AR Basics in Unity

Prerequisites

In Unity Hub, under the tab Installs, select the gear icon on your Unity installation and choose "Add modules."

- For Android builds, check **Android Build Support**, **OpenJDK**, and **Android SDK & NDK Tools**.
- For iOS, you **MUST BE ON A MAC** and choose **iOS Build Support**. You will also need **Xcode** installed on your machine as well as the **iOS Platform** installed (Xcode/Settings/Platforms/iOS xx.x). Finally, you have to be signed-in under Xcode/Settings/Accounts

> Do not install **iOS Build Support** on a **Windows** machine, you can only make Apple-based builds using **Xcode** which is only available on the mac. This add-on is available for windows machines for users who perform remote-builds and configure assets.

> Xcode generates a **LOT** of storage problems with cache to your machine without an intuitive way to remove the cache. Occasionally, it's a good idea to remove it. I have provided a small tool from the macOS App Store that I recommend: **DevCleaner**

Getting Started

Ensure you have read the prerequisites above. Fire up Unity Hub and select New Project at the top. Under **All templates**, select **Universal 3D** (you may have to **Download template**). Give your project a name, location, and be sure to uncheck **Connect to Unity Cloud**. Select **Create project** .

You will be greeted with the Unity Editor as well as the Build Settings window. The Build Settings window can be found by selecting File/Build Settings. Select iOS or Android under Platform and click the Switch Platform button.

Next, select the **Player Settings** button. On the left, choose XR Plugin Management and then choose Install **XR Plugin Management** with the button that appears. Select the **XR Plugin Management** tab again on the left. Check the **plug-in providers** you will be using. For instance, iOS will use **ARKit**. If you receive a warning about using the new input system, choose YES and the

editor restarts.

Return back to the **Player Settings** window. You will notice there is now a drop down under **XR Plug-in Management**. Browse through these to see how the settings are enabled. Under **Project Validation**, you may have issues here. Many can be fixed by selecting **Fix All** but you should always read the errors.

You can now close these extra windows and return to the main Unity editor.

> You may have the Read Me asset shown in the editor regarding URP. I usually click the remove button to remove the extra files.

Install Packages (Optional)

Go to Window/Package Manager. The button is small, but there is a dropdown button in the top left of the Package Manager window (next to a '+') where you can switch between packages In Project and Unity Registry. Switch to Unity Registry.

You can find additional packages here, depending on the Unity version. If you are looking to support both Android and iOS devices, you can install additional packages here.

AR Rig Setup

If you have set up VR in Unity before, the process is similar.

- Click on the game object Global Volume and either delete it or disable it. Post processing is very heavy with mobile devices.
- Delete your Main Camera.
- Right click in the Hierarchy and choose XR/XR Origin (Mobile AR).

This creates your rig! Be sure to browse the components in the inspector including the children objects so you obtain an understanding of what has been created. When finished, click somewhere in the Hierarchy to unselect any game object.

There is one last thing we need to do. We are using URP (Universal Render Pipeline), a modern way to create design-friendly shaders and performance modes. Currently, we need to add a rendering feature for AR so that our camera will work as intended in a build.

- In your Project window, you will find a folder called Settings. This is a great folder for (as you would expect) file-based settings for your project! Select each Renderer file (e.g. URP-Balanced-Renderer, URP-HighFidelity-Renderer, etc.) and within the Inspector, click the button Add Renderer Feature. Select AR Background Renderer Feature in the pop-up window. Again, do this for each file ending in "-Renderer."

Build Settings

Let's set up a few settings in our **Project Settings** so that our project is better optimized for assets/quality.

Go to File/Build Settings/Player Settings (same place we installed XR Plugin Management).

Under **Quality** you will find a grid-like area called **Levels**. Choose either **Balanced** or **Performant** .

Select **Player** and choose the following

- **Company Name**: Enter your name or the company name
- **Product Name**: Change this if needed.
- **Other Settings/Bundle Identifier**: Give a UNIQUE name here in reverse URL format. E.g. com.scil.MyAwesomeProject
- **Other Settings/Camera Usage Description**: For iOS, be sure something has been added here or the build will fail.

Build Your Project

Android

If you are using an Android device you may need to be in developer mode. Depending on your phone there are different procedures. After developer mode is set up, plug the phone into the computer with a USB cord. Select "allow USB debugging" on your phone if the message appears. Finally, under File/Build Settings, you can refresh in the drop down until your phone appears. After it appears, select **Build and Run**.

iOS

> iOS is tricky and it's best to decipher error messages as you get them. Results vary, but here is a general step-by-step guide:

Go to File/Build Settings and select Build. Create a new, empty folder to host the generated Xcode generated files. Open the directory after the build completes. Inside will be an Xcode project file called something like **Unity-iPhone.xcodeproj**. Open that file and a wild **Xcode** will appear.

- On the far left in Xcode you will see a list of files and a **blue root directory** called something like **Unity-iPhone**. Select that file and you will notice there are settings for the overall app that appears. At the top left is a row with General, Signing & Capabilities, Resource Tags, etc. Select Signing & Capabilities.

- Under **Signing & Capabilites**, first check your **Bundle Identifier**. Change it to a unique ID if not already in reverse URL order (e.g. com.scil.MyAwesomeProject). If you do not have a unique ID, the **build will fail**.
- Next check the box for **Automatically manage signing**.
- Under **team**, select your team (usually your signed-in account as Personal Team). If you have the unique ID, everything should work.
- Now, plug-in your iOS device you will be TESTING. Select allow after plugging in your device.
- You may need to set up **Developer Mode**, you can find info **here**
- When you are ready select the big **Play button icon** after your device is configured. You can also choose Product/Run.
- The very last step, is the first time you run a project on your device the build will fail and complain that this is an un-trustred developer. To "trust" yourself, you need to allow projects with your ID within the device. Go to your device Settings/General/VPN & Device Management and choose and allow your team. You will need to select **Run** again from **Xcode**.

> You should now be able to play and test your project. The nice thing, is after your project and device is configured, you won't have to run these extra steps again. The purpose for these steps is Apple has heavy security on their platforms, as well as the **walled-garden**

The initial setup is the most time consuming process, but you made it.

Where to go from here?

There are many paths to take when using AR. Try setting up more to your scene and placing objects. You will notice even simple objects, such as a simple cube in Unity, are quite large!

Special technologies, such as plane-detection, image tracking, object tracking, and more are available. I recommend starting with plane-detection which you can learn about here:

https://learn.unity.com/tutorial/configuring-plane-detection-for-ar-foundation#

Revision #2
Created 2024-08-07 15:58:27 UTC by Wes
Updated 2026-07-27 13:52:44 UTC by Wes