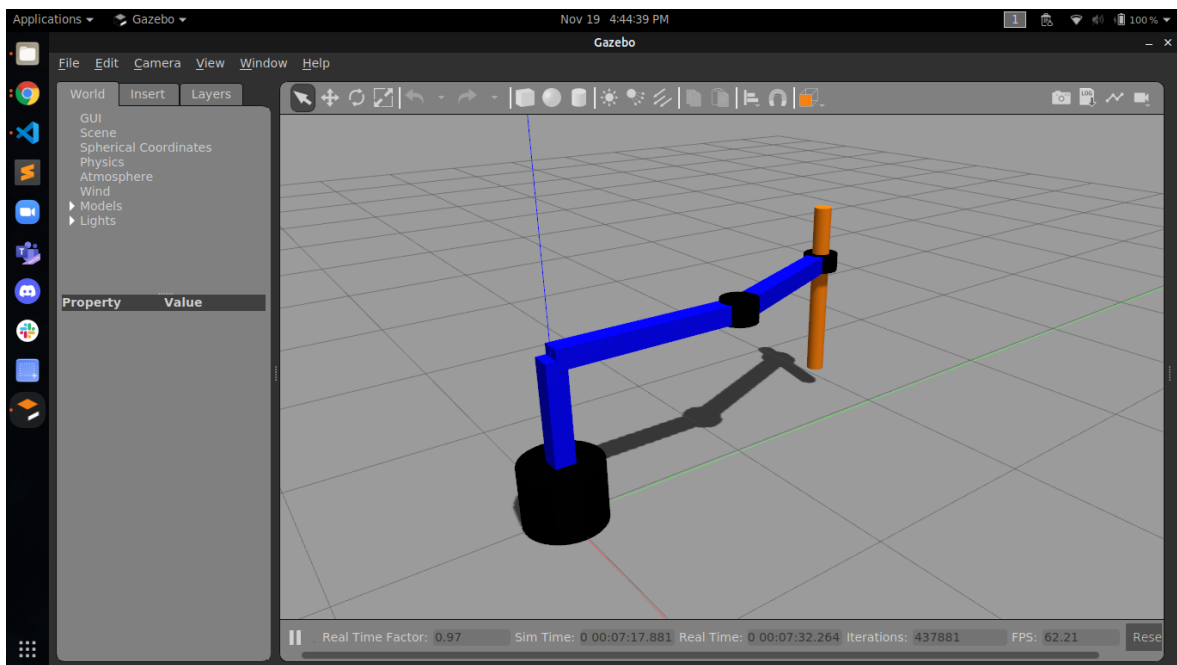
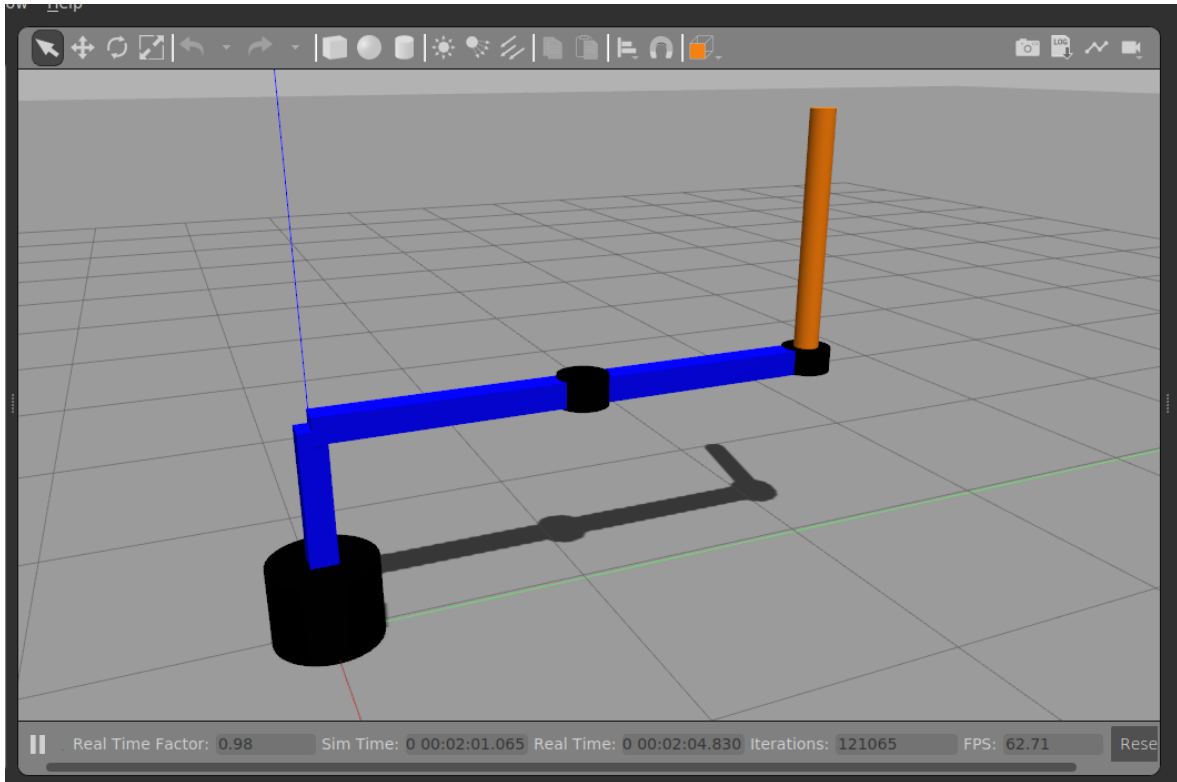


Group Assignment 1

Team Members: Aniket Patil, Venkatesh Mullur, Febin Fredi

Part 1. Create and Spawn Robot in Gazebo Environment



Robot has a base that is fixed to the world frame using a joint called joint0. A revolute joint1 is on top of this base which rotates the robot about its Z-axis. Then link1 and link1_h forms the L-shaped arm of the robot. Revolute joint2 connects link1_h and link2 and prismatic joint3 connects link2 and end effector arm link3.

Part 2.

The forward kinematics node subscribes to the topic `/scara_robot/joint_states` through which it receives the joint values of robot from gazebo. Then these three joint values are used to calculate the end effector pose using forward kinematics and then is published on the topic `/scara_robot/output_pose`.

The image shows two terminal windows side-by-side. The left window is titled 'aniket@aniket-HP: ~/RBE500_ws 58x8' and the right window is titled 'aniket@aniket-HP: ~/RBE500_ws 56x24'.

Left Terminal:

```
aniket@aniket-HP:~/RBE500_ws$ source devel/setup.bash
aniket@aniket-HP:~/RBE500_ws$ rosrunc scara_robot scara_for
ward
[ INFO] [1637361683.229729130]: =====
== FORWARD KINEMATICS =====
[ INFO] [1637361683.229907588]: Subscribe to the topic: /sc
ara_robot/output_pose
^C
aniket@aniket-HP:~/RBE500_ws 58x14
aniket@aniket-HP:~/RBE500_ws$ rostopic pub /scara_robot/jo
int1_position_controller/command std_msgs/Float64 "data: 0
.78"
publishing and latching message. Press ctrl-C to terminate
aniket@aniket-HP:~/RBE500_ws$ ^C
aniket@aniket-HP:~/RBE500_ws$
aniket@aniket-HP:~/RBE500_ws$
aniket@aniket-HP:~/RBE500_ws$
aniket@aniket-HP:~/RBE500_ws$ rostopic pub /scara_robot/jo
int2_position_controller/command std_msgs/Float64 "data: 0
.78"
publishing and latching message. Press ctrl-C to terminate
^C
aniket@aniket-HP:~/RBE500_ws$
```

Right Terminal:

```

z: 0.0
w: 1.0
---
position:
  x: 0.7218828814855218
  y: 1.7031839999650153
  z: 0.7999026076673846
orientation:
  x: 0.0
  y: 0.0
  z: 0.0
  w: 1.0
---
position:
  x: 0.7218828814855218
  y: 1.7031839999650153
  z: 0.7999026076673846
orientation:
  x: 0.0
  y: 0.0
  z: 0.0
  w: 1.0
---
```

Left terminal: Published 0.78 radians to joint1 and joint2

Right terminal: Pose of the end effector after joint1 and joint2 were rotated by 0.78 radians

Part 3.

The server file takes the pose values using `geometry_msgs/Pose` message type. Then server takes the position x, y and z values from `Pose.position` object and calculates the three joint variables joint1, joint2 and joint3 using inverse kinematics. Then the server file prints the joint state response using `sensor_msgs/JointState` message type.

```
aniket@aniket-HP: ~/RBE500_ws
aniket@aniket-HP: ~/RBE500_ws 85x28
aniket@aniket-HP:~/RBE500_ws$ rosservice call /inv "pose:
  position:
    x: 0.72
    y: 1.703
    z: 0.8
  orientation:
    x: 0.0
    y: 0.0
    z: 0.0
    w: 0.0"
joints:
  header:
    seq: 0
    stamp:
      secs: 0
      nsecs: 0
    frame_id: ''
  name:
    - joint1
    - joint2
    - joint3
  position: [0.7796620866597178, 0.7822852695677189, 0.0]
  velocity: []
  effort: []
aniket@aniket-HP:~/RBE500_ws$
```

This terminal shows that when you give the Pose from the forward kinematics node to the service call, the output gives you a Joint State object which shows the values of all 3 joints. We see that these are the same values that we had given to the joints in Part 2.

Result:

The values given to Gazebo to move the robot were given to forward kinematics node. Then we took the values for the End effector Pose from the forward kinematics node and sent it to the inverse kinematics server as a request. The values returned match the initial value.