IOS APP 1: PART 7

PREPARED BY FARHAJ AHMED

➤ Do the same for the other three constraints. Note that for two of the constraints, Safe Area will be the Second Item and not the First Item. So change either one based on which one specifies Safe Area.

➤ Run your app on both the iPhone X simulator and at least one of the other simulators like the iPhone 8 one, to make sure that your change works correctly on all devices.

Crossfade

There's one final bit of knowledge I want to impart before calling the game complete — Core Animation. This technology makes it very easy to create really sweet animations, with just a few lines of code, in your apps. Adding subtle animations (with emphasis on subtle!) can make your app a delight to use.

You will add a simple crossfade after the Start Over button is pressed, so the transition back to round one won't seem so abrupt.

➤ In **ViewController.swift**, change `startNewGame()` to:

```
@IBAction func startNewGame() {
  ...
  startNewRound()
  // Add the following lines
  let transition = CATransition()
  transition.type = CATransitionType.fade
  transition.duration = 1
  transition.timingFunction = CAMediaTimingFunction(name:
                              CAMediaTimingFunctionName.easeOut)
  view.layer.add(transition, forKey: nil)
}
```

Everything after the comment telling you to add the following lines, all the `CATransition` stuff, is new.

I'm not going to go into too much detail here. Suffice it to say you're setting up an animation that crossfades from what is currently on the screen to the changes you're making in `startNewRound()` – reset the slider to center position and reset the values of the labels.

iOS App 1: Part 7

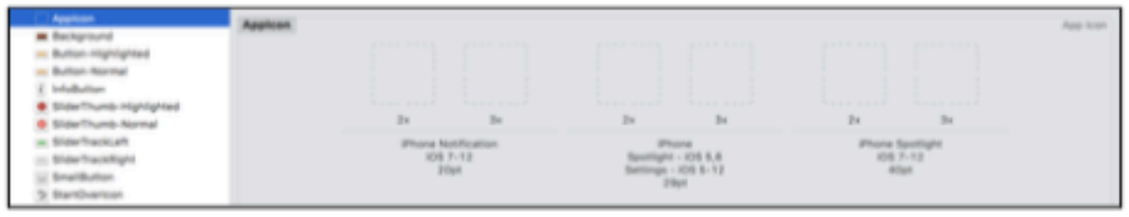
➤ Run the app and move the slider so that it is no longer in the center. Press the Start Over button and you should see a subtle crossfade animation.

The screen crossfades between the old and new states

The icon

You're almost done with the app, but there are still a few loose ends to tie up. You may have noticed that the app has a really boring white icon. That won't do!

➤ Open the asset catalog (**Assets.xcassets**) and select **AppIcon**:



The AppIcon group in the asset catalog

This has several icon groups for the different types of icons the app needs.

➤ In Finder, open the **Icon** folder from the resources. Drag the **Icon-40.png** file into the first slot, **iPhone Notification 20pt**:

Dragging the icon into the asset catalog

iOS App 1: Part 7

You may be wondering why you're dragging the Icon-40.png file and not the Icon-20.png into the slot for 20pt. Notice that this slot says **2x**, which means it's for Retina devices and on Retina screens one point counts as two pixels. So, 20pt = 40px. And the 40 in the icon name is for the size of the icon in pixels. Makes sense?

➤ Drag the **Icon-60.png** file into the **3x** slot next to it. This is for the iPhone Plus devices with their 3x resolution.

➤ For **iPhone Spotlight & Settings 29pt**, drag the **Icon-58.png** file into the 2x slot and **Icon-87.png** into the 3x slot. (What, you don't know your times table for 29?)

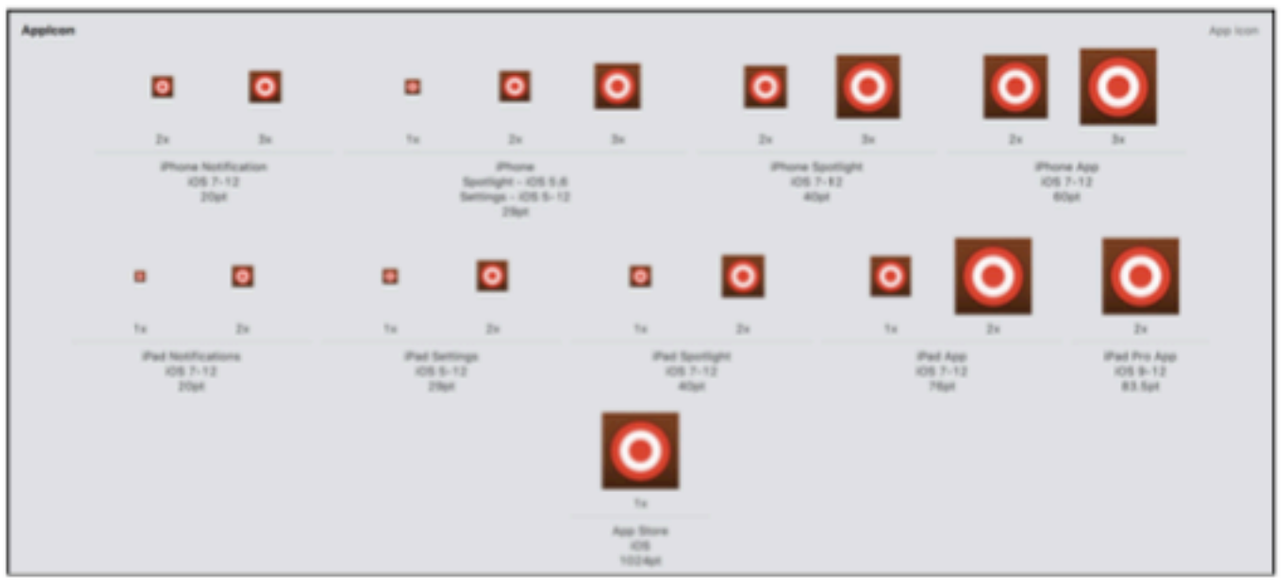
➤ For **iPhone Spotlight 40pt**, drag the **Icon-80.png** file into the 2x slot and **Icon-120.png** into the 3x slot.

➤ For **iPhone App 60pt**, drag the **Icon-120.png** file into the 2x slot and **Icon-180.png** into the 3x slot.

That's four icons in two different sizes. Phew!

The other AppIcon groups are mostly for the iPad.

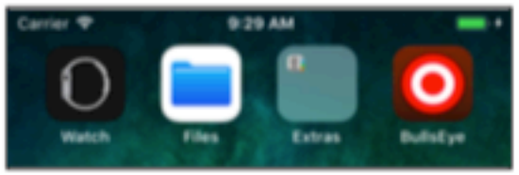
➤ Drag the specific icons (based on size) into the proper slots for iPad. Notice that the iPad icons need to be supplied in 1x as well as 2x sizes (but not 3x). You may need to do some mental arithmetic here to figure out which icon goes into which slot!



The full set of icons for the app

iOS App 1: Part 7

➤ Run the app and close it. You'll see that the icon has changed on the Simulator's springboard. If not, remove the app from the Simulator and try again (sometimes the Simulator keeps using the old icon and re-installing the app will fix this).



The icon on the Simulator's springboard

Display name

One last thing. You named the project **BullsEye** and that is the name that shows up under the icon. However, I'd prefer to spell it "**Bull's Eye**."

There is only limited space under the icon and for apps with longer names you have to get creative to make the name fit. For this game, however, there is enough room to add the space and the apostrophe.

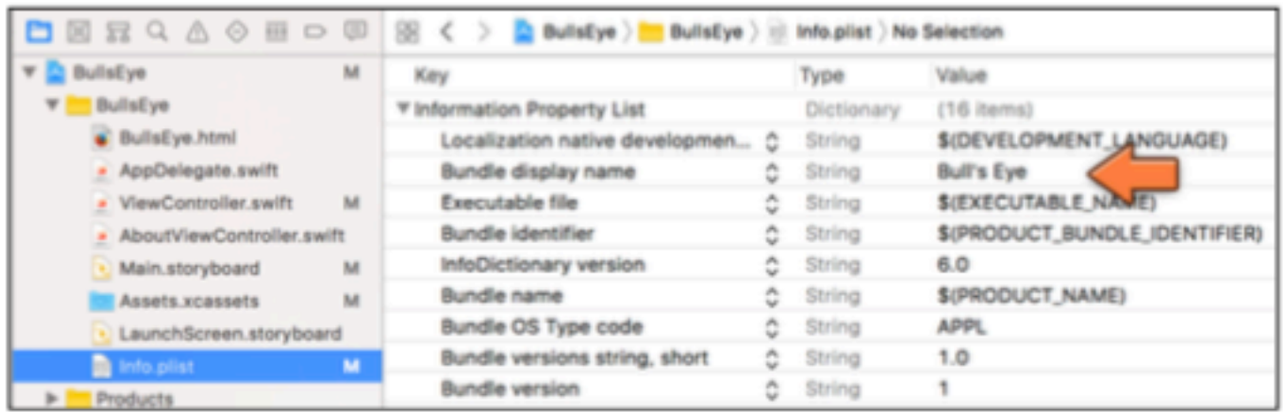
➤ Go to the **Project Settings** screen. The very first option is **Display Name**. Change this to **Bull's Eye**.

Changing the display name of the app

Like many of the project's settings you can also find the display name in the app's Info.plist file. Let's have a look.

iOS App 1: Part 7

➤ From the **Project navigator**, select **Info.plist**.



The display name of the app in Info.plist

The row **Bundle display name** contains the new name you've just entered.

> **Note**: If **Bundle display name** is not present, the app will use the value from the field **Bundle name**. That has the special value "$(PRODUCT_NAME)", meaning Xcode will automatically put the project name, BullsEye, in this field when it adds the Info.plist to the application bundle. By providing a **Bundle display name** you can override this default name and give the app any name you want.

➤ Run the app and quit it to see the new name under the icon.

The bundle display name setting changes the name under the icon

Awesome, that completes your very first app!

iOS App 1: Part 7

Running on device

So far, you've run the app on the Simulator. That's nice and all but probably not why you're learning iOS development. You want to make apps that run on real iPhones and iPads! There's hardly a thing more exciting than running an app that you made on your own phone. And, of course, to show off the fruits of your labor to other people!

Don't get me wrong: developing your apps on the Simulator works very well. When developing, I spend most of my time with the Simulator and only test the app on my iPhone every so often.

However, you do need to run your creations on a real device in order to test them properly. There are some things the Simulator simply cannot do. If your app needs the iPhone's accelerometer, for example, you have no choice but to test that functionality on an actual device. Don't sit there and shake your Mac!

Until a few years back, you needed a paid Developer Program account to run apps on your iPhone. Now, you can do it for free. All you need is an Apple ID. And the latest Xcode makes it easier than ever before.

Configuring your device for development

➤ Connect your iPhone, iPod touch, or iPad to your Mac using a USB cable.

➤ From the Xcode menu bar select **Window ▸ Devices and Simulators** to open the Devices and Simulators window.

Mine looks like this:

The Devices and Simulators window

On the left is a list of devices that are currently connected to my Mac and which can be used for development.

➤ Click on your device name to select it.

If this is the first time you're using the device with Xcode, the Devices window will say something like, "iPhone is not paired with your computer." To pair the device with Xcode, you need to unlock the device first (hold the home button, or use Face ID, depending on your device). After unlocking, an alert will pop up on the device asking you to trust the computer you're trying to pair with. Tap on **Trust** to continue.

Xcode will now refresh the page and let you use the device for development. Give it a few minutes (see the progress bar in the main Xcode window). If it takes too long, you may need to unplug the device and plug it back in.

At this point it's possible you may get the error message, "An error was encountered while enabling development on this device." You'll need to unplug the device and reboot it. Make sure to restart Xcode before you reconnect the device.

Also, note the checkbox that says **Connect via network**? That checkbox (gasp!) allows you to run and debug code on your iPhone over WiFi!

Cool, that is the device sorted.

Adding your developer account to Xcode

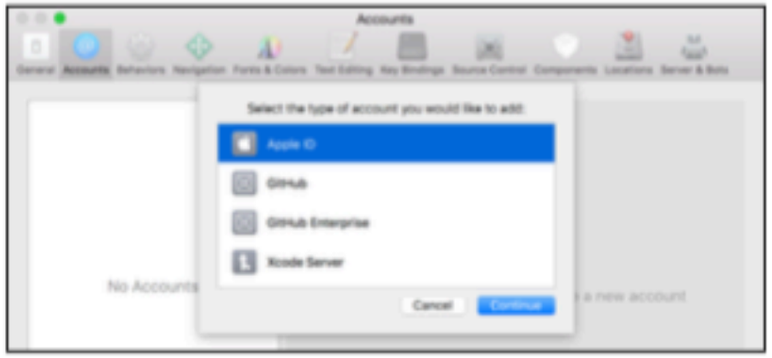
The next step is setting up your Apple ID with Xcode. It's OK to use the same Apple ID that you're already using with iTunes and your iPhone, but if you run a business, you might want to create a new Apple ID to keep things separate. Of course, if you've already registered for a paid Developer Program account, you should use that Apple ID.

➤ Open the **Accounts** pane in the Xcode Preferences window:

The Accounts preferences

iOS App 1: Part 7

➤ Click the + button at the bottom and select **Add Apple ID** from the list of options.



Xcode Account Type selection

Xcode will ask for your Apple ID:

Adding your Apple ID to Xcode

➤ Type your Apple ID username and password and click **Sign In**.

Xcode verifies your account details and adds it to the stored list of accounts.

> **Note**: It's possible that Xcode is unable to use the Apple ID your provided — for example, if it has been used with a Developer Program account in the past that is now expired. The simplest solution is to make a new Apple ID. It's free and only takes a few minutes. appleid.apple.com

You still need to tell Xcode to use this account when building your app.

iOS App 1: Part 7

Code signing

➤ Go to the **Project Settings** screen for your app target. In the **General** tab go to the **Signing** section.

The Signing options in the Project Settings screen

In order to allow Xcode to put an app on your iPhone, the app must be *digitally signed* with your **Development Certificate**. A *certificate* is an electronic document that identifies you as an iOS application developer and is valid only for a specific amount of time.

Apps that you want to submit to the App Store must be signed with another certificate, the **Distribution Certificate**. To create (and use) a distribution certificate, you must be a member of the paid Developer Program. However, creating/using a development certificate is free.

In addition to a valid certificate, you also need a **Provisioning Profile** for each app you make. Xcode uses this profile to sign the app for use on your particular device (or devices). The specifics don't really matter, just know that you need a provisioning profile or the app won't go on your device.

Making the certificates and provisioning profiles used to be a really frustrating and error-prone process. Fortunately, those days are over: Xcode now makes it really easy. When the **Automatically manage signing** option is enabled, Xcode will take care of all this business with certificates and provisioning profiles and you don't have to worry about a thing.

➤ Click on **Team** to select your Apple ID.

Xcode will now automatically register your device with your account, create a new Development Certificate, and download and install the Provisioning Profile on your device. These are all steps you had to do by hand in the past, but now Xcode takes care of all that.

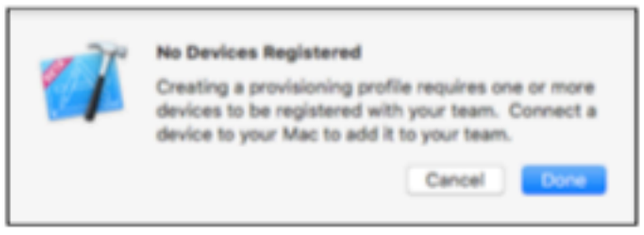
iOS App 1: Part 7

You could get some signing errors like these:

Signing/team set up errors

The app's Bundle Identifier – or App ID as it's called here – must be unique. If another app is already using that identifier, then you cannot use it anymore. That's why you're supposed to start the Bundle ID with your own domain name. The fix is easy: change the Bundle Identifier field to something else and try again.

It's also possible you get this error (or something similar):



No devices registered

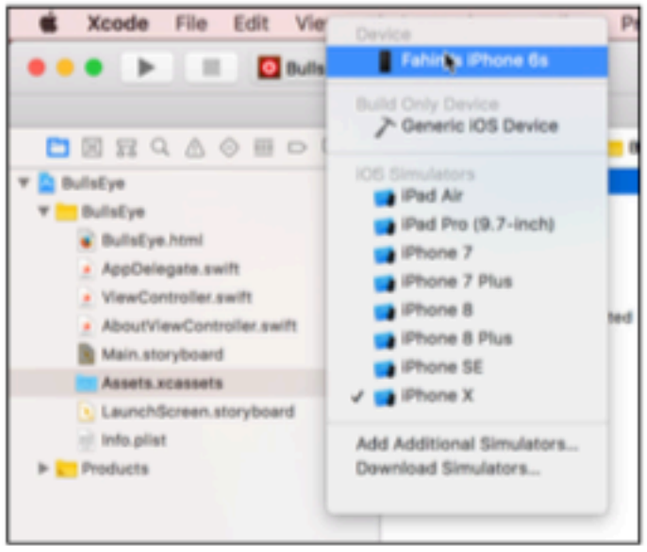
Xcode must know about the device that you're going to run the app on. That's why I asked you to connect your device first. Double-check that your iPhone or iPad is still connected to your Mac and that it is listed in the Devices window.

Running on device

If everything goes smoothly, go back to Xcode's main window and click on the dropdown in the toolbar to change where you will run the app. The name of your device should be in that list somewhere.

iOS App 1: Part 7

On my system it looks like this:



Setting the active device

You're all set and ready to go!

➤ Press **Run** to launch the app.

At this point you may get a pop-up with the question "codesign wants to sign using key... in your keychain." If so, answer with **Always Allow**. This is Xcode trying to use the new Development Certificate you just created — you just need to give it permission first.

Does the app work? Awesome! If not, read on...

When things go wrong...

There are a few things that can go wrong when you try to put the app on your device, especially if you've never done this before, so don't panic if you run into problems.

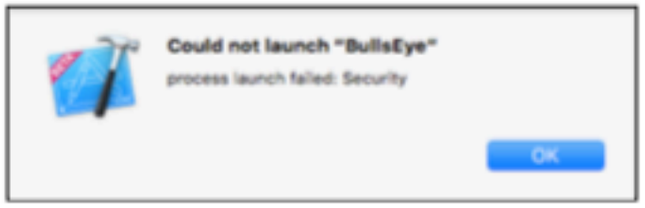
The device is not connected

Make sure your iPhone, iPod touch, or iPad is connected to your Mac. The device must be listed in Xcode's Devices window and there should not be a yellow warning icon.

iOS App 1: Part 7

The device does not trust you

You might get this warning:



Quick, call security!

On the device itself there will be a pop-up with the text, "Untrusted Developer. Your device management settings do not allow using apps from developer..."

If this happens, open the Settings app on the device and go to **General ▸ Profile**. Your Apple ID should be listed in that screen. Tap it, followed by the Trust button. Then try running the app again.

The device is locked

If your phone locks itself with a passcode after a few minutes, you might get this warning:

The app won't run if the device is locked

Simply unlock your device (hold the home button, type in the 4-digit passcode, or use FaceID) and press Run again.

Signing certificates

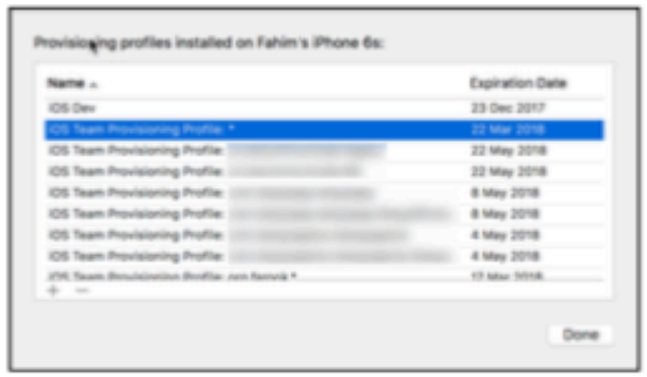
If you're curious about these certificates, then open the **Preferences** window and go to the **Accounts** tab. Select your account and click the **Manage Certificates...** button in the bottom-right corner.

This brings up another panel, listing your signing certificates:

The Manage Certificates panel

When you're done, close the panel and go to the **Devices and Simulators** window.

You can see the provisioning profiles that are installed on your device by right-clicking the device name and choosing **Show Provisioning Profiles**.



The provisioning profiles on your device

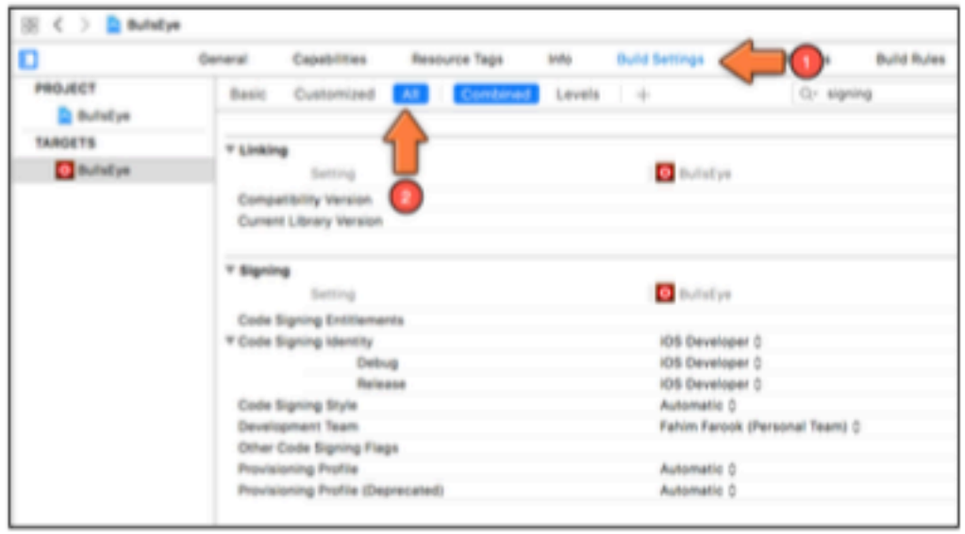
The "iOS Team Provisioning Profile" is the one that allows you to run the app on your device. (By the way, they call it the "team" profile because often there is more than one developer working on an app and they can all share the same profile.)

You can have more than one certificate and provisioning profile installed. This is useful if you're on multiple development teams or if you prefer to manage the provisioning profiles for different apps by hand.

To see how Xcode chooses which profile and certificate to sign your app with, go to the **Project Settings** screen and switch to the **Build Settings** tab. There are a lot of settings in this list, so filter them by typing **signing** in the search box. Also make sure **All** is selected, not Basic.

iOS App 1: Part 7

The screen will look something like this:



The Code Signing settings

Under **Code Signing Identity** it says **iOS Developer**. This is the certificate that Xcode uses to sign the app. If you click on that line, you can choose another certificate. Under **Provisioning Profile** you can change the active profile. Most of the time you won't need to change these settings, but at least you know where to find them now.

And that concludes everything you need to know about running your app on an actual device.

The end... or the beginning?

It has been a bit of a journey to get to this point – if you're new to programming, you've had to get a lot of new concepts into your head. I hope your brain didn't explode!

At least you should have gotten some insight into what it takes to develop an app.

I don't expect you to totally understand everything that you did, especially not the parts that involved writing Swift code. It is perfectly fine if you didn't, as long as you're enjoying yourself and you sort of get the basic concepts of objects, methods and variables.

If you were able to follow along and do the exercises, you're in good shape!

I encourage you to play around with the code a bit more. The best way to learn programming is to do it, and that includes making mistakes and messing things up. I hereby grant you full permission to do so! Maybe you can add some cool new features to the game (and if you do, please let me know).

In the Source Code folder for this book you can find the complete source code for the *Bull's Eye* app. If you're still unclear about something we did, it might be a good idea to look at this cleaned up source code.

If you're interested in how I made the graphics, then take a peek at the Photoshop files in the Resources folder. The wood background texture was made by Atle Mo from subtlepatterns.com.

If you're feeling exhausted after all that coding, pour yourself a drink and put your feet up for a bit. You've earned it! On the other hand, if you just can't wait to get to grips with more code, let's move on to our next app!

FEEDBACK TIME

http://bit.ly/iOSFeedback

THANK YOU