

<https://github.com/GroundSpeed/SwiftlyBreakingGround>

[PDF: http://sl.gshq.co/BreakingGround](http://sl.gshq.co/BreakingGround)

# BREAKING GROUND WITH IOS

Don Miller  
Founder of GroundSpeed™  
Co-founder of Seed Coworking  
@donmiller



# SCHEDULE

Introductions

Discuss Terminology, Xcode Overview

Create Your First App: Hello World!

MVC Overview

Color Me Demo

Swift Basics and the Playground

Storyboards

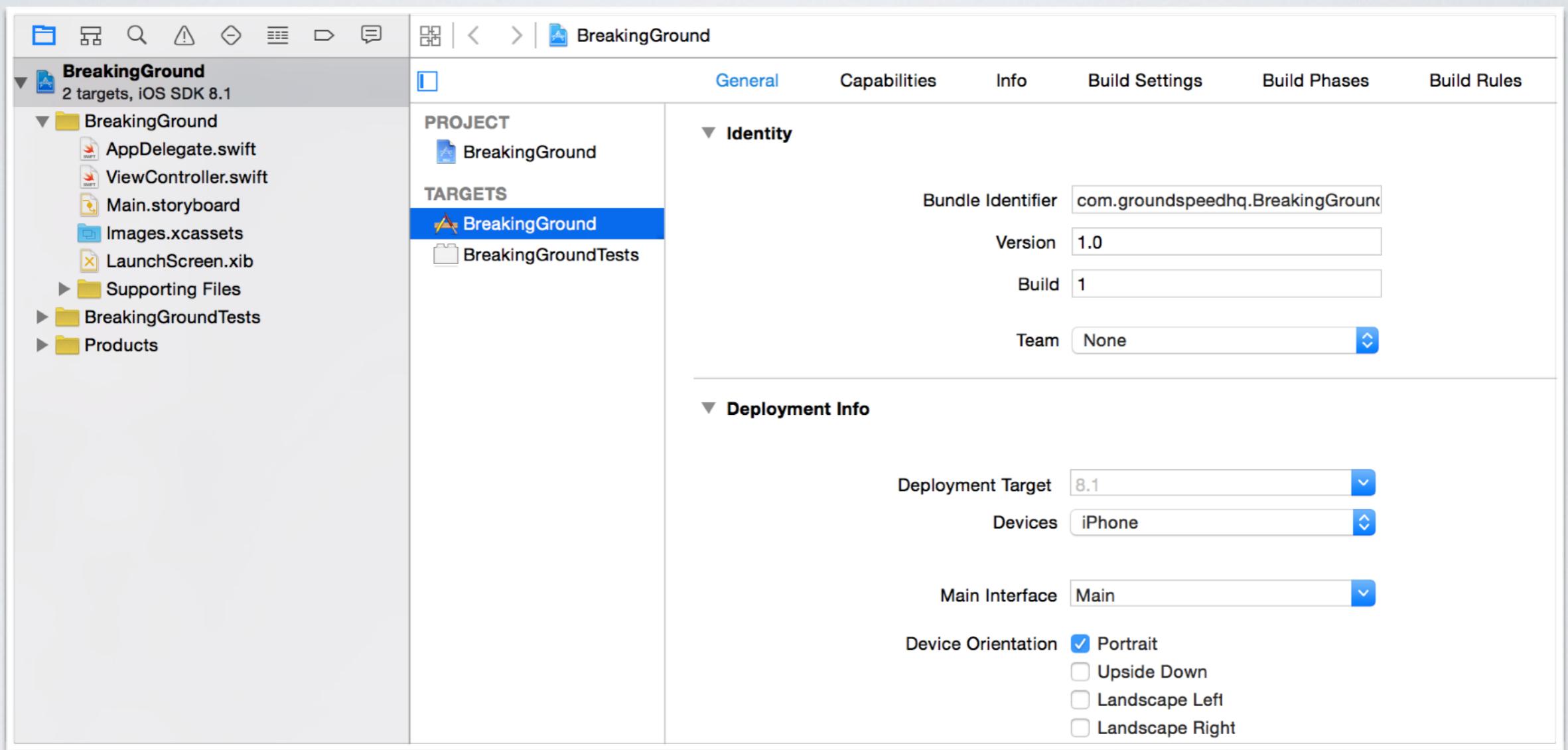


# TERMINOLOGY

- ▶ iOS
- ▶ iPhone
- ▶ iPad
- ▶ iPod Touch
- ▶ MacBook
- ▶ iMac
- ▶ SDK
- ▶ Xcode
- ▶ OSX
- ▶ Sierra (10.12)
- ▶ El Capitan (10.11)
- ▶ Yosemite (10.10)
- ▶ Mavericks,  
Mountain Lion,  
etc.
- ▶ Retina



# XCODE 8.X & IOS 10.X



# HELLO WORLD!



# CREATE A NEW SINGLE VIEW APPLICATION

Choose a template for your new project:

iOS watchOS tvOS macOS Cross-platform  Filter

**Application**

 Single View Application	 Game	 Master-Detail Application	 Page-Based Application	 Tabbed Application
 Sticker Pack Application	 iMessage Application			

**Framework & Library**

 Cocoa Touch Framework	 Cocoa Touch Static Library	 Metal Library
--------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------



# ENTER THE PRODUCT NAME

Choose options for your new project:

Product Name:

Team:  ▼

Organization Name:

Organization Identifier:

Bundle Identifier: com.groundspeedhq.HelloWorld

Language:  ▼

Devices:  ▼

Use Core Data

Include Unit Tests

Include UI Tests

Cancel

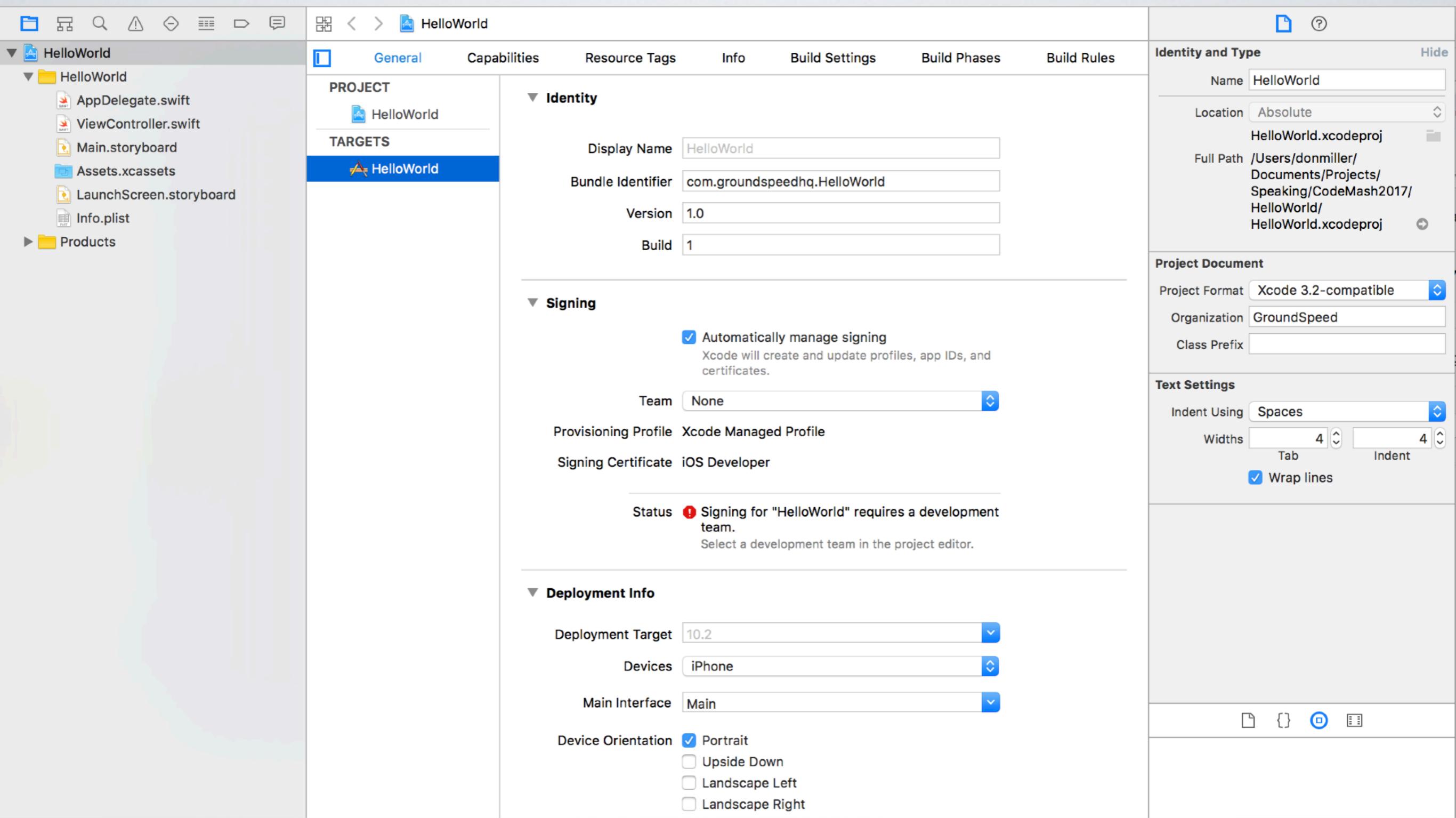
Previous

Next



GroundSpeed™  
rapid web + mobile software

# UPDATE THE GENERAL PROJECT ATTRIBUTES



The screenshot shows the Xcode interface for updating general project attributes. The left sidebar lists files like AppDelegate.swift, ViewController.swift, Main.storyboard, Assets.xcassets, LaunchScreen.storyboard, and Info.plist. The main area is the 'General' tab of the project settings, showing the 'Identity' section with fields for Display Name (HelloWorld), Bundle Identifier (com.groundspeedhq.HelloWorld), Version (1.0), and Build (1). The 'Signing' section has 'Automatically manage signing' checked and 'Team' set to 'None'. The 'Deployment Info' section shows Deployment Target (10.2), Devices (iPhone), Main Interface (Main), and Device Orientation (Portrait selected). The right side displays the 'Identity and Type' panel with project details: Name (HelloWorld), Location (Absolute), Full Path (/Users/donmiller/Documents/Projects/Speaking/CodeMash2017>HelloWorld>HelloWorld.xcodeproj). It also includes sections for Project Document (Project Format Xcode 3.2-compatible, Organization GroundSpeed) and Text Settings (Indent Using Spaces, Wrap lines checked).

**Identity and Type**

Name HelloWorld

Location Absolute

Full Path /Users/donmiller/Documents/Projects/Speaking/CodeMash2017>HelloWorld>HelloWorld.xcodeproj

**Project Document**

Project Format Xcode 3.2-compatible

Organization GroundSpeed

Class Prefix

**Text Settings**

Indent Using Spaces

Widths 4 Tab 4 Indent

Wrap lines

**General** Capabilities Resource Tags Info Build Settings Build Phases Build Rules

**PROJECT**

>HelloWorld

**TARGETS**

>HelloWorld

Display Name HelloWorld

Bundle Identifier com.groundspeedhq.HelloWorld

Version 1.0

Build 1

**Signing**

Automatically manage signing  
Xcode will create and update profiles, app IDs, and certificates.

Team None

Provisioning Profile Xcode Managed Profile

Signing Certificate iOS Developer

Status ! Signing for "HelloWorld" requires a development team.  
Select a development team in the project editor.

**Deployment Info**

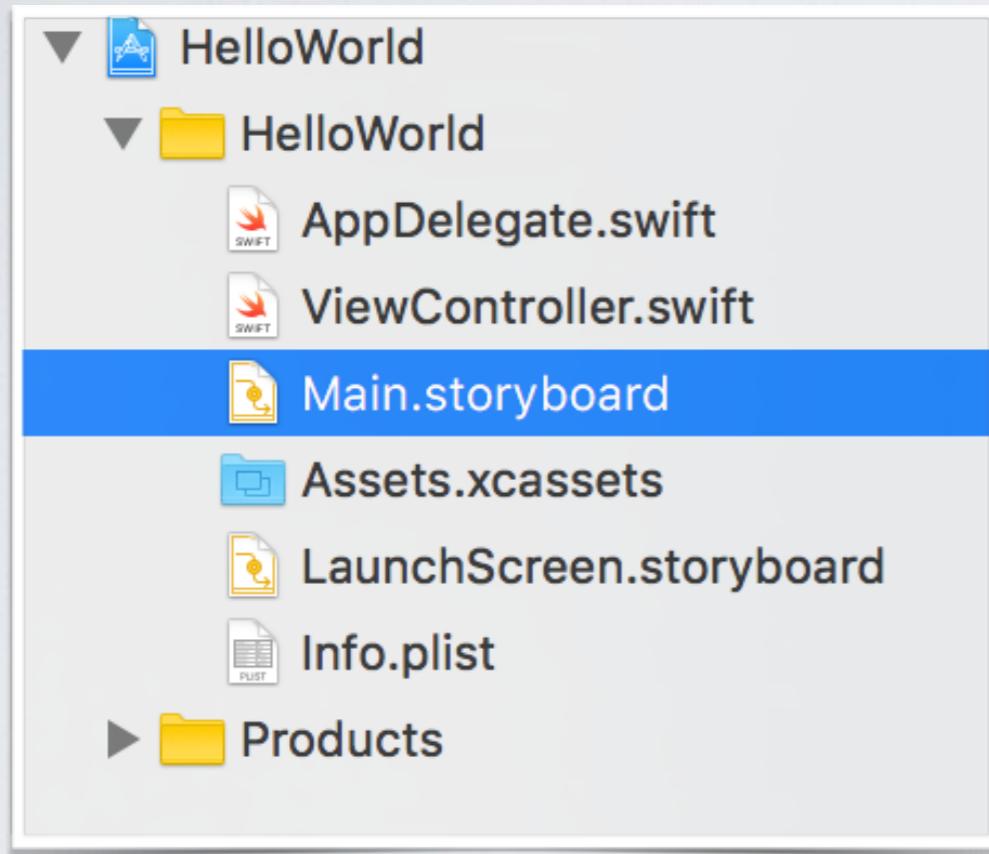
Deployment Target 10.2

Devices iPhone

Main Interface Main

Device Orientation  Portrait  
 Upside Down  
 Landscape Left  
 Landscape Right

# PROJECT STRUCTURE



- Folders do not match physical layout on file system.
- Actually you will not find any sub folder named like Supporting Files in your physical project folder.
- The folder hierarchy in the XCode's navigator is simply a logical grouping of the resources.
- At this stage, we would be interested only on the Main.storyboard file.



# FILE INSPECTOR & SCHEMA SETTINGS

The screenshot shows the Xcode interface with two main panels: the File Inspector on the left and the Schema Settings on the right.

**File Inspector (Left Panel):**

- Identity and Type:**
  - Name: Main.storyboard
  - Type: Default - Interface Builder...
  - Location: Relative to Group
  - Base.lproj/Main.storyboard
  - Full Path: /Users/donmiller/Documents/Projects/Speaking/CodeMash2017/HelloWorld/HelloWorld/Base.lproj/Main.storyboard
- On Demand Resource Tags:** Tags
- Interface Builder Document:**
  - Opens in: Latest Xcode (8.0)
  - Builds for: Deployment Target (10.2)
  - Use Auto Layout
  - Use Trait Variations
  - Use as Launch Screen
- Global Tint: Default

**Schema Settings (Right Panel):**

- Xcode menu bar: Apple, Xcode, File, Edit, View, Find, Navigate
- Toolbar: Stop, Run, Build, Scheme (HelloWorld), Device (iPhone 7)
- Generic iOS Device
- iOS Simulators
  - iPad Air
  - iPad Air 2
  - iPad Pro (9.7 inch)
  - iPad Pro (12.9 inch)
  - iPad Retina
  - iPhone 5
  - iPhone 5s
  - iPhone 6
  - iPhone 6 Plus
  - iPhone 6s
  - iPhone 6s Plus
  - iPhone 7
  - iPhone 7 Plus
  - iPhone SE
- Add Additional Simulators...
- Download Simulators...



# DEVICES & TARGETS

The screenshot shows the Xcode project settings for a target named "HelloWorld".

**General Tab:**

- PROJECT:** HelloWorld
- TARGETS:** HelloWorld

**Identity Section:**

- Display Name: HelloWorld
- Bundle Identifier: com.groundspeedhq.HelloWorld
- Version: 1.0
- Build: 1

**Signing Section:**

- Automatically manage signing  
Xcode will create and update profiles, app IDs, and certificates.
- Team: None
- Provisioning Profile: Xcode Managed Profile
- Signing Certificate: iOS Developer

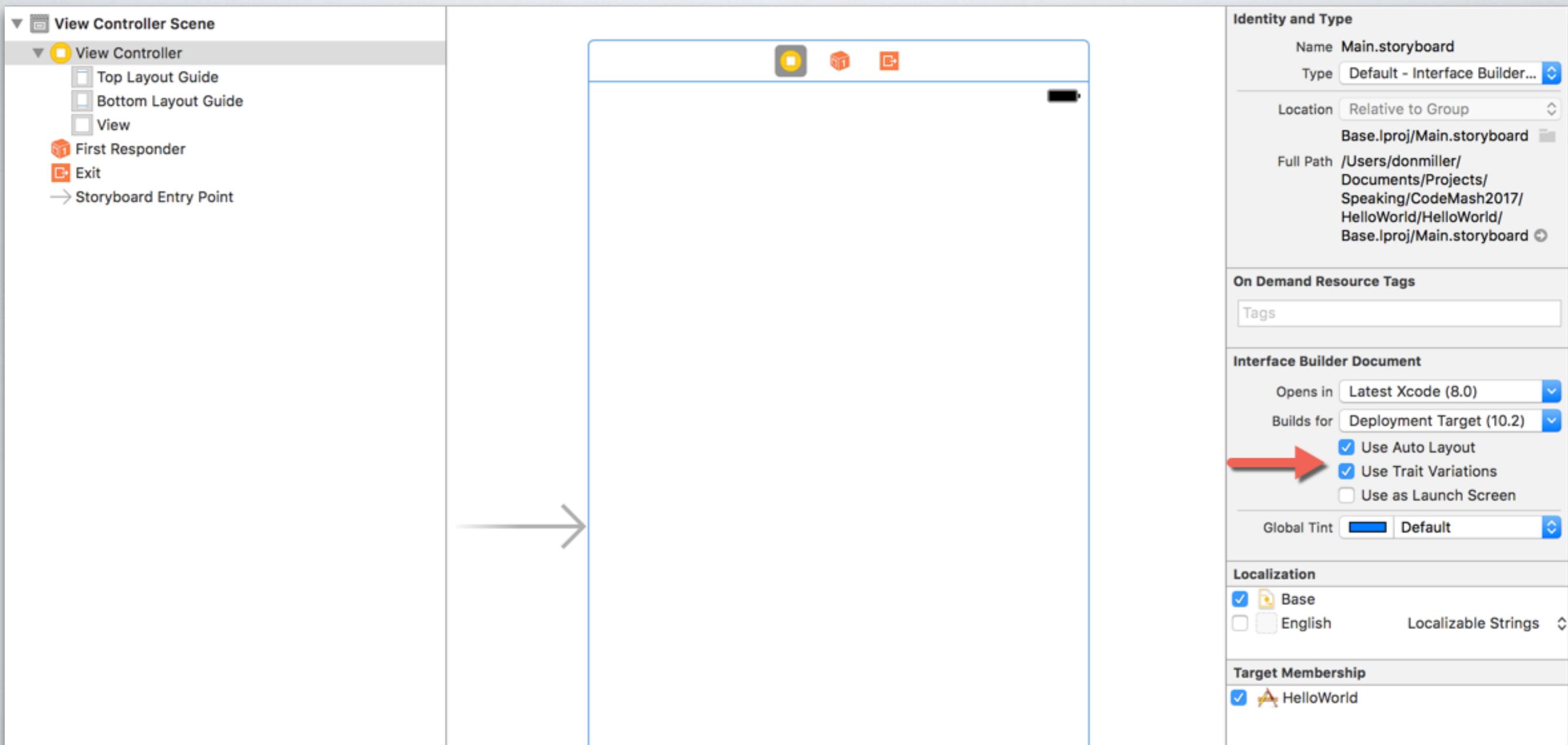
**Status:** ! Signing for "HelloWorld" requires a development team.  
Select a development team in the project editor.

**Deployment Info Section:**

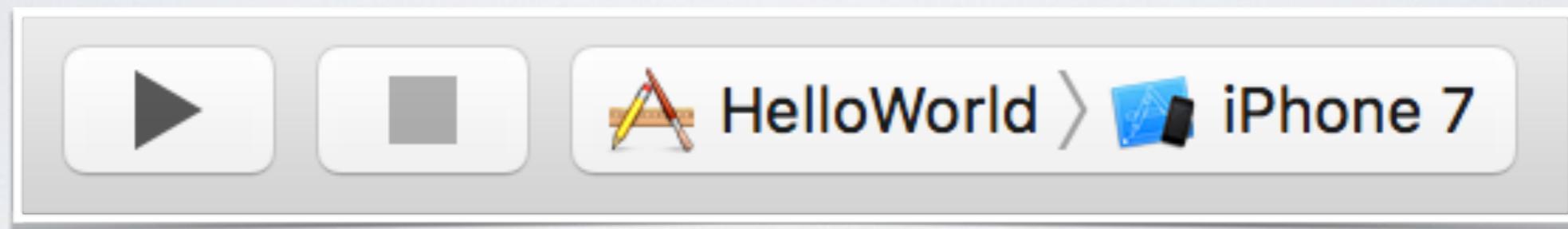
- Deployment Target: 10.2
- Devices: iPhone
- Main Interface: Main
- Device Orientation:
  - Portrait
  - Upside Down
  - Landscape Left
  - Landscape Right
- Status Bar Style: Default
  - Hide status bar
  - Requires full screen



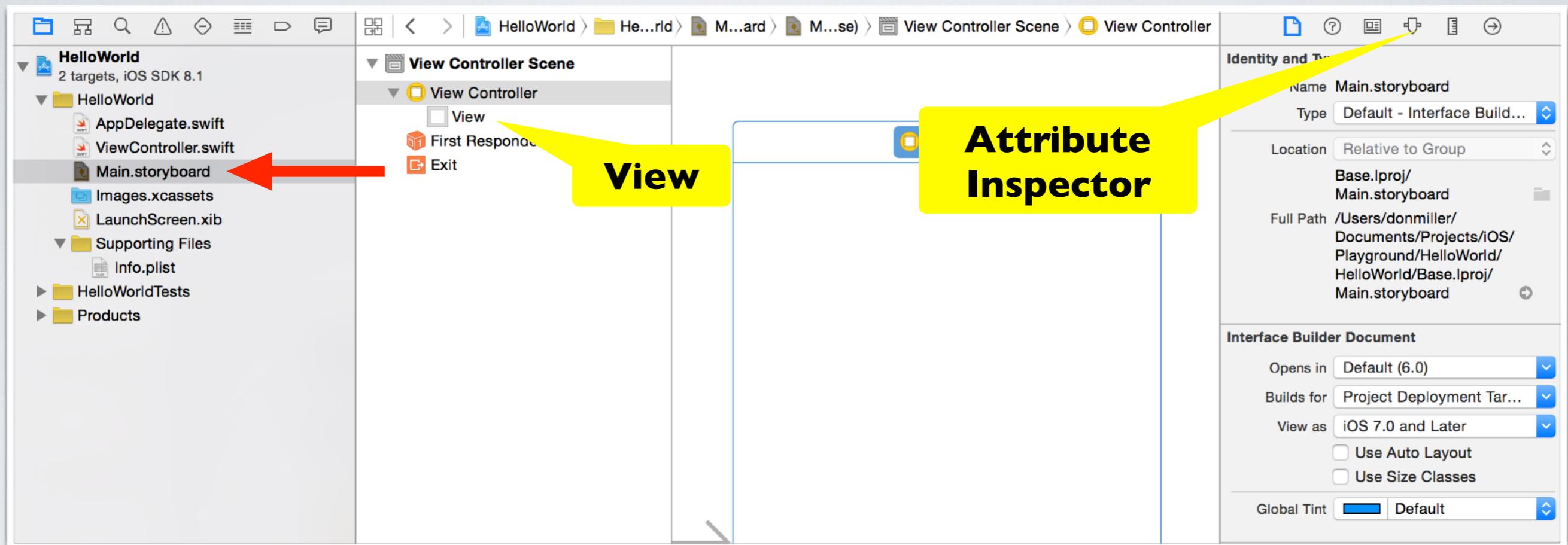
# FILE OWNER, VIEW, & FIRST RESPONDER



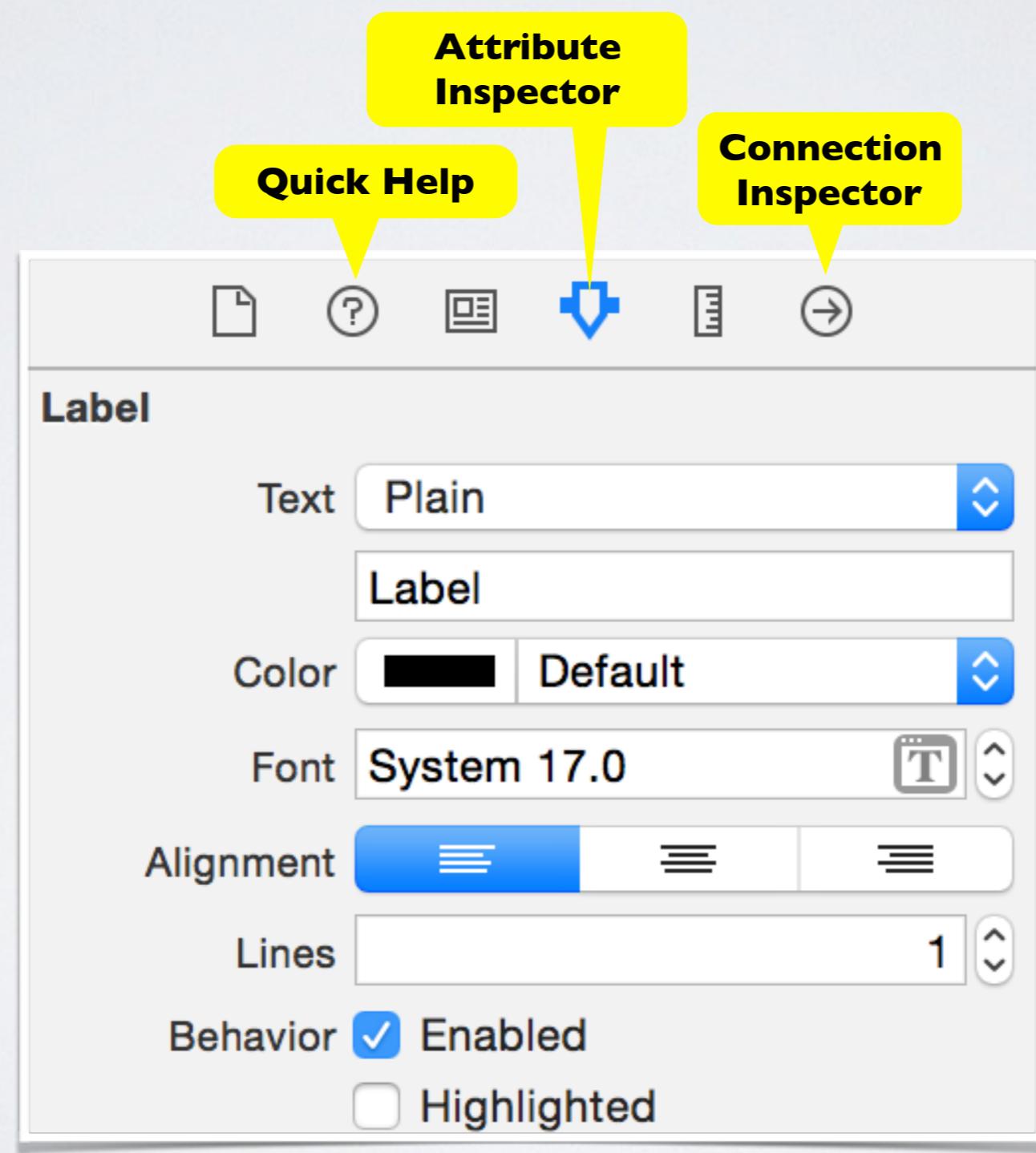
# RUN THE APPLICATION BY PRESSING PLAY



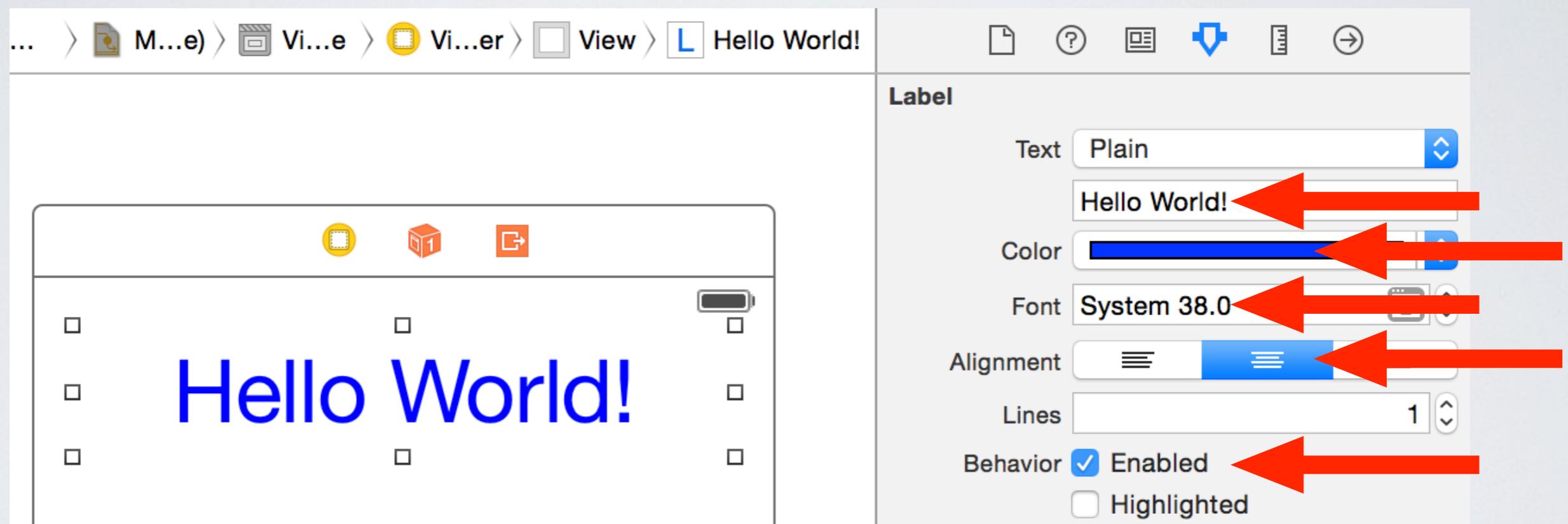
# ATTRIBUTE INSPECTOR



# INSPECTORS



# CREATE “HELLOWORLD!” LABEL



# Lab #1

## Hello World!

### Take 15 Minutes to Build It



# SWIFT BASICS

Xcode Playgrounds

Variables & Constants (Mutable and Immutable)

Types

- Strings, Integers, Floats, Booleans

Operators

- Binary and Unary Operators

Collection Types

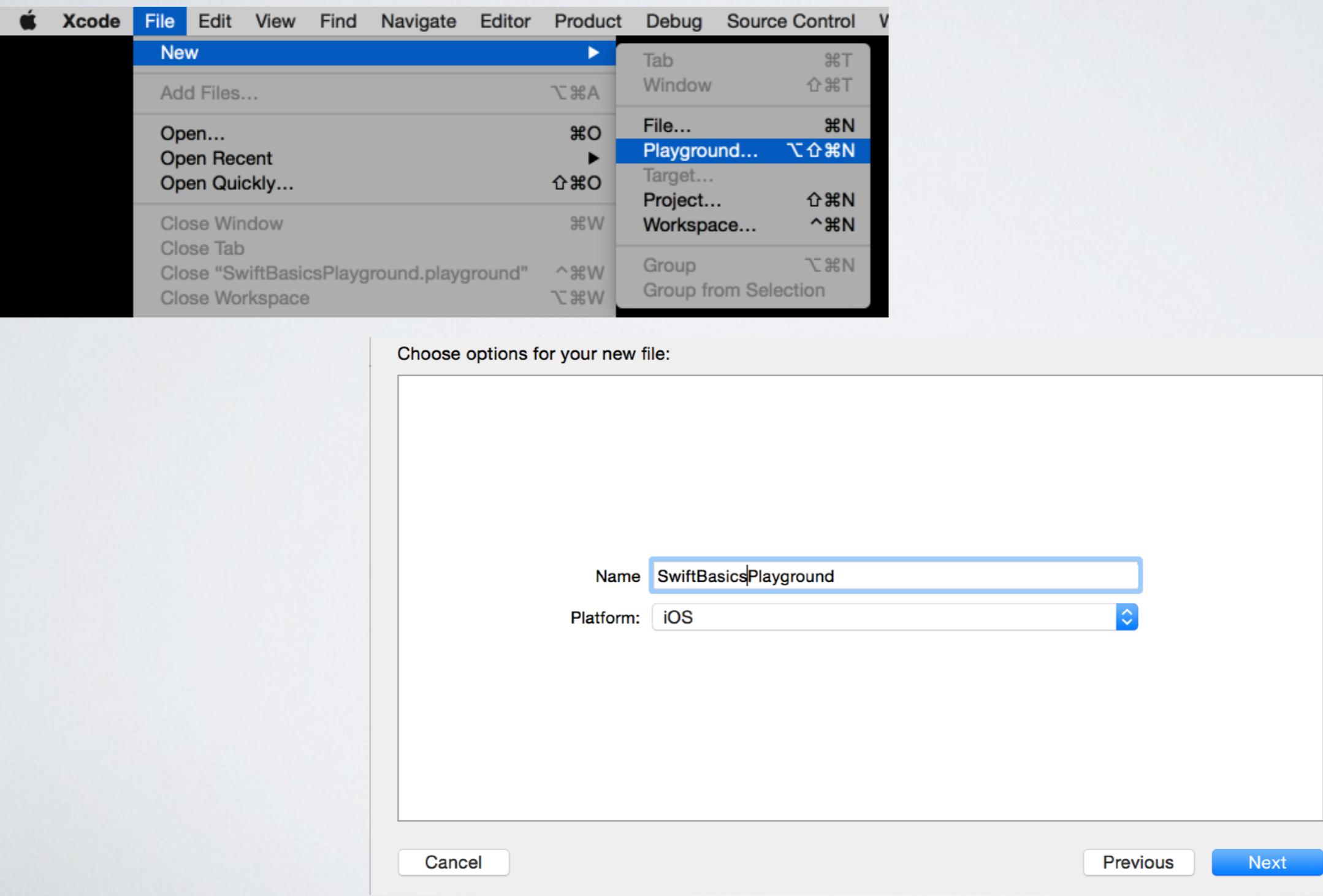
- Arrays and Dictionaries

Control Flow

- For-In, While, For-Condition, If, Switch, Comparison and Logical Operators



# EXAMPLE: HAVE FUN IN THE PLAYGROUND



# ABOUT MUTABLE AND IMMUTABLE OBJECTS OR VARIABLES AND CONSTANTS

```
3 import UIKit
4
5 let immutable = "This cannot be changed"
6 immutable = "This will give an error"!
7
8 var mutable = "This cannot be changed"
9 mutable = "This will NOT give an error"
10
```

"This cannot be changed"  
"This will give an error"  
"This cannot be changed"  
"This will NOT give an error"



# DECLARING TYPES: STRINGS, INTEGERS, FLOATS, AND BOOLEANS

5	let string1 = "This is a string"	"This is a string"
6	let string2: String = "This is string typed"	"This is string typed"
7		
8	var intNumber1 = 1	1
9	intNumber1 = 2.5	
10	! Type 'Int' does not conform to protocol 'FloatLiteralConvertible'	
11	let intNumber2: Int = 3	3
12		
13	var floatNumber1 = 1.0	1.0
14	floatNumber1 = 2	2.0
15	floatNumber1 = 2.5	2.5
16		
17	let floatNumber2: Float = 1	1.0
18		
19	let bool1 = true	true
20	let bool2: Bool = false	false
21		

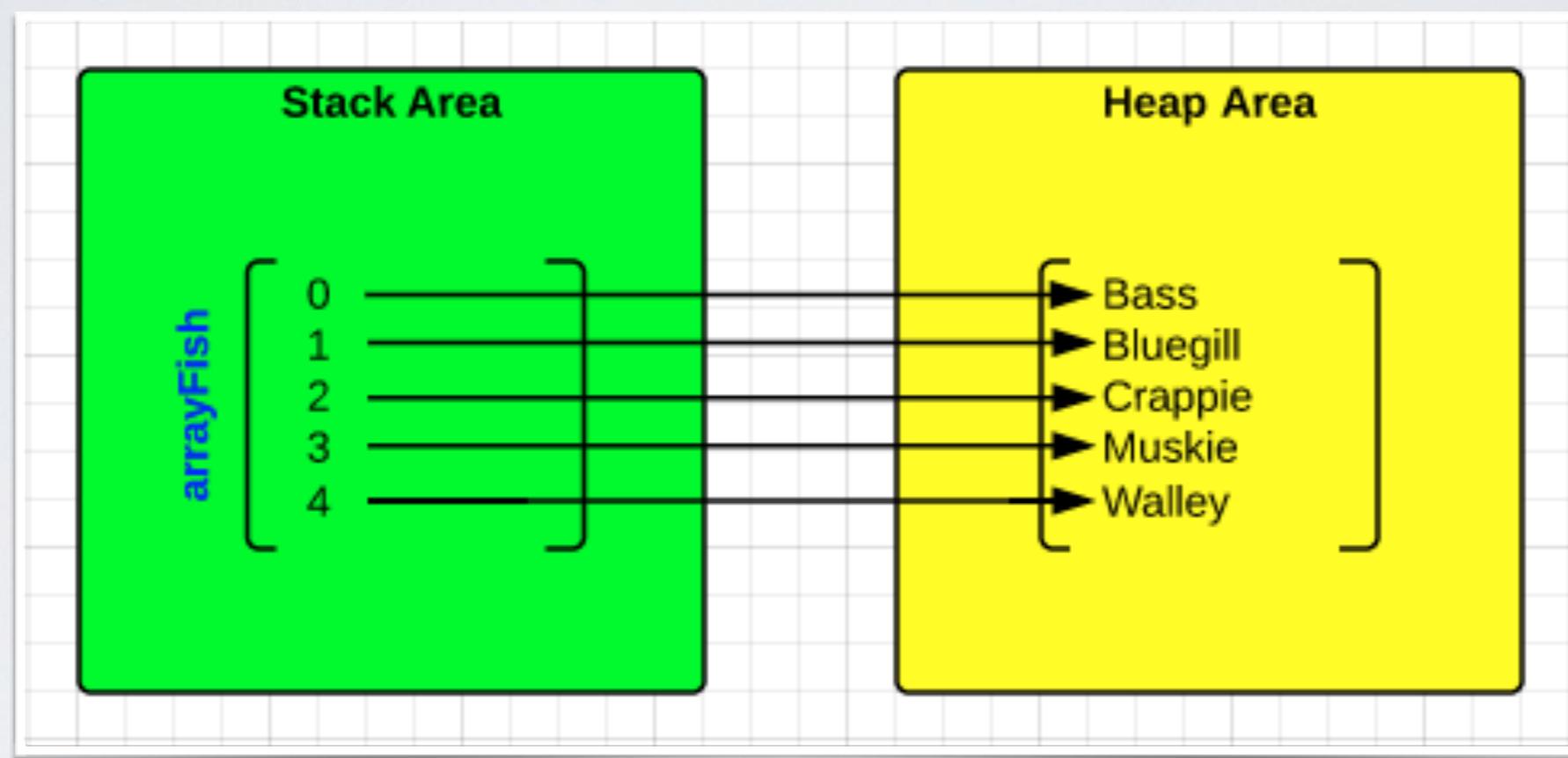


# OPERATORS: BINARY AND UNARY ...AND MAYBE A TERNARY TOO

```
5 // binary
6 var a = 1
7 a = a + 1
8 a = a - 1
9 a = a / 1
10 a = a * 4
11 a = a % 3
12
13 // unary
14 var b = 1
15 b += 1
16 b += 1
17 b
18 b -= 1
19 b
20
21 // ternary
22 var bool = true
23 var c = (bool ? 1 : 3)
24 bool = false
25 c
26 c = (bool ? 1 : 3)
27
```



# ARRAYS



```
5 var arrayFish = ["Bass", "Bluegill",
  "Crappie", "Muskie", "Walley"]
6
7 arrayFish.count
8 arrayFish[3]
9 arrayFish[5] ⚠ Execution was interrupted, reason: EXC_... [!] error
```

```
["Bass", "Bluegill", "Crappie", "Muskie", "Walley"]
```

```
5
```

```
"Muskie"
```



# CREATING ARRAYS AND DICTIONARIES WITH SWIFT

```
1 // Arrays
2 let arrayFish = ["Bass", "Bluegill", "Crappie", "Muskie", "Walley"]
3
4 arrayFish.count
5 arrayFish[3]
6
7
8 // Dictionaries
9 let dictStates = ["OH": "Ohio", "MI": "Michigan", "KY": "Kentucky", "IN": "Indiana", "IL": "Illinois", "PA": "Pennsylvania", "NY": "New York"]
10
11 dictStates.count
12 dictStates["OH"]!
13
14 // For Loop through dictionary
15 for (stateCode, stateName) in dictStates {
16     print("\(stateCode): \(stateName)")
17 }
18
19 for stateCode in dictStates.keys {
20     print("\(stateCode)")
21 }
22
23 for stateName in dictStates.values {
24     print("\(stateName)")
25 }
26
27 // Array of stateCodes
28 let stateCodes = [String](dictStates.keys)
```



# CONTROL FLOW - LOOPING

```
5 // For-In
6 for index in 1...5 {
7     print("\(index) times 5 is \(index * 5)")
8 }
9
10 let arrayFish = ["Bass", "Bluegill", "Crappie", "Muskie", "Walley"]
11 for fish in arrayFish {
12     print("Fish array: \(fish)")
13 }
14
15 // For-In with <
16 for index in 0 ..< 5 {
17     print("For Index is \(index)")
18 }
19
20 // While
21 var index = 0
22 var max = 5
23 while index < max {
24     print("While Index is \(index += 1)")
25 }
26
27 // Repeat-While
28 index = 0
29 max = 5
30 repeat {
31     print("Repeat-While Index is \(index += 1)")
32 } while index < max
33
```

## Console Output

```
1 times 5 is 5
2 times 5 is 10
3 times 5 is 15
4 times 5 is 20
5 times 5 is 25
Fish array: Bass
Fish array: Bluegill
Fish array: Crappie
Fish array: Muskie
Fish array: Walley
For Index is 0
For Index is 1
For Index is 2
For Index is 3
For Index is 4
While Index is ()
Repeat-While Index is ()
```



# CONTROL FLOW - DECISION

1 // IF	90
2 var temperatureInFahrenheit = 90	
3 if temperatureInFahrenheit <= 32 {	
4     print("It's very cold. Consider wearing a scarf.")	
5 } else if temperatureInFahrenheit >= 86 {	
6     print("It's really warm. Don't forget to wear sunscreen.")	"It's really warm. Don't forget to wear sunscreen.\n"
7 } else {	
8     print("It's not that cold. Wear a t-shirt.")	
9 }	
10	
11 // SWITCH	
12 let someCharacter: Character = "a"	"a"
13 switch someCharacter {	
14 case "a", "e", "i", "o", "u":	
15     print("\\$(someCharacter) is a vowel")	"a is a vowel\n"
16 case "b", "c", "d", "f", "g", "h", "j", "k", "l", "m",	
17 "n", "p", "q", "r", "s", "t", "v", "w", "x", "y", "z":	
18     print("\\$(someCharacter) is a consonant")	
19 default:	
20     print("\\$(someCharacter) is not a vowel or a consonant")	
21 }	



# CLASS ILLUSTRATIONS

```
5 class Person {  
6     var ssn: String!  
7     var name: String!  
8     var city: String!  
9     var state: String!  
10 }  
11  
12 class Employee: Person {  
13     var empId: String!  
14     var department: String!  
15     var healthInsurance: String!  
16  
17     func printChecks() {  
18         print("Printing Checks")  
19     }  
20 }  
21  
22 class SalariedEmployee: Employee {  
23     var salary: String!  
24 }  
25  
26 class HourlyEmployee: Employee {  
27     var hourlyRate: String!  
28 }
```

```
30 let hourlyEmployee = HourlyEmployee()  
31 hourlyEmployee.name = "Don Miller"  
32 hourlyEmployee.empId = "1234"  
33 hourlyEmployee.hourlyRate = "9.99"  
34  
35 var output = "Name: \(hourlyEmployee.name!) \r"  
36 output += "Employee ID is \(hourlyEmployee.empId!) \r"  
37 output += "Hourly wage is \(hourlyEmployee.hourlyRate!) \r"  
38  
39 print("\(output)")  
40 hourlyEmployee.printChecks()
```

## Console Output

```
Name: Don Miller  
Employee ID is 1234  
Hourly wage is 9.99  
Printing Checks
```



# Need a Break?



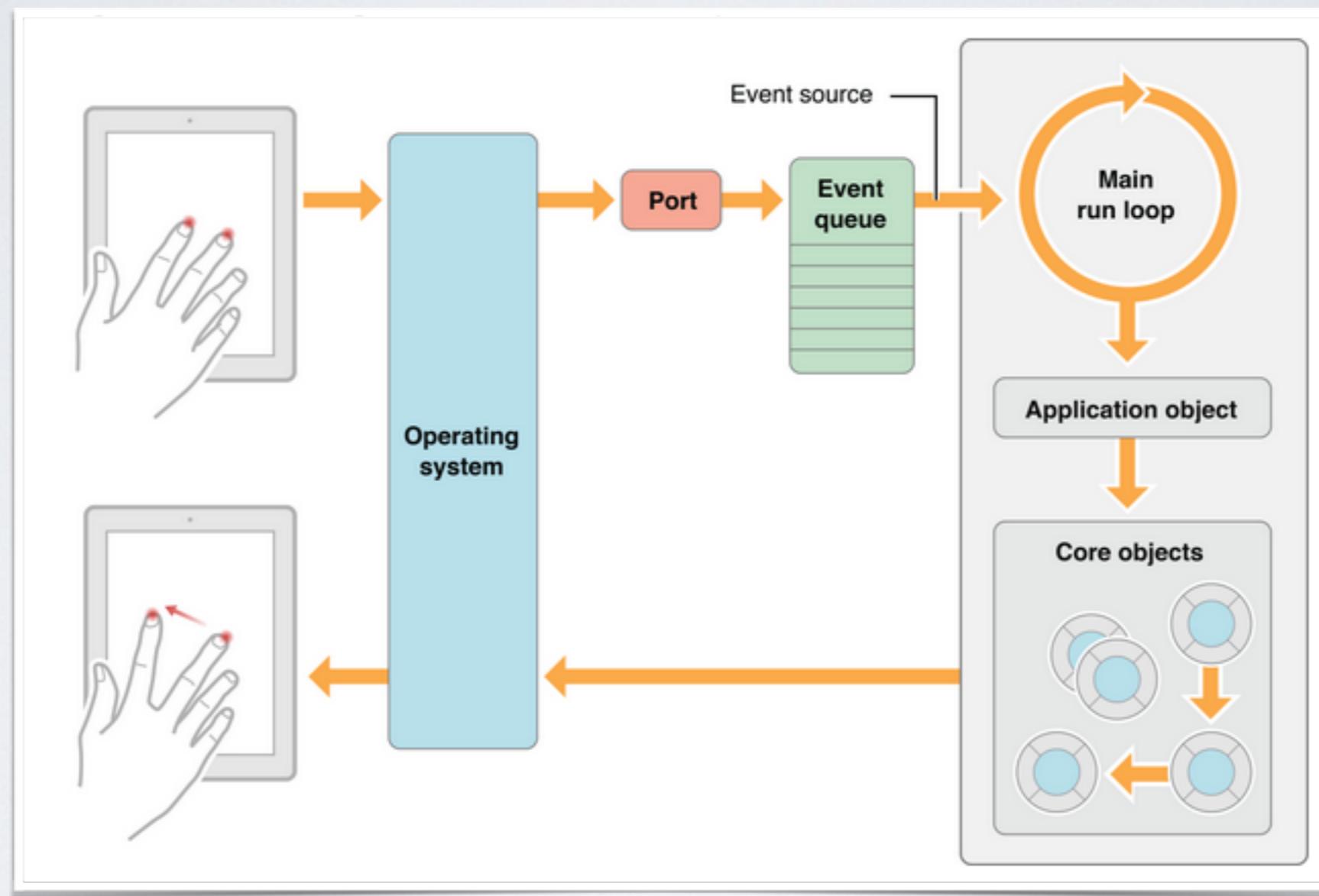
GroundSpeed™  
rapid web + mobile software

# MVC, Storyboards, and Events



GroundSpeed™  
rapid web + mobile software

# IOS APPLICATION RUNTIME CYCLE

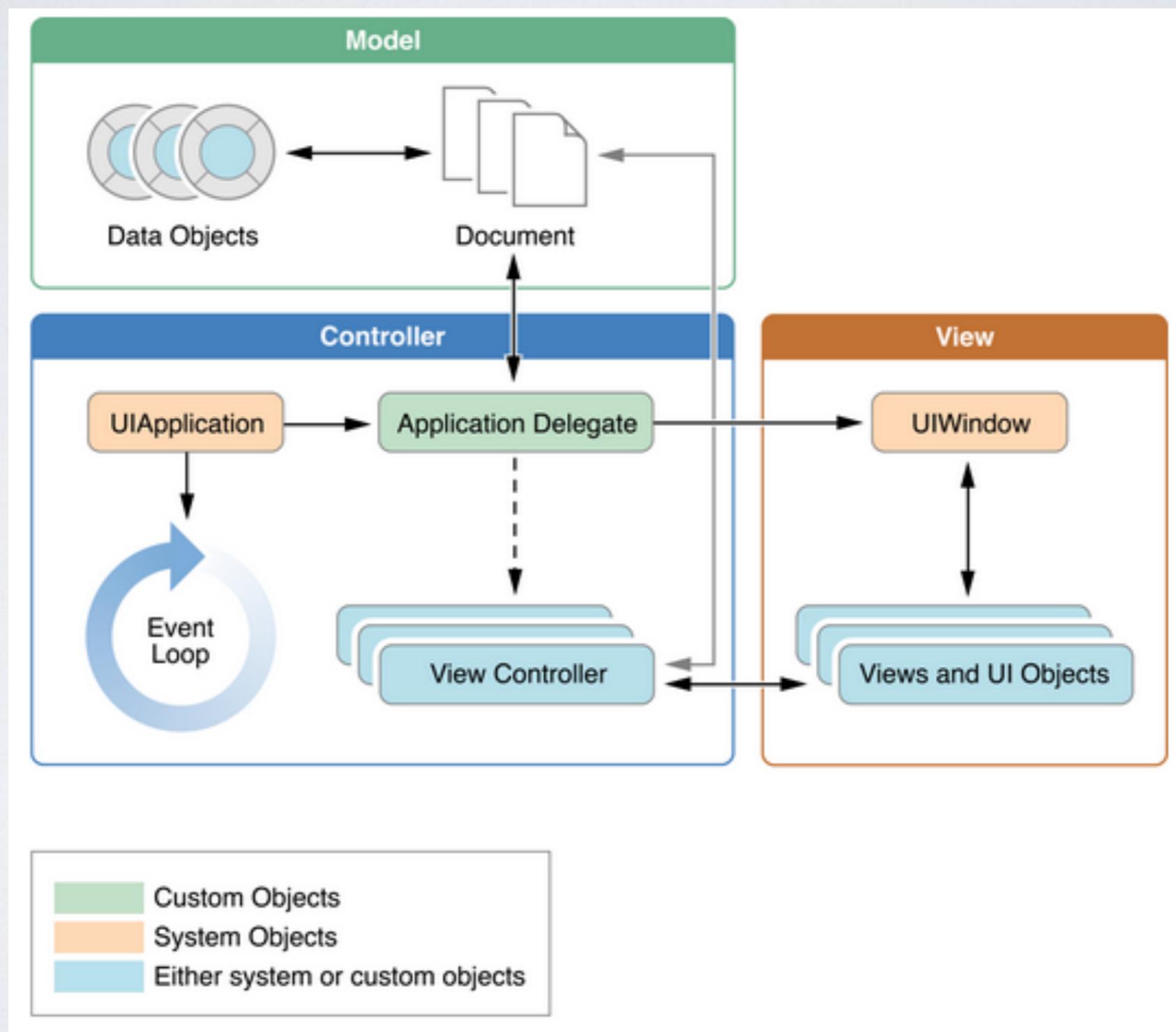


# COMMON TYPES OF EVENTS

Event type	Delivered to...	Notes
Touch	The view object in which the event occurred	Views are responder objects. Any touch events not handled by the view are forwarded down the responder chain for processing.
Remote control Shake motion events	First <a href="#">responder object</a>	Remote control events are for controlling media playback and are generated by headphones and other accessories.
Accelerometer Magnetometer Gyroscope	The object you designate	Events related to the accelerometer, magnetometer, and gyroscope hardware are delivered to the object you designate.
Location	The object you designate	You register to receive location events using the Core Location framework. For more information about using Core Location, see <a href="#">Location and Maps Programming Guide</a> .
Redraw	The view that needs the update	Redraw events do not involve an event object but are simply calls to the view to draw itself. The drawing architecture for iOS is described in <a href="#">Drawing and Printing Guide for iOS</a> .



# MODEL - VIEW - CONTROLLER



# MODEL - VIEW - CONTROLLER

**The Model:** The M in MVC stands for the Model: For the Model, we will need to develop an independent class (or a set of classes) that will serve the data requirements and standard business procedures.

**The View:** In this context the View is the UI screen's layout (the canvas). Here, we configure many User Interface items (also known as UI controls on the screen).

**The Controller:** This is a class that connects the view with the model. In its simplest form, it contains the name of the UIControls (that would be used in the app), and it also includes the signatures and the detail implementations of event handlers.



# STORYBOARD OVERVIEW

Storyboards are graphic organizers displayed in sequence for the purpose of pre-visualizing a motion picture or animation. The storyboarding process, in the form it is known today, is used by developers to give the designer, developer, and end user an opportunity to see what the application or web site will look like at a high level.

In iOS, a Storyboard is a container of screens that may have embedded view controllers, navigation controllers, and tab bar controllers. It allows the user to graphically connect screens to one another through embedded objects on the screen. These connections are called segues (pronounced seg-wey). The storyboard eliminates the need for xib files and has one file that contains all of your screens and transitions. It is possible to connect two storyboard files with one another; however, unlikely with smaller applications.



# PROS OF STORYBOARDS

## Pros of using Storyboards:

- With a storyboard you have a better conceptual overview of all the screens in your app and the connections between them. It's easier to keep track of everything because the entire design is in a single file, rather than spread out over many separate nibs.
- The storyboard describes the transitions between the various screens. These transitions are called “segues” and you create them by simply ctrl-dragging from one view controller to the next. Thanks to segues you need less code to take care of your UI.
- Storyboards make working with table views a lot easier with the new prototype cells and static cells features. You can design your table views almost completely in the storyboard editor, something else that cuts down on the amount of code you have to write.



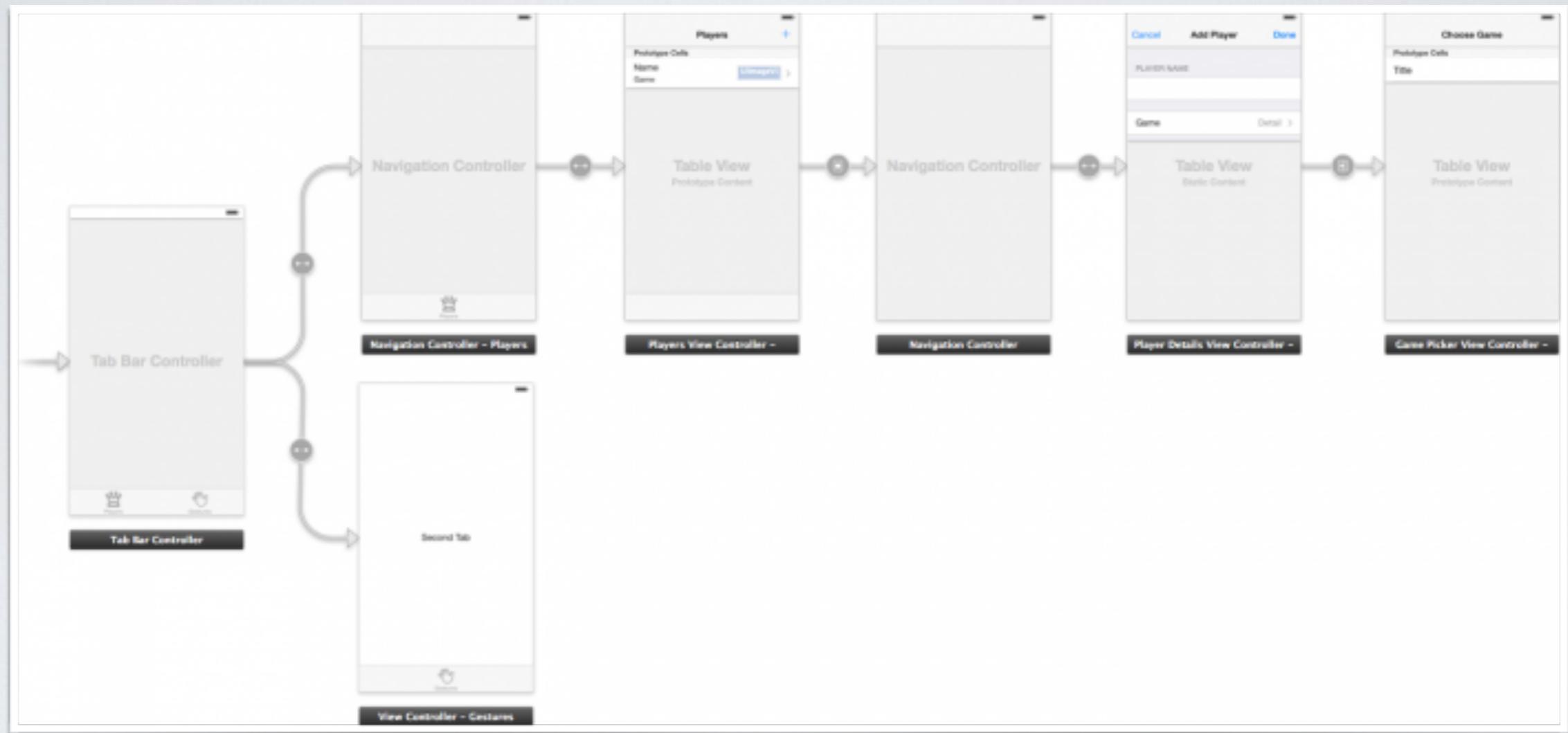
# CONS OF STORYBOARDS

## Cons of using Storyboards:

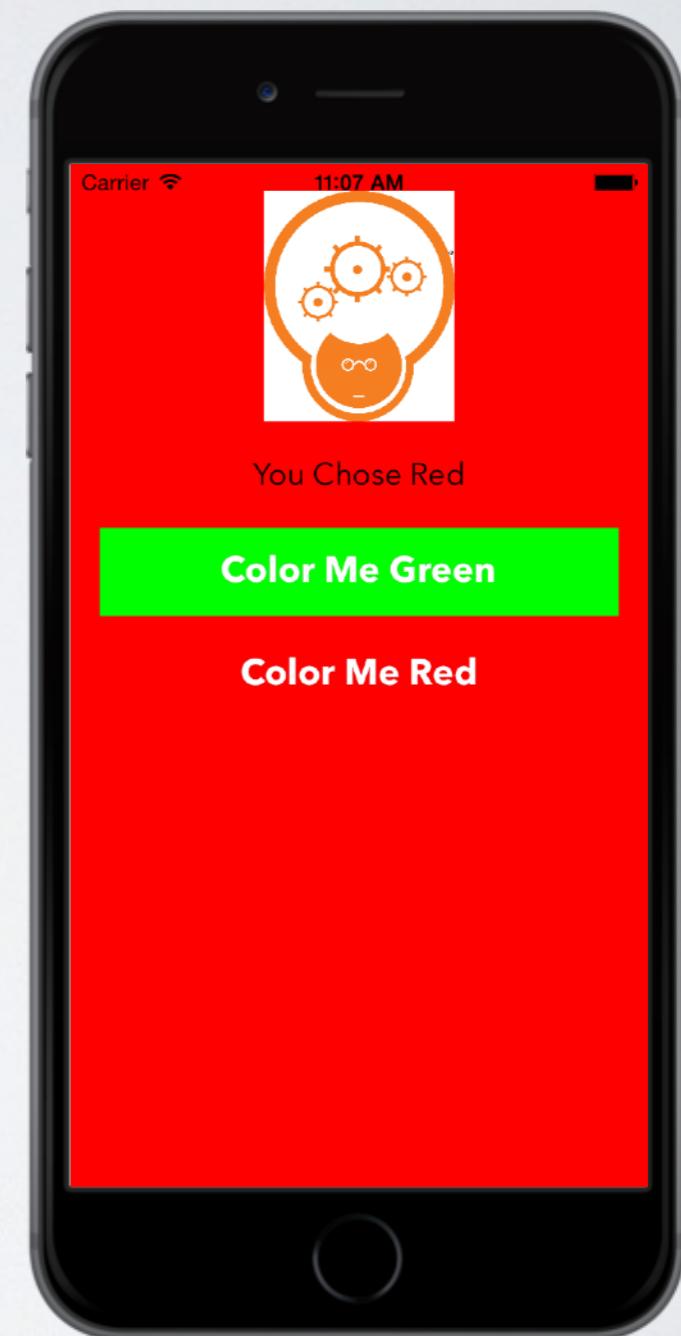
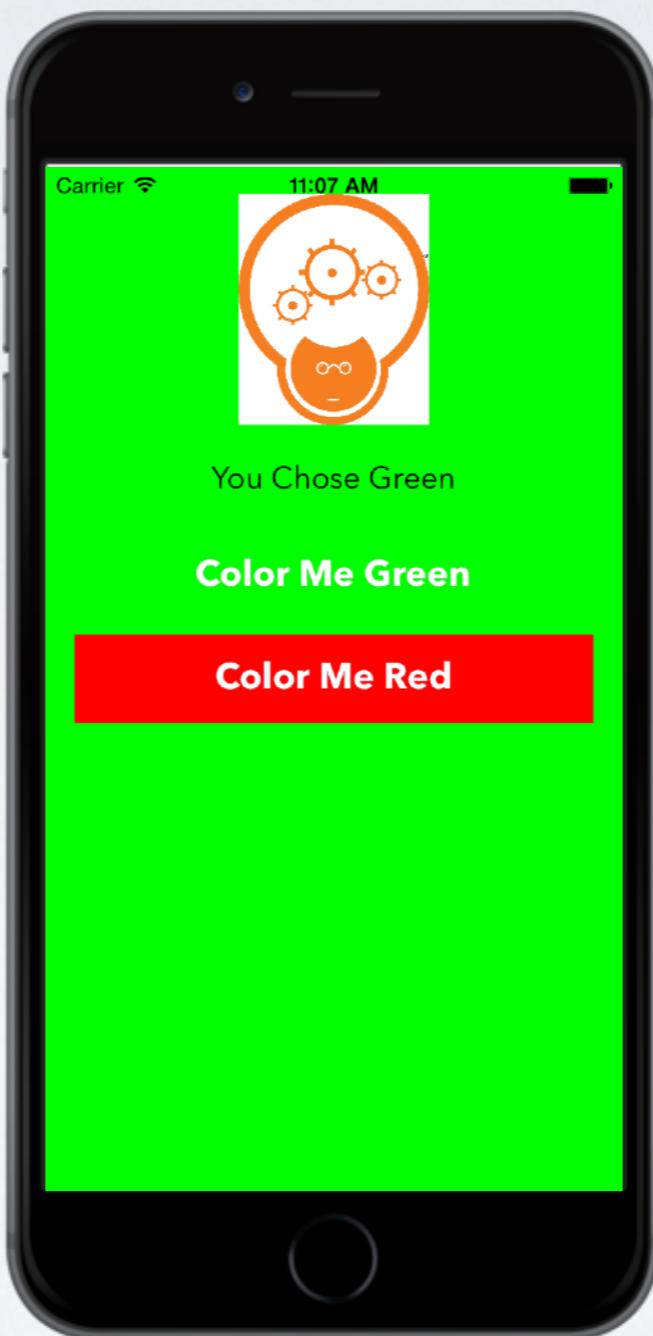
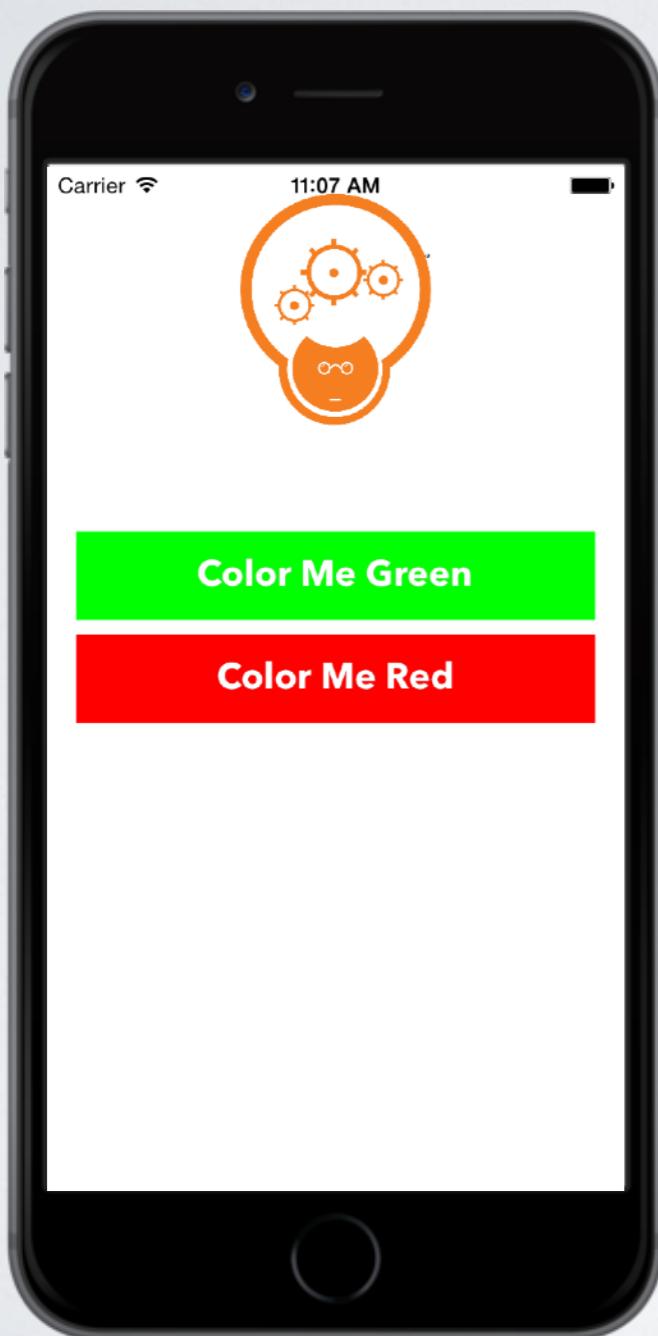
- It's not easy to work with storyboards in a team, since only one participant can work on the storyboard at once (because it's one binary file).
- If you need to do things storyboard doesn't offer, it's not quite easy to get storyboard mixed with programmatically created views)
- Since all objects are in one file, a larger project could cause the storyboard to be very slow and unresponsive. It loads the entire thing into memory. It loads the entire item into memory.



# STORYBOARD OVERVIEW

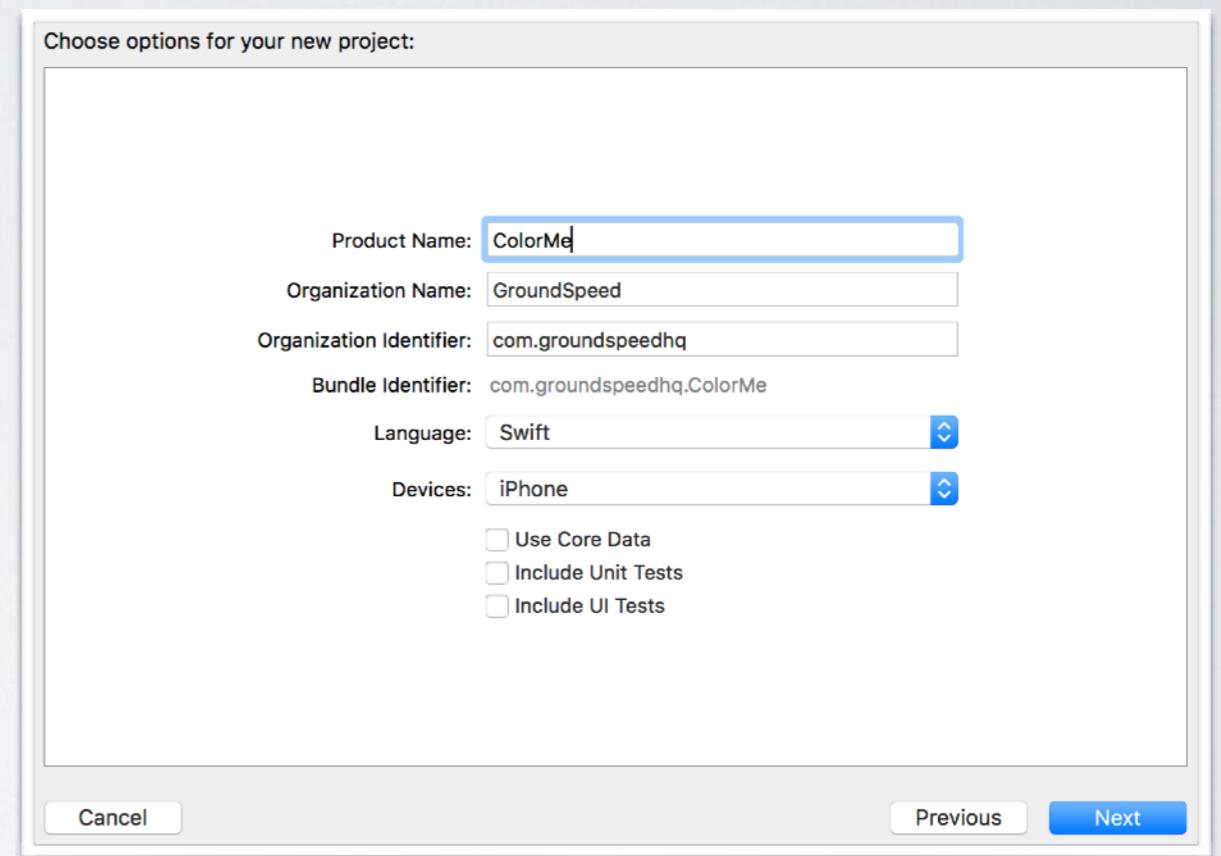
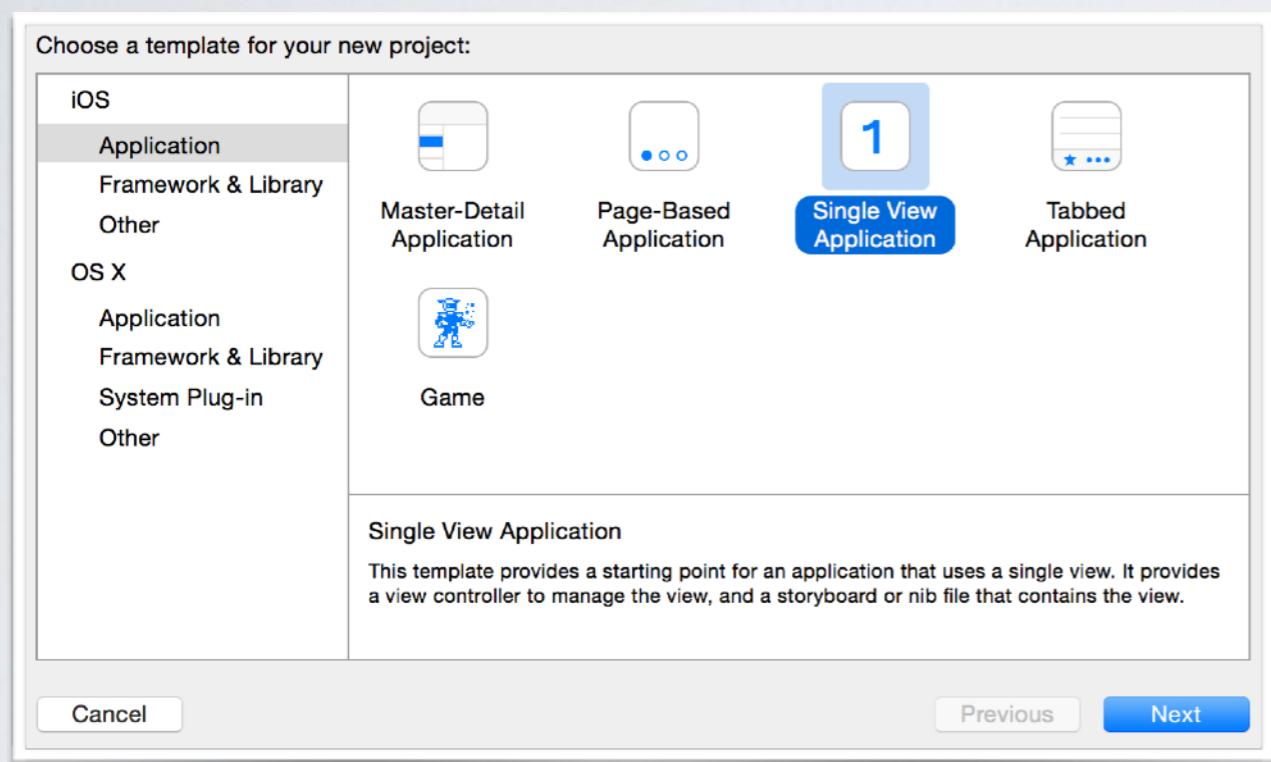


# ANOTHER QUICK DEMO - [COLOR ME]



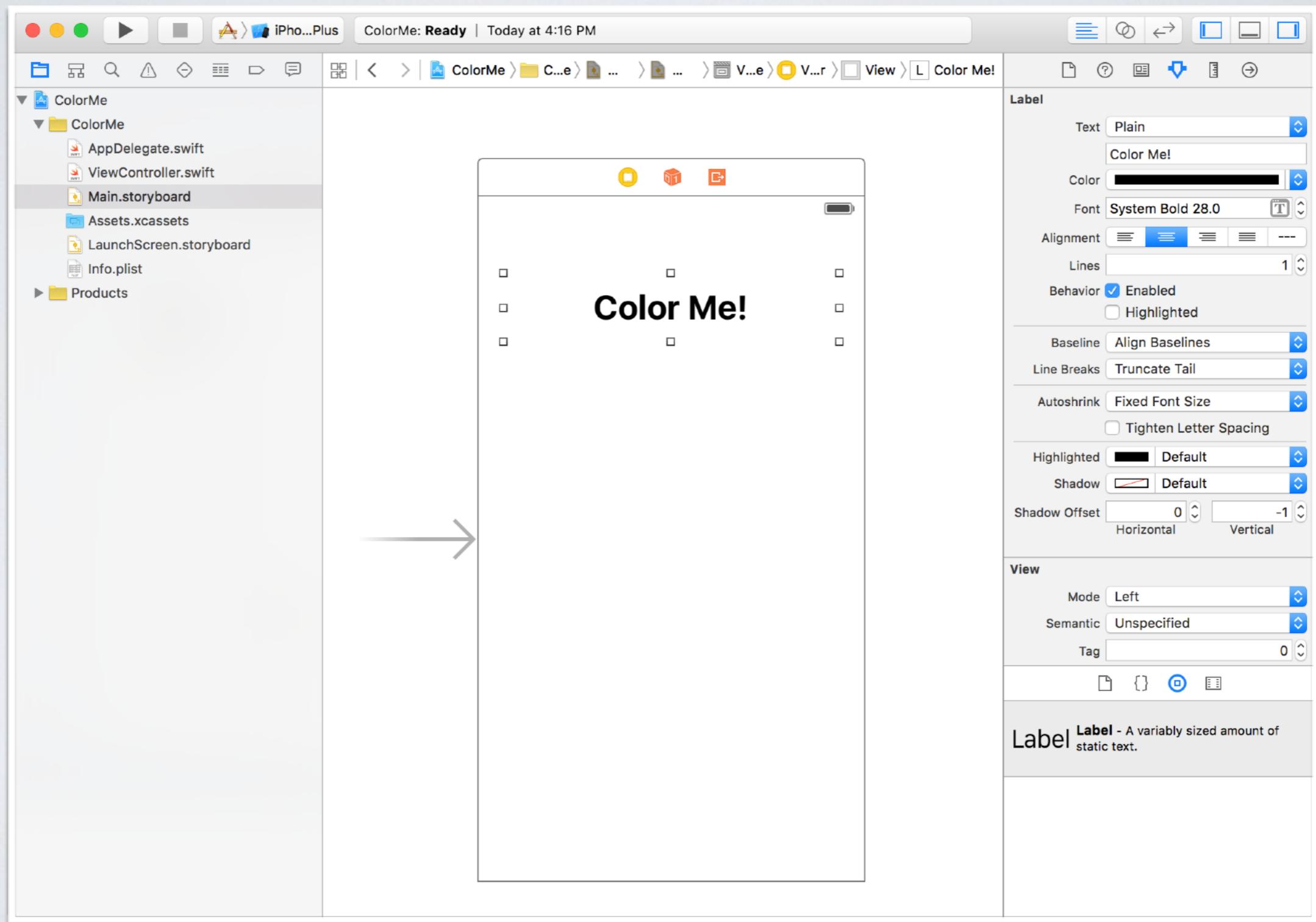
GroundSpeed™  
rapid web + mobile software

# CREATE A SINGLEVIEW APPLICATION

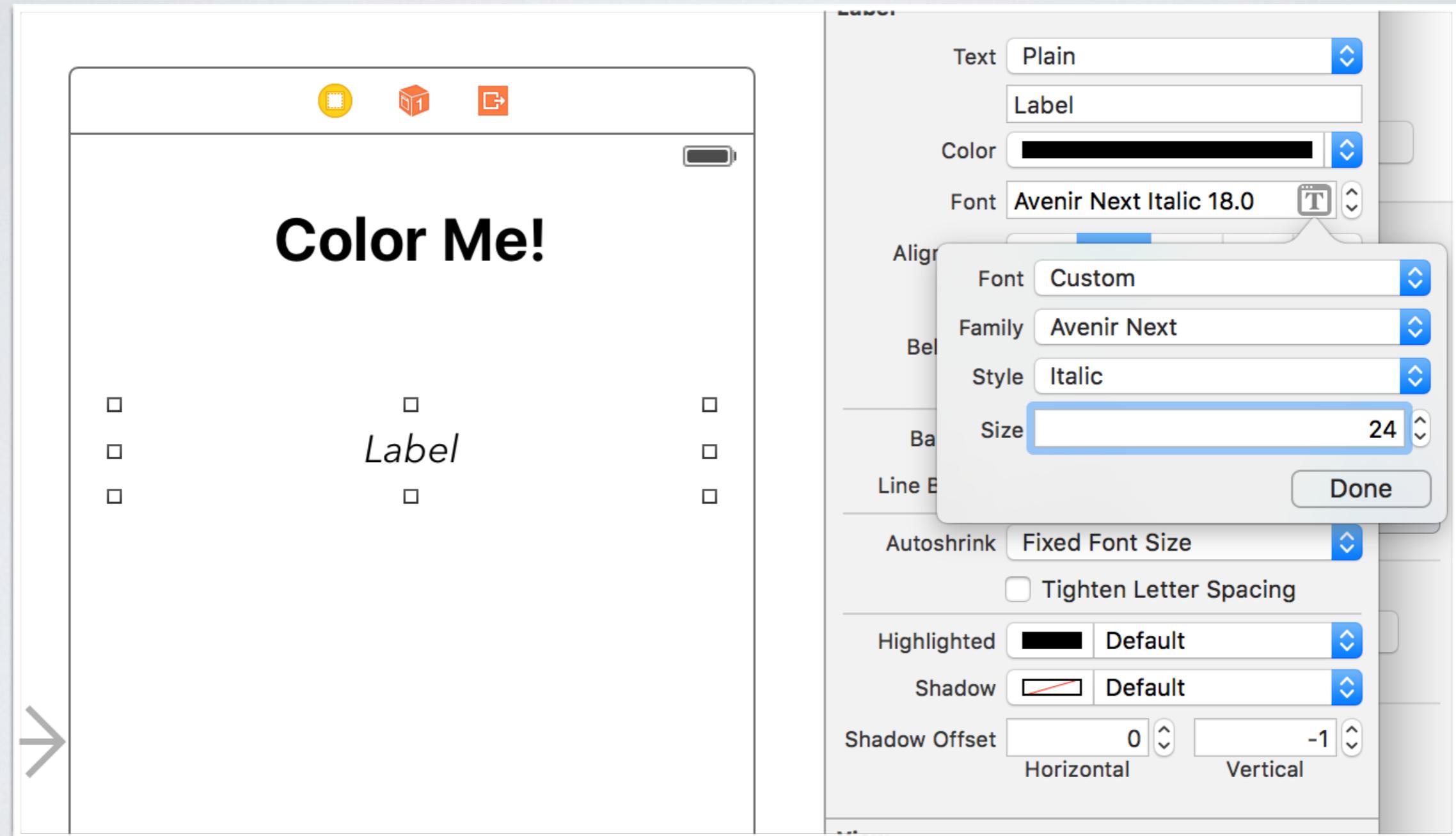


GroundSpeed™  
rapid web + mobile software

# DRAG A LABEL TO THE STORYBOARD



# CREATE A BLANK LABEL FOR OUTPUT



# CREATE TWO BUTTONS

The screenshot shows a mobile application interface with a navigation bar at the top featuring icons for file, project, and export. Below the bar, the title "Color Me!" is displayed in large bold letters. A "Label" is present below the title. Two buttons are visible: a green button labeled "Color Me Green" and a red button labeled "Color Me Red". A large gray arrow points from the red button towards the right side of the screen.

**Button**

Type: System

State Config: Default

Title: Plain

Color Me Red

Font: Avenir Next Regular 24...

Text Color: White Color

Shadow Color: Default

Image: Default Image

Background: Default Background Image

Shadow Offset: 0 0

Width: 0 Height: 0

Reverses On Highlight

Shows Touch On Highlight

Highlighted Adjusts Image

Disabled Adjusts Image

Line Break: Truncate Middle

Edge: Content

Inset: 0 0

Left: 0 Top: 0

Bottom: 0 Right: 0

Button - Intercepts touch events and sends an action message to a target object when it's tapped.

Button

  
GroundSpeed™  
rapid web + mobile software

# ADD OUTLETS TO CONTROLLER



The screenshot shows the Xcode interface with the project navigation bar on the left and the code editor on the right. The project is named 'ColorMe' and contains two targets for iOS SDK 8.1. The 'ViewController.swift' file is selected in the project tree, indicated by a dark grey background.

```
1 //ViewController.swift
2 // ColorMe
3 //
4 //
5 // Created by Don Miller on 12/30/14.
6 // Copyright (c) 2014 GroundSpeed. All rights reserved.
7 //
8
9 import UIKit
10
11 class ViewController: UIViewController {
12
13     @IBOutlet var lblMessage: UILabel!
14     @IBOutlet var btnGreen: UIButton!
15     @IBOutlet var btnRed: UIButton!
16
17     override func viewDidLoad() {
18         super.viewDidLoad()
19         // Do any additional setup after loading the view, typically from a nib.
20     }
21
22     override func didReceiveMemoryWarning() {
23         super.didReceiveMemoryWarning()
24         // Dispose of any resources that can be recreated.
25     }
26
27
28 }
```



# ADD ACTIONS TO CONTROLLER AND UPDATE VIEWDIDLLOAD



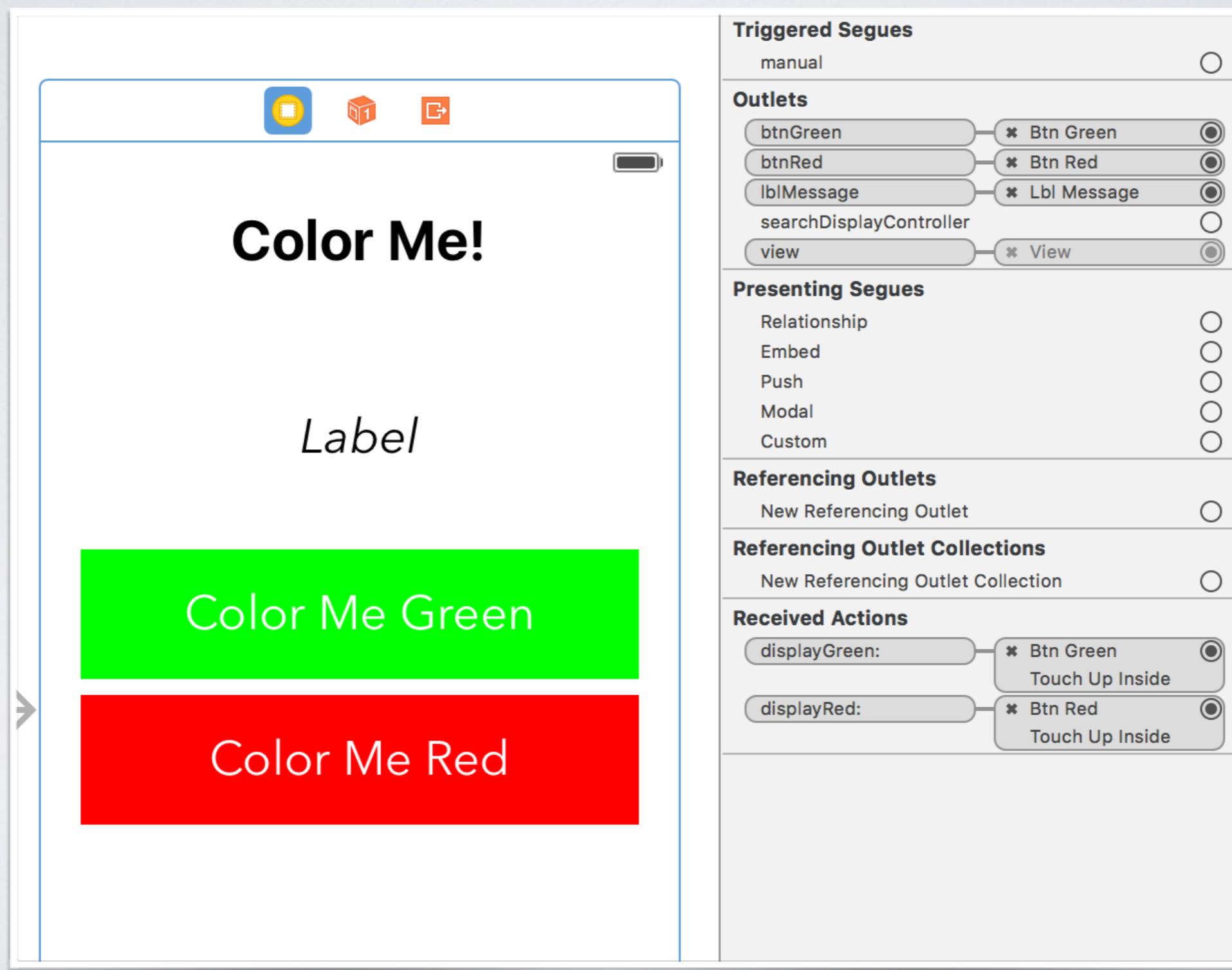
The screenshot shows the Xcode interface with the project 'ColorMe' selected. The left sidebar shows the project structure with 'ViewController.swift' highlighted. The main editor area displays the following Swift code:

```
17 override func viewDidLoad() {
18     super.viewDidLoad()
19     // Do any additional setup after loading the view, typically from a nib.
20     lblMessage.text = ""
21 }
22
23 override func didReceiveMemoryWarning() {
24     super.didReceiveMemoryWarning()
25     // Dispose of any resources that can be recreated.
26 }
27
28 @IBAction func displayGreen(sender: AnyObject) {
29     lblMessage.text = "You wanted green"
30     self.view.backgroundColor = UIColor.greenColor()
31 }
32
33 @IBAction func displayRed(sender: AnyObject) {
34     lblMessage.text = "You wanted red"
35     self.view.backgroundColor = UIColor.redColor()
36 }
37 }
```

The code defines two IBAction methods: 'displayGreen' and 'displayRed'. Each method changes the text of a label and the background color of the view.



# CONNECTING THE OUTLETS AND ACTIONS



# Lab #2 -

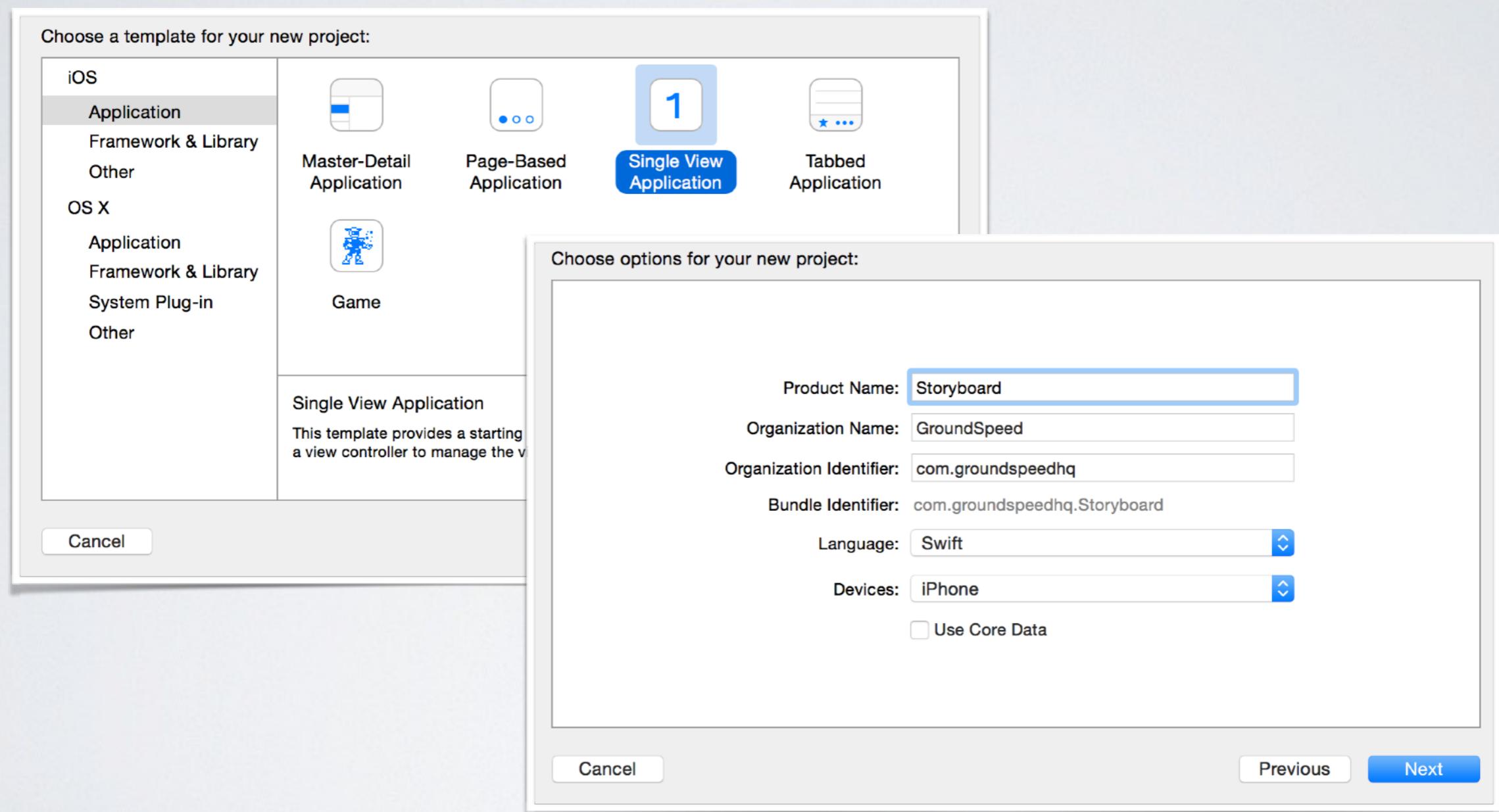
# Create Color Me

# Storyboard App



# STORYBOARD DEMO

## STEP 1: CREATE A NEW SINGLEVIEW APPLICATION



# STEP 2: OBSERVE THE SUMMARY PAGE

▼ **Identity**

Bundle Identifier com.groundspeedhq.Storyboard

Version 1.0

Build 1

Team None

---

▼ **Deployment Info**

Deployment Target 8.1

Devices iPhone

Main Interface Main

Device Orientation  Portrait  
 Upside Down  
 Landscape Left  
 Landscape Right

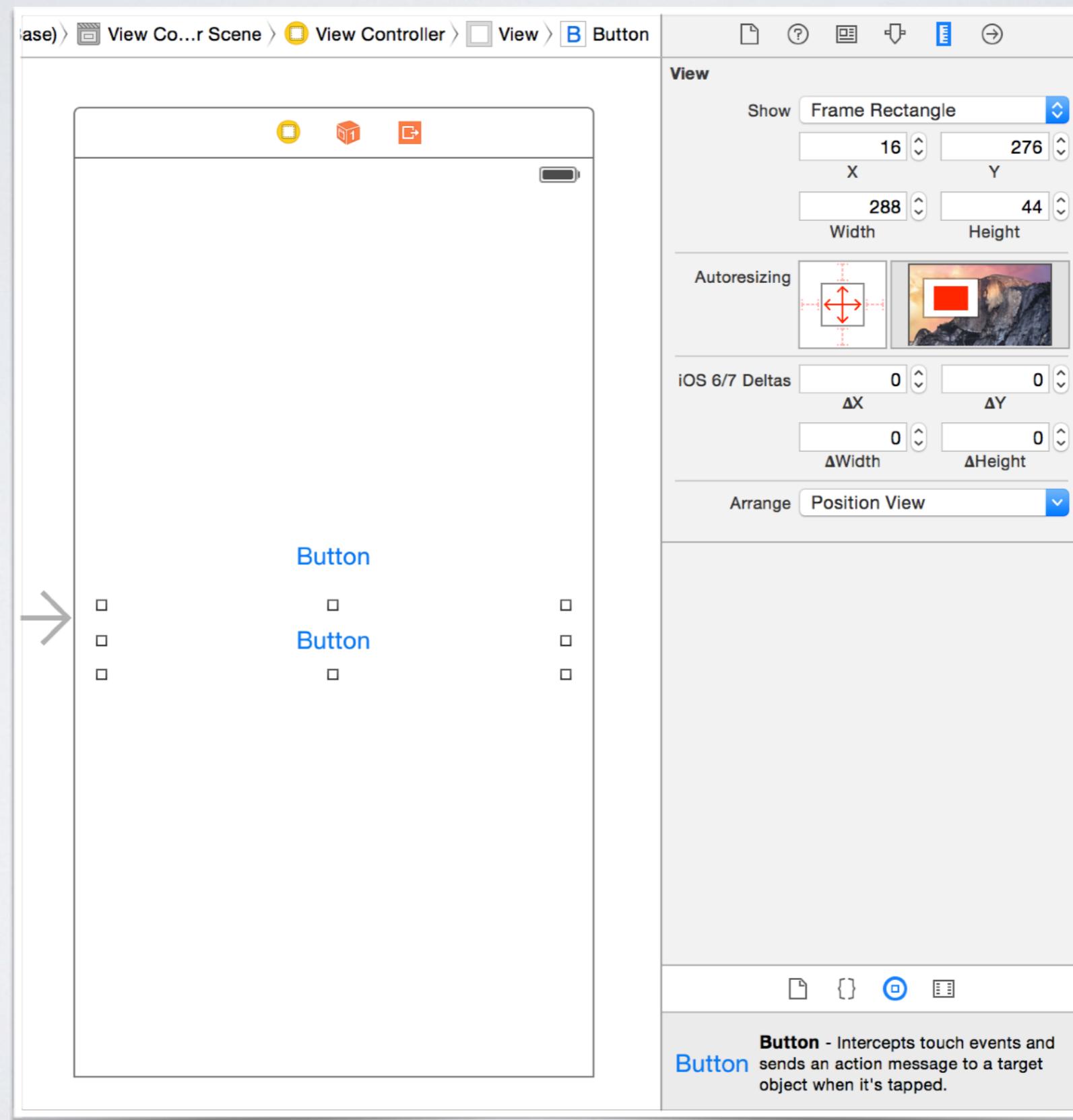
Status Bar Style Default

Hide status bar



# STEP 3: ADD 2 BUTTONS TO THE VIEW CONTROLLER

# STEP 4: LET'S MAKE THEM THE SAME WIDTH IN SIZE INSPECTOR



# STEP 5: CHANGE THE BUTTON LABELS TEXT TO RED AND GREEN

The image shows the Xcode interface with a storyboard on the left and the "Button" properties inspector on the right.

**Storyboard Preview:** Shows a single view controller with two buttons. The top button is labeled "Red Push" in red text. The bottom button is labeled "Green Modal" in green text.

**Properties Inspector (Button Properties):**

- Type: System
- State Config: Default
- Title: Plain
- Label Text: Green Modal
- Font: System 15.0
- Text Color: Green
- Shadow Color: Default
- Image: Default Image
- Background: Default Background Image
- Shadow Offset: 0 (Width), 0 (Height)
- Checkboxes:
  - Reverses On Highlight
  - Shows Touch On Highlight
  - Highlighted Adjusts Image
  - Disabled Adjusts Image
- Line Break: Truncate Middle
- Edge: Content
- Inset:
  - Top: 0
  - Bottom: 0
  - Left: 0
  - Right: 0

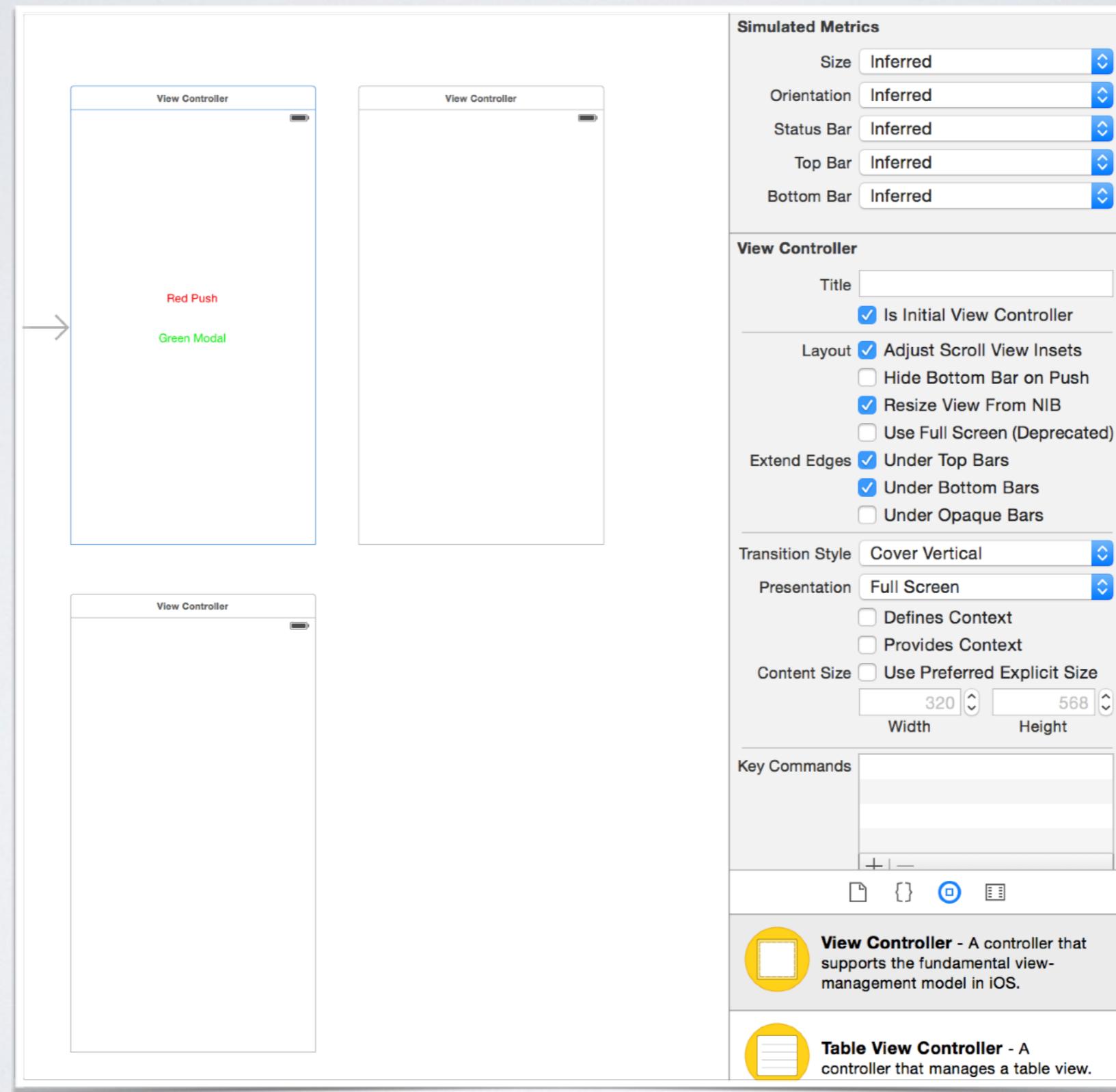
**Control Section:**

- Alignment:
  - Horizontal
  - Vertical
- Buttons:
  - File
  - Document
  - Print
  - Help

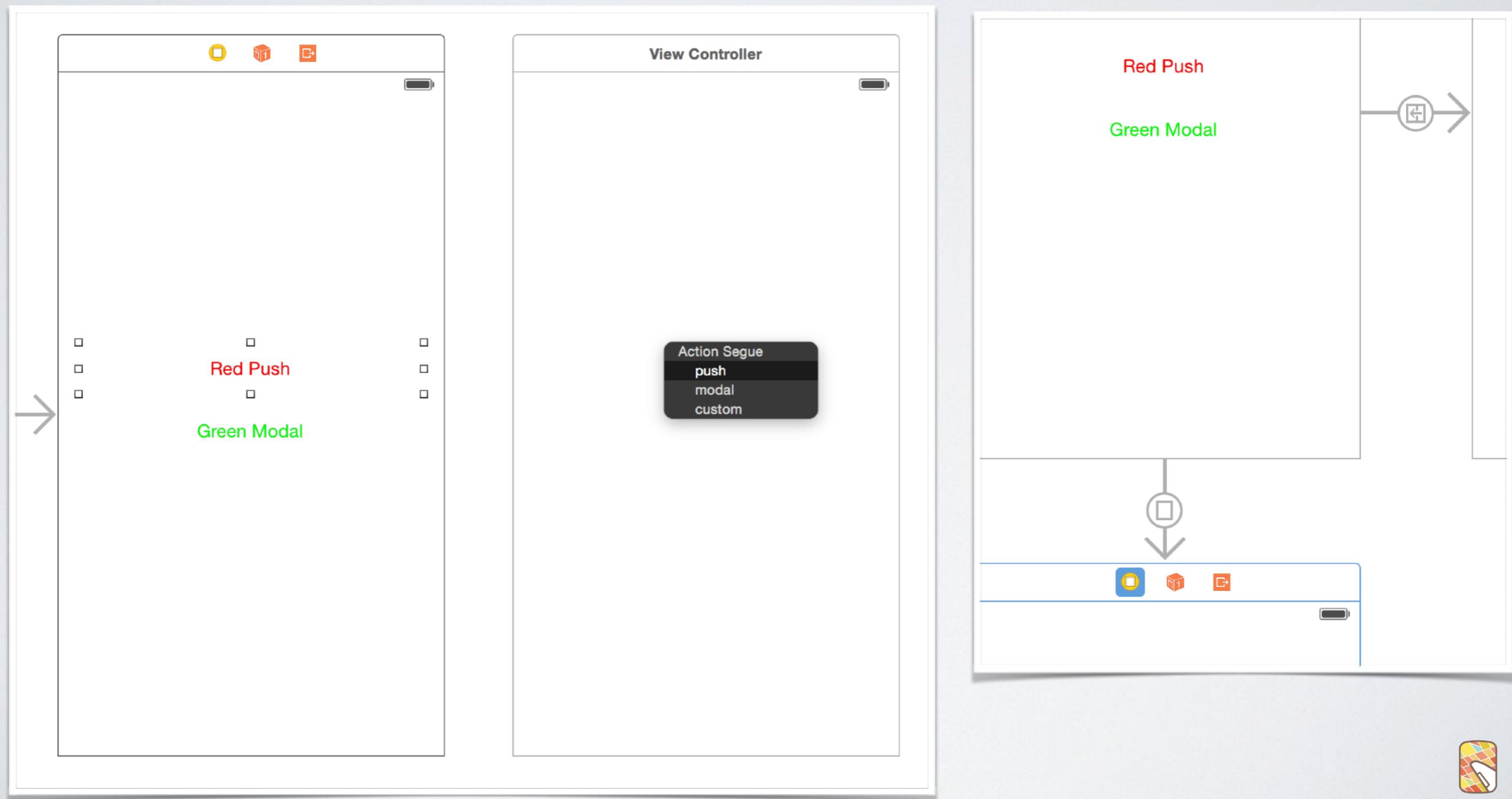
**Description:** Button - Intercepts touch events and sends an action message to a target object when it's tapped.



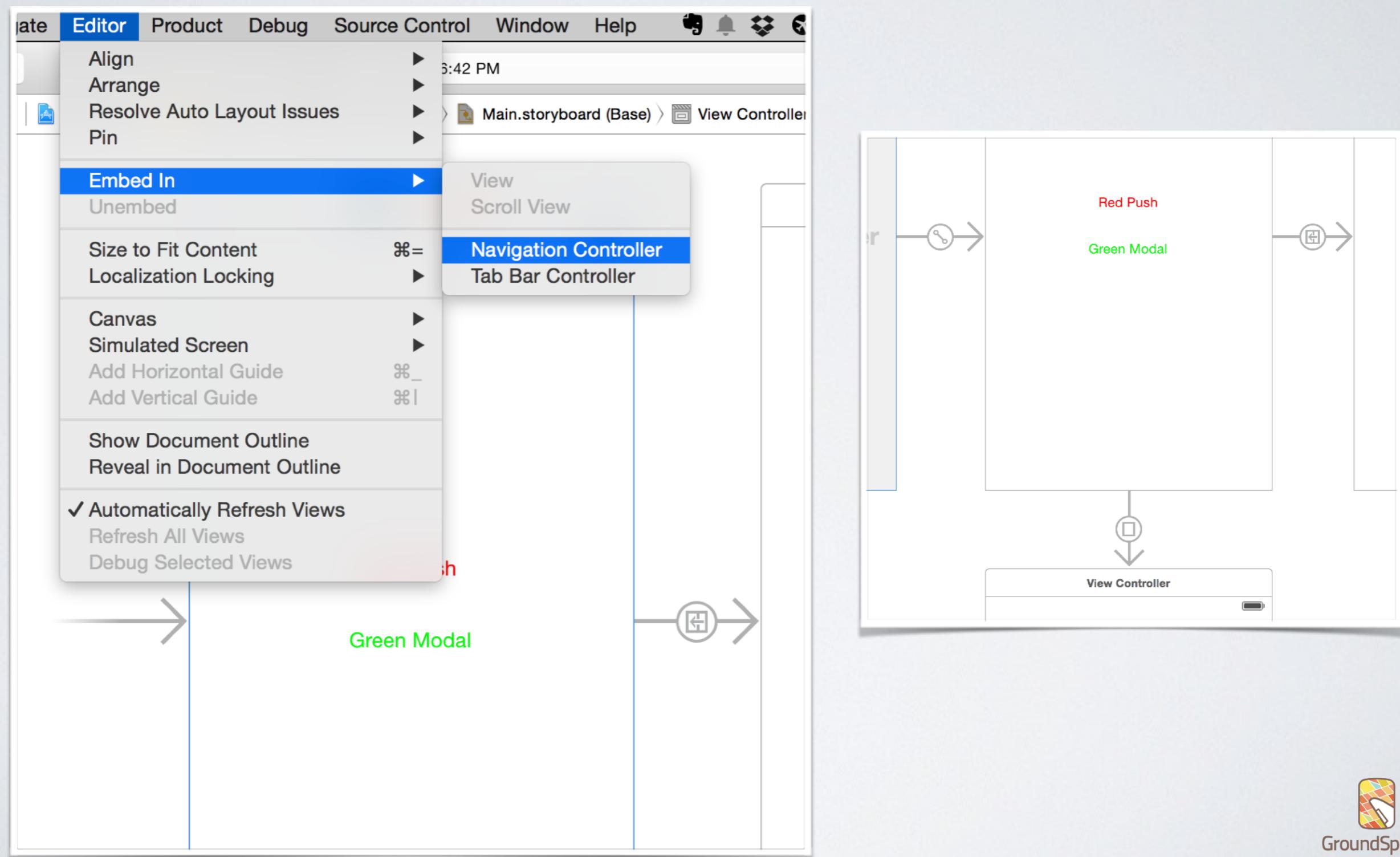
# STEP 6: ADD TWO VIEW CONTROLLERS TO THE STORYBOARD BY DRAGGING THEM OVER



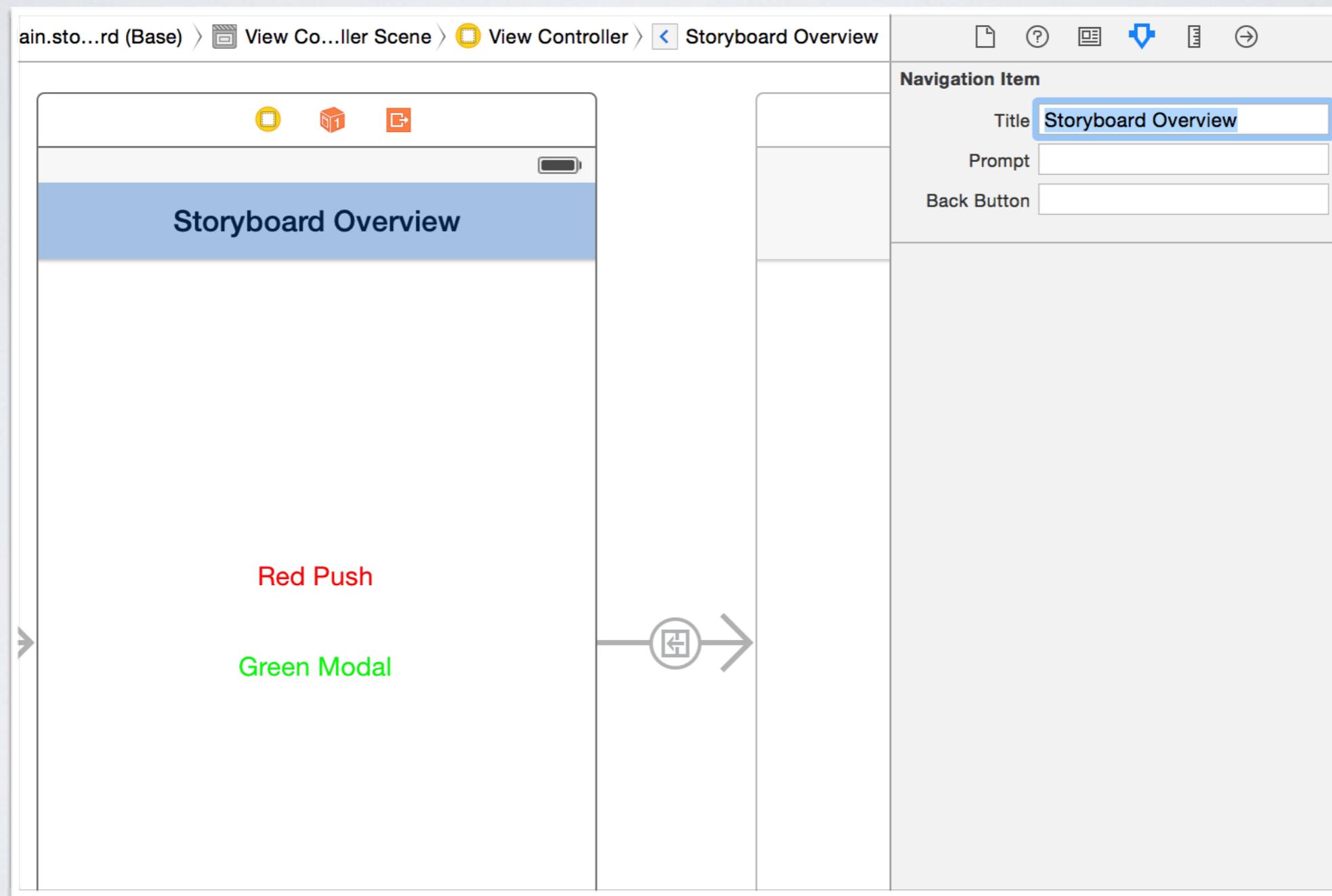
# STEP 7: CONNECT THE BUTTONS TO THE NEW VIEW CONTROLLERS BY HOLDING DOWN THE [CONTROL] KEY AND DRAGGING FROM THE BUTTON TO THE NEW VIEW CONTROLLER



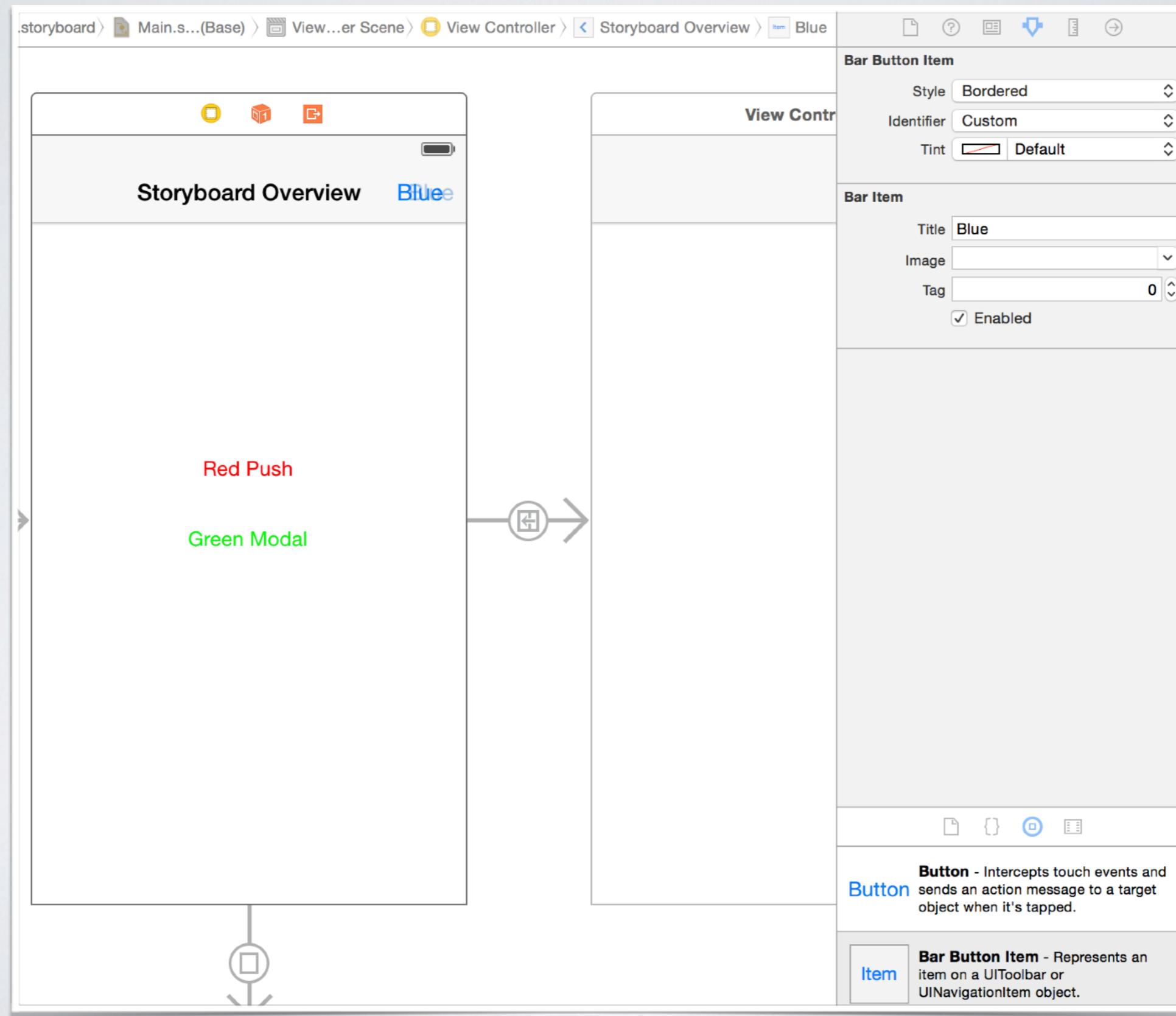
# STEP 8: EMBED A NAVIGATION CONTROLLER ON YOUR PRIMARY VIEW



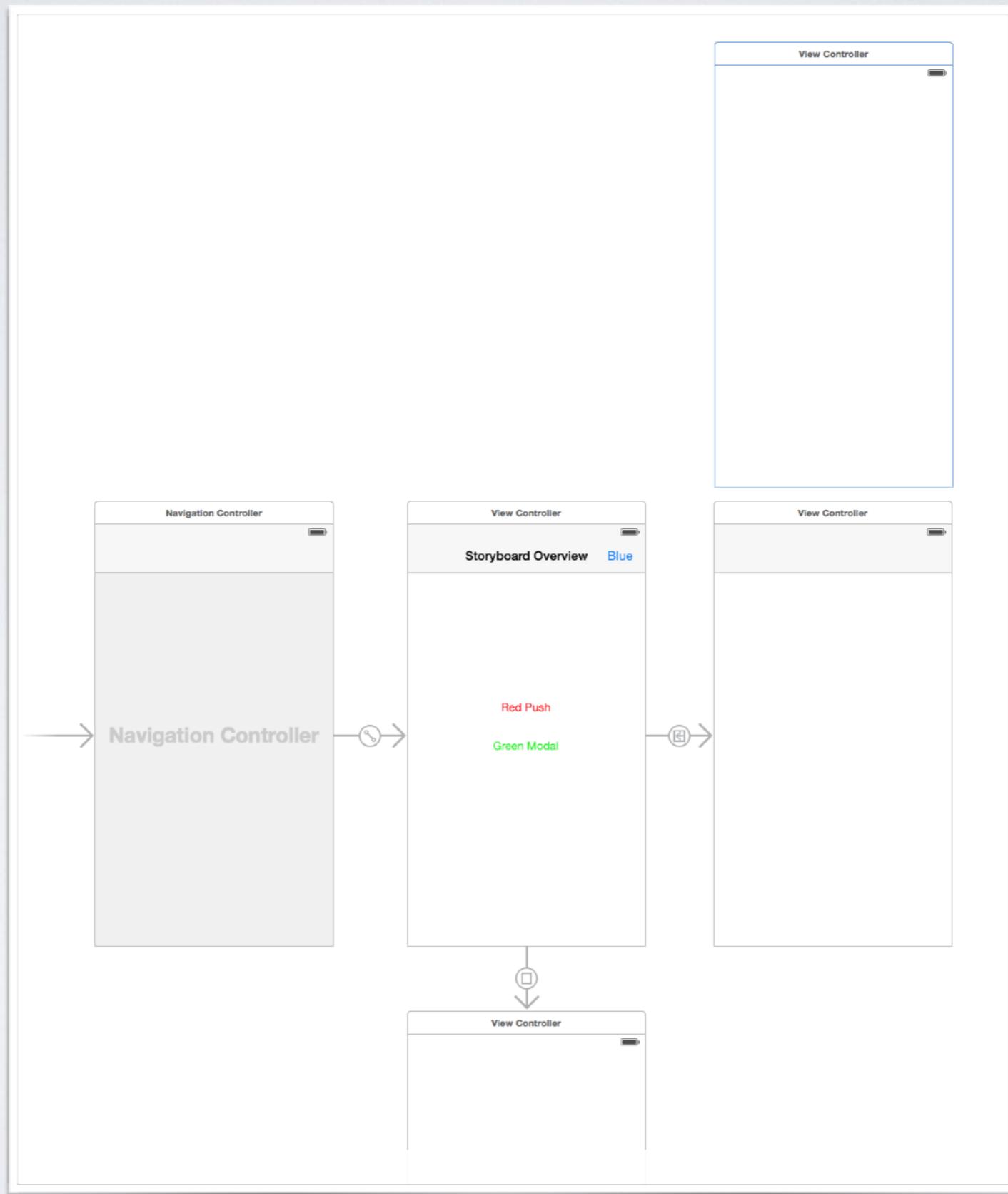
# STEP 9: CHANGE THE TITLE OF THE NAVIGATION BAR



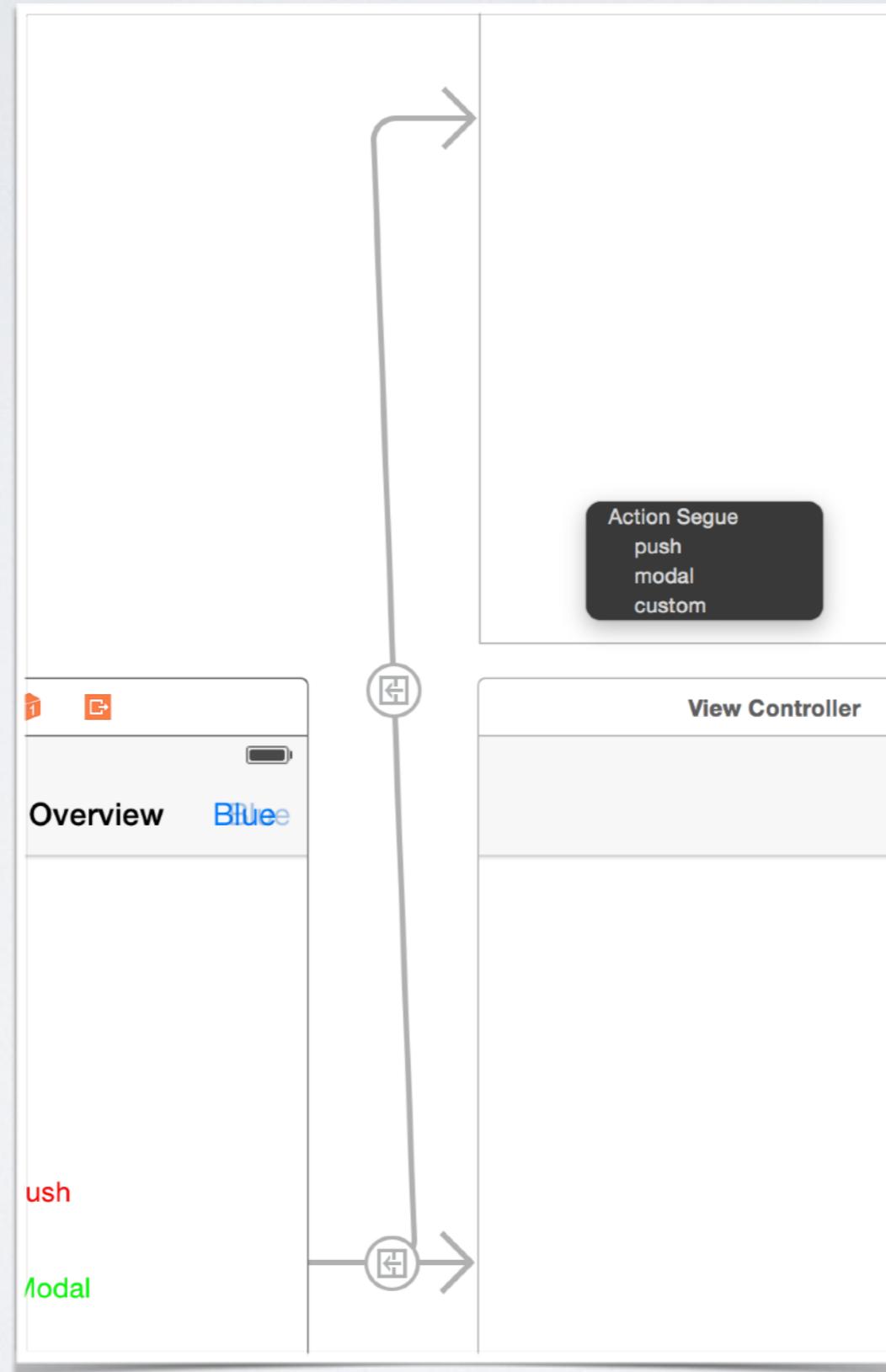
# STEP 10: ADD A BAR BUTTON ITEM TO THE NAVIGATION BAR AND CHANGE THE TITLE TO BLUE



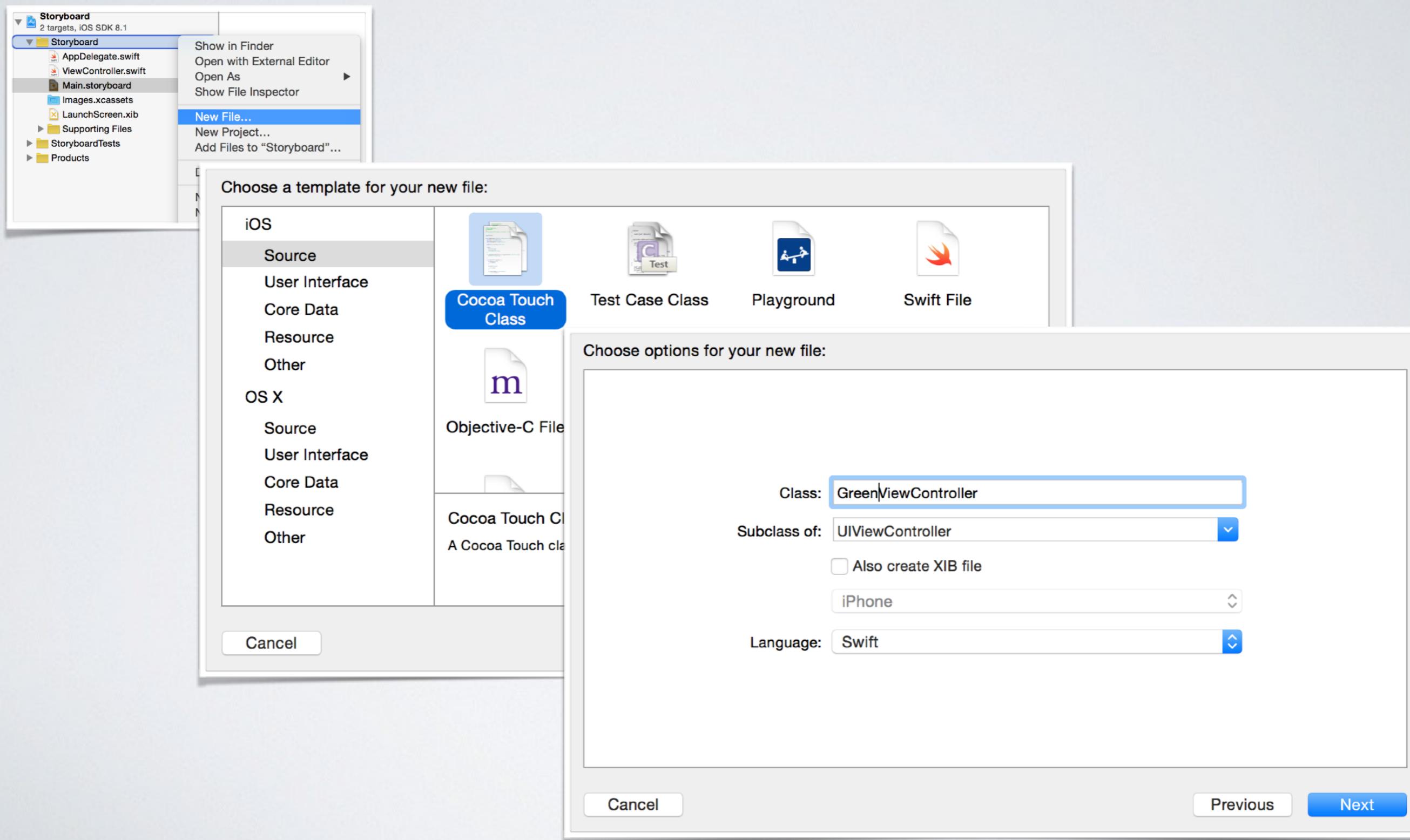
# STEP III: ADD OUR THIRD AND FINAL VIEW CONTROLLER TO THE STORYBOARD



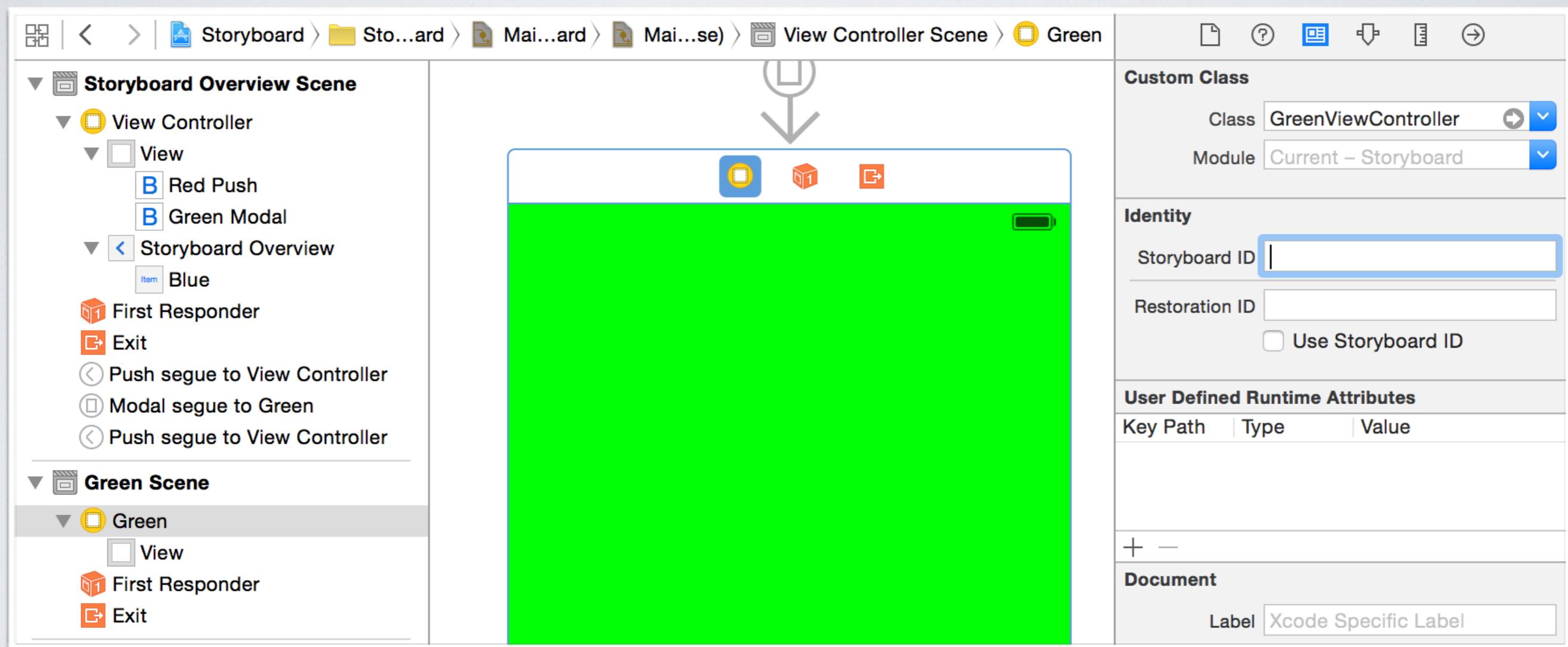
STEP 12: CONNECT OUR BAR BUTTON ITEM TO THE NEW VIEW CONTROLLER AND SELECT PUSH. REMEMBER TO HOLD DOWN THE [CONTROL] BUTTON WHILE CLICKING AND DRAGGING



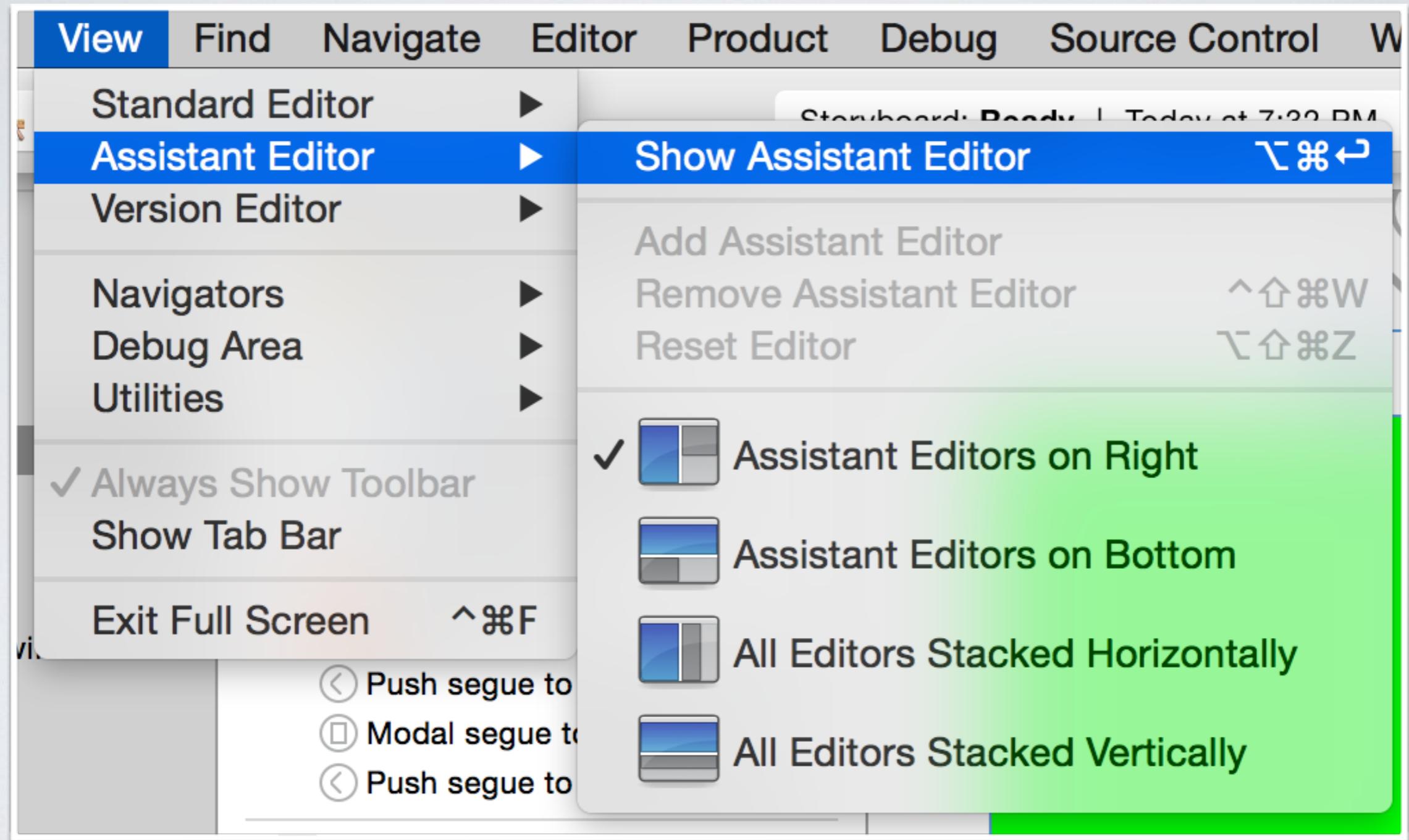
# STEP 13: NOW WE NEED TO DO A LITTLE PROGRAMMING. ADD A CONTROLLER OBJECT AND NAME IT “GREENVIEWCONTROLLER” AS SHOWN BELOW



# STEP 14: ASSIGN YOUR VIEW IN THE STORYBOARD TO THE NEW OBJECTVIEW CONTROLLER YOU JUST CREATED



# STEP 15: SHOW THE ASSISTANT EDITOR TO HAVE OUR CODE GENERATED FOR US BY CLICKING



# STEP 16: ADD A BUTTON TO THE GREEN VIEW

The screenshot shows the Xcode interface with the storyboard and code editor.

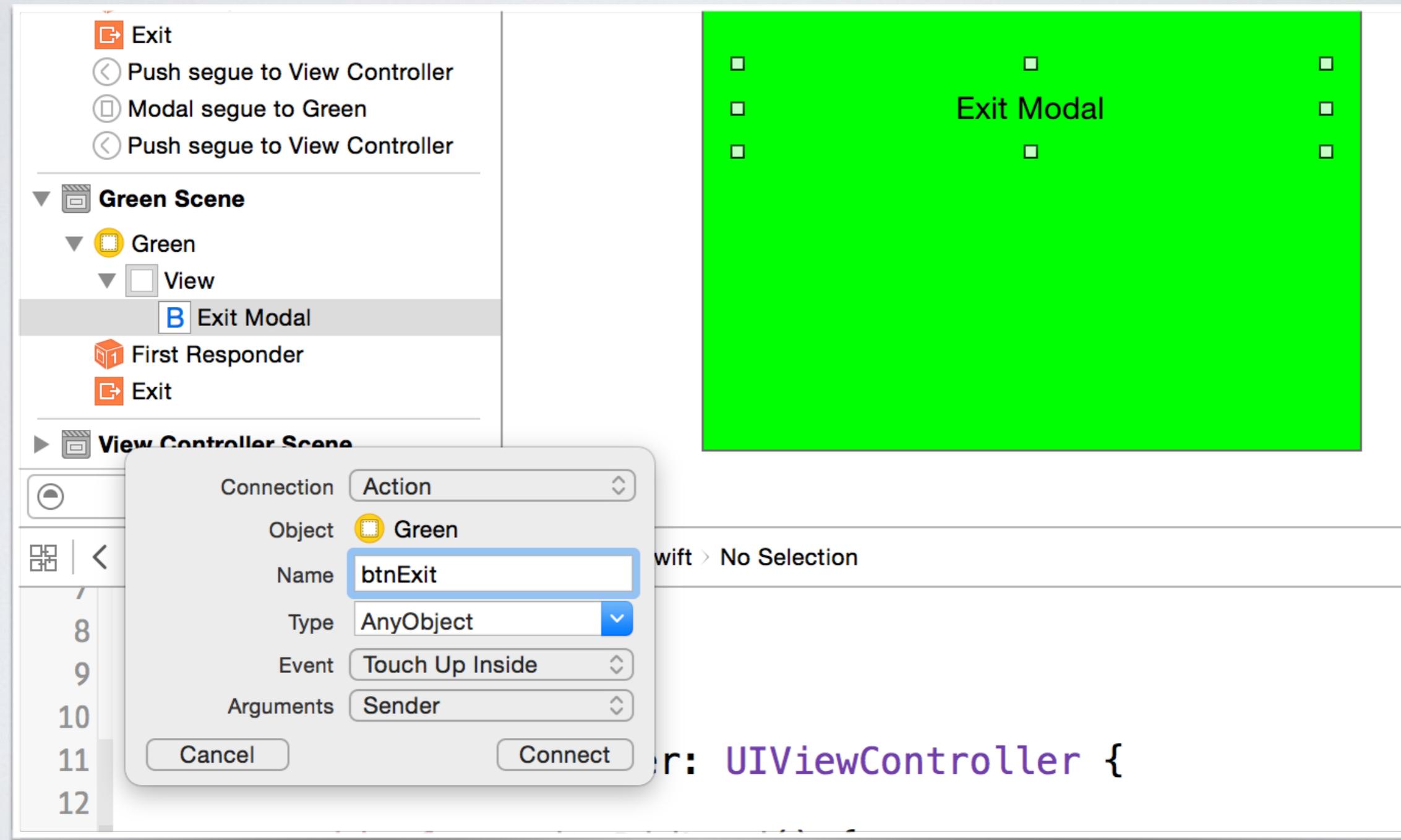
**Storyboard:** The storyboard contains a single green view controller. An "Exit Modal" segue is selected, highlighted with a blue border. The storyboard outline shows the following structure:

- Item Blue
- First Responder
- Exit
- Push segue to View Controller
- Modal segue to Green
- Push segue to View Controller
- Green Scene
- Green
- View
- B Exit Modal (selected)
- First Responder
- Exit

**Code Editor:** The code editor displays the `GreenViewController.swift` file:1 //  
2 // GreenViewController.swift  
3 // Storyboard  
4 //  
5 // Created by Don Miller on 12/30/14.  
6 // Copyright (c) 2014 GroundSpeed. All rights reserved.  
7 //  
8  
9 import UIKit  
10  
11 class GreenViewController: UIViewController {  
12  
13 }



STEP 17: HOLD DOWN THE CONTROL WHEN CLICKING FROM THE BUTTON TO YOU HEADER FILE AND DRAG IT UNDER THE INTERFACE LINE. CHANGE THE CONNECTION TO ACTION AND THE NAME TO BTNEXTIT

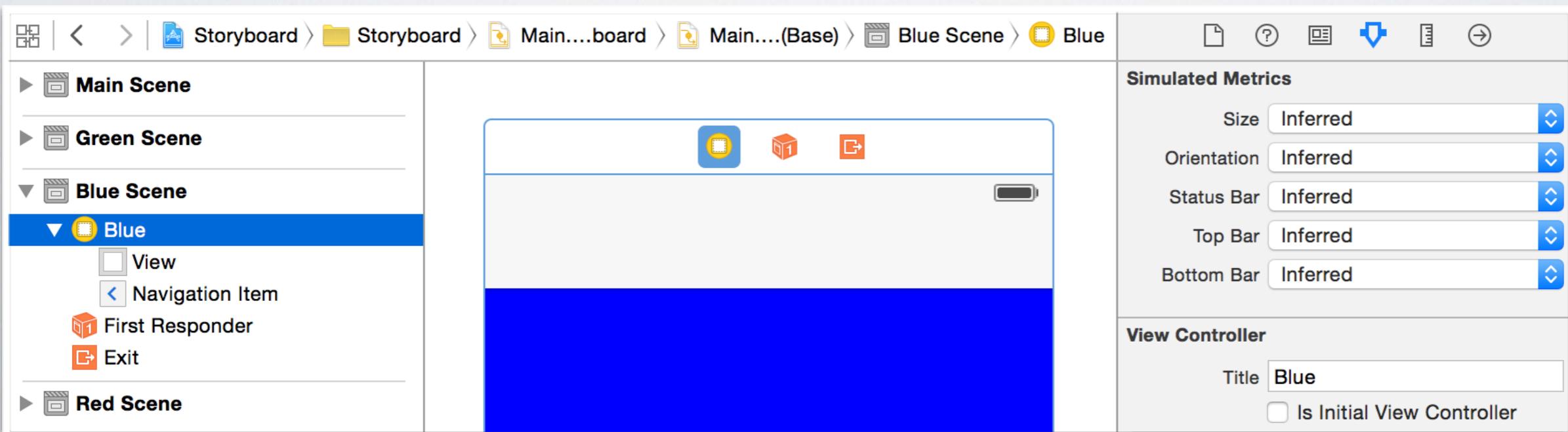
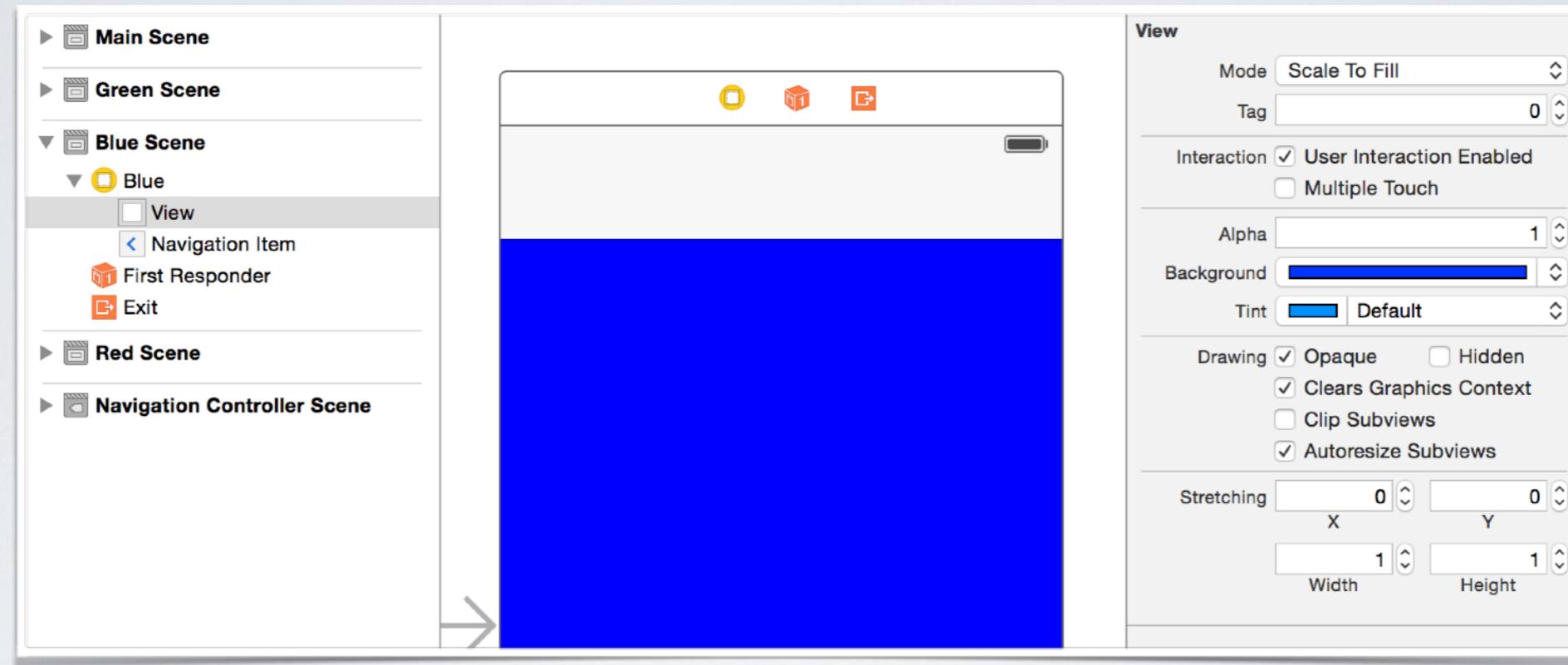


# STEP 18: ADD THE FOLLOWING LINE OF CODE IN THE BTNEXTIT METHOD TO DISMISS THE MODAL WINDOW

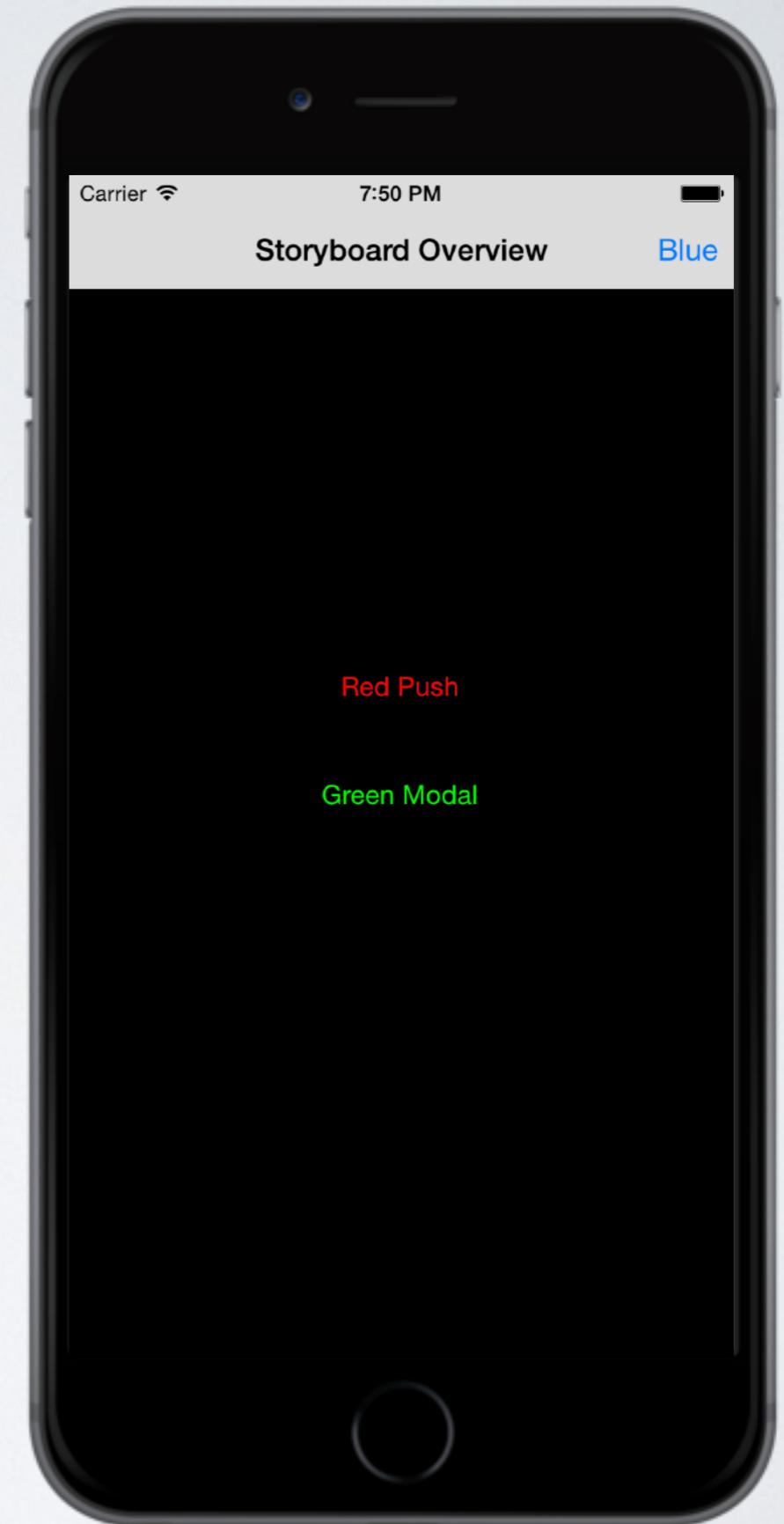
```
9 import UIKit  
10  
11 class GreenViewController: UIViewController {  
12  
13     override func viewDidLoad() {  
14         super.viewDidLoad()  
15  
16         // Do any additional setup after loading the view.  
17     }  
18  
19     override func didReceiveMemoryWarning() {  
20         super.didReceiveMemoryWarning()  
21         // Dispose of any resources that can be recreated.  
22     }  
23  
24     @IBAction func btnExit(sender: AnyObject) {  
25         self.dismissViewControllerAnimated(true, completion: nil)  
26     }  
27 }
```



# STEP 19: FINAL TOUCHES. CHANGE THE VIEW CONTROLLER TITLES AND BACKGROUNDS TO REFLECT THE COLOR DESCRIBED IN THE CALLING BUTTON



# STEP 20: RUN IT!



GroundSpeed™  
rapid web + mobile software

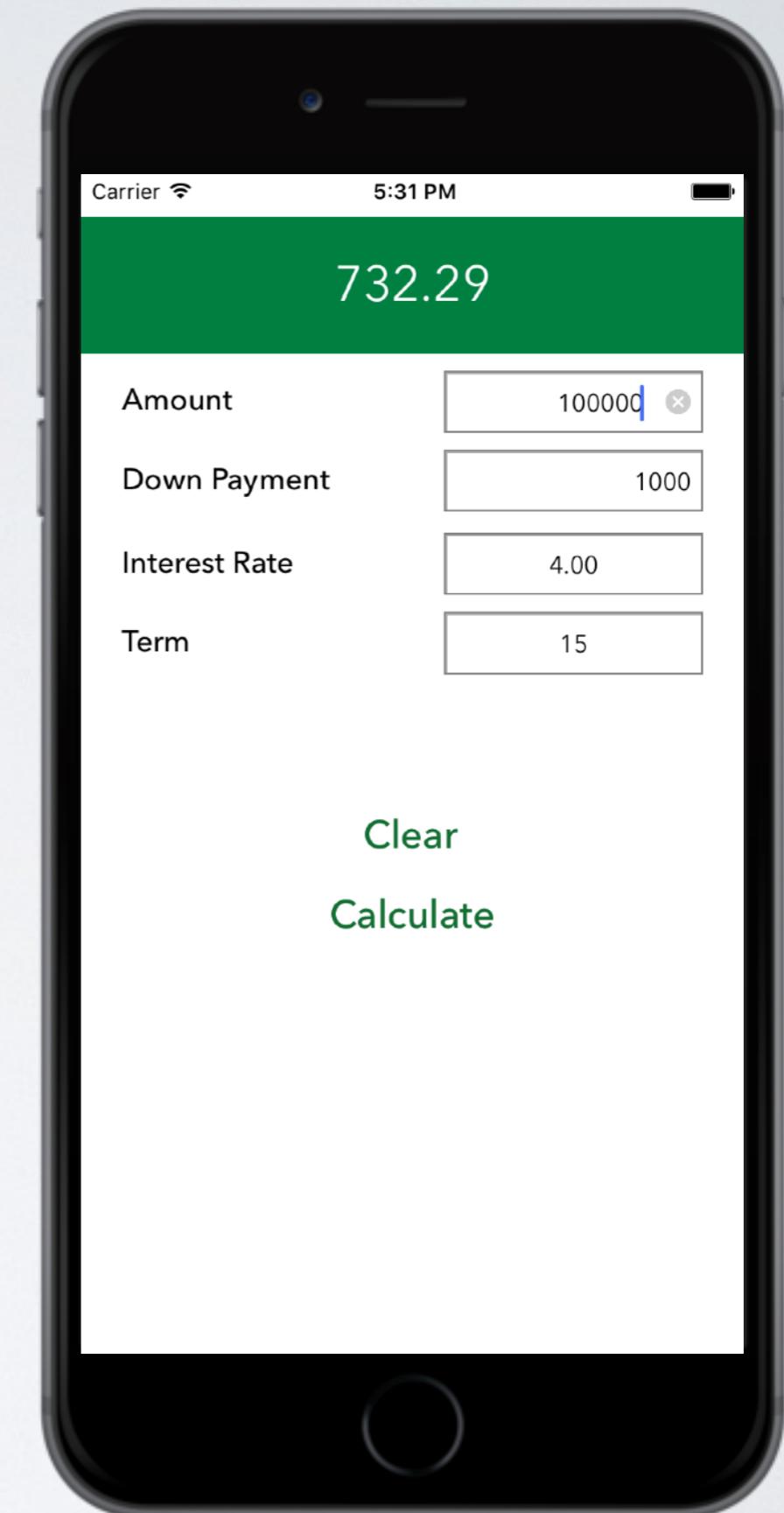
# Lab #3 - Create Amortize Payment App



# CREATE AN APP TO CALCULATE AN AMORTIZATION PAYMENT

1. Create 5 Labels
2. Create 4 Text Boxes
3. Create 2 Buttons
  - A. Clear clears all text boxes and total label
  - B. Calculate creates payment amount on top label

$$P = A \cdot \frac{1 - \left(\frac{1}{1+r}\right)^n}{r}$$



```

func getMonthlyPayment() {
    // A = payment Amount per period
    // P = initial Principal (loan amount)
    // r = interest rate per period
    // n = total number of payments or periods

```

$$P = A \cdot \frac{1 - \left(\frac{1}{1+r}\right)^n}{r}$$

```

        let principal : Float = Float(txtAmount.text!)!
        - Float(txtDownPayment.text!)
        let payments = Float(txtTerm.text!)!*12
        let rate = Float(txtInterestRate.text!)!/12/100
        let amount = calculatePMTWithRatePerPeriod(rate, numberOfPayments:
payments, loanAmount: principal, futureValue: 0, type: 0)

        lblPayment.text = String(format: "%.02f", amount)
    }

```

```

func calculatePMTWithRatePerPeriod (ratePerPeriod: Float, numberOfPayments:
Float, loanAmount: Float, futureValue: Float, type: Float) -> Float {

    var q : Float
    q = pow(1 + ratePerPeriod, numberOfPayments)

    let returnValue = (ratePerPeriod * (futureValue + (q * loanAmount))) /
((-1 + q) * (1 + ratePerPeriod * (type)))

    return returnValue
}

```



**Don Miller**

**don@GroundSpeedHQ.com**

**@donmiller**

**<http://github.com/donmiller>**

**<http://github.com/groundspeed>**

