Segarra Martinez, Esteban (Assoc)

National institute of Standards and Technology  Blding 202 – Rm 211

CRPI – Unity UI Project

Preface

# Before you begin

There are many parts to this project, some of which were documented and other parts that were not as well documented. Due to the nature of how everything is not localized to one device or system there were a series of deductive reasoning ideas that were created to solve some problems. Not everything was solved trough this method but several parts did not have straight-forward approaches to solve.

# Assumptions about hardware and software

The hardware that was utilized throughout this project remained consistent and can be summarized in the following list:

* Android tablet SM-T700 (Collablab2)
* Latitude E5570 Nithya’s old laptop
* UR5 robot
* CollablabA network structure
* Vicon system (10 camera setup)

The following list of IP addresses were utilized throughout the project:

* Vicon system:
  + 169.254.152.38 (CollablabA)
  + 129.6.35.213 (NistNet)
* Tablet IP address:
  + 169.254.152.27 (CollabllabA)
* Nithya’s laptop:
  + 169.254.152.2 (CollabllabA)
  + 129.6.35.81 (NistNet)

While not strictly relevant, the following are the IP addresses of the robots used in the project:

* 169.254.152.47 (UR5)
* 169.254.152.46 (UR10Left)
* 169.254.152.45 (UR10Right)
* 169.254.152.80 (ABB Left and right)

# Introduction

This project was created as part of a larger project constituting the goals for research into standards for human-robot interaction. This was meant to be as a tool that can be used to measure the effectiveness and performance of utilizing different human-machine interfaces with varying differences in their applications and implementations.

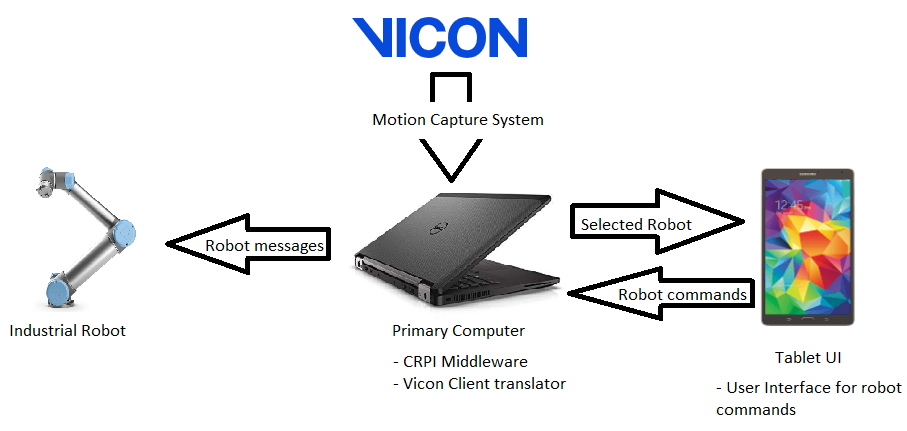
For the purposes of this project, the user interface based off from an android tablet was utilized for the purposes of creating an interface that could be used to flexibly implement different layouts of controls on the screen. This control panel was not based off any specific control panel and was developed with most of the controls found on a regular industrial robot.

Without a strict design goal, this user interface was designed to control items as fast and predictably as it could. Nonetheless there are many challenges to creating this design as despite having control in 3D space for control, many of the functionality aspects are not as straightforward to implement.

Please read the guide for any doubts, if there are doubts, please send an email to [esegarra3278@floridapoly.edu](mailto:esegarra3278@floridapoly.edu)

## Quick Start guide

There are three parts in total for this project. This is demonstrated in this following diagram:



The main components are:

* Vicon motion capture system proxy app
* CRPI middleware app
* Android app

# Vicon motion capture system proxy

## Purpose

The intent of this program was two-fold, to capture UDP/TCP messages from the motion capture system using the Vicon SDK and to calculate which object is being observed from the point of view of the user towards a robot. This system has many advantages in practice such as being able to see through most objects while the tracking system may not be prefect. The user only has to wear standard issue safety glasses that contain mo-cap implements to track the glasses throughout the motion capture world.

This section assumes you have downloaded the app project files at this point.

## Download

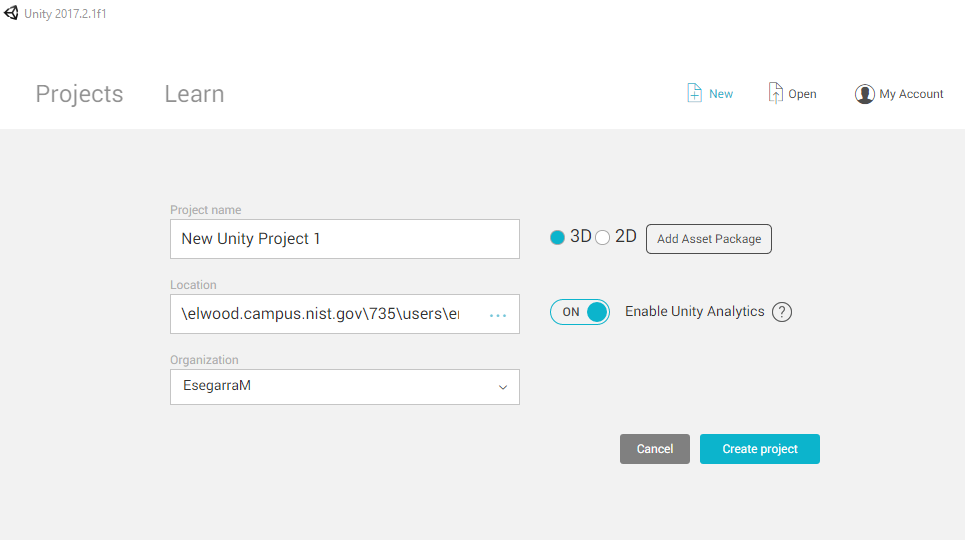
Download the project source folder from <https://github.com/OvercodedStack/MOTION_CAPTURE_UNITY-Summer-of-2019-NIST>

**This project was created and run on Unity 2017 2.1f1.**

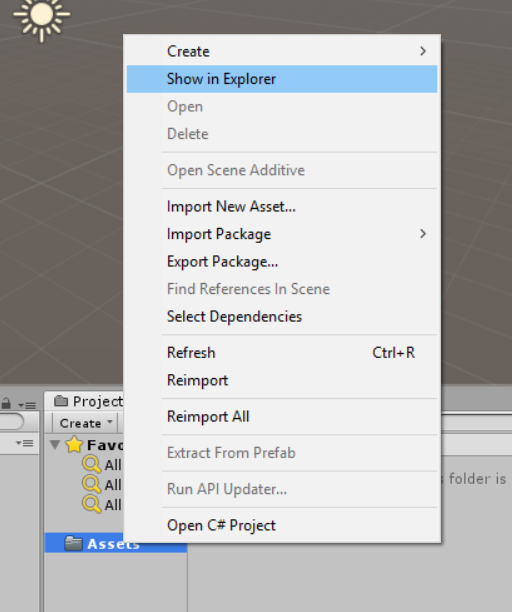
## Installation

As Github limits the size of uploads, this project will be limited to only including the source folder. Therefore a manual copy and overwrite of the project has to be done in order to download and install the project. This is the process to do so.

1. Start Unity editor
2. Click on new, name a new project, start it as any type of project

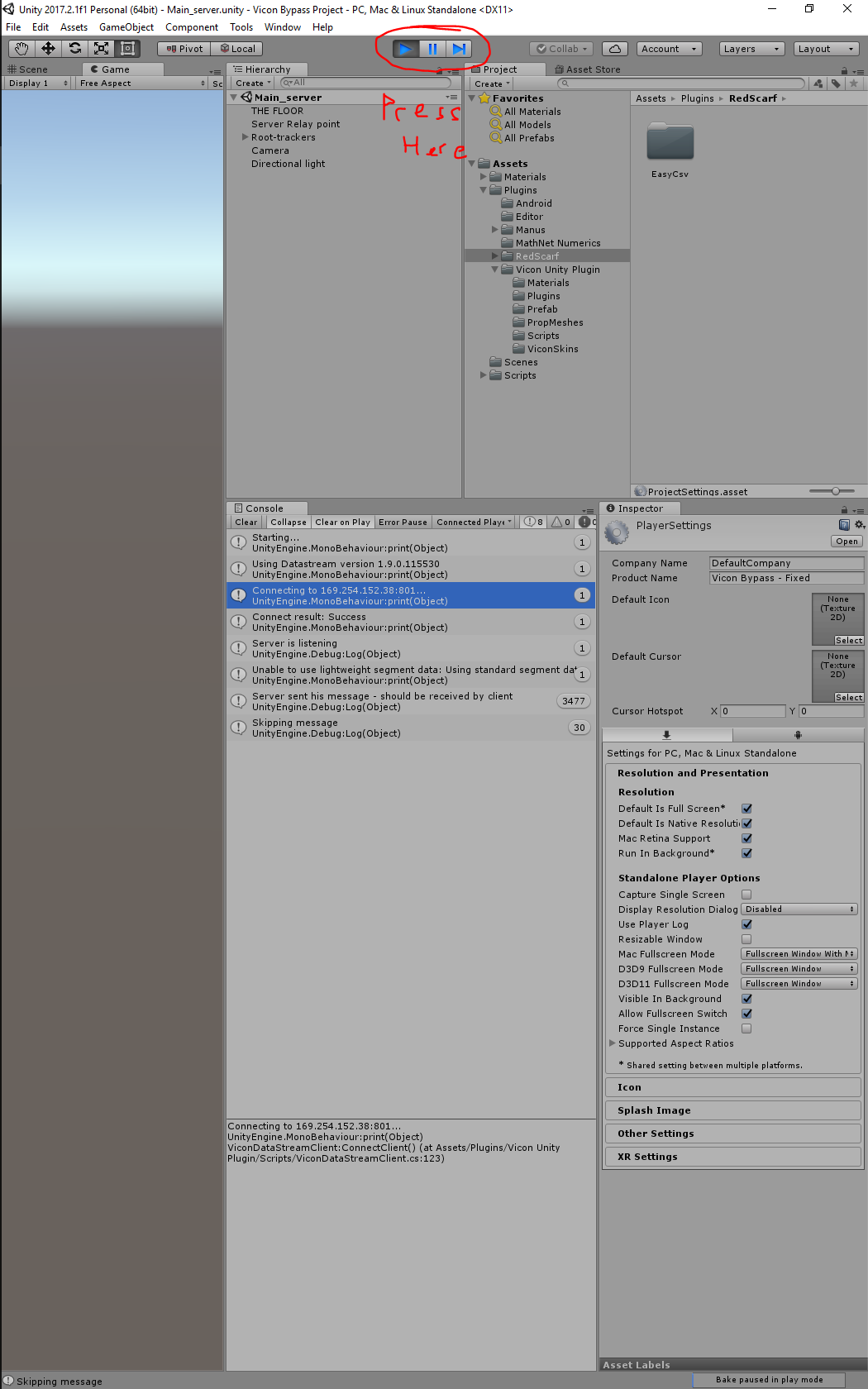


1. Navigate to the “Project” tab and right click on the assets folder. Click on “Show in Explorer”

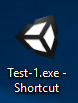


1. Delete all the files inside the folder and copy the repository files into the folder
2. You’ve successfully installed the project into unity

## Initialization

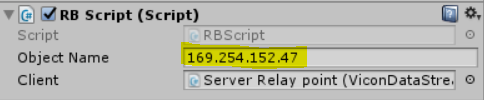
To start this app, simply open the unity project in unity and press play in the editor 

Alternatively, you can build the solution via File > Build Settings > Build and Run. This results in a stand-alone app that will transmit the required data to the CRPI middleware. You can then create a shortcut for the created app and place it somewhere familiar.



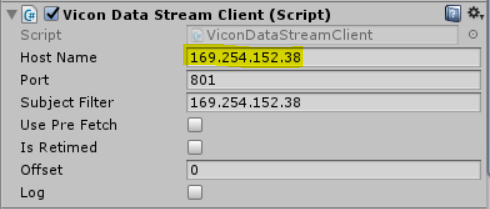
Upon starting the program, you should be able to see a preview of an overhead view of the objects that are being tracked by the system.

**Tracking new objects**

[](https://github.com/OvercodedStack/CRPI-UI-DOCUMENTATION-Summer-of-2019/blob/master/Images/Object%20tracking%20using%20Vicon.PNG?raw=true)

To track a new object, simply create a new object in the Vicon program, change its name to something you can remember, and utilizing that name apply it directly to the yellowed line on the script in Unity. Every object you would like to track with the system will require this script in order to be tracked.

**Setting up the connection to the Vicon**

[](https://github.com/OvercodedStack/CRPI-UI-DOCUMENTATION-Summer-of-2019/blob/master/Images/Vicon%20server%20connection.PNG?raw=true)

To pair the Vicon client from the SDK, change the IP address on the yellowed line on the server node game object in this script. This will change it to a different IP address. You may keep the same port address.

**A note on messages being sent**

This app sends data through a TCP connection on a localhost(127.0.0.1) connection on port 27001. This address can be changed in the **Server Relay point** gameobject. This program is reccomended to be utilized in collaboration with the [CRPI framework](https://github.com/OvercodedStack/CRPI_MIDDLEWARE_INTEGRATION-Summer-of-2019-NIST) program being used to interpret the flags sent.

This program sends one flag that is an integer that ranges from 0 - 6. The flag represents the following:

* 0: No robot is being observed
* 1: UR5 is being observed
* 2: UR10L(eft side robot arm) is being observed
* 3: UR10R(ight side robot arm) is being observed
* 4: ABBL(eft side robot arm) is being observed
* 5: ABBR(ight side robot arm) is being observed

The order and robot list being utilized can be changed under the **Minimal\_change\_robots.cs** script in Unity with a list of strings that contain the names of the desired robots to be controlled. The control and use of these objects can be changed by the user.

# Android app

This is the main application; this app runs on its own on an Android tablet or as a debugging feature on the unity editor. It is recommended to run this app on the tablet although the app can be running on the PC with some minor modifications to the control scheme of the app. This is the main control panel for the project and is the part that is controlling the robot.

## Download

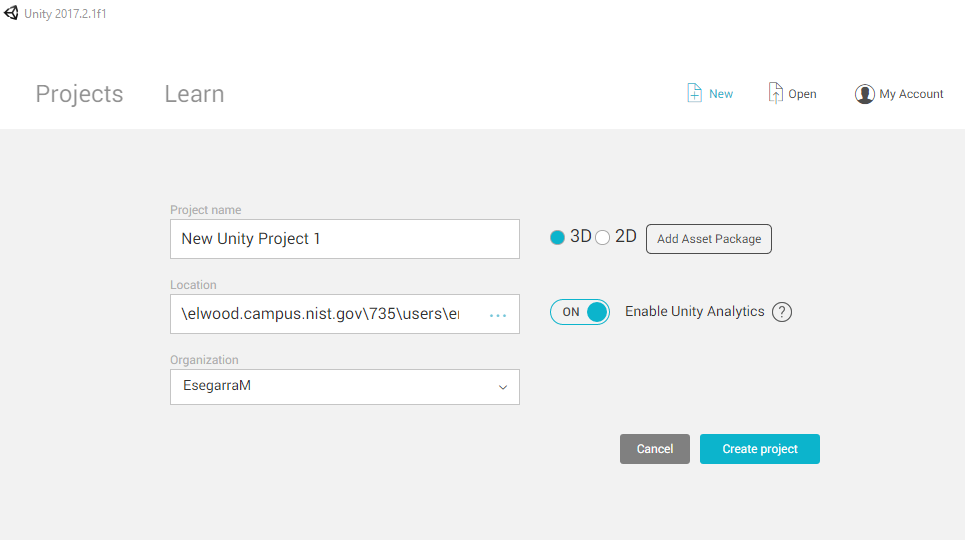
Download source from <https://github.com/OvercodedStack/ANDROID_UNITY_UI-Summer-2019-NIST/tree/Android-Branch>

**This project was created and run on Unity 2017 2.1f1.**

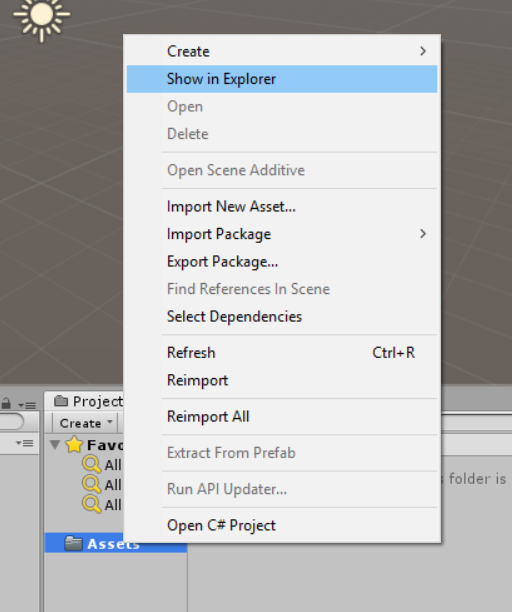
## Installation

As GitHub limits the size of uploads, this project will be limited to only including the source folder. Therefore, a manual copy and overwrite of the project has to be done in order to download and install the project. This is the process to do so.

1. Start Unity editor
2. Click on new, name a new project, start it as any type of project



1. Navigate to the “Project” tab and right click on the assets folder. Click on “Show in Explorer”



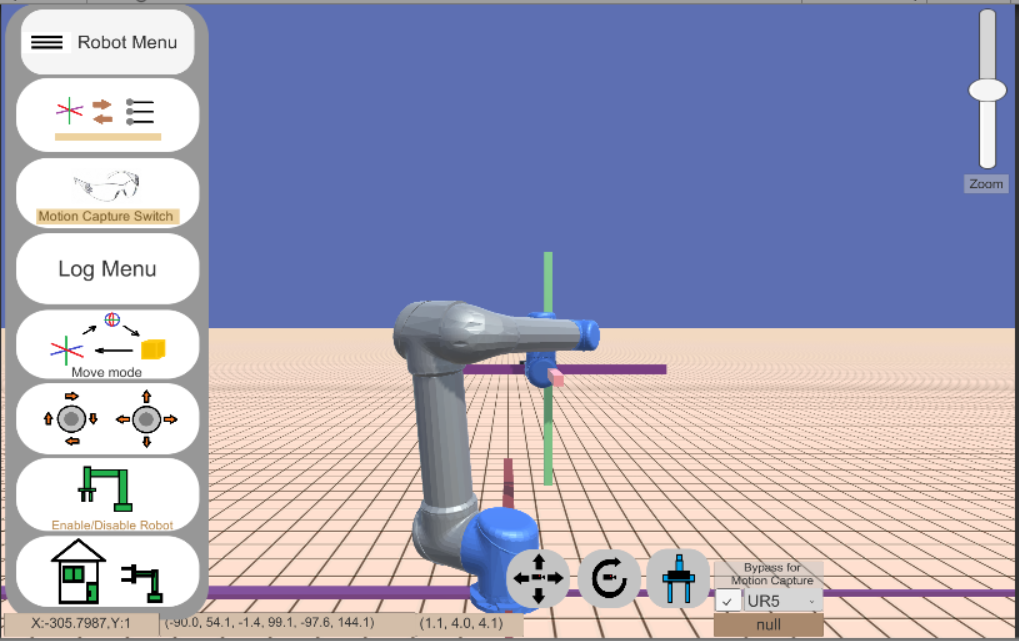
1. Delete all the files inside the folder and copy the repository files into the folder
2. You’ve successfully installed the project into unity

## Initialization

You can start the project through pressing the “play” button or you can compile directly onto a working android device with Android version 6.0.1. All presets are set for this version and it may not be guaranteed to work on newer Android versions. It is recommended to use a device with at least 2560x1600 pixels in the horizontal or vertical configuration as the app is optimized to use the most display available.

## Usage

Due to the nature of the Android app, the app can be running in the background and pause all commands going out to CRPI. Upon starting the app, you will arrive to a window similar to the one below.



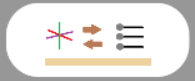
# COntrols for UI

### Main menu

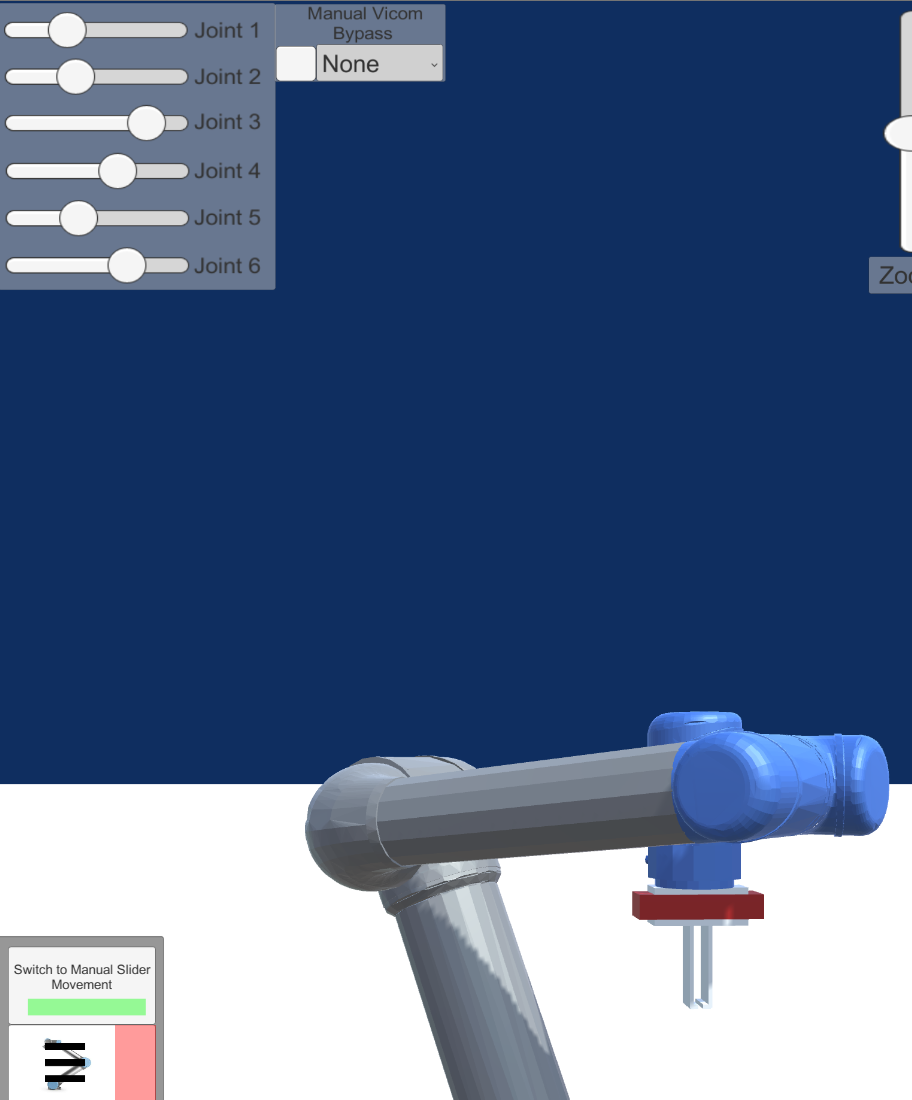


This button hides and shows the important features that the user interface has to offer, many of the subsequent button explanations are related to this button.

## Slider mode button:

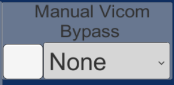


This button switches the mode for the joint value controller. This switches the mode from controlling from the end-effector to moving joint by joint. Noticeable changes in the UI include disabling the visual control from the end-effector, displaying the joint controller, and the change in the control scheme for the user.



Joint controllers

## Manual Vicom Bypass:



When triggered this button acts as a bypass for the Vicom input being received and instead of activating the robot through the use of the following button:



The robot will be manually activated through one of the available presets in the dropdown menu. Any option from this dropdown menu will be selected and sent to the CRPI menu for reinterpretation. This method is slightly more reliable than selecting the Vicom option as the Vicom option can be delayed and CRPI sends the hooking message more late than it should.

This indicator shows which robots is currently being selected by the bypass or the vicon.



## Logging menu



Through the use of this menu, the user can set where shall they set the motion for the start and end of the robot motion. They can set where the end-effector is in the virtual space, record it, start moving it to a different location and continue until they are finished. This option is still being worked on and has several bugs that require more exploration of the Android I/O storage and readout.



* Start replay: as the name implies, it will start replaying motions loaded from the file in the dropdown menu
* Start recording: starts appending more commands to the dropdown file
* Automove to cords: Forces the end-effector to follow the commands of the menu (Actually starts the robot motion)
* Next pose: Loads in a new pose from the loaded CSV file
* Chg Save File: changes which file shall be loaded from the list in the dropdown menu
* Save pose: stores data into the currently selected file from the dropdown menu
* Dropdown menu: sets the currently selected file.

## The zoom slider



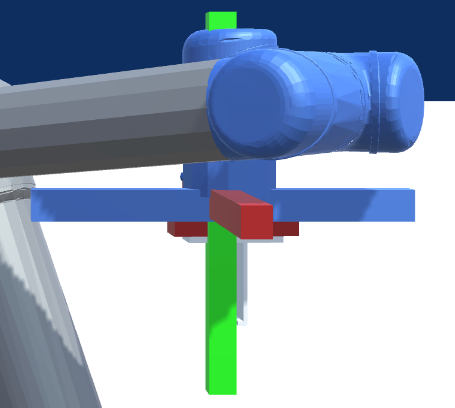
The zoom slider allows a change in the zoom scale of the camera, brings items closer or further from the point of view of the user.

## Change end-effector tool point mode



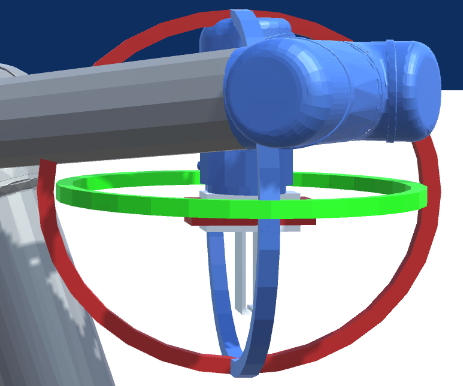
This button changes the mode that the end-effector shall be used for. There are three different modes that this can fall into, axial transformation, rotational transformation, and free-mode transformation. This is similar to tooltips as provided by CAD software or 3D modeling programs.

### Axial transformation



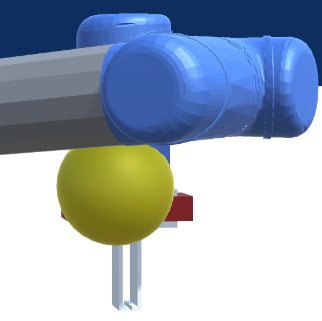
Under this mode, the axis of XYZ are displayed and allow the user to move the end-effector in any direction within that restriction.

### Rotational Transformation



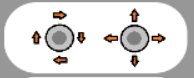
This mode rotates the end-effector in any of the XYZ directions.

### Free-form transformation

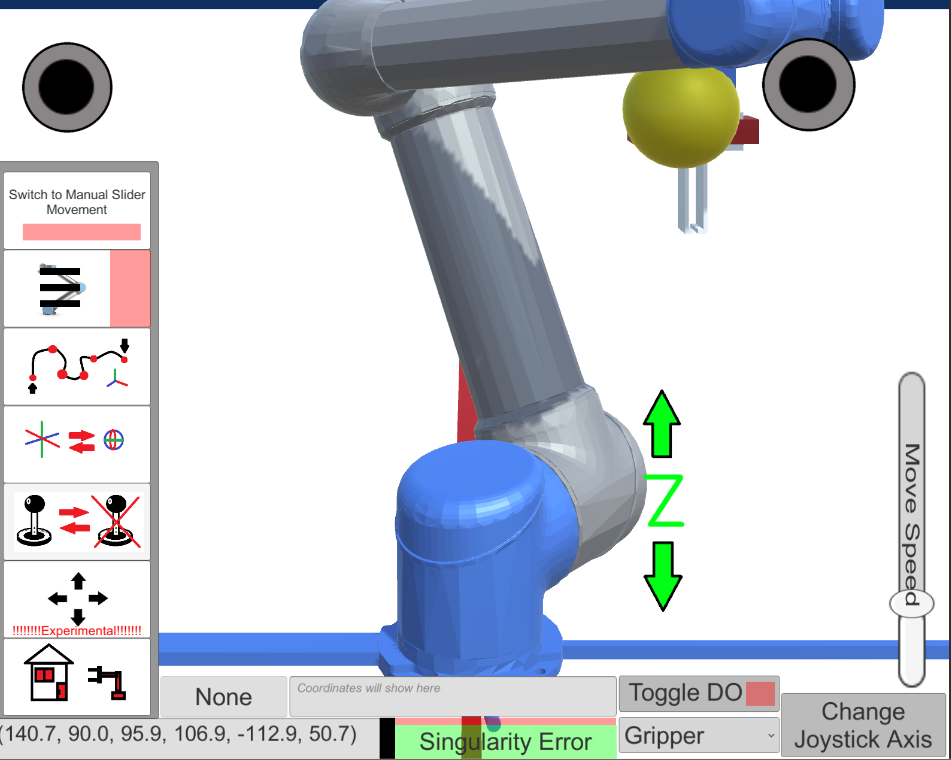
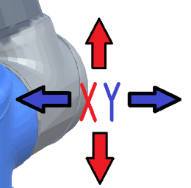


This mode allows the transformation of the end-effector in the perspective of the user in the XYZ directions relative to the front direction of the camera. Thus some fine tuning of control can be achieved but some confusion to the direction that it moves can happen.

### Joystick control



This button activates the use of the joystick controls. When pressed, the layout of the UI changes to the following setup:



Changes movement speed

Alternate direction

Directional indicator

Changes direction of movement

Move End-effector

Rotate End-effector

### Move camera button



This was an offshoot idea of being able to control the viewing camera in the direction of view, this idea was implemented but not fully completed. Currently moves the camera but there is difficulty in being able to visualize the simulation.

### Reset camera button



This button resets the position and rotation of the camera when activated.

### Enable and disable robot commands button



This button disables the IK logic of the robot and disables the TCP server from sending messages to CRPI in order to halt all possible commands to the robot for previewing purposes.

### “Home” button



This button allows the user to reset the robot position after moving the end-effector from it’s home position. Resets to the unity-given location.

### The status output bar:



### Coodrinate output to CRPI



This indicator shows what joint data values are being sent to the robot via CRPI.

### Mouse coordinates



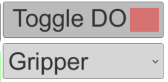
This output shows the coordinates of the mouse based on the size of the screen being used

### End-effector target coordinates



This output shows the coordinates in UNITY space for the end-effector of the robot as based by the end-tool point of the robot.

### Toggle Digital i/o



This control panel toggles the IO condition for the various digital ports on a robot. They can be configured to contain more or less options via the UR5\_TO\_TCP.cs script

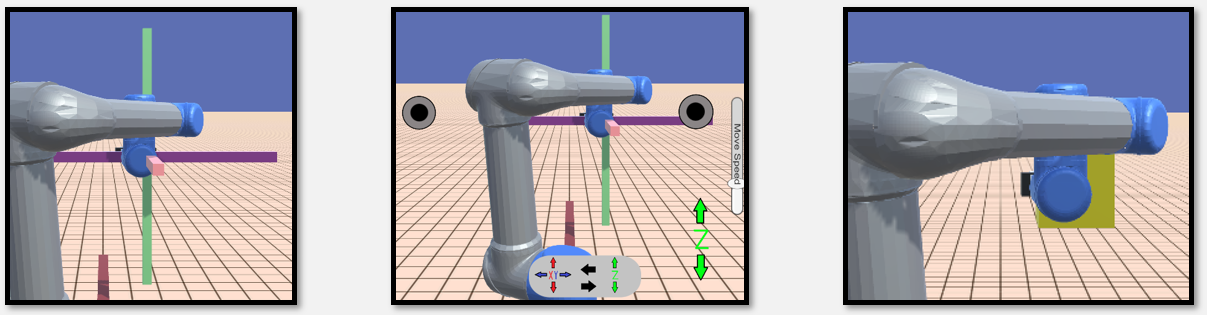
### Toggle gripper status



This button opens and closes the gripper on the virtual and real robots

## Notes

There are three possible ways of directly controlling the robot at the present moment; these are (from left to right) the XYZ position and rotation control, joystick controls, and free-move mode.

[](https://raw.githubusercontent.com/OvercodedStack/CRPI-UI-DOCUMENTATION-Summer-of-2019/master/Images/Control%20schemes.PNG)

The XYZ control is similar in action to those controls as provided by a traditional CAD or 3D modeling program and intended for ease of understanding.

The joystick moves the end-effector and rotates it accordingly.

The free-move mode moves the endeffector according to how the user desires to move the device relative to the angle of the camera and their touch-swipe.

**CRPI Message format**

In order for this system to operate correctly, this system was designed to communicate internally with the use of TCP websockets that streams a string to CRPI for internal communciation. This string order is sent out at the rate given by the Unity TCP Server script in the TCP\_Server\_node\_obj\_coord script. For convenience, this is streamed at a rate every 2 seconds.

**By default, transmission of data is sent through the local network pointed to the CRPI client at 169.254.152.2 on port 27000**

The message format for this string is stated as the following: **{$UR5\_pos:(joint value 1),(joint value 2),(joint value 3),(joint value 4),(joint value 5),(joint value 6), Robot Utilities:(Robot ID),(Gripper),(Digital Port 1),(Digital Port 2),(Digital Port 3),(Digital Port 4),(Manual Bypass flag),(Vicon Robot changer flag)#}**

1. Joint value 1 - 6: Angles as recieved to the UR5
2. Robot ID#: the enabled robot
3. Gripper: This allows the gripper to toggle between closed and open
4. DO(1 - 4): These are the digital port outputs as controlled by the user
5. Manual Bypass flag: This allows the user to bypass the Vicon system and also override the Vicon vision control
6. Vicon Robot changer flag: This flag determines if the user desires to switch robots either through the vicon or the override

**Log file**

The log file for the app can be found in the following locations:

For use in the Unity editor: **/Assets/Resources/**

For use in the Android App: **Device Storage/Android/Data/com.NIST.CRPI\_UI/files**

For most actions performed on the Unity App, there are various listeners and actions that are being recorded, including the time-stamp, the position of the end-effector, the position of the joints, digital ports, and many other features that are occuring at the time of use. This will be detailed in the following file list explanation. In chronological order from left to right the log file will appear as such:

1. Timestamp: Stored as hour minutes seconds with no comma spacing
2. Jnt(1 - 6): The joint angle value in degrees rotated towards. This is the data that has been sent to the robot for CRPI
3. Robot ID#: The enabled robot (desired robot) flag
4. Gripper: This allows the gripper to toggle between closed and open
5. DO(1 - 4): These are the digital port outputs as controlled by the user
6. Bypass Active?: This allows the user to bypass the Vicon system and also override the Vicon vision control
7. Chng\_robots: This flag determines if the user desires to switch robots either through the vicon or the override
8. X: Coordinate system based on Unity scale and location (position)
9. Y: Coordinate system based on Unity scale and location (position)
10. Z: Coordinate system based on Unity scale and location (position)
11. Q\_X: Coordinate system based on Unity scale and location (rotation in quaternion)
12. Q\_Y: Coordinate system based on Unity scale and location (rotation in quaternion)
13. Q\_Z: Coordinate system based on Unity scale and location (rotation in quaternion)
14. Q\_W: Coordinate system based on Unity scale and location (rotation in quaternion)
15. Mouse pointer X: Coordinates based on those relative to the position on the screen size
16. Mouse pointer Y: Coordinates based on those relative to the position on the screen size
17. Screen Mode: Direction of screen (Landscape or portrait)
18. Button being used: The action being performed by the user

**Bugs**

There are some bugs that didn't get enough time to be fixed or implemented. In specific this section will talk about the bugs that would have taken more time to fix than would have been to implement.

* The joystick controls control direction get disoriented when the target gameobject is moved with the other types of controls (e.g: XYZ rotation).
* Some script names were not the best choice for use in certain situations and due to the nature of Unity, when a game object is included with a different script, could result in a missing reference error. In specific, please note this is highly probable with the script *TCP\_Server\_node\_Obj\_coordinator* due to its name-specific reference in **rotator.cs**.
* The Z-axis on the XYZ axis control is emulated using XY mouse delta positions and as such may result in sometimes inaccurate readings or "floaty" control.
* The inverse kinematic control model is not very good at avoiding collisions with itself and may result in accidental clipping with the real robot itself.
* Some scripts are legacy and are not used anymore.

# CRPI Middleware

This is the backbone of the control system. This script contains two TCP clients that are listening to commands being provided by the Android Unity app and Vicom Unity app. It is **required** that both of these apps be running prior to running the CRPI middleware. This ensures that both systems can be running as an option.

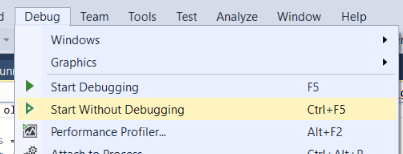
## Download

The source can be downloaded from <https://github.com/OvercodedStack/CRPI_MIDDLEWARE_INTEGRATION-Summer-of-2019-NIST>

## Installation

Simply drag and drop into a current installation of CRPI in a folder called C:/CRPI/Applications/ (Differs from where you may have installed it.)

Upon opening the visual studio solution, press “Start Without Debugging”.



## Initialization

Upon initialization, the program will attempt to connect to the TCP socket for the Vicom and the Android Apps, once connection is achieved, the script will continue running until one of the servers closes, upon which the client may close at the same time.

There are two bugs related with this script at this time:

* Servers that connect to the clients may stay stuck and require restart.
* CRPI may fail to latch to the robot TCP client and may eternally fail to latch, requiring restart as well. Happens more frequently when trying to switch between robots using the Vicon.

## Notes:

**Usage**

Upon initialization, the program will attempt to connect to the TCP socket for the Vicom and the Android Apps, once connection is achieved, the script will continue running until one of the servers closes, upon which the client may close at the same time. If desired, the application's settings can be tweaked to debug or change some functionality.

**Parameters**

* **SHUTOFF\_CRPI**: This disables the CRPI translation functionality and only enables the interpretation of received messages from the Android app.
* **DISABLE\_VICOM**: This disables the functionality of using the secondary system, [the motion capture system add-on](https://github.com/OvercodedStack/MOTION_CAPTURE_UNITY-Summer-of-2019-NIST/tree/master), with this system.

1 = Enables the flag

0 = Disables the flag

**Bugs**

There are two bugs related with this script at this time:

* Servers that connect to the clients may stay stuck and require restart.
* CRPI may fail to latch to the robot TCP client and may eternally fail to latch, requiring restart as well. Happens more frequently when trying to switch between robots using the Vicon.

# Disclaimer

* Certain commercial equipment, instruments, or materials are identified in this paper to foster understanding. Such identification does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

# Special Thanks

Special thanks to the following people during SURF 2019:

* Shelly Bagchi
* Dr. Jeremy Marvel
* Megan Zimmerman
* Holiday Inn SURF Fellows

Thank you all for being the great people you guys are!