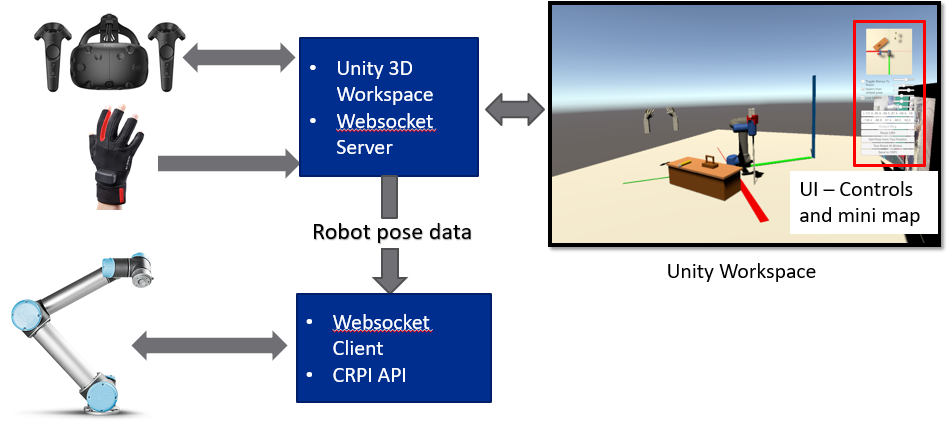
## A brief overview of the system:



### How to make it work:

#### Step 1:

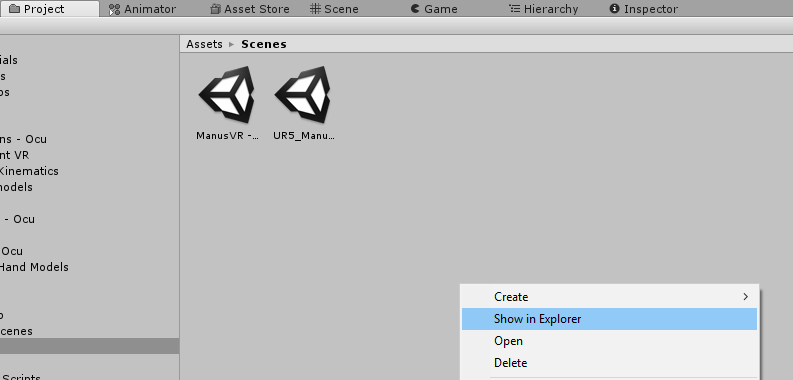
Download folder from

<https://github.com/OvercodedStack/Unity-Test-Scenario-UR5-ManusVR-HTC-VIVE>

and create a folder named Assets – this is a folder equivalent to a unity assets folder.

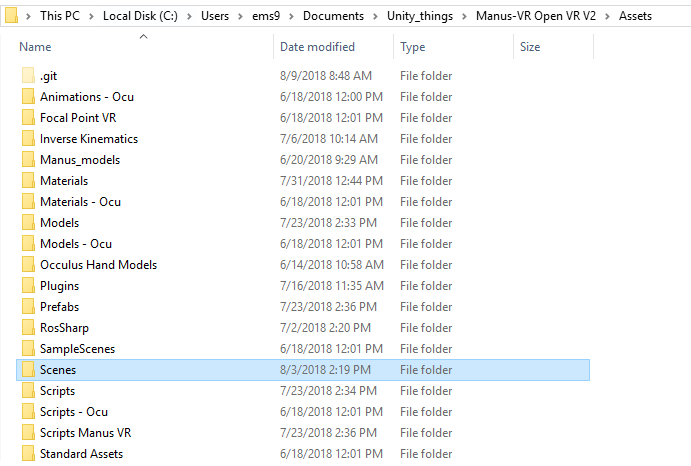
#### Step 2:

Create a blank Unity project and go to the root assets folder



#### Step 3:

Overwrite the blank assets folder in the new project with the files from the downloaded folder named Assets. When you’re done, the assets folder in the new unity project should look as follows:



#### Step 4:

Download folder from

<https://github.com/OvercodedStack/C-Sharp-API-Implementation-ManusVR>

#### Step 5:

Extract the previous folder into a folder called: Scripts – Manus

#### Step 6:

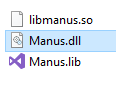
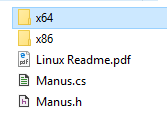
Move this folder into the assets folder of the new project. Note that this functions more or less the same way a scripts folder is.

#### Step 7:

Download the SDK Plugin from <https://developer.manus-vr.com/>

#### Step 8:

From the previously downloaded folder locate the x64 folder, navigate inside of it and copy the Manus.dll from inside of it.

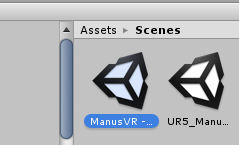


#### Step 9:

Paste Manus.dll into the plugins folder from the new assets folder.

#### Step 10:

While inside Unity, navigate to the Scenes folder and open the ManusVR – VIVE – UR5 Sim program.



All steps should be completed by now to start using the simulation.

### For robot CRPI communication:

#### Step 1:

Download <https://github.com/OvercodedStack/CRPI-UR5-Manus-VR> or locate the CRPI project that can communicate with

#### Step 2:

Compile with the respective CRPI files and dependencies

#### Step 3:

Locate program at C:\\CRPI\Debug

Run Manus\_interface.exe

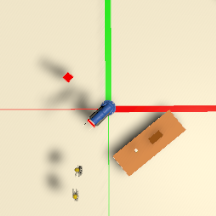
#### Step 4:

When prompt, the **client** program will ask for an amount of cycles to run before timing out. The amount of cycles varies with the amount of data being sent by a **server** program (Unity) therefore, in this case, the client has a minimum cycle rate of performing an action every 2 seconds only if the server is sending data every 2 seconds. If the server is sending information every 10 seconds, the client will respond every 10 seconds.

Please note that the 2 second delay is hardwired and can be changed, however note that tweaking the client to avoid receiving noise data from the server is necessary. The minimum rate discovered to move the robot has been 1 second.

### Special Notes:

* Due to a recent SteamVR update to enhance user experience, the HTC VIVE now has the tendency to remain in the menu mode while starting up a Unity debug program. This means that the VIVE controllers and trackers occasionally are not active in the scene when the user hits play on the debug menu on Unity. As a workaround, the way to avoid this situation is by having a controller in your hand, repeatedly press buttons while starting the debug program, and exit the VIVE menu when you’re done.
* The Manus VR rotation gesture could have been done better. At present the script that handles this only takes the rotation of the ManusVR itself rather than the rotation of the end effector as it should.
* UR5 physics are not exactly working properly while picking them up using the grippers. This is because a script that can continuously update the collisions between objects and avoid clipping thorough them is necessary. A suggested script is on the following link: <https://answers.unity.com/questions/34444/high-speed-object-collisionhow-to-avoid-pass-throu.html>
* The inverse kinematics algorithm on the virtual UR5 displays the literal prediction of what the real-life UR5 will do in practice. This means that if the user moves the UR5’s end effector in a direction and they see all the positions that the robot will perform through the simulation, it is highly likely that the UR5 will perform the same motions, *even if it involves clipping itself*.
* The inverse kinematics algorithm will perform its base rotation sequentially based on the 1,2,3 and 4th quadrants. This means that if the robot is located on the first quadrant going to the fourth quadrant, the robot will go through 1,2,3, and finally 4 to get there. This is primarily due to how the angle joints are limited to go to.
* There are a couple of scene options that are hidden inside the Hierarchy - > Scene tab. Activate or deactivate these as you see fit.



2

3

4

1