

INSTRUCTIONS 1.0 beta

Welcome to the open source section of MoMAR!

Our goal is to democratize physical exhibition spaces, museums, and the curation of art within them. If you like to create your own gallery or show your own work in any museum around the world – this will help you.

All you need is:

Unity

<https://store.unity.com/>

Unity is a cross-platform game engine developed by Unity Technologies, which is primarily used to develop both three-dimensional and two-dimensional video games and simulations for computers, consoles, and mobile devices. It's a simple – yet powerful – platform to create AR experiences. The personal plan is for free. Usually we don't need more than that plan.

Vuforia

<https://vuforia.com/>

Vuforia is an Augmented Reality Software Development Kit (SDK) for mobile devices that enables the creation of Augmented Reality applications. It uses Computer Vision technology to recognize and track planar images (Image Targets) and simple 3D objects, such as boxes, in real-time. The free plan is guess what for free and also here we don't need more.

APPLE iOS or Android developer account

<https://developer.apple.com/>

<https://developer.android.com/>

When you want to distribute your gallery or art to more people than you should create an app. We'll show you how to create a simple app. For publishing you app you need a developer account. The Google Play store has a one-time fee of \$25. The Apple fee is \$99/year.

Setting up Unity & Vuforia

1. Installing Unity

Download and install the Unity Editor from the [Unity download page](#). The installer uses a Download Assistant and has detailed instructions to follow.

More detailed info here: <https://docs.unity3d.com/Manual/InstallingUnity.html>

2. Starting Unity for the first time

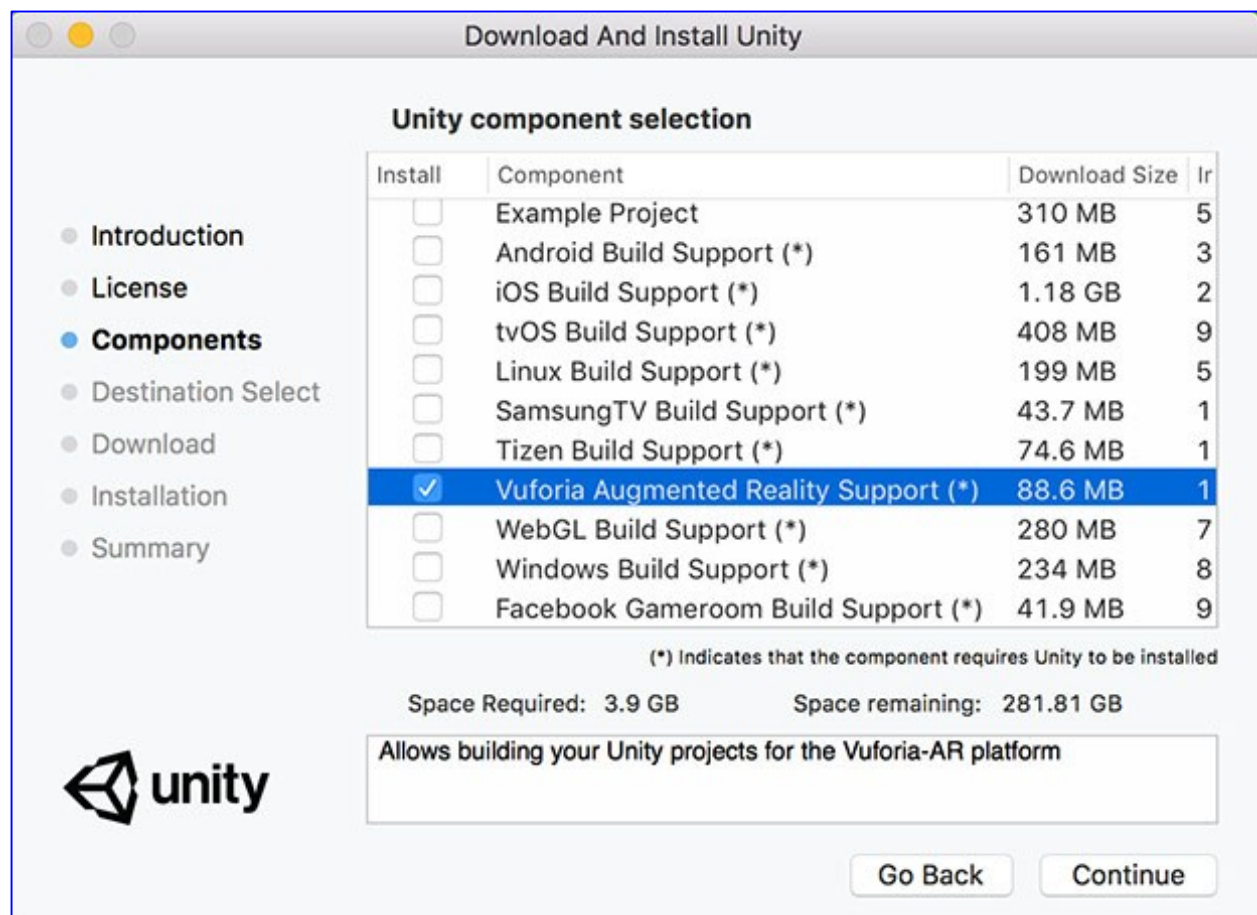
When you launch the Unity Editor, the Home Screen appears. If you have no existing Unity Projects on your computer, or if this is your first time opening the Unity Editor, the Home Screen displays the [Learn](#) tab. From the Learn tab, you can access tutorials and learning resources to help you get started with Unity. If you are new to using Unity, you should work through the Unity Learn tutorials before starting a new project.

You also find a lot of great manuals about working in Unity if you like to dive more into it:
<https://docs.unity3d.com/Manual/UnityOverview.html>

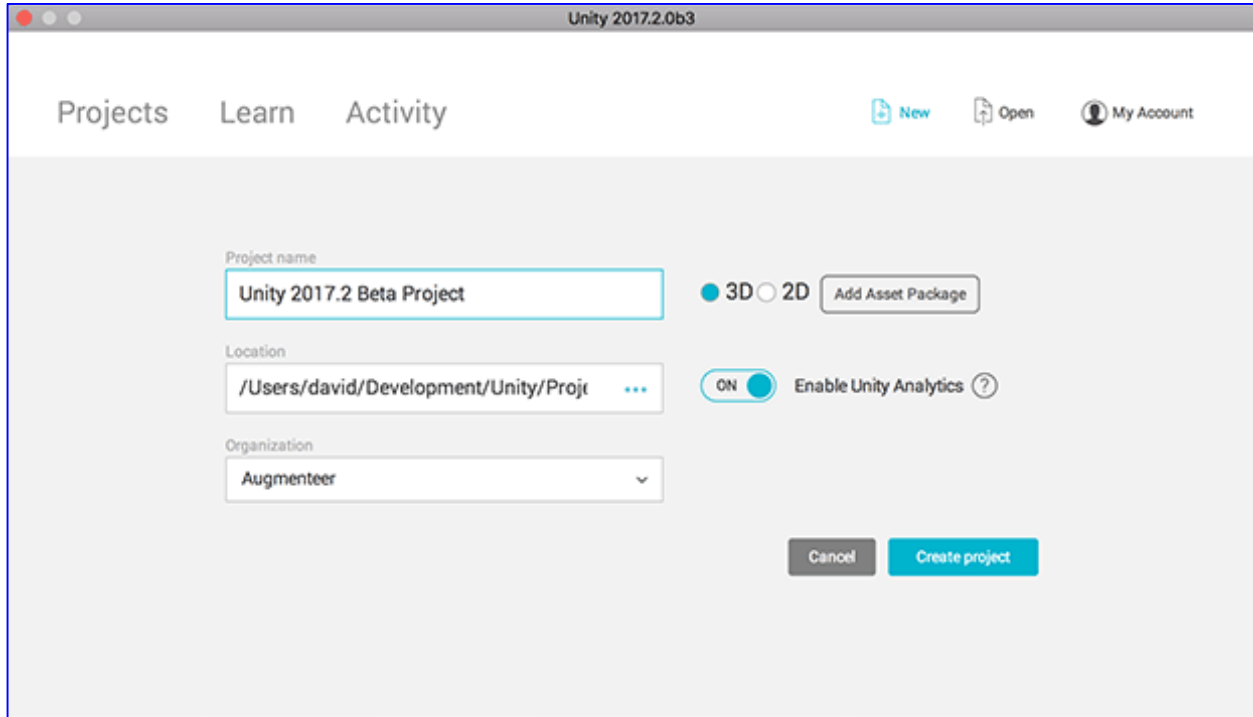
3. Installing Vuforia

Download and run the Unity Download Assistant 2017.2 or later from the Unity website: <https://unity3d.com/>. Accept Unity's license agreements.

In addition to your platform support (iOS, Android, UWP), select **Vuforia Augmented Reality Support** in the Components selection dialog. Then continue with your installation.



4. Create a new Unity project



The screenshot shows the 'New Project' dialog box in the Unity 2017.2.0b3 interface. The window has a title bar with standard macOS window controls and the text 'Unity 2017.2.0b3'. Below the title bar is a navigation bar with 'Projects', 'Learn', and 'Activity' tabs. On the right side of the navigation bar are three icons: a plus sign for 'New', a document icon for 'Open', and a user icon for 'My Account'. The main area of the dialog contains three input fields on the left and two toggle options on the right. The 'Project name' field is labeled 'Project name' and contains the text 'Unity 2017.2 Beta Project'. The 'Location' field is labeled 'Location' and contains the text '/Users/david/Development/Unity/Proje' followed by a three-dot menu icon. The 'Organization' field is labeled 'Organization' and contains the text 'Augmenteer' followed by a downward arrow icon. To the right of the 'Project name' field are two radio buttons for '3D' (selected) and '2D', and a button labeled 'Add Asset Package'. Below the 'Location' field is a toggle switch labeled 'ON' and a button labeled 'Enable Unity Analytics' with a question mark icon. At the bottom right of the dialog are two buttons: 'Cancel' and 'Create project'.

Unity 2017.2.0b3

Projects Learn Activity

New Open My Account

Project name

Unity 2017.2 Beta Project

3D 2D Add Asset Package

Location

/Users/david/Development/Unity/Proje ...

ON Enable Unity Analytics ?

Organization

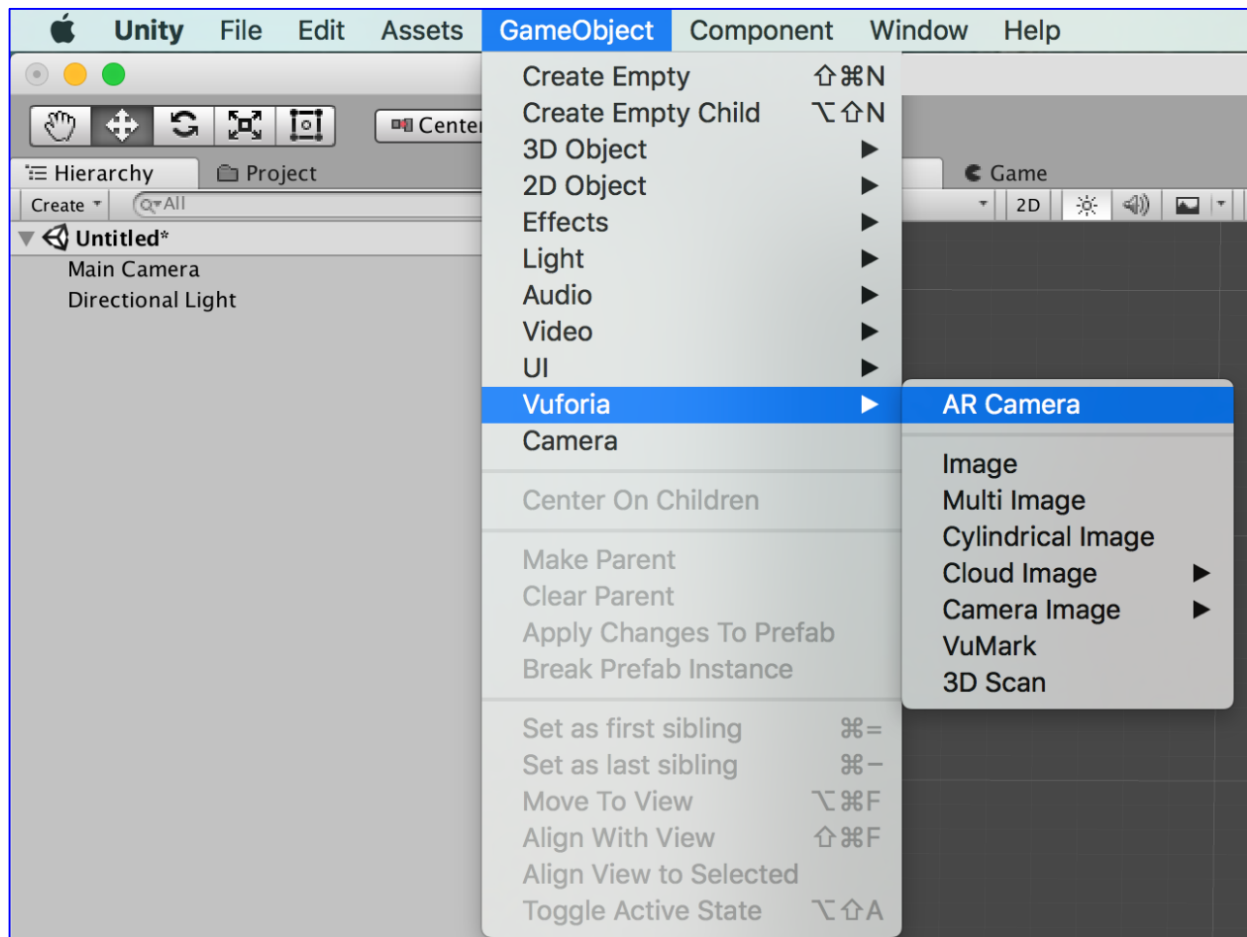
Augmenteer

Cancel Create project

It's recommended that you use a 3D project set-up.

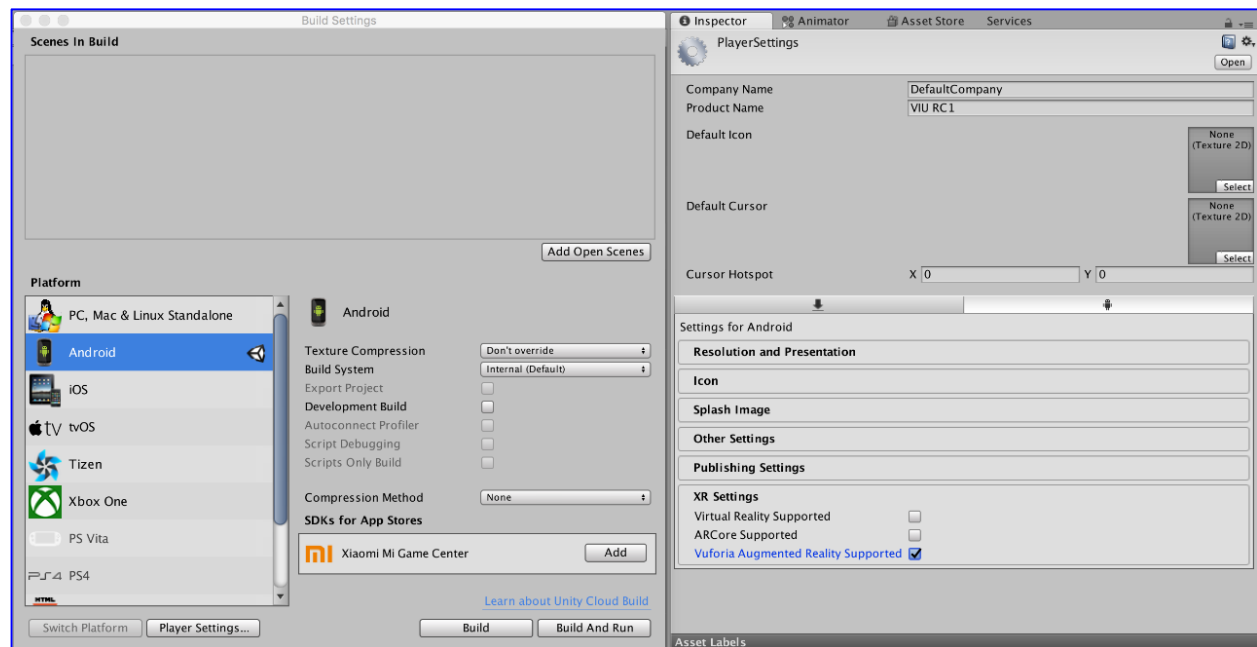
5. Vuforia Game Objects

Vuforia will be visible in the Unity Game Object menu and also in Build Settings and Player Settings.



6. Activate Vuforia in your project

You must activate Vuforia in your project before you will be able to build a Vuforia app, or use Vuforia in Play Mode. **Go to Player Settings to activate Vuforia under the "XR Settings" section**

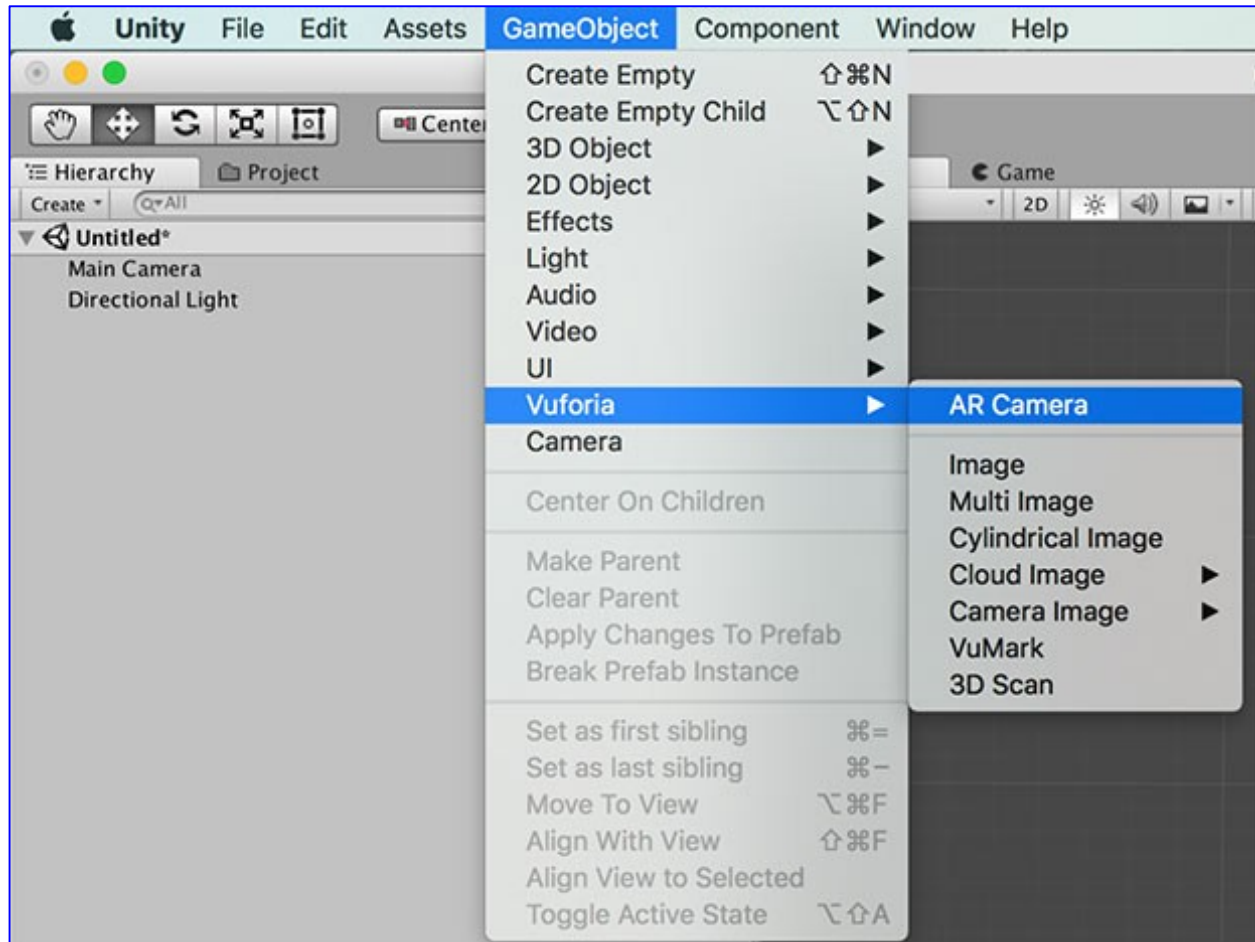


After activating Vuforia, you can add features of the Vuforia SDK to your project from the Unity GameObject Menu. The Vuforia SDK is to build Android, iOS, and UWP applications for mobile devices and digital eyewear. Depending if you want to build an iOS or Android app or both you choose those features.

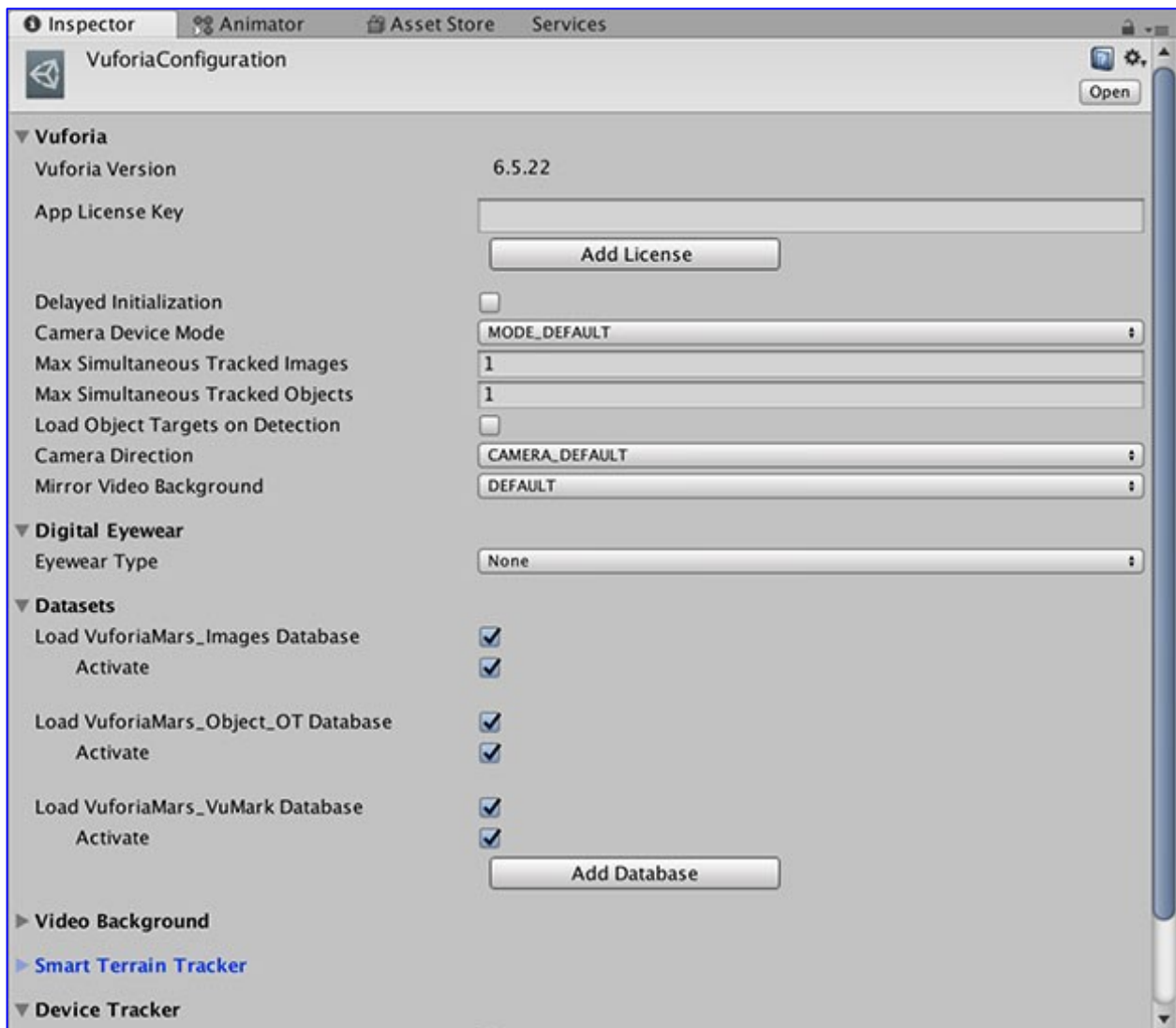
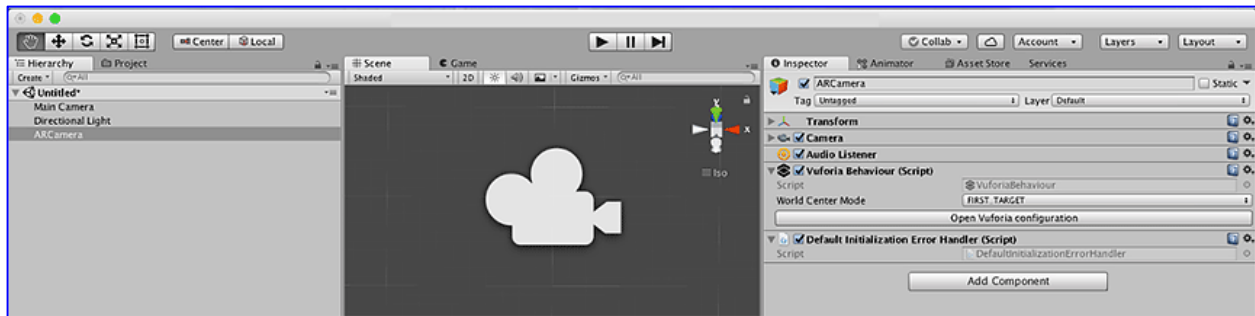
More info here: <https://developer.vuforia.com/downloads/sdk>

7. Add an ARCamera

This is a special camera type that supports augmented reality apps for both handheld devices.



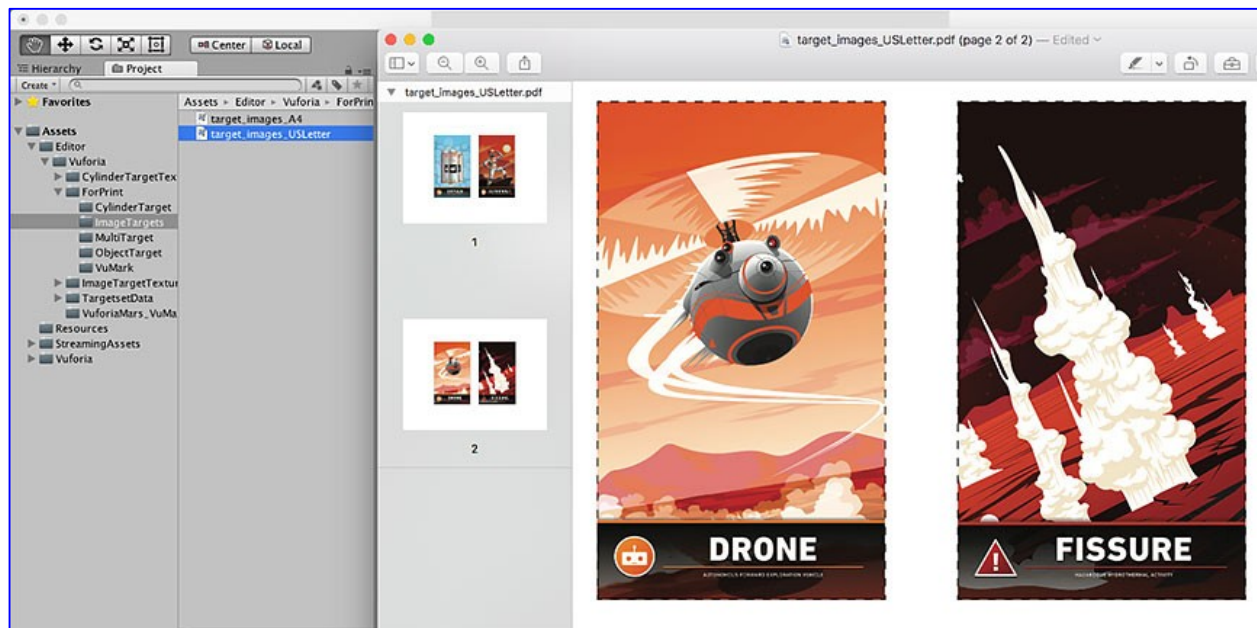
8. Open the global Vuforia Configuration Inspector



9. Activate the target databases

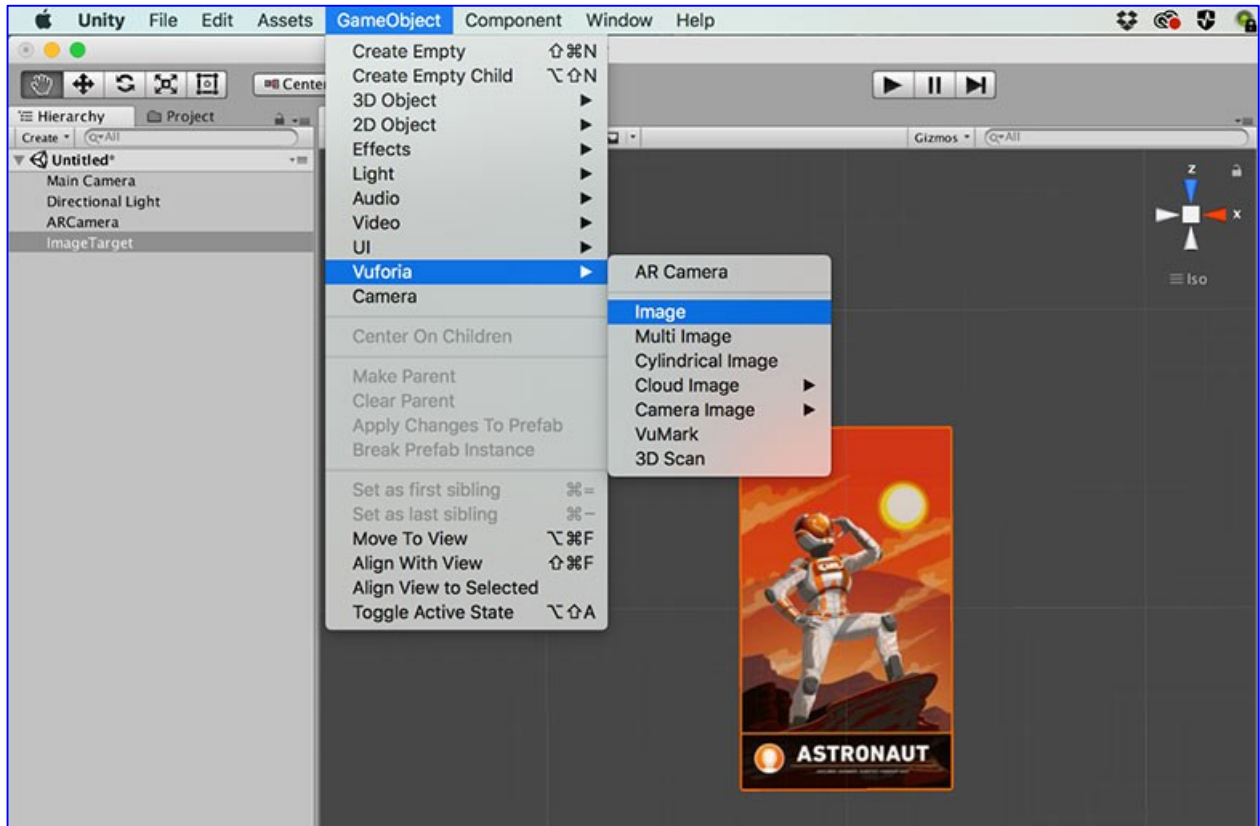
Load and Activate the target databases you want to use – we recommend loading and activating all of them if you're new to Vuforia and want to evaluate all of the target types.

Tip: This is a good time to print some targets to work with. You can find these in [/Editor/Vuforia/ForPrint](#). We've provided a complete set of high resolution target prints to get you started.



Printable target PDFs can be found in [/Editor/Vuforia/ForPrint](#)

10. Add Targets to your scene

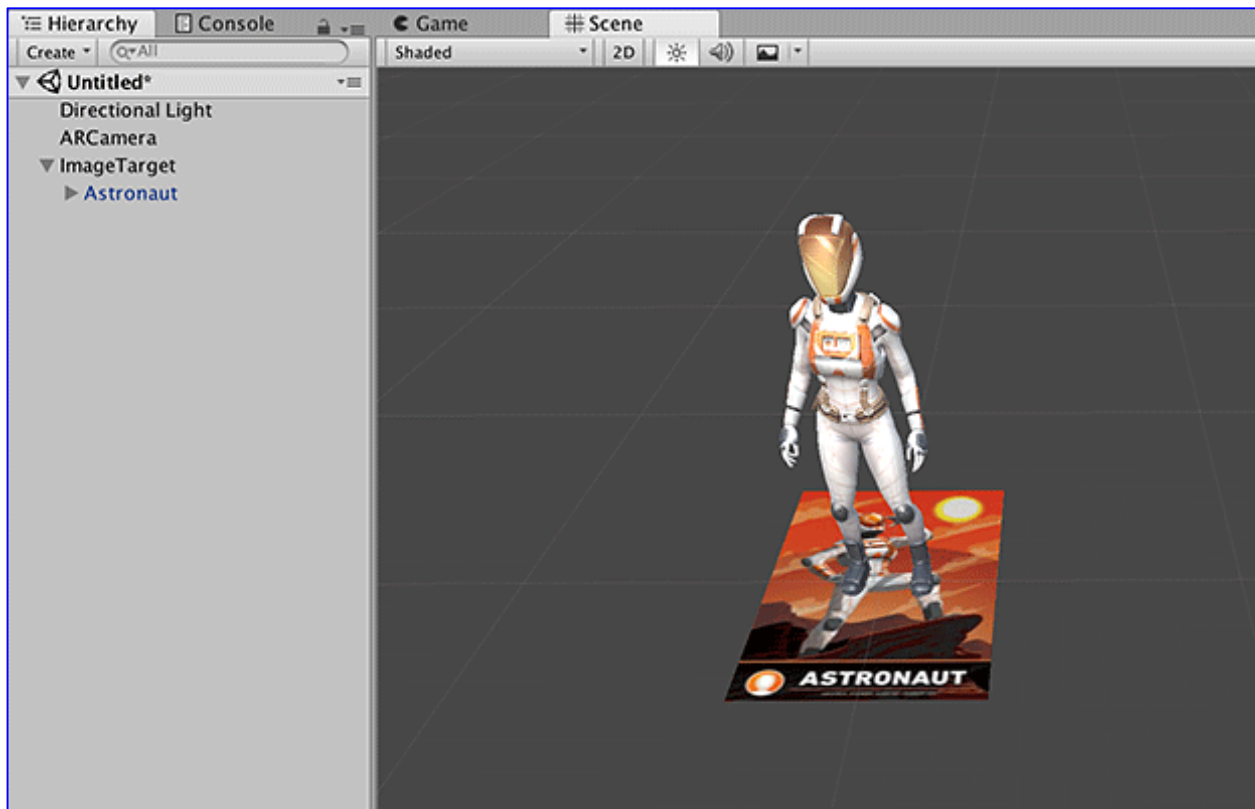


You can add Vuforia targets to your scene by selecting the associated Game Objects in the **GameObject>Vuforia** menu. A target Game Object will added in your scene hierarchy and be visible in your scene.

Each target object can be configured in the Inspector. Select the database and target name for the target you want to use.

11. Adding digital assets

You're now ready to add digital content to augment your target. You can do this by simply adding your assets as children of the target in the scene hierarchy. Parenting content with a target object automatically sets up the necessary rendering and physics behaviors (see: [DefaultTrackableEventHandler.cs](#)).



Add content as a child of the target.

Tip: Delete the default Main Camera after adding an ARCamera. The ARCamera contains its own scene Camera. You won't need the Main Camera unless you are using it to render a specific camera view.

12. Playing the scene

Vuforia provides a simulator in the Game view that you can activate by pressing the Play button. You can use this feature to evaluate and rapidly prototype your scene(s) without having to deploy to a device.

Play Mode is configured in the WebCam section of the Vuforia Configuration. Also [see Vuforia Play Mode for Unity](#).

13. Building and running your app

You can can build Vuforia Unity apps and run in the same way as other Unity apps for Android, iOS, and UWP. See: Building section: <https://unity3d.com/learn/tutorials/s/mobile-touch>

But we will focus here on using Unity.

For Android App start @ 1.12.1.1.

For Apple iOS App start @ 1.12.2.1.

ANDROID APP

14.1.1. Setting up the Android SDK Tools

The first thing we need to install is the Java Development Kit (known as the JDK).

-> Go to the Java site to download the most recent JDK. It's labelled as "Java Platform (JDK)". Choose the one with the highest version number.

-> Simply run the installer and follow instructions in the wizard to install it.

Next, we need to install the Android SDK Tools.

-> Go to the Android Developer site.

-> Download the Android SDK Tools (also referred to on the site as "the command line tools"), rather than the full download of Android Studio.

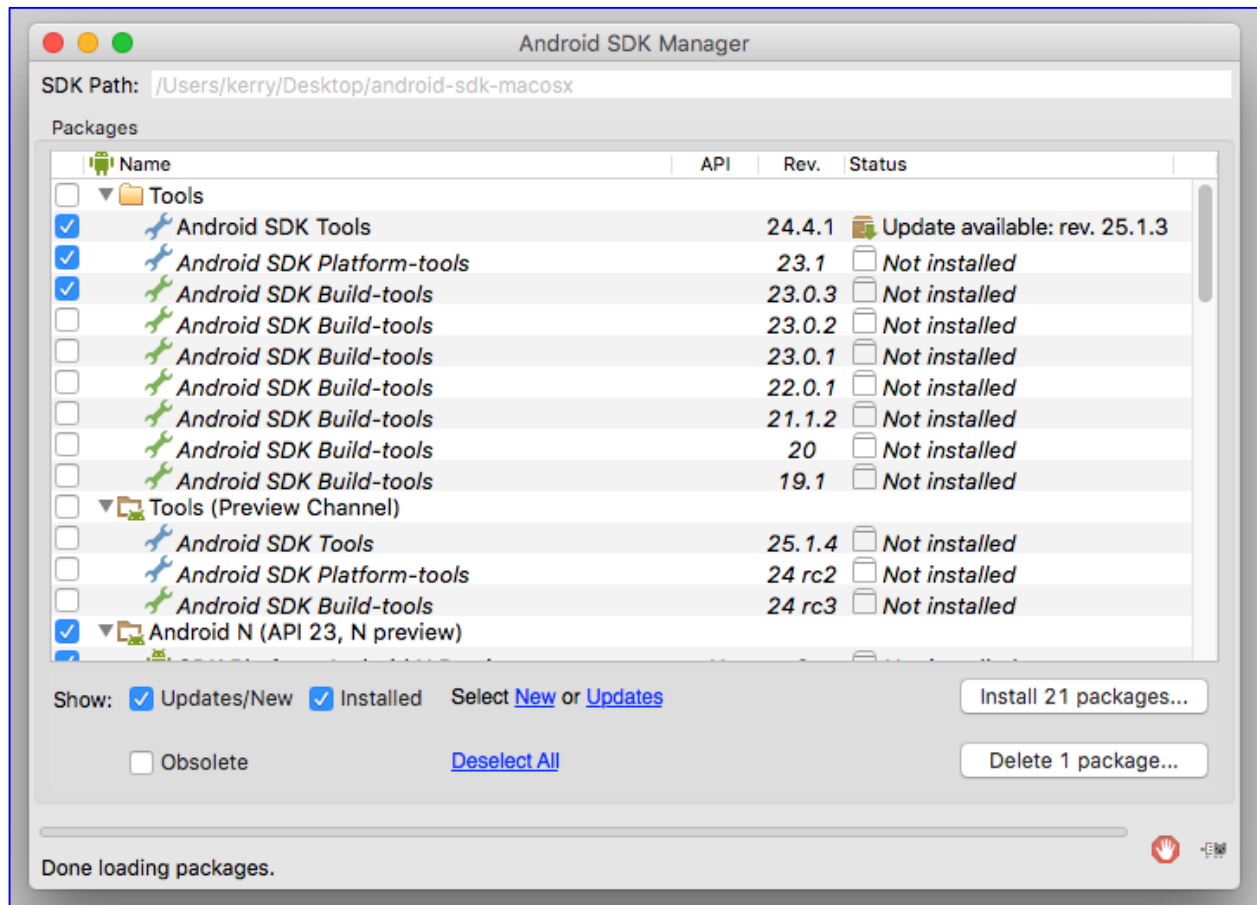
-> Unzip the downloaded file. The resulting directory is the directory that contains the Android SDK Tools.

-> Put the directory in a memorable, accessible location - you'll need to tell Unity where this is directory later.

-> Open the directory that contains the Android SDK Tools, and navigate to tools.

-> Double click the file called android to run it.

A popup will appear, showing a list of packages that can be installed. By default, the core packages for building and the package for the latest version of the Android OS are selected.

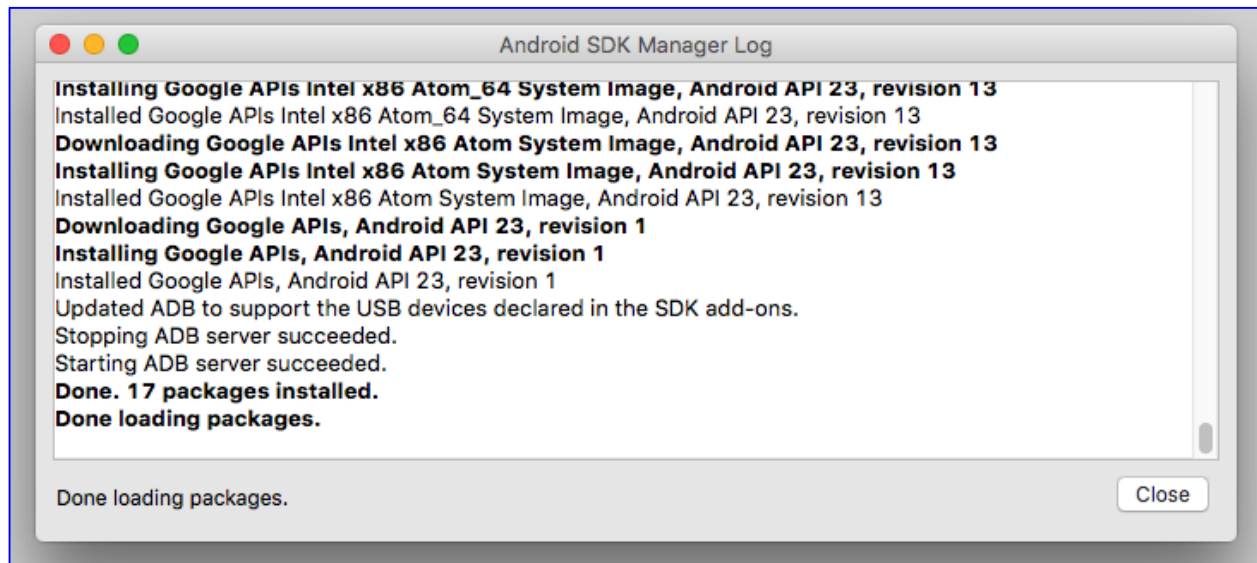


If the device you're building to runs the latest version of Android, you don't need to select anything else. If the device you're building to runs an earlier version of Android, scroll down the list of packages and select the version you need.

-> Click **Install [x] packages** to start the installation process. You will be prompted to accept the licenses for these packages.

-> For each package that you wish to install, select the license on the left, select **Accept License**, and then click **Install** to proceed.

A window will appear that shows the progress of the installation. The installation will take some time to complete, so it's a good time to take a break.



-> You may see a popup telling advising you to restart the manager window - if this happens, click **OK**.

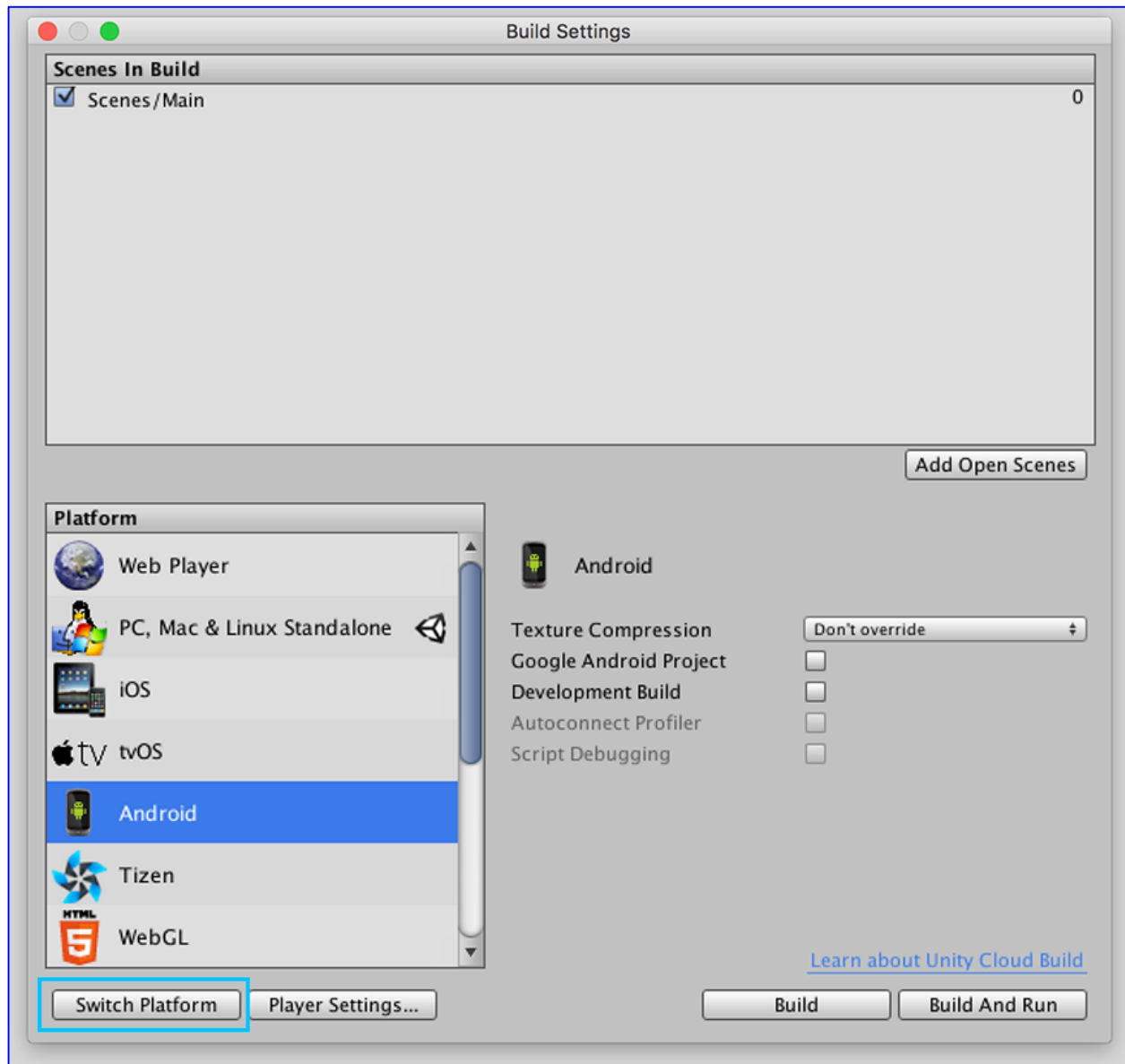
-> When the installation process has completed, close the window.

14.1.2. Preparing your Unity project for building to Android

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

-> Within Unity, open the **Build Settings** from the top menu (**File > Build Settings**).

-> Highlight **Android** from the list of platforms on the left and choose **Switch Platform** at the bottom of the window.



Switching platforms sets the build target of our current project to Android. This means that when we build, Unity will create an .apk file. 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.

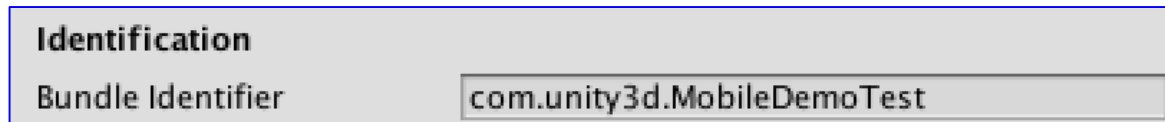
Next, we need to enter the bundle identifier for our app. A bundle identifier is a string that identifies an app. It's written in what is known as reverse-DNS style, following the format `com.yourCompanyName.yourappName`. Allowed characters are alphanumeric characters, periods and hyphens. We need to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

There are 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 published to the Google Play Store.

When we build our app for Android, the bundle identifier becomes what is known as the app's package name. For more information on package names, see [this Android 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**:

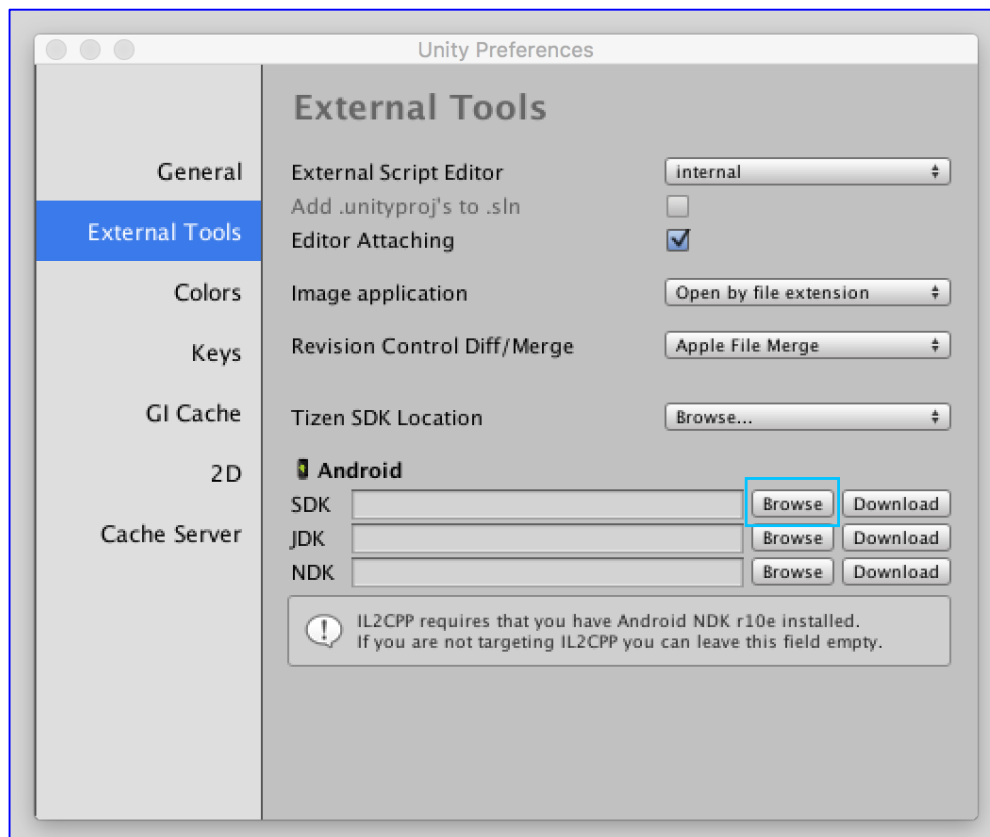


Finally, we need to tell Unity where we installed the Android SDK Tools.

-> Using the top menu, navigate to **Unity > Preferences** (on OSX) or **Edit > Preferences** (on Windows).

-> When the Preferences window opens, navigate to **External Tools**.

-> Where it says **Android SDK Location**, click **Browse**, navigate to where you put the directory containing Android SDK Tools and click **Choose**.



14.1.3. Preparing your Android device

Next, we need to enable developer mode on our Android devices - this gives you access to various options related to building and debugging, and lets you test the app on your device.

-> On your Android device, navigate to **Settings > About phone** or **Settings > About tablet**.

-> Scroll to the bottom and then tap **Build number** seven times. A popup will appear, confirming that you are now a developer.

-> Now navigate to **Settings > Developer options > Debugging** and enable **USB debugging**.

14.1.4. Building an Android project using Unity

Now we're ready to build!

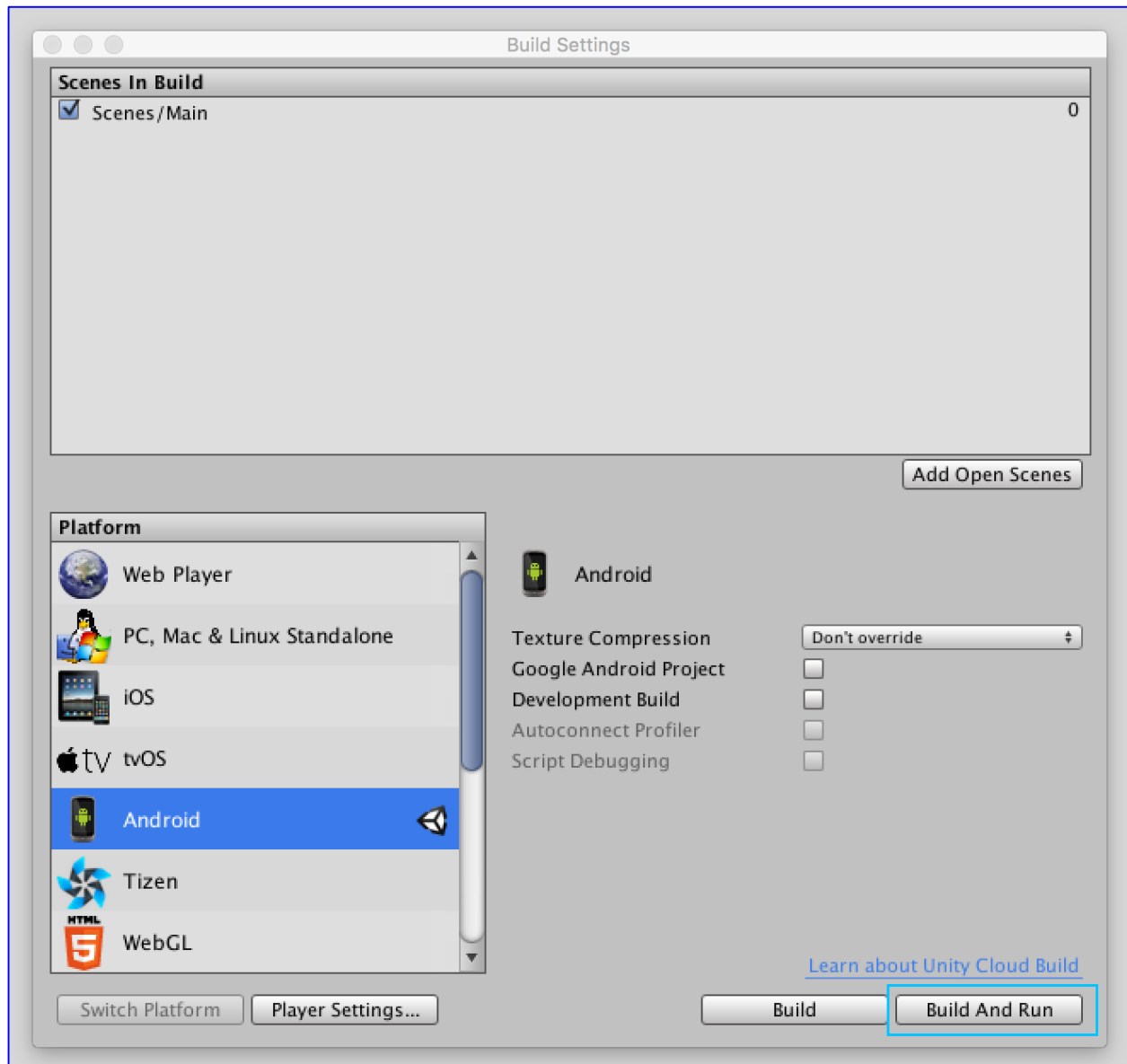
-> Connect your Android device to your computer using a USB cable.

-> You may see a prompt asking you to confirm that you wish to enable USB debugging on the device. If so, click **OK**.

-> In Unity, 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 And Run**.



You will be prompted to choose where to build your .apk. A good place to do this is in a dedicated builds directory in the root of your project.

-> **Create a directory called "Builds" in the root of your project (not within your Assets directory, but at the same level as it).**

-> **Select that directory as the location to build to.**

-> **In the text input field marked Save As, enter "Android" and click Save.**

Unity will now create an .apk file called "Android" in the "Builds" folder, build that file to your device and run it.

14.1.5. Upload your apk to Google play and publish your app

Google spent already a lot of time in creating a documentation 'how to publish an app' and it's really straight forward. Please check it out here: <https://support.google.com/googleplay/android-developer/answer/113469?hl=en>

Please keep in mind you need a Google Play developer account and that will cost \$25

APPLE IOS APP

14.2.1. Adding your Apple ID to Xcode

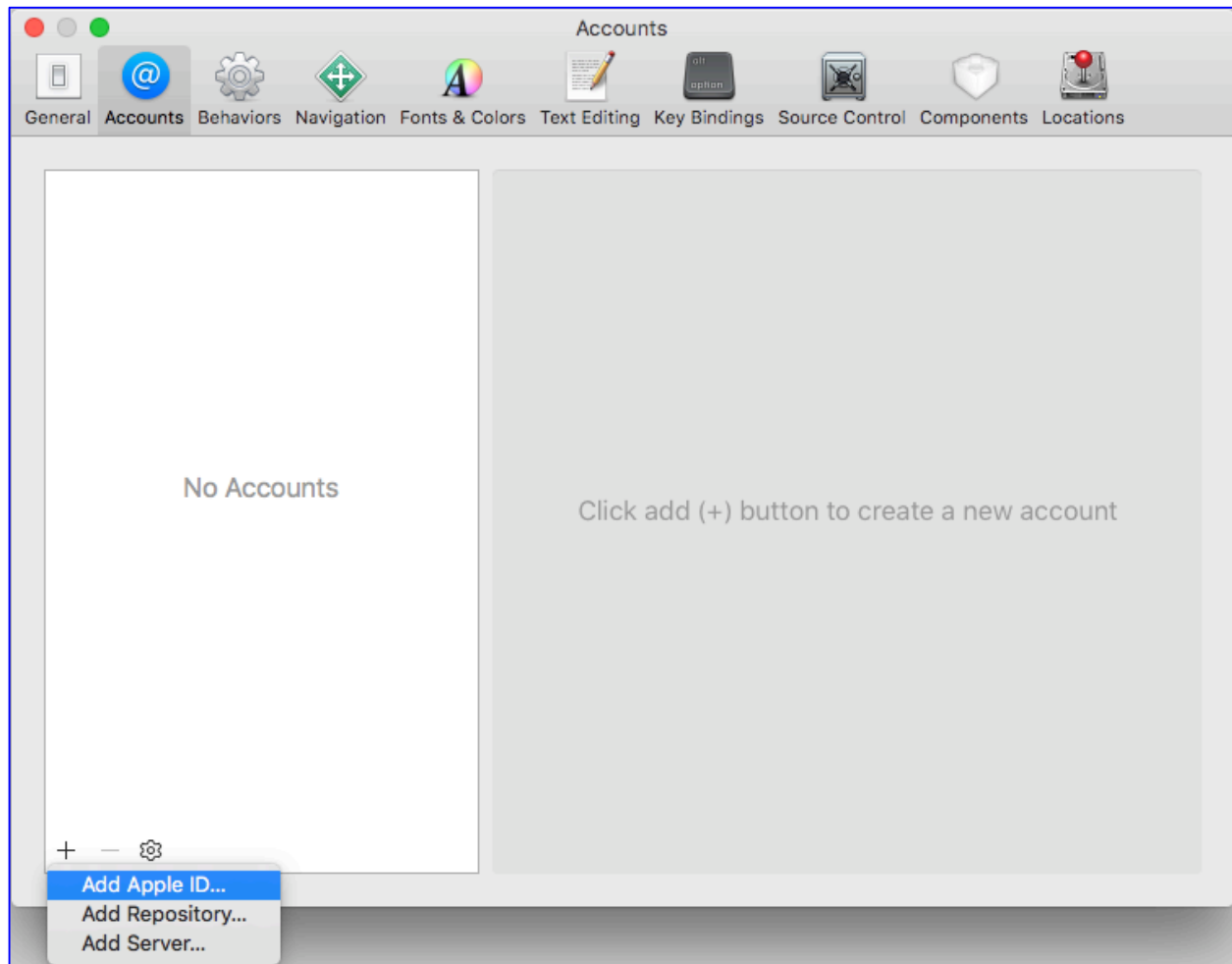
Before we can build to a device, we need to set up an Apple ID and add it to Xcode. If you don't yet have an Apple ID, obtain one from [the Apple ID site](#). Once you have obtained an Apple ID, you must add it to Xcode.

-> **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.**

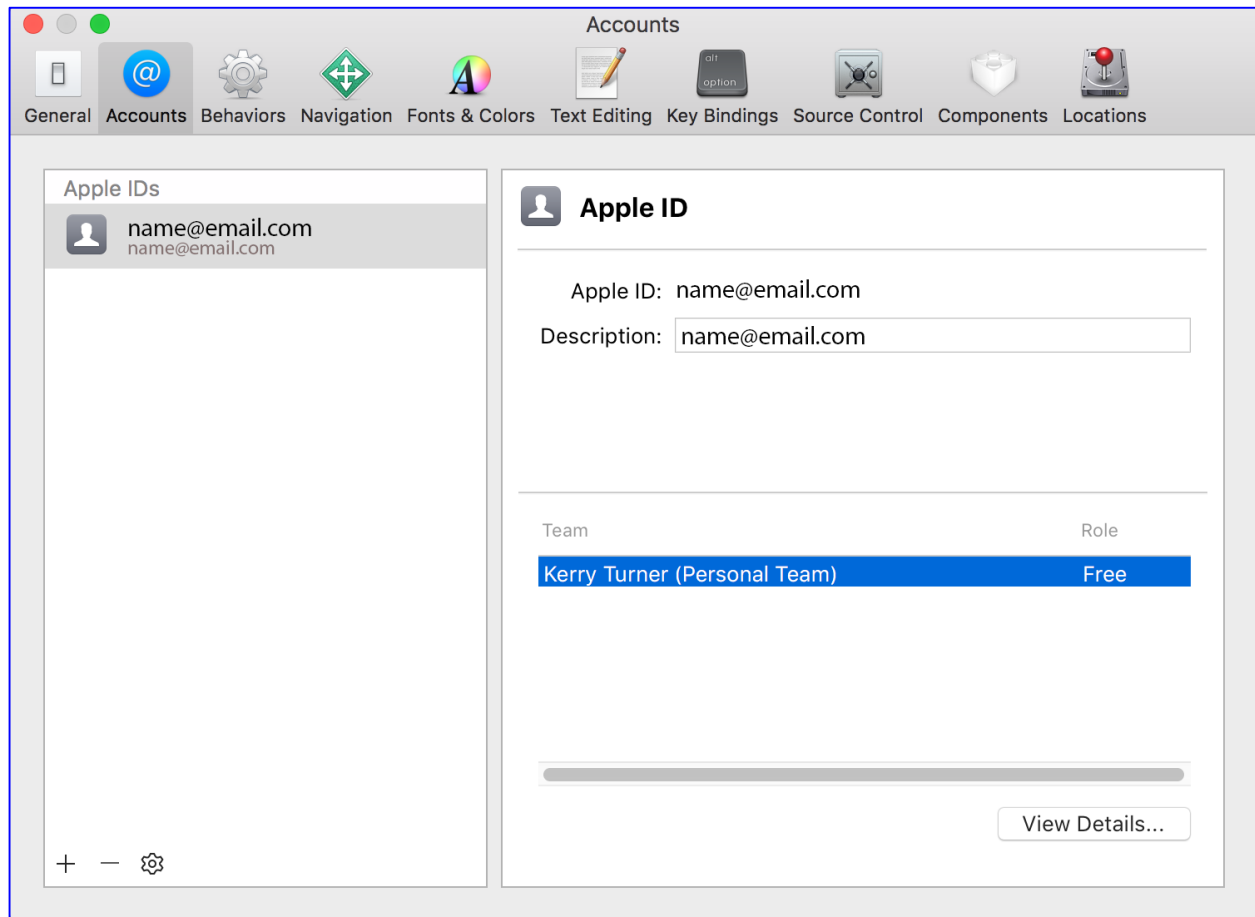
-> **To add your Apple ID, click the plus sign at the bottom left corner and choose Add Apple ID.**



-> 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)".



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.

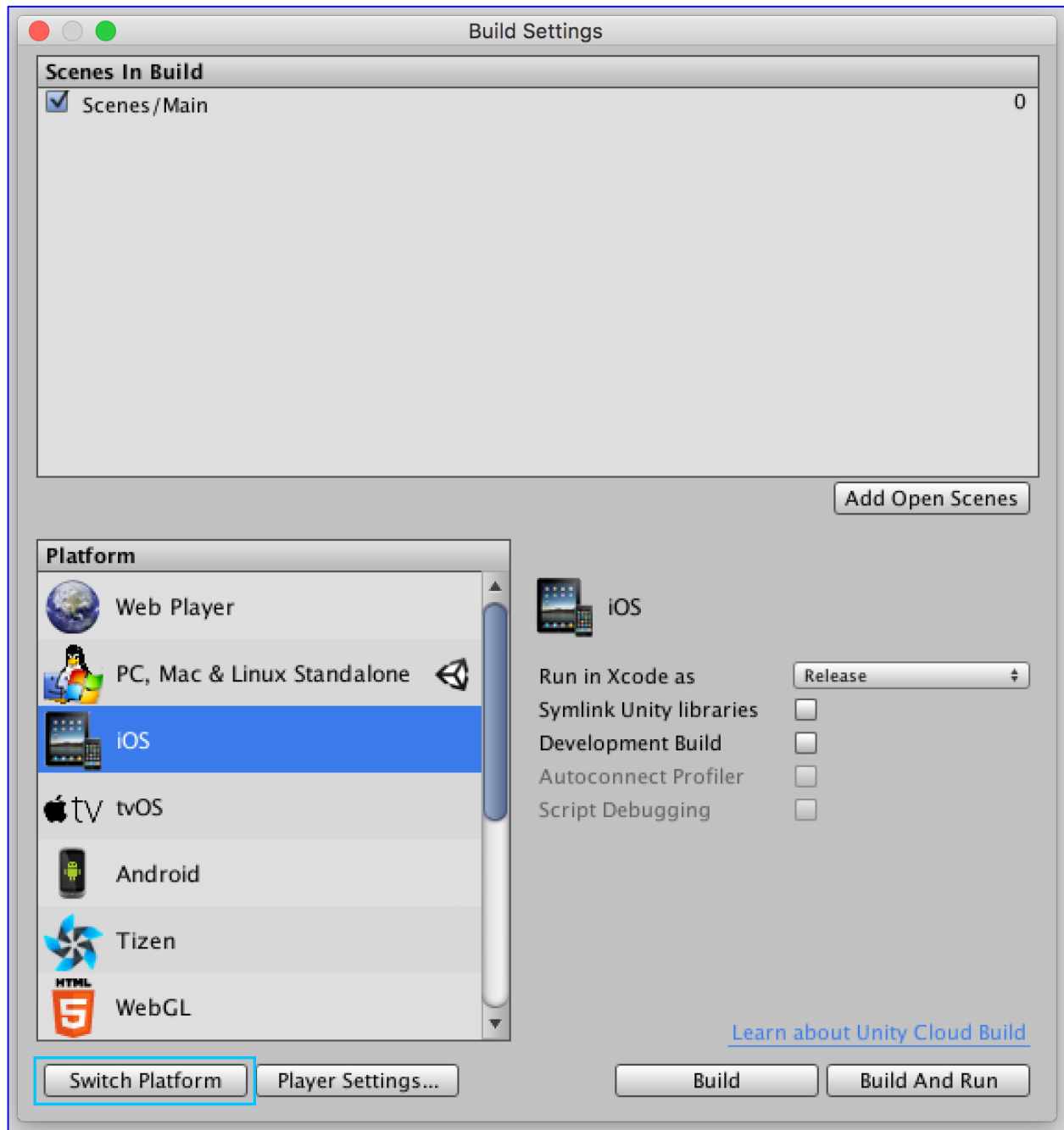
More information on managing accounts and teams in Xcode can be found in [this Apple documentation](#).

14.2.2. Preparing your Unity project for building to iOS

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



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 app. A bundle identifier is a string that identifies an app. It's written in what is known as reverse-DNS style, following the format

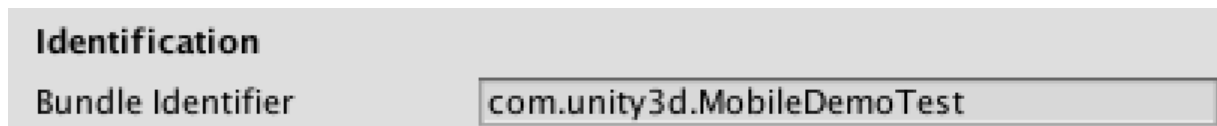
com.yourCompanyName.yourappName. Allowed characters are alphanumeric characters, periods and hyphens. We need to change the bundle identifier from the default setting in order to build.

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 app 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 app. 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.yourappNameTest.

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:**



Now we're ready to build!

14.2.3. Building an Xcode project using Unity

Building your app 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.

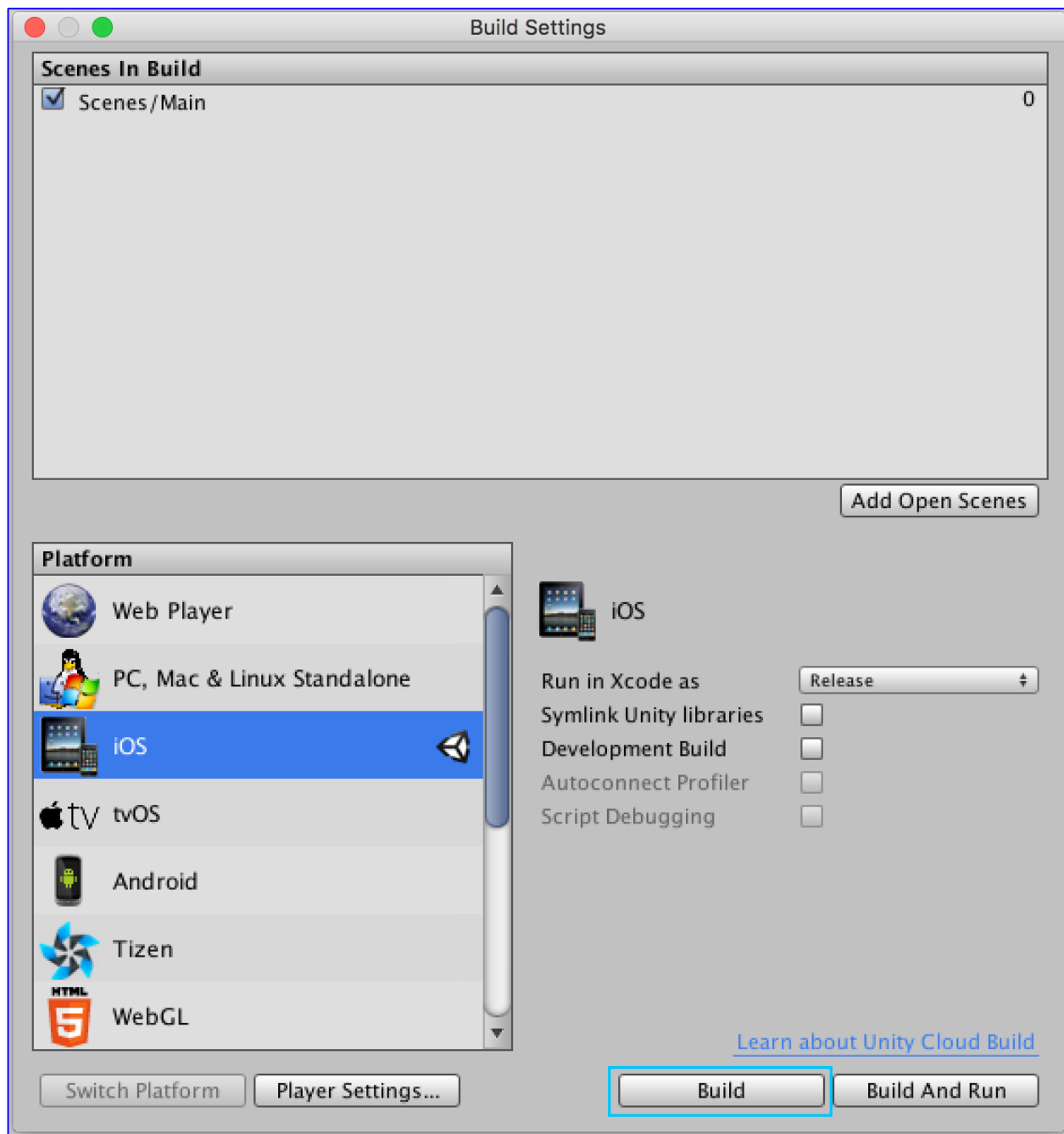
The reason for the extra work is security. Apple uses a technique called code signing to ensure that apps come from known sources and haven't been tampered with, and this needs to be set up before you can build. For more information on code signing, see [this Apple documentation](#).

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.

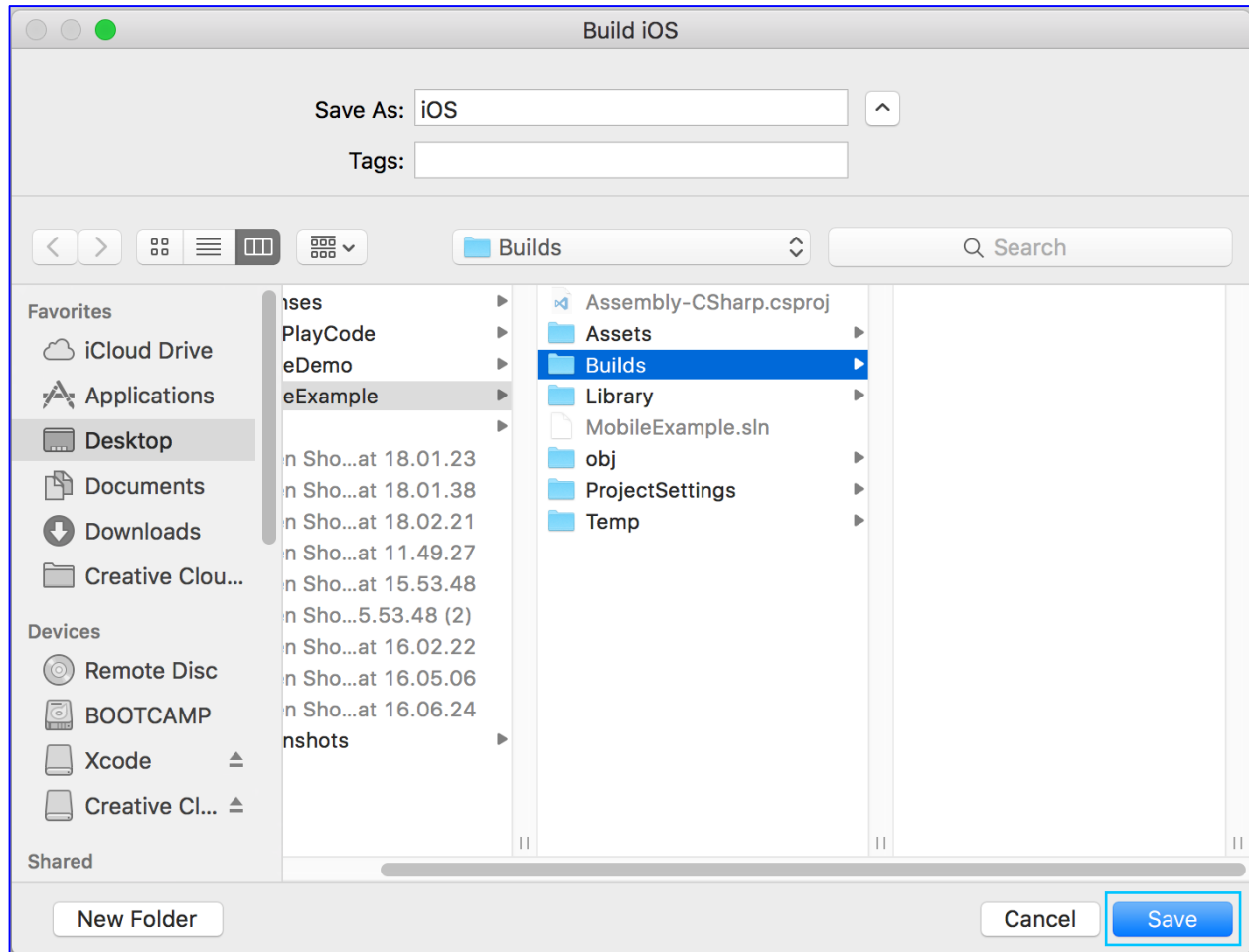


You will be prompted to choose where to build your Xcode project. A good place to do this is in a dedicated builds folder within your project folder. We'll make one now.

-> 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.

14.2.4. Building the sample project to your device using Xcode

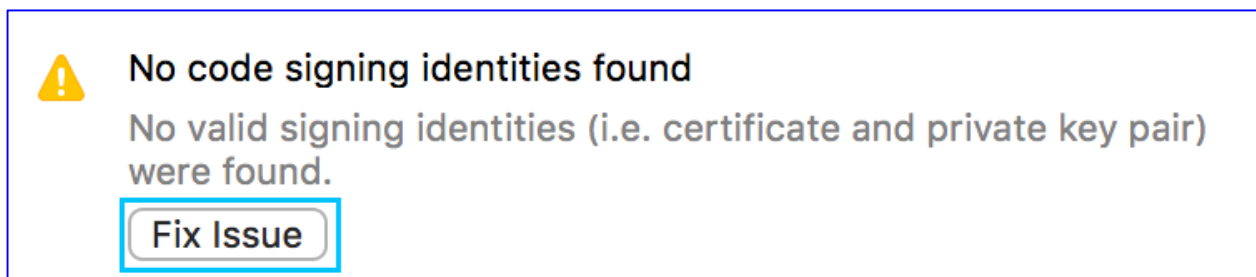
Once Unity has built the Xcode project, a Finder window will open at the project's location.

-> 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.

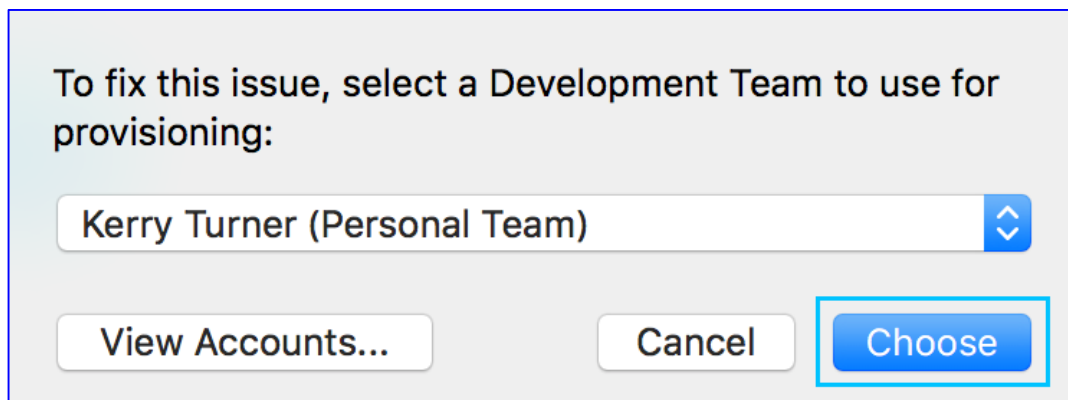
-> Click the **Fix Issue** button.



-> 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.



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**.

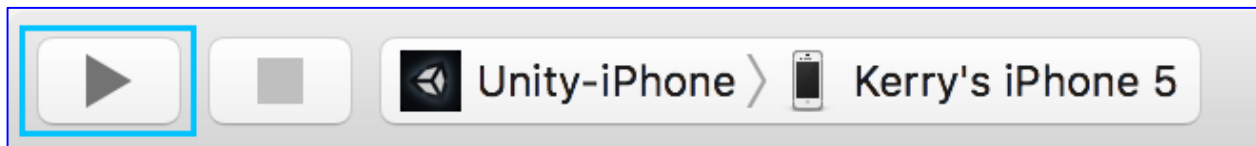
-> To disable locking, select **Never**.

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".

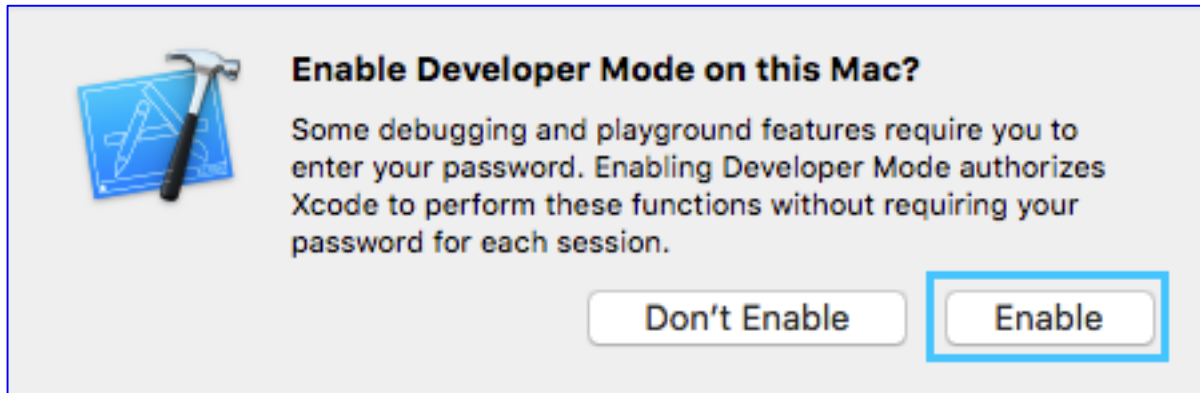
Now it's time to build to your device.

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

-> **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.

14.2.5. Testing the app on your iOS device

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

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

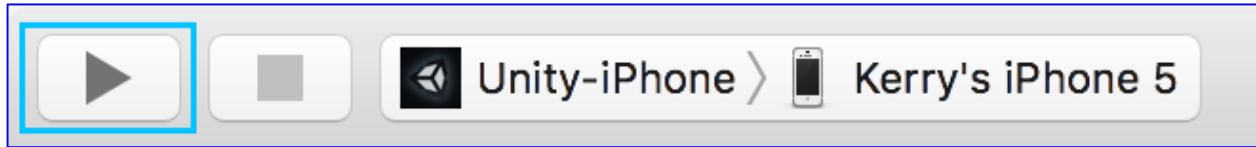
Simply playing your app on a device is one way of testing it. It's a good way of checking things like whether the controls work. However, if you want more information - if you'd like to see Unity's logs while you play, for example - you'll need to build and run the app while the device is still connected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

14.2.6. Upload your apk to Apple app store and publish your app

Apple spent as well a lot of time in creating a documentation 'how to publish an app' and it's really good. Please check it out here: <http://help.apple.com/itunes-connect/developer/#/dev34e9bbb5a>

Please keep in mind you need the Apple Developer account costs \$99 a year.

That's it! We'll keep on updating this tutorial and thank all the people who already put a lot of this online. Unity is a simple but also complex platform. You can dive deep and learn a lot of new things. For that there a ton youtube tutorials out there.

We're looking forward to see what you will do. Have fun and enjoy the ride!

MoMAR