Lab 3 Outline

If a coordinate or orientation value is not listed – it is assumed to be 0 or no changes are needed

Introduction

This lab will teach the user to attach and manipulate joints associated with primitive objects to build a robot with simple functionality. It will show the usage of revolute & prismatic joints, as well as show how to trigger joints and other functions with proximity sensors. In addition, the physics engine is shown off as the main purpose of the lab is to create a physics collision in the example of a soccer ball kick. The user will also learn to tinker with UI elements, that can also be triggered and updated by a proximity sensor.

Step 1. Building the Environment

Goals:

Apply positions and orientation of environment components

Add a texture to a sphere

- *Goal Posts*
- Components -> Sensors -> Passage counter
- Make it wider -> GateFrame -> View/modify geometry
- **X -> 3.0**
- **Z -> .75**
- **Position:**
- **X ++1.7500e-01**
- **Y+1.5250e+00**
- **Z+1.7500e-01**
- *Net*
- equipment -> pane grid 1.0 x 2.0
- View/modify geometry -> **.8**
- **Position:**
- **X +1.7500e-01**
- **Y+1.8000e+00**
- **Z+2.5000e-02**
- **Orientation**
- **A: -9.0000e+01**
- **B: -9.0000e+01**
- **G -4.0000e+01**
-
- *Soccer Ball*
- ***1.5000e-01***

- Add Texture
- Max Res
- **Along U .5**
- **Along V .5**

Step 2. Building Joints to shoot the soccer ball

Goals:

Build a shooting robot out of compiled joints and primitive shapes

Code the robot to choose different shots randomly

- *Soccer Ball Pos*
    - **Coordinates**
    - **X +2.0000e-01**
    - **Y -7.7500e-01**
    - **Z +7.5000e-02**
- *Create Base Cuboid*
    - **Geometry**
    - **X 1.0000e-01**
    - **Y 6.0000e-01**
    - **Z 2.0000e-02**
    - **Coordinates**
    - **X +2.0000e-01**
    - **Y -1.2250e+00**
    - **Z +1.0000e-02**
    - **Orientation**
    - **G +9.0000e+01**
- Attach Straight Shot Joint
    - Create Prismatic Joint
        - Change Joint Name
        - **Coordinates**
        - **X +2.0000e-01**
        - **Y -1.2250e+00**
        - **Z +8.5000e-02**
        - **Orientation**
        - **A: -9.0000e+01**
-
        - Move Prismatic Joint to be a child of Base cuboid
        - Enable Motor and Control loop in dynamic properties dialog
            - **Increase Maximum Torque to 2.5000e+04**
        - Reduce length to .05m
- Attach Straight Shot Cuboid
    - **Geometry**
    - **Y 2.0000e-02**
    - Move Cuboid to be a child of Straight Shot Joint
    - Apply Joint Position to Cuboid
    - Move cuboid along the y axis

- Add Threaded child script associated with the base cuboid
    - Create shootStraight function
-
    - function shootStraight()
        - straightJoint = sim.getObjectHandle('Prismatic_joint_straight')
        - sim.setJointTargetPosition(straightJoint, .5)
        - sim.wait(2)
        - sim.setJointTargetPosition(straightJoint, 0)
    - end
- Add Revolute Joint to set Shoot right rotation
    - Change Joint Name
        - **Coordinates**
        - **X +1.0000e-01**
        - **Y -1.2250e+00**
        - **Z -1.2250e+00**
        - Move Revolute Joint to be a child of Base cuboid
        - Enable Motor and Control loop in dynamic properties dialog
            - **Increase Maximum Torque to 2.5000e+04**
        - Reduce length to .05m
- Add Cuboid to Revolute joint
    - Copy & paste Cuboid from straight shot
    - Make the Cuboid a child of Revolute joint right
        - Apply position to Revolute joint right
        - Move cuboid along the y axis
- Add Prismatic joint to shoot right
    - Copy & paste straight prismatic joint
    - Make the prismatic joint a child of the cuboid
        - Rename joint
        - Apply position to Cuboid of Revolute joint right
        - Move the joint along the y axis
- Add Cuboid to the Prismatic joint to shoot right
    - Copy & paste cuboid from the revolute joint
    - Move it along the Y axis
- Create shoot Right function
    - function shootRight()
        - rightRotationJoint = sim.getObjectHandle('Revolute_joint_right')
        - rightShotJoint = sim.getObjectHandle('Prismatic_joint_right')
        - sim.setJointTargetPosition(straightShotJoint, -.1)
        - sim.setJointTargetPosition(rightRotationJoint, -15*math.pi/180)
        - sim.wait(2)
        - sim.setJointTargetPosition(rightShotJoint, .4)
        - sim.wait(2)
        - sim.setJointTargetPosition(rightShotJoint, 0)
        - sim.setJointTargetPosition(rightRotationJoint, 0)

```
-   End
-   Create left shot joints & cuboids
-       Copy & paste all right shot items
-       Move along X axis
-   Create shoot left function
-   function shootLeft()
-     leftRotationJoint = sim.getObjectHandle('Revolute_joint_left')
-     leftShotJoint = sim.getObjectHandle('Prismatic_joint_left')
-     sim.setJointTargetPosition(straightShotJoint, -.1)
-     sim.setJointTargetPosition(leftRotationJoint, 15*math.pi/180)
-     sim.wait(2)
-     sim.setJointTargetPosition(leftShotJoint, .35)
-     sim.wait(2)
-     sim.setJointTargetPosition(leftShotJoint, 0)
-     sim.setJointTargetPosition(leftRotationJoint, 0)
-   End
-   Randomize shooting direction & reset ball position
-   function sysCall_threadmain()
-
-     ball = sim.getObjectHandle('Sphere')
-     ballPos = sim.getObjectPosition(ball, -1)
-     ballOri = sim.getObjectOrientation(ball, -1)
-
-     while (true) do
-       randomNumber=sim.getRandom()
-         if (randomNumber < 0.33) then
-           shootLeft()
-         elseif (randomNumber >=0.33 and randomNumber <0.67) then
-           shootStraight()
-         else
-           shootRight()
-         end
-       sim.wait(2)
-       sim.setObjectPosition(ball, -1, ballPos)
-       sim.setObjectOrientation(ball, -1, ballOri)
-     end
-
-   end
-
```

Step 3. Building Goalie with Proximity sensors

Goals:

<u>Create Proximity sensors and trigger a response from a prismatic joint</u>

- *Goalie Prismatic Joint*
- **Coordinates**
- **X -6.5000e-01**
- **Y +1.4750e+00**
- **Z +1.7500e-01**
- *Attach Cuboid to Prismatic Joint*
- **Cuboid Geometry**
- **X 3.2400e-01**
- **Y 4.5000e-01**
- **Z 3.7500e-02**
- *Script positions for the prismatic joint to set to*
- goalie = sim.getObjectHandle(‘Prismatic_joint_goalie’)
- sim.setJointTargetPosition(goalie, -.3)
- sim.wait(2)
- sim.setJointTargetPosition(goalie, -.8)
- sim.wait(2)
- sim.setJointTargetPosition(goalie, -1.35)
- *Add Proximity Sensors.*
- Rotate 90 degrees about the X axis
- Move into the goal on the left side
- Increase the Range -> Show volume parameters -> increase range
- Entity to detect -> Sphere
- Copy & paste proximity sensor, & drag the new sensor to the right
- Name each proximity sensor based on their position
- *Ensure Sphere has detectable box checked in -> common*
- *Trigger prismatic joint positions based on proximity sensor detections*

```
- function sysCall_threadmain()
-
-     goalie = sim.getObjectHandle('Prismatic_joint_goalie')
-     sim.setJointTargetPosition(goalie, -.8)
-
-     ball = sim.getObjectHandle('Sphere')
-
-     rightProximitySensor = sim.getObjectHandle('Proximity_sensor_right')
-     leftProximitySensor = sim.getObjectHandle('Proximity_sensor_left')
-
-      while (true) do
```

```
-
-           right = sim.checkProximitySensor(rightProximitySensor, ball)
-           left = sim.checkProximitySensor(leftProximitySensor, ball)
-
-           if(left == 1) then
-               sim.setJointTargetPosition(goalie, -.3)
-               sim.wait(4)
-               sim.setJointTargetPosition(goalie, -.8)
-           elseif(right == 1) then
-
-               sim.setJointTargetPosition(goalie, -1.35)
-               sim.wait(4)
-               sim.setJointTargetPosition(goalie, -.8)
-           else
-               sim.setJointTargetPosition(goalie, -.8)
-           end
-       end
-
-   End
```

Step 4. Creating UI element to count Goals

<u>Goals:</u>

<u>Create a UI element, & update the element with a proximity sensor trigger</u>

<u>Ensure logic is sound when counting goals</u>

- *Create UI element*
- ui=simUI.create([[<ui>
-         <label id="100" text="Goal Count:" />
-         <label id="101" text="0" style="font-size: 30px" />
-      </ui>]])
-   count=0
- *Create Goal Proximity Sensor*
-     Copy & paste one of the proximity sensors from our previous module
-     Rotate about the X axis 90 degrees
-     Position the proximity sensor behind the goal posts
-     Reduce the range of the proximity sensor to 1.5m
- *Create logic to increase the goal count on the UI element*
- goalProximitySensor = sim.getObjectHandle('Proximity_sensor_goal')
-    while (true) do
-
-      right = sim.checkProximitySensor(rightProximitySensor, ball)
-      left = sim.checkProximitySensor(leftProximitySensor, ball)
-      goal = sim.checkProximitySensor(goalProximitySensor, ball)
-      goalCheck = true
-
-      if(goal == 1 and goalCheck == true) then
-        count = count + 1
-        goal = 0
-        goalCheck = false
-        simUI.setLabelText(ui,101,''..count)
-        print("GOAL!")
-        sim.wait(4)
-      end