OCULUS PROJECT SETUP (USING OPENXR)

1. Make sure Unity latest version (2020.3.8f1 is the one I used) and Android modules are installed (no need for Android Studio; easier this way)

2. Make sure in Oculus App in PC -> Settings -> General -> Allow Unknown Sources is On

3. Make sure Developer Mode is on in Oculus App on phone

4. Make the Quest 2 active (Devices -> click on Quest 2 -> Set As Active)

SETTING UP THE BACKEND

1. Switched to URP (instead of HD pipeline or VRCore) – *personal preference*

2. New Project

3. Edit -> Project Settings -> Package Manager -> Enable Preview Packages -> Click "I Understand"

4. Click **Window -> Package Manager**

5. Click on **Packages: In Project** (near the plus icon on the top left) and change it to **Unity Registry**

6. Find OpenXR Plugin (should be somewhere in the middle) -> Click Install. After a while it will ask you, do you want to Restart. Click "Yes". Wait for Unity to restart.

OpenXR plugin basically switches Unity's old way of handling inputs where now we have a generic input system that can be custom tailored to be deployed to any device (Quest, Rift, Vive, etc)

7. Still in the Package Manager, scroll to the bottom, and select **XR Interaction Toolkit**. Click Install.

This contains a lot of the pre-built movement systems and interaction systems.

8. Still in the same window, expand "Samples" and Import "Default Input Actions". It might give you a warning. Ignore it for now.

9. Go to **Edit -> Project Settings -> XR Plug-in Management**. Here follow these steps:

    9.1. In the PC tab (hover over the icons to show their names), click OpenXR. There will be a tick mark and a red exclamation mark. **Click on the red exclamation mark and click "Fix All"**. It is ok if it does not fix one warning.

    9.2. Click on Features underneath XR Plug-in Management and in the PC tab select "Oculus Touch Controller Profile"

    9.3. Go back to XR Plug-in Management. You can see no more exclamation marks exist. This means we have a default controller device for our input maps. Switch over to the Android tab (icon of the Android symbol) and click on Oculus check mark. Unity will start installing packages and loading the scripts.

Note: If you select OpenXR in this step, all unity apps are played in a window inside the Quest 2 (like a movie)

9.4. You can see a new option under XR Plug-in Management called “Oculus”. This means everything has been imported fine. Make sure that in “Target Devices”, Quest 2 is checked.

10. ALL BACKEND FILES HAVE BEEN SUCCESSFULLY IMPORTED. Now let us build a VR friendly scene.

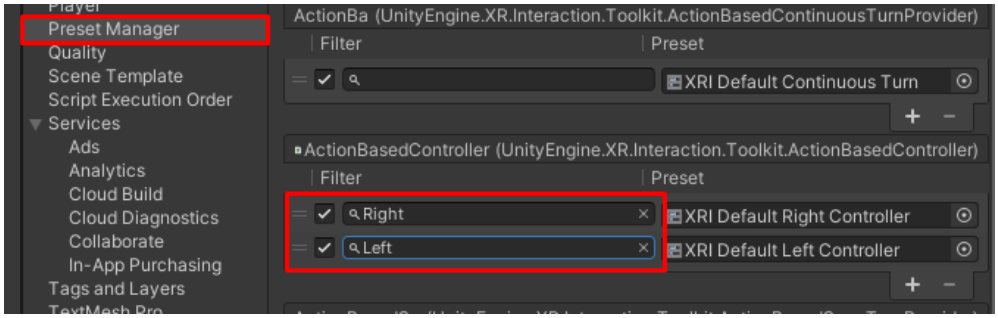
Tips: If your Unity is performing slow, go to Window -> Rendering -> Lighting -> Scene, and uncheck Auto Generate. Also, if you are using URP, uncheck Fog under Other Settings in Window -> Rendering -> Lighting -> Environment.

BUILDING THE SCENE TO BE VR READY

11. Before setting up the XR Rig it is essential to load up the Default Input presets. Go to **Assets -> Samples -> XR Interaction Toolkit -> 1.0.0 prev6 (or whatever the name is) -> Default Input**.

12. Click on each of the files, and on the Inspector (at the top) **click “AddToActionBase…..” button**. Do this for all the files (as long as they have the button on top; not needed for “XRI Default Input Actions …” file (the one whose logo is a bit different). This automatically sets up the controls specific to Oculus for locomotion, interaction, etc.

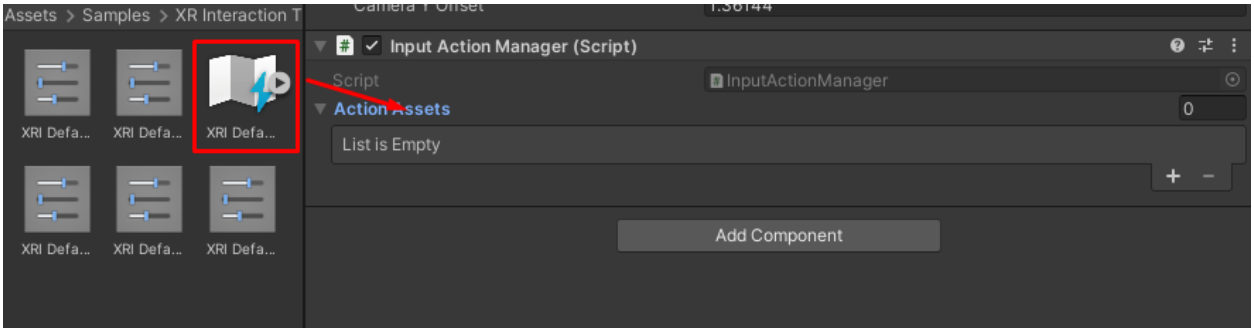
13. Now go to **Edit -> Project Settings -> Preset Manager** and under “ActionBasedController” name the “XRI Default Right Controller” as “Right” (under the Filter field) and “XRI Default Left Controller” as “Left”.



14. Now, right click on **the hierarchy -> XR -> XR Rig (Action Based)**. This will create the room-scale volume inside which the VR play area will be considered. Keep anything you want interactable, inside this area.

15. If your XR rig is not in the right place, move it to (0,0,0) and adjust accordingly. Don’t lift it up above the floor (unless your project has that sort of creative aspect).

16. In the inspector (with the XR Rig selected in the hierarchy), **Add Component -> Input Action Manager**. Under Action Assets (it will be empty), drag and drop the XRI Default Actions file. Look at pic below. We are now ready to play.



At this point, **put the Quest 2 in the Link Mode, and then press Play**, and you should be seeing the scene inside the headset. Rotating the controllers will not do anything since we are using the Roomscale XR Rif, which expects us to physically walk around in the room. [STILL YET TO TEST THE LOCTIOMOTION RIG; AFAIK we can add Locomotion system, Continuous Move, and Continuous Turn to convert it to move].

BUILDING APPS

17. Safest and fastest way to debug is the Linked mode. However, to deploy you final playable versions, you will have to build the project into an app.

18. Go to File -> Build Settings -> Add Open Scenes.

19. Click on **Android** under Platform. Keep **"Run Device"** to **Default Device** for now. Click Switch Platform on the bottom. Switching may take some time.

20. Once platform has been switched (confirm by seeing the Unity logo beside the word "Android"), **switch from Default Device to Quest 2** (yours may have a large string of alphabets and numbers, it should be that one). Make sure that the Quest 2 is connected.

21. Click "**Build and Run**". It will ask you to save the .APK file. I suggest you create a new folder called "BUILDS" (Ctrl + Shift + N is the shortcut to create a new folder) and save the .APK inside there.

Note: Always use "Build and Run", rather than "Build". "Build and Run" automatically copies the .APK to the Quest 2 and runs it. Only "Build" will need you to manually copy over the APK somewhere.

22. If its your first time doing a build, it will take some time. subsequent builds will finish faster. Put on the headset now, and voila, you should be ready to rock and roll.

23. You can find your app inside the Oculus, when you open Apps, on the right side where it says "All" click that and change it to unknown sources.

Since we are not using traditional Oculus plugins, you will not be coding using the Oculus SDK. Instead, we have to switch to the OpenXR SDK. The good thing about it, is that it is highly configurable, and Unity backed, so chances are it will have better and more updated documentation. Link: https://docs.unity3d.com/Packages/com.unity.xr.openxr@1.2/api/index.html

Also most things (grab, distance grab, etc) already are implemented as scripts (using Add Component).

Sources: Valem (Youtube); Justin P Barnett (Youtube); Martina Verse (Youtube)