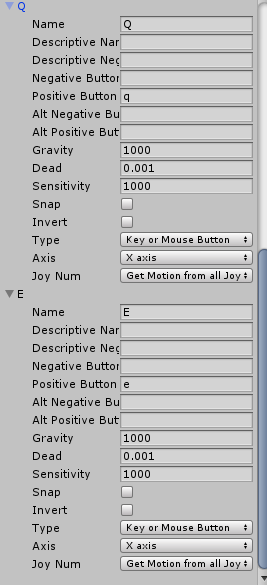
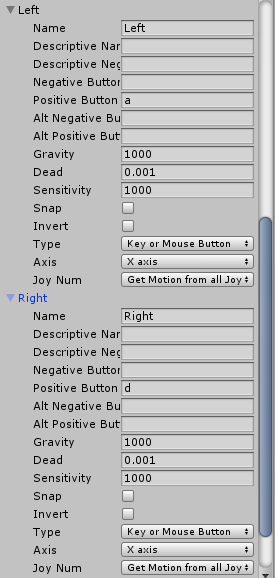
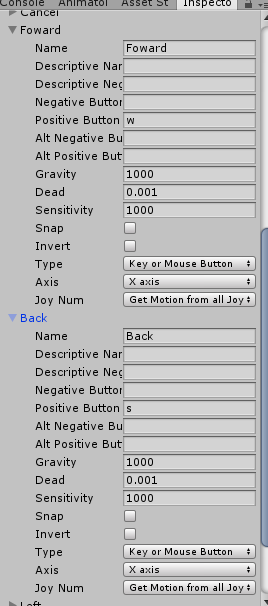
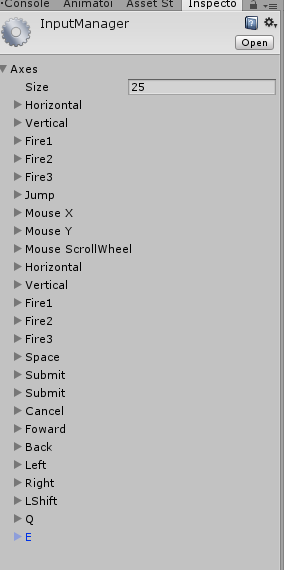
General report:

While most of the code has been commented with references in this project, there are still some hidden features/ problems that should be addressed. The following bulletpoints should hopefully cover most of these issues

# First time running issue

While setting up the project for the first time using the GitHub repo, there will be an issue of setting up the necessary keybinds for a script to function properly. These keybinds can be set through the Edit - > Project Settings -> Input tab. These keybinds should be set as follows.



-Adjust image size if needed.

# How to “officially” running the program

To run the full setup with the Manus VR, HTC VIVE, and the UR5 robot, you have to perform the following sequence of events to complete the start up.

1. Run the debug Unity program
2. Start the CRPI Application
3. When prompted, enter an amount of cycles to run the program (recommended: 100)
4. Move the gameobject “Handlepoint” to move the end effector of the UR5 robot to perform a desired motion.
   1. Moving the game object can be done by selecting it from the hierarchy menu or through using a game controller/Manus VR

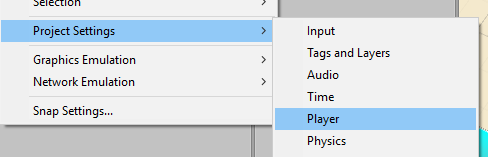
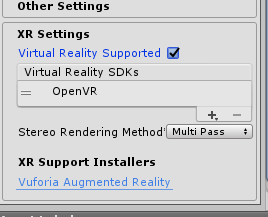
# Problems while running

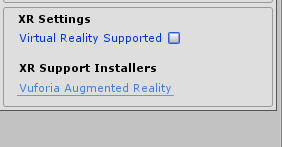
The Unity simulation provides a realistic motion of what will happen in real life. This is, the motion that you see in the sim will be done by the robot when set in motion. Sometimes this causes the problem that the robot will clip into itself or will start doing motions that will crash into objects such as the floor. It’s best to understand the simulation’s motions before performing them in the real robot.

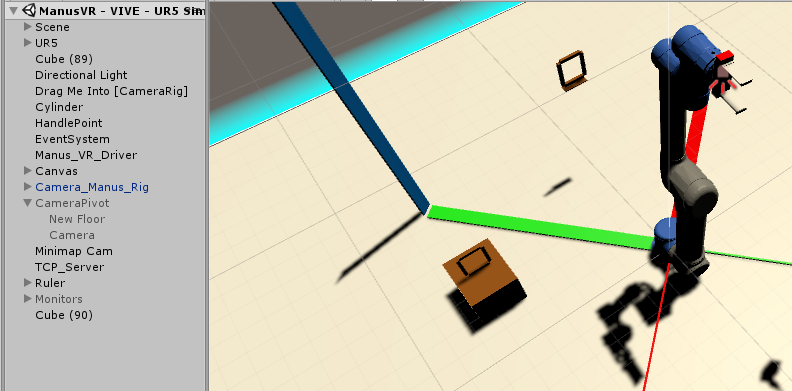
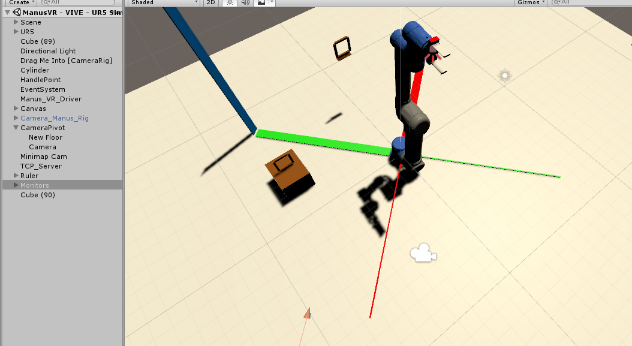
# Unlocking the second mode of the Unity Project

There is a separate way of using the simulation without having to use the HTC VIVE or Manus VR Gloves. This mode can be activated through the activation and deactivation of certain features of the Unity project. These include the following steps:

1. Deactivate VR support
   1. This can be done by shutting off the VR enabled option in the player settings.
   2. Edit -> Project Settings -> Player





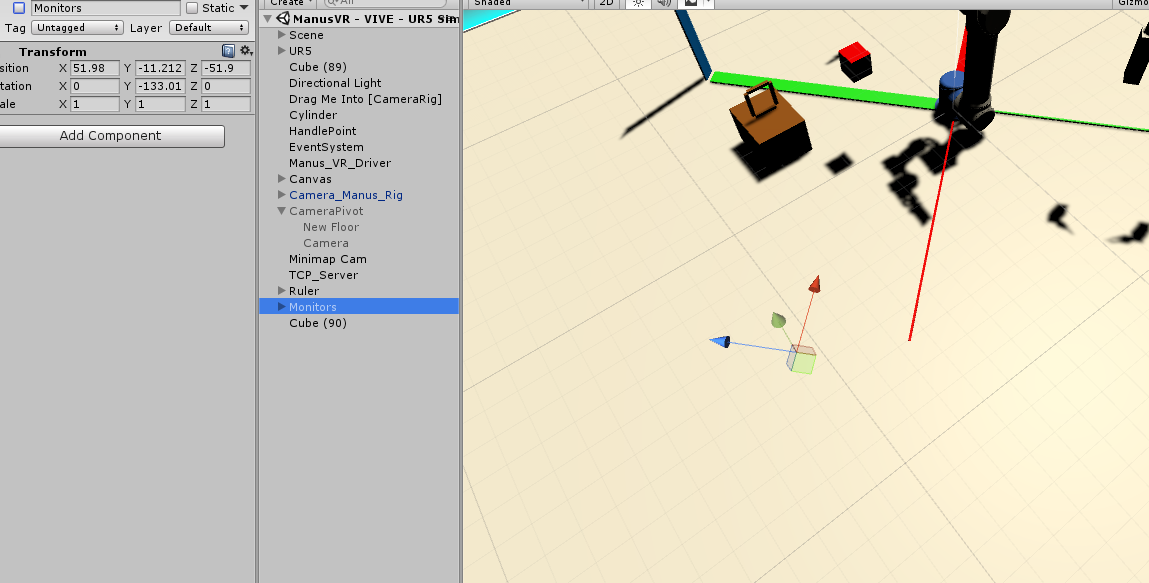
1. Activate necessary gameobjects
   1. This can be done by deactivating the Camera\_Manus\_Rig gameobject and activating the CameraPivot gameobject. You will notice this step once the glow around the floor is gone.
2. Run.

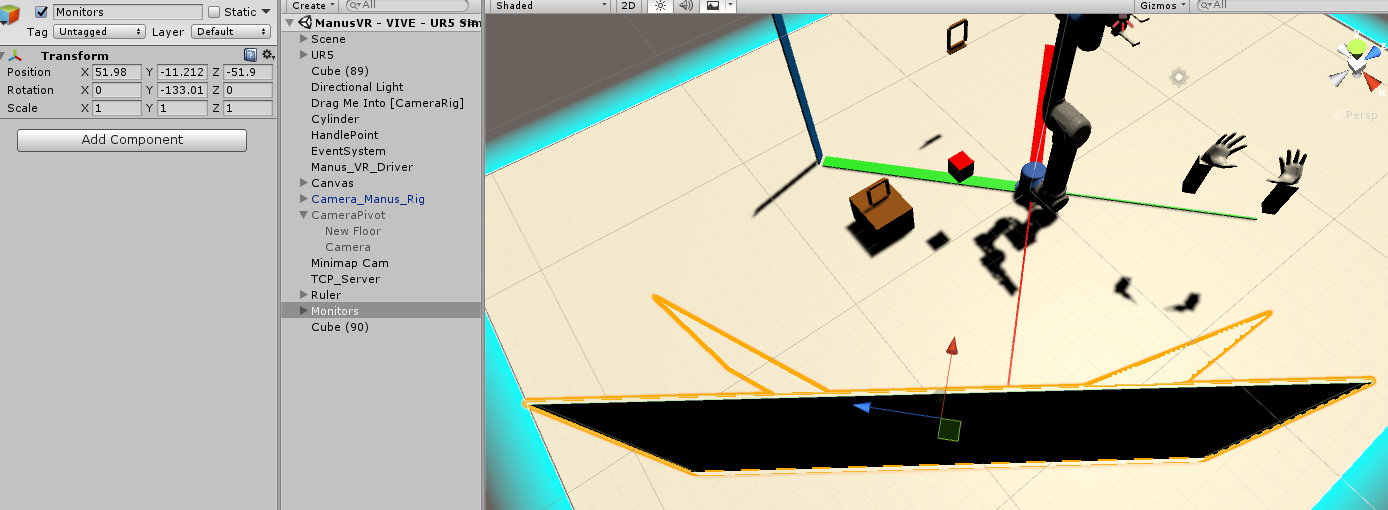
The game view will switch over to a main camera that is set to be the same as an orbiting camera in a game. This functions the same way a common RTS or simulation 3D camera works with the difference of speed.

# Activating Monitors

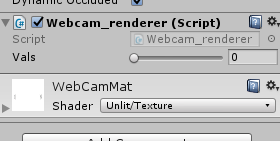
The Unity space also contains monitors that can be used to visualize the external world through webcams. The way this functions is through array indexing, where the first camera plugged in will be set at the channel 0 and every sequentially added camera is added on top of this.

Activating:





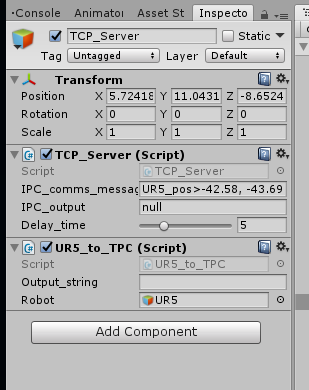
Adjusting for webcam preview:



Move slider left or right to select a different camera on the array.

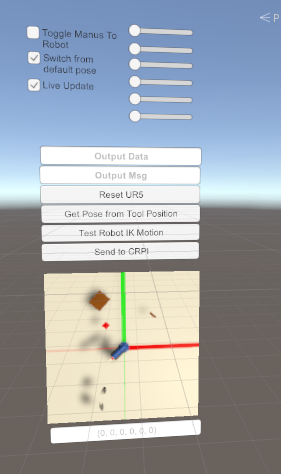
# Adjusting server posting speed/debug

The Unity project has a standalone TPC server that outputs data to the localhost. The posting speed of this project can be adjusted as desired from the settings menu on the game object.



Adjusting the delay time to be less than 2 seconds in this case however, will result in null effect since the client CRPI UR5 application has not been adjusted to receive messages faster than 2 seconds.

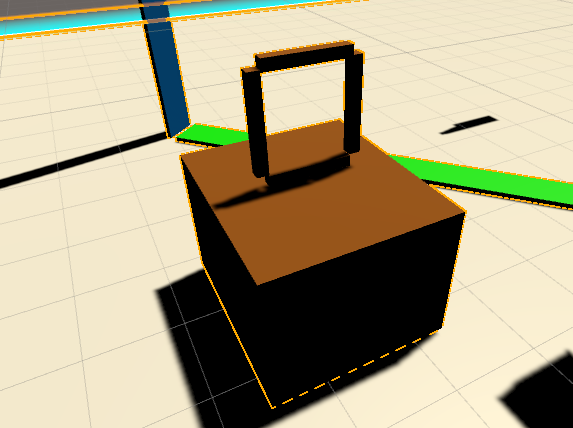
# UI in camera



The provided UI is set to adjust the angles of the UR5 without having the inverse kinematics algorithm running to set the joint positions. This means that while the joints are being calculated, they cannot be manually adjusted by the UI. Included is also a minimap of the world around the robot.

# Provided Layout

The rings around the scene are locations of positions that were utilized to move a blue tape around the real space. While the unity units do not reflect the scale that was used for the project, the positions relative to the following measurements are closely related/estimated.

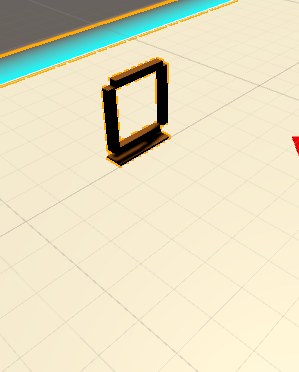


Point on top of cardboard:

X: 38 cm

Y: 62.5 cm

Z: 17 cm



Point floating:

X: 57 cm

Y: -7 cm

Z: 44.5 cm

# Acknowledgements

Special thanks to the NIST SURF program

Megan Zimmerman

Shelly Bagchi

Jeremy Marvel

Karl van Wyk