Unity tutorial -- building for iOS

1.Testing on iOS device

If you are working with a Mac or Windows machine and want to test your Unity code on an iphone or iPad, the easiest way is to use an app -- Unity Remote 5.

First you need to install Unity Remote 5 from the apple store on your iOS device. Start this app and connect your device to machine.

Second, open Unity on your machine, go to edit->project setting->editor, where you can find a dropbox and select any iOS device. Refer to the following doc:
https://docs.unity3d.com/Manual/UnityRemote5.html

Here is a video in case you run into the same issue on the previous step:
https://www.youtube.com/watch?v=eAFIg5JaHyw

2. Building your Unity game to an iOS device for testing

This tutorial was last tested on September 8th 2020, using Unity 2019.4, Xcode 11.7 and iOS 13.7. This file is modified from an earlier version provided by Unity:
https://learn.unity.com/tutorial/building-for-mobile

Introduction

In this lesson, we’re going to build a sample Unity project to an iOS device for testing.

This lesson doesn’t cover building a game for distribution to other devices, or submitting a game to the App Store.

Since the release of Xcode 7 in 2015, it has been possible to build apps - including games made with Unity - using a free Apple ID. Prior to this, only members of the paid Apple Developer Program could do so. The free option is limited: you can’t use services such as Game Center or In-App Purchases, and you can’t submit your game to the App Store. However, you can test your game on an iOS device using a free Apple ID.

This lesson assumes that you will be using a free Apple ID and are not enrolled in the Apple Developer Program. If you’re enrolled in the Apple Developer Program, you can still follow this lesson - however, you will probably wish to use the Apple Developer Portal to configure your certificates, provisioning profiles and devices rather than allowing Xcode to do it automatically, as we will do in this lesson.

What you will need

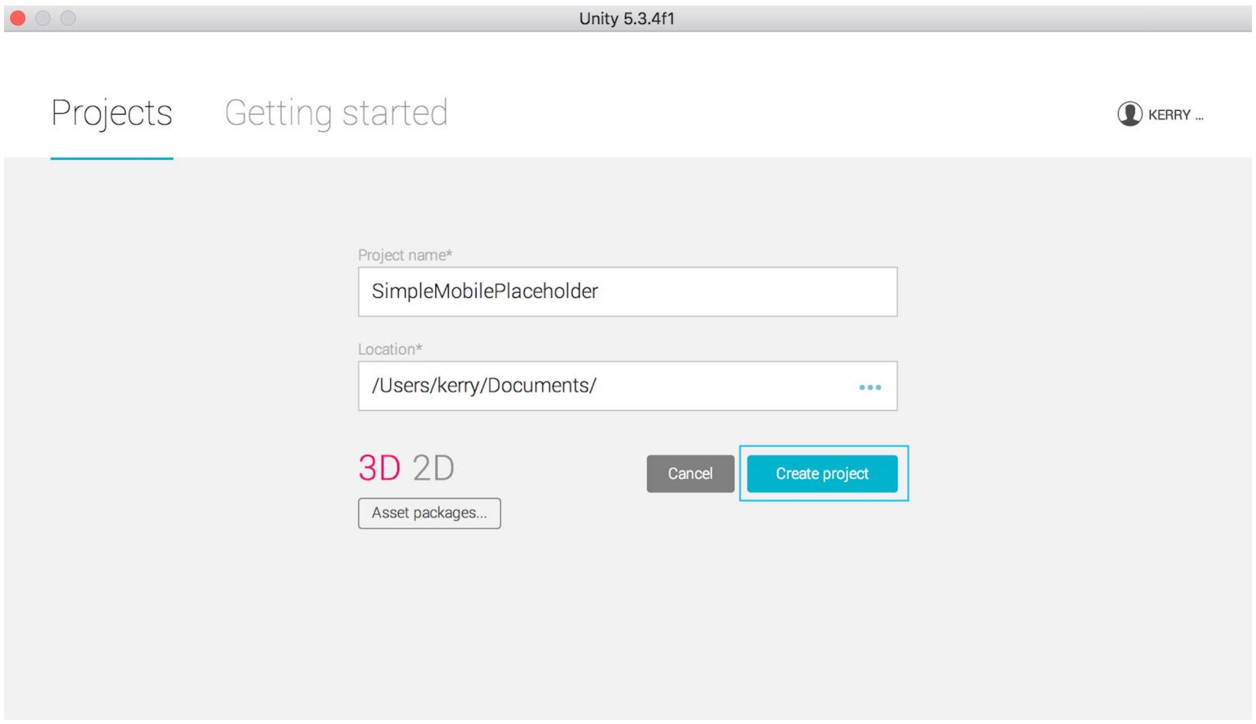
To follow this lesson, you will need:

- A Mac computer running running OS X 10.11 or later
- The latest version of Xcode (available from the Mac App Store)
- The latest version of Unity (available here)
- An iOS device

Setting up a simple Unity project

First, let’s build a simple Unity project.

- Open Unity and create a new project.
- Name the project.
- Choose a location to save it to.
- Ensure that "3D" is selected to use the Unity Editor in 3D mode.
- Click the **Create project** button to create the project.



Project name*
SimpleMobilePlaceholder

Location*
/Users/kerry/Documents/

3D 2D

- Add an object to the scene, such as a cube.
- You should now see a cube in the Scene View. Enter Play Mode to test the project. You should see the cube in the Game View too.

That’s it! Now that we know how the project works, let’s get ready to build it to a device.

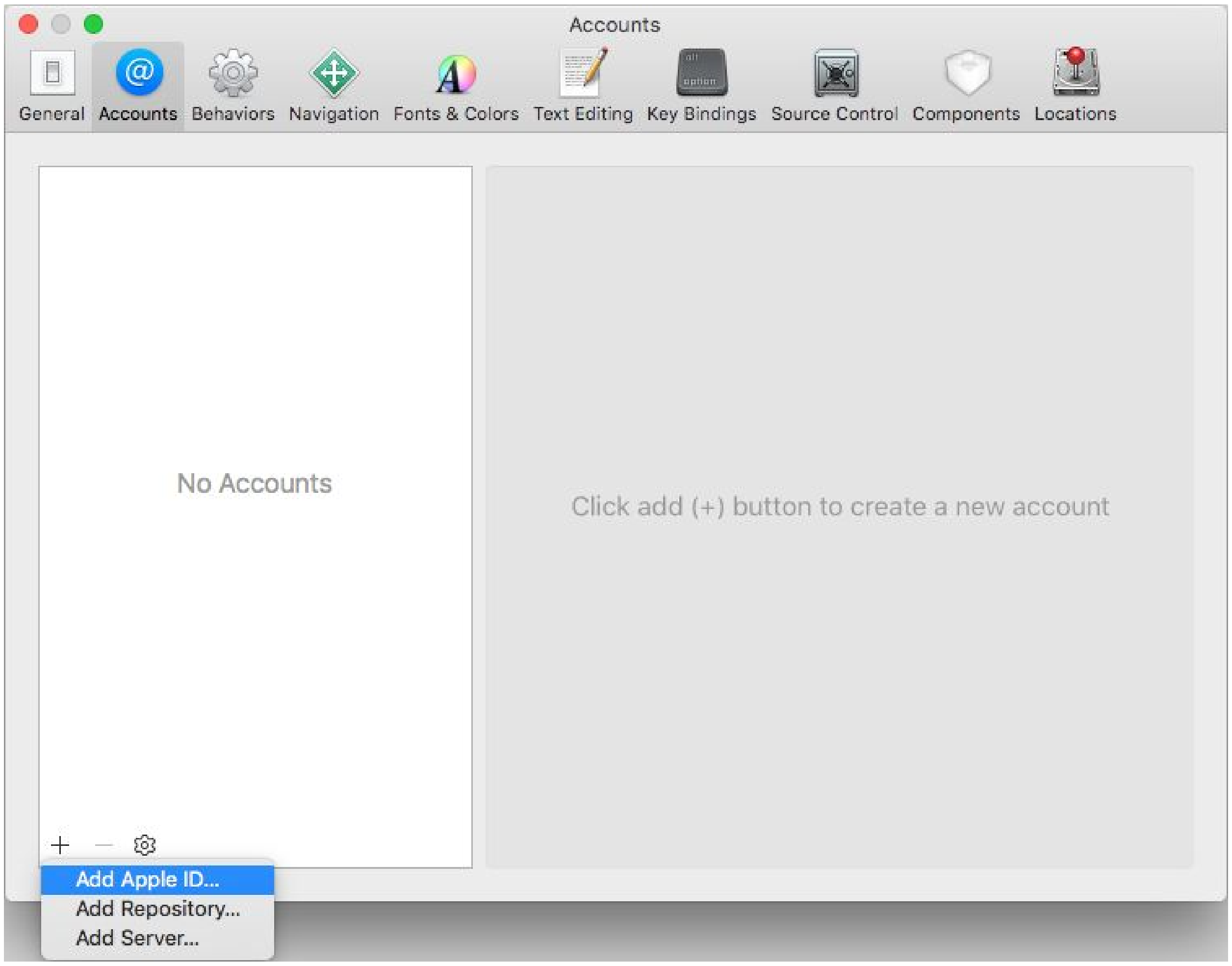
Adding your Apple ID to Xcode

Before we can build to a device, we need to set up an Apple ID and add it to Xcode.

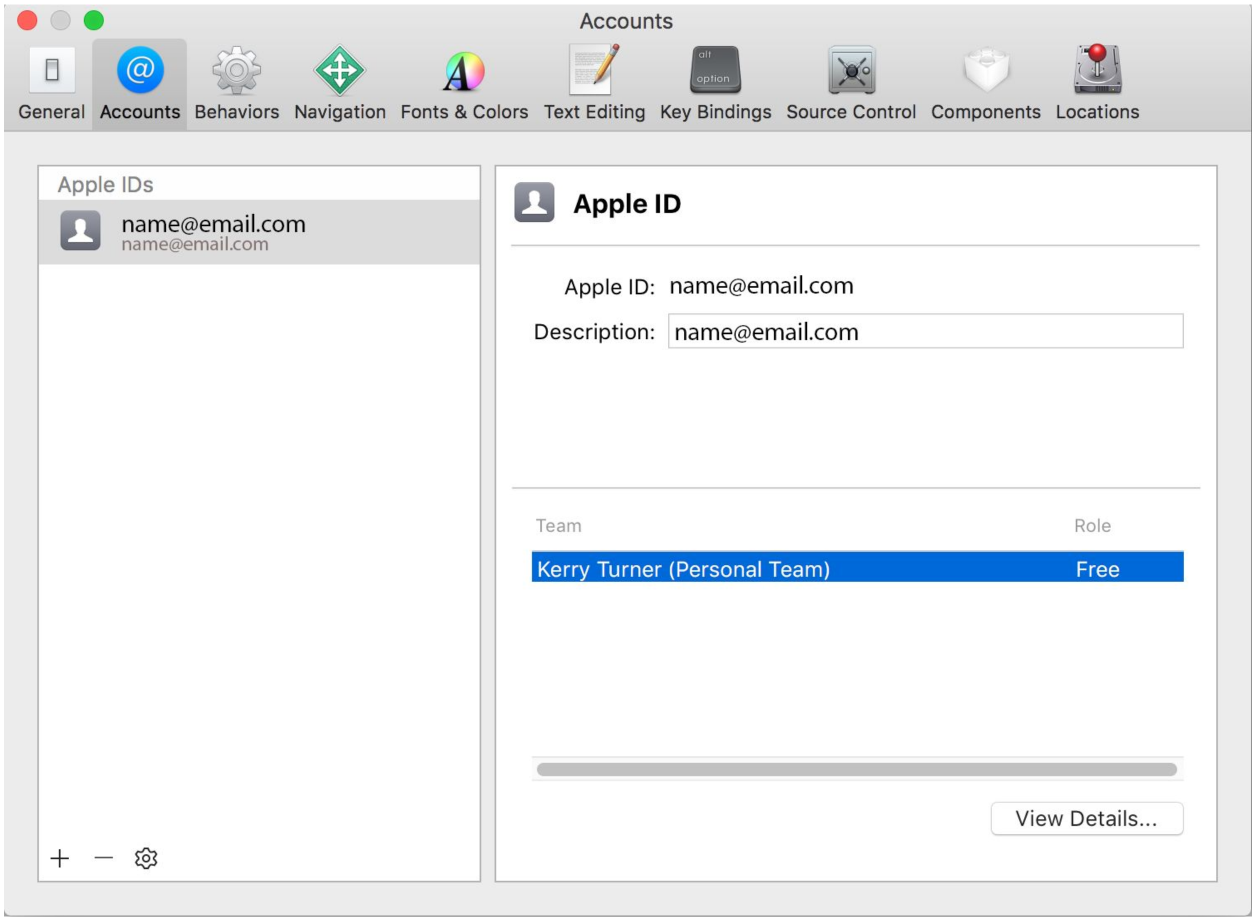
If you don’t yet have an Apple ID, obtain one from the Apple ID site.

Once you have obtained an Apple ID, you must add it to Xcode.

- Open Xcode.
- From the menu bar at the top of the screen choose **Xcode** > **Preferences**. This will open the Preferences window.
- Choose **Accounts** at the top of the window to display information about the Apple IDs that have been added to Xcode.
- To add your Apple ID, click the plus sign at the bottom left corner and choose **Add Apple ID**.



- A popup will appear, requesting your Apple ID and password. Enter these.
- Your Apple ID will then appear in the list. Select your Apple ID to see more information about it.
- Under the heading **Team**, you will see a list of all Apple Developer Program teams that you are a part of. If you're using a free Apple ID that isn't enrolled in the Apple Developer Program, you will see your name followed by "(Personal Team)".



Apple ID: name@email.com
Description: name@email.com

Team	Role
Kerry Turner (Personal Team)	Free

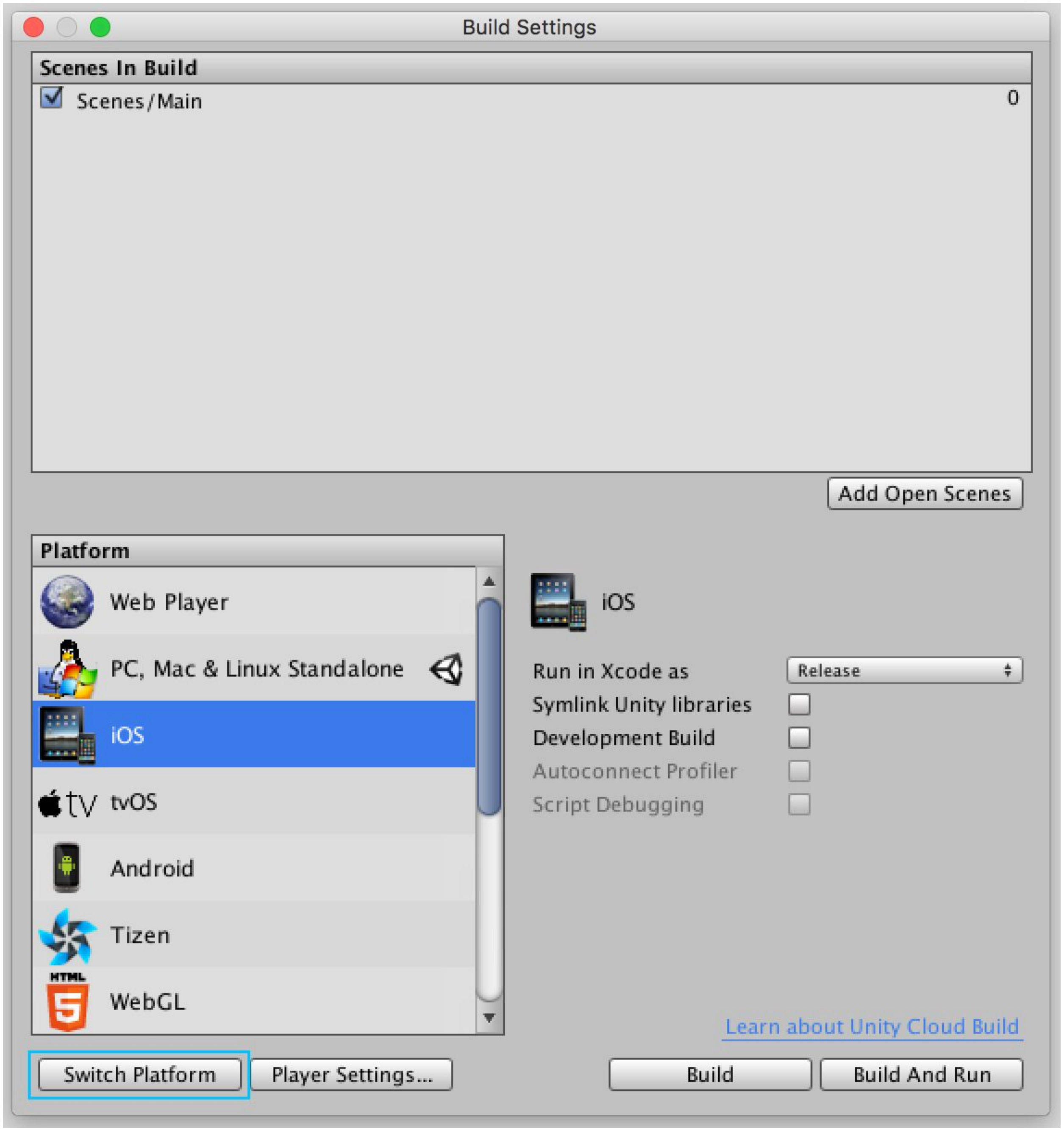
In the Apple Developer Program, teams are how you organise who has access to a project, what permissions they have and so on. When you use a free Apple ID, Apple creates what is known as a Personal Team for your Apple ID that only has you on it. Don't worry about it for now - it's just one of the steps needed to test your app.

More information on managing accounts and teams in Xcode can be found in this Apple documentation.

Preparing your Unity project for building to iOS

We now need to return to Unity and switch platforms so that we can build our game for iOS.

- Within Unity, open the Build Settings from the top menu (**File** > **Build Settings**).
- Highlight **iOS** from the list of platforms on the left and choose **Switch Platform** at the bottom of the window.



☑ Scenes/Main 0

iOS

Run in Xcode as	Release
Symlink Unity libraries	☐
Development Build	☐
Autoconnect Profiler	☐
Script Debugging	☐

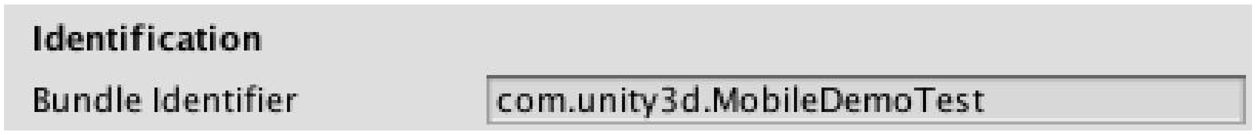
Switching platforms sets the build target of our current project to iOS. This means that when we build, Unity will create an Xcode project. Switching platforms also forces Unity to reimport all assets in the project. This doesn't take long on a small project like this, but be aware that on a larger project this may take some time.

Next, we need to enter the bundle identifier for our game. A bundle identifier is a string that identifies an app. It’s written in what is known as reverse-DNS style, following the format com.yourCompanyName.yourGameName. Allowed characters are alphanumeric characters, periods and hyphens. We need to change the bundle identifier from the default setting in order to build.

It is important to note that at the time of writing, once you have registered a bundle identifier to a Personal Team in Xcode the same bundle identifier cannot be registered to another Apple Developer Program team in the future. This means that while you are testing your game using a free Apple ID and a Personal Team, you should choose a bundle identifier that is for testing only - you won’t be able to use the same bundle identifier to release the game. An easy way to do this is to add "Test" to the end of whatever bundle identifier you were going to use - for example, com.yourCompanyName.yourGameNameTest.

There are other restrictions on bundle identifiers. When you release an app, its bundle identifier must be unique to your app, and cannot be changed after your app has been submitted to the App Store. For more information on bundle identifiers on iOS, see this Apple documentation.

- Open the Player Settings in the Inspector panel (**Edit** > **Project Settings** > **Player**).
- Expand the section at the bottom called Other Settings, and enter your chosen bundle identifier where it says Bundle identifier.


Identification

Bundle Identifier com.unity3d.MobileDemoTest

Now we’re ready to build!

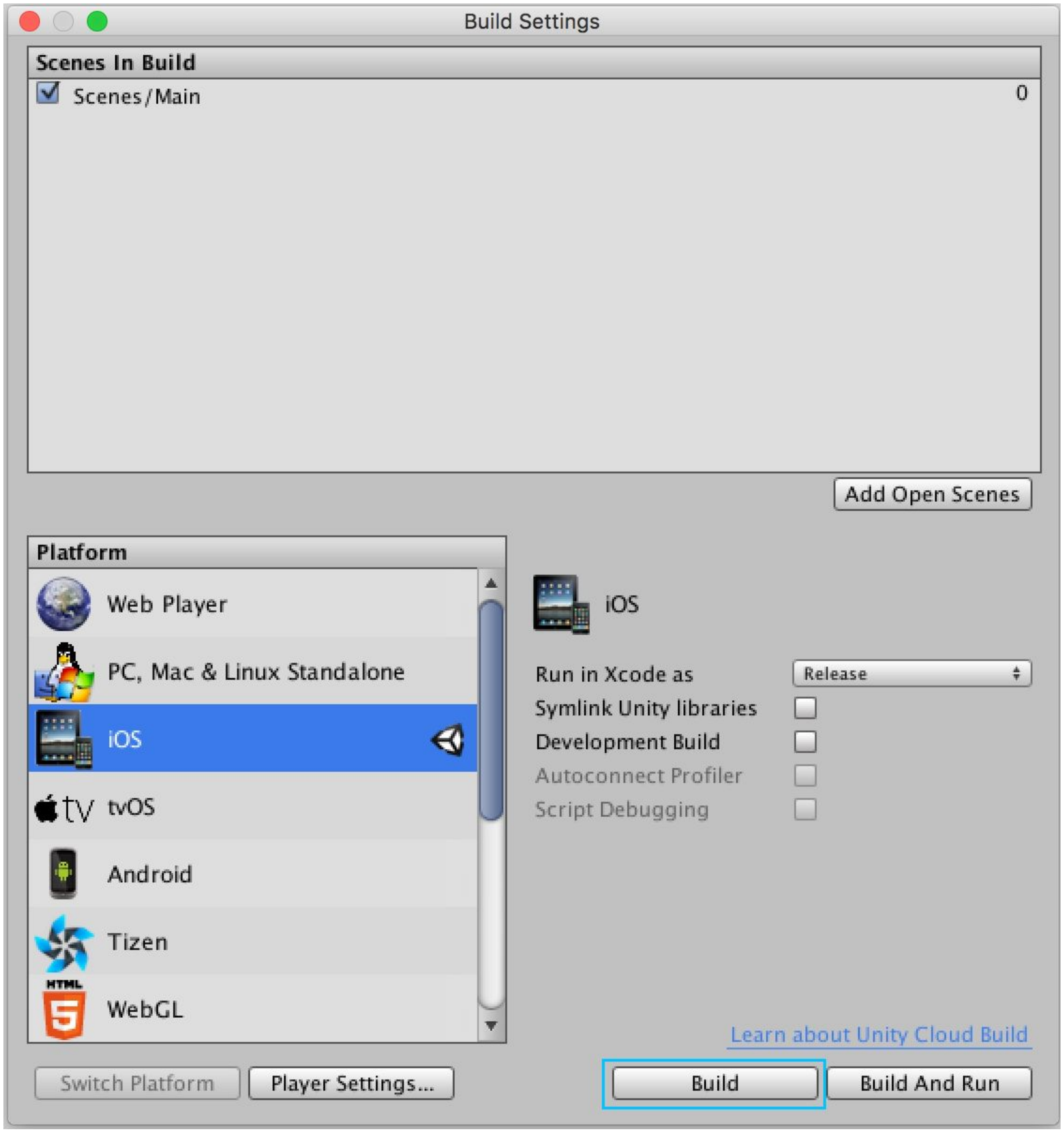
Building an Xcode project using Unity

Building your game to an iOS device involves two steps. First, Unity builds an Xcode project. Then, Xcode builds that project to your device. Once you’ve set everything up, Unity can trigger both of these steps for you - however, the first time you build a project to your device involves a little extra work and you must do both steps separately.

The reason for the extra work is security. Apple uses a technique called code signing to ensure that apps come from known sources and haven’t been tampered with, and this needs to be set up before you can build. For more information on code signing, see this Apple documentation.

First, we’re going to get Unity to build the Xcode project.

- Open the Build Settings from the top menu (**File** > **Build Settings**).
- Click **Add Open Scenes** to add the Main scene to the list of scenes that will be built.
- Click **Build** to build.


Build Settings

Scenes In Build

☑ Scenes/Main 0

Platform: iOS (selected)

Run in Xcode as: Release

Symlink Unity libraries: ☐

Development Build: ☐

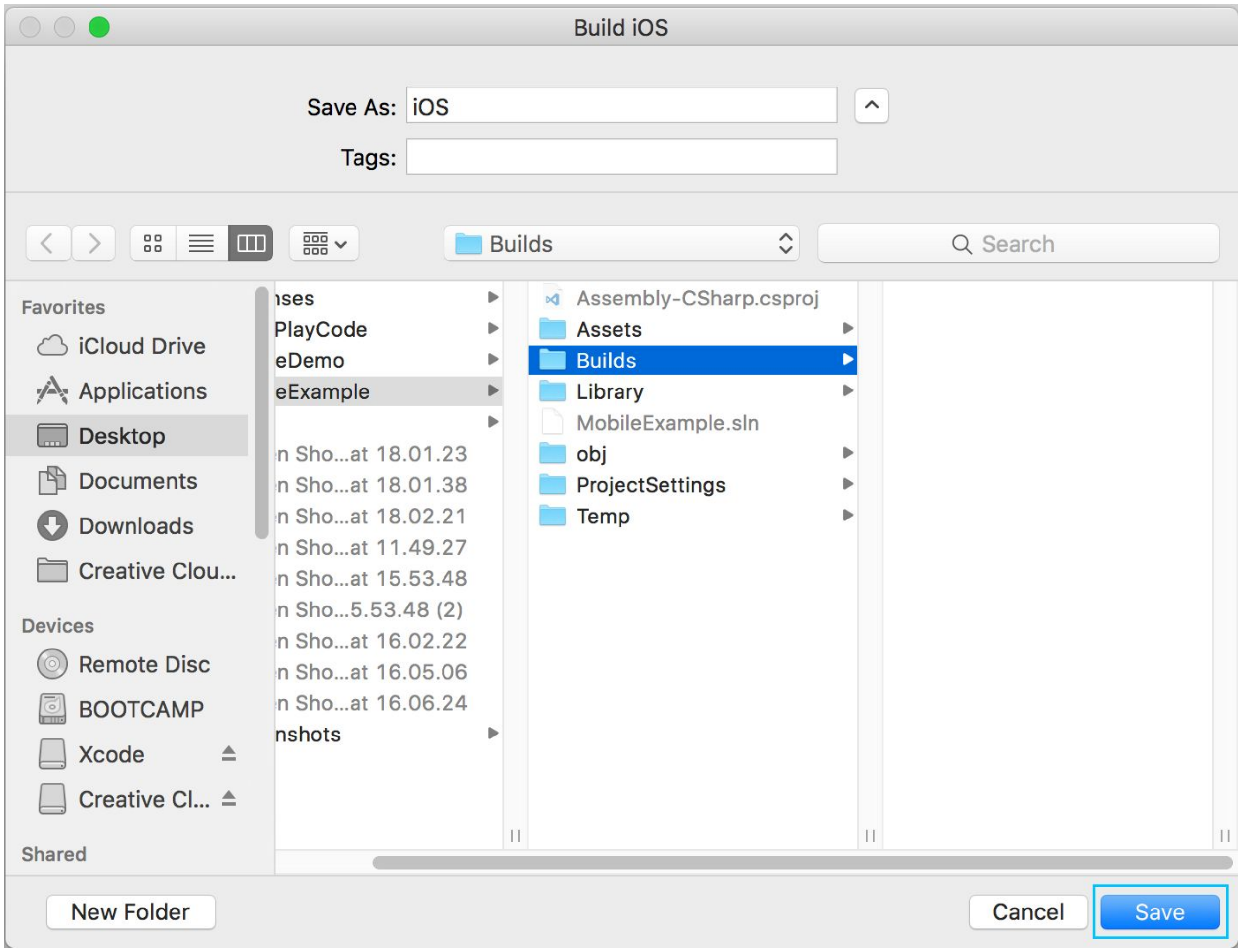
Autoconnect Profiler: ☐

Script Debugging: ☐

You will be prompted to choose where to build your Xcode project. A good place to do this is in a dedicated builds folder within your project folder. We'll make one now.

- Click the down arrow in the top right of the prompt to expand it, and then click **New Folder**.

- When prompted to choose a name, enter "Builds" and click **Create**. This will create a new folder called “Builds” in the root directory for your project.
- In the text input field marked Save As, enter "iOS" and click **Save**.



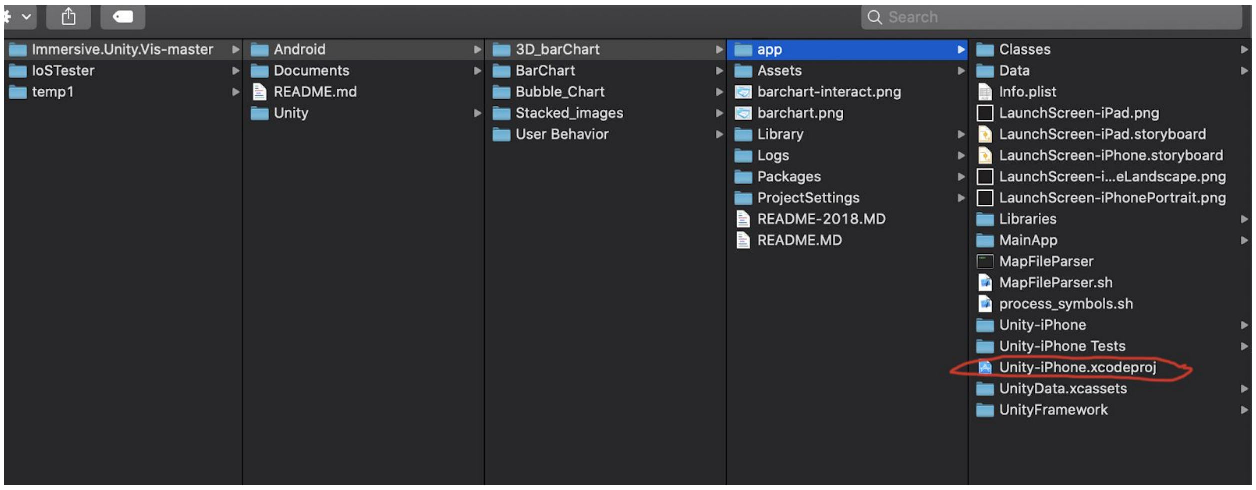
Unity will now create an Xcode project called "iOS" in the “Builds” folder.

An Xcode project is all the files and information required to build an app using Xcode, organised into folders containing code, image assets and so on. For more information on the Xcode project that Unity creates, see the Unity Manual Xcode Project Structure page.

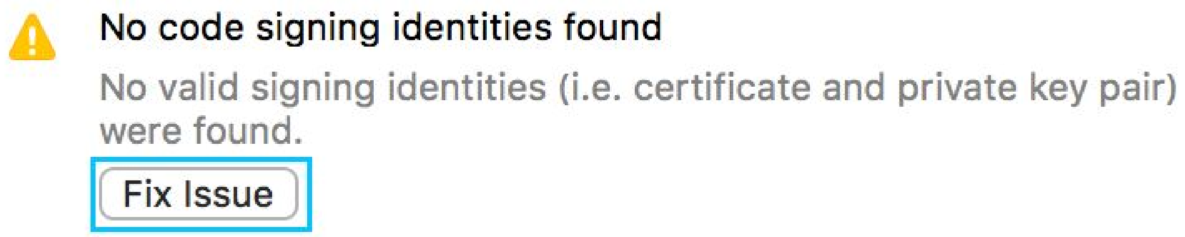
Building the sample project to your device using Xcode

Once Unity has built the Xcode project, a Finder window will open at the project's location.

- Double click the .xcodeproj file to open the project with Xcode.



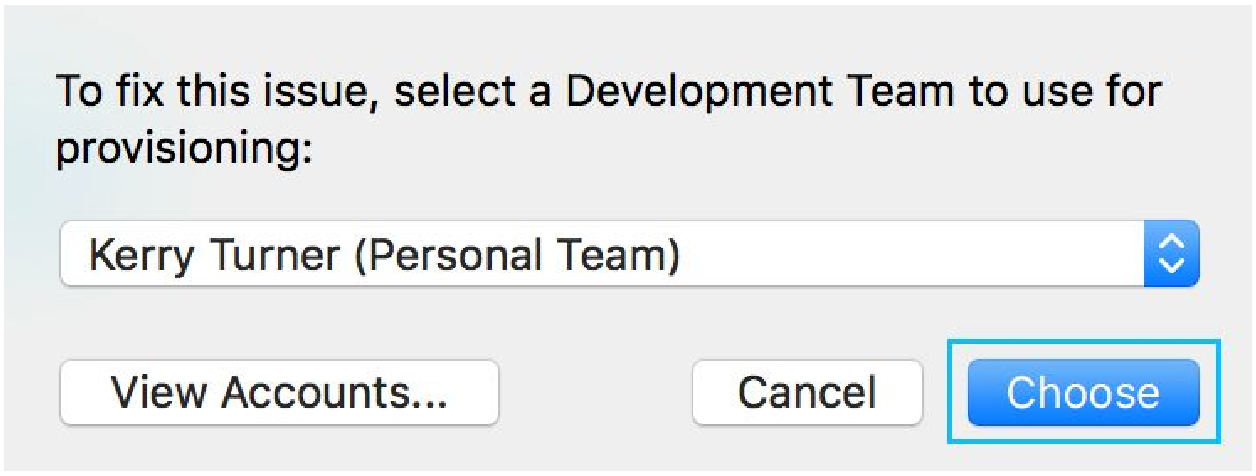
- In the top left, select **Unity-iPhone** to view the project settings. It will open with the **General** tab selected.
- Under the topmost section called **Identity**, you may see a warning and a button that says **Fix Issue**. This warning doesn't mean we've done anything wrong - it just means that Xcode needs to download or create some files for code signing.
- Click the **Fix Issue** button.



No code signing identities found

No valid signing identities (i.e. certificate and private key pair) were found.

- A popup will appear, showing details of any teams that have been added to Xcode.
- Make sure that the correct team is shown in the dropdown - if you're using a free Apple ID, it should be your name followed by "(Personal Team)".
- Click **Choose** to instruct Xcode to download any required certificates and generate a provisioning profile. The warning will then disappear.



To fix this issue, select a Development Team to use for provisioning:

Kerry Turner (Personal Team)

Certificates and provisioning profiles the files required for code signing. You don't need to worry about what they do at the moment - but if you'd like to know more about them, see this Apple documentation.

Now connect your device to your computer using a USB cable. If it's the first time you've connected this device, you may see a message that says that Xcode is "processing symbol files" - this means that Xcode is getting information from the device that will allow you debug apps on this device. Wait for this to complete. Once it has finished processing the symbol files, the message will disappear and your device will be ready to use.

The final step before we build to the device is to make sure that the device is unlocked, because Xcode can't launch apps on a device that is locked with a passcode. If your device is set to lock with a passcode, it's best to change this setting before you build to your device and then change it back after you have finished testing.

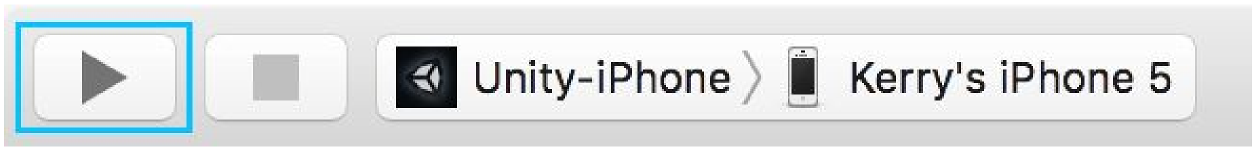
- On your device, go to **Settings** > **Display & Brightness** > **Auto-Lock**.
- To disable locking, select **Never**.

It is worth noting that when in Low Power Mode, the Auto-Lock settings cannot be changed until Low Power Mode is turned off.

- To turn off Low Power Mode, go to **Settings** > **Battery** > **Low Power Mode** and set this to "Off".

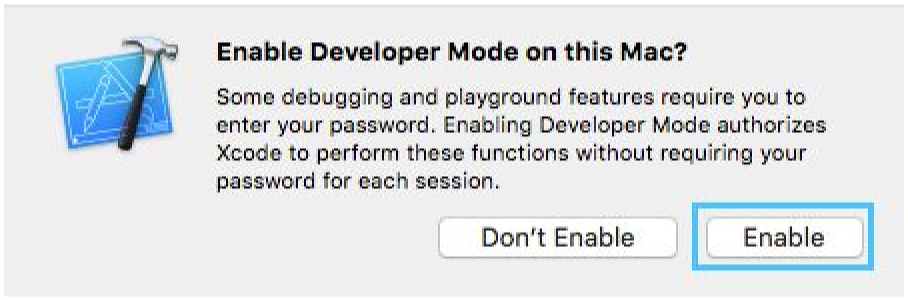
Now it's time to build to your device.

- In the top left of the Xcode interface, click **Run** (the "play" button).



If you haven't used your Mac to develop for iOS or OSX before, you may see a popup at this point asking if you would like to enable Developer Mode. Enabling Developer Mode will mean that you won’t be prompted for your password when carrying out common development tasks.

- Enable Developer Mode by choosing **Enable**, and enter your password when prompted.



Enable Developer Mode on this Mac?

Some debugging and playground features require you to enter your password. Enabling Developer Mode authorizes Xcode to perform these functions without requiring your password for each session.

After a moment, you’ll see a message in the bar at the top centre of Xcode that says "Build succeeded", and the app will load on your device. We’re almost done!

You may receive a warning prompt on the device with the title "Untrusted Developer", or a popup in Xcode that says “Could not launch [your app name]”. If you see either of these, it means the that there’s one last step left: you need to set your device to trust your Apple ID.

- On your device, go to **Settings** > **General** > **Device Management** > **Developer App** > [your app name].
- Choose your Apple ID, and then choose **Trust**.

As long as you have at least one app built using that Apple ID on your device, your device will allow content built with that Apple ID to run. If you ever remove all of the apps built with that Apple ID from your device, you’ll need to go back to this setting and choose to trust it again.

Testing the game on your iOS device

The game has now been built to your device. If you disconnect your phone from your computer, the game will still be there. To play the game on your device, tap its icon on the home screen - the same way you would launch any app on the device.

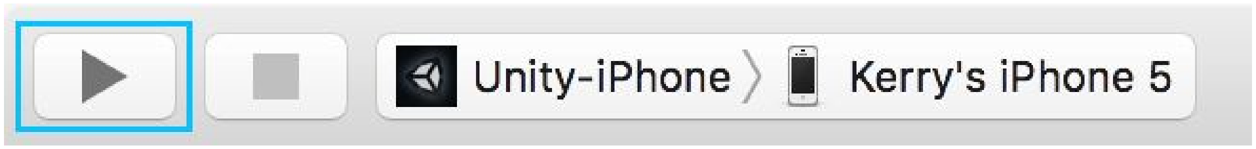
Our game works! The cube spins, and tapping it causes it to change direction.

Simply playing your game on a device is one way of testing it. It’s a good way of checking things like whether the controls work. However, if you want more information - if you’d like to see Unity’s logs while you play, for example - you’ll need to build and run the app while the device is still connected to Xcode.

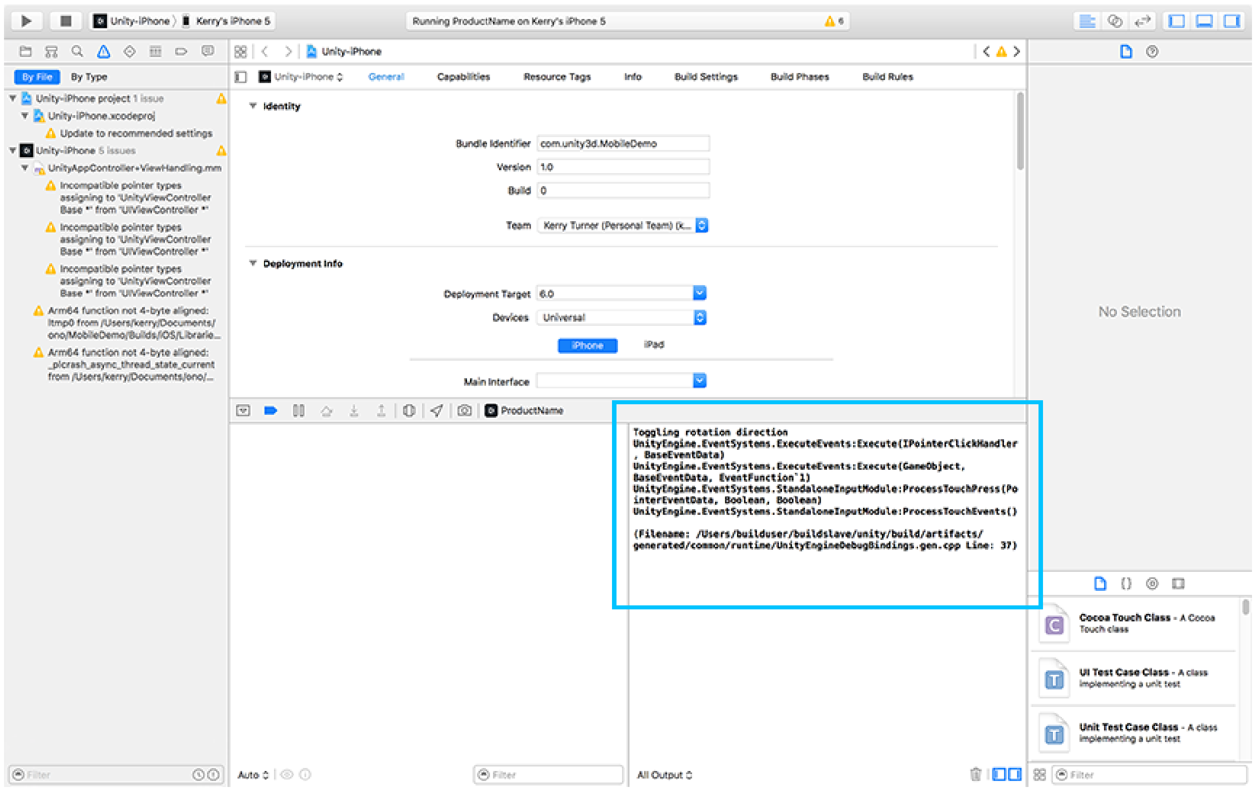
Let’s do this now.

- Connect your device to your computer.
- Open the Xcode project by double clicking the .xcodeproj icon, as before.
- In Xcode, choose **Run** (the "play" button).

Xcode will now build the game to your device and begin a debugging session.



If you look to the bottom right of Xcode while your game is running, you’ll see the Xcode debug console. This contains all of the information that you’d see in the Unity console, along with debug information about other things that are happening in the OS. You can use this in the same way that you use the console within Unity, to help you debug your game.



Unity-iPhone project 1 issue
Unity-iPhone.xcodeproj
Update to recommended settings
Unity-iPhone 5 issues
UnityAppController+ViewHandling.mm
Incompatible pointer types assigning to 'UnityViewController Base *' from 'UIViewController *'
Incompatible pointer types assigning to 'UnityViewController Base *' from 'UIViewController *'
Incompatible pointer types assigning to 'UnityViewController Base *' from 'UIViewController *'
Arm64 function not 4-byte aligned: ltmp0 from /Users/kerry/Documents/ono/MobileDemo/Builds/iOS/Librarie...
Arm64 function not 4-byte aligned: _plcrash_async_thread_state_current from /Users/kerry/Documents/ono/...

Identity

Bundle Identifier: com.unity3d.MobileDemo
Version: 1.0
Build: 0
Team: Kerry Turner (Personal Team) (k...

Deployment Info

Deployment Target: 6.0
Devices: Universal
Main Interface:

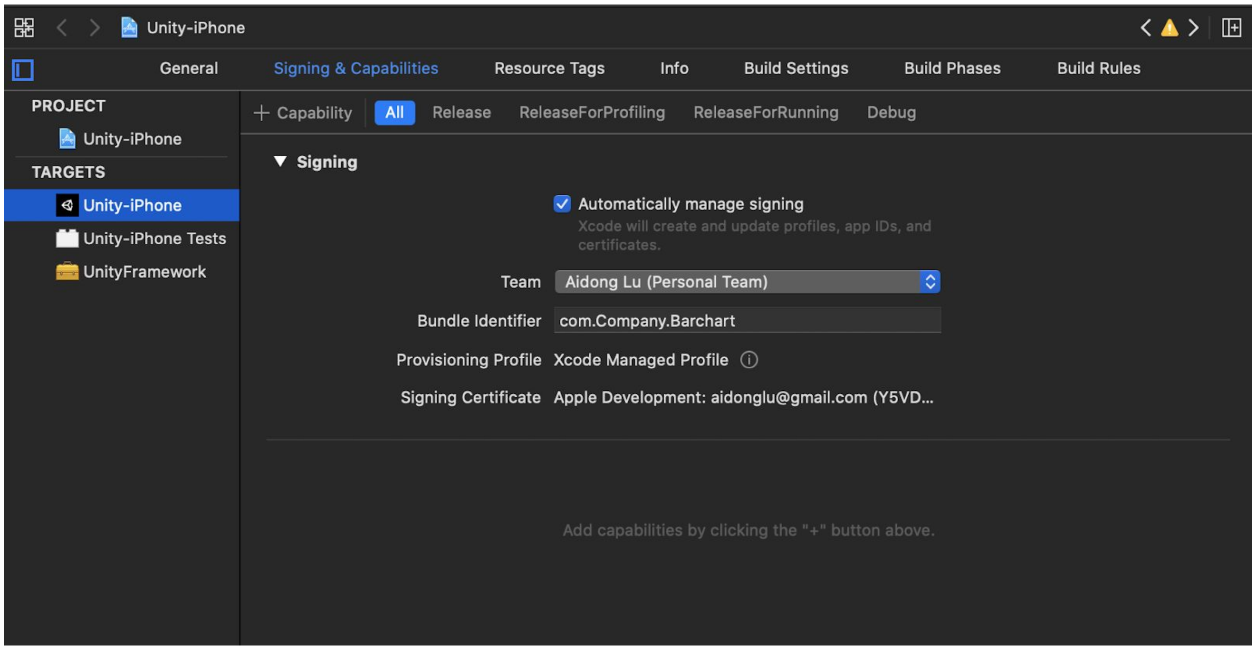
```
Toggling rotation direction
UnityEngine.EventSystems.ExecuteEvents:Execute(IPointerClickHandler
, BaseEventData)
UnityEngine.EventSystems.ExecuteEvents:Execute(GameObject,
BaseEventData, EventFunction`1)
UnityEngine.EventSystems.StandaloneInputModule:ProcessTouchPress(Po
interEventData, Boolean, Boolean)
UnityEngine.EventSystems.StandaloneInputModule:ProcessTouchEvents()

(Filename: /Users/builduser/buildslave/unity/build/artifacts/
generated/common/runtime/UnityEngineDebugBindings.gen.cpp Line: 37)
```

Try tapping the cube and looking at the debug console. You will see the same log message as you would see in Unity, along with some additional information.

Using the console to get information is just one of the ways that Xcode can help you test, profile and debug your game. It's a complex subject and we won't cover all of the options here, but this post on the Unity blog about profiling is a great start.

The most common problem is about the signing & capabilities. As the snapshots below show, choose your personal team with free registration and make sure the Bundle Identifier is the same on both tabs of "general" and "signing & capabilities".

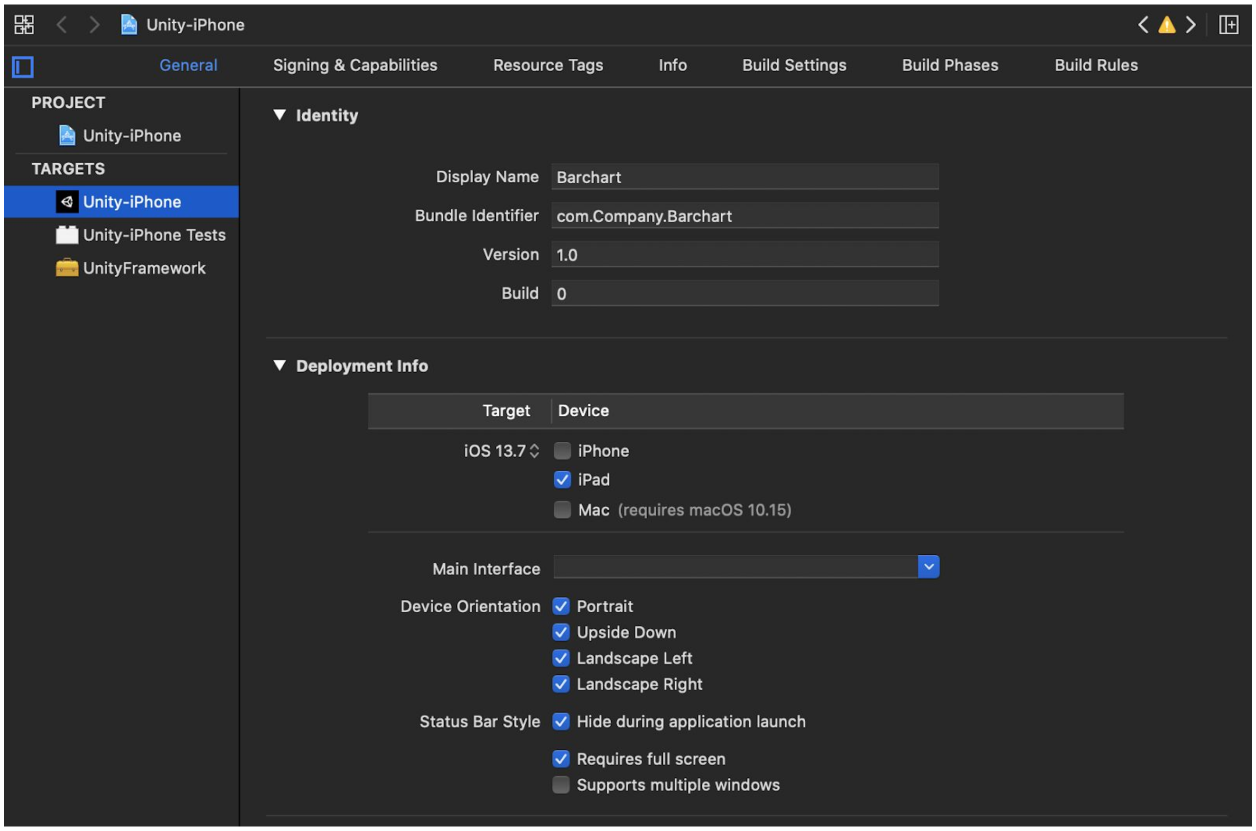


Signing

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

Team	Aidong Lu (Personal Team)
Bundle Identifier	com.Company.Barchart
Provisioning Profile	Xcode Managed Profile
Signing Certificate	Apple Development: aidonglu@gmail.com (Y5VD...

Add capabilities by clicking the "+" button above.



Identity

Display Name	Barchart
Bundle Identifier	com.Company.Barchart
Version	1.0
Build	0

Deployment Info

Target	Device
iOS 13.7	☐ iPhone
	☑ iPad
	☐ Mac (requires macOS 10.15)

Main Interface

Device Orientation: ☑ Portrait, ☑ Upside Down, ☑ Landscape Left, ☑ Landscape Right

Status Bar Style: ☑ Hide during application launch

☑ Requires full screen
☐ Supports multiple windows

If your build was successful but could not run with an error message “[device name] could not be found. Please reconnect your device”, check the versions of your Xcode, the iOS version on your device, and the “deployment info” on the figure above (I chose iOS 13.7 for iPad specifically and you adjust for your device).

How to Trust "Untrusted Enterprise Developer" All iPhones/iPads/iPods: https://www.youtube.com/watch?v=7ejkoLgoPGk

Conclusion

In this lesson, we learned how to build a Unity game to an iOS device for testing. We learned how to switch build targets in Unity, what a bundle identifier is, how to use Xcode to build your Unity game to your device for testing, and how to debug your Unity game on your iOS device using the console in Xcode.

Questions?

It is very likely that you will encounter some issues during this deploy process. It is common and don’t panic! First, try to search for solutions to the problems. Second, you can always contact us. Third, let’s share problems and solutions in the class.