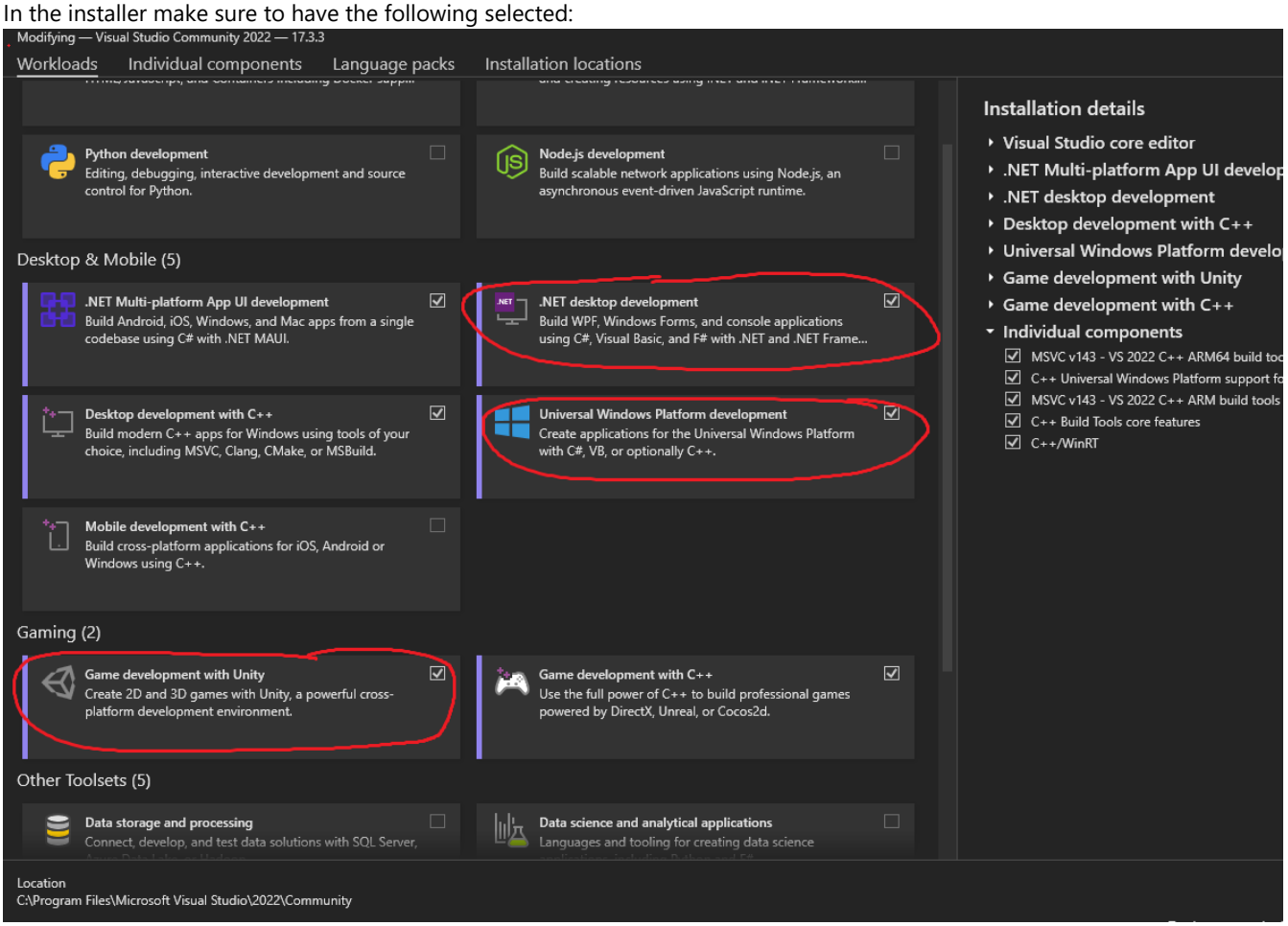
Install Visual Studio 2022

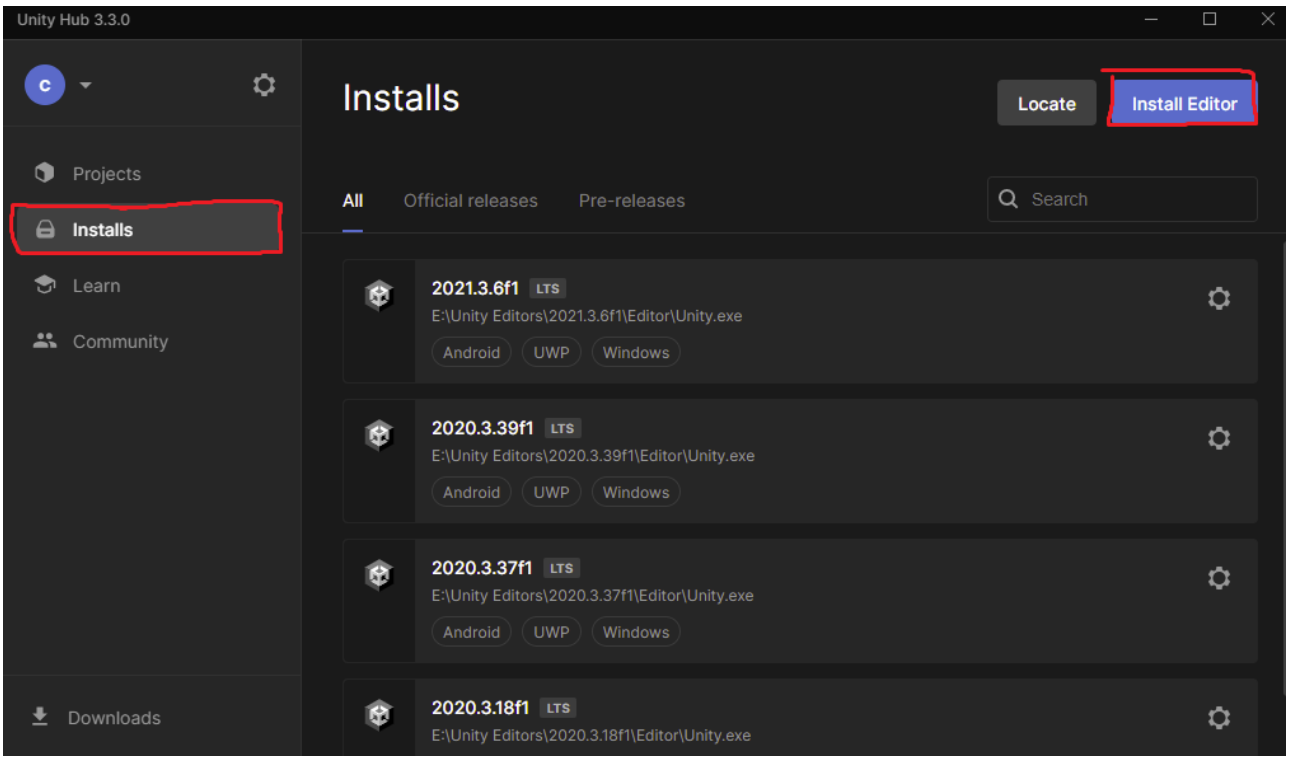
Download Visual Studio

In the installer make sure to have the following selected:

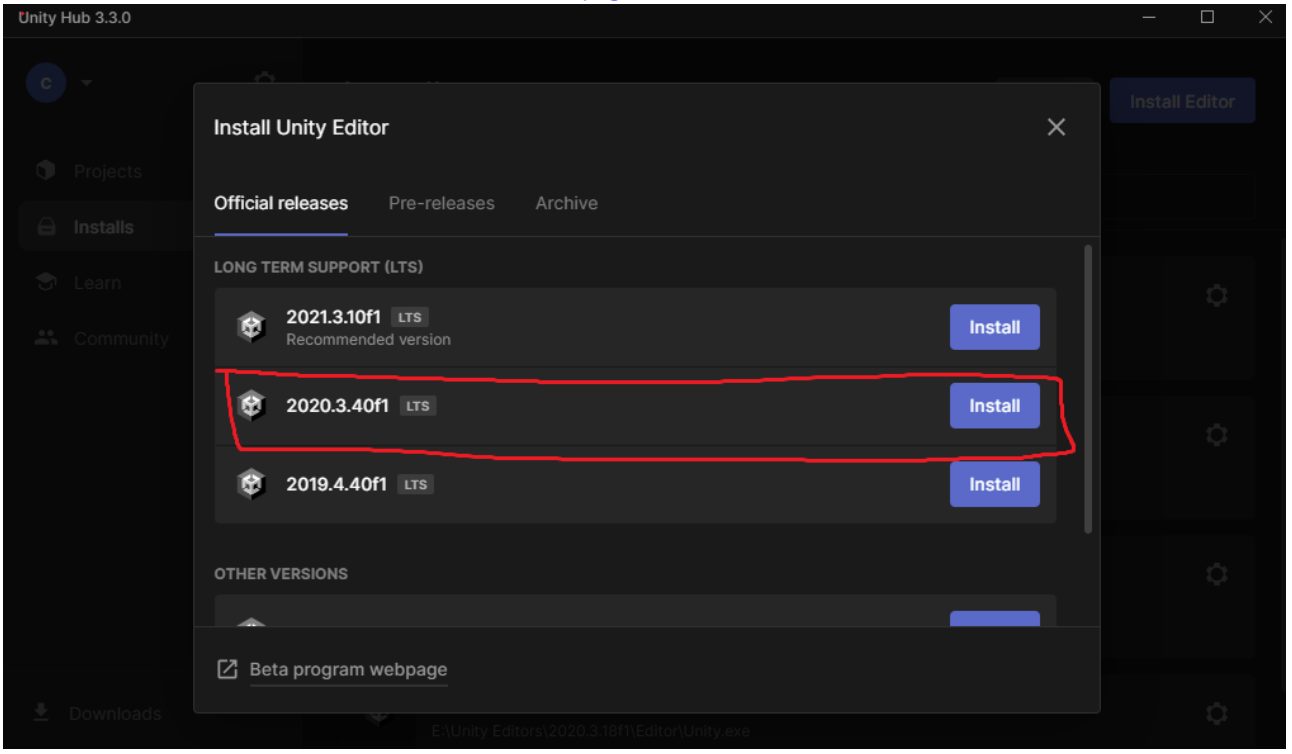


Install Unity

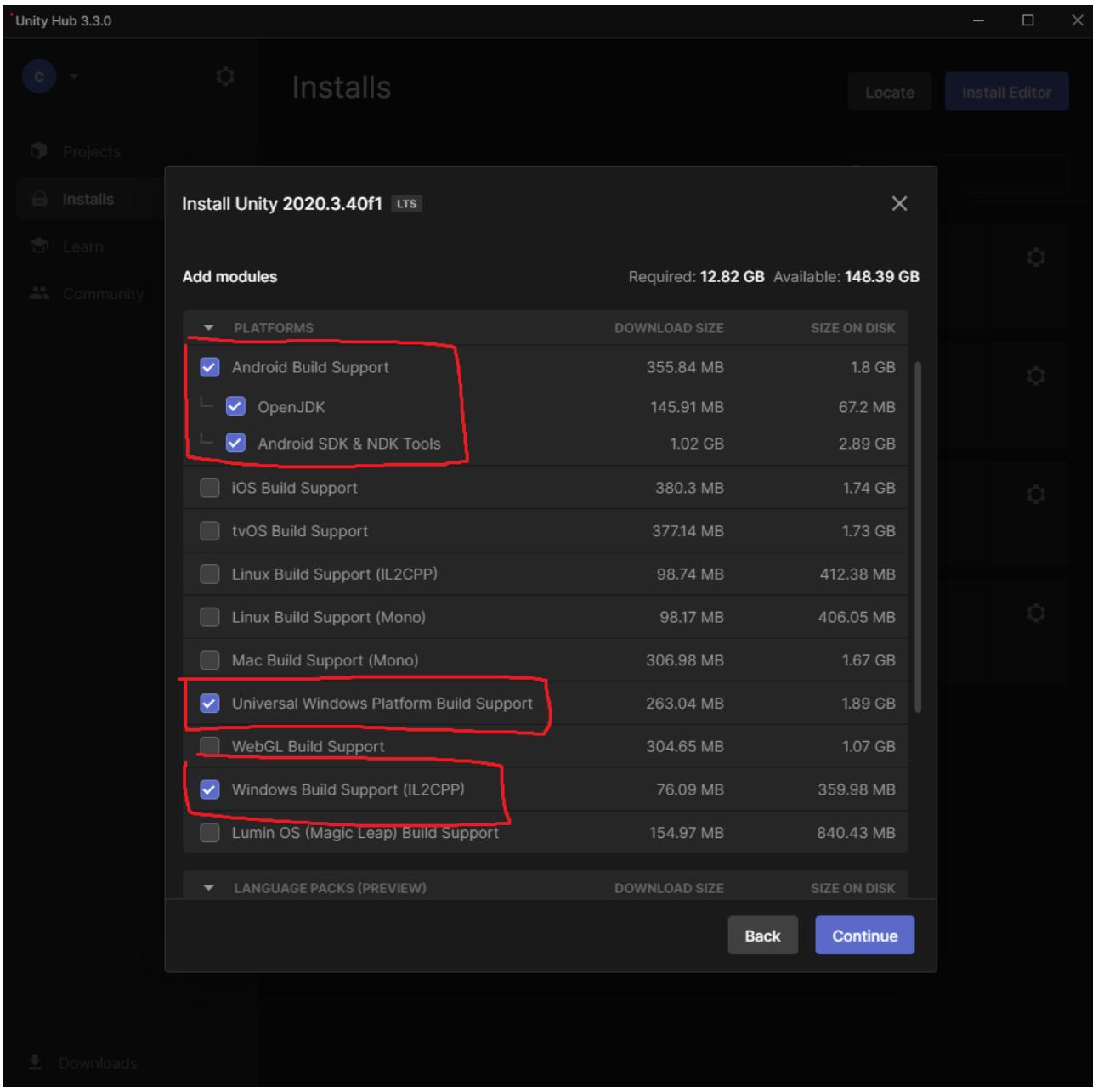
1. Install Unity Hub
2. Navigate to `Installs` on the left panel and click `Install Editor` on the top right:



3. Choose version `2020.3.40f1` (or whatever 2020 LTS version is available)
   NOTE:: The reason `2020.3.40f1` is used over `2021.3.10f1` is because it is more stable with the hololens as of when this document was written. Refer to Microsoft's known issues page.



4. Make sure you install the following modules:



Download MRTK Feature Tool

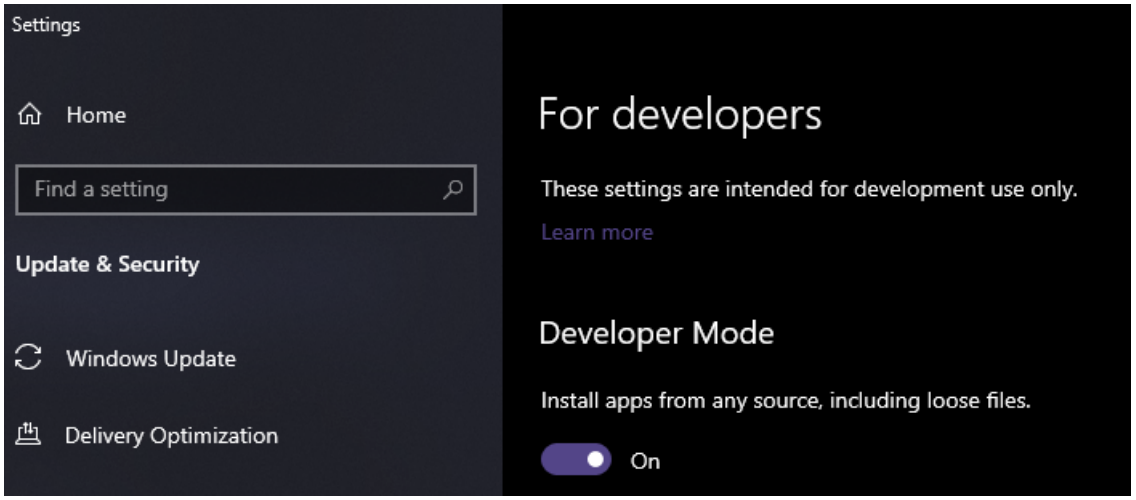
Download MRTK Feature Tool from the download section, this is required to use the majority of tools provided.

Setup headsets

Different headsets require some additional setup:

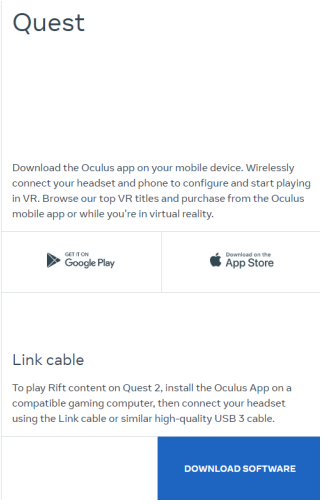
Hololens 2

1. Turn on Developer Mode in windows settings on *both* the hololens and desktop: (Make sure they are paired, click pair on hololens and input code)

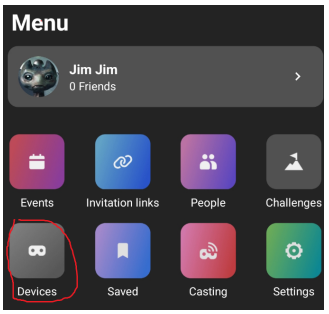


Oculus Quest 2

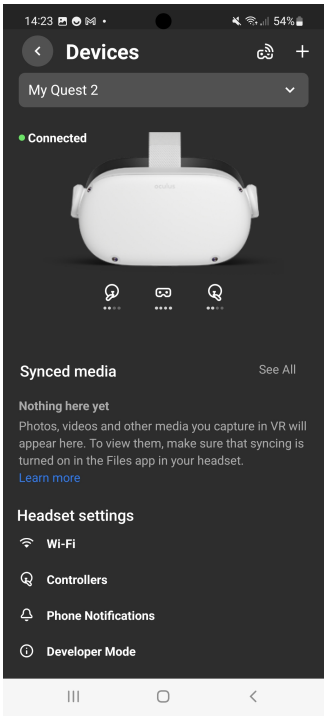
1. Create Oculus account.
2. Sign up for developer. - link
3. Download the Oculus desktop app. - link



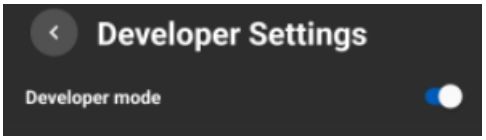
4. Download the Oculus mobile app.
5. Login to Oculus account on both desktop and mobile app.
6. Connect to Oculus on mobile app using bluetooth.
   - Go to `menu` in the bottom right and select devices



   - Scroll down to `Developer Mode`



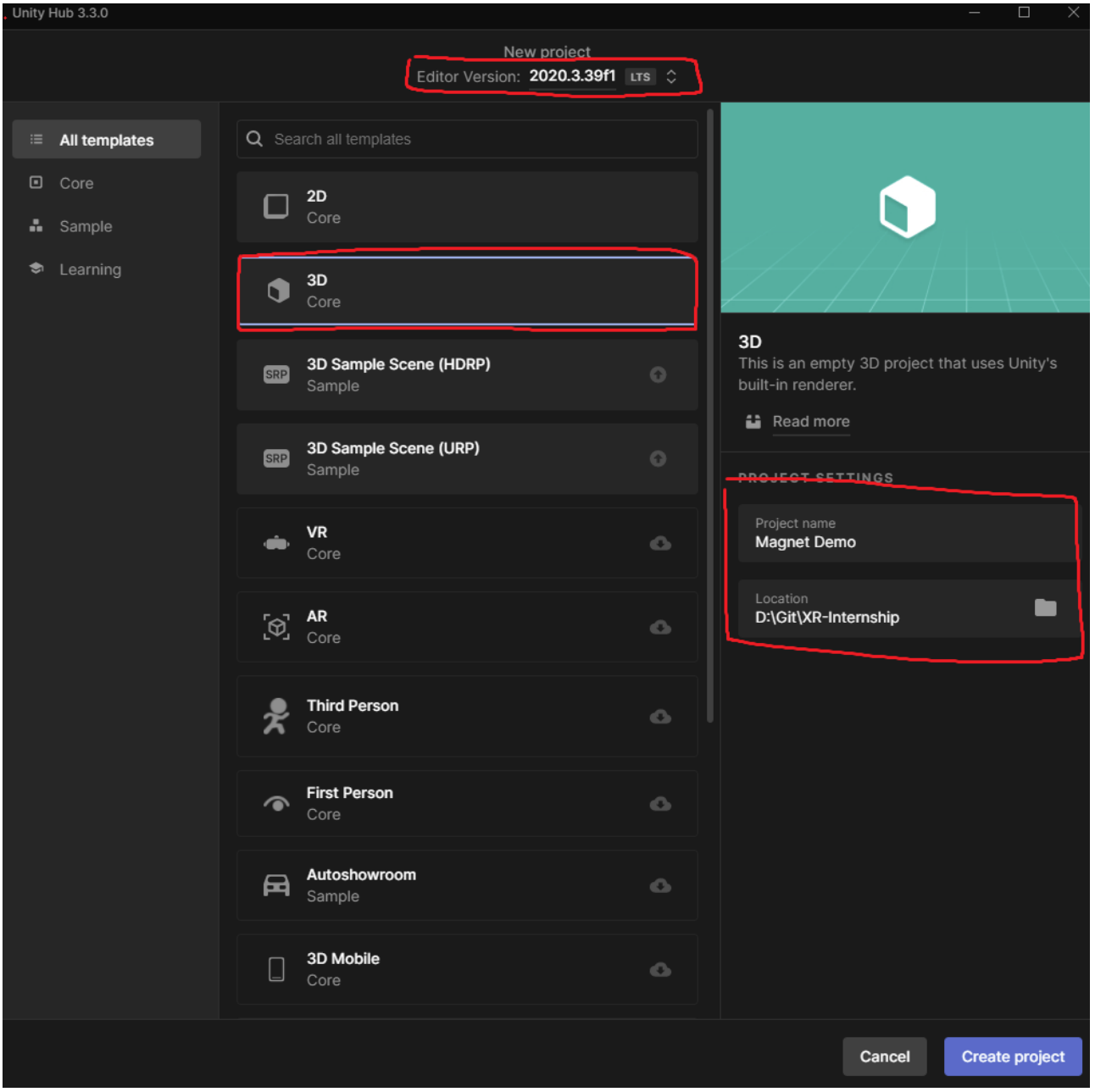
- Enable `Developer Mode`



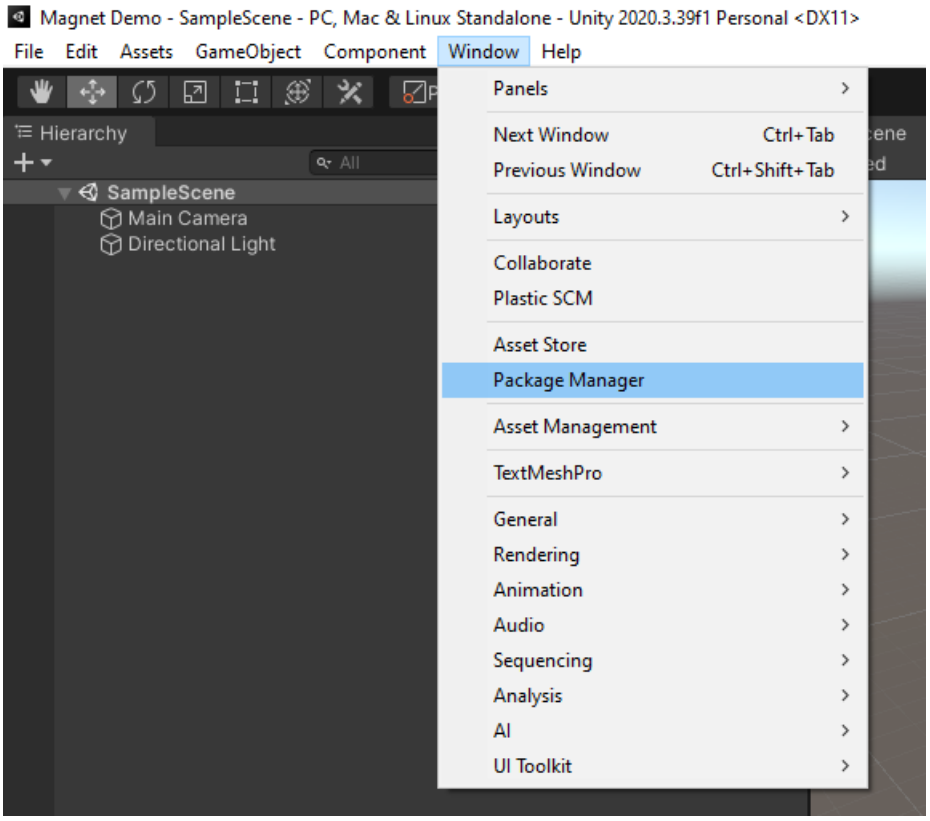
7. Install [oculus adb drivers]
8. Connect the headset to your PC via link cable (USB-C connection). When you put the headset on it should prompt you to enable `USB Debugging`. Click `Allow`.

Setup Unity Project

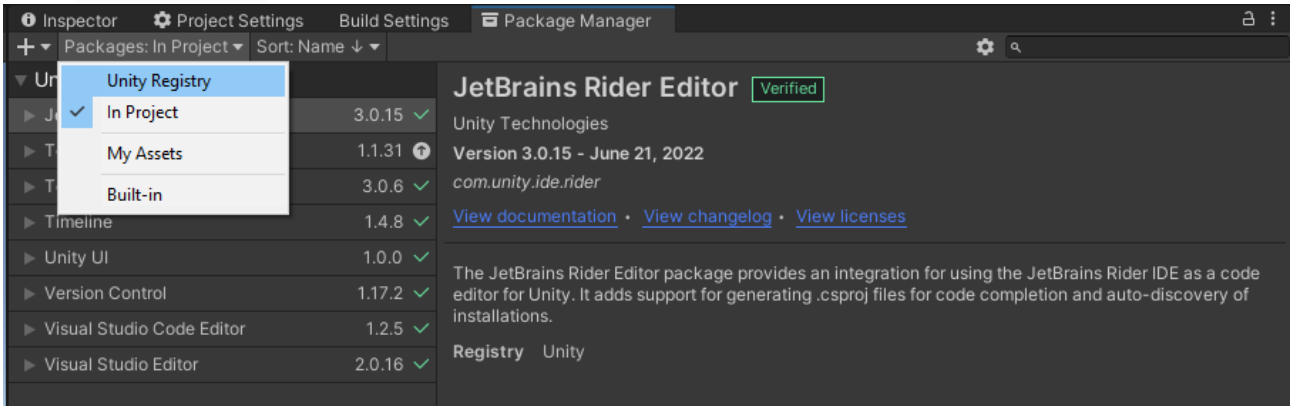
1. In `Unity Hub` click `New Project` in the top right and select a `3D project` and make sure you are using the correct editor version. Set the project name and location to your preference:



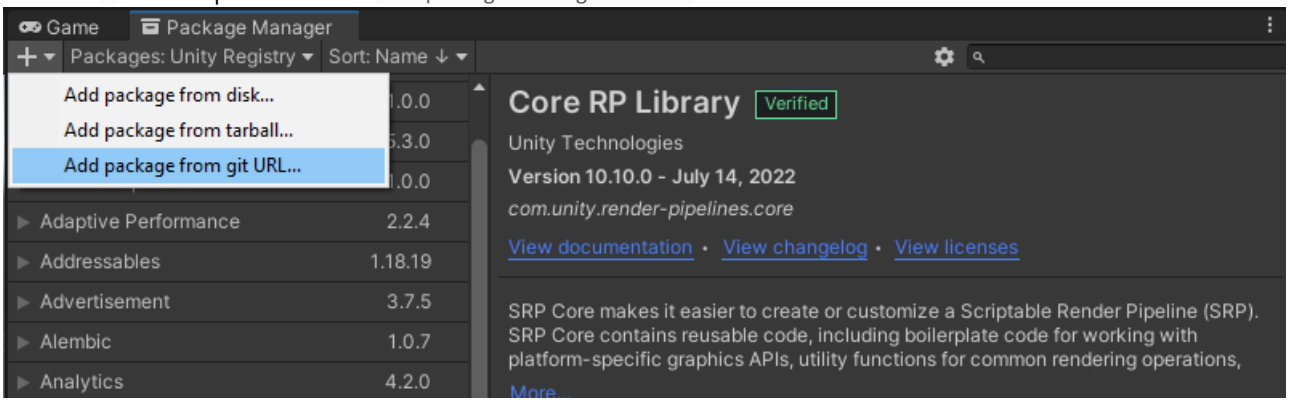
2. Open up the `package manager`:



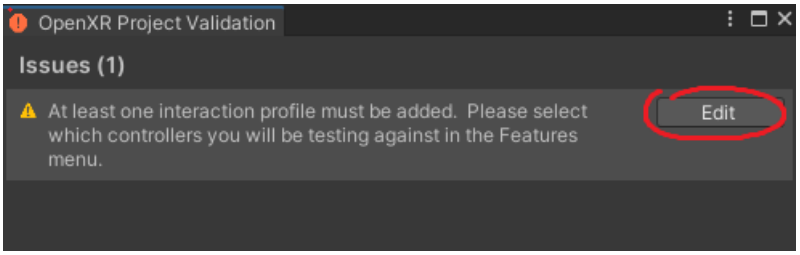
3. Select `Unity Registries`:



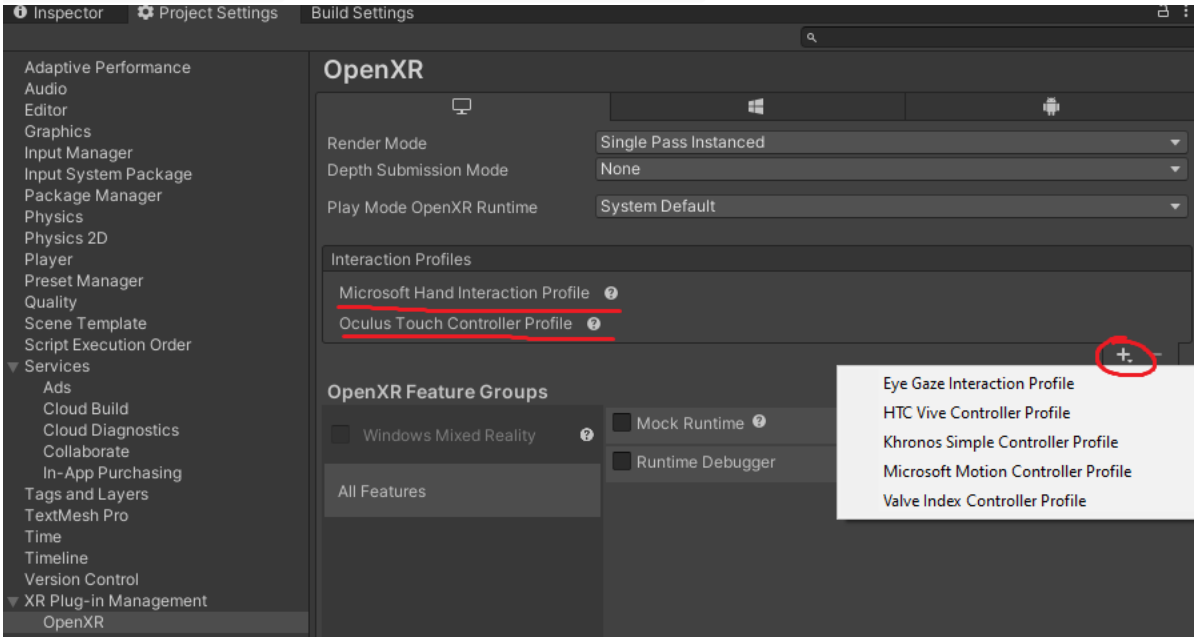
4. Click the `+` in the top left and click `Add package from git URL...`:



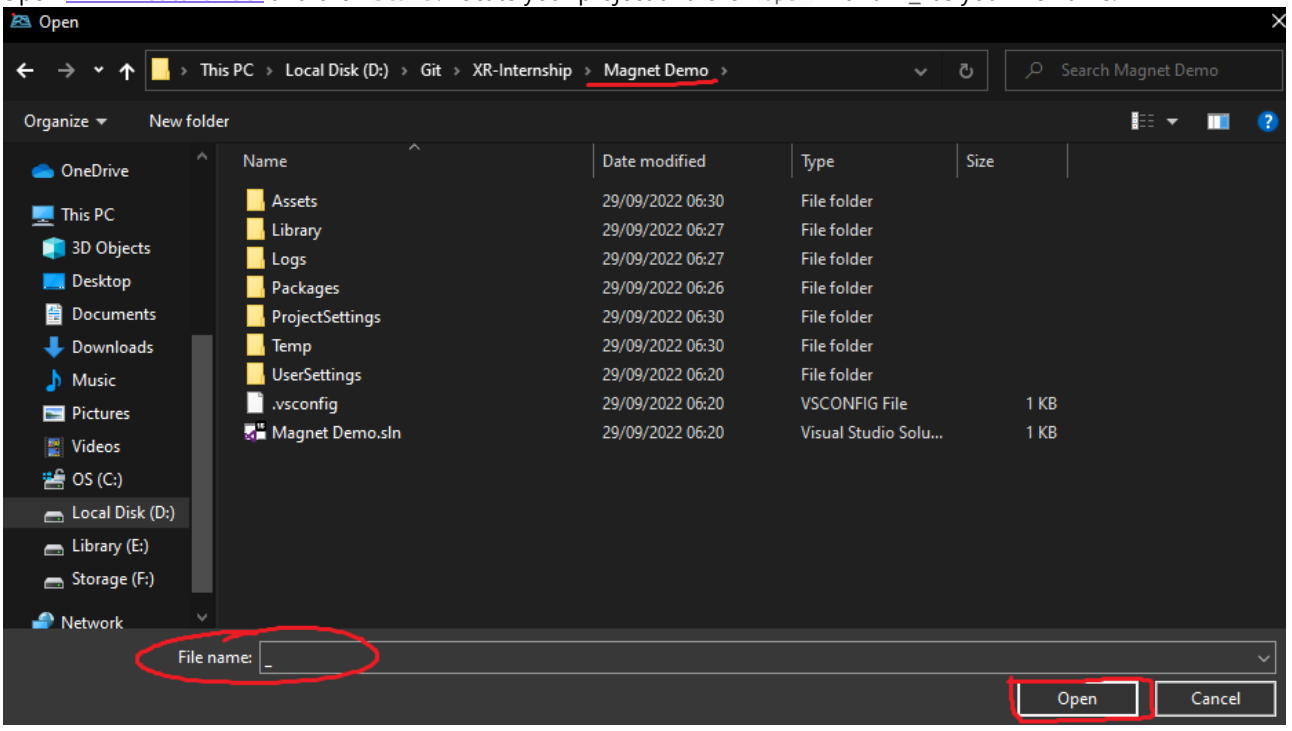
- Insert the following URLs one by one to install them:
  - `com.unity.render-pipelines.core`
  - `com.unity.render-pipelines.universal`
  - `com.unity.xr.openxr` (A popup will occure for turning on the old input system, click `yes`)
    - At the end a window will appear with an error, click `Edit`:



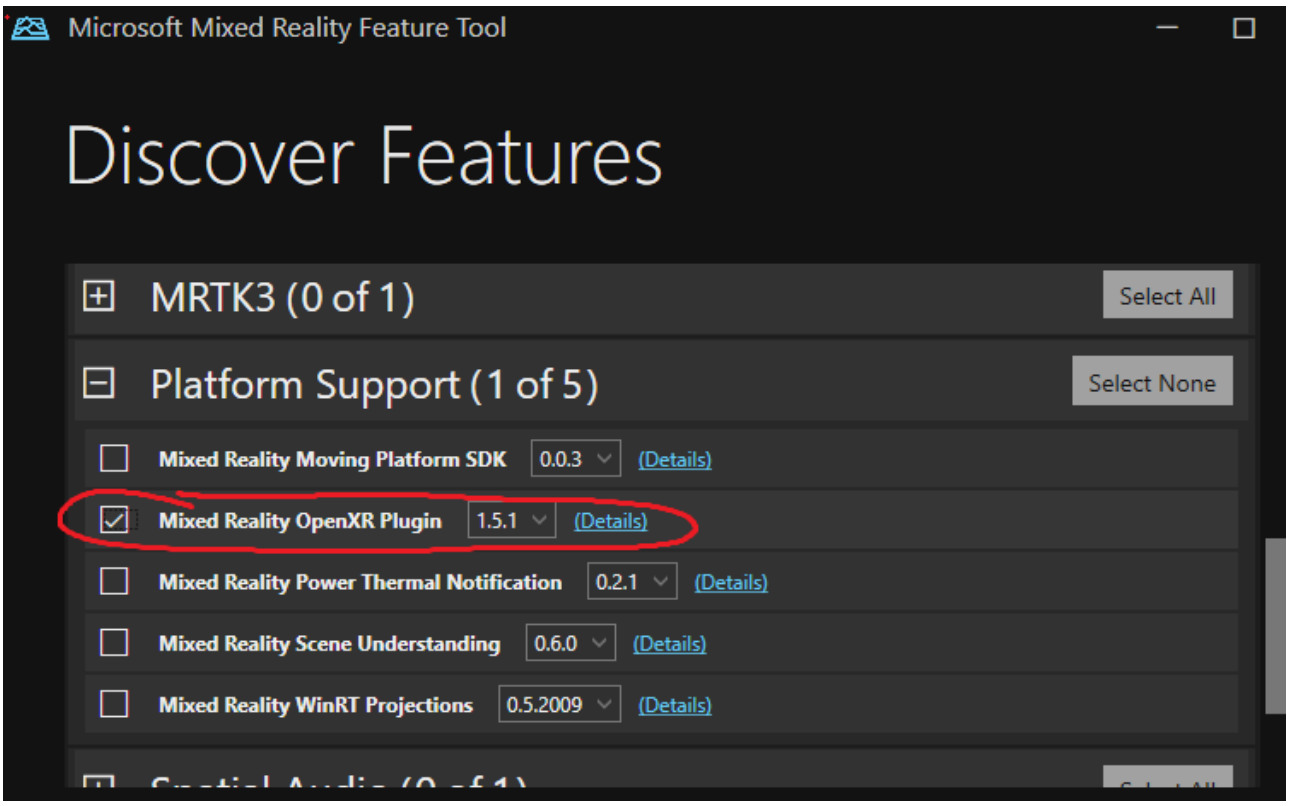
- Click the plus in the window that appears on the bottom right under `Interaction Profiles` and add `Microsoft Hand Interaction Profile` and `Oculus Touch Controller Profile`:

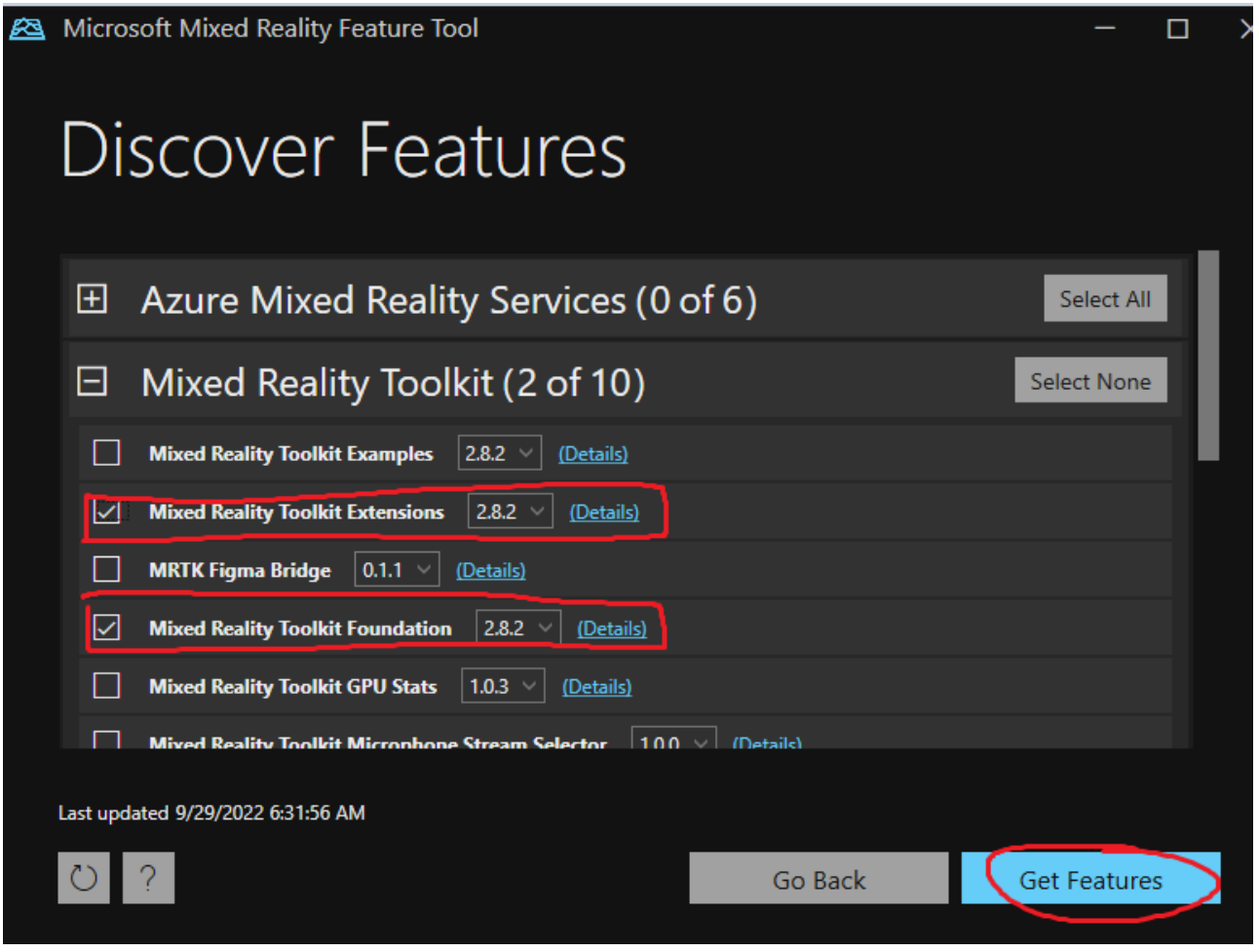


5. Open MRTK Feature Tool and click `Start`. Locate your project and click `open` with an `_` as your file name:

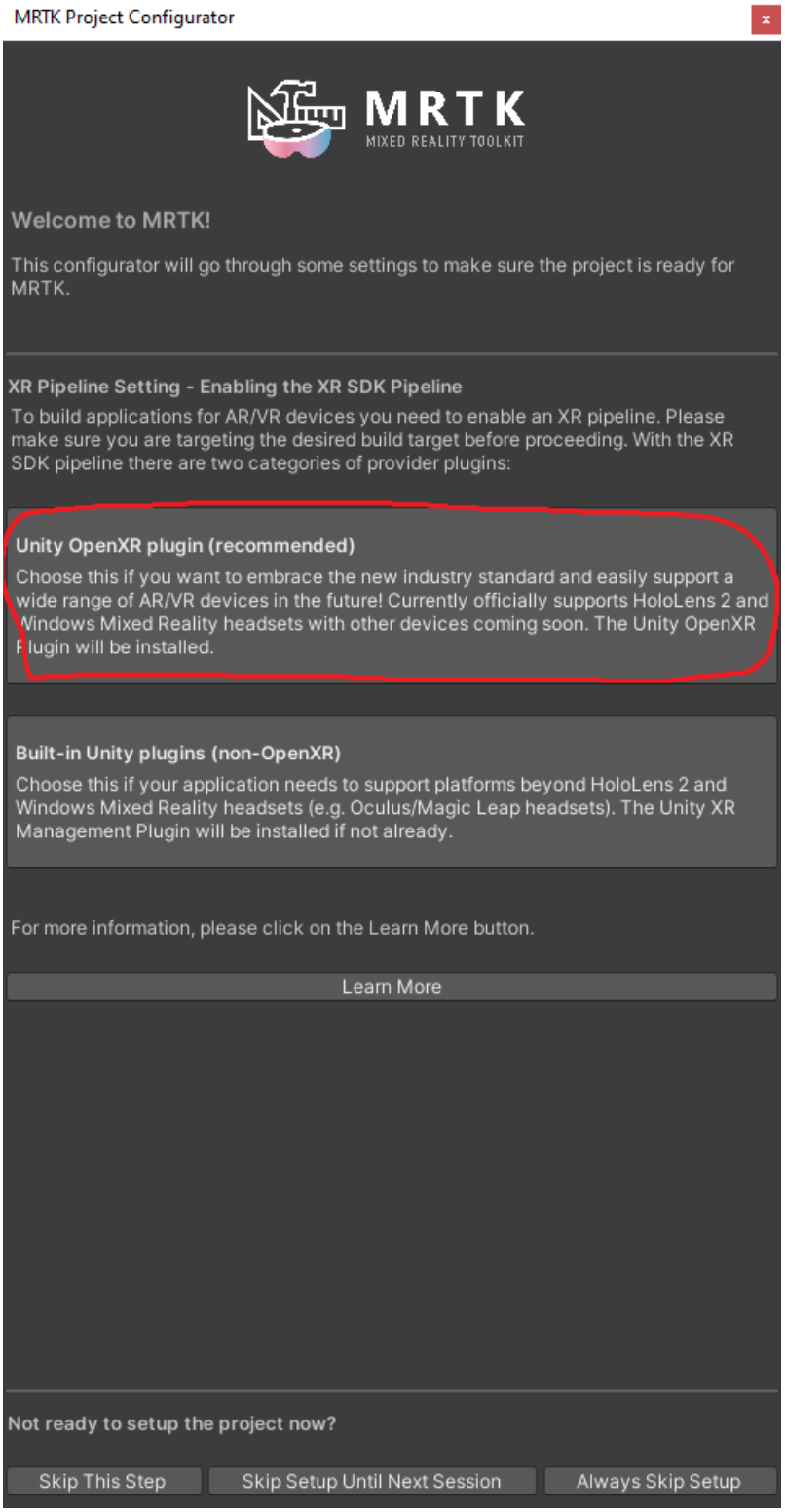


6. Click `Discover Features` and select `Mixed Reality Toolkit Extensions` and `Mixed Reality Toolkit Foundation` from `Mixed Reality Toolkit` header and `Mixed Reality OpenXR plugin` from `Platform Support`. Then click `Get Features`, `Import`, `Approve`:

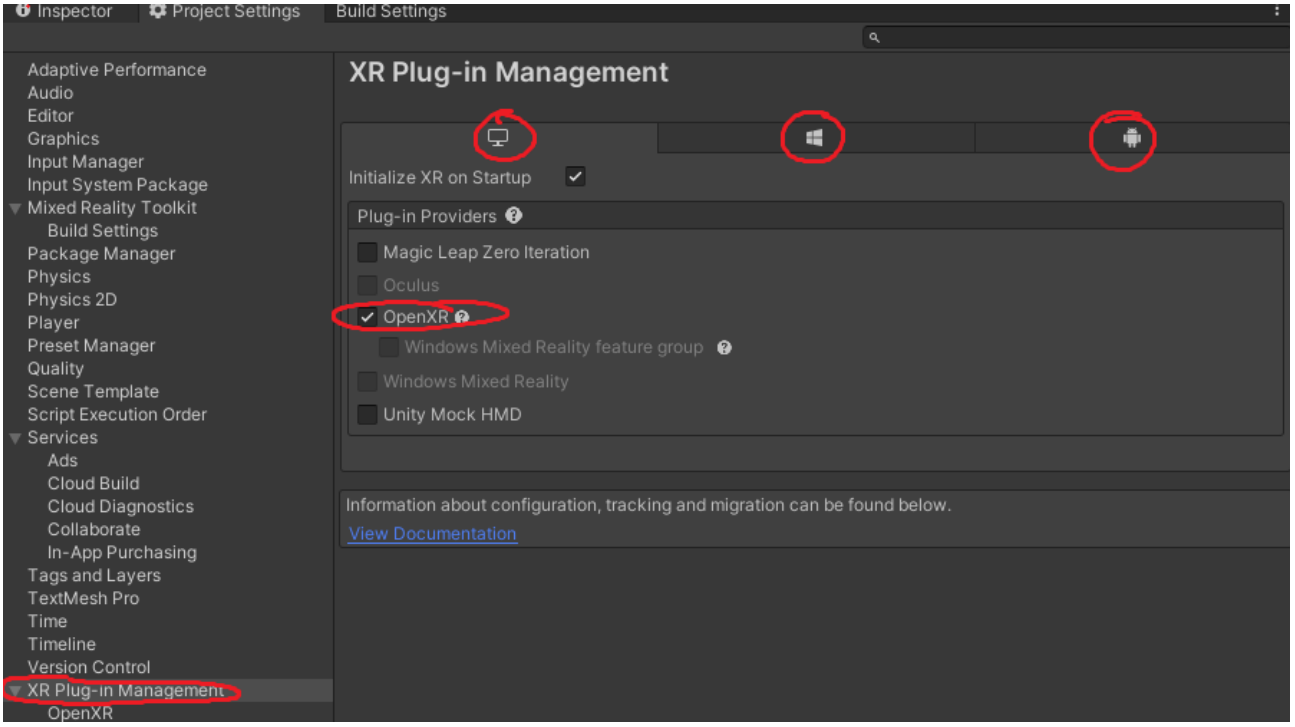




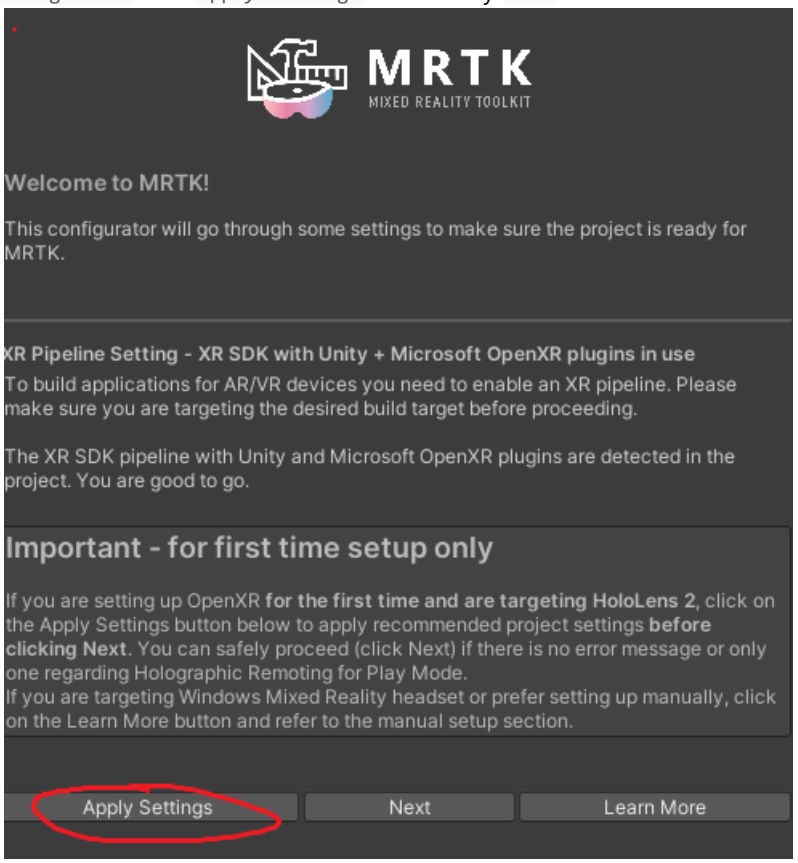
7. Close MRTK Feature Tool and switch back to Unity (It may take a while for unity to finish importing). Once completed, a popup should appear. Click `Unity OpenXR plugin`:



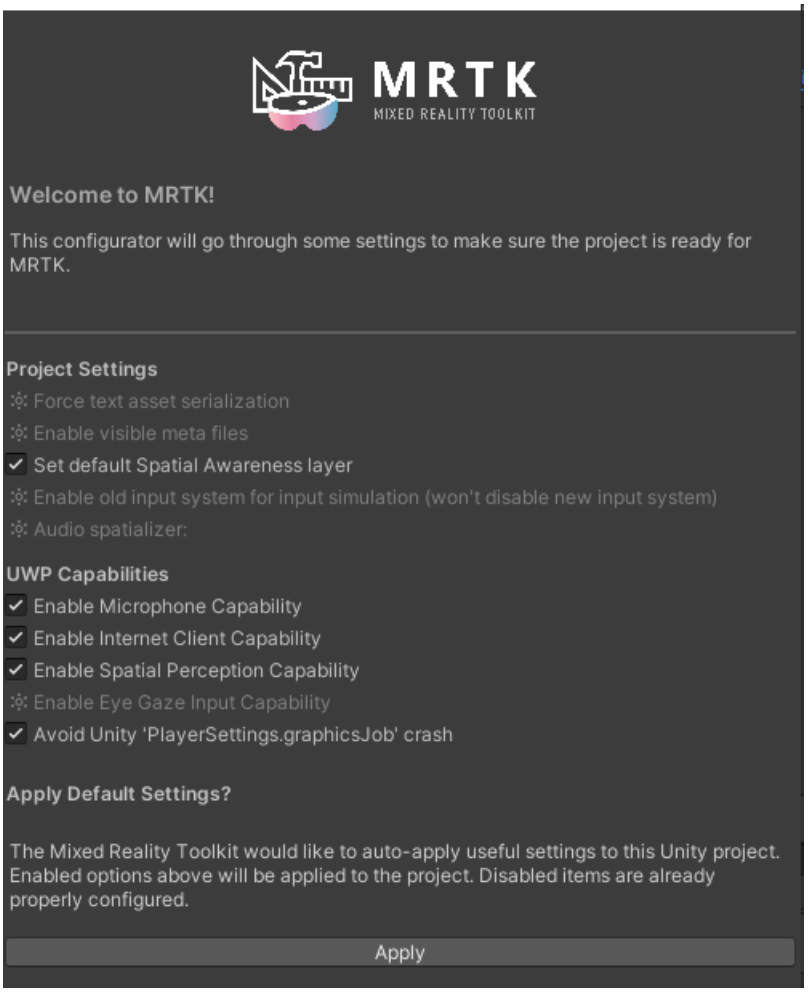
8. The `MRTK Project Configurator` should now prompt to click `Show settings`, this should navigate you to `Edit > Project Settings`. Select `XR Plug-in Management` and under each build target, tick `OpenXR`:



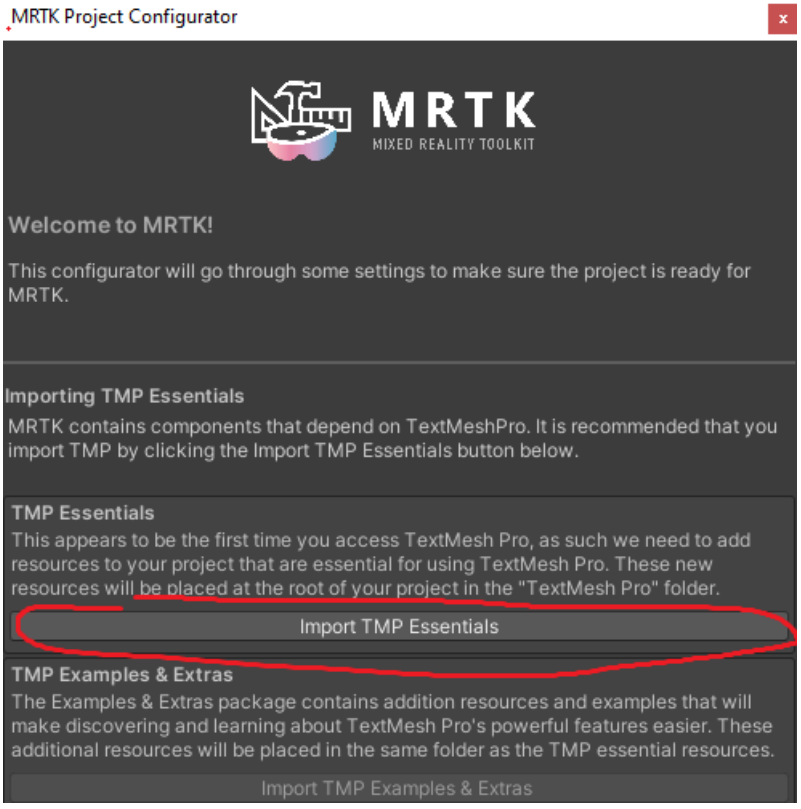
9. Back in `MRTK Project Configurator` click `Apply Settings` followed by `Next`



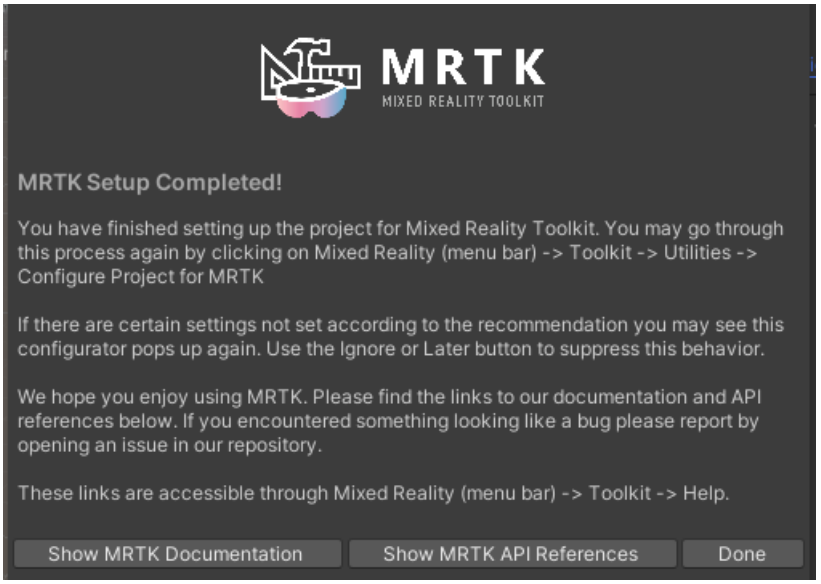
10. Click `Apply` then `Next`:



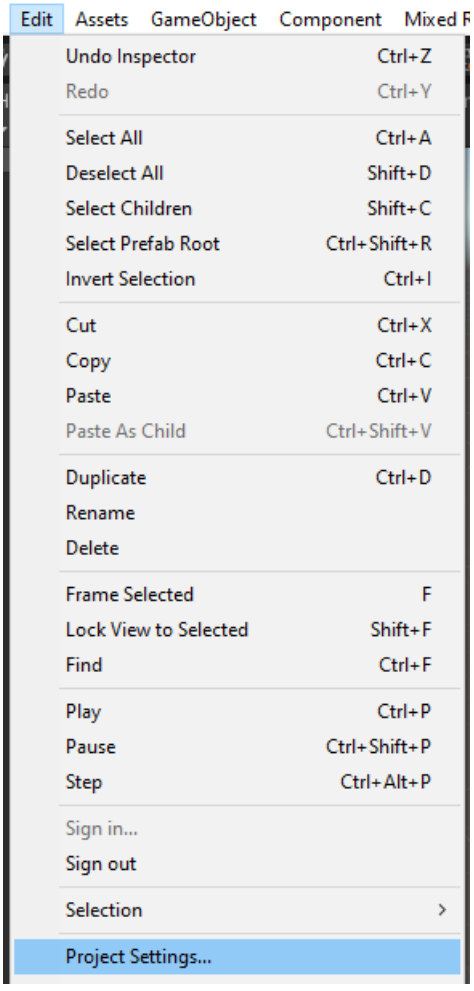
11. Click `Import TMP Essentials`:



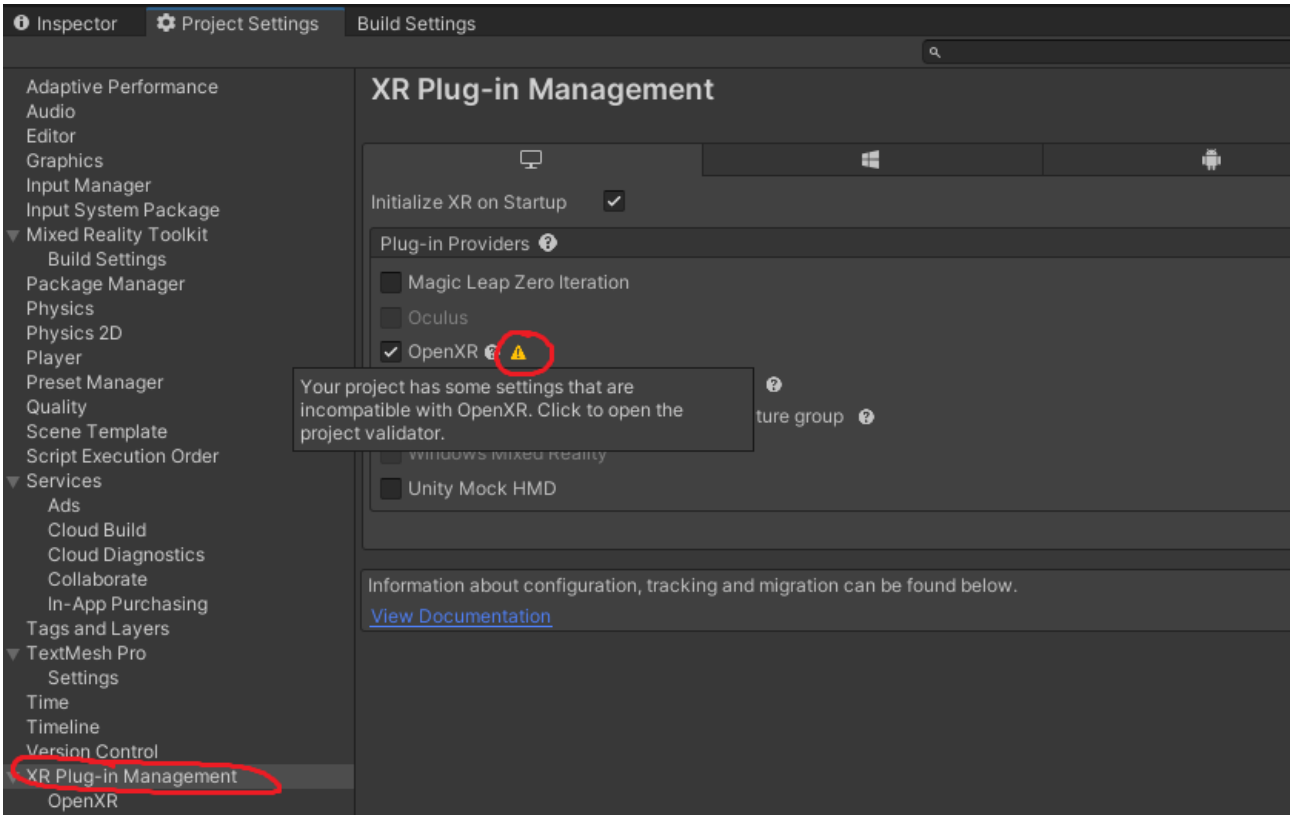
12. Click `Done`:



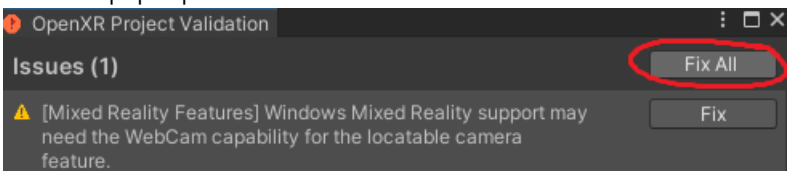
13. Navigate to `Edit > Project Settings`



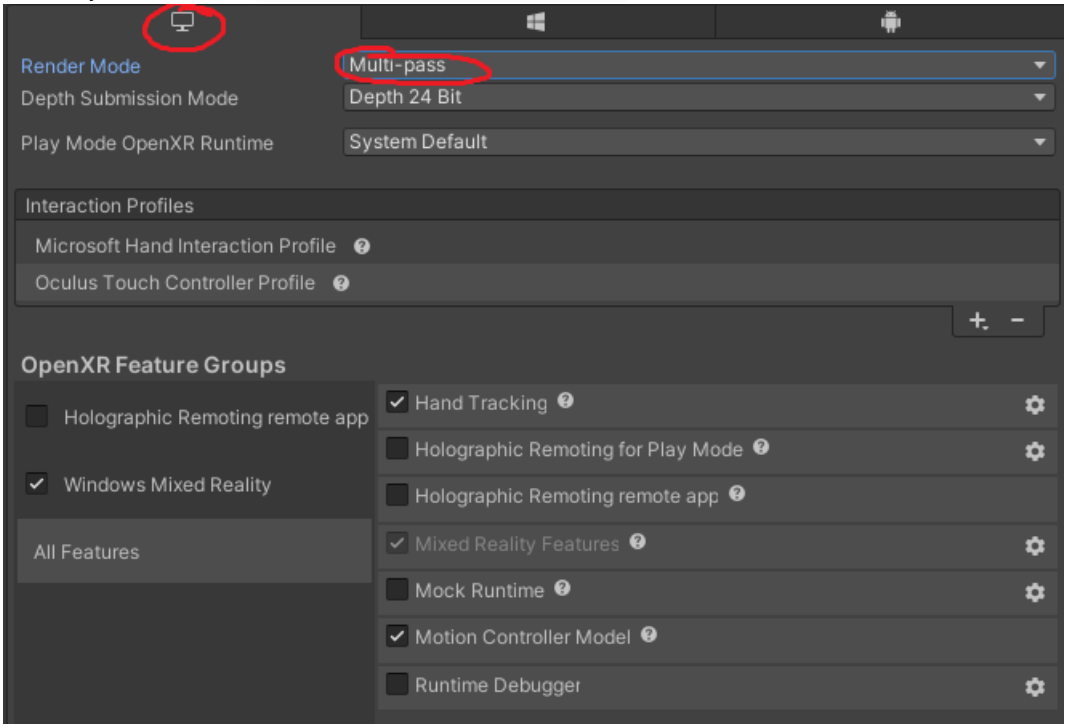
14. In `XR Plug-in Management` click the little warning symbol:

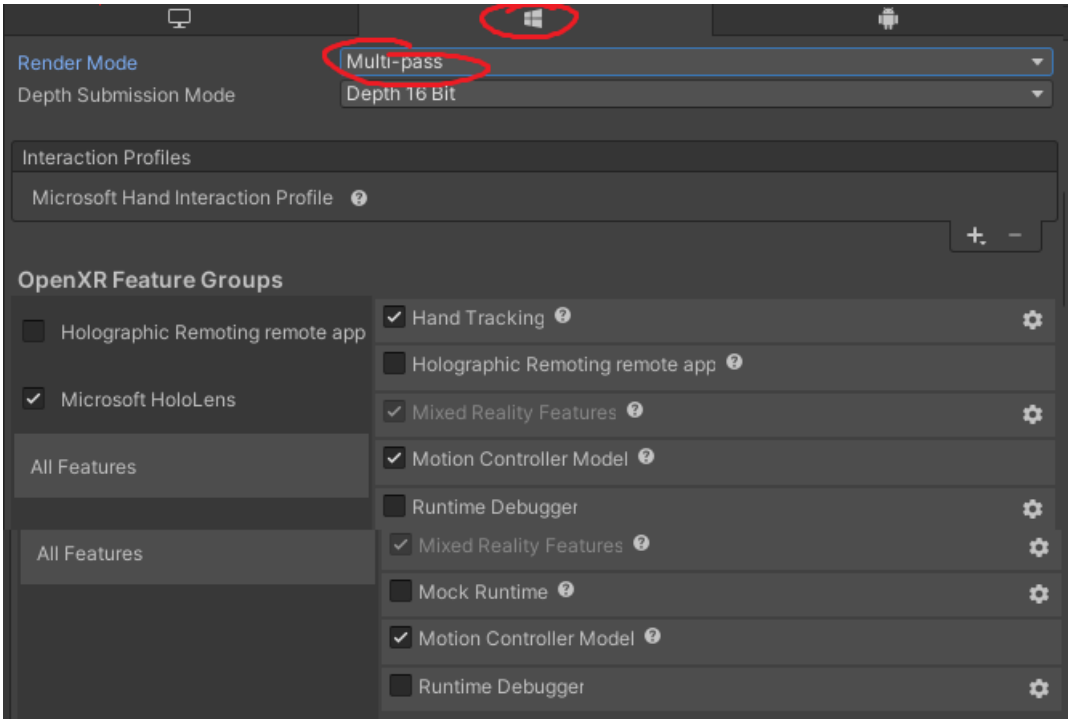


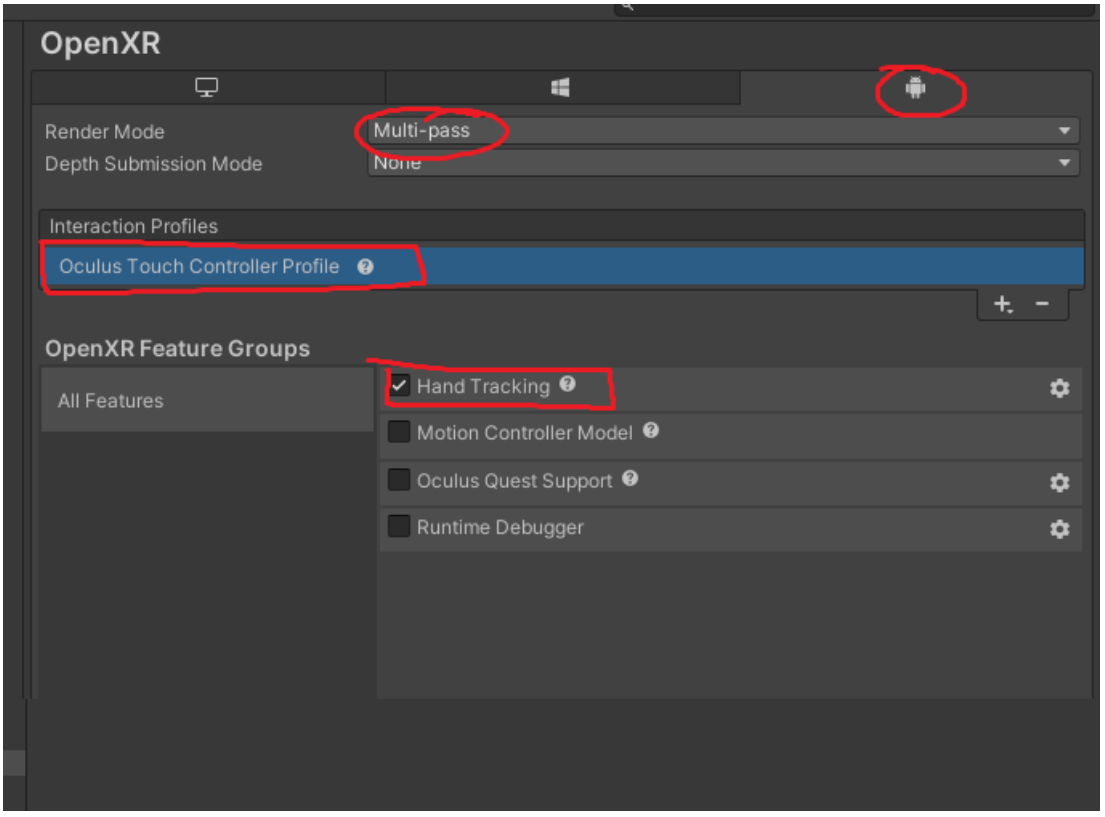
15. Click `Fix All` in the window that pops up.



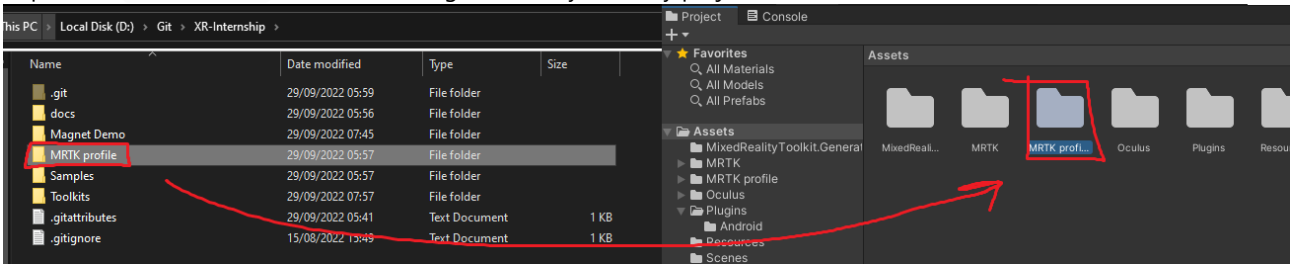
16. Repeat steps *14* and *15* for the other 3 build targets and set their `Render Mode` to `Multi-pass` such that they look like so (Android will need you to add the `oculus touch interaction profile`):



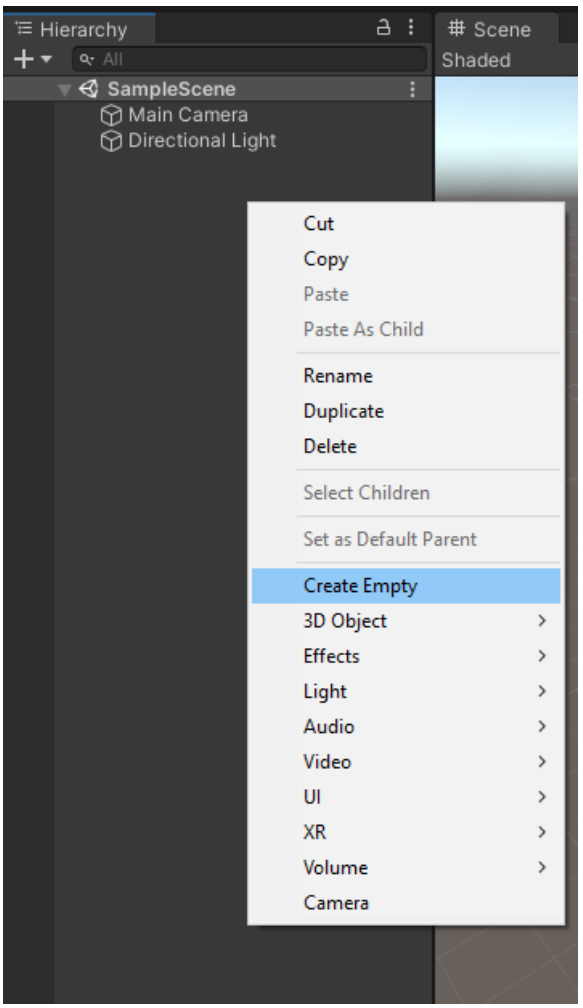




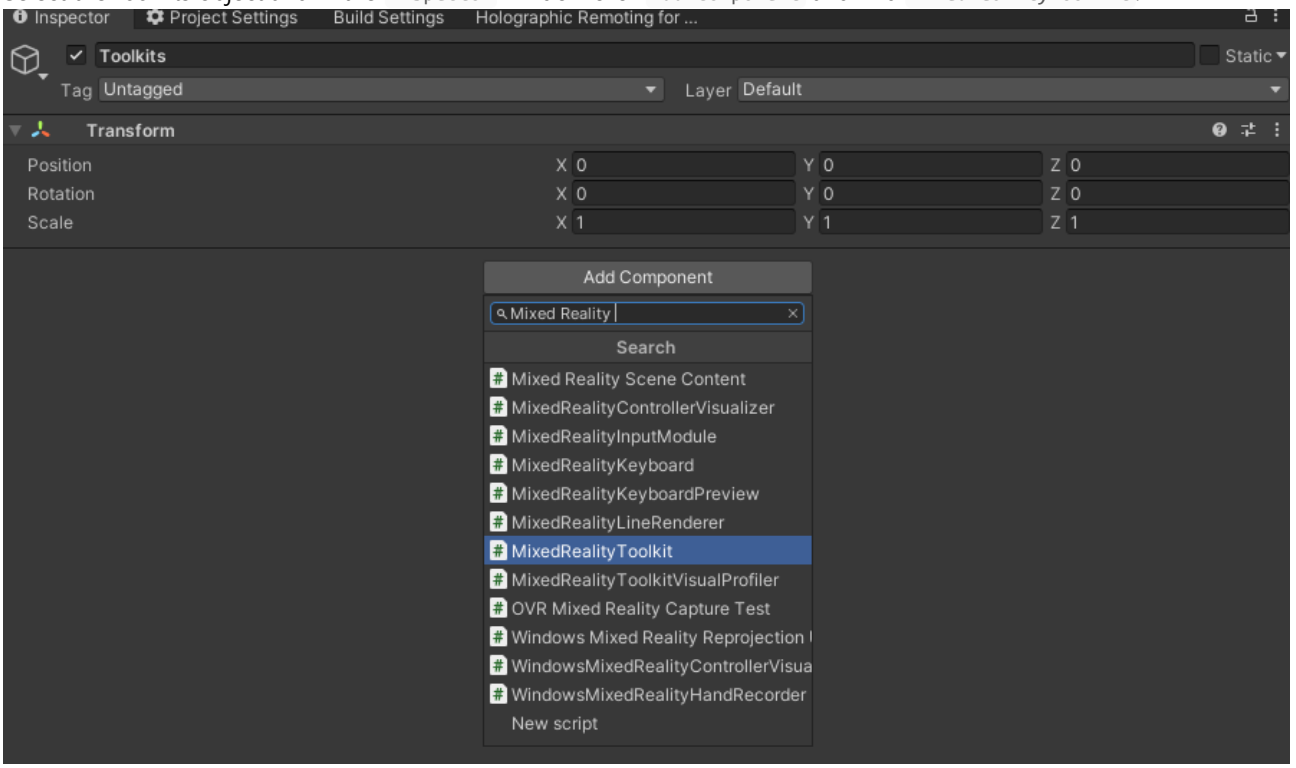
17. Import the `MRTK Profile` folder from the github into your unity project:



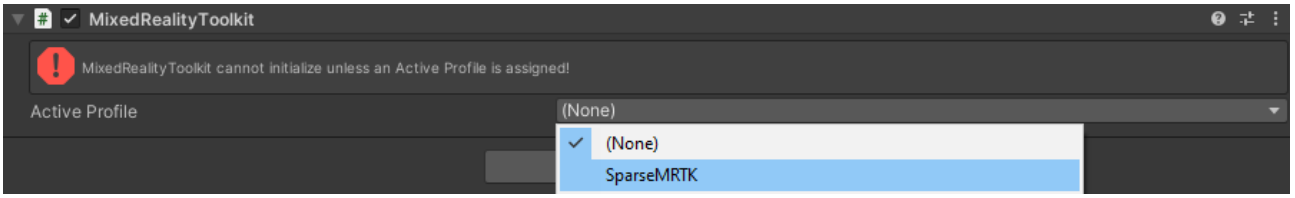
18. In your scene right click and click `Create Empty`, name the object *Toolkits*:



19. Select the *Toolkits* object and in the `Inspector` window click `Add Component` and find `MixedRealityToolkit`:



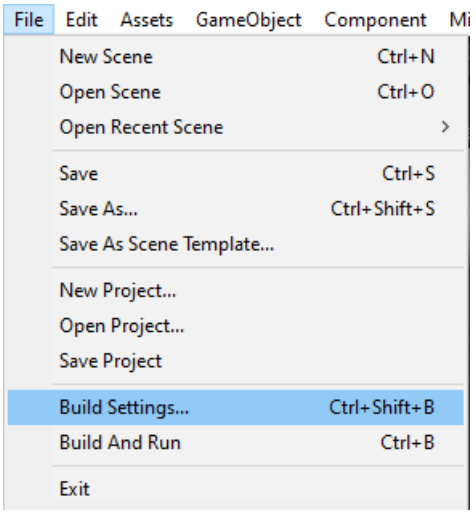
20. Change `Active Profile` to `Sparse MRTK`:

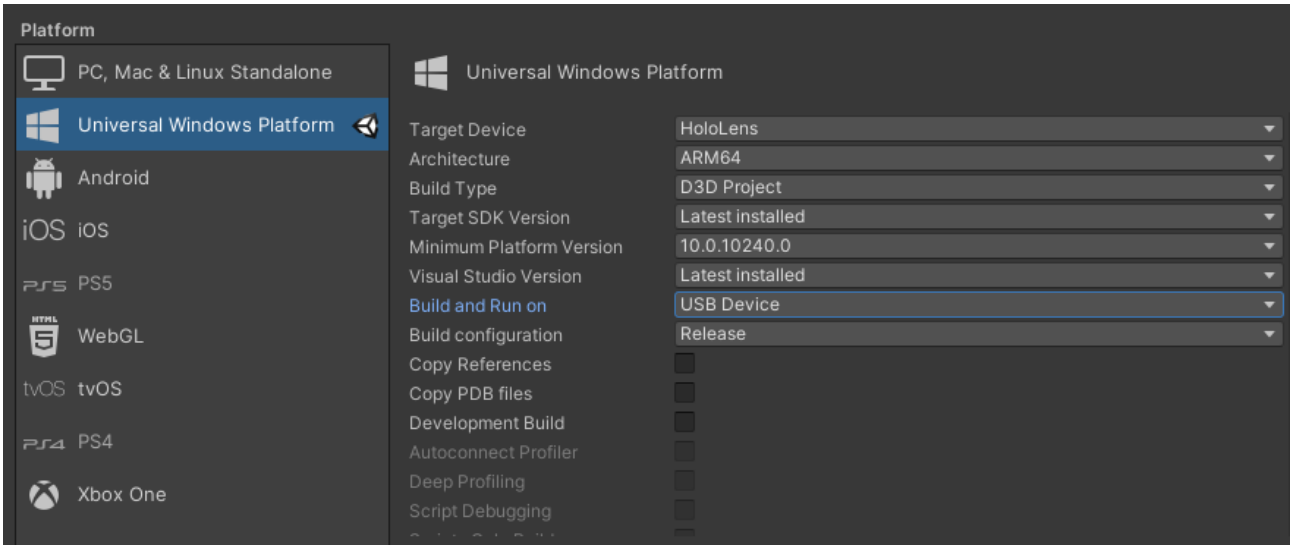


NOTE:: Perform both setups below only if you want your project to be switchable between both devices without having a different project for each device. Otherwise only perform the setup for your chosen device.
NOTE:: It is not recommended to prepare a single project for both Hololens and Oculus because it creates large project files with no crossover resulting in large executables.

Setup for hololens

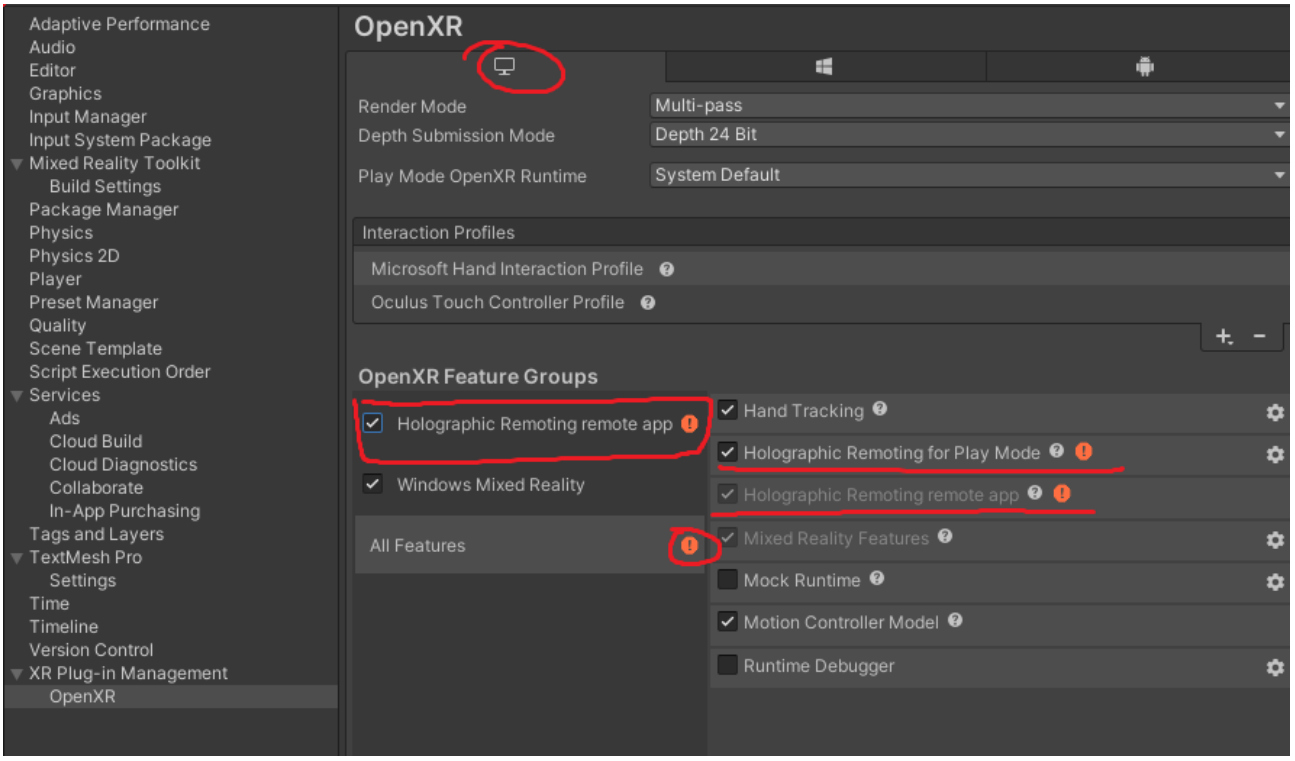
1. Navigate to `File > Build Settings` and select `Universal Windows Platform`. Click `Switch platform` on the bottom right and make sure the settings are the same as shown here:

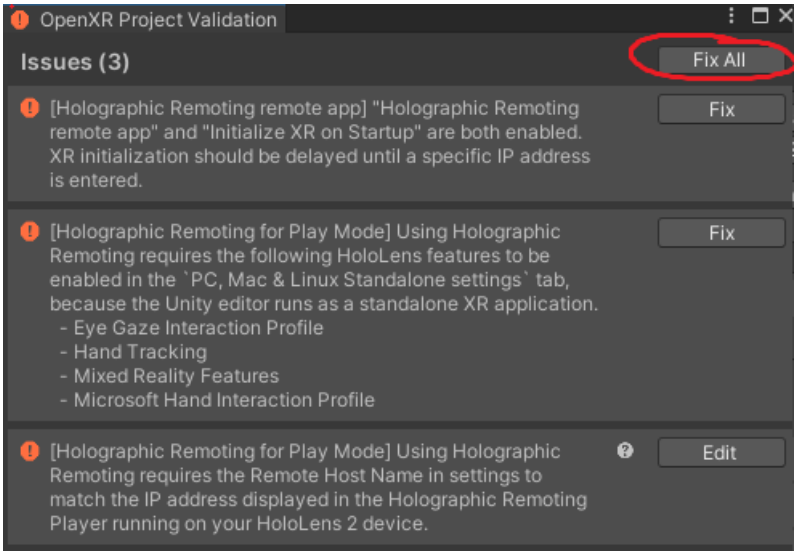




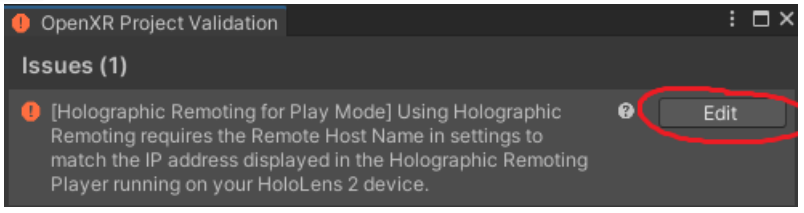
Play Mode

1. To allow for play mode in unity, download the `Holoremoting app` on the hololens from the microsoft store.
2. In `Project Settings > XR Plug-in Management` under desktop build target, select `holoremoting` and then click the red alert and click `Fix All`.

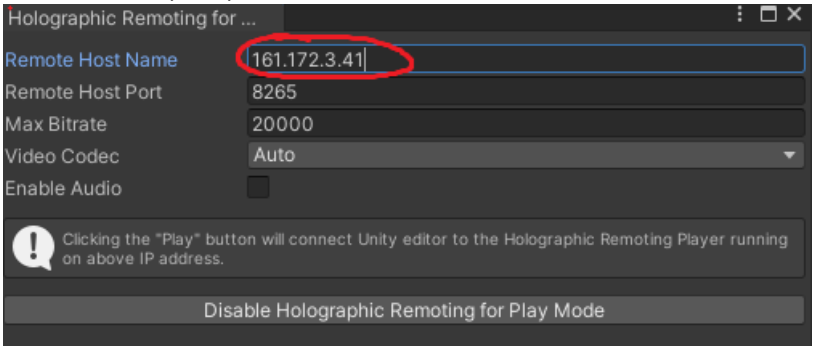




3. On the last error click `edit`:

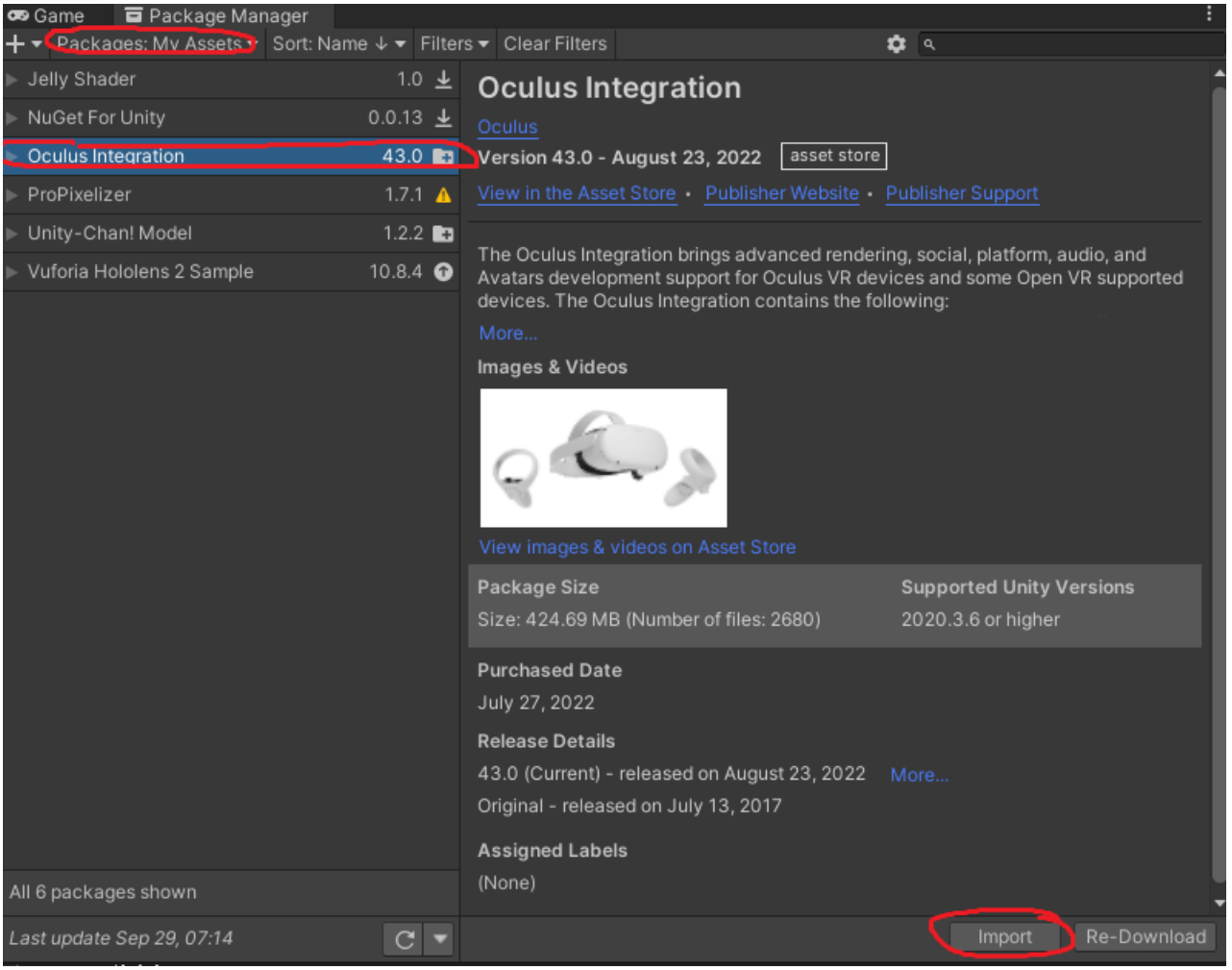


4. In the new window that popped up, type the IP address of the hololens provided by the hololens remoting app on the hololens (Boot up the app on the headset and it will give you an IP. If the hololens is not connected to the internet, providing a wired connection will also prompt an IP which can be used):

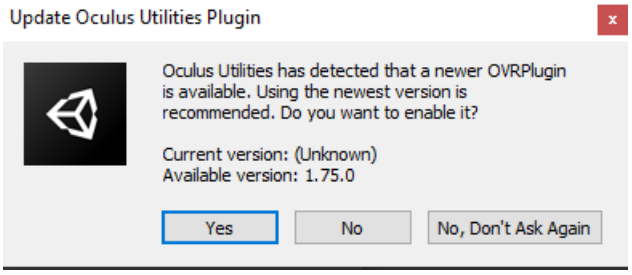


Setup for oculus

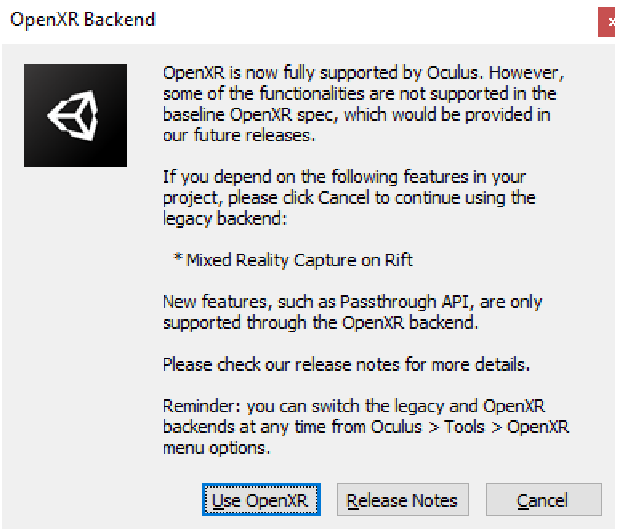
1. Via the package manager, import the Oculus Integration package and download via git url: `com.unity.xr.oculus`.
   NOTE:: You will need to go the unity asset store and download the asset onto your unity account so that it shows up under `My Assets`



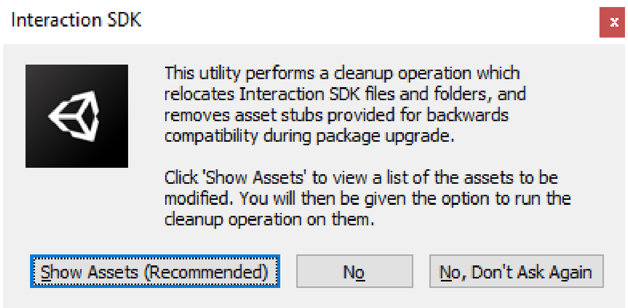
   NOTE:: If Unity asks what specific files to import, simply select all and click `Import`
2. When prompted to update `OVR` click `yes`:



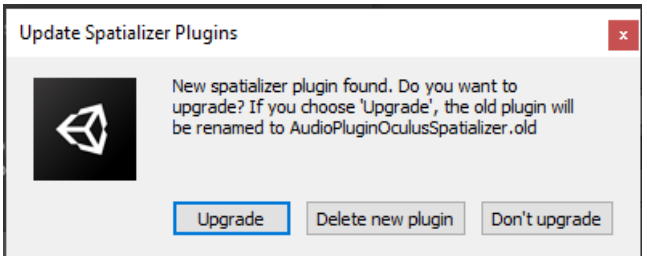
3. When prompted to use `OpenXR` click `Use OpenXR`:



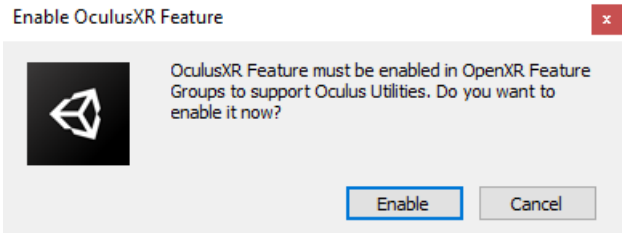
4. When prompted to restart Unity click `restart`.
5. When prompted to cleanup assets click `Show Assets`, the click `Clean up`:



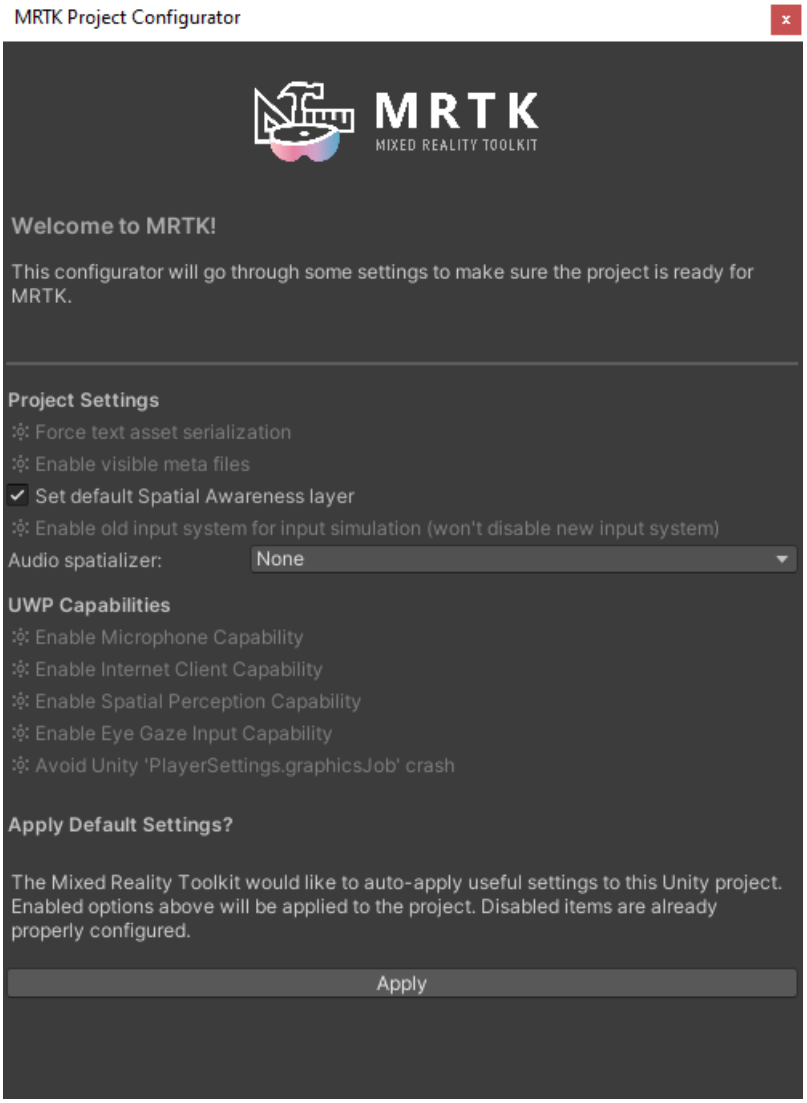
6. When prompted to update `Spatializer plugin` click `upgrade`:



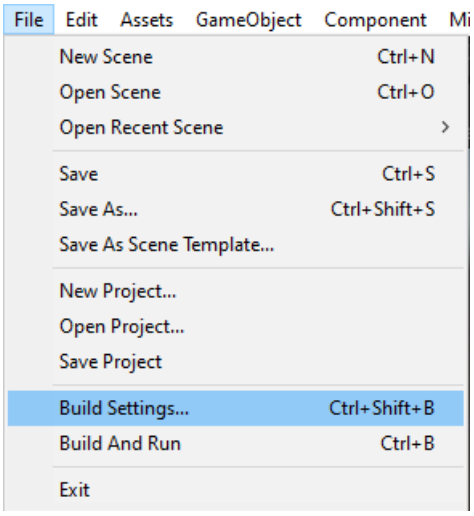
7. When prompted to restart unity click `restart`.
8. When prompted to enable Oculus XR click `enable`:

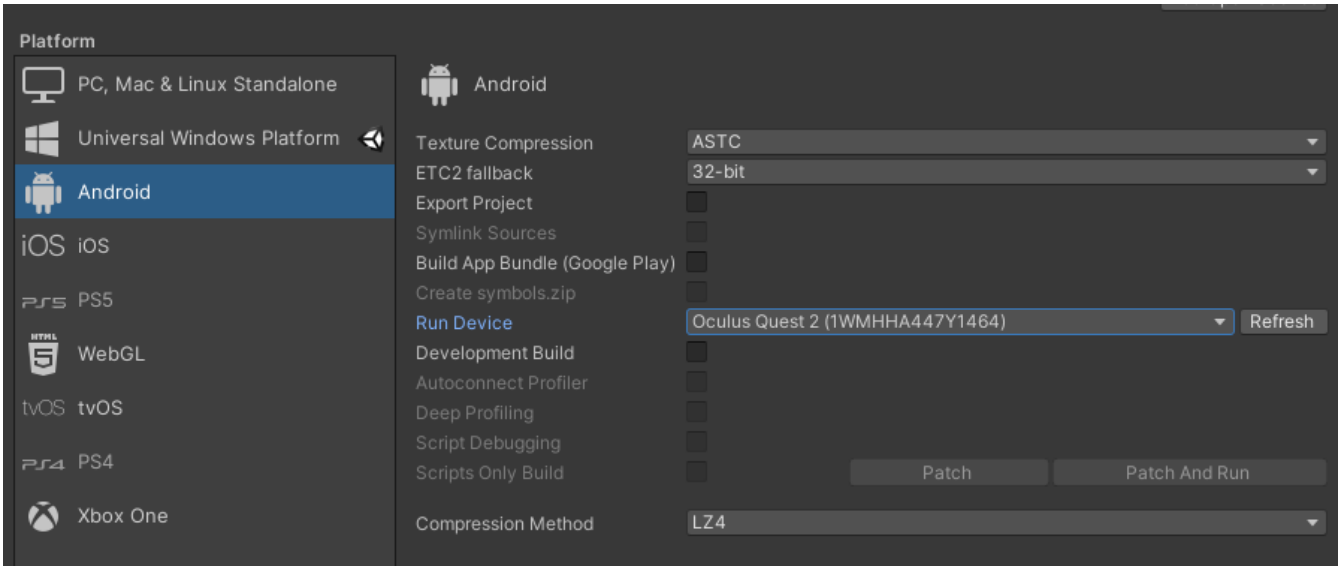


9. If MRKT prompts you to apply `Spatial Awareness layer`, just click `Apply`, `Next` and then `Done`:



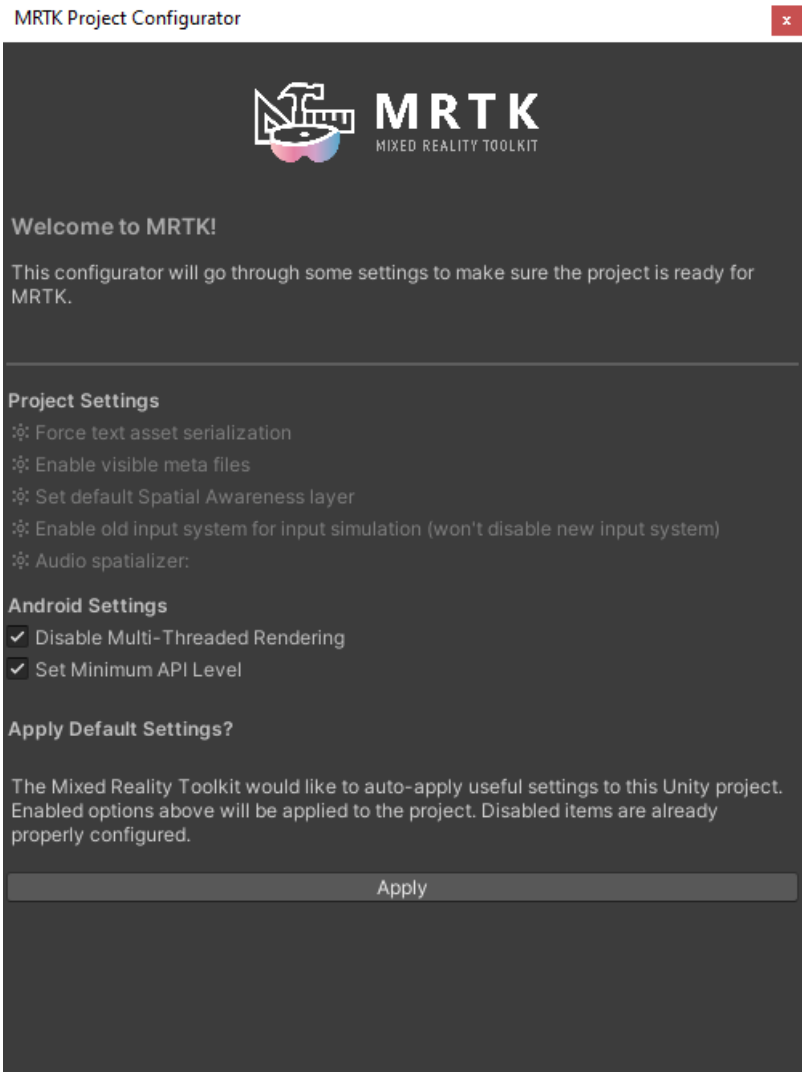
10. Plug in your Oculus Quest 2 headset and make sure it is connected via the Oculus Desktop App.
11. Navigate to `File > Build Settings` and select `Android`. Click `Switch platform` on the bottom right and make sure the settings are the same as shown here:



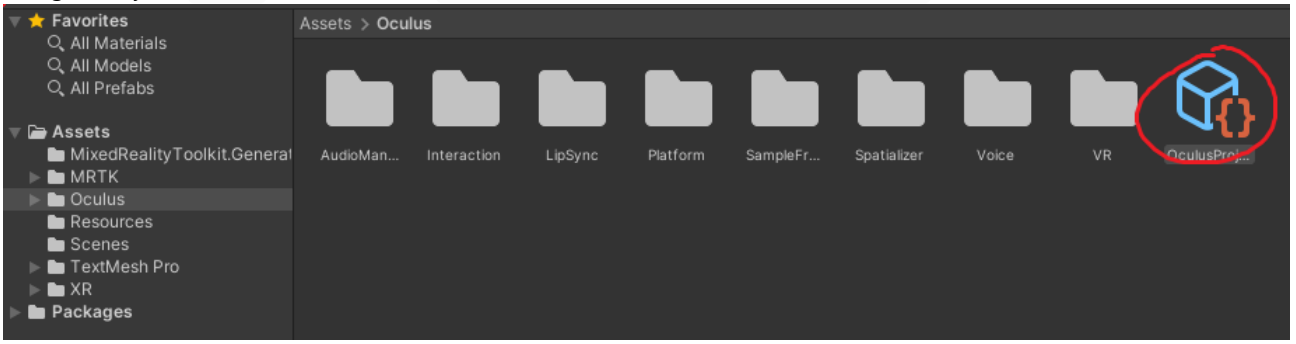


NOTE:: If your headset does not showup under `Run Device` double check if USB debugging is enabled on the headset and that developer mode was enabled for your oculus account.

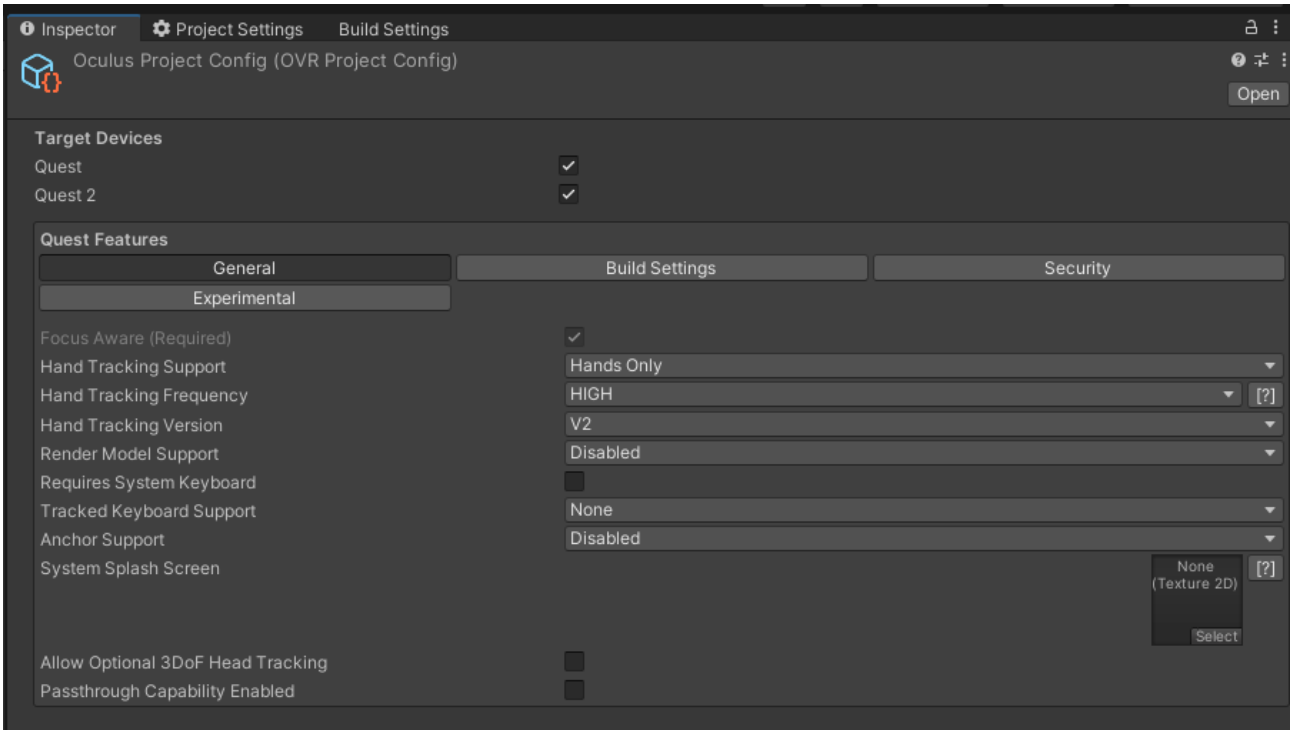
12. Upon switching platform to `Android` you may be prompted by MRTK to apply android settings. If so, click `Apply`, `Next`, `Done`:



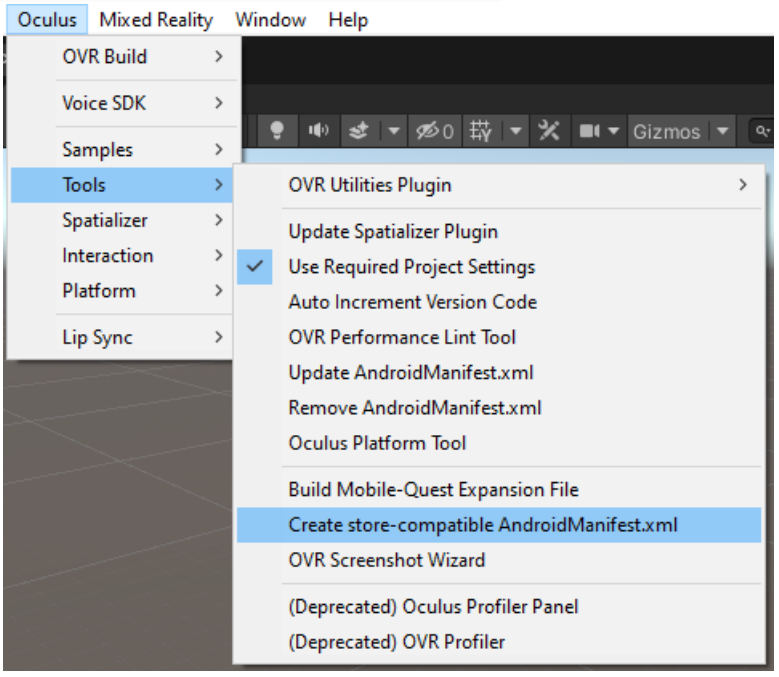
13. Navigate to your `Oculus` folder and find `Oculus Project Config (OVR Project Config)` asset.



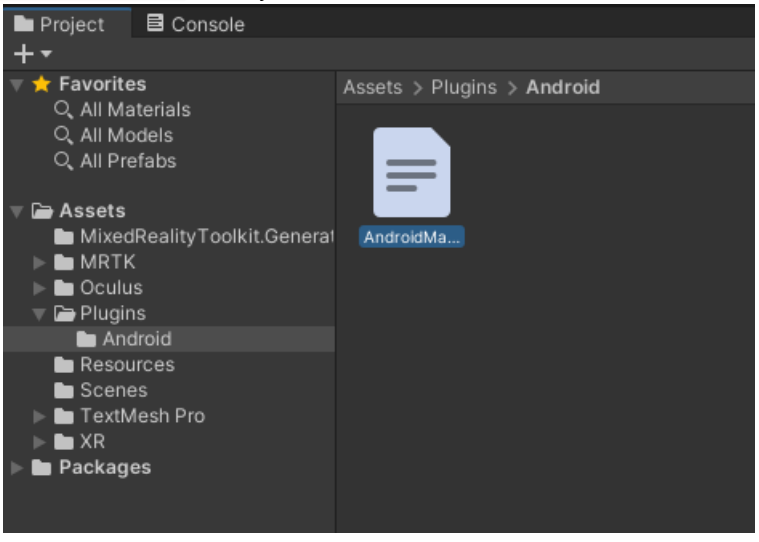
14. After selecting it, in the `inspector` window change the settings to match:



15. Click `Oculus > Tools > Create store-compatible AndroidMainfest.xml`:



16. Locate and open the `AndroidManifest.xml` file in your assets:



17. Add the denoted lines to the file:

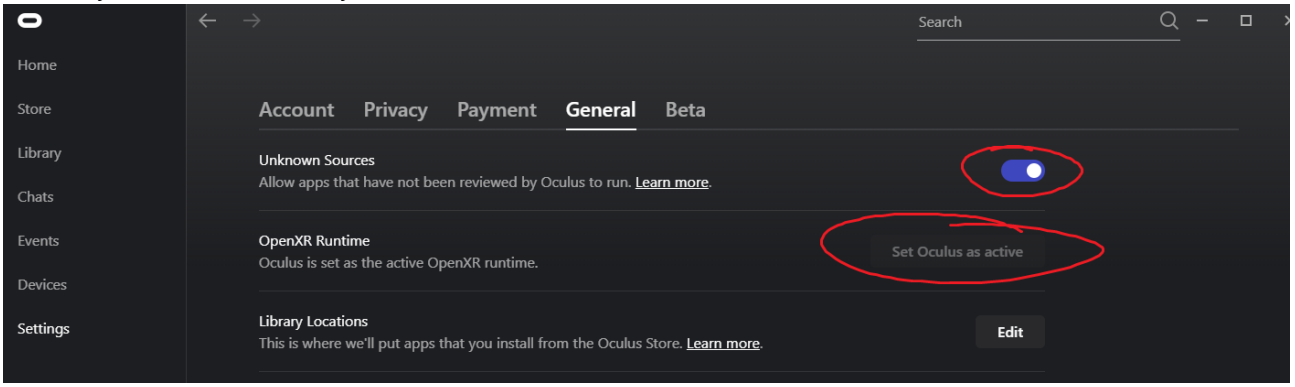
```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:installLocation="auto">
  <application android:label="@string/app_name" android:icon="@mipmap/app_icon" android:allowBackup="false">
    <activity android:theme="@android:style/Theme.Black.NoTitleBar.Fullscreen"
android:configChanges="locale|fontScale|keyboard|keyboardHidden|mcc|mnc|navigation|orientation|screenLayout|scr
eenSize|smallestScreenSize|touchscreen|uiMode" android:launchMode="singleTask"
android:name="com.unity3d.player.UnityPlayerActivity" android:excludeFromRecents="true">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
        <category android:name="com.oculus.intent.category.VR" />
      </intent-filter>
      <meta-data android:name="com.oculus.vr.focusaware" android:value="true" />
    </activity>
    <meta-data android:name="unityplayer.SkipPermissionsDialog" android:value="false" />
    <meta-data android:name="com.samsung.android.vr.application.mode" android:value="vr_only" />
    <meta-data android:name="com.oculus.handtracking.frequency" android:value="HIGH" />
    <meta-data android:name="com.oculus.handtracking.version" android:value="V2.0" />
    <meta-data android:name="com.oculus.supportedDevices" android:value="quest|quest2" />
  </application>
  <uses-feature android:name="android.hardware.vr.headtracking" android:version="1" android:required="true" />
  <uses-feature android:name="oculus.software.handtracking" android:required="true" />
  <uses-permission android:name="com.oculus.permission.HAND_TRACKING" />

  <!-- ADD THESE 2 LINES HERE -->
  <uses-permission android:name="android.permission.INTERNET" />
  <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>

</manifest>
```

Play Mode

1. On the oculus headset navigate to `Quest Link` and then simply click play in Unity whenever.
2. In the oculus desktop app go to settings and enable `Unknown sources` and set oculs as active `OpenXR Runtime`:
   NOTE:: If you do not do this, Unity will crash.



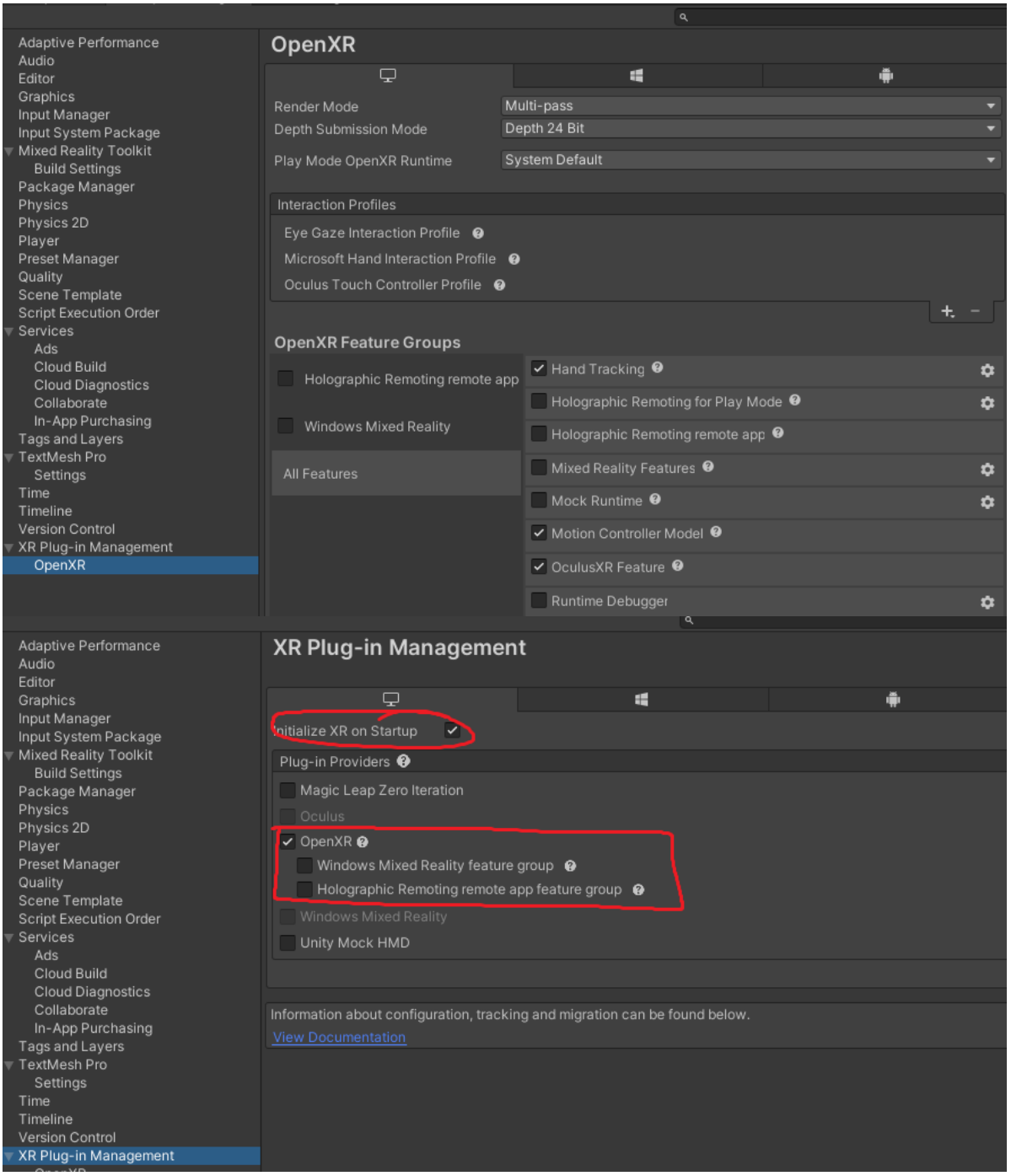
Switching between hololens and oculus

NOTE:: only relevant if you performed both setups for hololens and oculus.

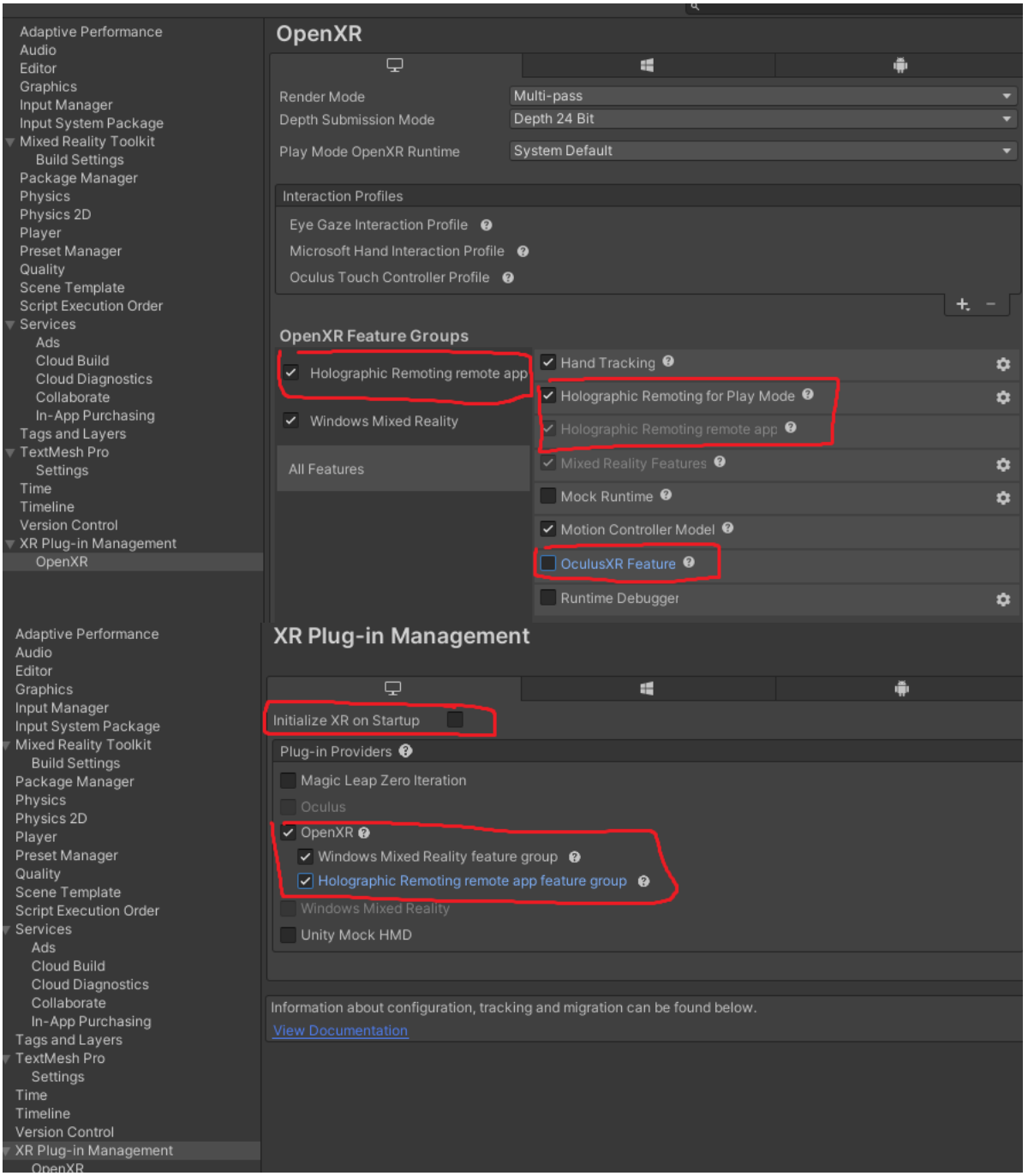
- For builds, simply switch build target (`Android` for oculus and `Universal Windows Platform` for hololens).

- For play mode:

  - Hololens -> Oculus
    1. In project settings disable holographic remoting and enable `OculusXR Feature`



- Oculus -> Hololens
    1. In project settings enable holographic remoting and disable `OculusXR Feature`


Adaptive Performance
Audio
Editor
Graphics
Input Manager
Input System Package
Mixed Reality Toolkit
Build Settings
Package Manager
Physics
Physics 2D
Player
Preset Manager
Quality
Scene Template
Script Execution Order
Services
Ads
Cloud Build
Cloud Diagnostics
Collaborate
In-App Purchasing
Tags and Layers
TextMesh Pro
Settings
Time
Timeline
Version Control
XR Plug-in Management
OpenXR
OpenXR
Render Mode
Multi-pass
Depth Submission Mode
Depth 24 Bit
Play Mode OpenXR Runtime
System Default
Interaction Profiles
Eye Gaze Interaction Profile
Microsoft Hand Interaction Profile
Oculus Touch Controller Profile
OpenXR Feature Groups
Holographic Remoting remote app
Windows Mixed Reality
All Features
Hand Tracking
Holographic Remoting for Play Mode
Holographic Remoting remote app
Mixed Reality Features
Mock Runtime
Motion Controller Model
OculusXR Feature
Runtime Debugger
XR Plug-in Management
Initialize XR on Startup
Plug-in Providers
Magic Leap Zero Iteration
Oculus
OpenXR
Windows Mixed Reality feature group
Holographic Remoting remote app feature group
Windows Mixed Reality
Unity Mock HMD
Information about configuration, tracking and migration can be found below.
View Documentation