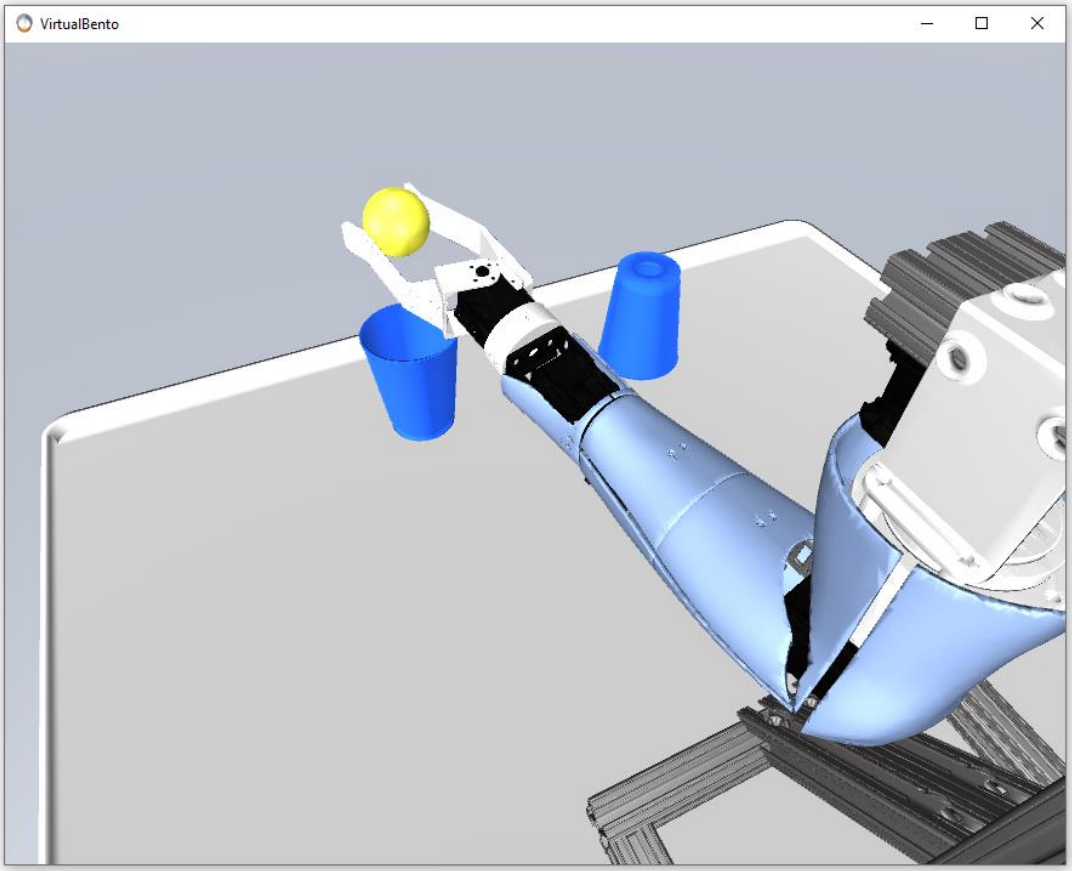
virtualbento

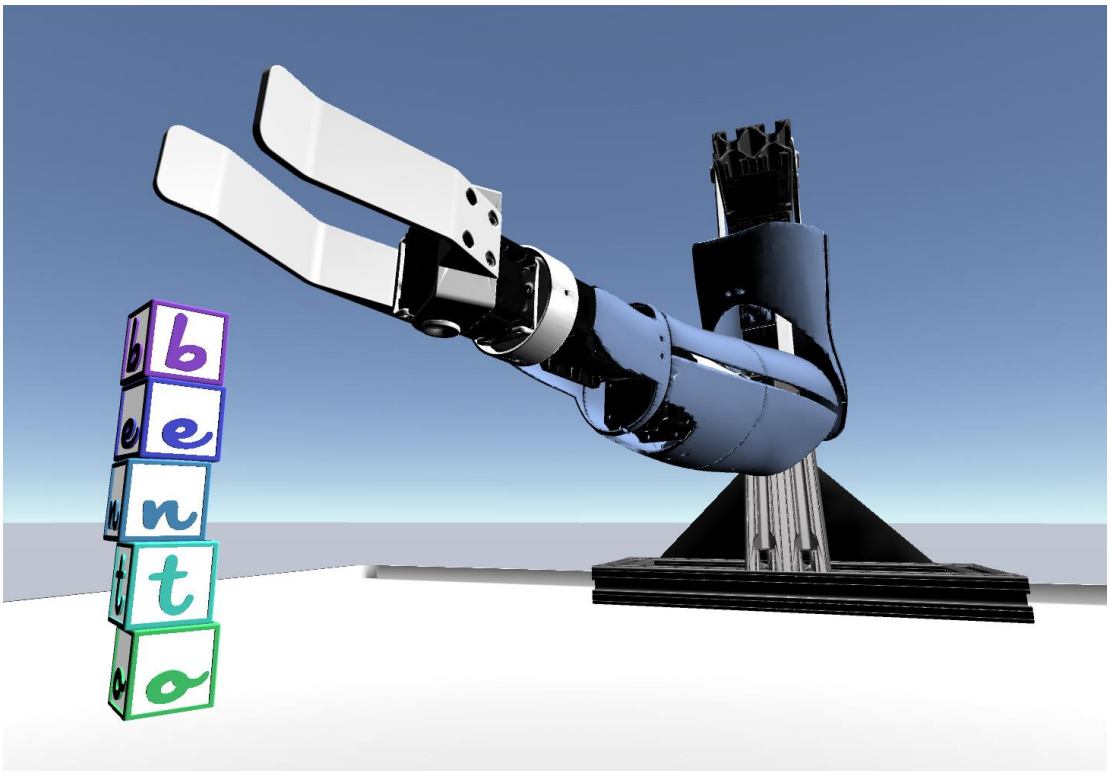


User Guide

Helen Zhao, Michael R. Dawson

July 21, 2021

Table of Contents



Virtual Bento - User Guide

1.1 Introduction

This guide will help you from start to finish with installing and operating the Virtual Bento Arm software. The purpose of the software is to act as a complementary platform to the physical Bento Arm for myoelectric training and research applications. The latest release of this software includes a realistic simulation of the Bento Arm that can be viewed on an LCD monitor or laptop display, adjustable camera views, and six tasks, five of which include objects that the arm can interact with as part of a virtual environment. Future releases will include more gripper options, virtual reality headset support, and additional tasks.

NOTE: The condensed title 'Virtual Bento' is used in the logo and title texts where space is limited, but in long form text we also sometimes use the terms 'Virtual Bento Arm' or 'Virtual Bento Arm Environment' to refer to the same software.

1.2 System Requirements

Supported Operating Systems

- Windows 10

NOTE: The Virtual Bento Arm has only been tested on Windows 10 and in web browsers via the WEBGL build options, but it may also possibly run on Mac, Linux, Android, or iOS if the build option is selected in Unity (see section 1.6 for more info).

Minimum Hardware Requirements

- 2.3 GHz or faster processor with 4 or more threads
- Integrated Graphics Card (Intel HD Graphics 4600 or better)
- 8 GB of RAM
- 70 MB of available hard disk space

NOTE: The Virtual Bento Arm may work on slower systems, but this is the slowest system we had available to test on where we achieved reasonable performance (i.e. where the GPU did not max out at 100%)

Recommended Hardware Requirements

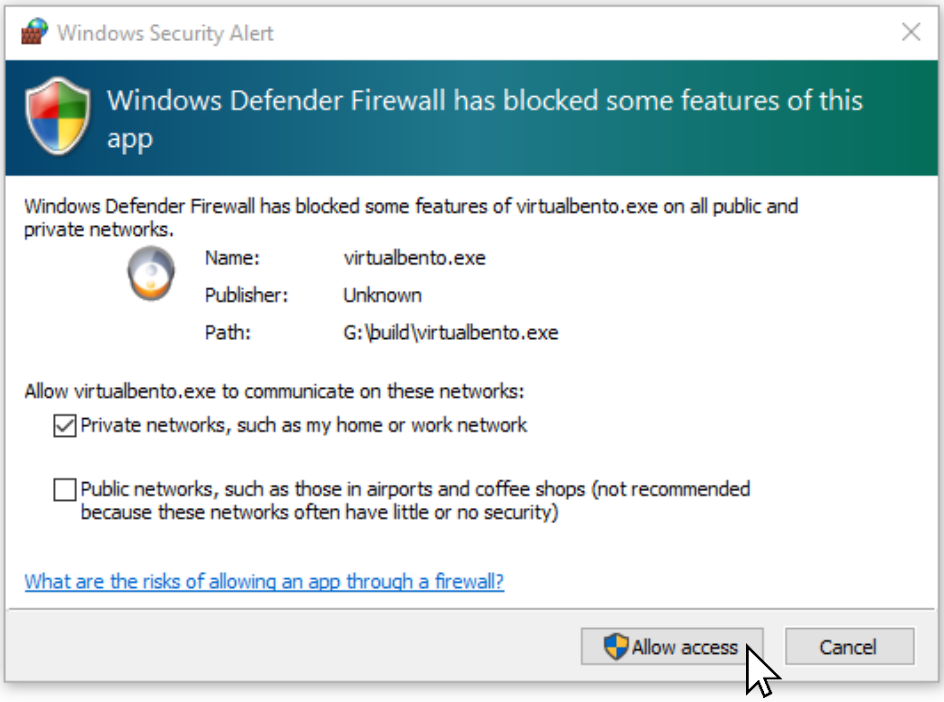
- 3 GHz or faster processor with 6 or more threads
- Dedicated Graphics Card
- 16 GB of RAM
- 600 MB of available hard disk space

Required Peripherals

- Keyboard that includes insert, home, and page up keys
- Mouse that includes a middle click button

1.3 Installation Instructions

1. Download the latest release from github: https://github.com/blincdev/virtual-bento/releases
2. Extract the downloaded files to a folder.
3. All of the files to run the program are included in this folder, so there is no need to run a separate installer. Navigate to the folder where you extracted the files and double click on '`VirtualBento.exe`' to start the program.
    a. Optionally, if you would like to have a desktop shortcut, you can right click on '`VirtualBento.exe`' and select '`Create Shortcut`' and then drag it onto the desktop.
4. If this is the first time you are opening the program it may pop up with the following alert. Select '`Private networks`' and deselect '`Public networks`' and then click '`Allow access`' to continue.



NOTE: This step is necessary in order to allow for an external software such as brachI/Oplexus to drive the virtual arm via a UDP network connection.

5. The program should now launch and looks similar to the image below indicating a successful installation.

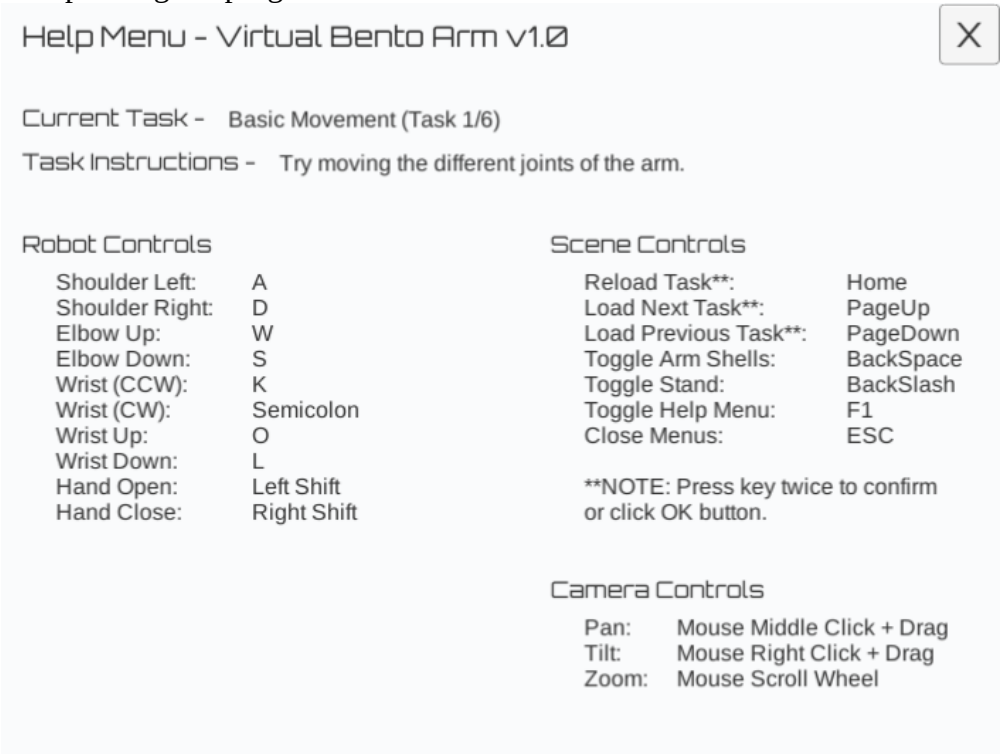
6. For instructions on operating the software please see section 1.4.

1.4 Running the Software

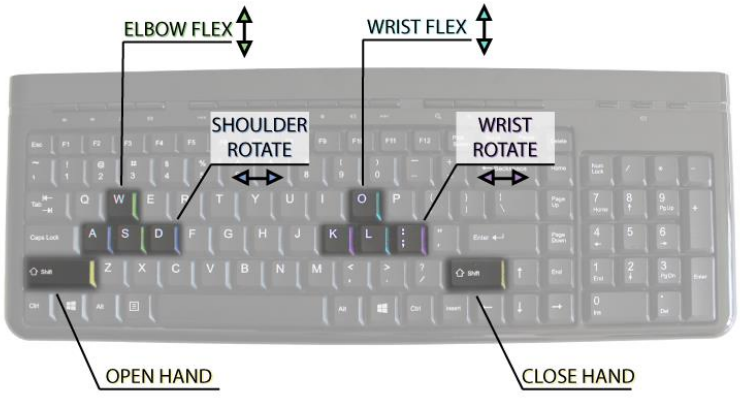
There are two main ways of running the software including a standalone mode where the arm can be controlled with a fixed mapping to a keyboard and an externally driven mode where the arm can be controlled by an external software such as brachI/Oplexus or by your own custom software. Please see the brachI/Oplexus User Guide for detailed instructions on how to use the externally driven mode to control the arm with more input devices such as an Xbox controller and Myo Armband. Please see section 1.5.5 for instructions on how to modify your own custom software to drive the Virtual Bento Arm. This guide will focus on instructions for using the standalone mode:

1. Open the Virtual Bento Arm software by double clicking the shortcut on the desktop or by double clicking on '`VirtualBento.exe`' in the folder where you extracted the files from github.
2. Once it opens you can press any key to continue to the first task. If you do not connect an external software during the initial loading screen then the Virtual Bento Arm will by default proceed in the standalone mode.
3. At any time while the program is running you can press the '`F1`' key on the keyboard to access the help menu which includes instructions for each task, the keyboard keys for controlling the robot and task selection and the mouse buttons for controlling the camera.

4. Here is the help menu from the first task which includes all of the controls for operating the program:



5. Current Task → This is the name of the current task that is selected. In addition to the initial loading page there are 6 tasks that can be selected. They are roughly ordered from least difficult to most difficult.
6. Task Instructions → These are some basic instructions for how to complete the current task. We do not yet have any feedback when you've completed the task, but we plan to add this in a future release.
7. Robot Controls → These are the keyboard controls for moving the individual joints of the Virtual Bento Arm. The joint position and velocity limits of the arm in standalone mode are set to default values that are hard coded in the Unity project. The default joint limits should be suitable for completing all the tasks included in this version of the software. See below for a graphical representation of the robot controls:



8. Scene Controls → These are the keyboard controls for moving between the different scenes (also known as tasks) and for hiding and showing the arm shells and stand. When changing scenes a dialog box will come up asking for you to confirm your selection. You can quickly skip through this by tapping the same key again or you can also click on the buttons in the dialog box using the mouse.
9. Camera Controls → These are the camera controls for adjusting the camera position in the scene. These controls are similar to the ones used to move around the camera in Unity. To achieve a finer zoom than is achievable with the scroll wheel, you can also press '`Ctrl`' on the keyboard while clicking and dragging the right mouse button. By default when you load each task the camera position will re-orientate to the recommended default position and then can be readjusted as necessary.
10. The Virtual Bento Arm pops up in a window that is resizable with a default resolution of 1024 by 768.
11. After you have fun trying the various tasks you can close the software by clicking the '`X`' button in the top right of the window.

1.5 Modifying the Software

This section describes how you can make changes to the software such as adding additional objects and scenes.

1.5.1 Development Environments

We recommend using Visual Studio Express 2015 for Windows Desktop for editing the C# Unity scripts since it is the same version we recommend for editing brachI/Oplexus and newer versions of Visual Studio can sometimes conflict with some libraries in brachI/Oplexus. It can be freely downloaded and installed by individual developers, open source contributors, academic researchers, and small organizations from the visual studio website: https://www.visualstudio.com/vs/visual-studio-express/

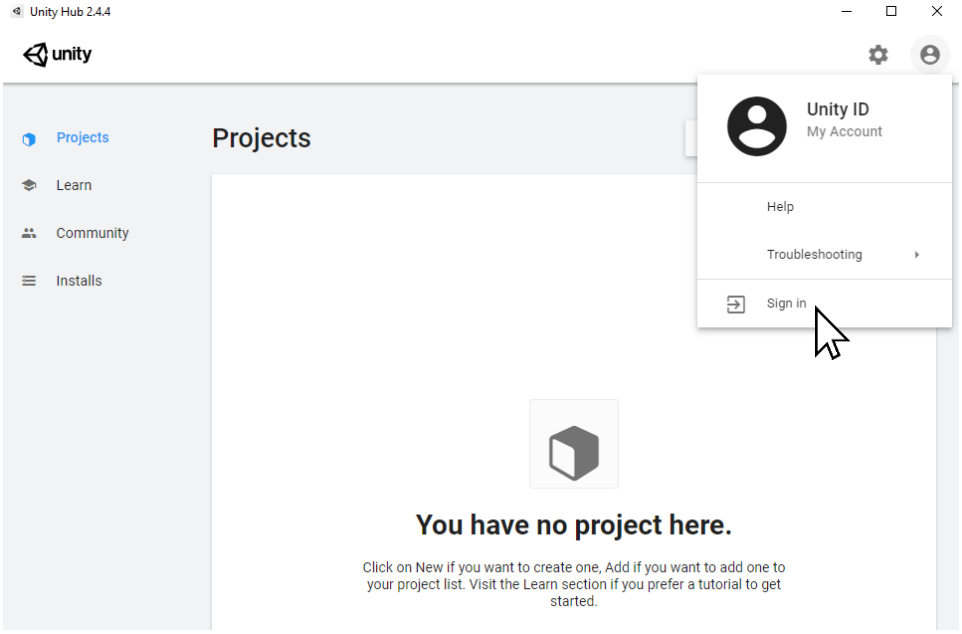
The main development environment used in this project was Unity (Version 2020.1.0b5). Unity has a free personal edition and can be used to edit the main project files including scenes, objects, camera positions, and lighting. To get started you should download and install Unity Hub from the Unity website:
https://store.unity.com/download

NOTE: We recommend installing Visual Studio Express 2015 before Unity since we have had some reports that installing Unity first can lead to subsequent issues with installing Visual Studio.

1.5.2 Opening the Project Files

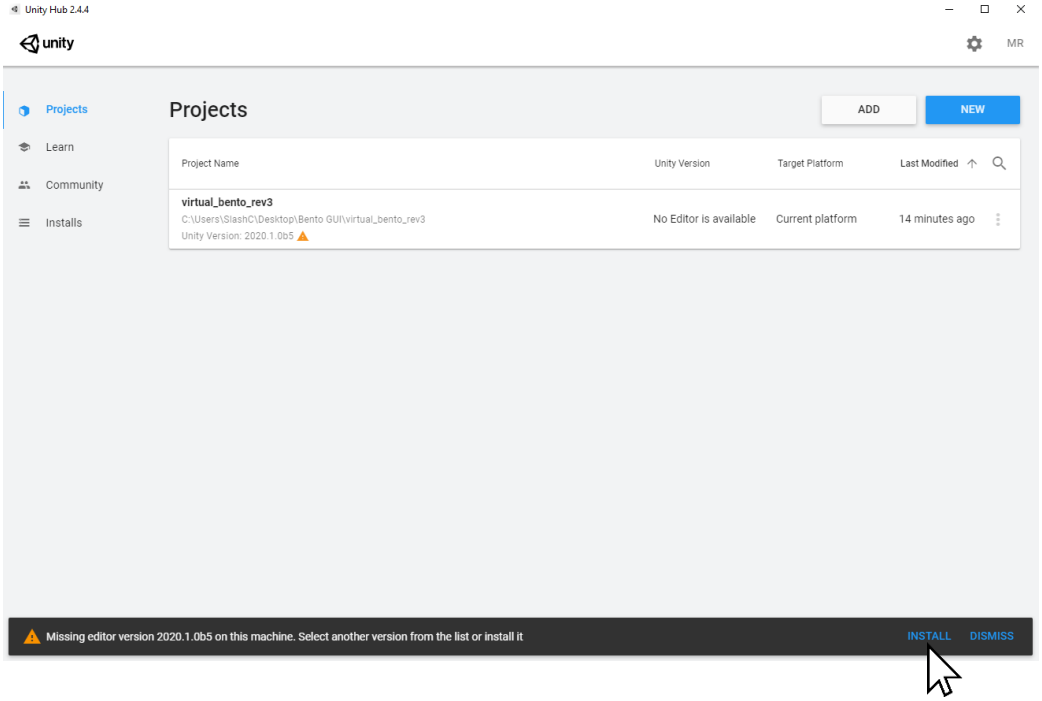
1. To download the project files please visit our github page and click the '`clone or download`' button to download the latest master branch:
   https://github.com/blincdev/virtual-bento
2. Extract the project files to a new directory.

3. Open Unity Hub.
    a. The first time you open Unity Hub you will need to sign into your user account by clicking on the account icon in the top right corner of Unity hub and clicking '`sign in`'. If you don't have an account it will give you an option to create one.

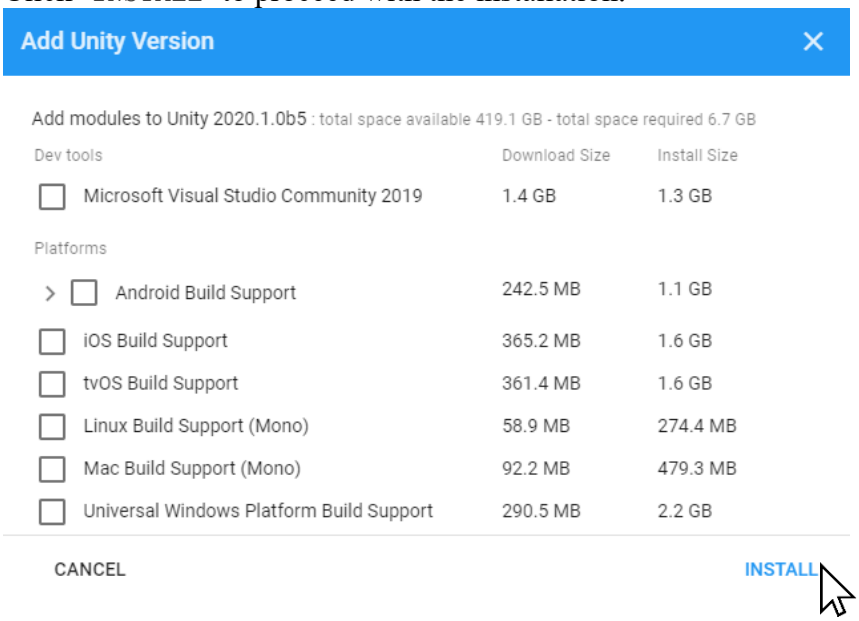


4. After signing in you will have to activate a new license which you can do by clicking on the account icon in the top right corner of unity and selecting '`Manage License`'.
    a. Then click the '`ACTIVATE NEW LICENSE`' button and select the appropriate license agreement.
    b. Click '`DONE`' to finish activating the license.
5. Click the back button in the top left of Unity Hub to go back to the '`Projects`' page.
6. The easiest way to install the rest of Unity is to click on the '`ADD`' button in Unity Hub and select the project files directory where you extracted the files from github.

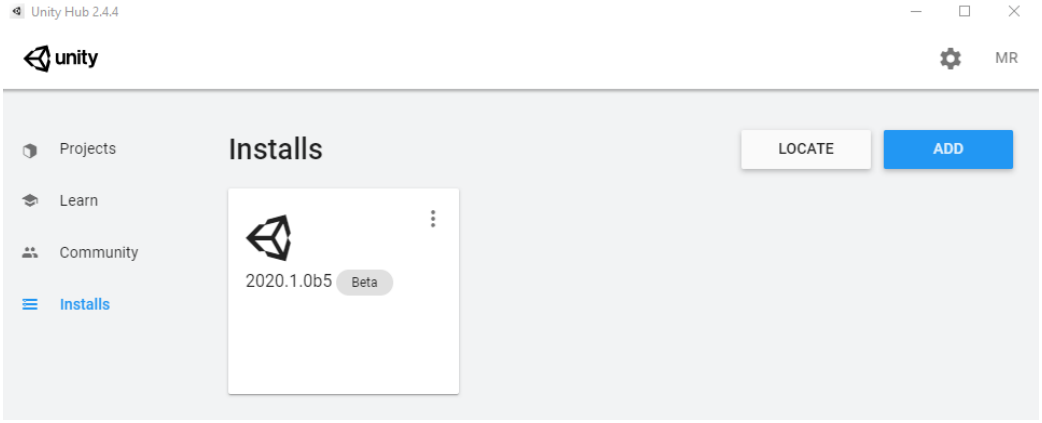
7. Click on the project and a notice will come up at the bottom of the screen that will let you install Unity Version: 2020.1.0b5. Click the 'INSTALL' option to proceed.



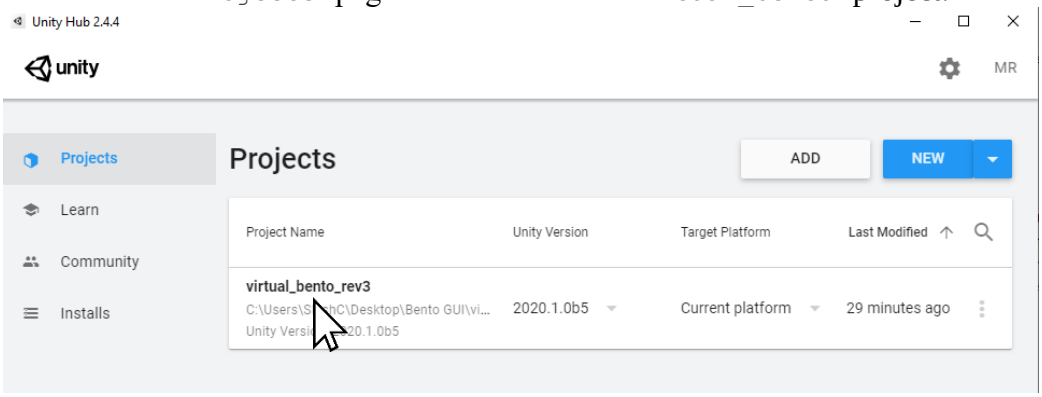
8. A window will pop up for adding the Unity version.
    a. Deselect 'Microsoft Visual Studio Community 2019'. This secondary software is not required since you can use Visual Studio Express 2015 for editing the unity scripts.
    b. The only thing that will still be auto-selected is 'Documentation' which is fine and then leave the other platforms unselected. You can always install them later if you decide to deploy on a different platform.
    c. Click 'INSTALL' to proceed with the installation.



9. Unity Hub will change to the '`Installs`' page and after a few minutes Unity version 2020.1.0b5 will be downloaded and installed.

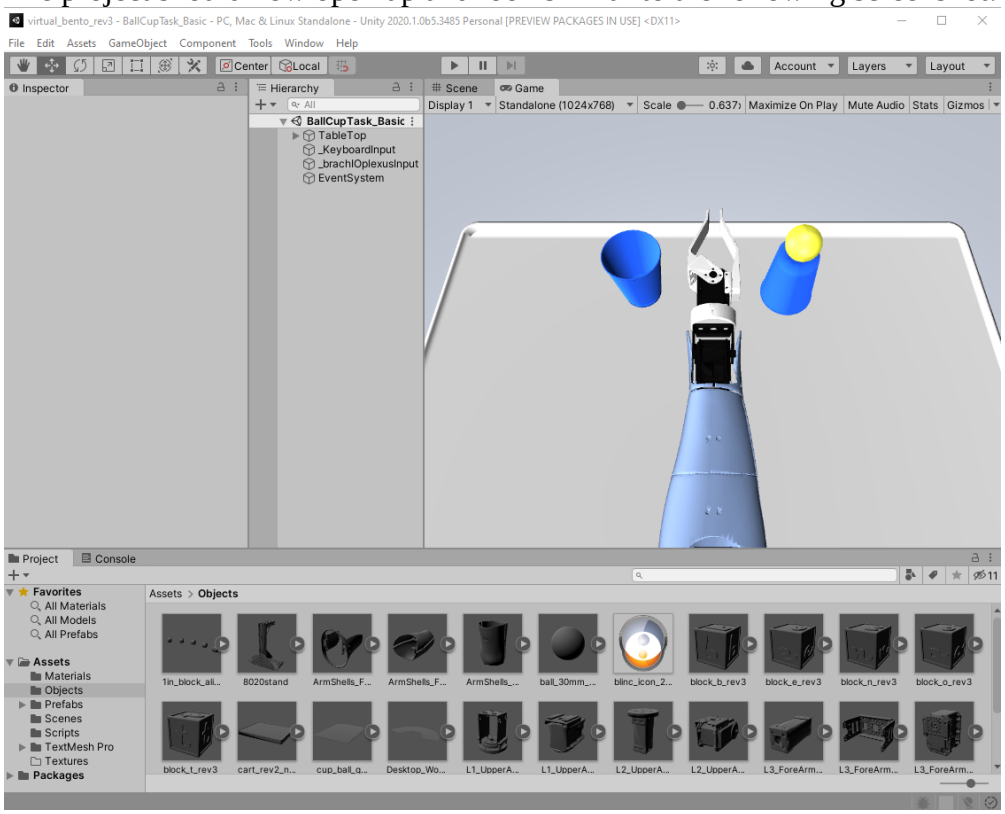


10. Return to the '`Projects`' page and click on the '`virtual_bento`' project.

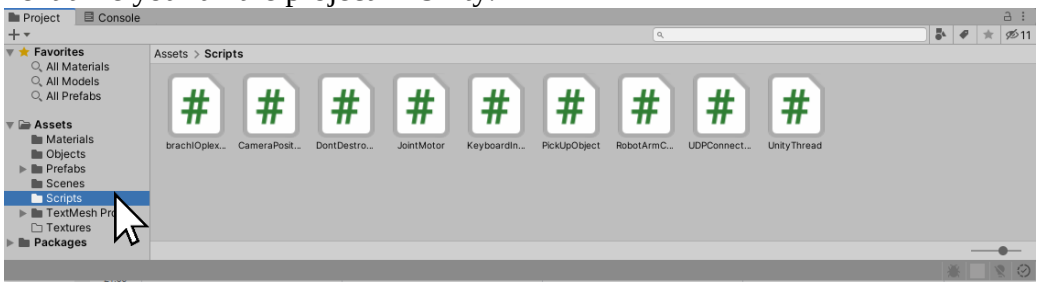


11. When it first opens a window may pop up notifying that there is a new version of the Unity Editor available. It is possible that the project works fine on newer versions, but we have not had a chance to test it, so we recommend for now clicking '`Skip new version`'.

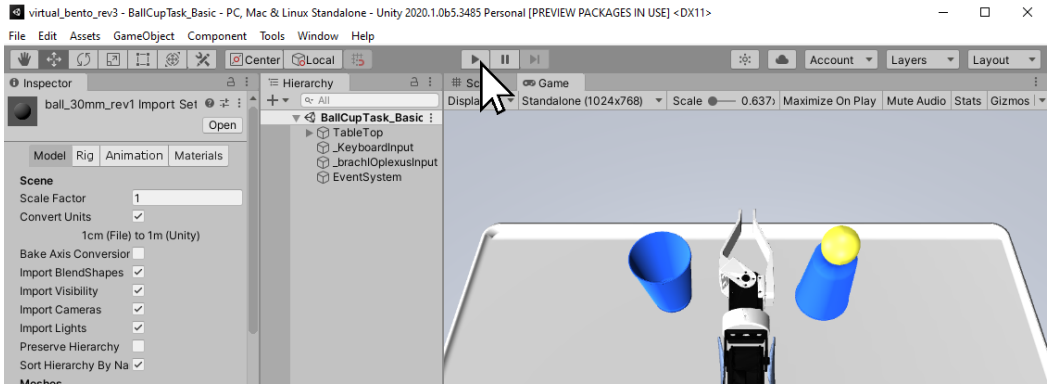
12. The project should now open up and look similar to the following screenshot:



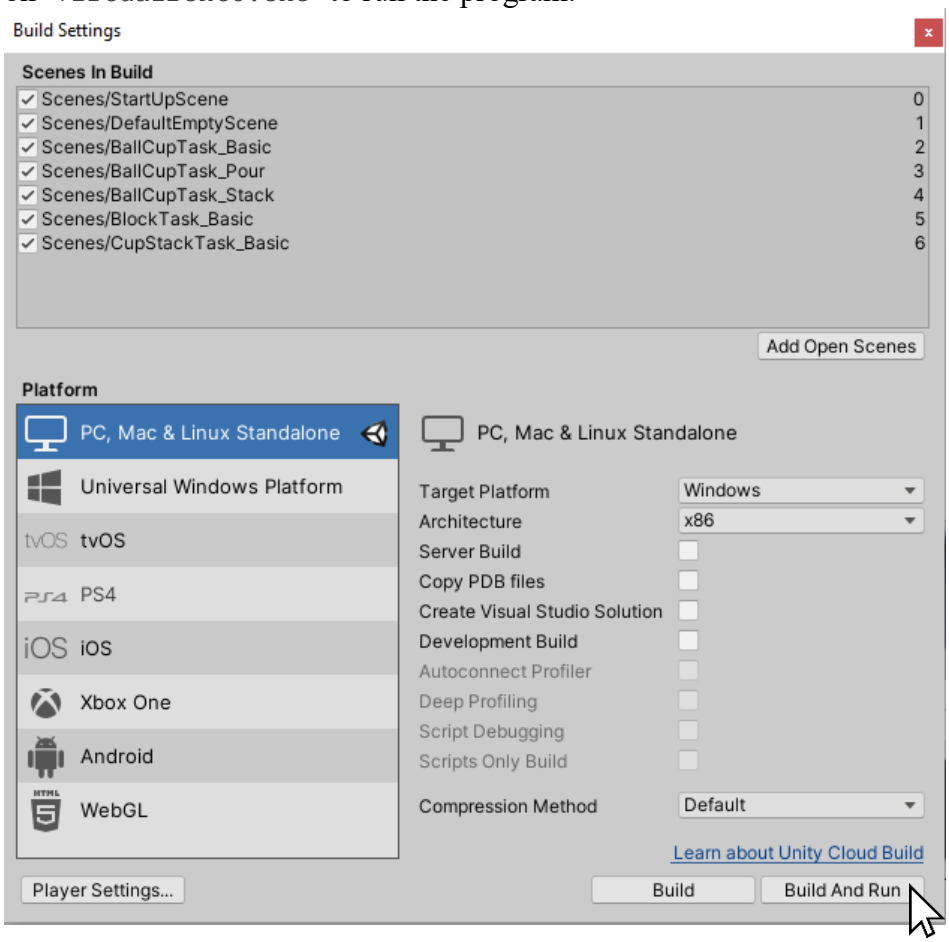
13. To view the source code you can click on '`Assets\Scripts`' in the '`Project`' Tab and then double-click on any of the scripts. The script will then open in Visual Studio 2015 where it can be edited and re-saved in order to take effect the next time you run the project in Unity.



14. To run the project you can click on the '`Play`' button near the top and then use the controls as defined in section 1.5 to control the virtual arm and camera and navigate the tasks.



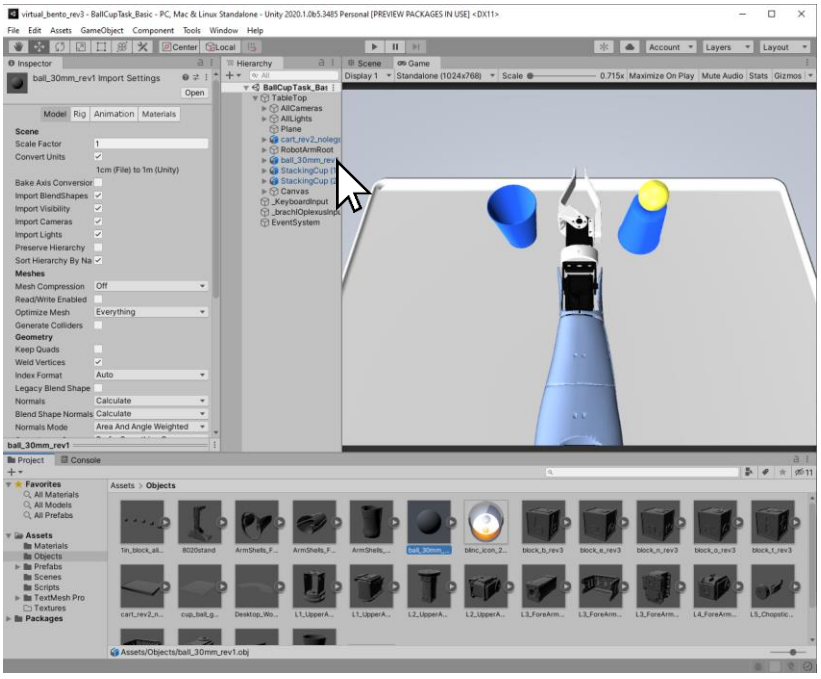
15. To build a standalone version of the project that can run separately from Unity you can go '`File → Build Settings`' and then click '`Build And Run`'. It will prompt you to select a folder to place the build files and then will automatically run the executable. You can also navigate to the selected folder and double-click on '`VirtualBento.exe`' to run the program.



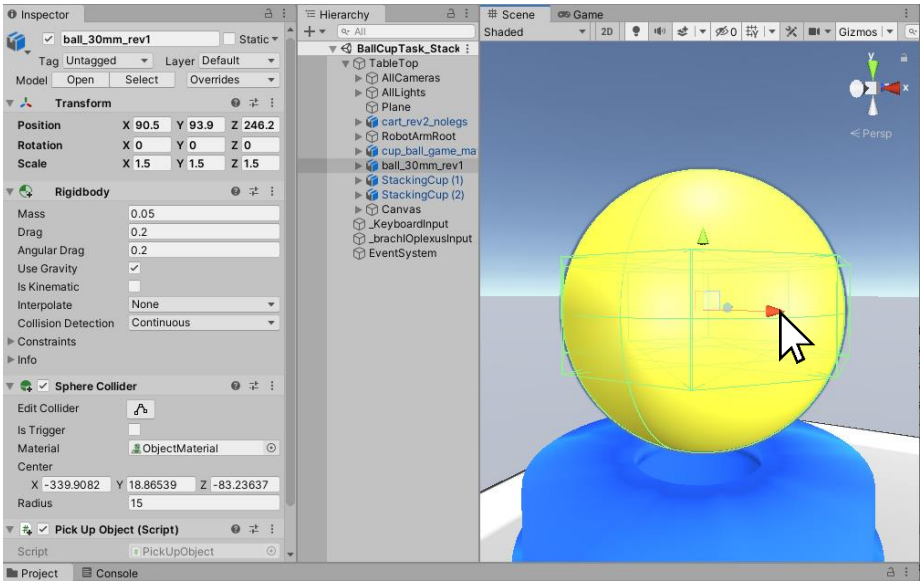
1.5.3 Adding an Object

You can use any 3D modelling program that can export to the '.STL' or '.OBJ' format to create your 3D models. One program that you might consider using is Blender which is open source, freely available, and commonly used with Unity. For the following example we used the Solidworks software since it was also the software that the Bento Arm Hardware was designed in. The exact procedure will vary depending on which 3D modelling program you start with and what file types it can export to.

1. Design your object in Solidworks or a similar 3D modelling program.
    a. Go '`File → Save As`' and change the '`Save as type:`' to '`STL (*.stl)`'.
    b. You can also adjust the detail of the model in the 'Options…' setting if desired.
    c. Click '`Save`' to finish saving the file in the new format.
2. Convert the .STL files into .OBJ files in Autodesk Meshmixer
    a. Go '`File → Export`' and change the '`Save as type:`' to '`OBJ Format (*.obj)`'.
    b. You can adjust the detail and file size of the model by going '`Select → Modify → Select All`' and then going '`Select → Edit → Reduce`'. We find it is a bit easier to adjust the detail in Meshmixer compared to Solidworks.
    c. Click 'Save' to finish saving the file in the new format.
3. Import the .OBJ file into Unity
    a. Drag the .OBJ file into the '`Assets\Objects`' folder. This will import it into Unity.
    b. Now you can drag the object from the '`Assets\Objects`' folder to the '`TableTop`' parent object in the '`Hierarchy`' tab of a given scene as seen below:



4. Finish setting up the object in Unity, so that it can interact with the arm and the environment
    a. Click on the '`Add Component`' button in the '`Inspector`' window and select '`Rigidbody`'. You can also scale and change the mass of the object.
    b. Click on the '`Add Component`' button in the '`Inspector`' window and select '`Box Collider`' or '`Spherical Collider`' depending on the geometry of the object. We recommend using the simplest collider that you can get away with for an object. More complex colliders such as mesh colliders can make the physics glitchy or cause the model to slow down. If necessary you can for example use multiple box colliders to approximate the shape of a complex object. Also be careful of using overlapping colliders since this sometimes also causes the object to have glitch or slowed down interactions.
    c. Click on the '`Add Component`' button in the '`Inspector`' window and select '`Pick up Object (Script)`' object.
    d. You can now select the '`Scene`' tab and drag around the object to the desired location in the scene.



NOTE: For most applications it is sufficient to just eyeball the positioning of the object in Unity by dragging it around. However, if you need exact positioning we recommend importing it into the Bento Arm assembly in Solidworks, position it exactly, hide all arm models, and then when saving the remaining object as an .STL go into options and select "save all compnents of an assembly in a single file". This will retain the original coordinate system of the Bento Arm model, so that when you import it into Unity it is placed exactly where you want it.

1.5.4 Adding a Scene

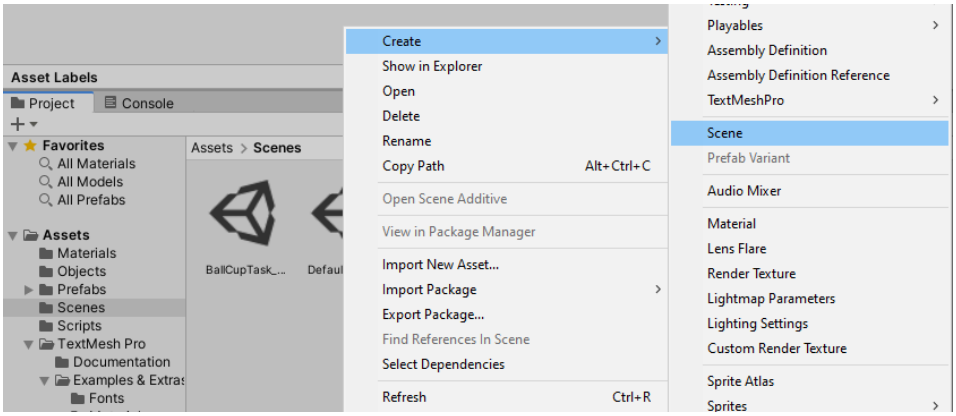
The simple way to create a new scene is to duplicate one of the existing scenes that is the closest to the scene that you want to make. There is also an advanced procedure that includes steps for building a new scene from scratch. Both of the methods are described in the next sections.

Simple Method

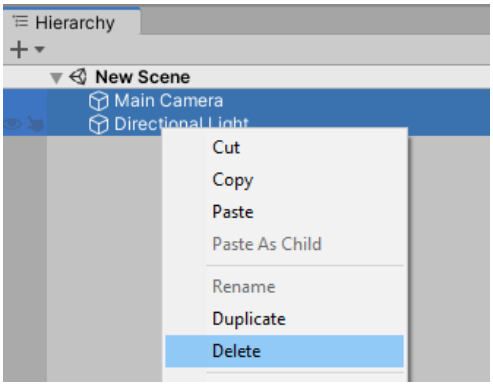
1. Open the project in Unity and select the closest scene in the '`Assets\Scenes`' directory in the '`Project`' Tab.
2. Go '`Edit → Duplicate`'.
3. Right click on the duplicated scene and select '`Rename`' to name it something new.
4. Add the new scene into the '`Scenes in Build`' list by going '`File → Build Settings`' and clicking the '`Add Open Scenes`' button. You can drag it up or down to re-order where it appears in the scene list.

Advanced Method

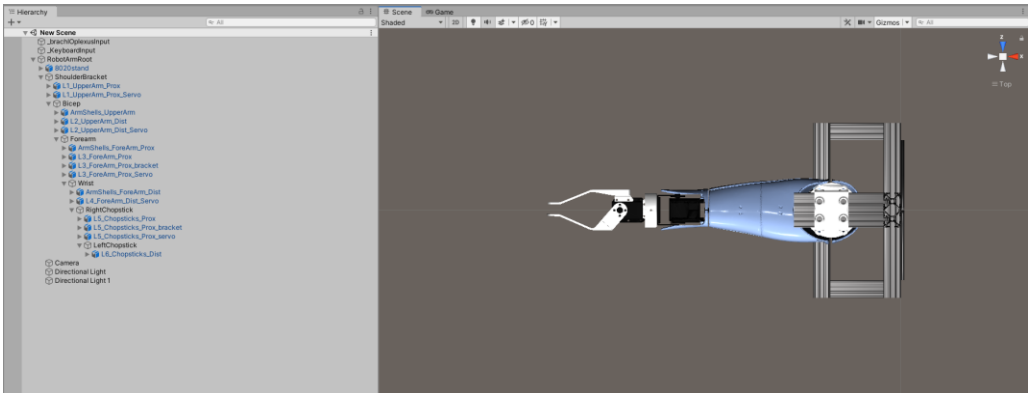
1. Open the Unity project and navigate to the '`Assets\Scenes`' directory.
2. Create a new scene in this directory and rename the scene with the format '`TaskName_DifficultyLevel`'.



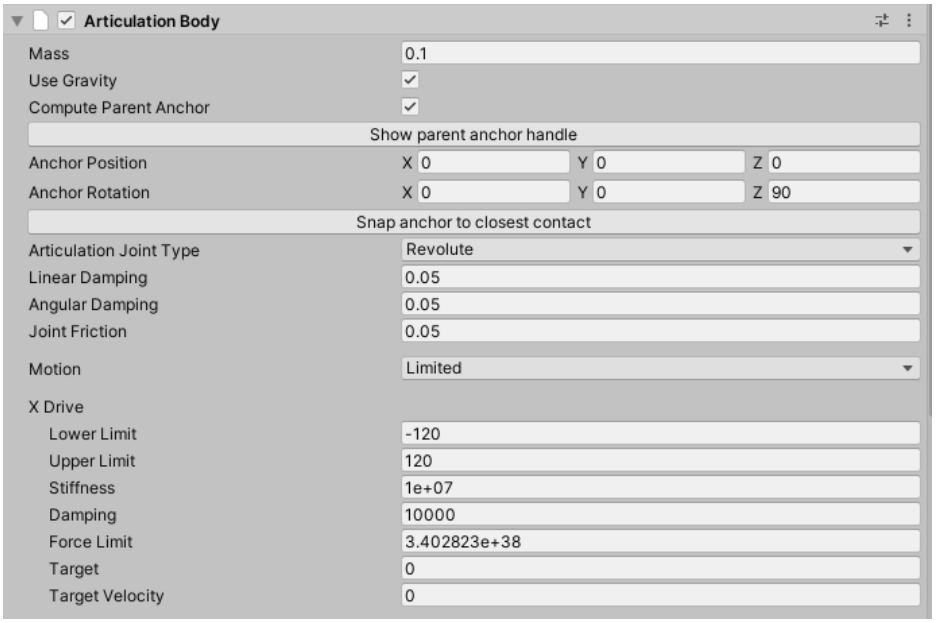
3. Delete all default camera and lighting objects in the new scene and save the new scene.



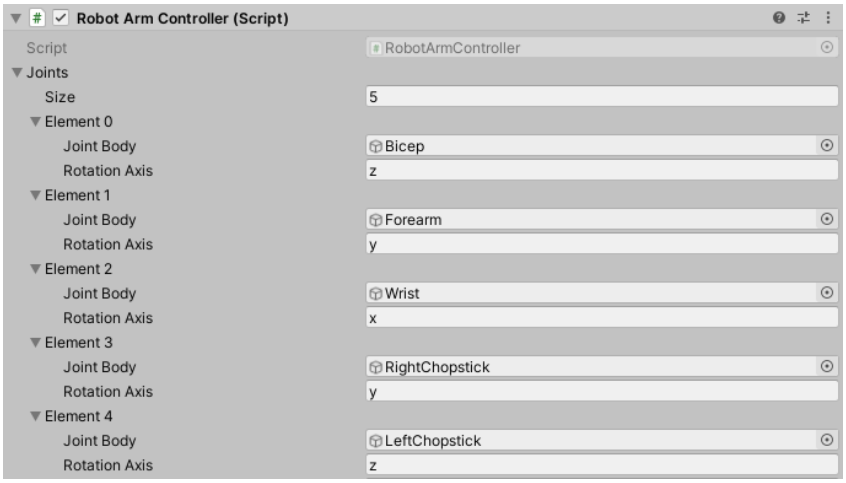
4. Open the '`DefaultEmptyScene`' scene and copy all game objects from the '`DefaultEmptyScene`' scene to the new scene.



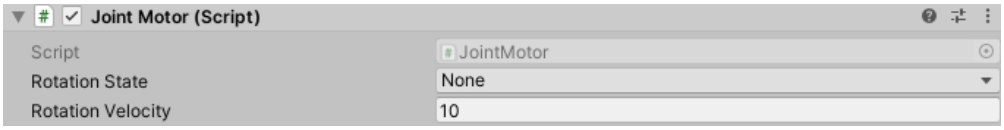
5. Make sure that the '`Articulation Body`' component is attached to all the following game objects: '`RobotArmRoot`', '`Bicep`', '`Forearm`', '`Wrist`', '`RightChopstick`' and '`LeftChopstick`'.



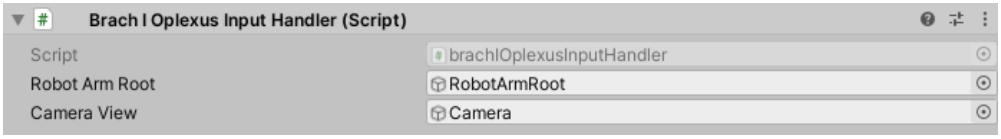
6. Make sure that the '`RobotArmController`' script is attached to the '`RobotArmRoot`' object and that the rotation axis for each joint is configured as the following:



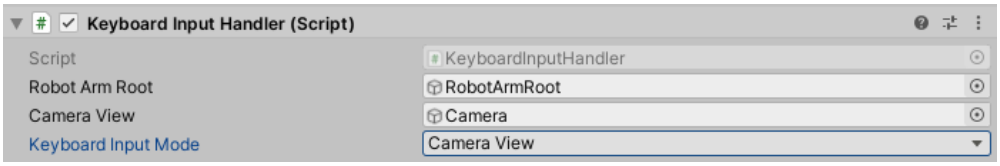
7. Make sure that the '`JointMotor`' script is attached to all the following game objects: '`Bicep`', '`Forearm`', '`Wrist`', '`RightChopstick`' and '`LeftChopstick`'.



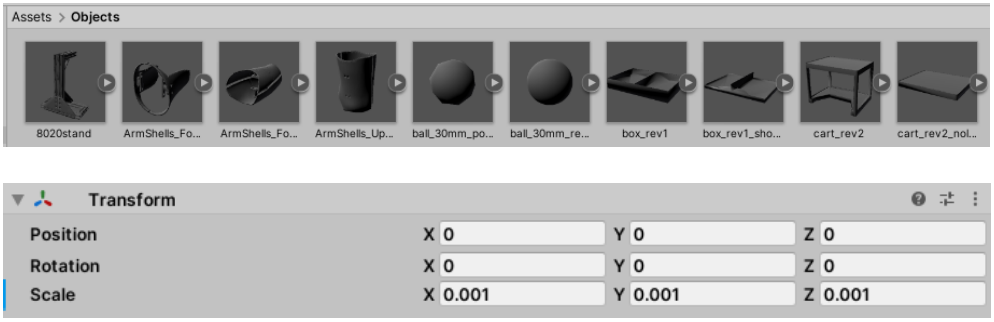
8. Make sure that the '`brachIOplexusInputHandler`' script is attached to the '`_brachIOplexusInput`' game object and is connected to a robot arm and camera game object.



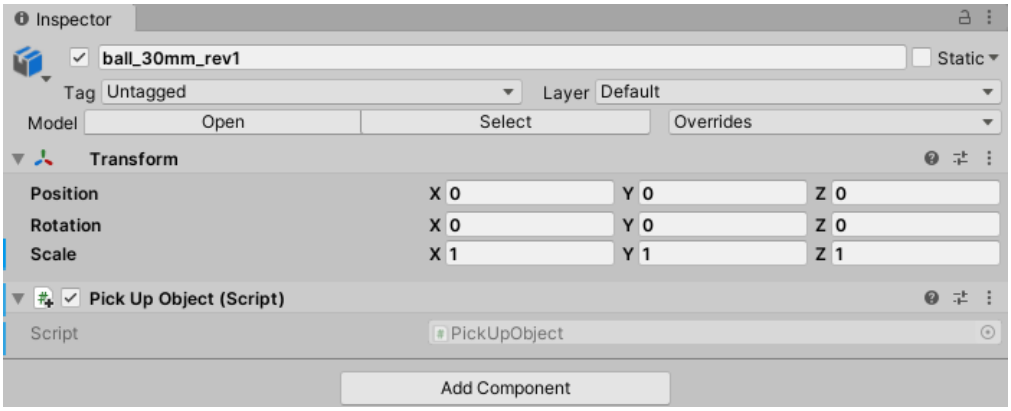
9. Make sure that the '`KeyboardInputHandler`' script is attached to the '`_KeyboardInput`' game object and is connected to a robot arm and camera game object.



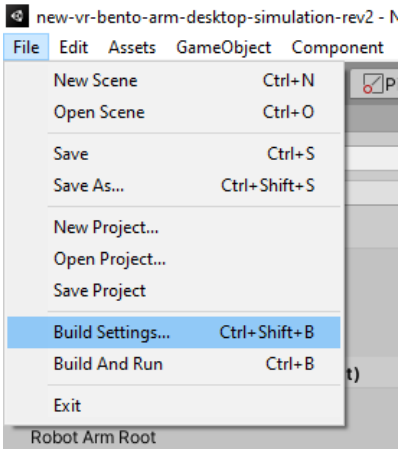
10. Drag and drop additional game objects from the '`Assets\Objects`' directory into the scene hierarchy in order to add them to the scene; new added objects need to be scaled to a size of 0.001.



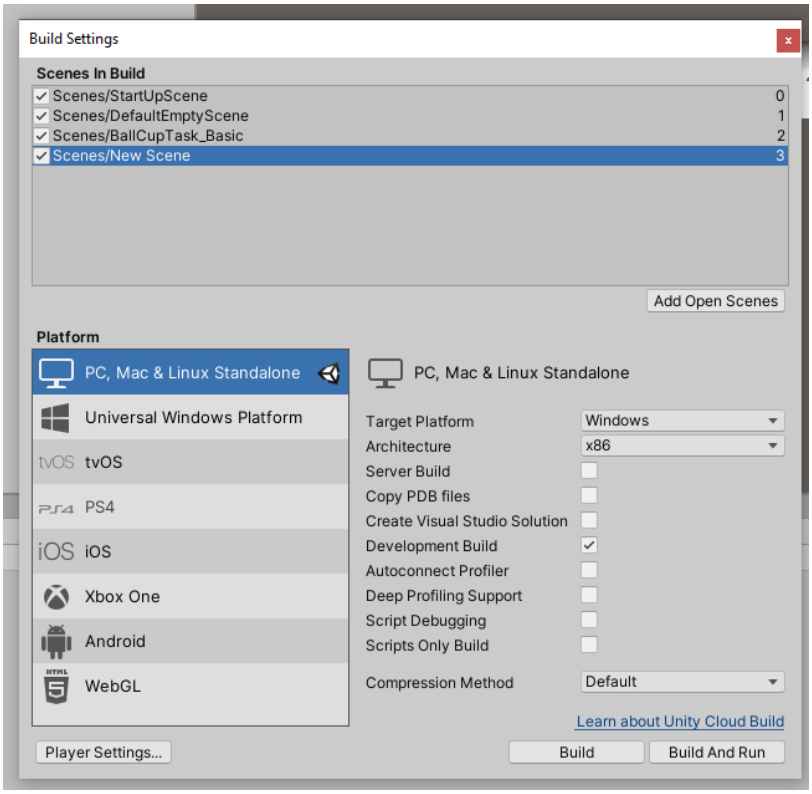
11. For all objects that need to be picked up, navigate to the '`Assets\Scripts`' directory and attach the '`PickUpObject`' script to the object by dragging the script and dropping it on the object.



12. From the unity editor menu, select '`File → Build Settings`' to open the build settings window.



13. Add the open scene to the build and check the build index of the new scene (Note: You can reorder the scenes, just make sure that the '`StartUpScene`' always has a build index of 0, and the ordering of the remaining scenes in build settings is the same as in brachI/Oplexus).



14. Optionally if you would like to create a new default camera position for the new scene you can open '`brachIOplexusInputHandler.cs`' and add a new entry to the '`defaultCameraPositions`' array. If you skip this step and put your new scene at the end of the build list then the new scene will by default use the camera position from scene 0.
15. Click on '`Build and Run`' to build the Unity executable.

1.5.5 Externally Driven Mode

The externally driven mode allows the Virtual Bento Arm to be controlled with an external software via a UDP network interface. Using an external software to control the Virtual Bento Arm unlocks several advanced features including the ability to create flexible mappings (direct or machine learning based) between various input devices and the movements of the arm, saveable camera positions, adjustable joint limits, and the ability to receive feedback from the arm that can be used for a number of applications including providing haptic feedback and automating joint movements. The interface requires two UDP ports and 4 threads with the following specifications and can operate at rates of up to 65 Hz with minimal CPU usage.

- External Software → Unity communication
  - Port: 30004
  - IP Address: 127.0.0.1
  - 2 threads: A transmit thread in the external software that sends a packet every 15ms and a receive thread in Unity that blocks until it receives a packet.
- Unity → External Software communication
  - Port: 30005
  - IP Address: 127.0.0.1
  - 2 threads: A transmit thread in Unity that sends every 15ms and a receive thread in the external software that blocks until it receives a packet.
- *NOTE: You should use an IP Address of 127.0.0.1 (localhost) when the external software and Unity are on the same computer, but you can also specify different IP addresses if you want them running on different computers.*

A summary of the types of communication packets can be found in the following table:

Type ID	Type Name	From	To	Type Description
0	Scene Packet	External Software	Unity	Contains the scene index, frame rate, and initial camera positions and joint limits for scene initialization in Unity
1	Control Packet	External Software	Unity	Contains the ID, motor state and velocity of each joint in the external software
2	Update Packet	External Software	Unity	Contains the updated camera positions and joint limits for a running scene
3	Camera Request Packet	External Software	Unity	An empty packet that acts as a request to retrieve current camera positions from Unity
4	Feedback Packet	Unity	External Software	Contains the ID, current position and current velocity of each joint in Unity
5	Camera Position Packet	Unity	External Software	Contains the current camera positions in Unity

For more specific details about the packet structures please the "UDP_Packet_Structure.pdf" document.

An example of an implementation of this interface can be found in its integration with the latest version of brachI/Oplexus. Please see the '`Communication with unity`' region in '`mainform.cs`' of brachI/Oplexus to see one possible way of coding up the interface. The corresponding script in the Virtual Bento Arm Unity project that handles the UDP communication is '`UDPconnection.cs`'.

1.5.6 Contributing to the Project

If you are interested in contributing to the development of the program or have ideas for future features please contact us through our website: https://blincdev.ca

1.6 Acknowledgements

We would like to thank the University of Alberta (UofA), Alberta Machine Intelligence Institute (Amii), Bionic Limbs for Improved Natural Control (BLINC) Lab, the Sensory Motor Adaptive Rehabilitation Technology (SMART) Network, Alberta Innovates, Canadian Research Chair (CRC) program, Canadian Foundation for Innovation (CFI), Canada CIFAR AI Chairs program, and the Natural Sciences and Engineering Research Council of Canada (NSERC) for their continued support in this project.