The About Screen | iOS Apprentice Part 13

Ray Wenderlich para mim 16 de ago

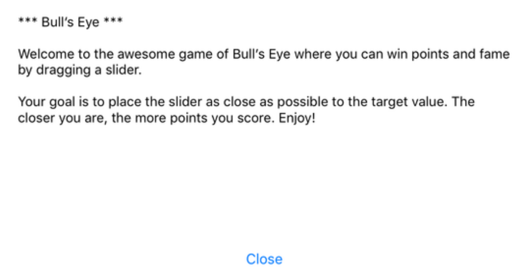
José, welcome back to the raywenderlich.com **iOS Apprentice Email Course**!

Your game looks awesome and your to-do list is done. So, does this mean that you are done with **Bull's Eye**?

Not so fast :] Remember the ⓘ button on the game screen? Try tapping it. Does it do anything? No?

Ooops! Looks as if we forgot to add any functionality to that button :] It's time to rectify that - let's add an "about" screen to the game which shows some information about the game and have it display when the user taps on the ⓘ button.

Initially, the screen will look something like this (but we'll prettify it soon enough):



*** Bull's Eye ***

Welcome to the awesome game of Bull's Eye where you can win points and fame by dragging a slider.

Your goal is to place the slider as close as possible to the target value. The closer you are, the more points you score. Enjoy!

Close

This new screen contains a **text view** with the gameplay rules and a button to close the screen.

Most apps have more than one screen, even very simple games. So, this is as good a time as any to learn how to add additional screens to your apps.

I have pointed it out a few times already: each screen in your app will have its own view controller. If you think "screen", think "view controller".

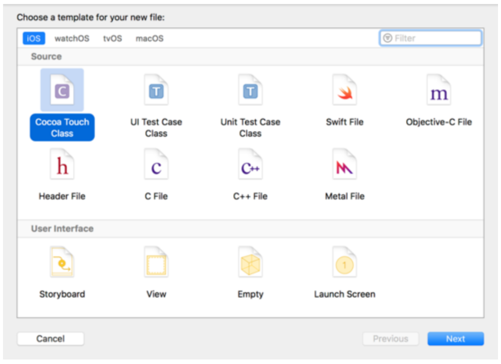
Xcode automatically created the main `ViewController` object for you, but you'll have to create the view controller for the About screen yourself.

> **Note:** Prefer learning via video? You might like to check out videos #34-46 and #41-42 of the free video version of this course, which correspond to this email.

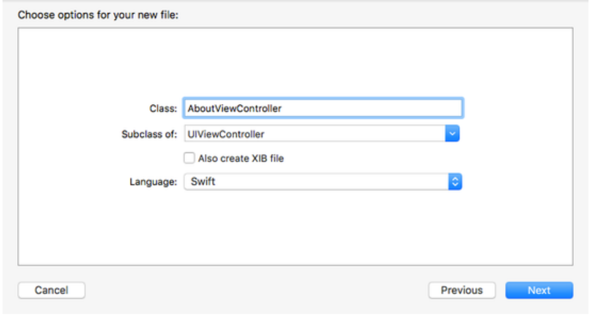
Add a new view controller

To get started, open up the Bull's Eye project where you left it off last time (or download the starter project from the corresponding forum discussion thread).

➤ Go to Xcode's **File** menu and choose **New → File...** In the window that pops up, choose the **Cocoa Touch Class** template (if you don't see it then make sure **iOS** is selected at the top).



Click **Next**. Xcode gives you some options to fill out:

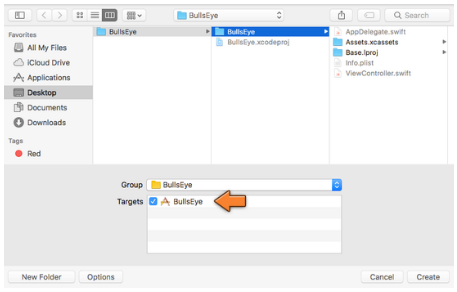

Choose options for your new file:

Class: AboutViewController
Subclass of: UIViewController
☐ Also create XIB file
Language: Swift

Choose the following:

- Class: **AboutViewController**
- Subclass of: **UIViewController**
- Also create XIB file: Leave this box unchecked.
- Language: **Swift**

Click **Next**. Xcode will ask you where to save this new view controller.



Group: BullsEye
Targets: ✓ BullsEye

➤ Choose the **BullsEye** folder (this folder should already be selected).

Also make sure **Group** says **BullsEye** and that there is a checkmark in front of BullsEye in the list of **Targets**. (If you don't see this panel, click the Options button at the bottom of the dialog.)

➤ Click **Create**.

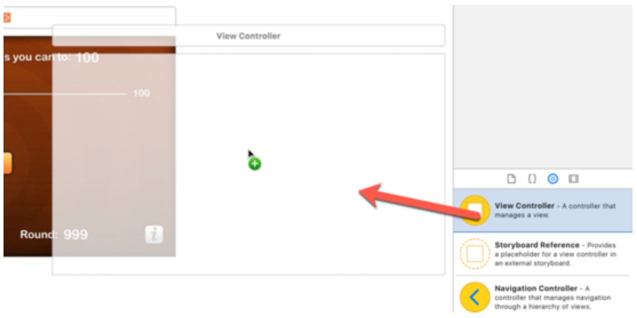
Xcode will create a new file and add it to your project. As you might have guessed, the new file is **AboutViewController.swift**.

Design the view controller in Interface Builder

To design this new view controller, you need to pay a visit to Interface Builder.

➤ Open **Main.storyboard**. There is no scene representing the About view controller in the storyboard yet. So, you'll have to add this first.

➤ From the **Object Library**, choose **View Controller** and drag it on to the canvas, to the right of the main View Controller.

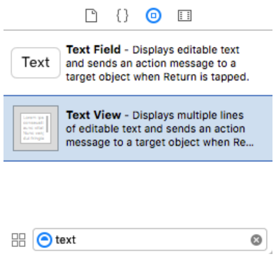

s you can to: 100
100
Round: 999
View Controller
View Controller - A controller that manages a view.
Storyboard Reference - Provides a placeholder for a view controller in an external storyboard.
Navigation Controller - A controller that manages navigation through a hierarchy of views.

This new view controller is totally blank. You may need to rearrange the storyboard so that the two view controllers don't overlap. Interface Builder isn't very tidy about where it puts things.

➤ Drag a new **Button** on to the screen and give it the title **Close**. Put it somewhere in the bottom center of the view (use the blue guidelines to help with positioning).

➤ Drag a **Text View** on to the view and make it cover most of the space above the button.

You can find these components in the Object Library. If you don't feel like scrolling, you can filter the components by typing in the field at the bottom:



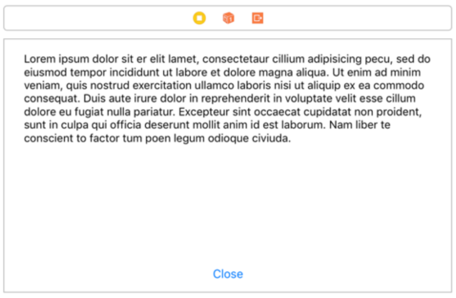
Text

Text Field - Displays editable text and sends an action message to a target object when Return is tapped.

Text View - Displays multiple lines of editable text and sends an action message to a target object when Re...

Note that there is also a Text Field, which is a single-line text component - that's not what you want. You're looking for Text View, which can contain multiple lines of text.

After dragging both the text view and the button on to the canvas, it should look something like this:



Lorem ipsum dolor sit er elit lamet, consectetaur cillium adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Nam liber te conscient to factor tum poen legum odioque civiuda.

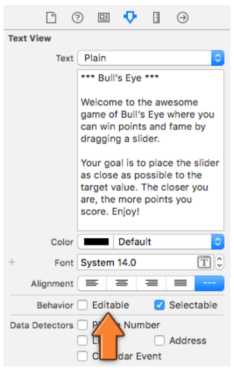
➤ Double-click the text view to make its content is editable. By default, the Text View contains a bunch of Latin placeholder text (also known as "Lorem Ipsum").

Copy-paste this new text into the Text View:

```
*** Bull's Eye ***

Welcome to the awesome game of Bull's Eye where you can win points and fa
me by dragging a slider.

Your goal is to place the slider as close as possible to the target valu
e. The closer you are, the more points you score. Enjoy!
```

You can also paste that text into the Attributes inspector's **Text** property for the text view if you find that easier.

➤ Make sure to uncheck the **Editable** checkbox in the Attribute Inspector. Otherwise, the user can actually type into the text view and you don't want that.



Text View

Text: Plain

*** Bull's Eye ***

Welcome to the awesome game of Bull's Eye where you can win points and fame by dragging a slider.

Your goal is to place the slider as close as possible to the target value. The closer you are, the more points you score. Enjoy!

Color: Default

Font: System 14.0

Behavior: ☐ Editable ☑ Selectable

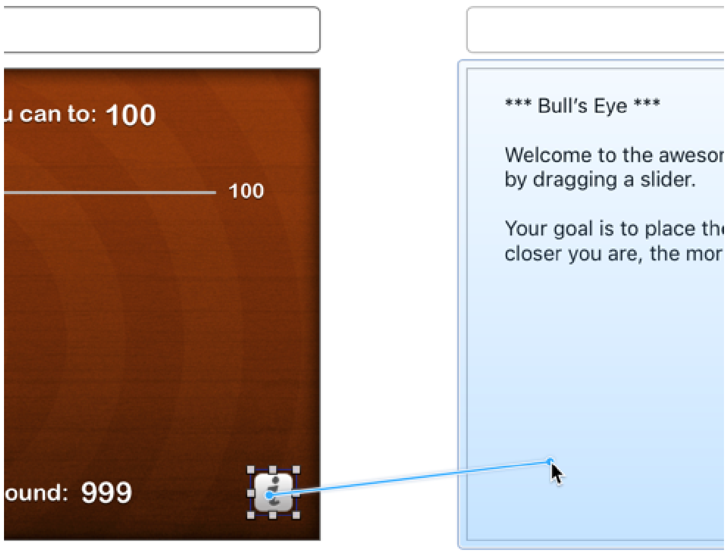
Data Detectors: ☐ Phone Number ☐ Link ☐ Address ☐ Calendar Event

That's the design of the screen done for now.

Show the new view controller

So how do you open this new About screen when the user presses the ⓘ button? Storyboards have a neat trick for this: **segues** (pronounced "seg-way" like the silly scooters). A segue is a transition from one screen to another. They are really easy to add.

➤ Click the ⓘ button in the **View Controller** to select it. Then hold down **Control** and drag over to the **About** screen.

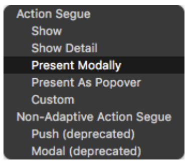


*** Bull's Eye ***

Welcome to the awesor
by dragging a slider.

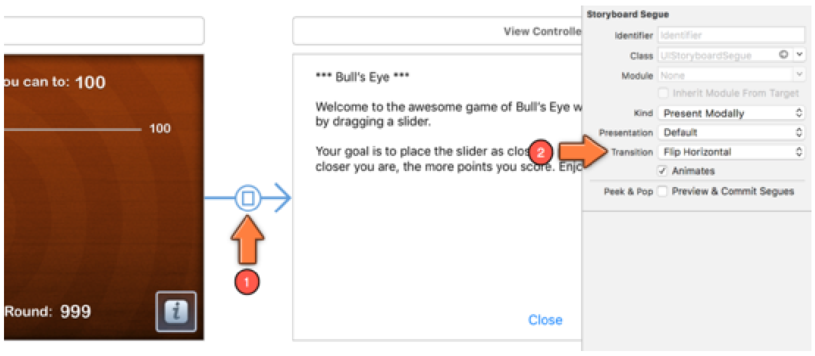
Your goal is to place th
closer you are, the mor

➤ Let go of the mouse button and a popup appears with several options. Choose **Present Modally**.



Now an arrow will appear between the two screens. This arrow represents the segue from the main scene to the About scene.

➤ Click the arrow to select it. Segues also have attributes. In the **Attributes inspector**, choose **Transition**, **Flip Horizontal**. That is the animation that UIKit will use to move between these screens.



*** Bull's Eye ***

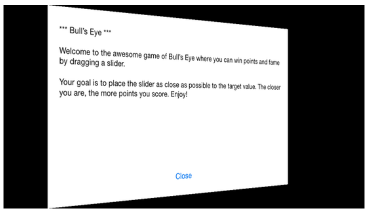
Welcome to the awesome game of Bull's Eye w
by dragging a slider.

Your goal is to place the slider as clos
closer you are, the more points you score. Enjo

Storyboard Segue

Identifier: Identifier
Class: UIStoryboardSegue
Module: None
Inherit Module From Target
Kind: Present Modally
Presentation: Default
Transition: Flip Horizontal
✓ Animates
Peak & Pop: Preview & Commit Segues

➤ Now you can run the app. Press the ⓘ button to see the new screen.



*** Bull's Eye ***

Welcome to the awesome game of Bull's Eye where you can win points and fame by dragging a slider.

Your goal is to place the slider as close as possible to the target value. The closer you are, the more points you score. Enjoy!

The About screen should appear with a neat animation. Good, that seems to work.

Dismiss the About view controller

However, there is an obvious shortcoming here: tapping the Close button seems to have no effect. Once the user enters the About screen they can never leave… that doesn't sound like good user interface design to me, does it?

The problem with segues is that they only go one way. To close this screen, you have to hook up some code to the Close button. As a budding iOS developer you already know how to do that: use an action method!

This time you will add the action method to `AboutViewController` instead of `ViewController`, because the Close button is part of the About screen, not the main game screen.

➤ Open **AboutViewController.swift** and replace its contents with the following:

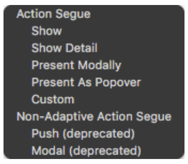
```
import UIKit

class AboutViewController: UIViewController {

  @IBAction func close() {
    dismiss(animated: true, completion: nil)
  }
}
```

This code in the `close()` action method tells UIKit to close the About screen with an animation.

If you had said `dismiss(animated: false, …)`, then there would be no page flip and the main screen would instantly reappear. From a user experience perspective, it's often better to show transitions from one screen to another via an animation.

That leaves you with one final step, hooking up the Close button's Touch Up Inside event to this new `close` action.

➤ Open the storyboard and Control-drag from the **Close** button to the About scene's View Controller. Hmm, strange, the **close** action should be listed in this popup, but it isn't. Instead, this is the same popup you saw when you made the segue:



Challenge: Bonus points if you can spot the error. It's a very common – and frustrating! – mistake.

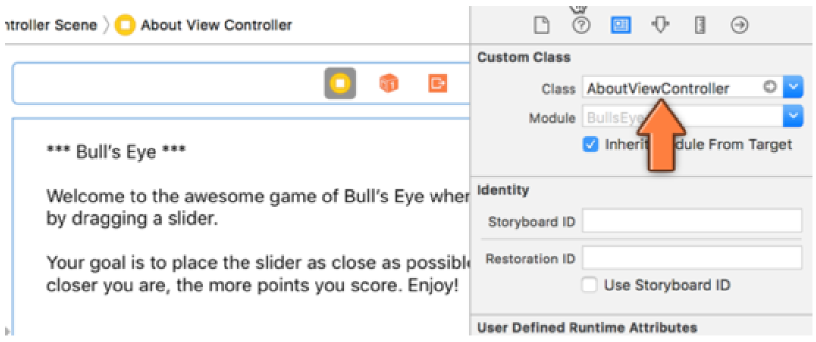
The problem is that this scene in the storyboard doesn't know yet that it is supposed to represent the `AboutViewController`.

Set the class for a view controller

You first added the AboutViewController.swift source file, and then dragged a new view controller on to the storyboard. But, you haven't told the storyboard that the design for this new view controller, in fact, belongs to `AboutViewController`. (That's why in the Document Outline it just says View Controller and not About View Controller.)

➤ Fortunately, this is easily remedied. In Interface Builder, select the About scene's **View Controller** and go to the **Identity inspector** (that's the button to the left of the Attributes inspector).

➤ Under **Custom Class**, type **AboutViewController**.


troller Scene 〉 About View Controller
Custom Class
Identity
User Defined Runtime Attributes

Class: AboutViewController
Module: BullsEye
[✓] Inherit Module From Target
Storyboard ID:
Restoration ID:
[ ] Use Storyboard ID

*** Bull's Eye ***

Welcome to the awesome game of Bull's Eye wher
by dragging a slider.

Your goal is to place the slider as close as possibl
closer you are, the more points you score. Enjoy!

Xcode should auto-complete this for you once you type the first few characters. If it doesn't, then double-check that you really have selected the View Controller and not one of the views inside it. (The view controller should also have a blue border on the storyboard to indicate it is selected.)

Now you should be able to connect the Close button to the action method.

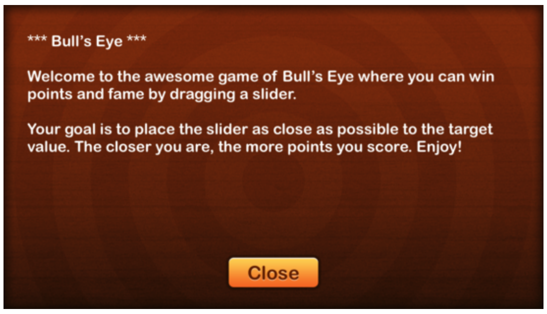
➤ Control-drag from the **Close** button to **About View Controller** in the Document Outline (or to the yellow circle at the top of the scene in storyboard). This should be old hat by now. The popup menu now does have an option for the **close** action (under Sent Events). Connect the button to that action.

➤ Run the app again. You should now be able to return from the About screen.

OK, that does get us a working about screen, but it does look a little plain doesn't it? What if you added some of the design changes you made to the main screen?

> **Exercise:** Add a background image to the About screen. Also, change the Close button on the About screen to look like the Hit Me! button and play around with the Text View properties in the Attribute Inspector. You should be able to do this by yourself now. Piece of cake! Refer back to the instructions for the main screen if you get stuck.

When you are done, you should have an About screen which looks something like this:



*** Bull's Eye ***

Welcome to the awesome game of Bull's Eye where you can win points and fame by dragging a slider.

Your goal is to place the slider as close as possible to the target value. The closer you are, the more points you score. Enjoy!

That looks good, but it could be better :] So how do you improve upon it?

Use a web view for HTML content

➤ Now select the **text view** and press the **Delete** key on your keyboard. (Yep, you're throwing it away, and after all those changes, too! But don't grieve for the Text View too much, you'll replace it with something better next.)

➤ Put a **Web View (deprecated)** in its place (as always, you can find this view in the Object Library).

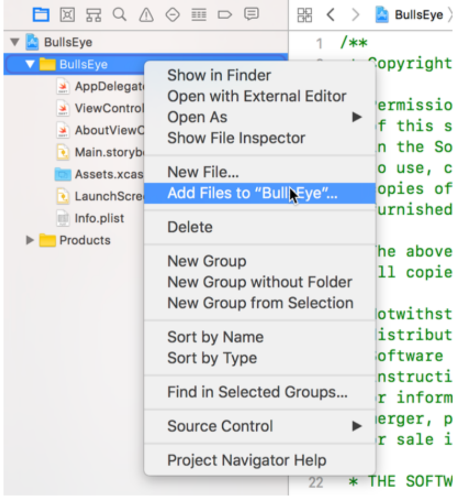
> **Note:** Web View has been deprecated, which means at some point in the future it will be replaced by its successor `WebKit View`. However, Web View still works fine at the moment, so for the purposes of this tutorial please use the deprecated version.

A web view, as its name implies, can show web pages. All you have to do is give it a URL to a web site or the name of a file to load. The web view object is named `UIWebView`.

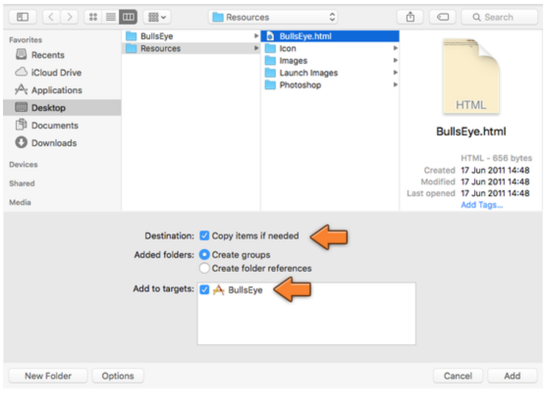
For this app, you will make it display a static HTML page from the application bundle, so it won't actually have to go online and download anything.

➤Download the static HTML page and upzip it.

➤ Then go to the **Project navigator** and right-click on the **BullsEye** group (the yellow folder). From the menu, choose **Add Files to "BullsEye"…**



➤ In the file picker, select the **BullsEye.html** file from the Resources folder. This is an HTML5 document that contains the gameplay instructions.



Destination: ☑ Copy items if needed
Added folders: ● Create groups
○ Create folder references
Add to targets: ☑ BullsEye

Make sure that **Copy items if needed** is selected and that under **Add to targets**, there is a checkmark in front of **BullsEye**. (If you don't see these options, click the Options button at the bottom of the dialog.)

➤ Press **Add** to add the HTML file to the project.

➤ In **AboutViewController.swift**, add an outlet for the web view:

```
class AboutViewController: UIViewController {
  @IBOutlet weak var webView: UIWebView!
  . . .
}
```

➤ In the storyboard file, connect the `UIWebView` to this new outlet. The easiest way to do this is to Control-drag from **About View Controller** (in the Document Outline) to the **Web View**.

(If you do it the other way around, from the Web View to About View Controller, then you'll connect the wrong thing and the web view will stay empty when you run the app.)

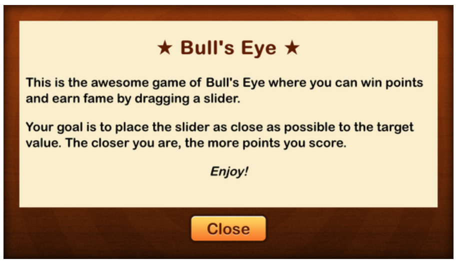
➤ In **AboutViewController.swift**, add a `viewDidLoad()` implementation:

```
override func viewDidLoad() {
  super.viewDidLoad()

  if let url = Bundle.main.url(forResource: "BullsEye",
                             withExtension: "html") {
    if let htmlData = try? Data(contentsOf: url) {
      let baseURL = URL(fileURLWithPath: Bundle.main.bundlePath)
      webView.load(htmlData, mimeType: "text/html",
                   textEncodingName: "UTF-8",
                            baseURL: baseURL)
    }
  }
}
```

This displays the HTML file using the web view.

The code may look scary but what goes on is not really that complicated: first it finds the **BullsEye.html** file in the application bundle, then loads it into a `Data` object, and finally it asks the web view to show the contents of this data object.

➤ Run the app and press the info button. The About screen should appear with a description of the gameplay rules, this time in the form of an HTML document:



★ Bull's Eye ★

This is the awesome game of Bull's Eye where you can win points and earn fame by dragging a slider.

Your goal is to place the slider as close as possible to the target value. The closer you are, the more points you score.

Enjoy!

Congrats! This completes the game. All the functionality is there and – as far as I can tell – there are no bugs to spoil the fun.

- Ray