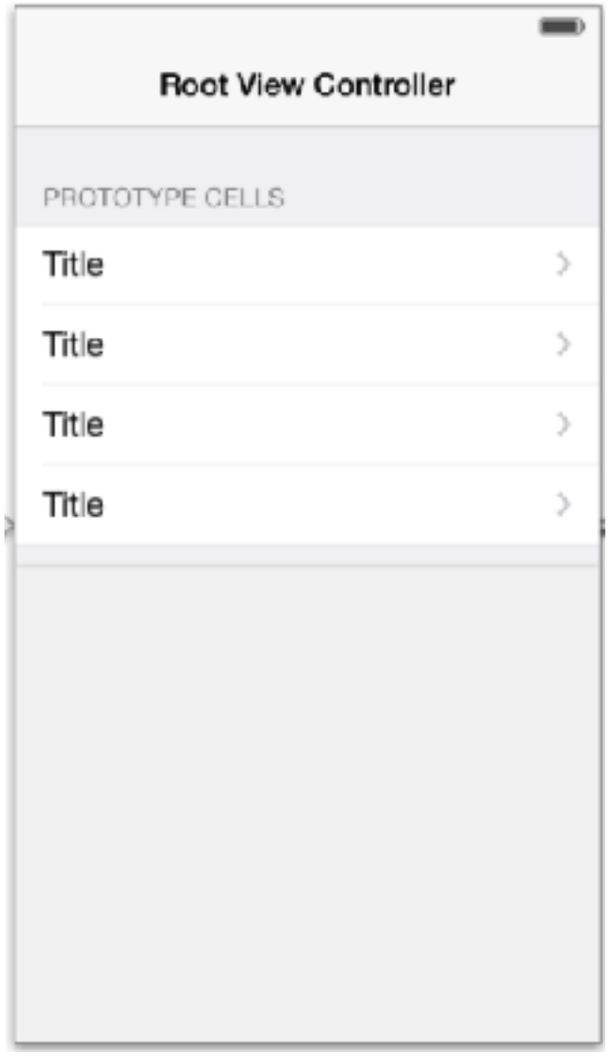
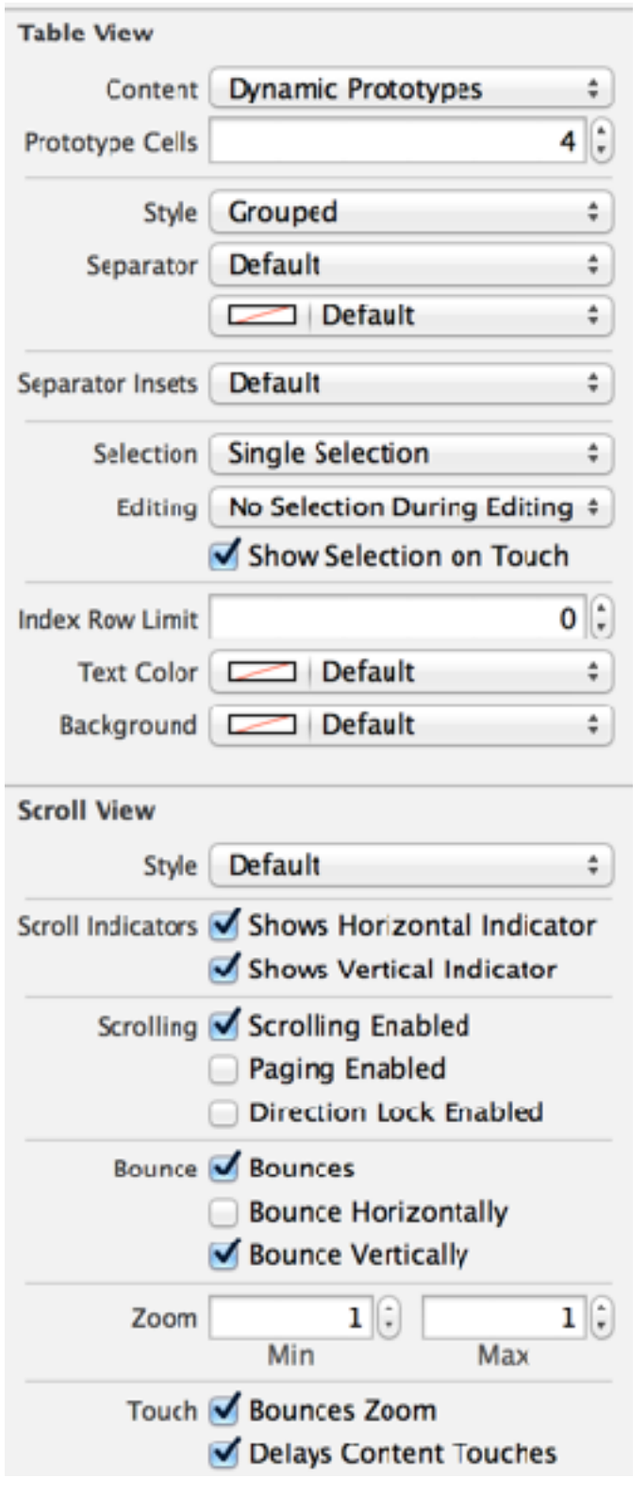
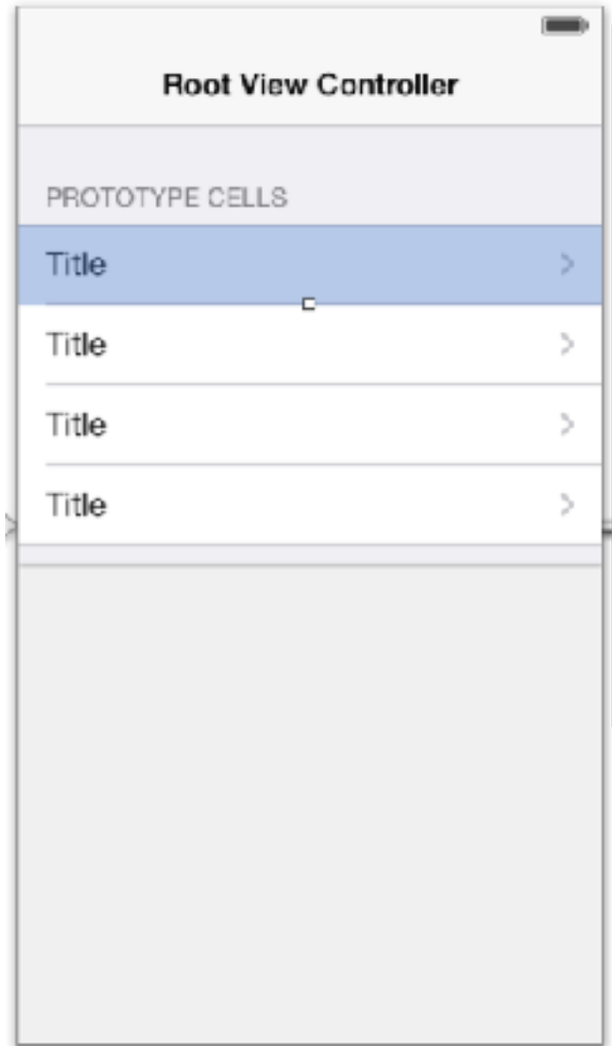


table view controller







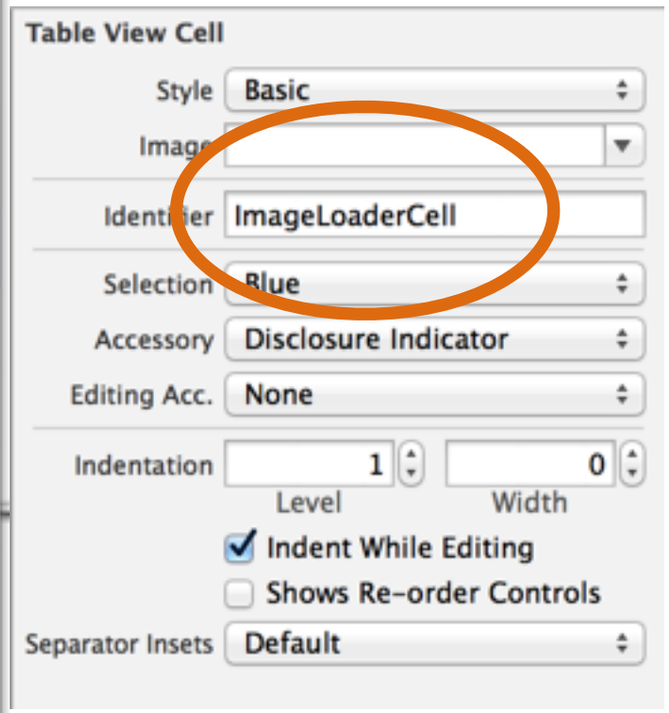


table view controller

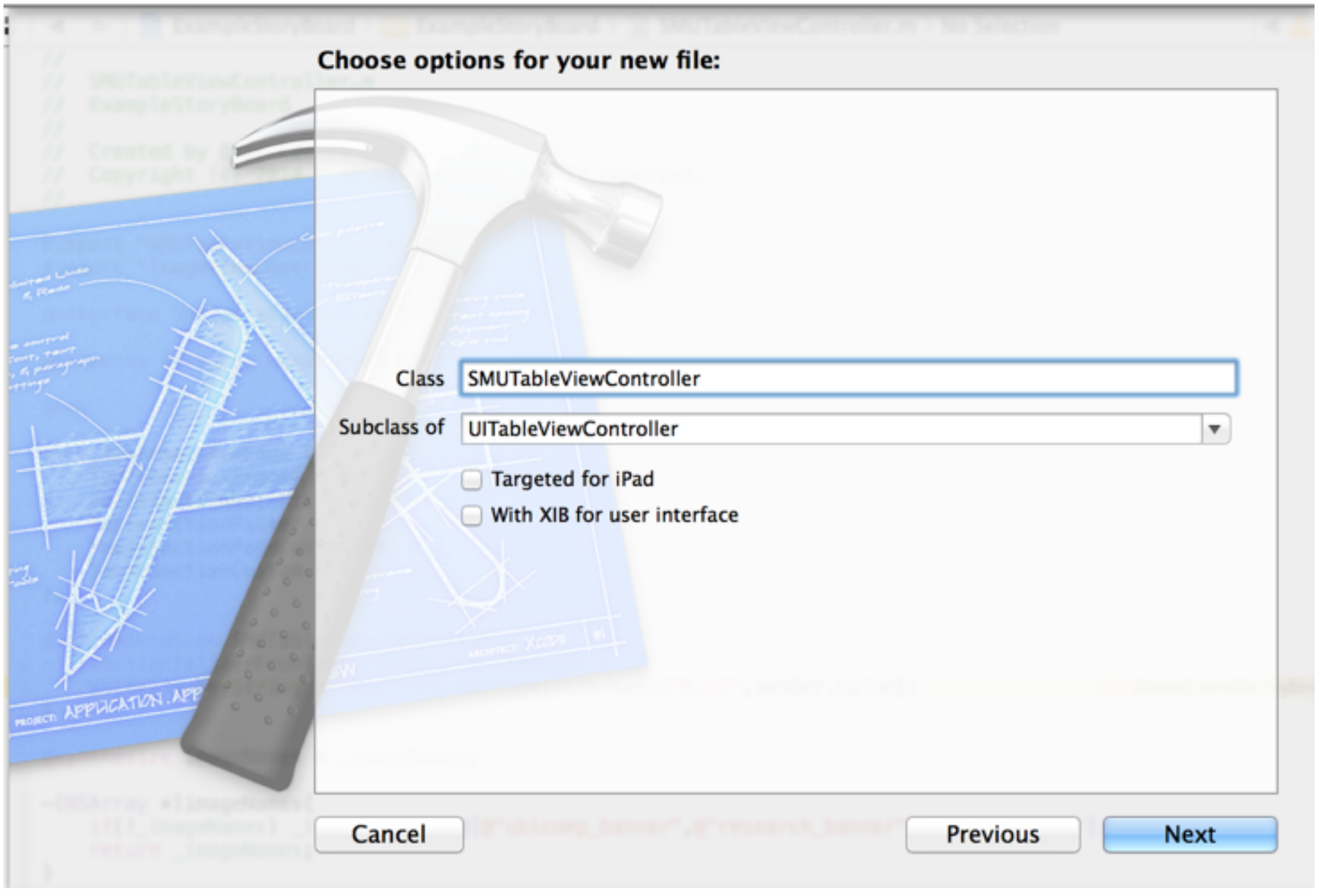


table view controller

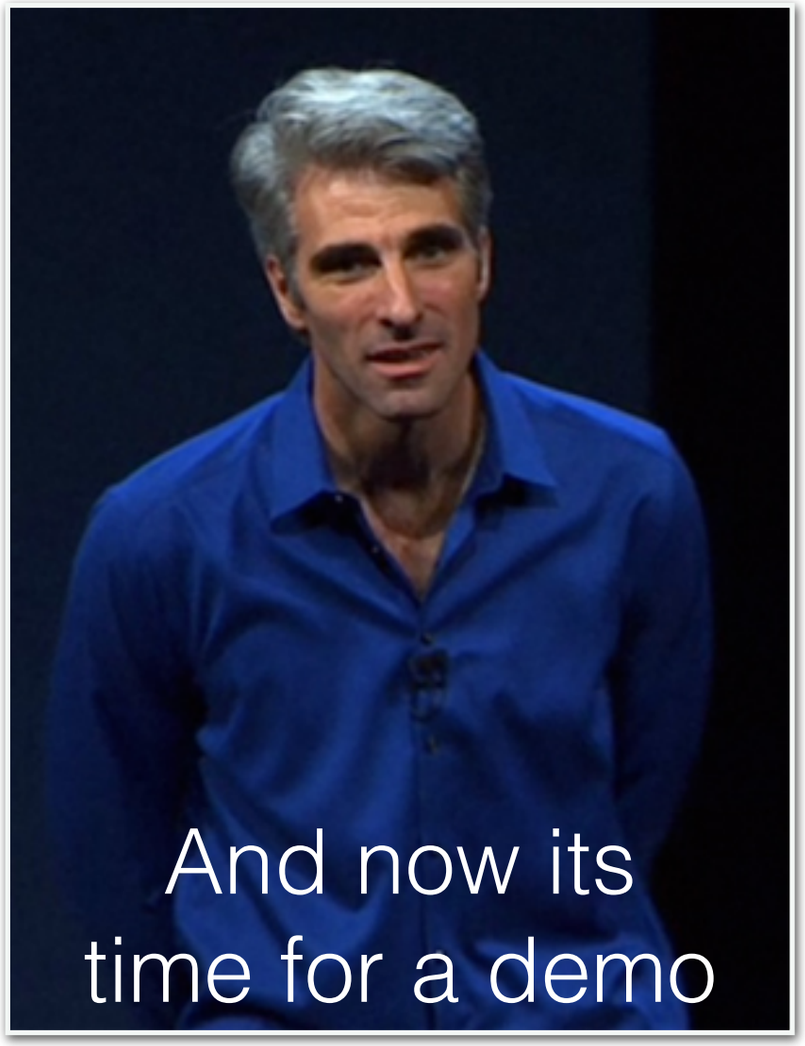
- must implement “data source” methods

```
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
    return numSections;
}


- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
{
    return rowsInSectionNumber[section];
}

- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath
*)indexPath
{
    static NSString *CellIdentifier = nil;
    UITableViewCell *cell = nil;

    CellIdentifier = @"ImageLoaderCell";
    cell = [tableView dequeueReusableCellWithIdentifier:CellIdentifier forIndexPath:indexPath];

    // Configure the cell
    cell.textLabel.text = @“An Image”;
    return cell;
}
```

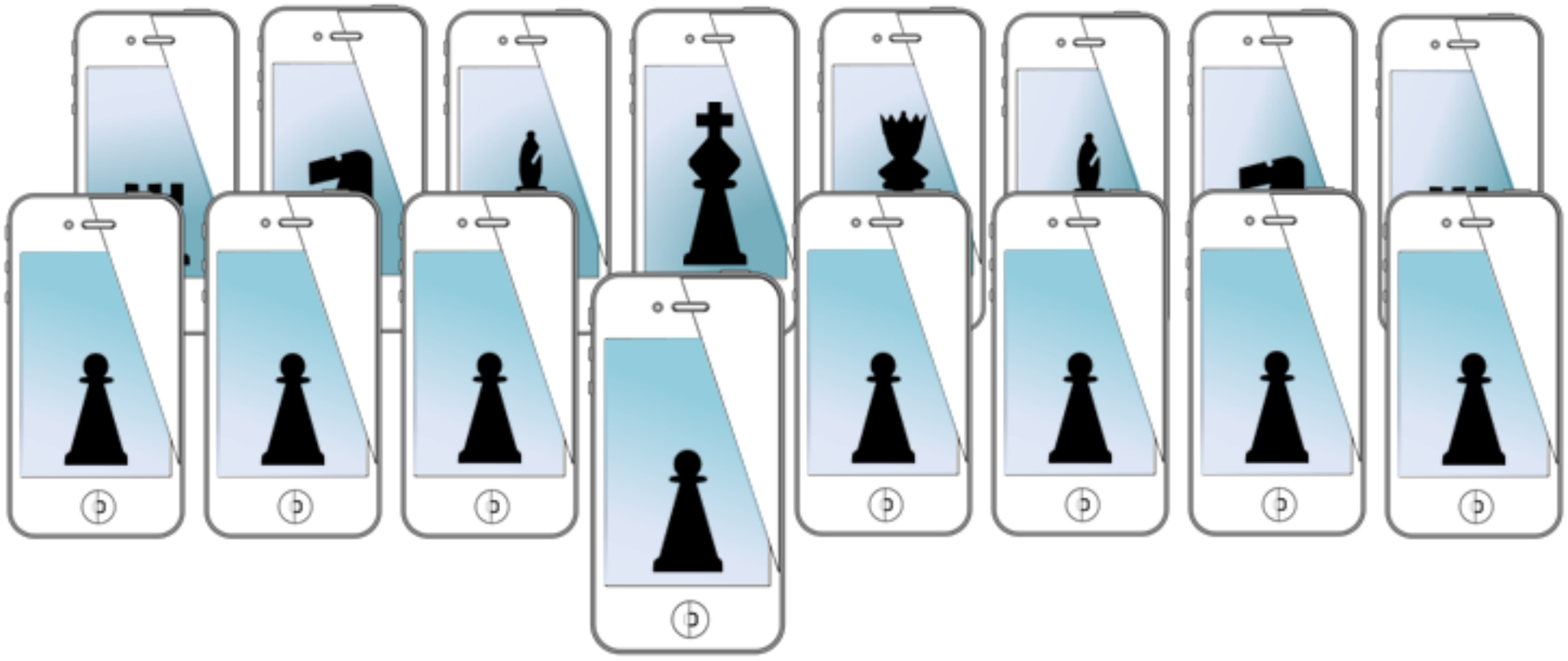
cell prototype from storyboard

set cell attributes

table view controller demo



MOBILE SENSING & LEARNING



CS5323 & 7323

Mobile Sensing & Learning

Video Module One, model view controllers

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

scroll view delegate

```
@interface SomeViewController () <UIScrollViewDelegate>

[self.someScrollView addSubview:self.imageView];
self.someScrollView.contentSize = self.image.size;
self.someScrollView.minimumZoomScale = 0.1;
self.someScrollView.delegate = self;
```

add view to the scroll view

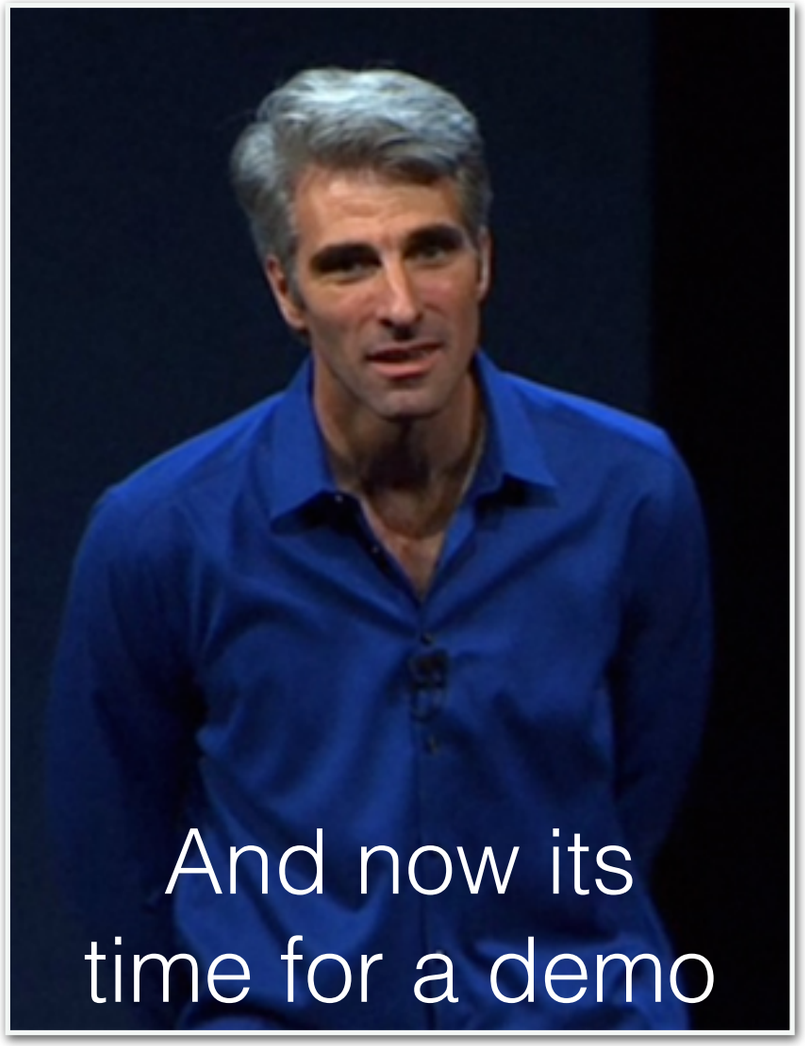
I am a delegate for the Scroll View: I implement methods in the Scroll View Protocol!

set VC as delegate

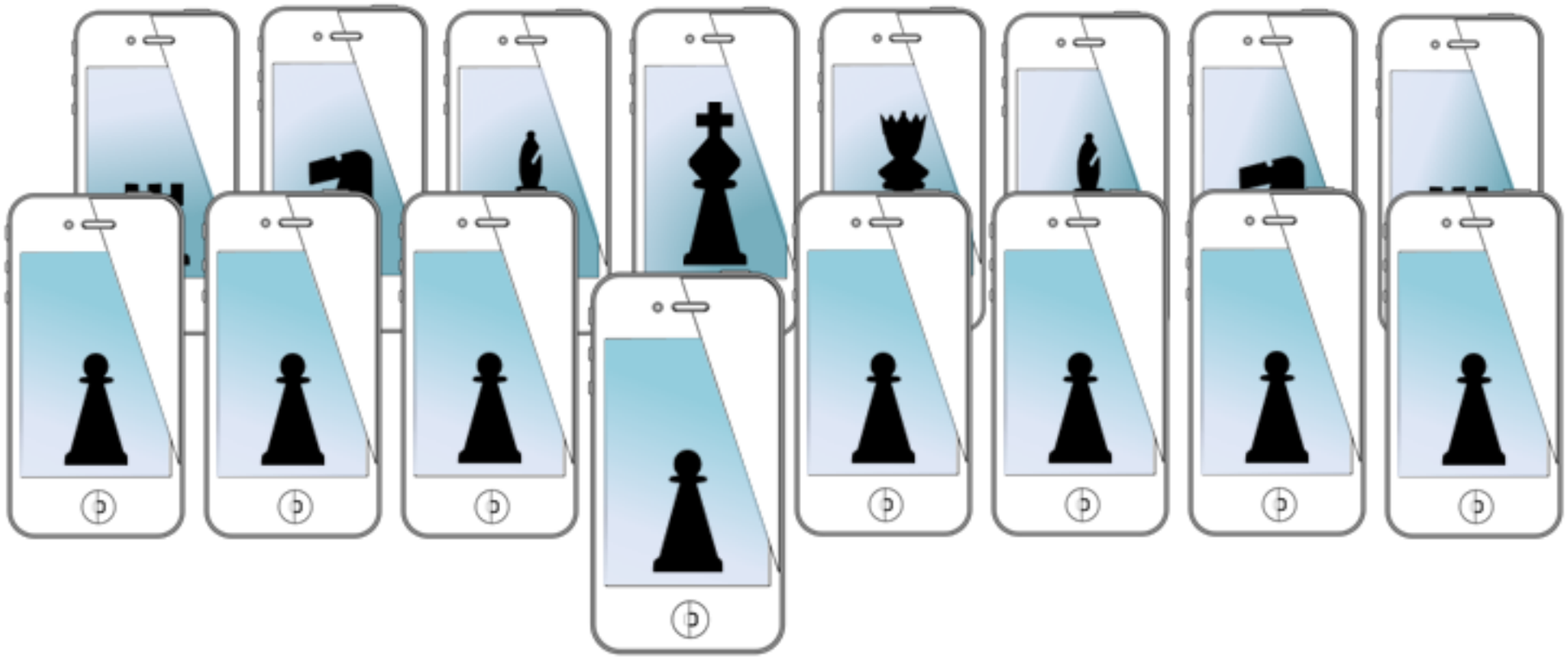
```
#pragma Delegate Methods
-(UIView*) viewForZoomingInScrollView:(UIScrollView *)scrollView
{
    return self.imageView;
}
```

one of many methods in the protocol

demo



MOBILE SENSING & LEARNING



CS5323 & 7323

Mobile Sensing & Learning

Video Module One, model view controllers

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

Supplemental Slides

- we do not explicitly cover these topics in class anymore
- some of the info in these slides may be deprecated!
- otherwise, have fun browsing the material

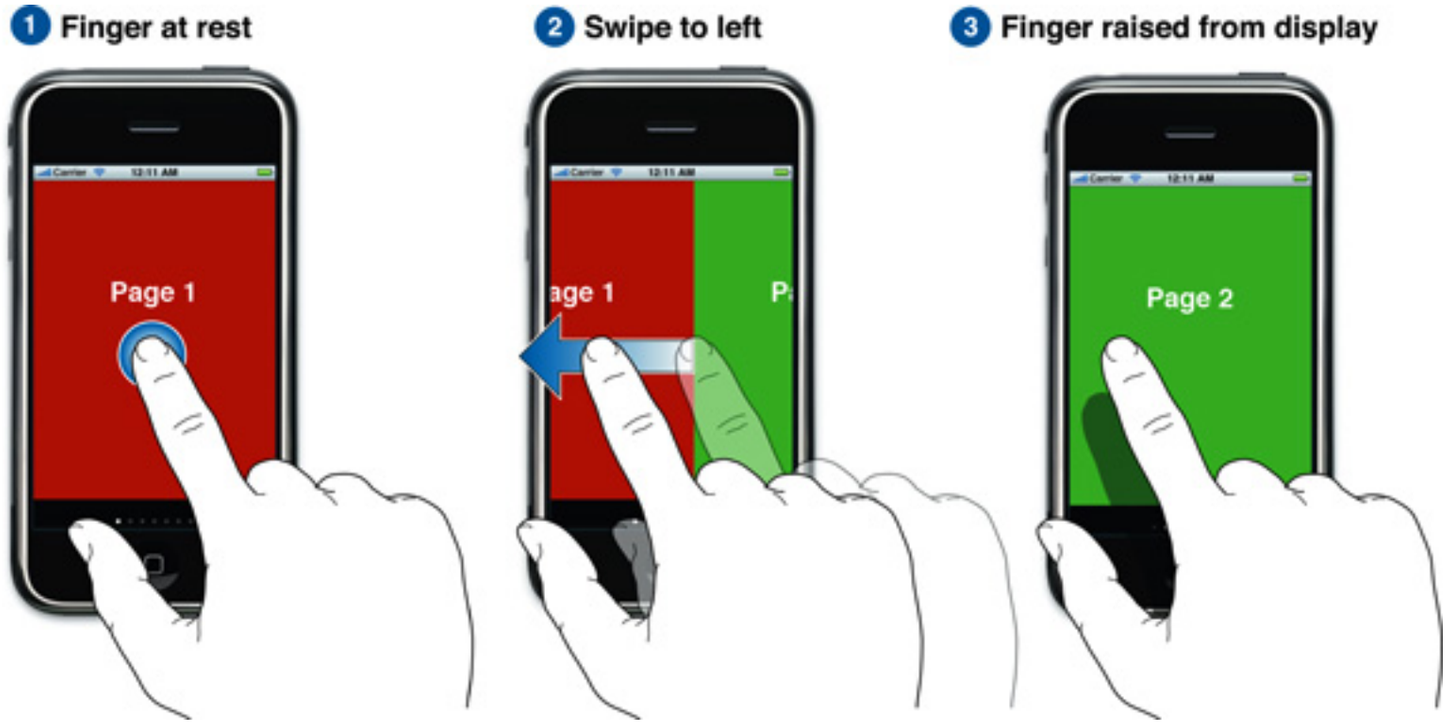
disclaimer!

page view controller

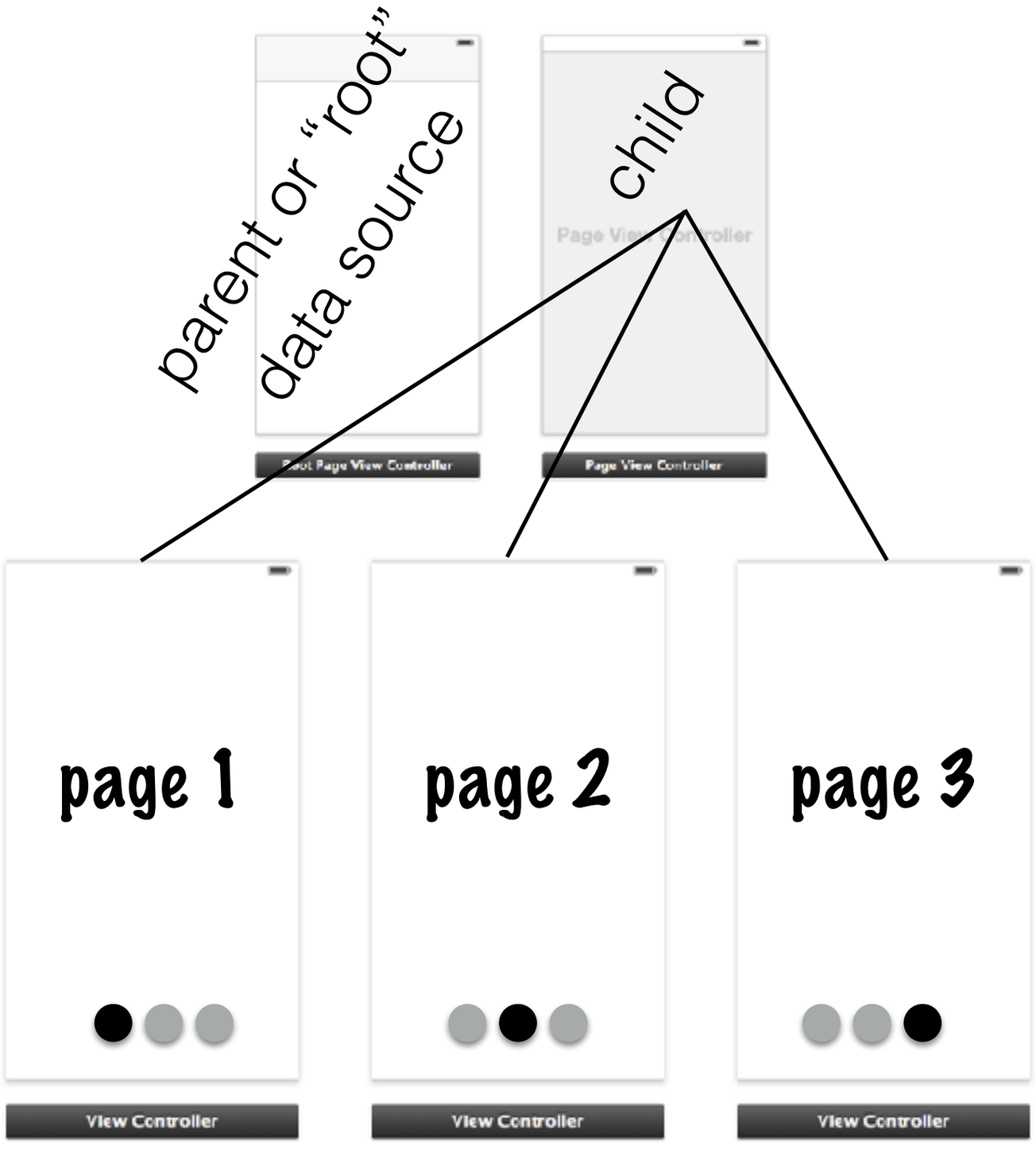
- place UIPageViewController in storyboard
- place a “root controller” for the page
  - adopt <UIPageViewControllerDataSource>
  - instantiate pageViewController
  - instantiate views to be paged

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”



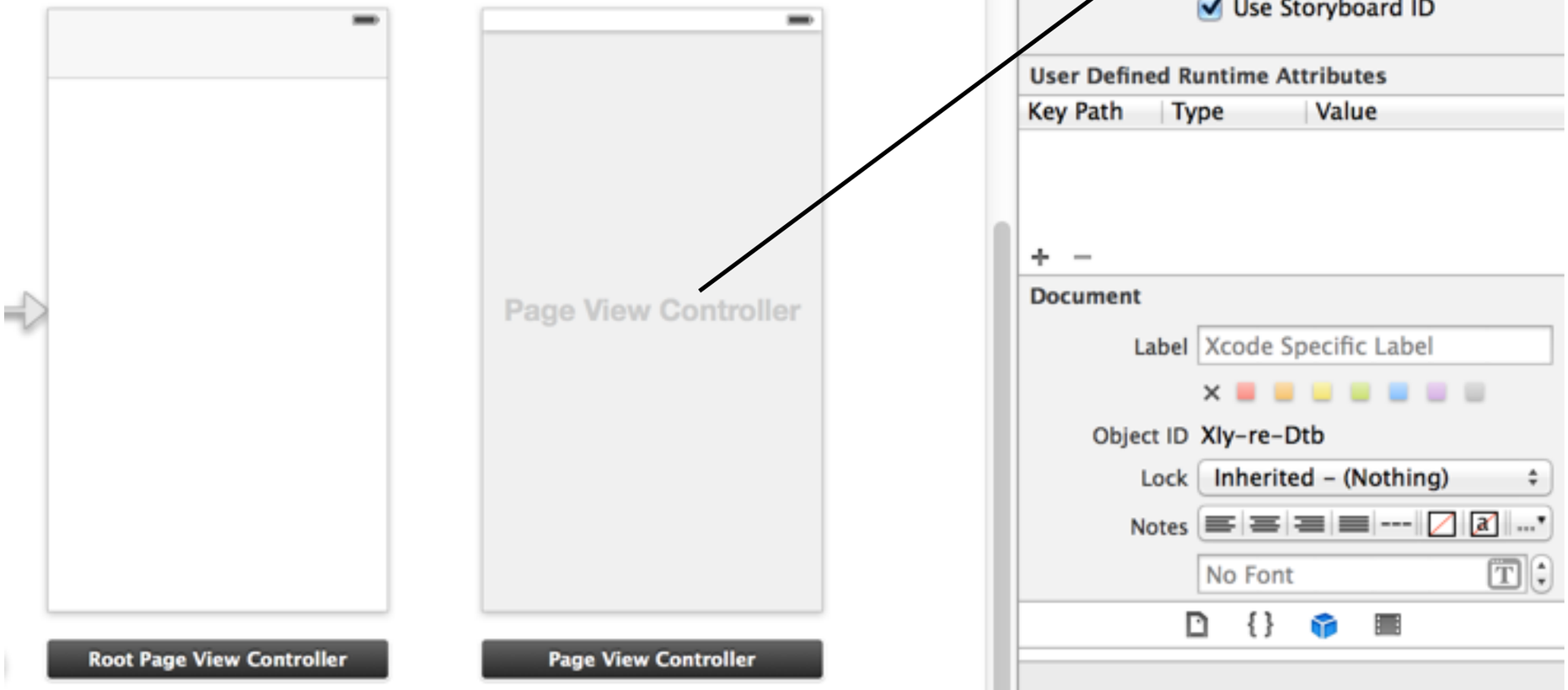
page view controller



different instantiations of view controller

page view controller

no need to subclass the page controller!



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

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;
}
```

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

programatic UI creation

```
@property (strong, nonatomic) IBOutlet UIButton *button;
```

…

```
// a button, created programmatically
self.button = [UIButton buttonWithType:UIButtonTypeSystem];

// set a target method for a control event, touch down
[self.button addTarget:self
                action:@selector(updateLabelFromProgramButton:)
 forControlEvents:UIControlEventTouchDown];

// set the button attribute
[self.button setTitle:@"PButton" forState:UIControlStateNormal];
```

visual format language

```
// say that these exist and are initialized and added to the view as subviews
UIButton *button;
UILabel *label;
[button setTranslatesAutoresizingMaskIntoConstraints:NO];

// setup button and label constraints, also make same size
NSDictionary *varBindings = NSDictionaryOfVariableBindings(button, label);
NSArray *constraints =
    [NSLayoutConstraint constraintsWithVisualFormat:@"|-[button]-24-[label(==button)]-|"
                                            options:0
                                            metrics:nil
                                              views:varBindings];
    [self.view addConstraints:constraints];
```

same size as button

8 points from left side

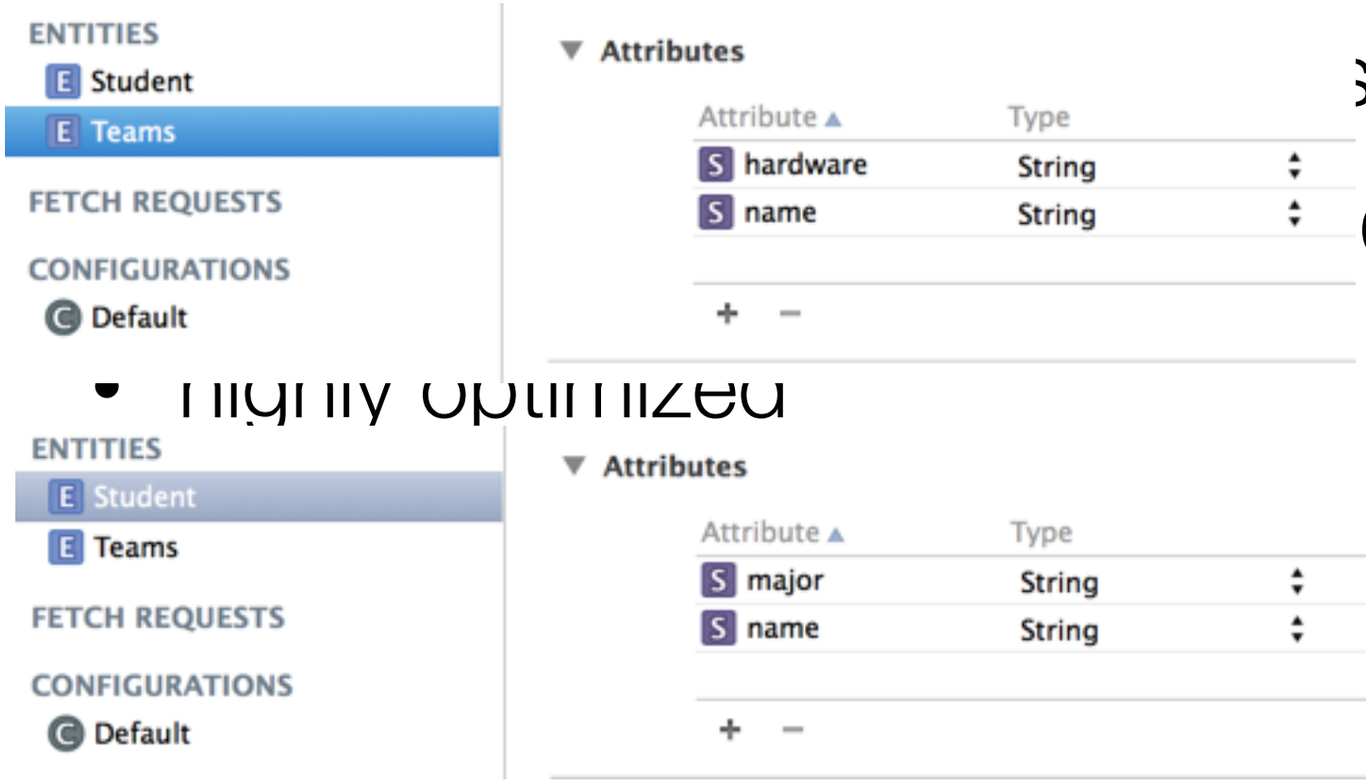
24 points between

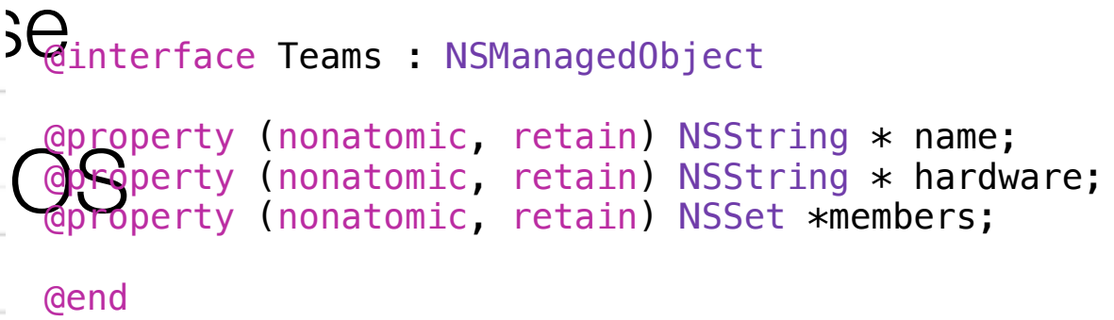
```
// metrics for use in visual constraints
NSDictionary *metrics = @{@“spacing":@10.0};

[NSLayoutConstraint constraintsWithVisualFormat:@"|-[button]-spacing-[label(==button)]-|"
                                            options:0
                                            metrics:metrics
                                              views:varBindings];


        @“V:|-[button]”
        options:NSLayoutFormatAlignAllTop | NSLayoutFormatAlignAllCenterX
```

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

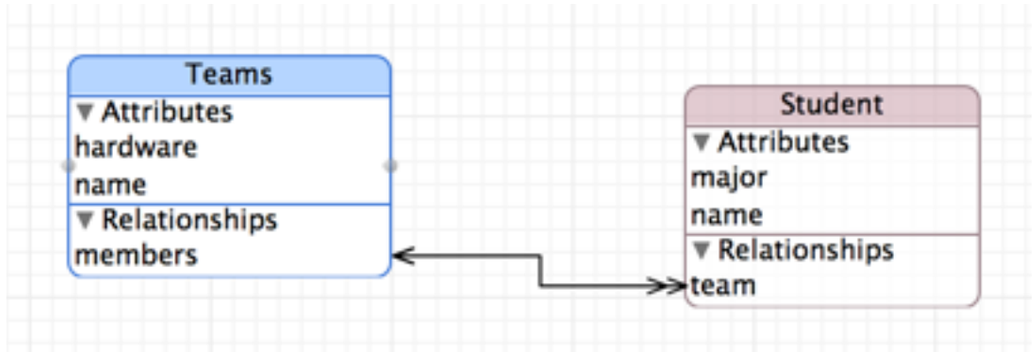




```
@interface Teams : NSManagedObject

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

@end
```

- highly optimized


```
@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

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

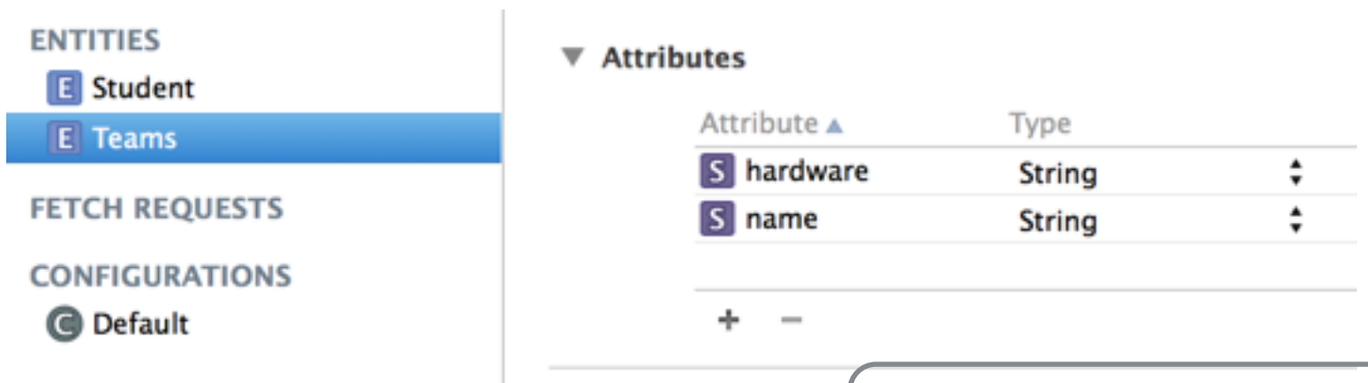
```
// Getter for managed context
- (NSManagedObjectContext *) managedObjectContext {

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

    return _managedObjectContext;
}
```

```
// 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



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;
}


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

    …

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

    …

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

request

entity to request from

fetch

array of results, even if size=0

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

notifications

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

```
MCDAppDelegate* appDelegate =                    delegate;

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

access notification center

do this

when this happens

from this context

lets add notifications to WhoWasInThat!

if time slides!

swift

- syntax is nothing like objective c
- a lot like python syntax (but not)
- weakly typed, no need for semicolons
- can be hard to read or interpret
- powerful use of *tuples,optionals,switch*
- you need to look online for more material than this lecture
  - https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html

swift

variables

```
let maximumNumberOfLoginAttempts = 10
var currentLoginAttempt = 0
```

not mutable

mutable

```
let pi = 3.14159
// pi is inferred to be of type Double
let three = 3
let pointOneFourOneFiveNine = 0.14159
let pi = Double(three) + pointOneFourOneFiveNine
// pi equals 3.14159, and is inferred to be of type Double
let meaningOfLife = 42
// meaningOfLife is inferred to be of type Int
```

and then there is this…

```
let orangesAreOrange = true
let turnipsAreDelicious = false
```

```
let π = 3.14159
let 你好 = "你好世界"
let 🐶🐮 = "dogcow"
```

```
var friendlyWelcome = “Hello World!"
var friendlyWelcome:  String = “Hello World!"

println(friendlyWelcome)

println("The current value of friendlyWelcome is \(friendlyWelcome)")
```

no need to set

see this: https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html

swift

tuples

```
let http404Error = (404, "Not Found")
// http404Error is of type (Int, String), and equals (404, "Not Found")
```

```
let (statusCode, statusMessage) = http404Error
println("The status code is \(statusCode)")
// prints "The status code is 404"
println("The status message is \(statusMessage)")
// prints "The status message is Not Found"
```

```
println("The status code is \(http404Error.0)")
// prints "The status code is 404"
println("The status message is \(http404Error.1)")
// prints "The status message is Not Found"
```

```
let http200Status = (statusCode: 200, description: "OK")
```

```
println("The status code is \(http200Status.statusCode)")
// prints "The status code is 200"
println("The status message is \(http200Status.description)")
// prints "The status message is OK"
```

see this: https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html

swift

optionals

```
let possibleNumber = "123"
let convertedNumber = possibleNumber.toInt()
// convertedNumber is inferred to be of type "Int?", or "optional Int"
```

```
var serverResponseCode:  Int? = 404
// serverResponseCode contains an actual Int value of 404
serverResponseCode = nil
// serverResponseCode now contains no value
```

can now set to nil :)

```
var surveyAnswer:  String?
// surveyAnswer is automatically set to nil
```

```
if convertedNumber != nil {
    println("convertedNumber has an integer value of \(convertedNumber!).")
}
// prints "convertedNumber has an integer value of 123."
```

```
if let actualNumber = possibleNumber.toInt() {
    println("\'\(possibleNumber)\' has an integer value of \(actualNumber)")
} else {
    println("\'\(possibleNumber)\' could not be converted to an integer")
}
// prints "'123' has an integer value of 123"
```

see this: https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html

swift

accessing optionals

optional

! **unwrap** output to be **string**. Else: **error**

```
let possibleString:  String? = "An optional string."
let forcedString:  String  = possibleString! // requires an exclamation mark
```

implicit unwrap

```
let assumedString:  String! = "An implicitly unwrapped optional string."
let implicitString:  String  = assumedString // no need for an exclamation mark
```

output always unwrapped to be **string**. Else: **error**

```
if assumedString != nil {
    println(assumedString)
}
// prints "An implicitly unwrapped optional string."


if let definiteString = assumedString {
    println(definiteString)
}
// prints "An implicitly unwrapped optional string."
```

Optional unwrapping is not my favorite part of swift

see this: https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html

swift

arrays

```
var shoppingList = ["Eggs", "Milk"]

println("The shopping list contains \(shoppingList.count) items.")
// prints "The shopping list contains 2 items."
```

```
if shoppingList.isEmpty {
    println("The shopping list is empty.")
} else {
    println("The shopping list is not empty.")
}
// prints "The shopping list is not empty."
```

```
shoppingList += ["Baking Powder"]
shoppingList += ["Chocolate Spread", "Cheese", "Butter"]

var firstItem = shoppingList[0]
// firstItem is equal to "Eggs"

shoppingList[0] = "Six eggs"
```

```
let butter = shoppingList.removeLast()
let sixEggs = shoppingList.removeAtIndex(0)
```

like a dequeue

like a pop

see this: https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html

dictionaries

```
var airports = ["YYZ": "Toronto Pearson", "DUB": "Dublin"]

 airports["LHR"] = "London"
 // the airports dictionary now contains 3 items
```

```
if let oldValue = airports.updateValue("Dublin Airport", forKey: "DUB") {
    println("The old value for DUB was \(oldValue).")
}
// prints "The old value for DUB was Dublin."
```

```
airports["APL"] = "Apple International"
// "Apple International" is not the real airport for APL, so delete it
airports["APL"] = nil
// APL has now been removed from the dictionary
```

```
let airportCodes = [String](airports.keys)
// airportCodes is ["YYZ", "LHR"]

let airportNames = [String](airports.values)
// airportNames is ["Toronto Pearson", "London Heathrow"]
```

see this: https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html

loops

```
for index in 1...3 {
    println("\(index) times 5 is \(index * 5)")
}
// 1 times 5 is 5
// 2 times 5 is 10
// 3 times 5 is 15
```

```
let names = ["Anna", "Alex", "Brian"]
for name in names {
    println("Hello, \(name)!")
}
// Hello, Anna!
// Hello, Alex!
// Hello, Brian!
```

```
for (index, value) in enumerate(names) {
    println("Item \(index + 1): \(value)")
}
// Item 1: Anna
// Item 2: Alex
// Item 3: Brian
```

```
let numberOfLegs = ["spider": 8, "ant": 6, "cat": 4]
for (animalName, legCount) in numberOfLegs {
    println("\(animalName)s have \(legCount) legs")
}
// ants have 6 legs
// cats have 4 legs
// spiders have 8 legs
```

see this: https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html

swift

switch

```
let someCharacter:  Character  = "e"
switch someCharacter {
case "a", "e", "i", "o", "u":
    println("\(someCharacter) is a vowel")
case "b", "c", "d", "f", "g", "h", "j", "k", "l", "m",
"n", "p", "q", "r", "s", "t", "v", "w", "x", "y", "z":
    println("\(someCharacter) is a consonant")
default:
    println("\(someCharacter) is not a vowel or a consonant")
}
// prints "e is a vowel"
```

no pass through

```
let somePoint = (1, 1)
switch somePoint {
case (0, 0):
    println("(0, 0) is at the orig
case (_, 0):
    println("(\(so  oint.0), 0) is on the x-axis")
case (0, _):
    println("(0, \(somePoint.1)) is on the y-axis")
case (-2...2, -2...2):
    println("(\(somePoint.0), \(somePoint.1)) is inside the box")
default:
    println("(\(somePoint.0), \(somePoint.1)) is outside of the box")
}
// prints "(1, 1) is inside the box"
```

“any” value

see this: https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html

swift

switch continued…

```
let anotherPoint = (2, 0)
switch anotherPoint {
case (let x, 0):
    println("on the x-axis with an x value of \(x)")
case (0, let y):
    println("on the y-axis with a y value of \(y)")
case let (x, y):
    println("somewhere else at (\(x), \(y))")
}
// prints "on the x-axis with an x value of 2"
```

"any" value and set

```
let yetAnotherPoint = (1, -1)
switch yetAnotherPoint {
case let (x, y) where x == y:
    println("(\(x), \(y)) is on the line x == y")
case let (x, y) where x == -y:
    println("(\(x), \(y)) is on the line x == -y")
case let (x, y):
    println("(\(x), \(y)) is just some arbitrary point")
}
// prints "(1, -1) is on the line x == -y"
```

very powerful, concise, readable

see this: https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html

swift

functions

internal name

input type

```
func sayHello(personName: String ) ->  String  {
    let greeting = "Hello, " + personName + "!"
    return greeting
}
```

return type

internal name

input type

external name

```
func join(string s1:  String, toString s2: String, withJoiner joiner: String)
    ->  String {
        return s1 + joiner + s2
}
```

return type

external name

passed value

```
join(string: "hello", toString: "world", withJoiner: ", ")
// returns "hello, world"
```

see this: https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html

There are too many ways of defining functions to cover it all. For instance you can also setup default values…

array of ints

```
func minMax(array: [ Int ]) -> (min:  Int , max:  Int ) {
    var currentMin = array[0]
    var currentMax = array[0]
    for value in array[1..<array.count] {
        if value < currentMin {
            currentMin = value
        } else if value > currentMax {
            currentMax = value
        }
    }
    return (currentMin, currentMax)
}
```

return tuple

tuple keys are external names!!

```
let bounds = minMax([8, -6, 2, 109, 3, 71])
println("min is \(bounds.min) and max is \(bounds.max)")
// prints "min is -6 and max is 109"
```

see this: https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html

swift

classes and properties

```
class DataImporter {
    var fileName = "data.txt"
    // the DataImporter class would provide data importing functionality here
}

class DataManager {
    lazy var importer = DataImporter()
    var data = [String]()
    // the DataManager class would provide data management functionality here
}
```

class variable

lazy instantiation

```
let manager = DataManager()
manager.data.append("Some data")
manager.data.append("Some more data")
// the DataImporter instance for the importer property has not yet been created
```

class initialized, but importer is not set

```
println(manager.importer.fileName)
// the DataImporter instance for the importer property has now been created
// prints "data.txt"
```

when accessed first, sets value

see this: https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html

swift

classes and properties

```
class StepCounter {
  var totalSteps: Int  = 0 {
       willSet(newTotalSteps) {
           println("About to set totalSteps to \(newTotalSteps)")
       }
       didSet {
           if totalSteps > oldValue  {
               println("Added \(totalSteps - oldValue) steps")
           }
       }
   }
```

```
let stepCounter = StepCounter()
stepCounter.totalSteps = 200
// About to set totalSteps to 200
// Added 200 steps
stepCounter.totalSteps = 360
// About to set totalSteps to 360
// Added 160 steps
stepCounter.totalSteps = 896
// About to set totalSteps to 896
// Added 536 steps
```

see this: https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html

class methods

```
class Counter {
    var count = 0
    func increment() {
        count++
    }
    func incrementBy(amount: Int) {
        count += amount
    }
    func reset() {
        count = 0
    }
}
```

```
class Counter {
    var count: Int  = 0
    func incrementBy(amount:  Int, numberOfTimes: Int) {
        count += amount * numberOfTimes
    }
}
```

see this: https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html

see this: https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html

Lots more on the inter-webs!

Need more help on MVC's ? Check out Ray Wenderlich:

http://www.raywenderlich.com/46988/ios-design-patterns

for next time…

- View Controllers in iOS
  - Watch videos **before class**
- Come ready to work in teams on an in class project