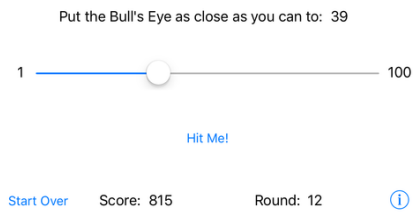
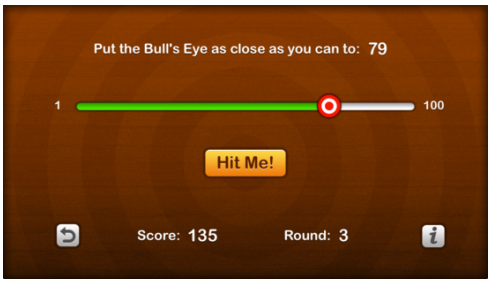
Ray Wenderlich para mim 11:01

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

Last time, we got rid of the status bar, but that's only the first step. We want to go from this:



To something that's more like this:



The actual controls don't change. You'll simply use images to smarten up their look, and you will also adjust the colors and typefaces.

You can put an image in the background, on the buttons, and even on the slider, to customize the appearance of each. The images you use should generally be in PNG format, though JPG files would work too.

> **Note:** Prefer learning via video? You might like to check out videos #37-40 of the free video version of this course, which correspond to this email.

Add the image assets

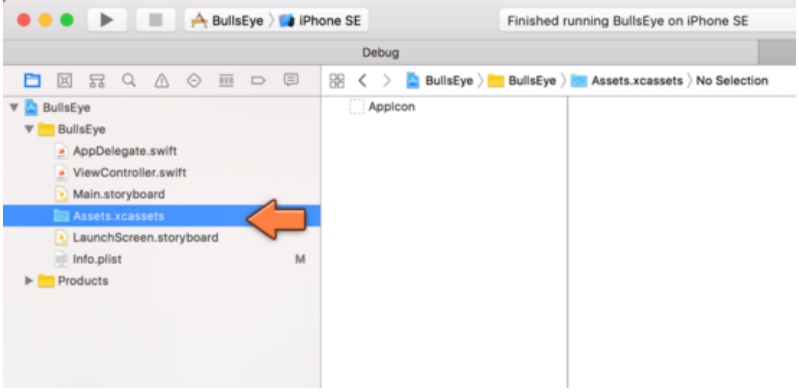
To get started, open up the Bull's Eye project where you left it off last time (or download the starter project from the corresponding forum discussion thread).

If you are artistically challenged, then don't worry, I have a set of images for you. But if you do have mad Photoshop skillz, then by all means feel free to design (and use) your own images.

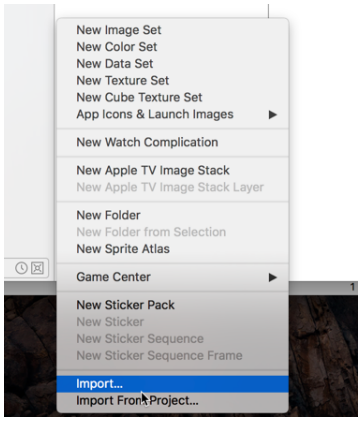
Download the assets here. Unzip it, and you should see a bunch of images. You will first import these images into the Xcode project.

➤ In the **Project navigator**, find **Assets.xcassets** and click on it.

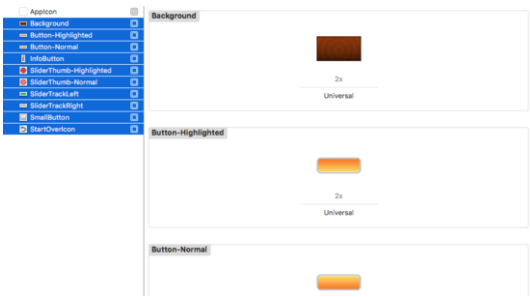
This is known as the asset catalog for the app and it contains all the app's images. Right now, it is empty and contains just a placeholder for the app icon, which you'll add soon.



➤ At the bottom of the secondary pane, the one with AppIcon, there is a **+** button. Click it and then select the **Import…** option:



Xcode shows a file picker. Select the **Resources** folder, click **Open**, and Xcode copies all the image files from that folder into the asset catalog:

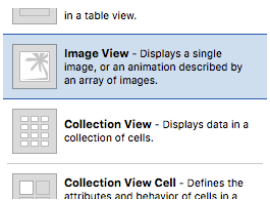


> **Note:** Instead of using the **Import…** menu option as above, you could also simply drag the necessary files from Finder on to the Xcode asset catalog view. As ever, there's more than one way to do the same thing in Xcode.

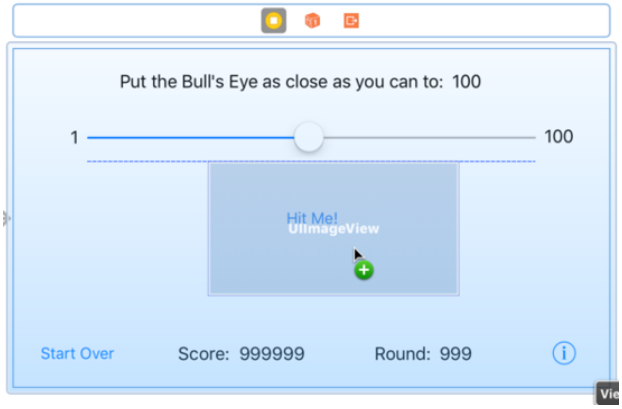
Put up the wallpaper

Let's begin by changing the drab white background in **Bull's Eye** to something more fancy.

➤ Open **Main.storyboard**. Go into the **Object Library** and locate an **Image View**. (Tip: if you type "image" into the search box at the bottom of the Object Library, it will quickly filter out all the other views.)

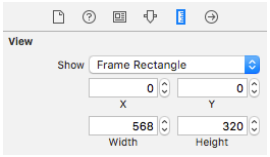


➤ Drag the image view on top of the existing user interface. It doesn't really matter where you put it, as long as it's inside the Bull's Eye View Controller.



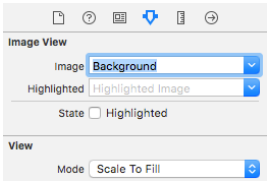
➤ With the image view still selected, go to the **Size inspector** (that's the one next to the Attributes inspector) and set X and Y to 0, Width to 568 and Height to 320.

This will make the image view cover the entire screen.



➤ Go to the **Attributes inspector** for the image view. At the top there is an option named **Image**. Click the downward arrow and choose **Background** from the list.

This will put the image named "Background" from the asset catalog into the image view.



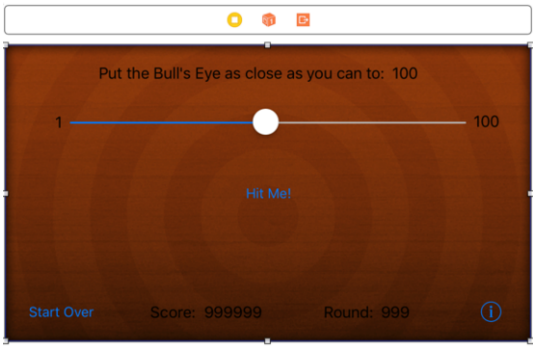
There is only one problem: the image now covers all the other controls. There is an easy fix for that; you have to move the image view behind the other views.

➤ In the **Editor** menu in Xcode's menu bar at the top of the screen, choose **Arrange → Send to Back**.

Sometimes Xcode gives you a hard time with this (it still has a few bugs) and you might not see the Send to Back item enabled. If so, try de-selecting the Image View and then selecting it again. Now the menu item should be available.

Alternatively, pick up the image view in the Document Outline and drag it to the top of the list of views, just below View, to accomplish the same thing. (The items in the Document Outline view are listed so that the backmost item is at the top of the list and the frontmost one is at the bottom.)

Your interface should now look something like this:



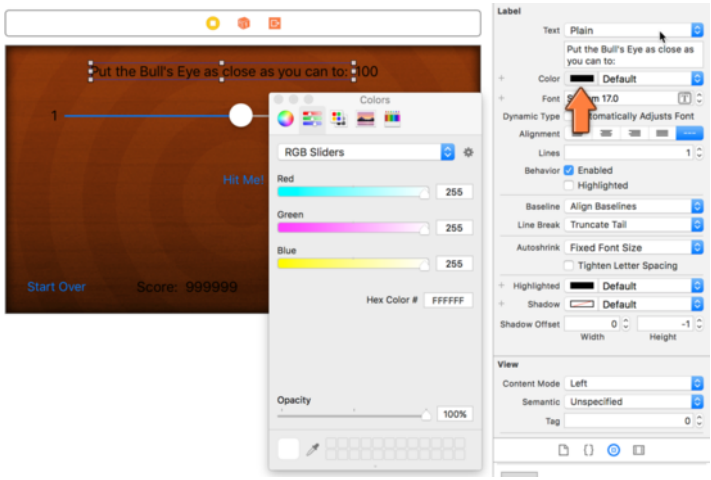
That takes care of the background. Run the app and marvel at the new graphics.

Change the labels

Because the background image is quite dark, the black text labels have become hard to read. Fortunately, Interface Builder lets you change their color. While you're at it, you

might change the font as well.

➤ Still in the storyboard, select the label at the top, open the **Attributes inspector** and click on the **Color** item - there are two parts to the item, so you need to click on the actual color and not the text part.



This opens the Color Picker, which has several ways to select colors. I prefer the sliders (second tab). If all you see is a gray scale slider, then select RGB Sliders from the select box at the top.

➤ Pick a pure white color, Red: 255, Green: 255, Blue: 255, Opacity: 100%.

➤ Click on the **Shadow** item from the Attributes inspector. This lets you add a subtle shadow to the label. By default this color is transparent (also known as “Clear Color”) so you won’t see the shadow. Using the Color Picker, choose a pure black color that is half transparent, Red: 0, Green: 0, Blue: 0, Opacity: 50%.

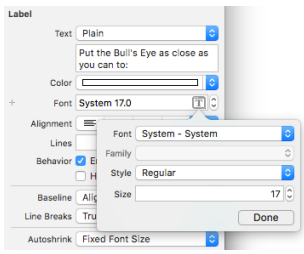
Note: Sometimes when you change the Color or Shadow attributes, the background color of the view also changes. This is a bug in Xcode. Put it back to Clear Color if that happens.

➤ Change the **Shadow Offset** to Width: 0, Height: 1. This puts the shadow below the label.

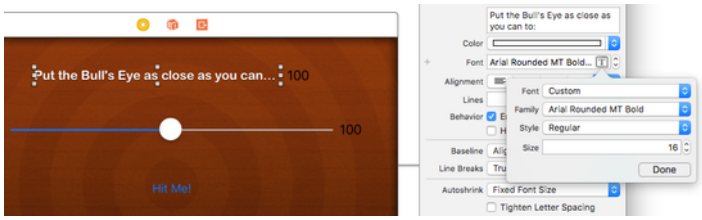
The shadow you’ve chosen is very subtle. If you’re not sure that it’s actually visible, then toggle the height offset between 1 and 0 a few times. Look closely and you should be able to see the difference. As I said, it’s very subtle.

➤ Click on the **[T]** icon of the **Font** attribute. This opens the Font Picker.

By default the System font is selected. That uses whatever is the standard system font for the user’s device. The system font is nice enough but we want something more exciting for this game.



➤ Choose **Font: Custom**. That enables the Family field. Choose **Family: Arial Rounded MT Bold**. Set the Size to 16.



➤ The label also has an attribute **Autoshrink**. Make sure this is set to **Fixed Font Size**.

If enabled, Autoshrink will dynamically change the size of the font if the text is larger than will fit into the label. That is useful in certain apps, but not in this one. Instead, you'll change the size of the label to fit the text rather than the other way around.

➤ With the label selected, press ⌘= on your keyboard, or choose **Size to Fit Content** from the **Editor** menu.

(If the Size to Fit Content menu item is disabled, then de-select the label and select it again. Sometimes Xcode gets confused about what is selected. Poor thing.)

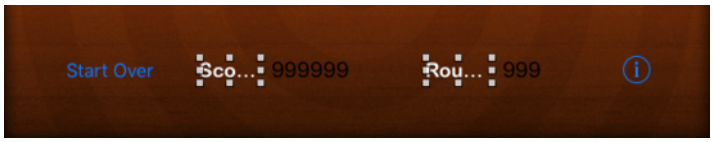
The label will now become slightly larger or smaller so that it fits snugly around the text. If the text got cut off when you changed the font, now all the text will show again.

You don't have to set these properties for the other labels one by one; that would be a big chore. You can speed up the process by selecting multiple labels and then applying these changes to that entire selection.

➤ Click on the **Score:** label to select it. Hold ⌘ and click on the **Round:** label. Now both labels will be selected. Repeat what you did above for these labels:

- Set Color to pure white, 100% opaque.
- Set Shadow to pure black, 50% opaque.
- Set Shadow Offset to width 0, height 1.
- Set Font to Arial Rounded MT Bold, size 16.
- Make sure Autoshrink is set to Fixed Font Size.

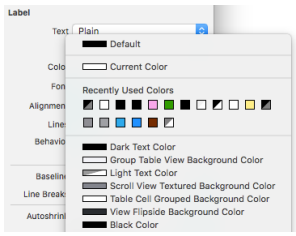
As you can see, in my storyboard the text no longer fits into the Score and Round labels:



You can either make the labels larger by dragging their handles to resize them manually, or you can use the **Size to Fit Content** option (⌘=). I prefer the latter because it's less work.

Tip: Xcode is smart enough to remember the colors you have used recently. Instead of going into the Color Picker all the time, you can simply choose a color from the Recently Used Colors menu.

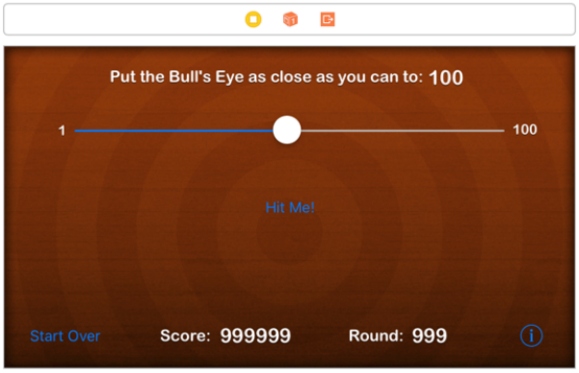
Click the tiny arrows at the end of the color field (or, if there is a text name for the color, click on the text part) and the menu will pop up:



Exercise: You still have a few labels to go. Repeat what you just did for the other labels. They should all become white, have the same shadow and have the same font. However, the two labels on either side of the slider (1 and 100) will have font size 14, while the other labels (the ones that will hold the target value, the score and the round number) will have font size 20 so they stand out more.

Because you've changed the sizes of some of the labels, your carefully constructed layout may have been messed up a bit. You may want to clean it up a little.

At this point, the game screen should look something like this:



All right, it's starting to look like something now. By the way, feel free to experiment with the fonts and colors. If you want to make it look completely different, then go right ahead. It's your app!

The buttons

Changing the look of the buttons works very much the same way.

➤ Select the **Hit Me!** button. In the **Size inspector** set its Width to 100 and its Height to 37.

➤ Center the position of the button on the inner circle of the background image.

➤ Go to the **Attributes inspector**. Change **Type** from System to **Custom**.

A "system" button just has a label and no border. By making it a custom button, you can style it any way you wish.

➤ Still in the **Attributes inspector**, press the arrow on the **Background** field and choose **Button-Normal** from the list.

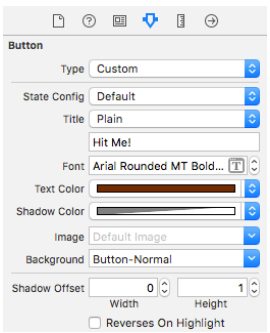
➤ Set the **Font** to **Arial Rounded MT Bold**, size 20.

➤ Set the **Text Color** to red: 96, green: 30, blue: 0, opacity: 100%. This is a dark brown color.

➤ Set the **Shadow Color** to pure white, 50% opacity. The shadow offset should be Width 0, Height 1.

> **Blending in**
> Setting the opacity to anything less than 100% will make the color slightly transparent (with opacity of 0% being fully transparent). Partial transparency makes the color blend in with the background and makes it appear softer.
> Try setting the shadow color to 100% opaque pure white and notice the difference.

This finishes the setup for the Hit Me! button in its "default" state:



Buttons can have more than one state. When you tap a button and hold it down, it should appear "pressed down" to let you know that the button will be activated when you lift your finger. This is known as the **highlighted** state and is an important visual cue to the user.

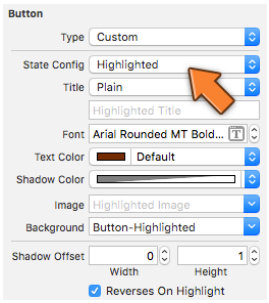
➤ With the button still selected, click the **State Config** setting and pick **Highlighted** from the menu. Now the attributes in this section reflect the highlighted state of the button.

➤ In the **Background** field, select **Button-Highlighted**.

➤ Make sure the highlighted **Text Color** is the same color as before (red 96, green 30, blue 0, or simply pick it from the Recently Used Colors menu). Change the **Shadow Color** to half-transparent white again.

➤ Check the **Reverses On Highlight** option. This will give the appearance of the label being pressed down when the user taps the button.

You could change the other properties too, but don't get too carried away. The highlight effect should not be too jarring.



To test the highlighted look of the button in Interface Builder you can toggle the **Highlighted** box in the **Control** section, but make sure to turn it off again or the button will initially appear highlighted when the screen is shown.

That's it for the Hit Me! button. Styling the Start Over button is very similar, except you will replace its title text with an icon.

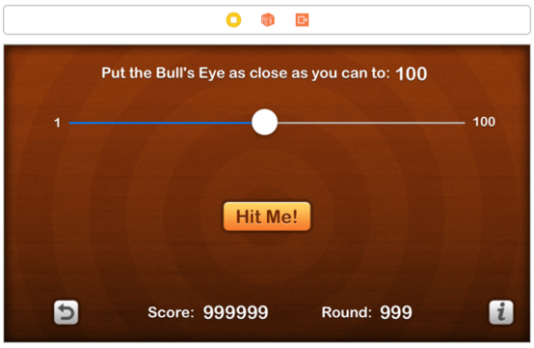
➤ Select the **Start Over** button and change the following attributes:

- Set Type to Custom.
- Remove the text "Start Over" from the button.
- For Image choose **StartOverIcon**
- For Background choose **SmallButton**
- Set Width and Height to 32.

You won't set a highlighted state on this button - let UIKit take care of this. If you don't specify a different image for the highlighted state, UIKit will automatically darken the button to indicate that it is pressed.

➤ Make the same changes to the ⓘ button, but this time choose **InfoButton** for the image.

The user interface is almost done. Only the slider is left…



The slider

Unfortunately, you can only customize the slider a little bit in Interface Builder. For the more advanced customization that this game needs – putting your own images on the thumb and the track – you have to resort to writing source code.

Everything you have done so far in Interface Builder you could also have done in code. Setting the color on a button, for example, can be done by sending the `setTitleColor()` message to the button. (You would normally do this in `viewDidLoad`.)

However, I find that doing visual design work is much easier and quicker in a visual editor such as Interface Builder than writing the equivalent source code. But for the slider you have no choice.

➤ Go to **ViewController.swift**, and add the following to `viewDidLoad()`:

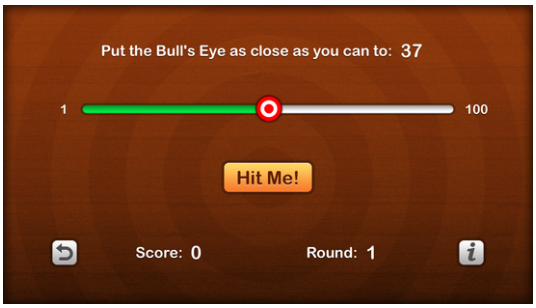
```
let thumbImageNormal = UIImage(named: "SliderThumb-Normal")!
slider.setThumbImage(thumbImageNormal, for: .normal)

let thumbImageHighlighted = UIImage(named: "SliderThumb-Highlighted")!
slider.setThumbImage(thumbImageHighlighted, for: .highlighted)

let insets = UIEdgeInsets(top: 0, left: 14, bottom: 0, right: 14)

let trackLeftImage = UIImage(named: "SliderTrackLeft")!
let trackLeftResizable =
                trackLeftImage.resizableImage(withCapInsets: insets)
slider.setMinimumTrackImage(trackLeftResizable, for: .normal)

let trackRightImage = UIImage(named: "SliderTrackRight")!
let trackRightResizable =
                trackRightImage.resizableImage(withCapInsets: insets)
slider.setMaximumTrackImage(trackRightResizable, for: .normal)
```

This sets four images on the slider: two for the thumb and two for the track. (And if you're wondering what the "thumb" is, that's the little circle in the center of the slider, the one that you drag around to set the slider value.)

The thumb works like a button so it gets an image for the normal (un-pressed) state and one for the highlighted state.

The slider uses different images for the track on the left of the thumb (green) and the track to the right of the thumb (gray).

➤ Run the app. You have to admit it looks fantastic now!



- Ray

If you'd like to stop receiving the iOS Apprentice Email Course but still stay subscribed to raywenderlich.com Weekly, click here.
