Karl Dadivas
500345383
CPS 643 Virtual Reality
20/04/19

USER MANUAL:

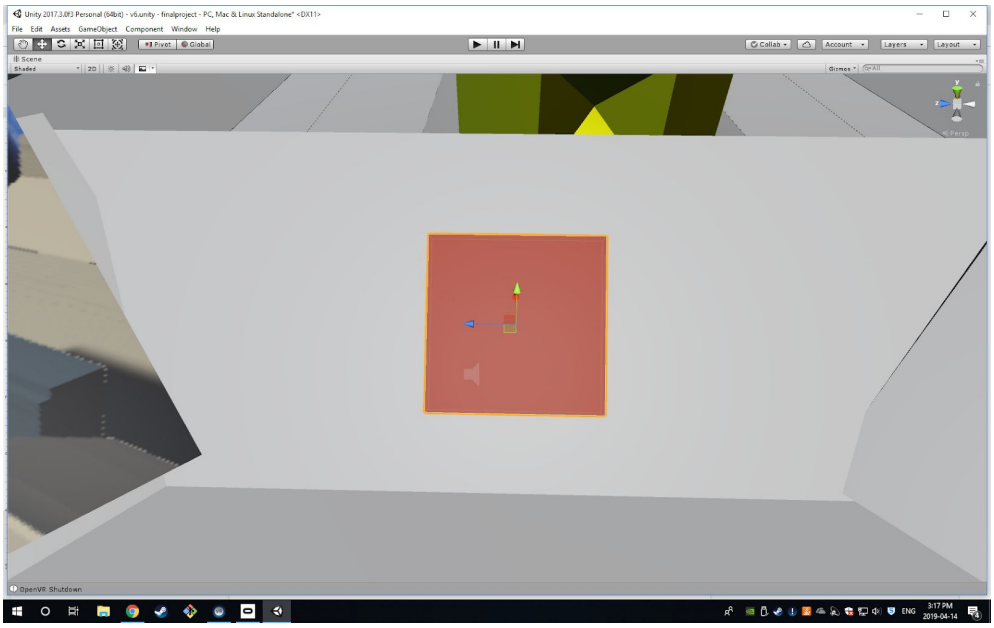
How to install:
Download project folder, and run .exe file to play simulator.

Setup:

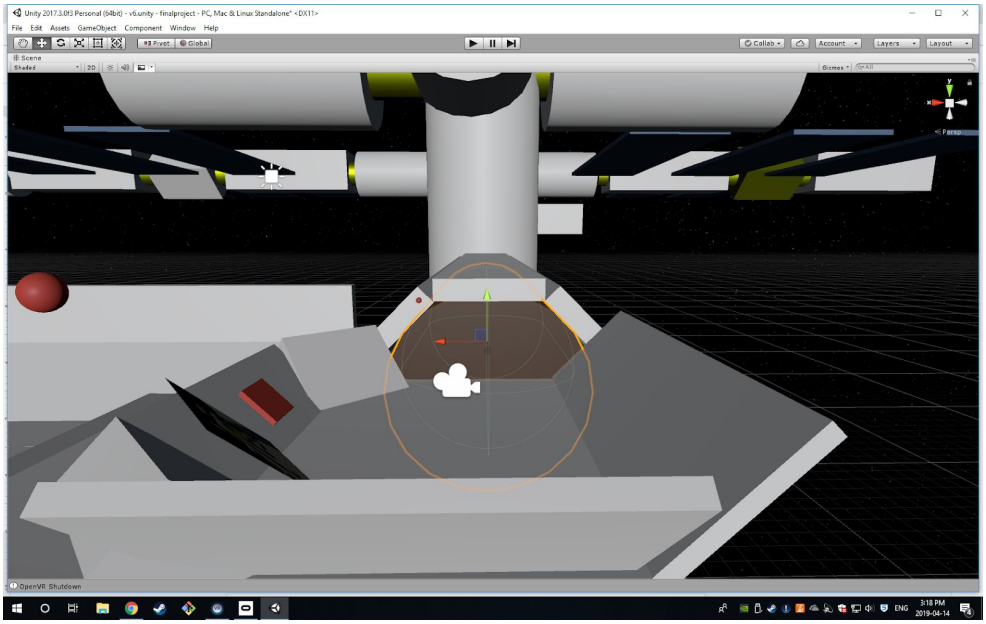
The simulator can be played seated or standing and requires no movement from the player. Ensure that there is enough room for the arm to move freely. The game scene has two levels, “spaceship”, and “spacestation”. There are 3 buttons that are used in the game:

-Right Trigger
-Right Grip
-”A” button

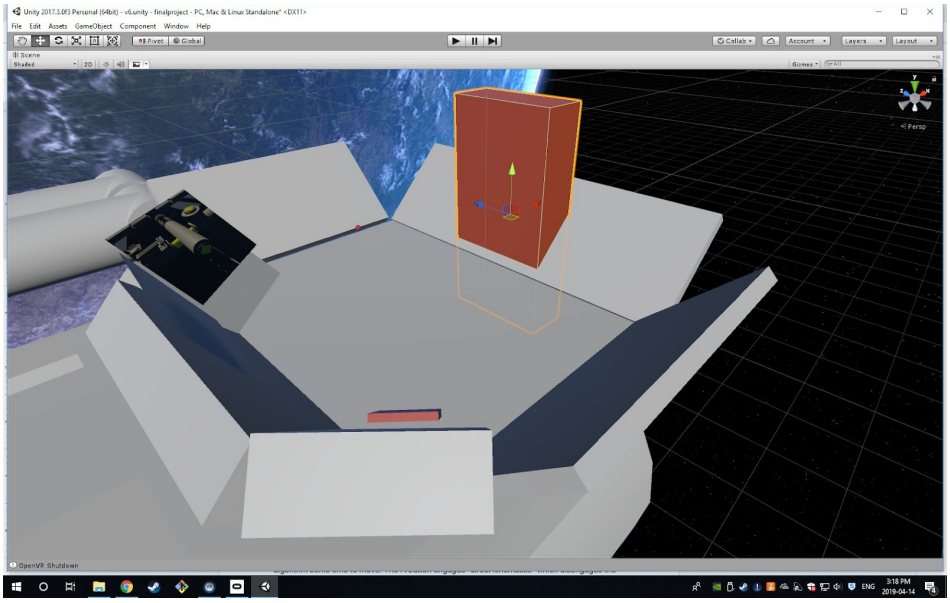
The right trigger is used to interact with game objects via ray cast collision. Some objects to interact with are the red buttons,or black cylinders which teleports the player to different locations in the scene.



Red buttons to open doors or reset game

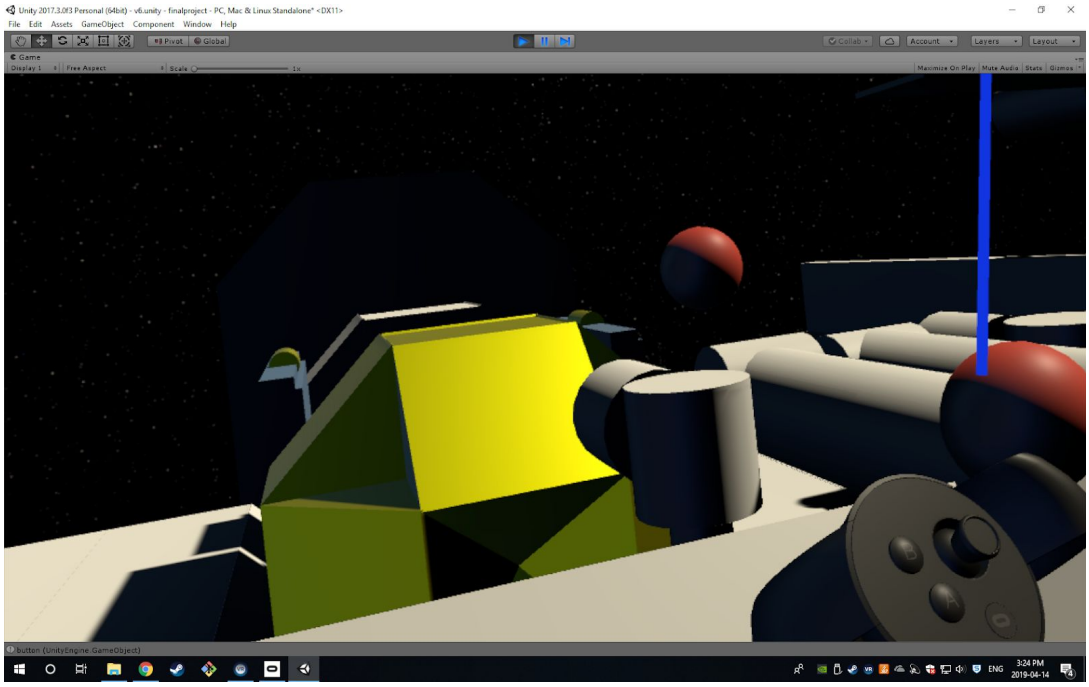


Black cylinder to space station

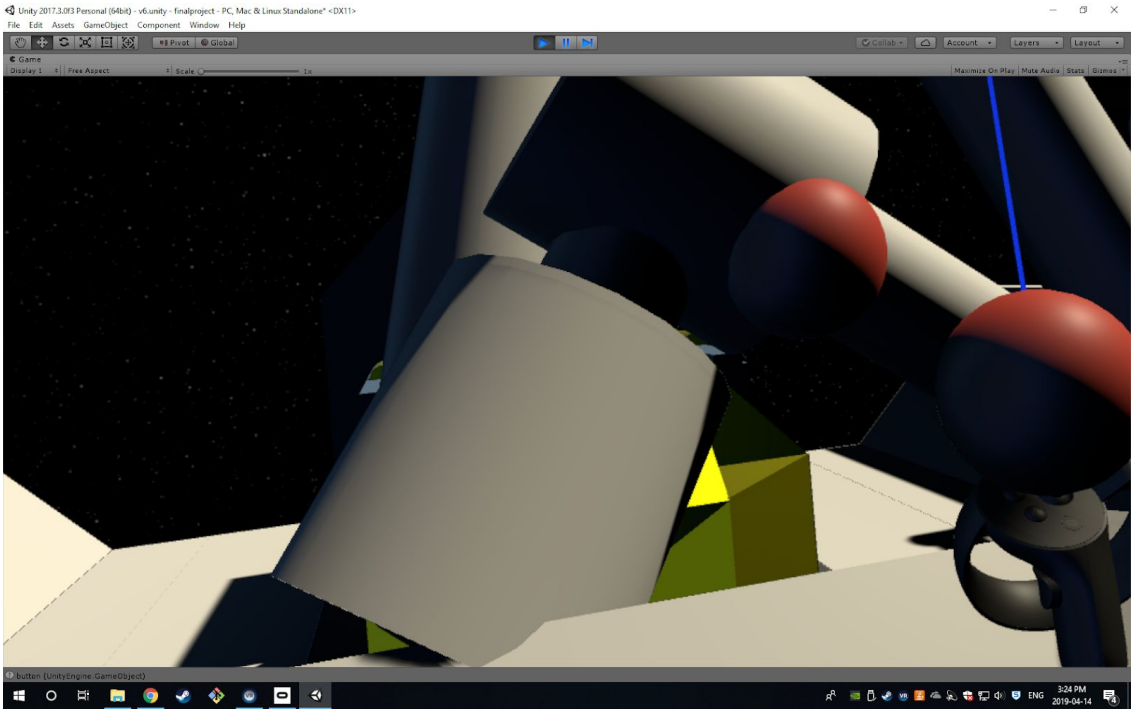


Red door to space ship

The right grip sets the position of the shoulder joint of the arm relative to the player. For best results press the right grip button when the controller is close to the players shoulder. When the grip is pressed the robotic arm will begin following a red ball. The red ball moves relative to the players controller.

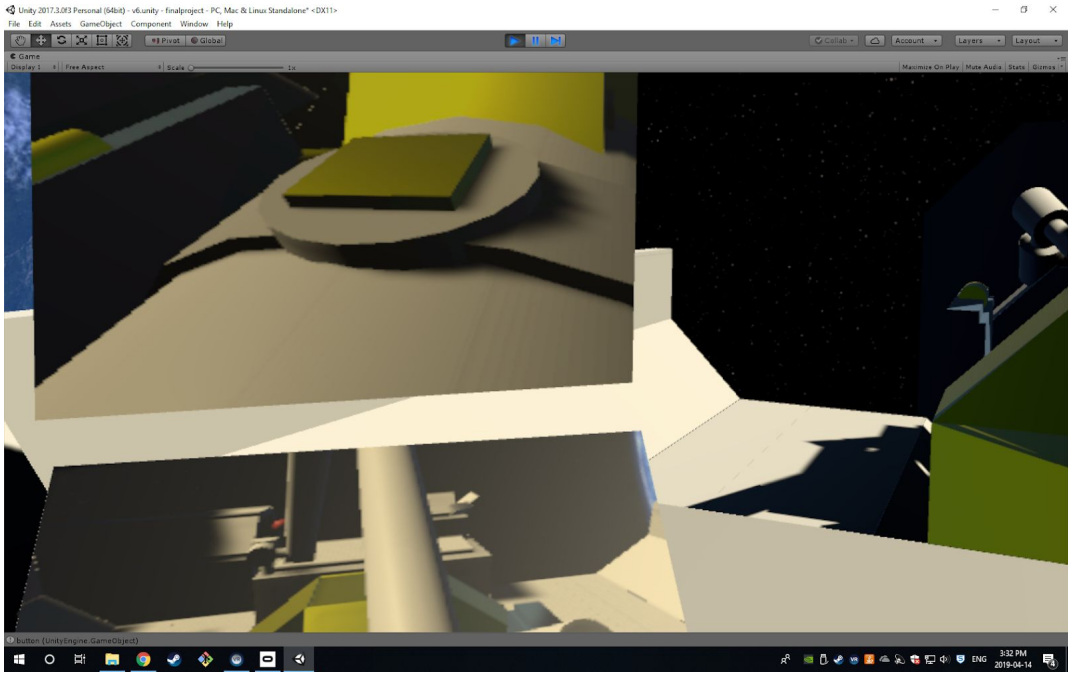


Before grip press

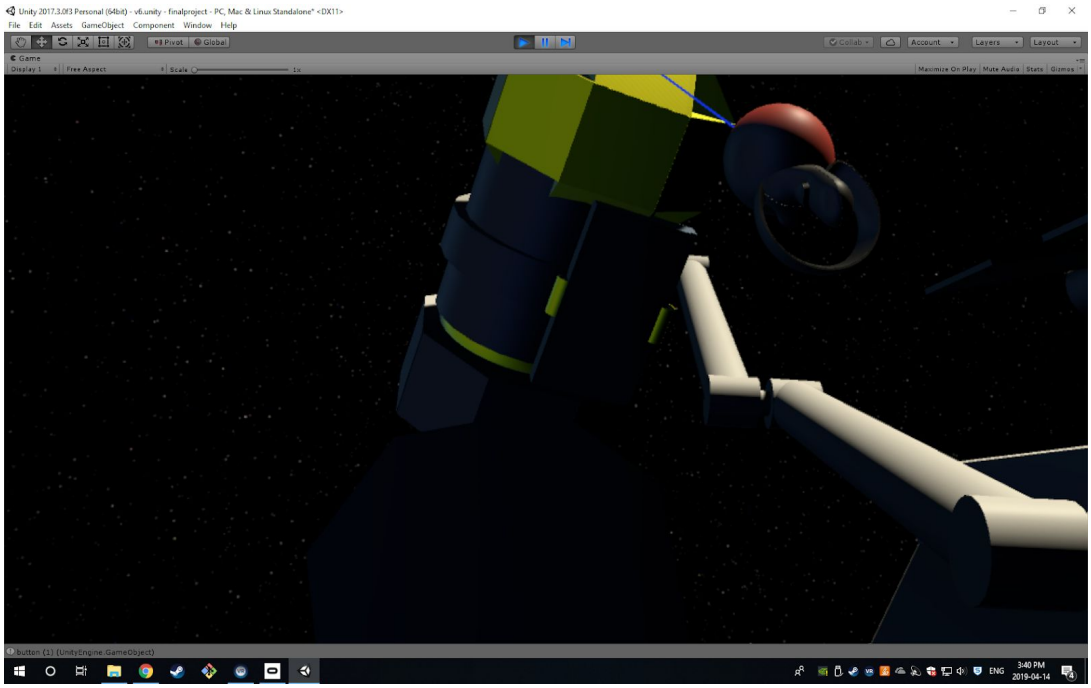


After grip press, and waiting for robotic arm to reach target position..

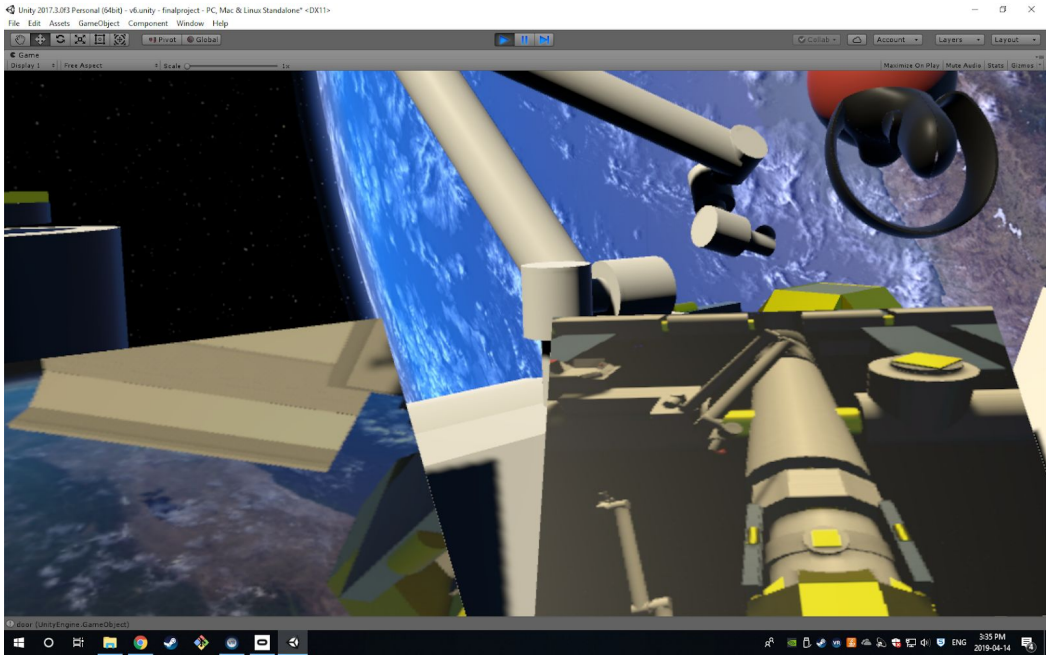
The effector ball moves the robotic arm via inverse kinematics and should allow the algorithm some time to move to the specified location. The goal of the game is to align the end effector of the arm to the yellow cube on the module. There are screens to display different camera angles to assist in guiding the arm.



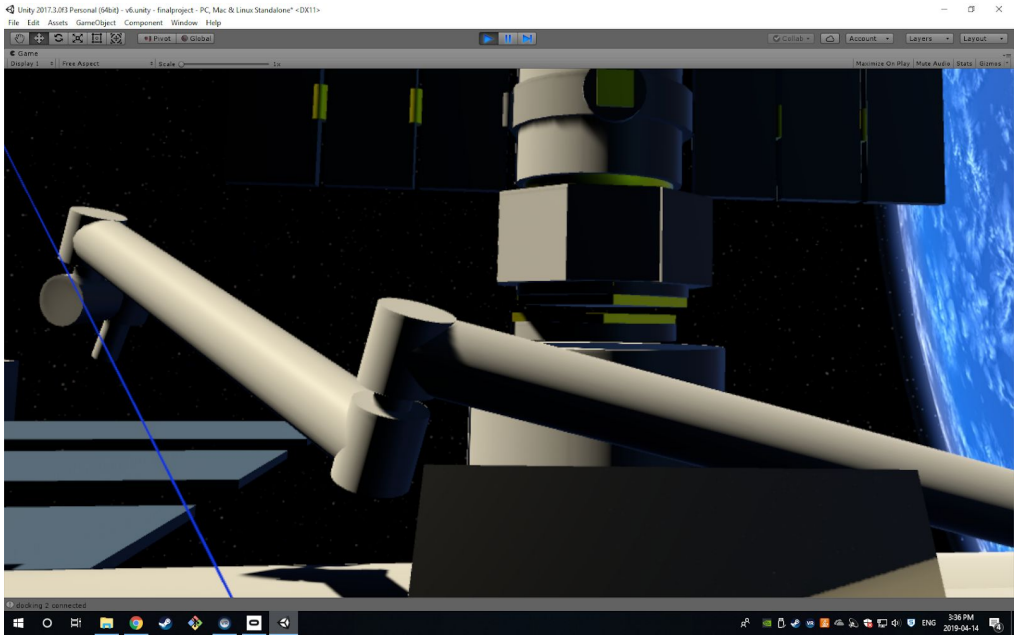
The A button engages “direct kinematics” which disengages the inverse kinematics and arm tracking to move each joint on the arm separately with joystick movements. Pressing the grip again will set the arm to inverse kinematics and begin following the players arm. Once the arm makes contact with the module it will automatically attached and ready to place to the next goal.



Move the arm to connect the module to the space station. Once the module is on the space station use the trigger to board the space station and control the arm to place the module onto the correct goal.



Second level (Space Station)



Goal met

Triggering the red button on the space station will reset the game.

Project Implementation

Three main scripts used are:

-RobotArm2
-VRIKController
-ballController

RobotArm2

This script is an implementation of an algorithm as described here https://www.alanzucconi.com/2017/04/10/robotic-arms/

The data structure used in this script “robotJoint” consists on array which holds the values of angles and the axis of rotation. This data structure is given to RobotArm2 script as an array of robotJoints. The algorithm consists of three method calls, Direct Kinematics, Inverse Kinematics, Partial Gradient. Direct Kinematics returns the position of the end effector and is used in other method calls. Partial Gradient is an important method for it performs the iterative solution of inverse Kinematics. Inverse Kinematics executes per frame to adjust the angles of each joint in the data structure which will terminate once target position specified has been reached. The target position in this case is set to a game object called goal controller. Once the arm iteratively moves its end effector close to the target position the algorithm stops. This script procedurally animates the arms kinematics by iteratively approximating the angles for the joints to rotate. The angles drive the joints local euler angles. Gimbal lock can occur in the arm. To exit Gimbal lock in game, raising the target position to extend the arm can restore rotation in the lost degrees of freedom.

ballController

This script and gameobject interfaces with robotArm2 to specify the target for the robotic arms inverse kinematics. The script references two ball game objects, one where the player controls(goalController) and the other where the robotic arm follows(target). The player ball is parented to the players controller to map the robotic arm to the the players arm. As the player ball moves, the target ball moves relative to players ball by a certain scale. This achieves interactive control for the arm, rather than using levers and other typical control systems.

VRIKController

This script interfaces the simulator with the player in a VR setting. The right controller emits a laser which can be used to interact with game objects in the scene. The script manages which arm and which goal controller is used based on which level the player is in. If the controller is gripped, it will map the correct robot arm to the players controller. Pressing A will change controls from inverse kinematics to direct kinematics. To also note this script also contains methods to teleport the player to different areas of the virtual environment. Implementing fading techniques to alleviate vection.

Other scripts contain collider triggers and procedural animations to open doors and panels.

Stagit Asset

The skybox implemented in the scene to encourage immersion in a space environment. All other assets and scripts have been implemented on my own.

Conclusion

The algorithm implemented has opportunities to be improved.

The current algorithm can Gimbal lock. This could be prevented by implementing comfort functions, min max angles, or using quaternions to rotate the joints. Although movement through interaction is intuitive and simple, it does not provide accurate controls. By implementing a combination of direct kinematics and inverse kinematics, the robotic arm can move more specifically. Selecting a combination of which joints to freeze and which joints to follow inverse kinematics can achieve controls similar to the actual CanadaArm2. Rather than an iterative solution, the arm can be procedurally animated with an exact solution which could have more robust controls. Although this solution would be unique to this robotic arm, unlike the algorithm which works for many robotic arms.

The ball controller can be furthered improved. Rather than mapping the ball from the right controller, it should be mapped to the camera rig. This will guarantee that the goal controller will always be initialized from the shoulder, rather than placing the

controller near it. This will also compensate for torso rotations which will help with tracking.

There are two arms in the scene, with more time it was desired to include net play. A possible game mode would be that rather than resetting the game, the space station will instantiate more modules to keep building up the space station. If rigid bodies were implemented it would be an additional game feature to not collide with other objects. The purpose of this project was to demonstrate direct and inverse kinematic procedural animations. The project has illuminated the speed and simplicity of iterative solutions and how to improve them.