hello (virtual reality) world

Here is a small workshop on how to setup a VR game for your unity projects. We are going to be using the Oculus sdk for our worlds. There are other options such as XR Toolkit that allow you to build to other headsets that we won't be covering.

**prerequisites*: all you'll need to follow along is a unity project, you'll need to install some other things along the way.

Step 1: Configure Unity Settings

Follow along to the guide down below to make sure you have all config settings correct
https://developer.oculus.com/documentation/unity/unity-conf-settings/

Step 2: Downloading Meta XR SDK Packages

The following link gives you several options & descriptions about the sdks meta offers for unity developers.

https://developer.oculus.com/blog/unity-package-manager-sdk-updates-v59-meta-quest/

- **com.meta.xr.sdk.core**
  - Contains many essential functionalities which Meta XR headsets provide, including the camera rig, touch controller, hand tracking, composition layer, passthrough, anchor, scene management, and much more.
- **com.meta.xr.simulator**
  - Meta XR Simulator is a lightweight OpenXR runtime built for developers that enables the simulation of Meta Quest headsets and features on the API level. It makes day-to-day development easier by enabling testing and debugging of apps without the need to put on and take off a headset frequently and helps scale automation by simplifying your testing environment setup.
- **com.meta.xr.sdk.platform**
  - Use the Platform SDK to create social VR applications. Add Matchmaking, DLC, In-App Purchases, Cloud Storage, Voice Chat, Custom Items, Achievements, and more to your experience using the individual components of the SDK.
- **com.meta.xr.sdk.audio**
  - Meta XR Audio SDK provides spatial audio functionality including head-related transfer function (HRTF) based object and ambisonic spatialization, as well as room acoustics simulation. It's a replacement for the Oculus Spatializer plugin.
- **comcom.meta.xr.sdk.interaction**
  - Interaction SDK adds interactions like ray, poke, locomotion, and grab for controllers, hands, and controllers as hands. Each interaction is designed to be modular and work in both simple and complex VR applications. Interaction SDK also has features just for hands, including hand-specific interactions, pose and gesture detection, and debug visuals.
- **com.meta.xr.sdk.interaction.ovr**
  - This package allows Interaction SDK to interface with OVRPlugin. Use this package if you're using OVRPlugin or the Utilities package.
- **com.meta.xr.sdk.interaction.ovr.samples**
  - Contains sample scenes, prefabs, and art assets for Interaction SDK, using OVR variants of the player rig.
- **com.meta.xr.sdk.voice**
  - Use the Voice SDK to bring voice interactions to your app and enhance the AR/VR experience with more natural and flexible ways for people to interact with their surroundings *and* each other.

To add packages go to **Window>Package Manager**

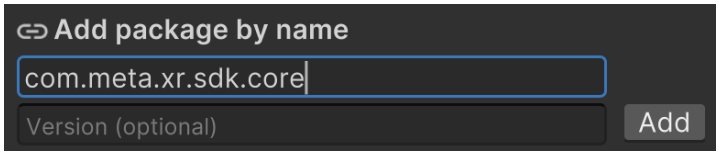
Now select the + button at the top left of the package manager then hit “add package by name”.

We are going to add the following packages

com.meta.xr.sdk.core

com.meta.xr.sdk.interaction

com.meta.xr.sdk.interaction.ovr



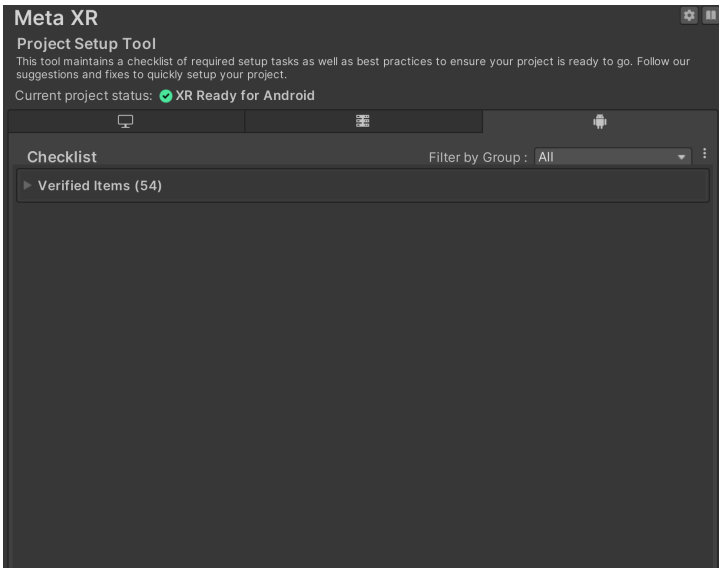
I encourage you to explore other packages that Unity offers if you want to introduce immersive audio or voice interactions.

Before moving on let’s make sure all the warnings are clear from the OVR Manager window. To view any problems select the Meta icon in the bottom right corner of the window.

Alternatively, go to **Edit>Project Settings> Meta XR**

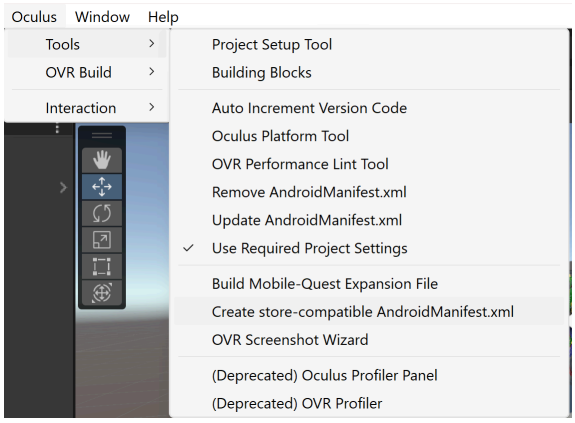
If we see any required fixes, hit the button that says “Fix” or “Fix All”

Now, we should see “XR Ready for Android”



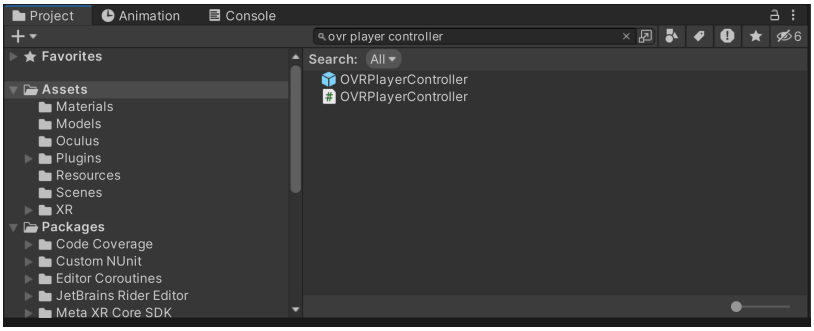
Next, let’s generate an android manifest file for our project

Go to **Occulus > Tools >Create store-compatible AndroidManifest.xml**

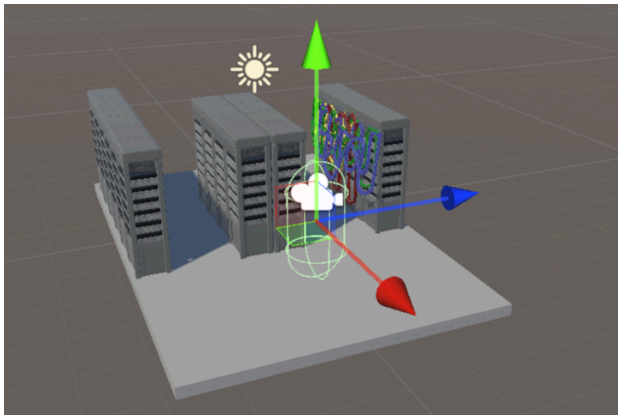


Step 3: Adding OVR Player Controller

Now that we have all of our settings & packages, let's search in our project folder for the prefab called "OVR Player Controller" and add it to our scene.



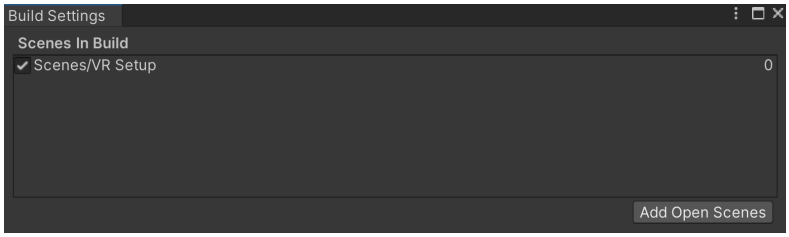
Now, we can populate our scene with assets from our artists.



Step 4: Building

Now let's go to **File>Build Settings**

Make sure you see your scene under "Scenes in Build". If you don't hit "Add Open Scenes"



Next, hit Build.

Now after a few minutes (the time always takes the longest the first time you build it) we have a .apk file.

Step 5: Android Developer Bridge

ADB is how were going to send the apk (build file) from our machine to the headset.

I recommend installing ADB using Chocolatey

After installing run the following on the command line

choco install adb

Or

If you have Mac use Homebrew

brew install android-platform-tools

Or

for VMs or Linux machine you can also use

sudo apt install adb

Step 6: Enabling Develper Mode

We need to turn on dev mode on our headsets to be able to receive builds from our computers. To do this we need to install the Meta Quest App for IOS or Android. You then need to Sync your phone to your headset then go into the headset setting while on the app and go to developer mode and enable it. There are several tutorials on how to do this.

Step 7: Installing Build on Headset

Make sure you have a usb 2.0 to usb-c or usb-c to usb-c. Now open up the location to your build file on the command line. Connect your headset to your computer, ensure that you allow the computer to access your files on your headset.

Type adb devices to see that your headset is connected.

```
PS D:\Unity\Projects\VIW TA Tutorial\Builds> adb devices
List of devices attached
2G0YC1ZF880BY9  device

PS D:\Unity\Projects\VIW TA Tutorial\Builds>
```

Now type adb install (your build file name)

```
PS D:\Unity\Projects\VIW TA Tutorial\Builds> adb install .\TAVIW.tk01.apk
Performing Streamed Install
Success
PS D:\Unity\Projects\VIW TA Tutorial\Builds>
```

Step 8: Opening the Project on Headset.

You can navigate to your game by going to **Library>All>Unknown Sources**

I also cover how to get there in the video attached to this pdf.