# Virtual Reality — Spring 2022

**Date out: January 28, 2022**

**Date due: February 11, 2022** (Note: There's nothing to turn in for this lab other than a single line indicating you have done this.)

"I have finished Lab 2".

# Lab 2: Installing and Testing Your Unity Android or iOS Development Environment

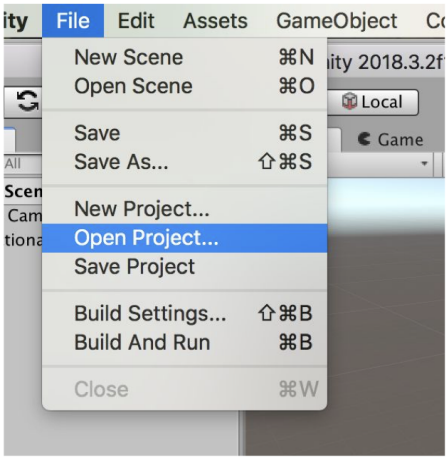
This lab builds on Lab 1. If you have not yet installed Unity 2019.4 and gotten it to run on your Windows or macOS computer, please complete that lab first!

You are now ready to install and test the mobile development environment that you will use to deploy apps from within Unity. As in Lab 1, we have provided step-by-step instructions for preparing and testing your environment. We would like you to complete Lab 2 soon since Assignment 1 will require that you deploy your mobile device, and we want that part of Lab 2 to be a minor detail.

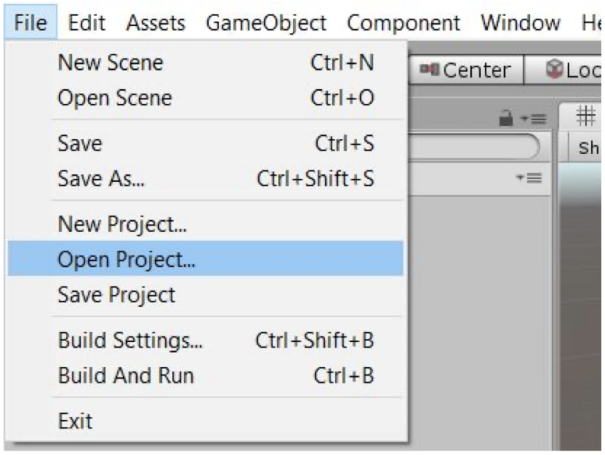
Visit the online guide to Unity. Note the software and hardware requirements listed on that page above Step 1, and follow the iOS-specific parts of Step 1, if needed, followed by Step 3. (Lab 1 has included the rest of Step 1 and all of Step 2.) Step 3 will require much more work for iOS than for Android! *When done with Step 3, the following instructions replace Step 4 on that*:

Once your mobile development environment is ready, you will create and deploy to your mobile device a slightly extended version of the simple project you created in Lab 1. First, you will need to run Unity Hub and open the project you created earlier. If the Unity Editor is already open and you had last opened a different project, then select **File→Open Project**, as shown in the following screenshot:

**Mac:**




**PC:**



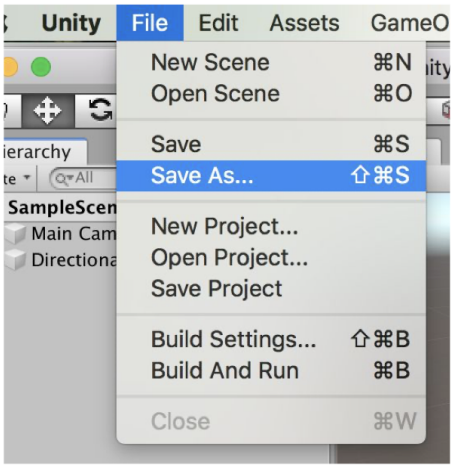
Choose from Unity Hub the correct project.

Unity stores *GameObjects* (scene objects) and their configurations in the *hierarchy* (scene graph) in Scene files in a project. As we saw in Lab 1, a default Scene file called “SampleScene” is created when you create a project. (This file stored the initial hierarchy you modified in Lab 1 and saved it as “TestScene.”)

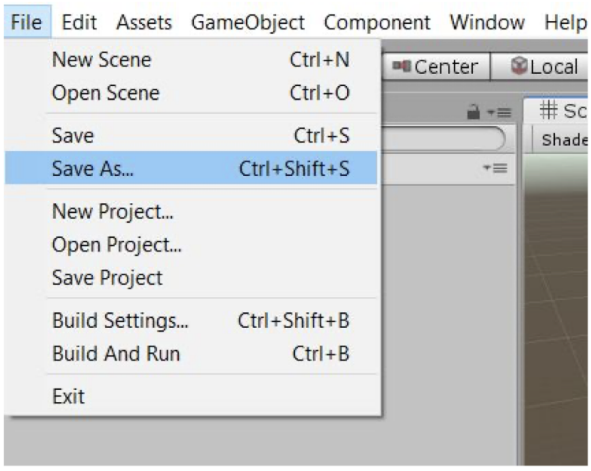
If you had saved the Scene you created in Lab 1, as instructed, and the project opens on the default “SampleScene,” then open the Scene you had made for Lab 1 (and saved in the **Scenes** folder), by double-clicking its icon in the **Project** View.

However, if you had *not* saved the scene as instructed in Lab 1 and had *not* accepted the prompt to save the scene you created when you had exited Unity, then you should recreate that Scene (as described in Lab 1) and save it by selecting **File→Save as...**

**Mac:**

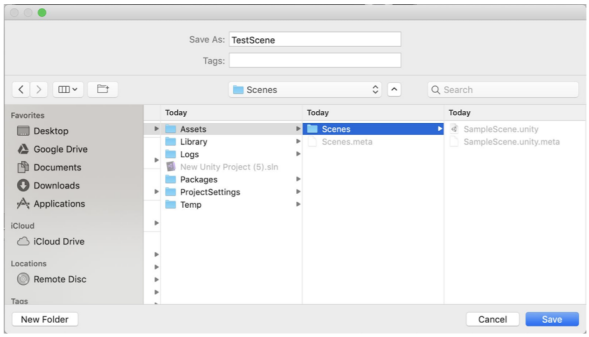



**PC:**

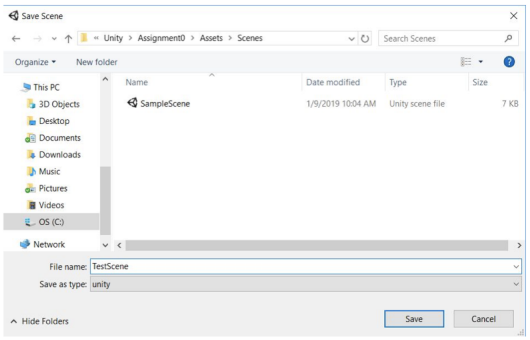



A dialog window will then appear, which will allow you to name the scene and specify a folder in which to save it. Name your scene “TestScene,” navigate to the **Scenes** folder in the **Assets** folder, and press the **Save** button:

**Mac:**

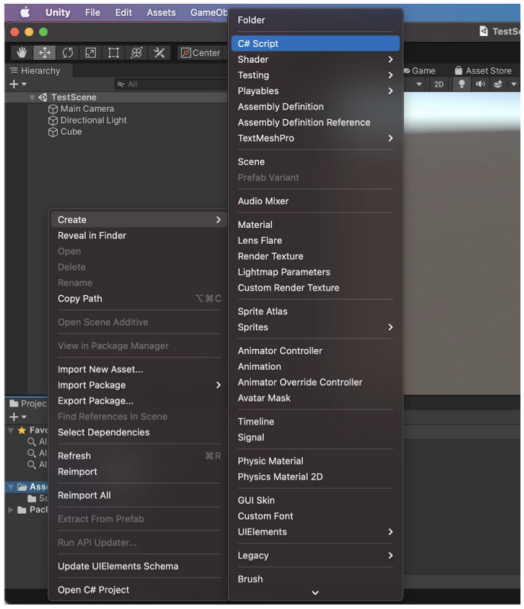



**PC:**




Now, you will add a new feature to the app that will cause the cube you created in Lab 1 to rotate. Start by right-clicking the Assets folder in the **Project** View in the lower left of the Unity window and selecting **Create→C# Script.** When you see the file icon appear in the right column of the Project View, type “Script1” to change its highlighted name. If you do anything else before changing the

name—even pressing return or clicking outside the icon—you're going to have an additional step to do later!

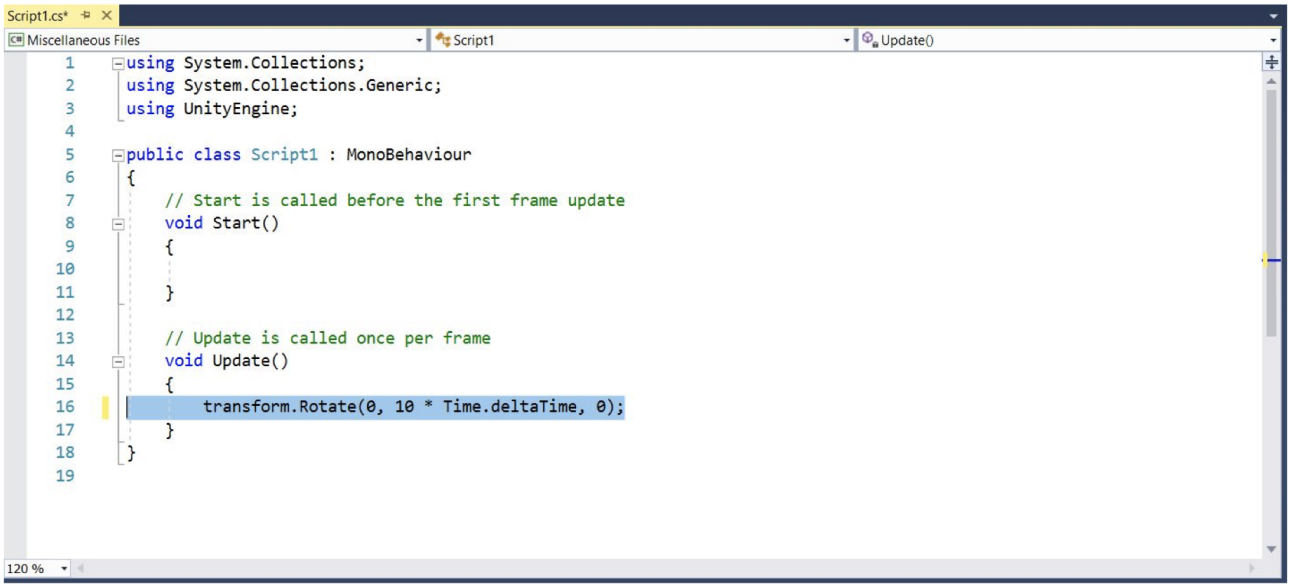




Next, double-click the file. The source file will open in Visual Studio (or any other default code editor under Windows). The file contains two functions: Start () and Update (). Add the following line of code to the Update () function, which runs once per frame:

transform.Rotate (0, 10 * Time.deltaTime, 0);

Noting the documentation for Transform.Rotate and Time.deltaTime, when your app is running, this line will cause the GameObject to which it is attached to rotate during this frame about the local *Y* axis by an angle (in degrees) equal to ten times the amount of time in seconds since the last frame. (That is, to a good first approximation, over a period of one second, your cube will rotate by 10°. Please think about that carefully.) Your code should now look like this:

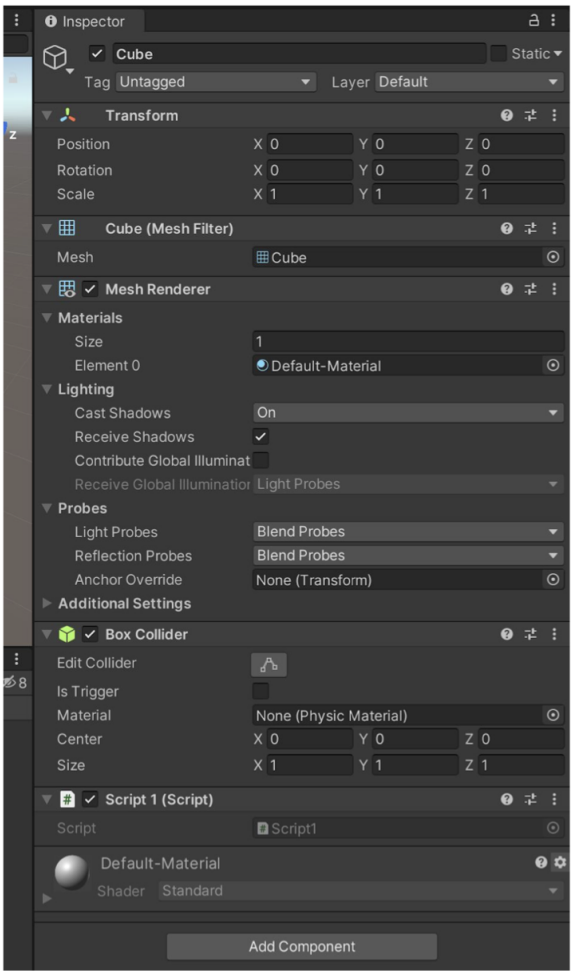



The thick vertical line next to the change you made will initially be yellow and then green when you save the file.

**Note: If you didn't change the script name immediately upon creating the C# file, make sure that you change the Class Name (in the screenshot above, it appears in line 5 right after "class") to match the file name ("Script1" in this case).**

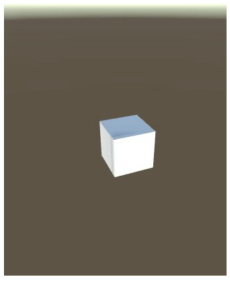
Now, save the file without exiting Visual Studio, and return to Unity. You will want to attach the script you just created to the cube. To do this, left-click and (without unclicking!) drag the "Script1" file in the **Project** View to the cube in the **Scene** View or the word "Cube" in the **Hierarchy** View, dropping it when you see the destination highlight. (Note: When you do this, Unity will, unfortunately, *not* provide any confirmatory feedback to attach the script. ☹)

However, click on the cube in either the **Scene** View or **Hierarchy** View. You will see the script you added listed as a component of the cube toward the bottom of the **Inspector** View (with an extra space between "Script" and "1" automatically added by Unity):




Then, select the“MainCamera” in the **Hierarchy** View and, in the **Inspector** View, set the **Transform** “Position” “Y” value to “3” and “Z” value to “−5”, and the “Rotation” “X” value to “30”.

Now, press the play button to preview the project in Play Mode. The **Game** View will open, and you will see the cube rotate by 10° per second (shown cropped here):



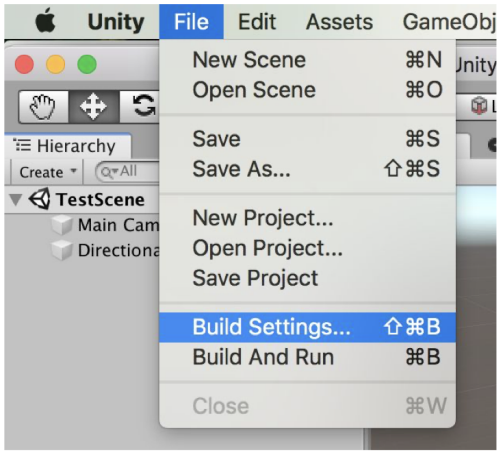
Press the play button again to stop the project from running.

**Note: You can make changes to scene objects while in Play Mode. However, any changes**

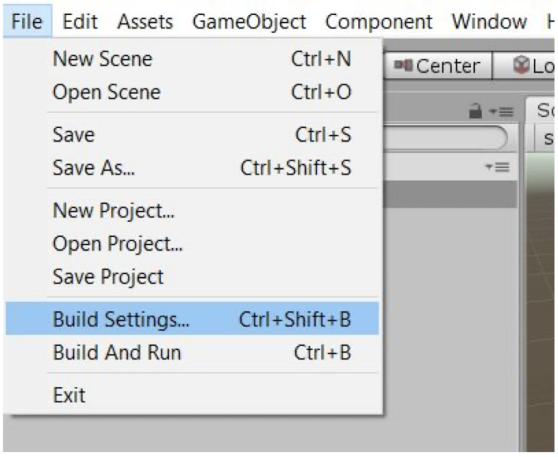
**to scene objects made while Unity is in Play Mode will be *discarded* when you stop the project from running! We guarantee you will forget this many times during the semester.** ☺

Your last step is to deploy the modified app to your mobile device, which should be connected by USB to your computer. To do this, first select **File→Build Settings...**. The **Build Settings** window will allow you to change your deployment target to **Android**:

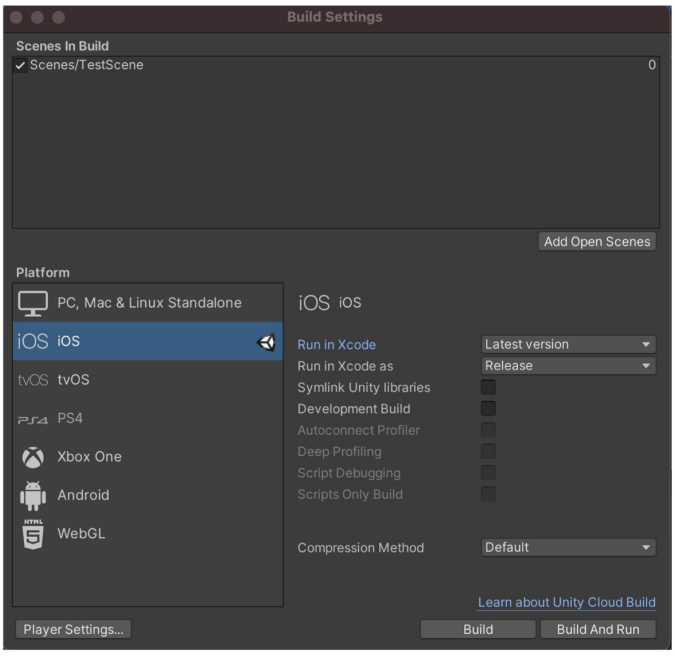
**Mac:**




**PC:**




Otherwise, if you are deploying to an iOS device (allowed only from macOS), select iOS. Press “Install with Unity Hub” if you didn’t install the corresponding modules and SDK in Lab 1. Then press the **Switch Platform** button in the **Build Settings** window:




Next, make sure that your scene will be included in the build. (If your scene is not listed in **Scenes In Build**, then press the **Add Open Scenes** button to include it.) Now, press the **Player Settings...**

button. This will open the **PlayerSettings** in the **Inspector** View, where you will need to expand the **Other Settings.** Under “Identification”, you will see the “Package Name” (Windows) or “Bundle Identifier” (macOS) text input box.

*You must set the “Bundle Identifier” or “Package Name” to something other than the Unity default! For example, you can use “edu.columbia.myapp”. (This does not have to be unique, so it’s fine to use anything you’d like for an app that is not going to be sold.)*

Now, please read the section that follows for the device to which you will deploy *very carefully* and follow its instructions.

*If deploying to Android:*

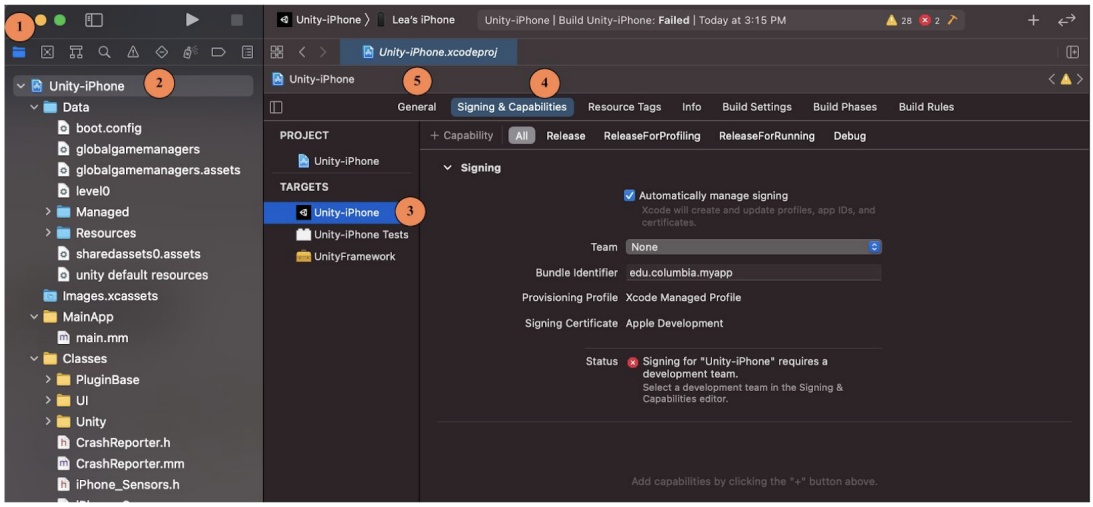
- When you select **Build And Run**, Unity will build the project. You will be prompted to name your .apk (Android application package) file. The name you choose will not matter, as long as you realize that it will overwrite any file of that name. (We suggest that you place your .apk files in a *bin* directory that you create at the same level as your *Assets* directory to avoid clutter and simplify the process of code versioning.)
- The first time you build for Android, you may be asked to specify the root directory for the Android SDK if you didn’t install Android SDK in Unity Hub.
- If you already have Android Studio installed, you can find that SDK path on Windows by opening Android Studio, and selecting **Configure→SDK Manager**, where you’ll see a text box labeled “Android SDK Location” at the top. (Alternatively, start up an Android Studio project and look in the **File→Settings** dialog box, in which you may need to expand A**ppearance & Behavior→System Settings→Android SDK** to find the text box labeled “Android SDK Location:”.) If you specify the wrong directory, you will need to change it in Unity in the text box for “SDK” under “Android” in **Edit→Preferences→External Tools**.
- At this point, you may receive the error message “Error building Player: CommandInvokationFailure: Unable to list target platforms. If so, please make sure the android sdk path is correct.” (Yes, they can’t spell.) This used to happen if JDK 9 or 10 was installed since Unity requires JDK 8. If this occurs, please go to **Edit→Preferences→External Tools** and check the “Use embedded JDK” box. (This should have been fixed by Unity embedding the needed JDK, but the box might have somehow gotten unchecked.)
- Once the build process completes, Unity will automatically attempt to deploy the app to your Android device and run it. For many Android devices, you will need to ensure that the device is awake and unlocked when you do this. If you get a message about no Android device being found, you may also find it necessary to disconnect and reconnect your device. And you may also find that your computer issues annoying warnings about a new device being connected or not being recognized when you tilt your device to change between portrait and landscape orientations.

*If deploying to iOS:*

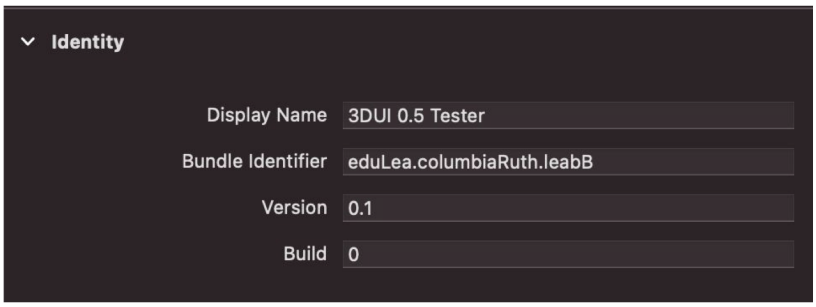
- *If you already followed the instructions in Step 3 on our* online Unity software Installation and *you use an XCode-managed provisioning profile, you do not need to change the “Bundle Identifier.” However, if you created your provisioning profile through the Apple*

*Developer Portal, you will need to ensure that the "Bundle Identifier" uses the exact information entered when creating that profile.*

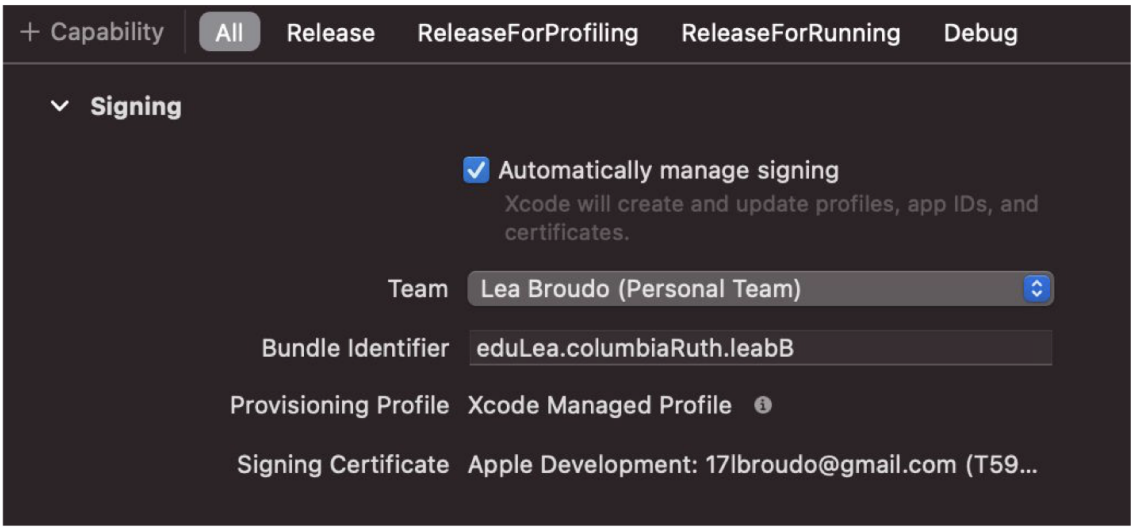
- When you select **Build And Run**, Unity will build the project. You will be prompted to choose the location and name for your XCode project. The name you choose will not matter, as long as you realize that it will overwrite any file of that name. Once the build process completes, Unity will open XCode, which will handle deployment to your iOS device, where it will run. It's ok if XCode gives you warnings as long as your app deploys and runs.
- If a previous Unity XCode project was open, Unity will close it and reopen XCode. XCode will then run your app.
- Xcode might give you the error "Signing for Unity-iPhone requires a development team". Here's how to fix it.
    - First, make sure you've set up your Apple account like this page says.



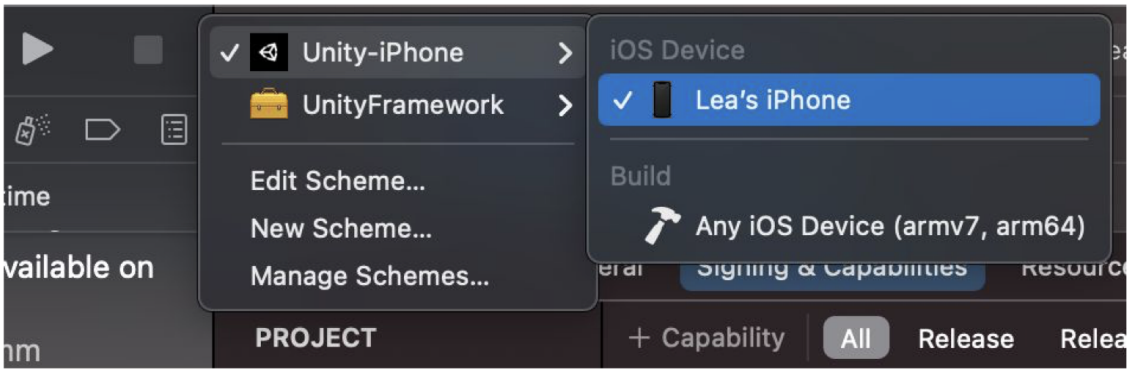

    - Back in Xcode, click the little folder icon on the upper left corner (orange circle labeled "1" in above picture), then click on the root level "Unity-iPhone" (2), make sure you are using the first build target (3), and select the "General" tab (5).
    - Change the "Bundle Identifier" to match the one you specified in the Unity Player Settings.




    - Go to the "Signing and Capabilities" tab (4), and make sure the tab "All" is selected (this is next to "Capability" in the below picture.




- It should create a certificate for you if you don't have one, and everything should work after that. Also, make sure you've chosen your iOS device as the device to use (rather than "Generic iOS Device").




- Now, click the play button to Run, and it should run on your iOS device!

Congratulations! You have deployed and run your first mobile Unity app! You can stop it the way you would stop any other app on your device. And if you'd like to run it again, you'll find it on your device in the usual place that a newly downloaded app will appear.

# Optional:

To help me learn your expertise, share a link to a Unity project you developed that you are most proud of with me by email chen.8028@osu.edu. Thank you!