

Build intelligent robot that can
interact with human

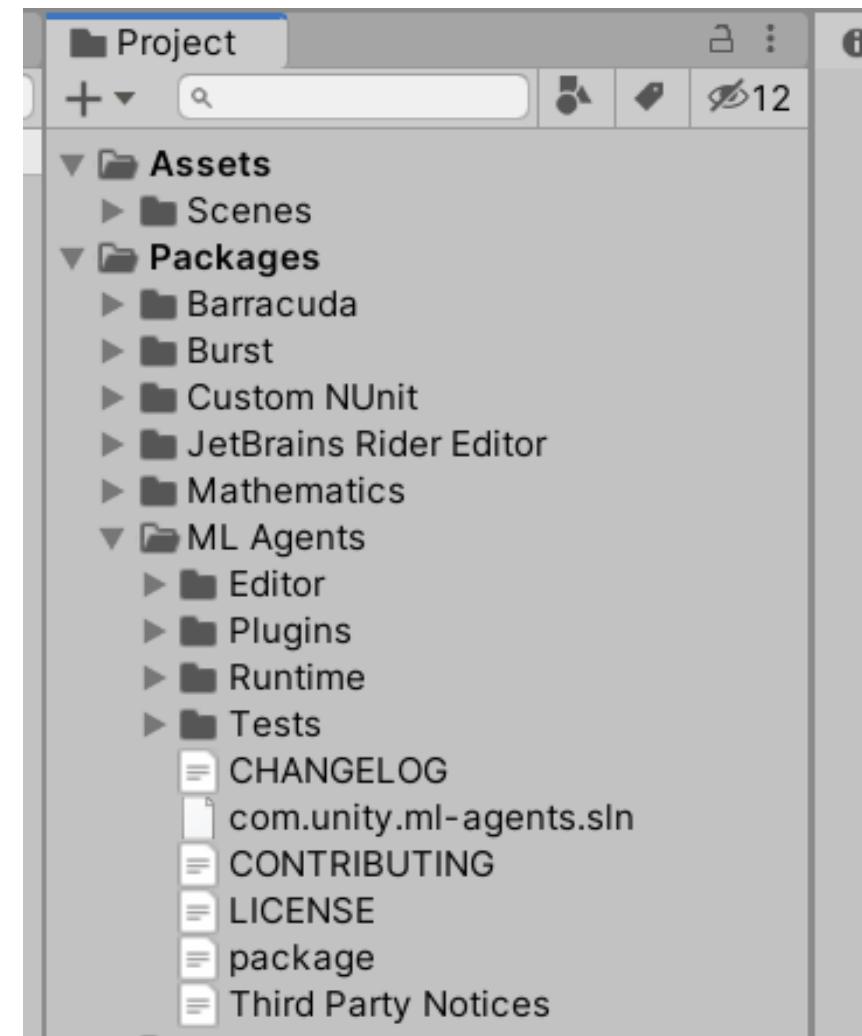
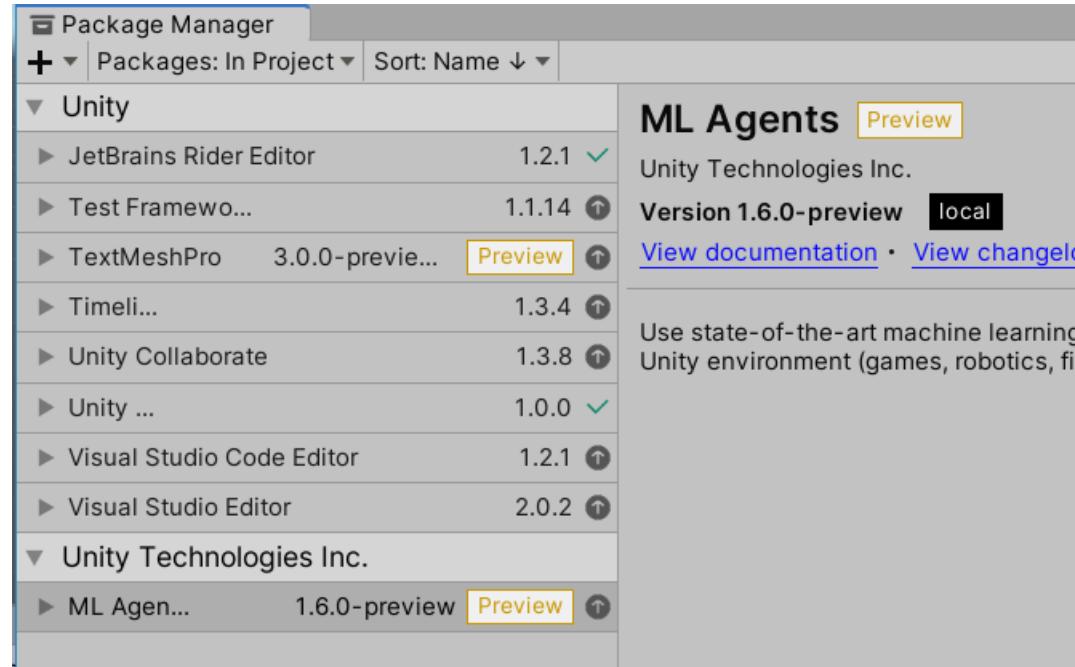
1. Download and save ML Agent to C:\

This will take a few minutes.
and then you can run the command below to train your agent.

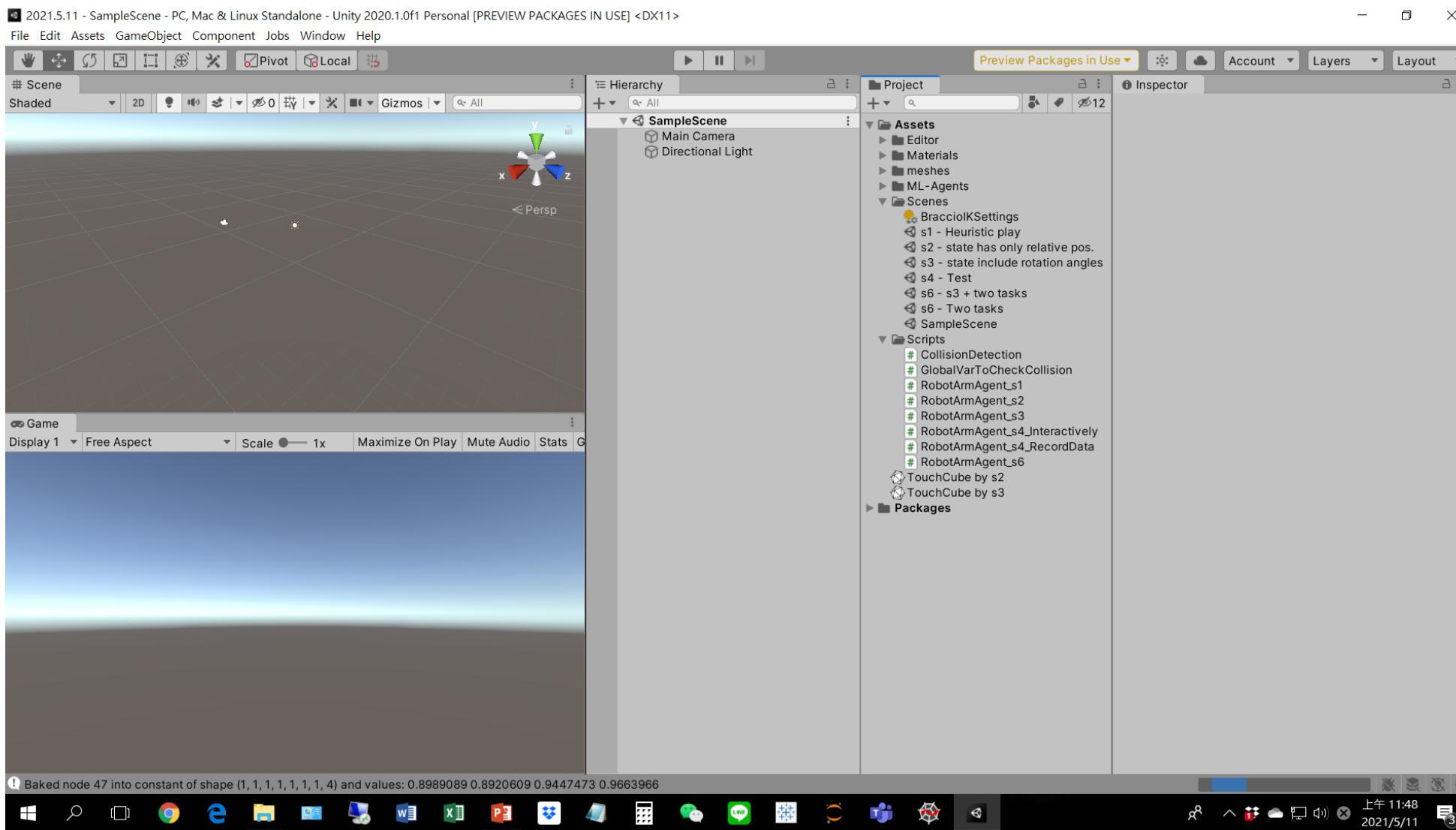
```
cd C:\ml-agents-release_10\config\ppo\results  
tensorboard --logdir=1
```



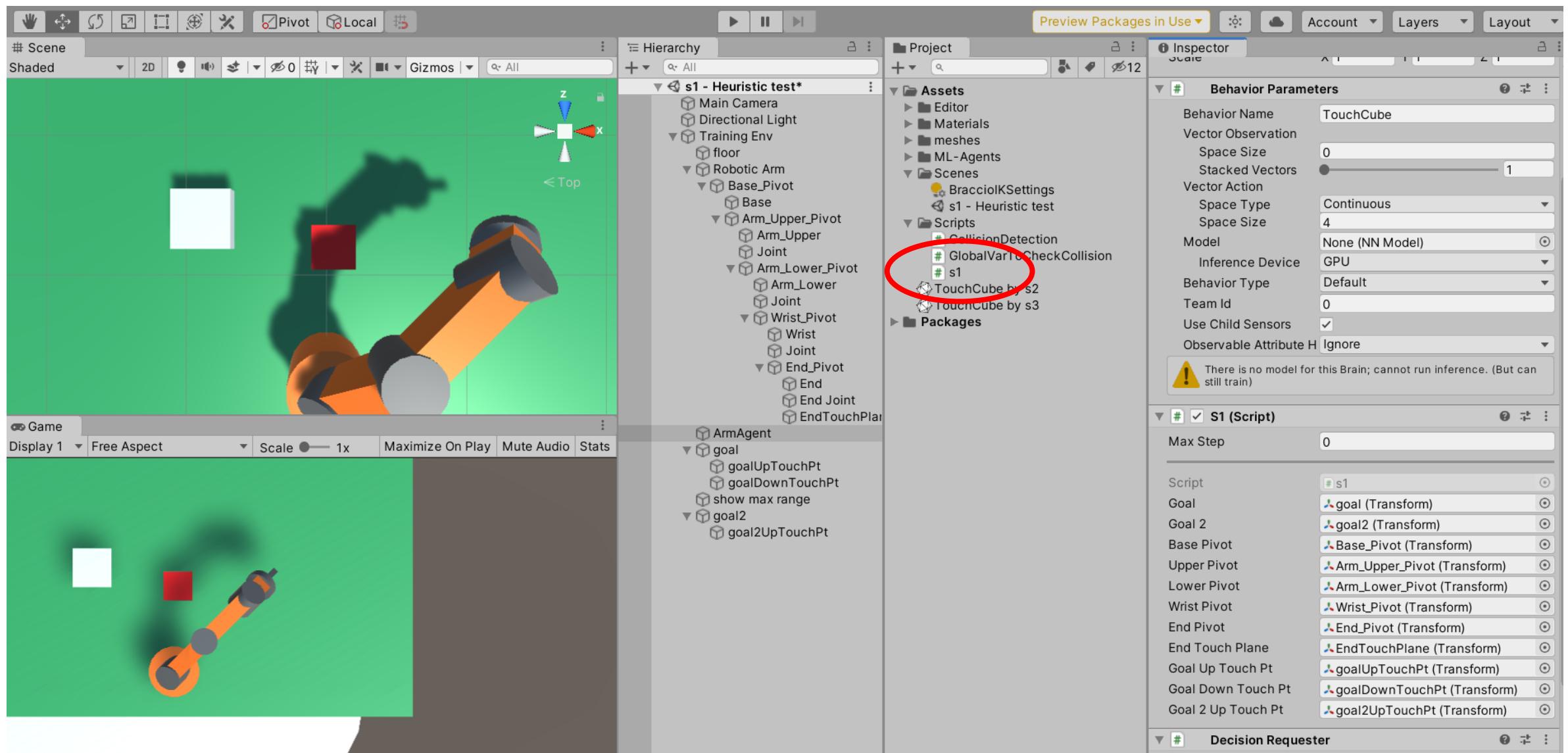
2. Create a new Unity project and import ML Agent package to this new project



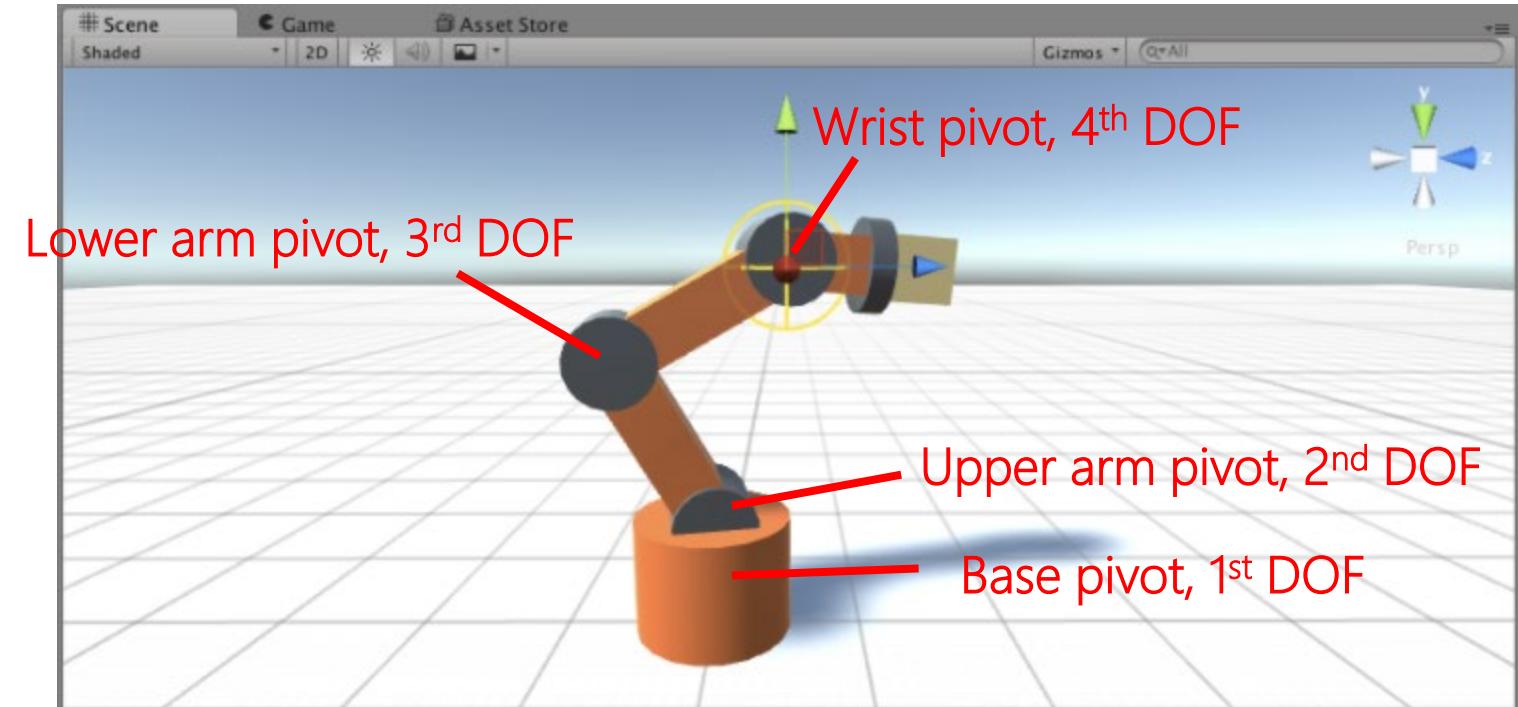
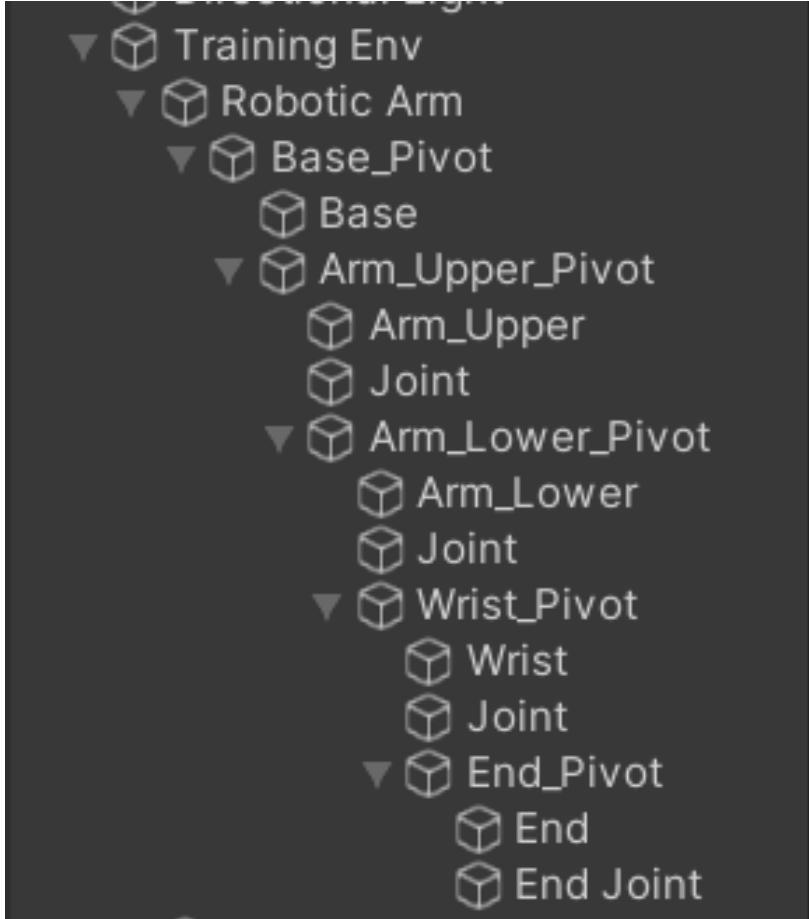
3. Import Robot arm package to Unity project



Open scene "s1 - Heuristic play"



This Unity project contains a Braccio robot arm

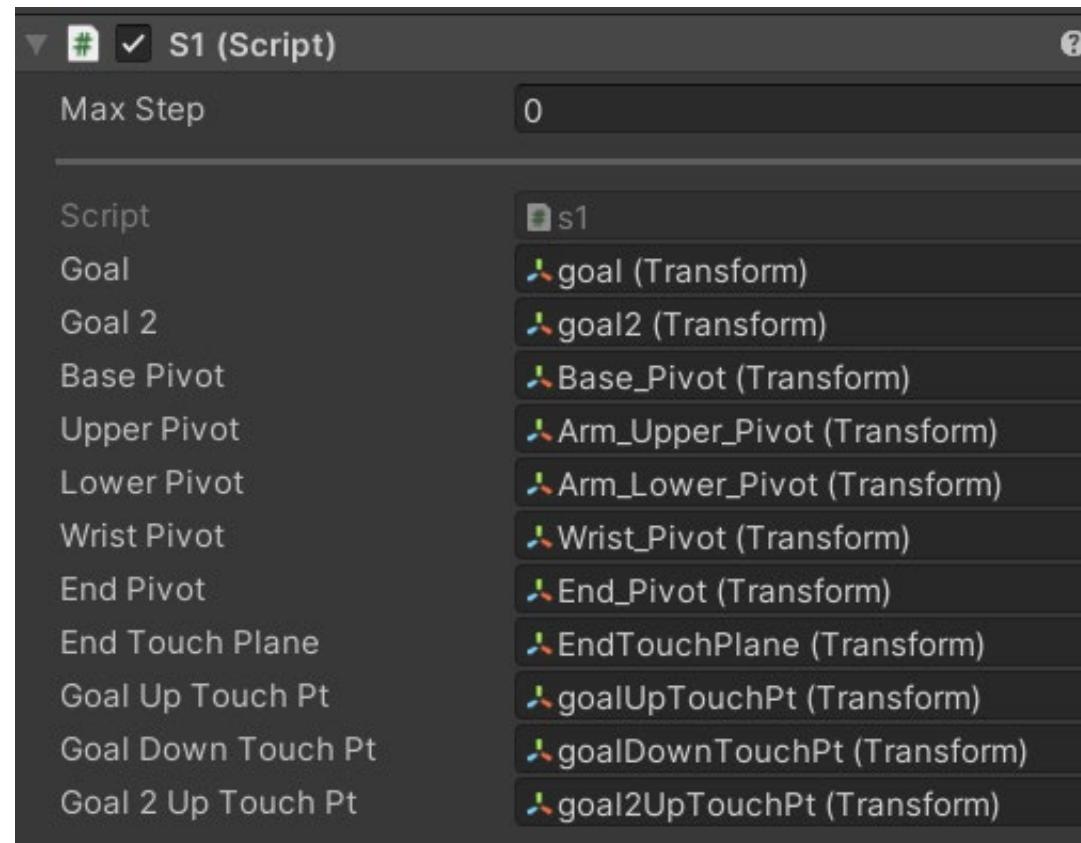


<https://github.com/tanyuan/braccio-ik-unity>

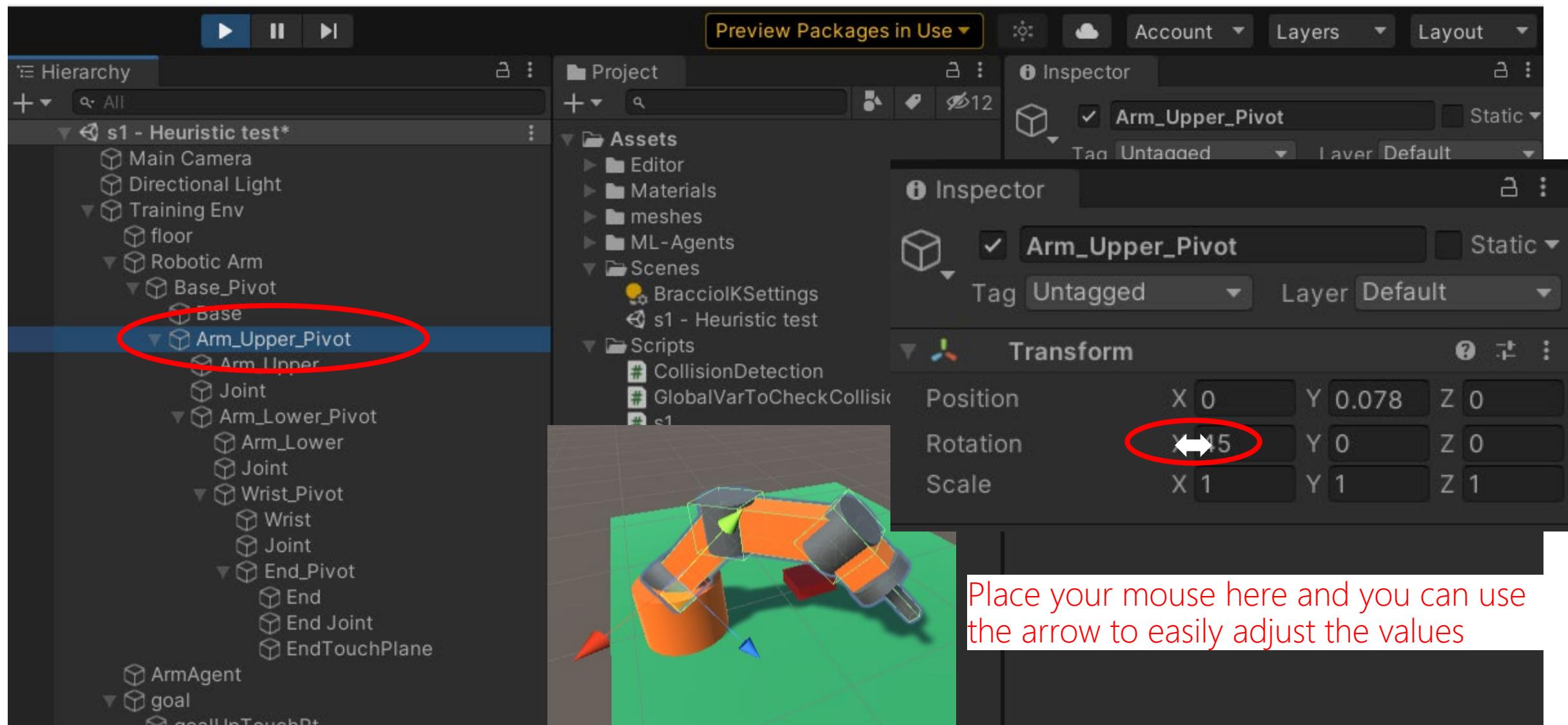
(1) Manually operates the robot arm
in VE

Public variables to link agent script with scene objects

```
public Transform goal, goal2;  
public Transform BasePivot, UpperPivot, LowerPivot, WristPivot, EndPivot;  
public Transform EndTouchPlane, goalUpTouchPt, goalDownTouchPt, goal2UpTouchPt;  
int stage = 1;
```



Play, rotate arm by changing rotation angle in Inspector window



Use Up/Down, L/R keys to rotate arm

← and → key

↑ and ↓ key

```
actionsOut[0] = Input.GetAxis("Horizontal");
actionsOut[1] = Input.GetAxis("Vertical");
```

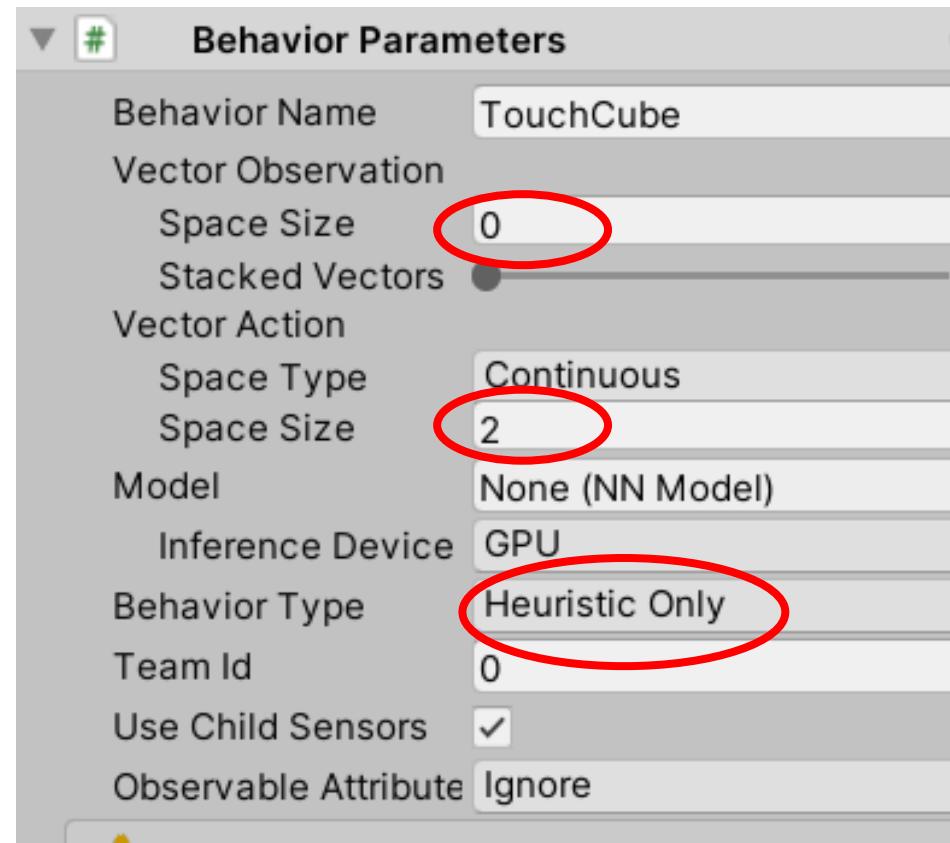
Base -90 ~ 90

U/L arm and wrist : 0 ~ 90

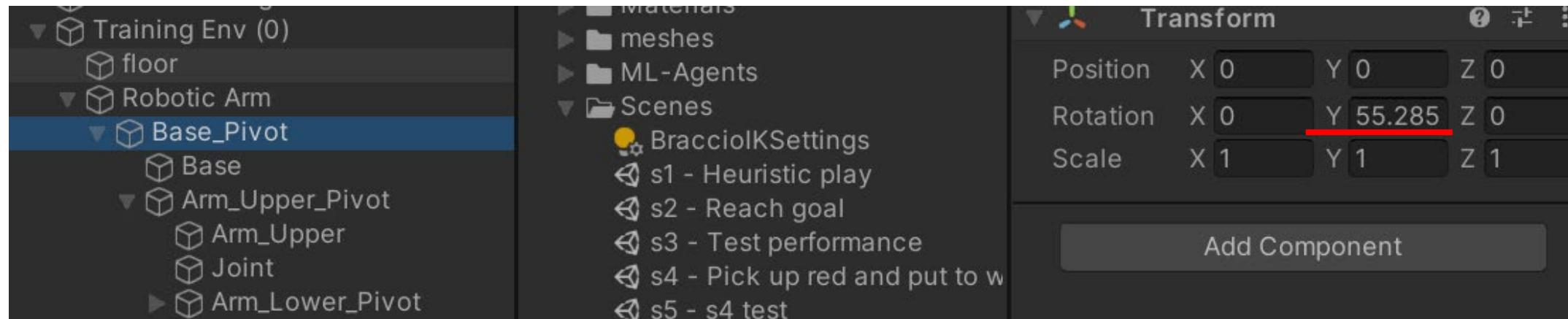
```
BasePivot.Rotate(0, vectorAction[0] * speed, 0);
float RotationAngle = UnityEditor.TransformUtils.GetInspectorRotation(BasePivot).y;
```

```
UpperPivot.Rotate(vectorAction[1] * speed, 0, 0);
```

```
//float RotationAngle = UnityEditor.TransformUtils.GetInspectorRotation(UpperPivot).x;
```



Check whether arm rotation is out of range



```
float BaseRotationAngle = UnityEditor.TransformUtils.GetInspectorRotation(BasePivot).y;
float UArmRotationAngle = UnityEditor.TransformUtils.GetInspectorRotation(UpperPivot).x;
float LArmRotationAngle = UnityEditor.TransformUtils.GetInspectorRotation(LowerPivot).x;
float WRotationAngle = UnityEditor.TransformUtils.GetInspectorRotation(WristPivot).x;
if ((BaseRotationAngle >= -90 && BaseRotationAngle <= 90) &&
    (UArmRotationAngle >= 0 && UArmRotationAngle <= 90) &&
    (LArmRotationAngle >= 0 && LArmRotationAngle <= 90) &&
    (WRotationAngle >= 0 && WRotationAngle <= 90))
{
    return true;
}
```

Collision detections

Static global variables to record collision of lower arm, wrist, end, and goal

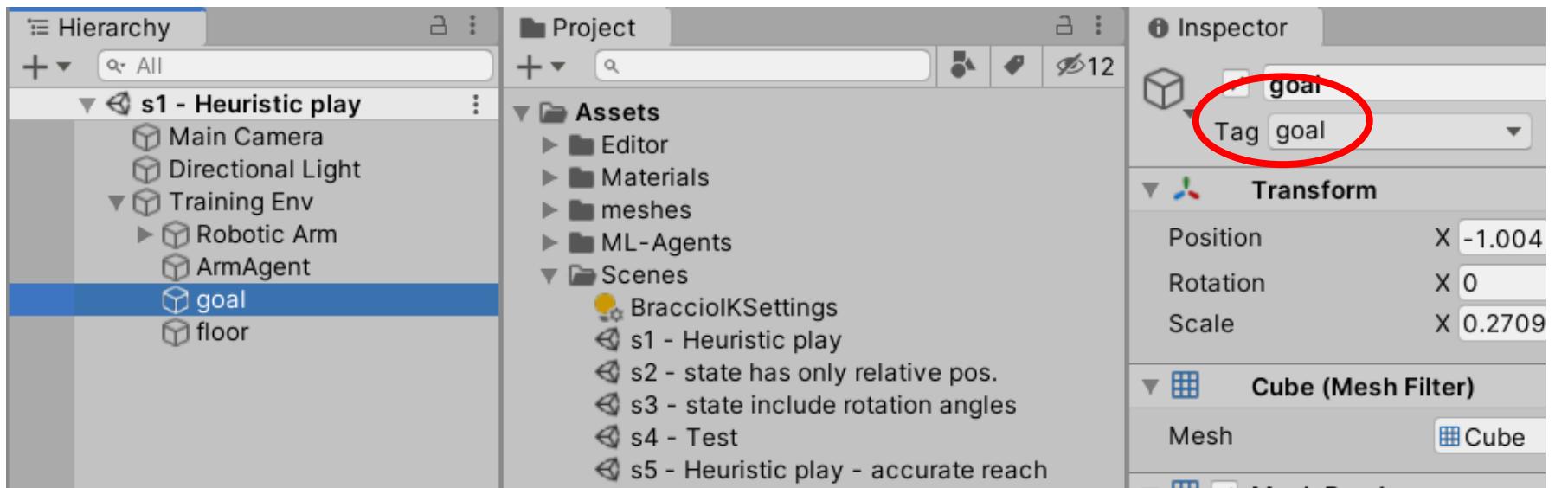
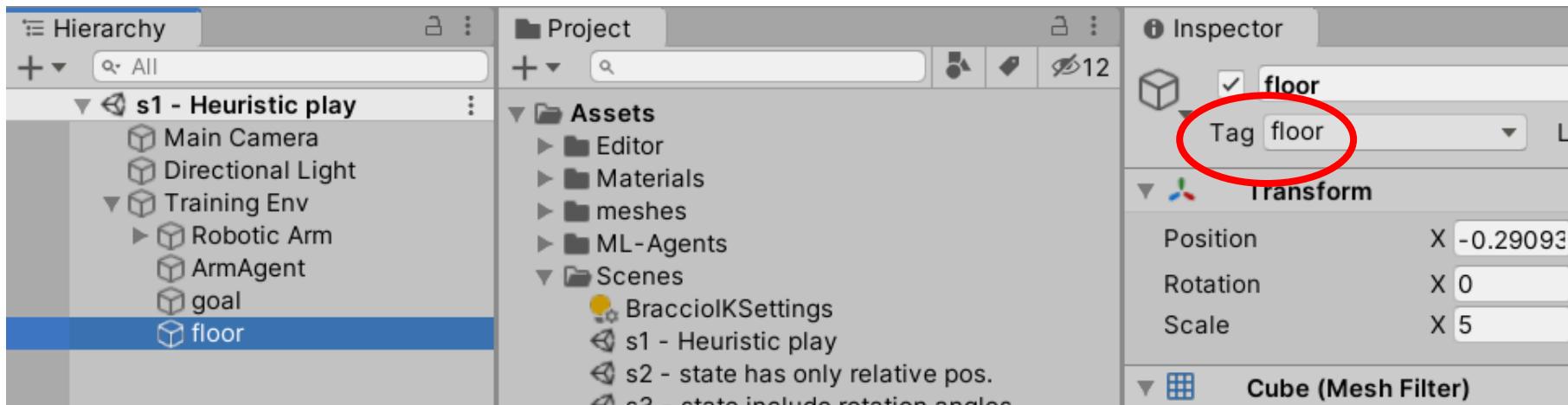
```
public class MyGlobalVar : MonoBehaviour
{
    public static bool LowerArmCollisionHappens = false;
    public static bool WristCollisionHappens = false;
    public static bool EndCollisionHappens = false;
    public static bool goalCollisionHappens = false;
```

On trigger enter/exist to record collisions of lower arm, wrist,
and end with floor and goal objects

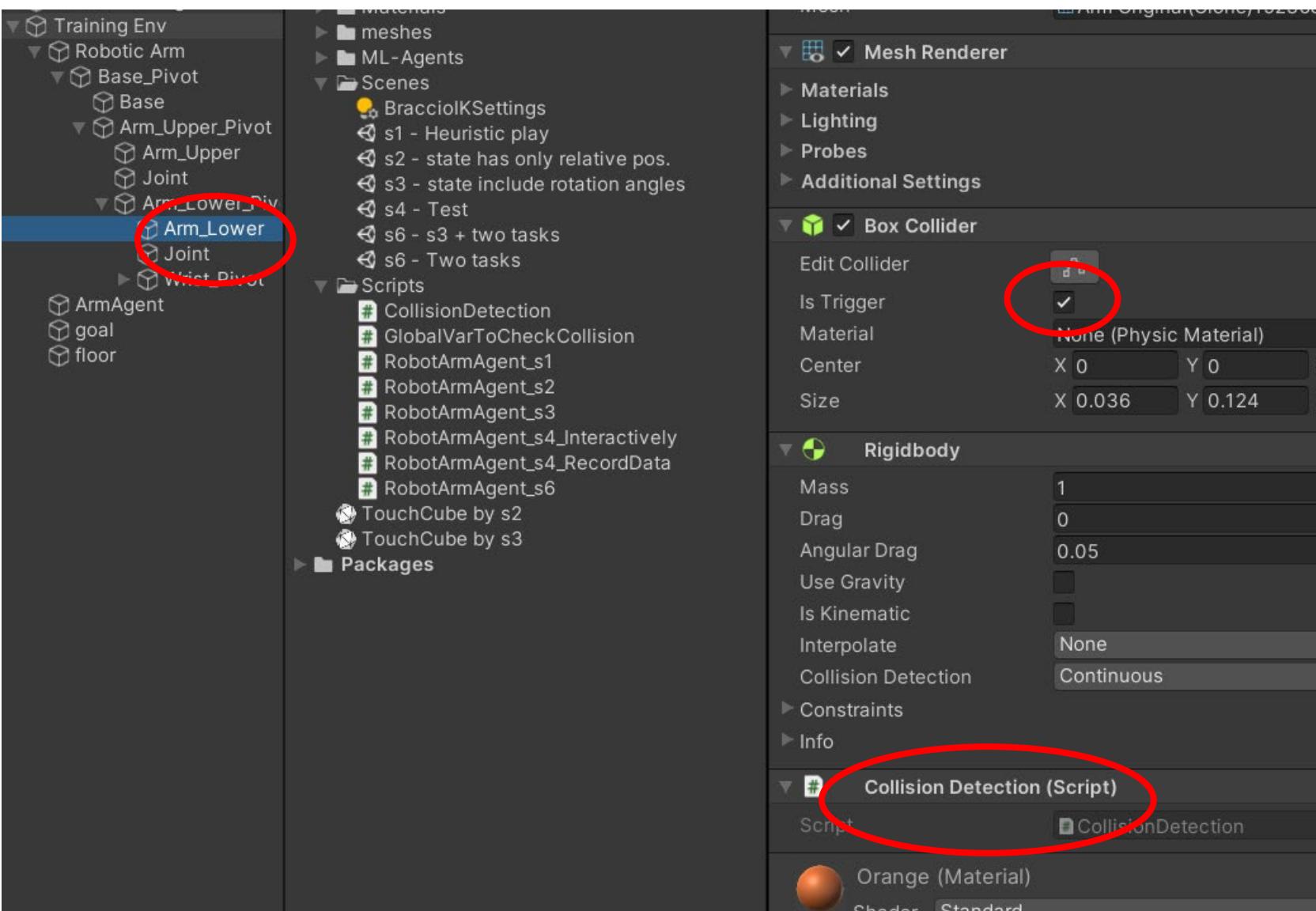
```
public class CollisionDetection : MonoBehaviour
{
    void OnTriggerEnter(Collider other)
    {
        if (other.gameObject.tag == "floor" || other.gameObject
        {
            if(this.gameObject.tag == "Lower arm")
                MyGlobalVar.LowerArmCollisionHappens = true;
            else if(this.gameObject.tag == "Wrist")
                MyGlobalVar.WristCollisionHappens = true;
            else if(this.gameObject.tag == "End")
                MyGlobalVar.EndCollisionHappens = true;
            else if(this.gameObject.tag == "goal")
                MyGlobalVar.GoalCollisionHappens = true;
        }
    }

    void OnTriggerExit(Collider other)
    {
        if (other.gameObject.tag == "floor")
        {
            if (this.gameObject.tag == "Lower arm")
                MyGlobalVar.LowerArmCollisionHappens = false;
            else if (this.gameObject.tag == "Wrist")
                MyGlobalVar.WristCollisionHappens = false;
            else if (this.gameObject.tag == "End")
                MyGlobalVar.EndCollisionHappens = false;
            else if (this.gameObject.tag == "goal")
                MyGlobalVar.GoalCollisionHappens = false;
        }
    }
}
```

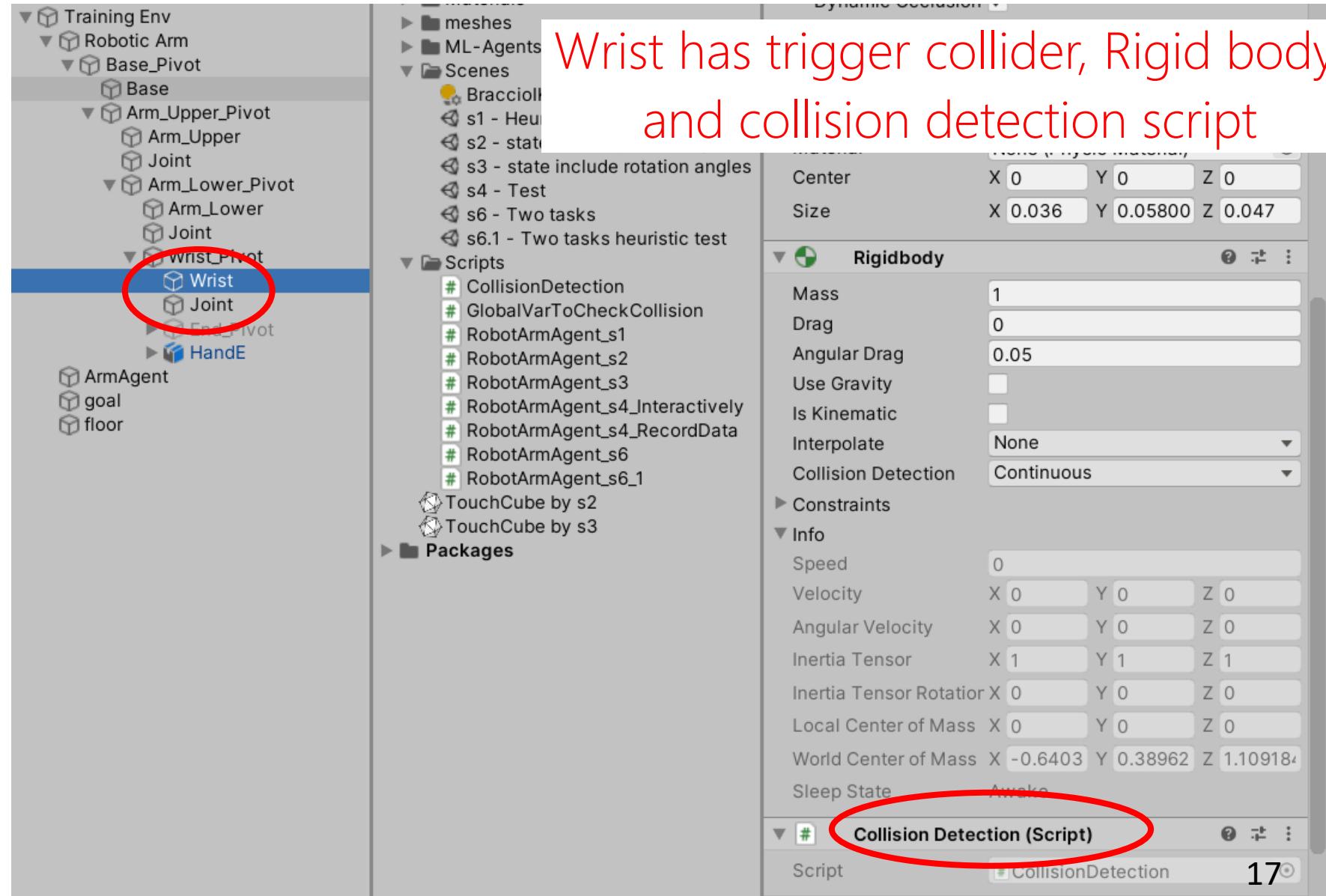
Add tags to floor and goal object



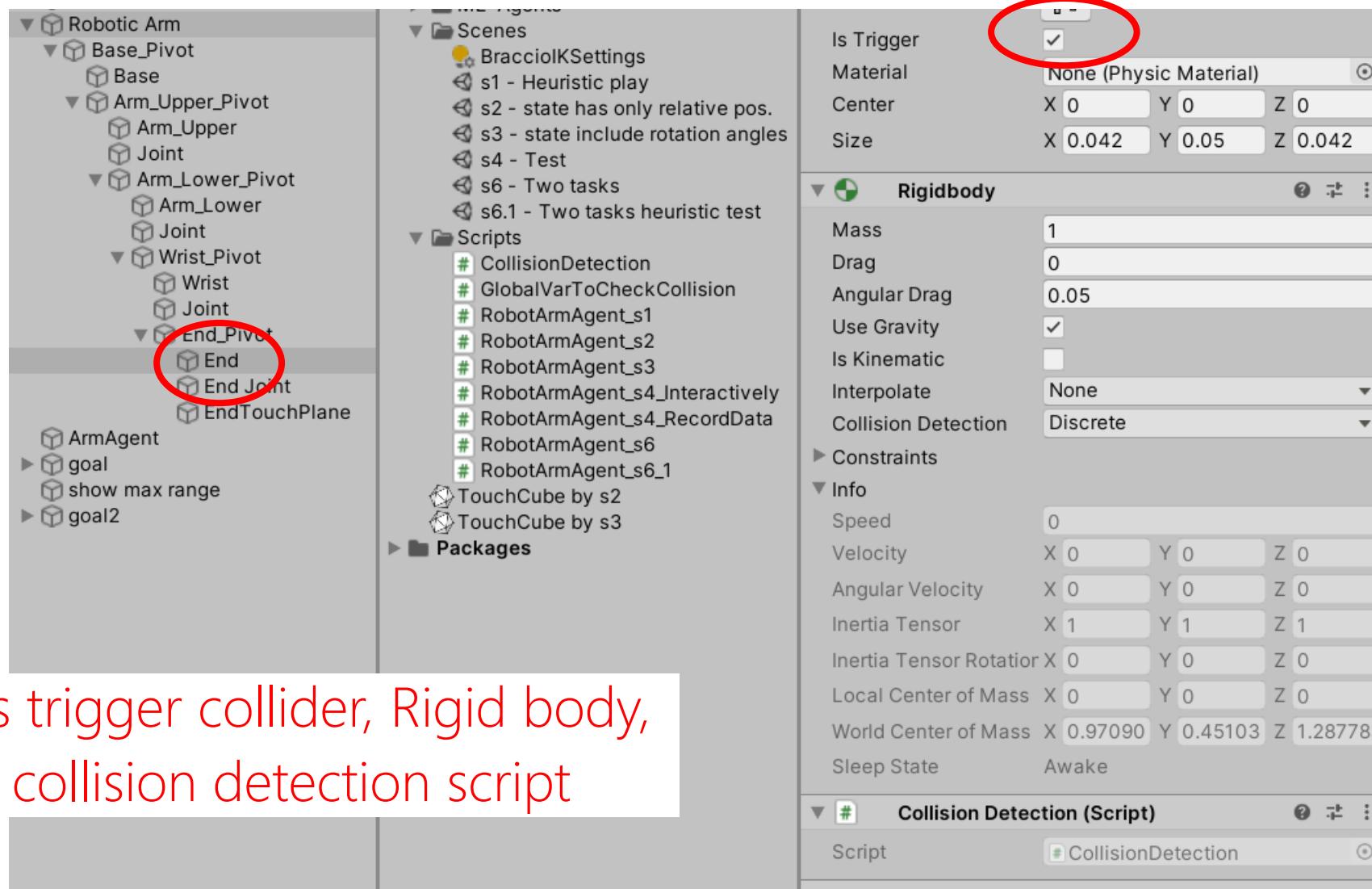
Add trigger collider, Rigid body, and collision detection script to Lower Arm



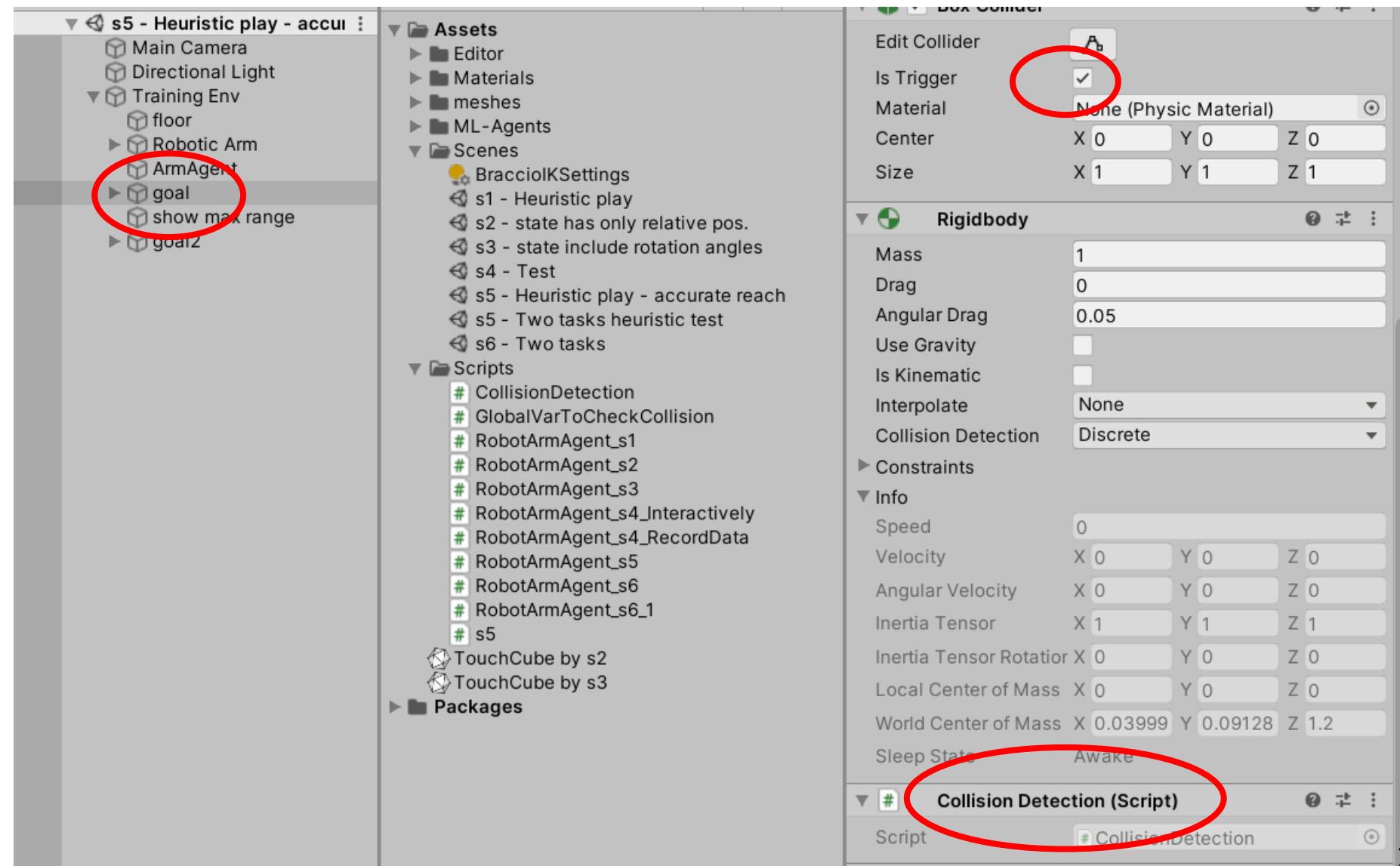
Add trigger collider, Rigid body, and collision detection script to Wrist



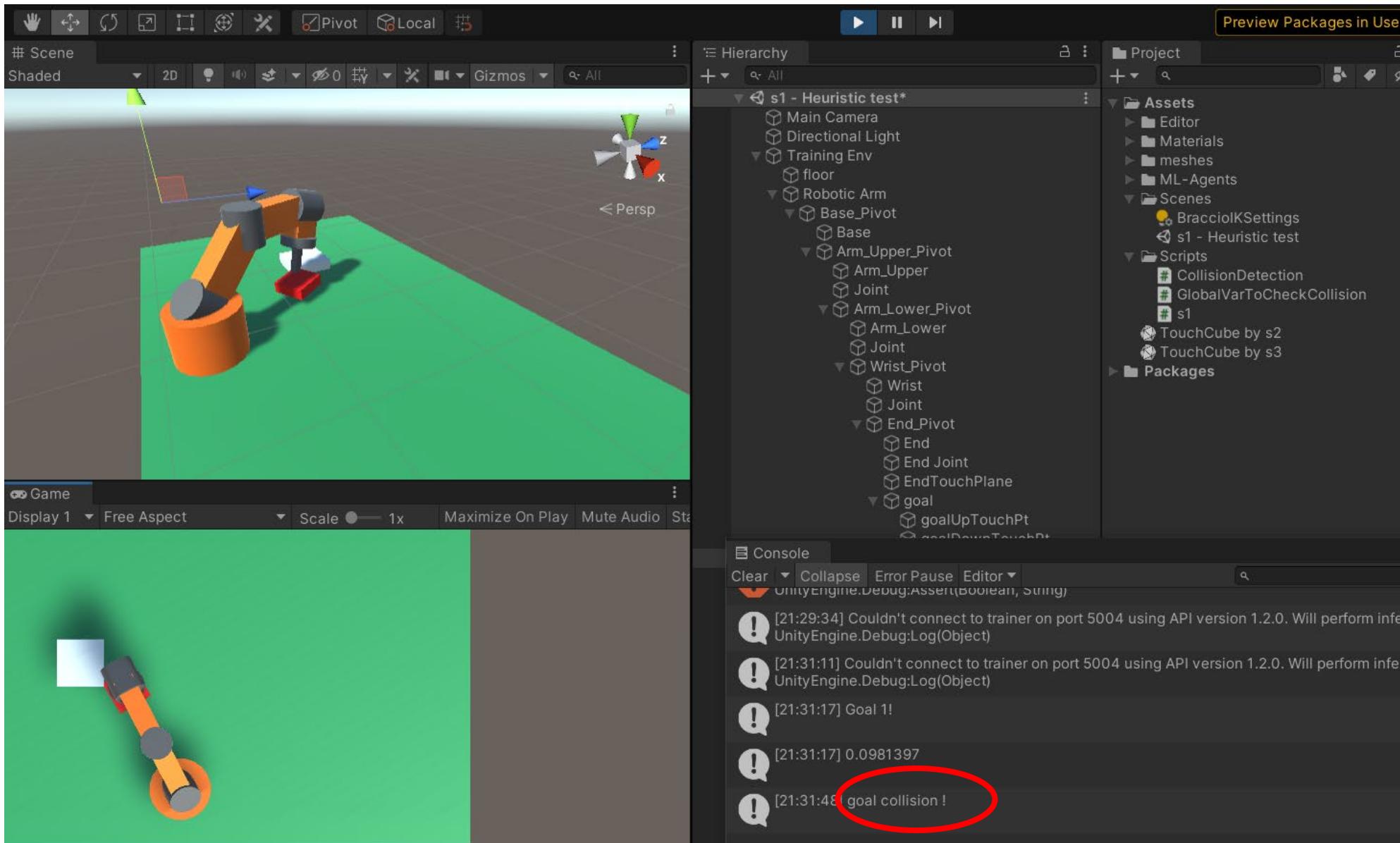
Add trigger collider, Rigid body, and collision detection script to Robot End



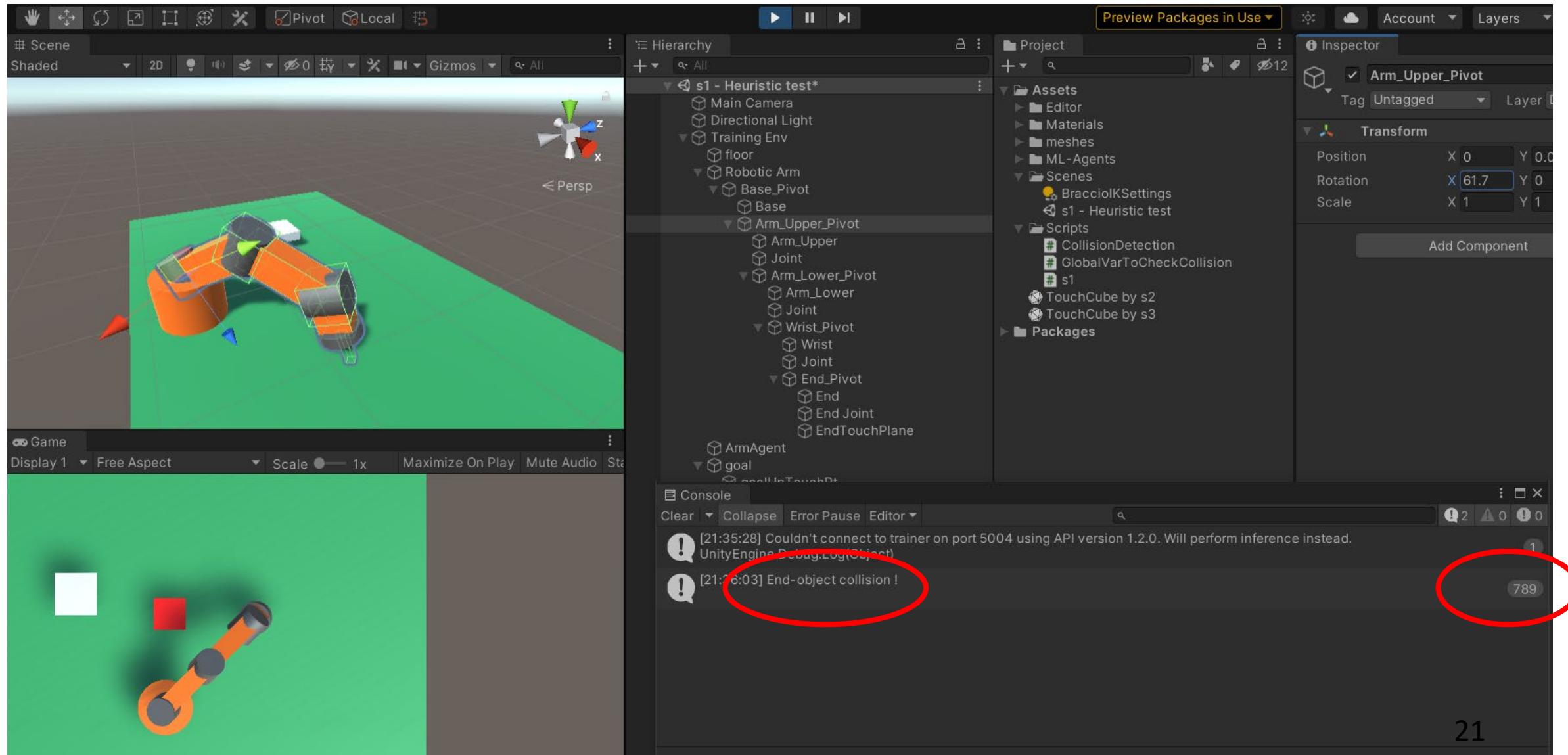
Add trigger collider, Rigid body, and collision detection script to goal object



Control the robot arm to collide with the floor or goal and check the error message

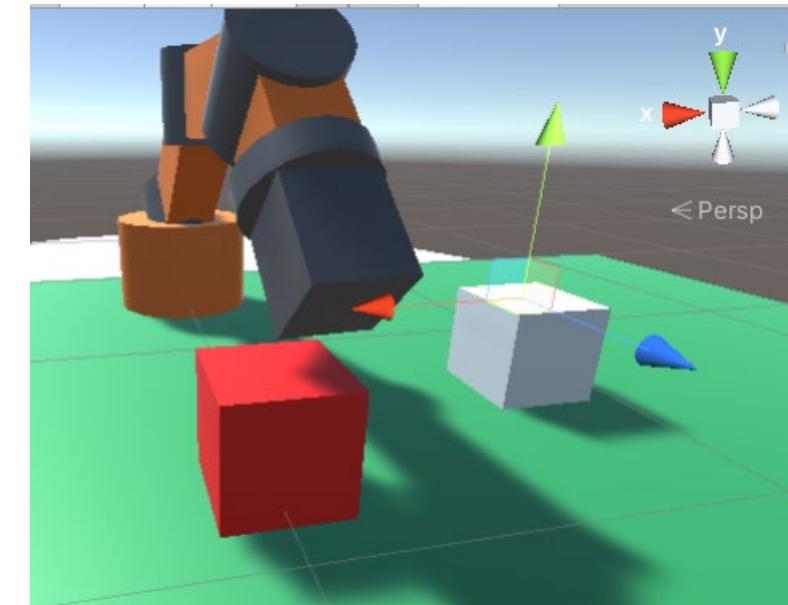
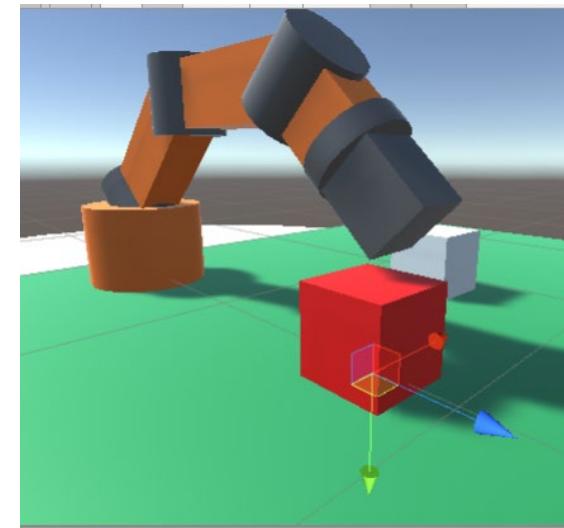
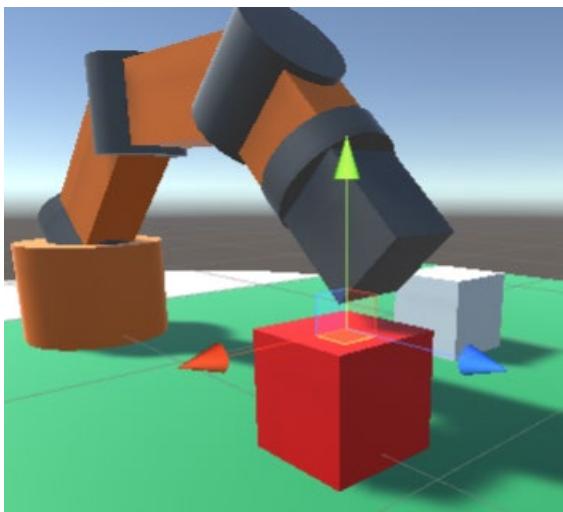
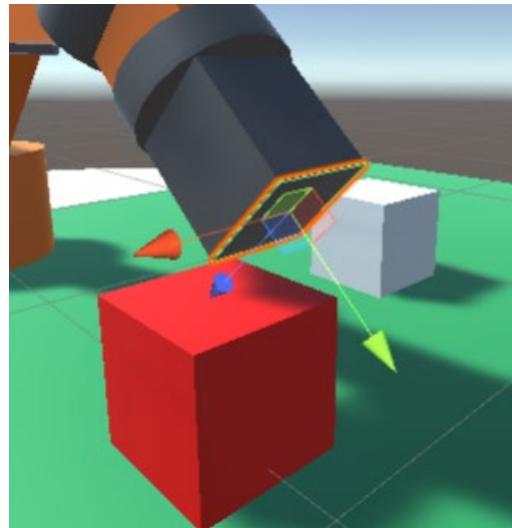
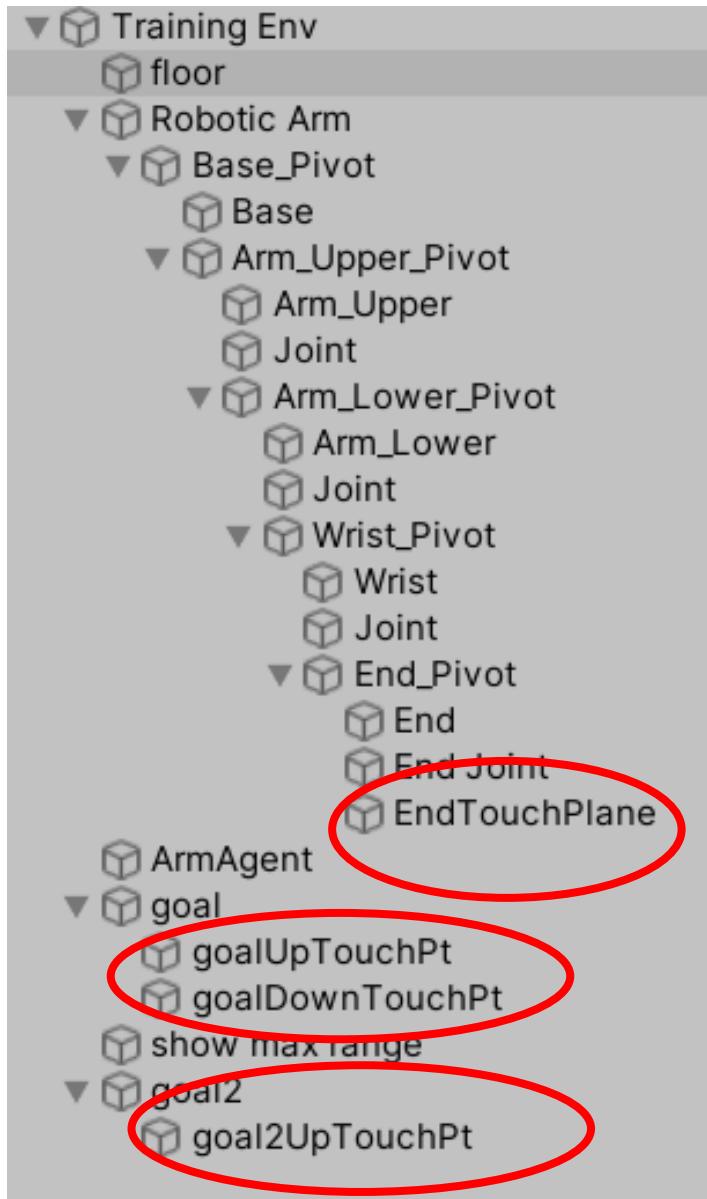


Control the robot arm to collide with the floor or goal and check the error message



Goal reaching detections

Points for goal-reaching detection



Use dx, dy and dz to determine point touch

```
dx = Mathf.Abs(pt1.position.x - pt2.position.x);
dz = Mathf.Abs(pt1.position.z - pt2.position.z);
dy = pt1.position.y - pt2.position.y;
//msg = dx.ToString() + ", " + dy.ToString() + ", " + dz.ToString();
//print(msg);
if (dy > 0 && dy < threshold && dx < threshold && dz < threshold)
    return true;
else
    return false;
```

Manually control robot arm to reach goal (avoid collision!)

The Unity Editor interface is shown, featuring the Scene View, Hierarchy View, Project View, Inspector View, and Console View.

Scene View: Displays a 3D scene with a robotic arm (orange and black) reaching towards a red cube. A green cube is also present on the green floor. A coordinate system (x, y, z) is visible in the top right corner of the scene view.

Hierarchy View: Shows the project structure with scenes like s5 - Heuristic play - accur, Assets, and Scenes. Under the Robotic Arm, there are components like Base_Pivot, Arm_Upper_Pivot, Arm_Lower_Pivot, Wrist_Pivot, and End_Pivot.

Inspector View: Shows the properties for the selected object, "Arm_Lower_Pivot". The rotation values are highlighted with a red circle: X: 60.5 and Z: 1. The Inspector also shows the Transform component with Position (X: 0, Y: 0.124, Z: 0), Rotation (X: 60.5, Y: 0, Z: 1), and Scale (X: 1, Y: 1, Z: 1).

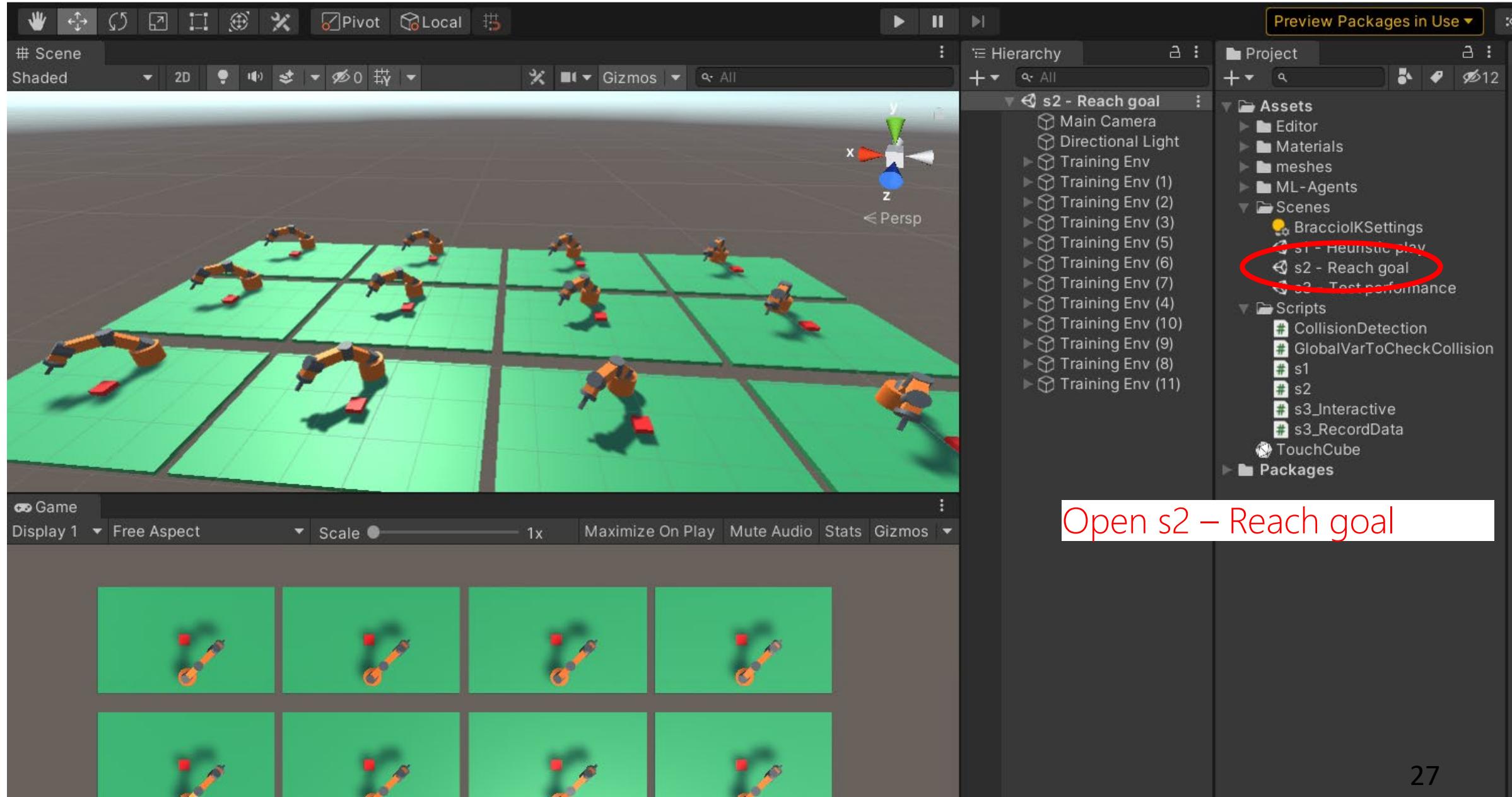
Console View: Displays log messages. The message "[16:28:18] Goal 1!" is circled in red.

Text Labels:

- 2DOF using keyboards**: Located in the bottom left corner of the Unity interface.
- Base: ← and →**: Control for the base of the arm.
- wrist: ↑ and ↓**: Control for the wrist of the arm.
- 2DOF using Inspector window**: Located in the middle right area of the screen.
- Adjust Upper/Lower arm in Inspector window**: Sub-instruction for the 2DOF using Inspector window.

(2) Train AI to operate robot arm to reach goal

4. Open the training environment



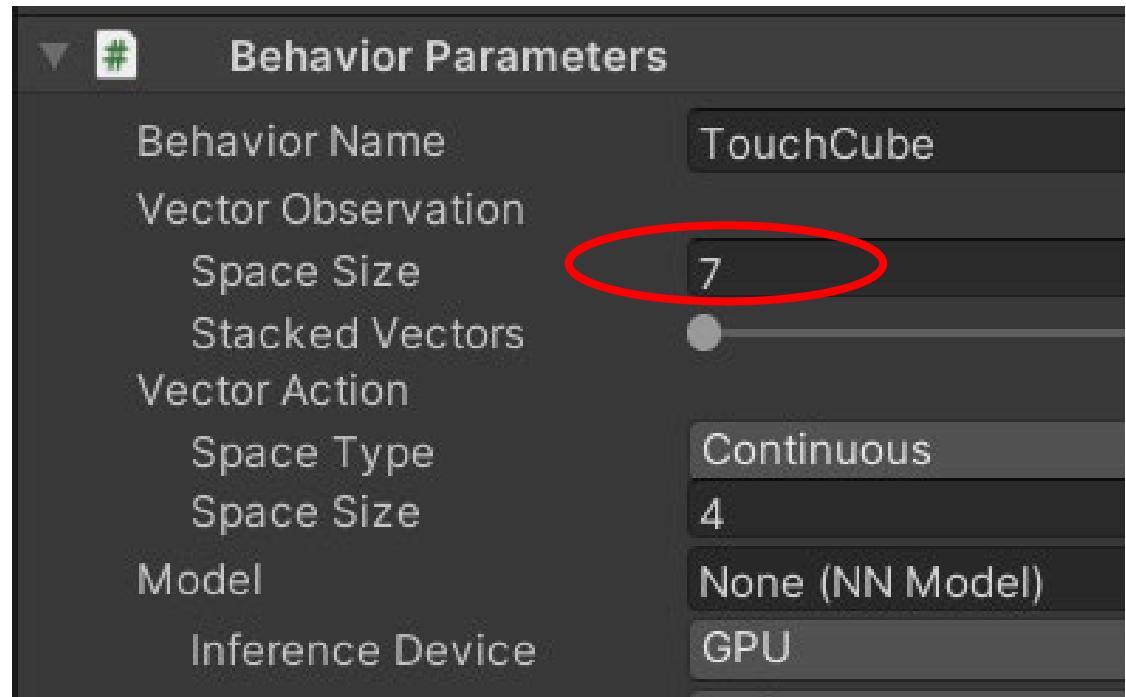
Use polar system to generate initial position

```
//use polar coordinate to calculate x, z to place goal1
float radius = Random.Range(0.8f, 1.5f);
float theta = (Random.Range(-80.0f, 80.0f) / 180.0f) * Mathf.PI;
float x = radius * Mathf.Sin(theta);
float z = radius * Mathf.Cos(theta);
goal.transform.localPosition = new Vector3(x, -1.46f, z);
goal.rotation = GoalRotation;
```

State has 7 variables

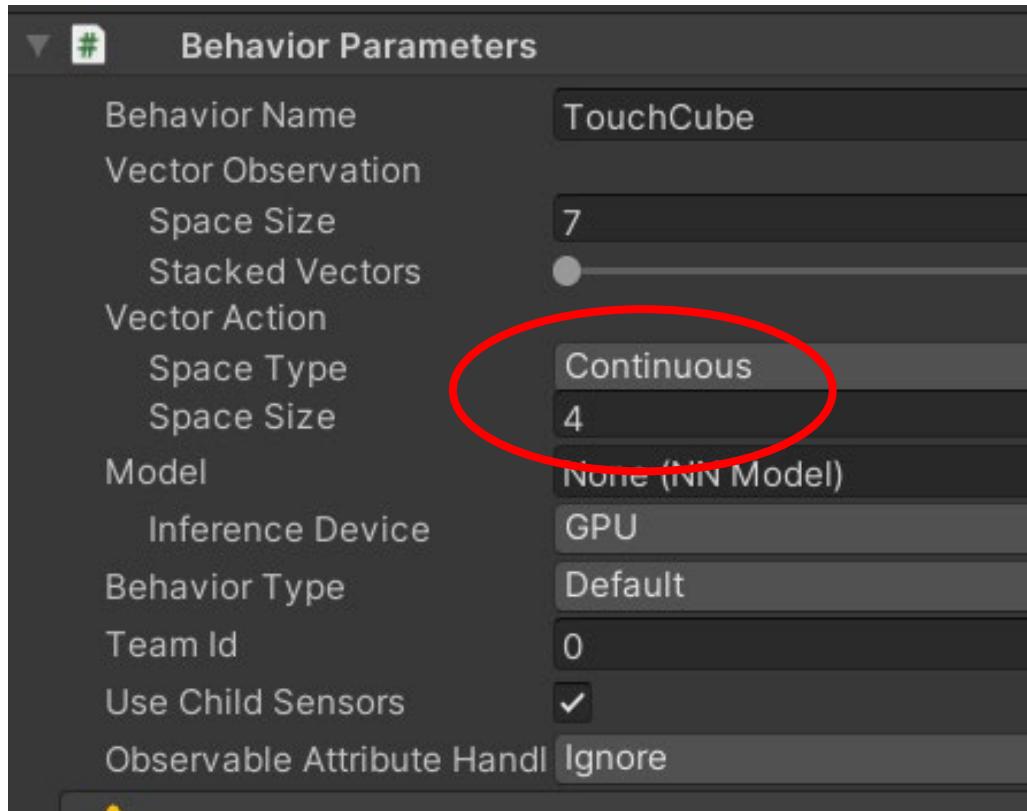
```
sensor.AddObservation(EndTouchPlane.position - goalUpTouchPt.transform.position);  
  
float BaseRotationAngle = UnityEditor.TransformUtils.GetInspectorRotation(BasePivot).y;  
float UArmRotationAngle = UnityEditor.TransformUtils.GetInspectorRotation(UpperPivot).x;  
float LArmRotationAngle = UnityEditor.TransformUtils.GetInspectorRotation(LowerPivot).x;  
float WRotationAngle = UnityEditor.TransformUtils.GetInspectorRotation(WristPivot).x;
```

```
sensor.AddObservation(BaseRotationAngle);  
sensor.AddObservation(UArmRotationAngle);  
sensor.AddObservation(LArmRotationAngle);  
sensor.AddObservation(WRotationAngle);
```



Action has 4 values

```
BasePivot.Rotate(0, vectorAction[0] * speed, 0);  
UpperPivot.Rotate(vectorAction[1] * speed, 0, 0);  
LowerPivot.Rotate(vectorAction[2] * speed, 0, 0);  
WristPivot.Rotate(vectorAction[3] * speed, 0, 0);
```



3 types of rewards

```
AddReward(-0.005f);

BasePivot.Rotate(0, vectorAction[0] * speed, 0);
UpperPivot.Rotate(vectorAction[1] * speed, 0, 0);
LowerPivot.Rotate(vectorAction[2] * speed, 0, 0);
WristPivot.Rotate(vectorAction[3] * speed, 0, 0);

//if rotation angle is out of range or collision happens, terminate this training session
if (!Rotation_in_range() || MyGlobalVar.LowerArmCollisionHappens || MyGlobalVar.WristCollisionHappens ||
    MyGlobalVar.EndCollisionHappens || MyGlobalVar.goalCollisionHappens)
{
    AddReward(-5.0f);
    EndEpisode();
}

float distToGoal = Vector3.Distance(EndTouchPlane.position, goalUpTouchPt.position);
if (distToGoal <= 0.1f && (EndTouchPlane.position.y > goalUpTouchPt.position.y))
{
    print("Goal 1!\n");
    AddReward(20.0f);
}
```

Training configuration file

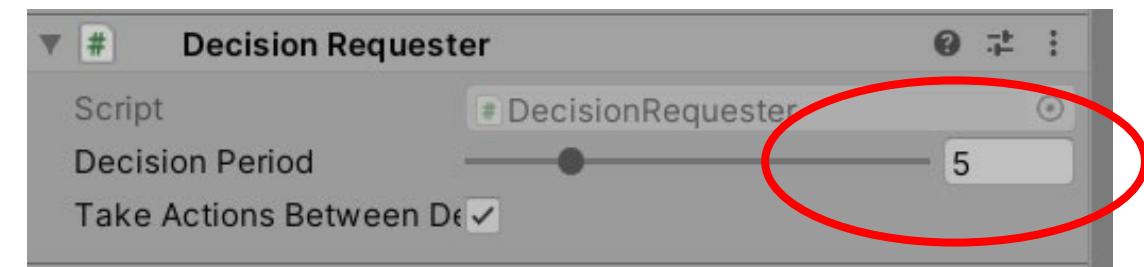
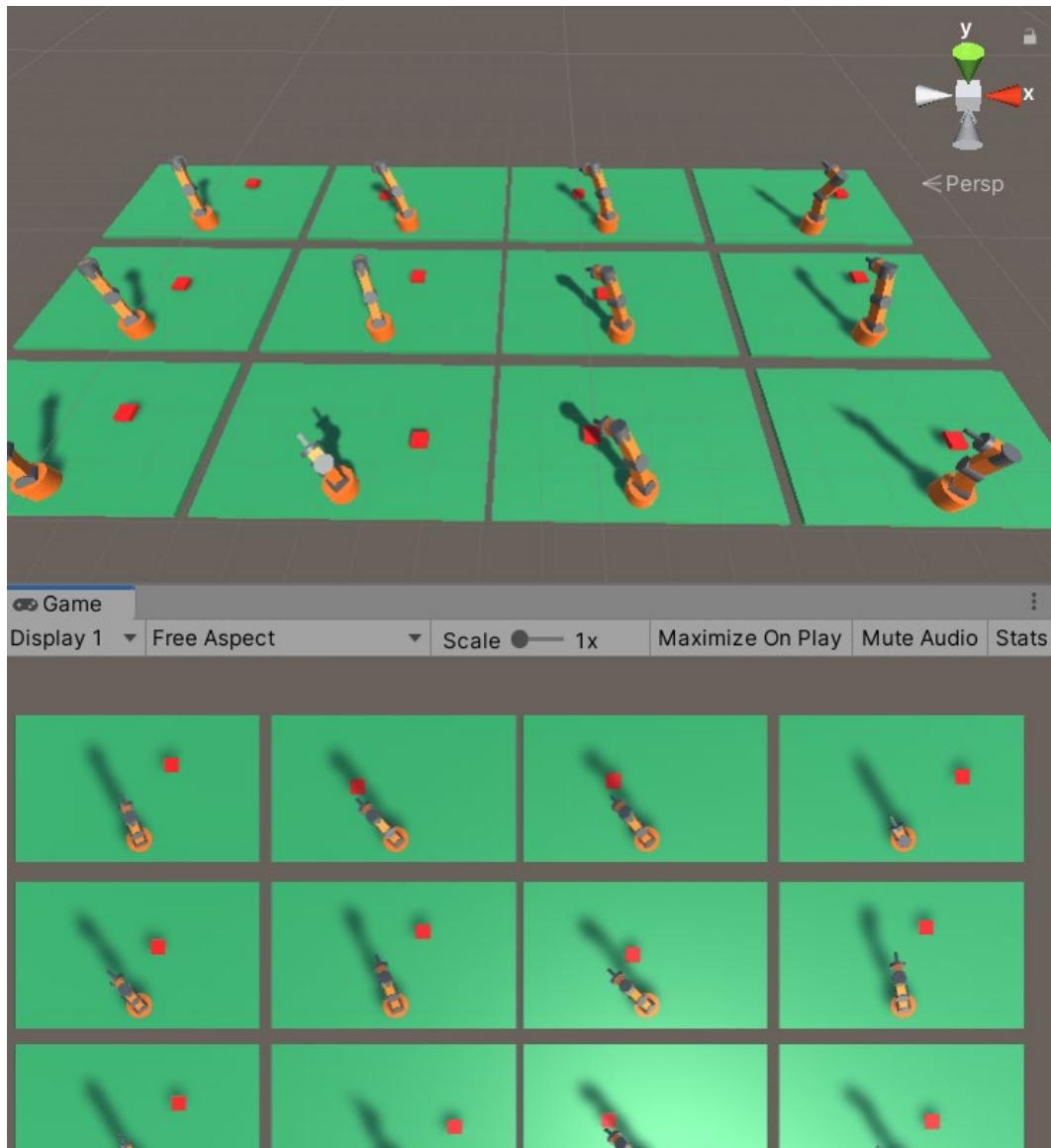
```
TouchCube:  
    trainer_type: ppo  
    hyperparameters:  
        batch_size: 2048  
        buffer_size: 20480  
        learning_rate: 0.0003  
        beta: 0.001  
        epsilon: 0.2  
        lambd: 0.95  
        num_epoch: 3  
        learning_rate_schedule:  
            network_settings:  
                normalize: true  
                hidden_units: 512  
                num_layers: 3  
                vis_encode_type: s  
            reward_signals:  
                extrinsic:  
                    gamma: 0.995  
                    strength: 1.0  
    keep_checkpoints: 5  
    max_steps: 5000000  
    time_horizon: 2000  
    summary_freq: 30000  
    threaded: true
```

In one environment the red cube runs very fast. Why?

```
if(MyGlobalVar.LowerArmCollisionHappens || MyGlobalVar.WristCollisionHappens || MyGlobalVar.EndCollisionHappens || MyGlobalVar.goalCollisionHappens)
{
    msg = System.DateTime.Now.ToShortTimeString();
    msg = msg + trainingVE.name + MyGlobalVar.LowerArmCollisionHappens;
    msg = msg + MyGlobalVar.WristCollisionHappens.ToString() + ", " +
        + ", " + MyGlobalVar.goalCollisionHappens.ToString() + "\n";
    print(msg);
    AddReward(-5.0f);
    EndEpisode();
}
```

下午 04:34	Training Env (2)	False, False, True, False
下午 04:34	Training Env (2)	False, False, True, False
下午 04:34	Training Env (2)	False, False, True, False
下午 04:34	Training Env (2)	False, False, True, False
下午 04:34	Training Env (2)	False, False, True, False
下午 04:34	Training Env (2)	False, False, True, False
下午 04:34	Training Env (2)	False, False, True, False
下午 04:34	Training Env (2)	False, False, True, False
下午 04:34	Training Env (2)	False, False, True, False
下午 04:34	Training Env (2)	False, True, False, False

Use default decision period = 5, decision period = 1 will result strange arm behavior



Print environment number that reaches goal



```
msg = System.DateTime.Now.ToString("HH:mm:ss");
msg = msg + trainingVE.name + " Goal 1! ==> " + distToGoal.ToString() + "\n";
print(msg);
AddReward(20.0f);
EndEpisode();
```

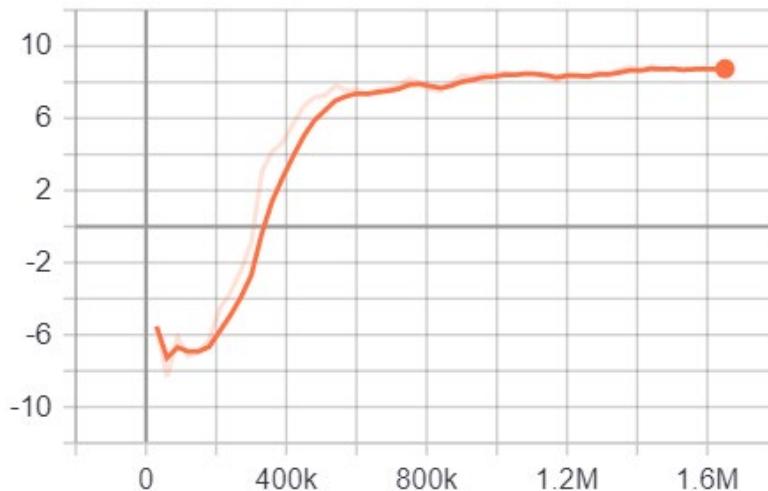
```
[16:38:06] 下午 04:38Training Env Goal 1! ==> 0.05521972
[16:38:07] 下午 04:38Training Env (10) Goal 1! ==> 0.09773364
[16:38:08] 下午 04:38Training Env (6) Goal 1! ==> 0.08763794
[16:38:09] 下午 04:38Training Env (6) Goal 1! ==> 0.09873476
[16:38:22] 下午 04:38Training Env (5) Goal 1! ==> 0.08890654
[16:38:22] 下午 04:38Training Env (4) Goal 1! ==> 0.09917516
[16:38:44] 下午 04:38Training Env (9) Goal 1! ==> 0.09907977
[16:38:45] 下午 04:38Training Env Goal 1! ==> 0.09666377
```

Results after 1.7M steps, looks promising

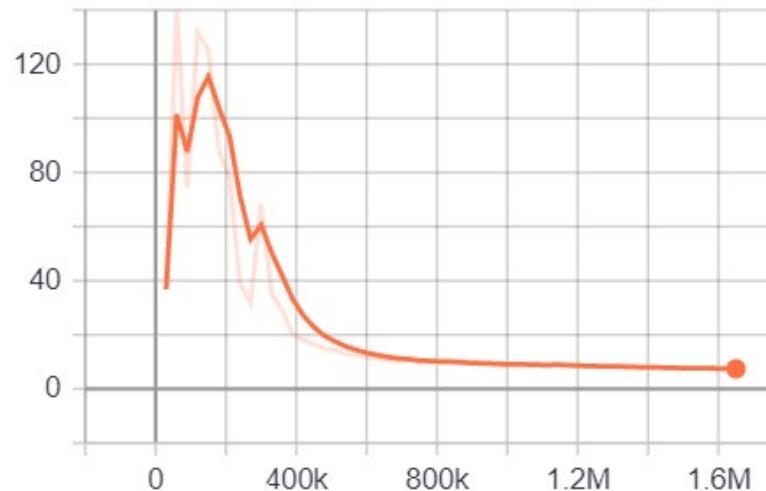
```
TouchCube. Step: 510000. Time Elapsed: 667.226 s. Mean Reward: 7.267. Std of Reward: 12.186. Train
TouchCube. Step: 540000. Time Elapsed: 707.059 s. Mean Reward: 7.803. Std of Reward: 12.211. Train
TouchCube. Step: 570000. Time Elapsed: 755.981 s. Mean Reward: 7.567. Std of Reward: 12.226. Train
TouchCube. Step: 600000. Time Elapsed: 796.691 s. Mean Reward: 7.605. Std of Reward: 12.246. Train
TouchCube. Step: 630000. Time Elapsed: 848.452 s. Mean Reward: 7.315. Std of Reward: 12.247. Train
TouchCube. Step: 660000. Time Elapsed: 890.687 s. Mean Reward: 7.594. Std of Reward: 12.260. Train
TouchCube. Step: 690000. Time Elapsed: 947.977 s. Mean Reward: 7.625. Std of Reward: 12.267. Train
TouchCube. Step: 720000. Time Elapsed: 988.268 s. Mean Reward: 7.792. Std of Reward: 12.319. Train
TouchCube. Step: 750000. Time Elapsed: 1037.415 s. Mean Reward: 8.202. Std of Reward: 12.260. Train
TouchCube. Step: 780000. Time Elapsed: 1088.936 s. Mean Reward: 7.957. Std of Reward: 12.165. Train
TouchCube. Step: 810000. Time Elapsed: 1141.701 s. Mean Reward: 7.619. Std of Reward: 12.277. Train
TouchCube. Step: 840000. Time Elapsed: 1182.285 s. Mean Reward: 7.524. Std of Reward: 12.280. Train
TouchCube. Step: 870000. Time Elapsed: 1222.890 s. Mean Reward: 7.983. Std of Reward: 12.273. Train
TouchCube. Step: 900000. Time Elapsed: 1274.109 s. Mean Reward: 8.380. Std of Reward: 12.261. Train
TouchCube. Step: 930000. Time Elapsed: 1315.381 s. Mean Reward: 8.268. Std of Reward: 12.277. Train
TouchCube. Step: 960000. Time Elapsed: 1366.218 s. Mean Reward: 8.481. Std of Reward: 12.253. Train
TouchCube. Step: 990000. Time Elapsed: 1413.950 s. Mean Reward: 8.361. Std of Reward: 12.270. Train
ation.py:93] Converting to results\1\TouchCube\TouchCube-999992.onnx
ation.py:105] Exported results\1\TouchCube\TouchCube-999992.onnx
TouchCube. Step: 1020000. Time Elapsed: 1468.478 s. Mean Reward: 8.561. Std of Reward: 12.284. Train
TouchCube. Step: 1050000. Time Elapsed: 1512.854 s. Mean Reward: 8.413. Std of Reward: 12.279. Train
TouchCube. Step: 1080000. Time Elapsed: 1563.380 s. Mean Reward: 8.533. Std of Reward: 12.259. Train
TouchCube. Step: 1110000. Time Elapsed: 1605.463 s. Mean Reward: 8.456. Std of Reward: 12.268. Train
TouchCube. Step: 1140000. Time Elapsed: 1654.158 s. Mean Reward: 8.288. Std of Reward: 12.286. Train
TouchCube. Step: 1170000. Time Elapsed: 1696.093 s. Mean Reward: 8.091. Std of Reward: 12.291. Train
TouchCube. Step: 1200000. Time Elapsed: 1746.003 s. Mean Reward: 8.472. Std of Reward: 12.270. Train
TouchCube. Step: 1230000. Time Elapsed: 1787.666 s. Mean Reward: 8.369. Std of Reward: 12.276. Train
TouchCube. Step: 1260000. Time Elapsed: 1837.945 s. Mean Reward: 8.287. Std of Reward: 12.287. Train
TouchCube. Step: 1290000. Time Elapsed: 1879.433 s. Mean Reward: 8.588. Std of Reward: 12.260. Train
TouchCube. Step: 1320000. Time Elapsed: 1921.159 s. Mean Reward: 8.455. Std of Reward: 12.275. Train
TouchCube. Step: 1350000. Time Elapsed: 1971.810 s. Mean Reward: 8.685. Std of Reward: 12.258. Train
TouchCube. Step: 1380000. Time Elapsed: 2013.756 s. Mean Reward: 8.849. Std of Reward: 12.244. Train
TouchCube. Step: 1410000. Time Elapsed: 2063.943 s. Mean Reward: 8.627. Std of Reward: 12.264. Train
TouchCube. Step: 1440000. Time Elapsed: 2105.873 s. Mean Reward: 8.891. Std of Reward: 12.241. Train
TouchCube. Step: 1470000. Time Elapsed: 2156.424 s. Mean Reward: 8.702. Std of Reward: 12.264. Train
TouchCube. Step: 1500000. Time Elapsed: 2198.716 s. Mean Reward: 8.772. Std of Reward: 12.257. Train
```

Results after 1.7M steps, looks promising

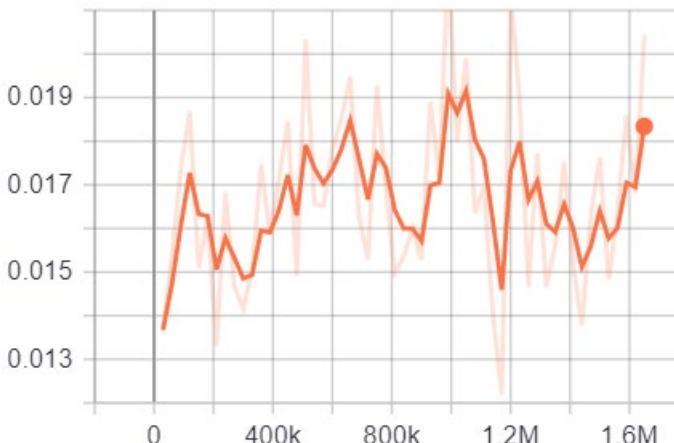
Cumulative Reward
tag: Environment/Cumulative Reward



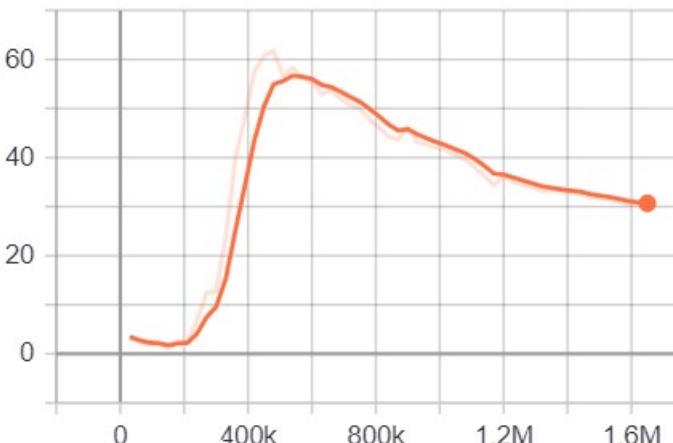
Episode Length
tag: Environment/Episode Length



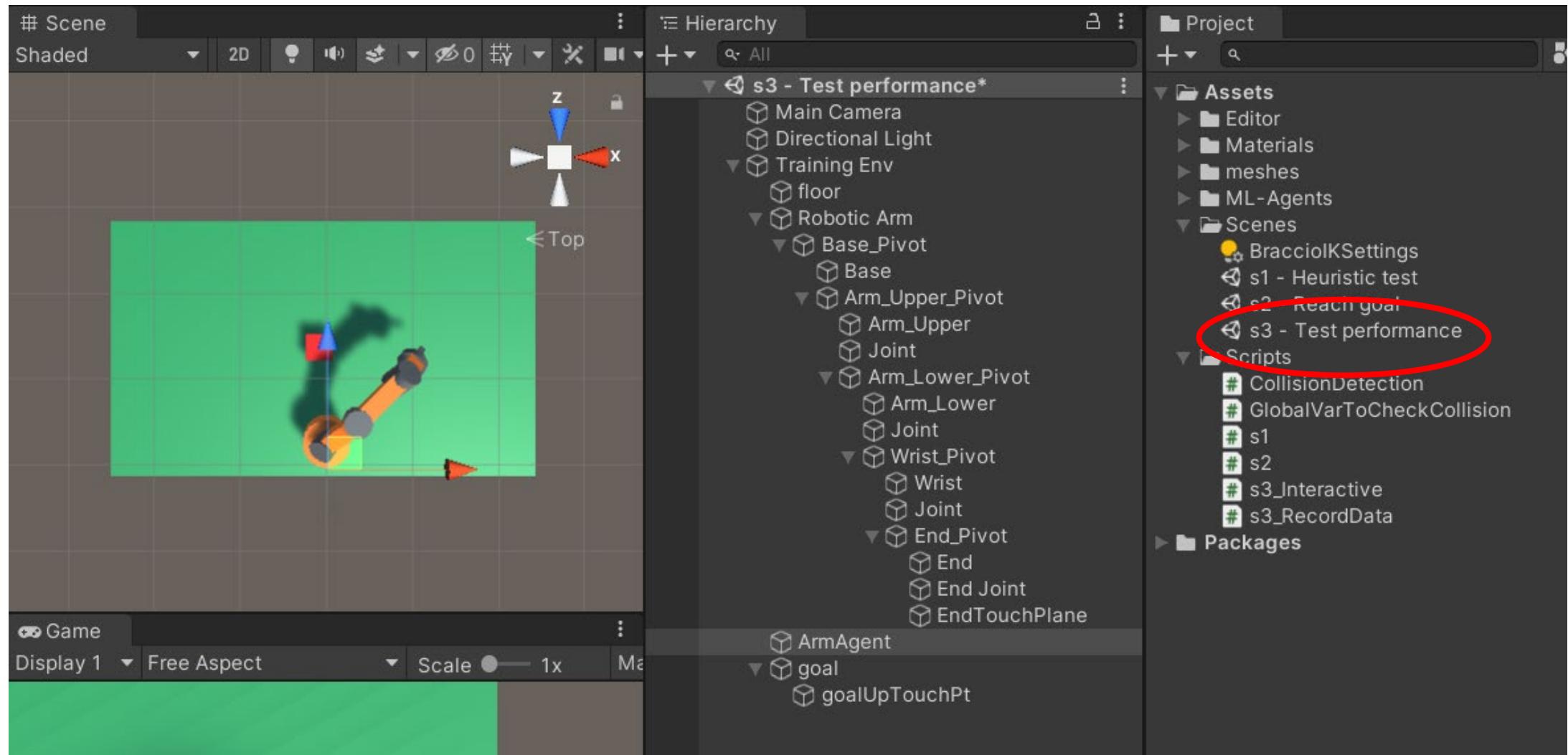
Policy Loss
tag: Losses/Policy Loss



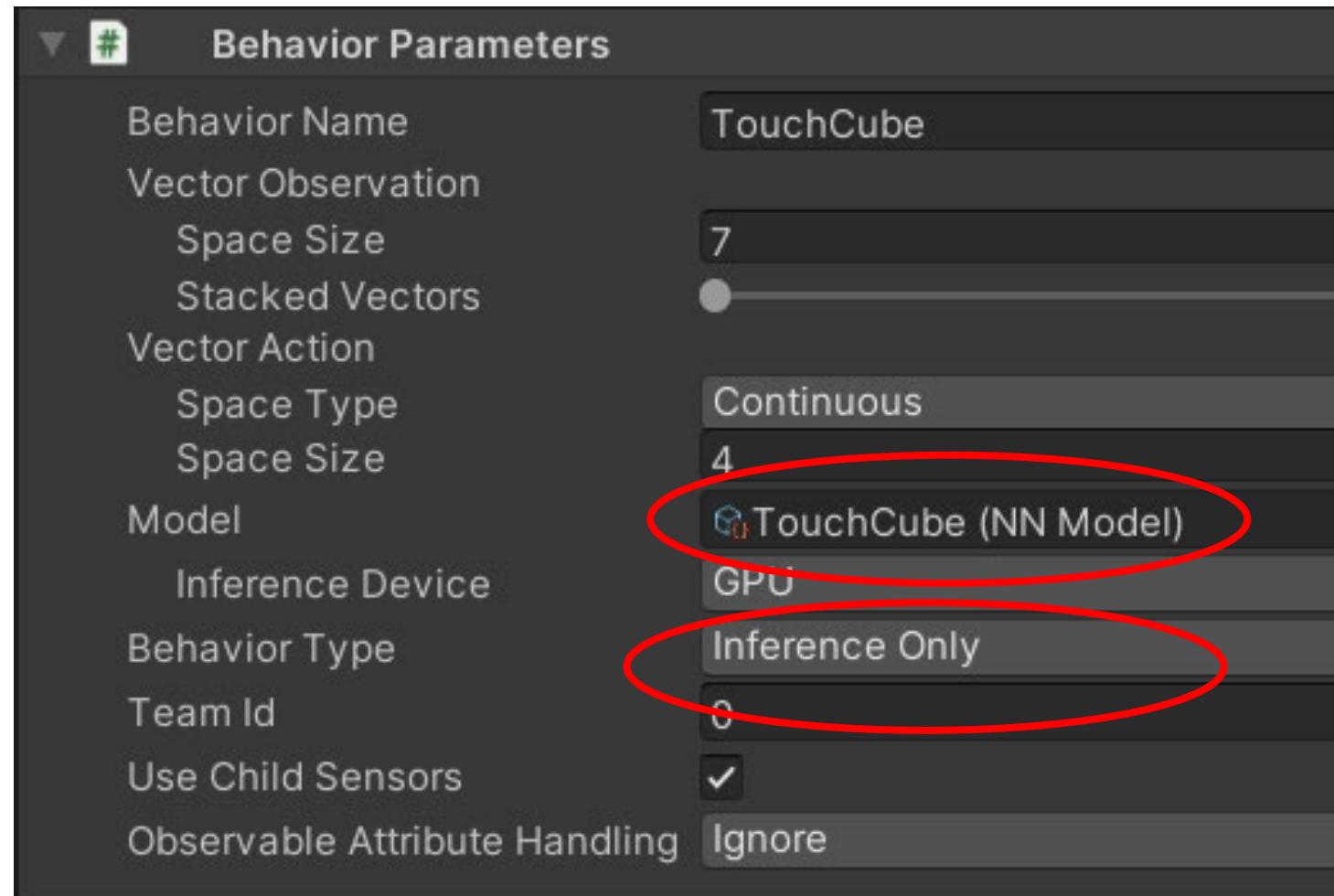
Value Loss
tag: Losses/Value Loss



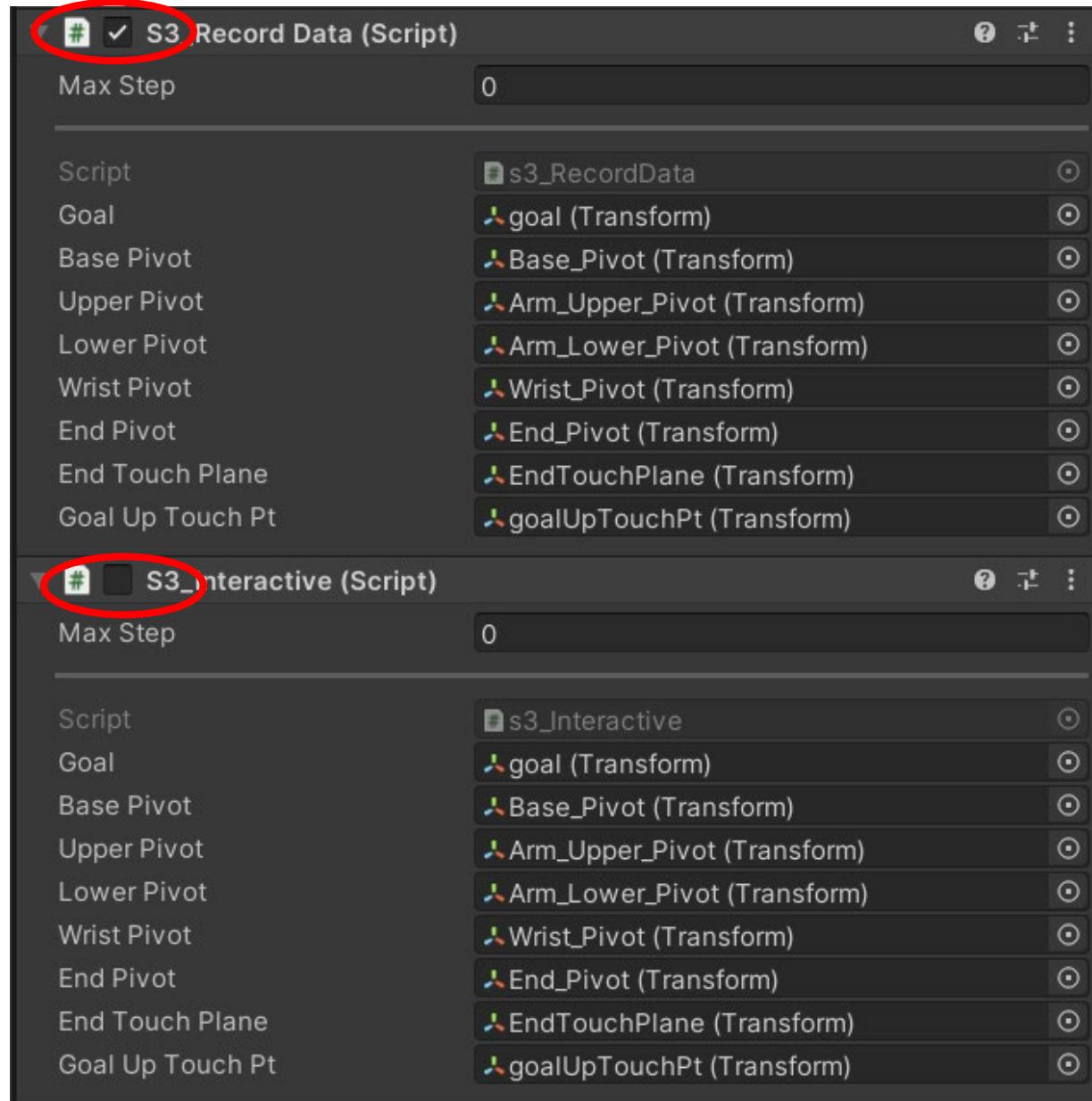
5. Test performance



Assign trained NN



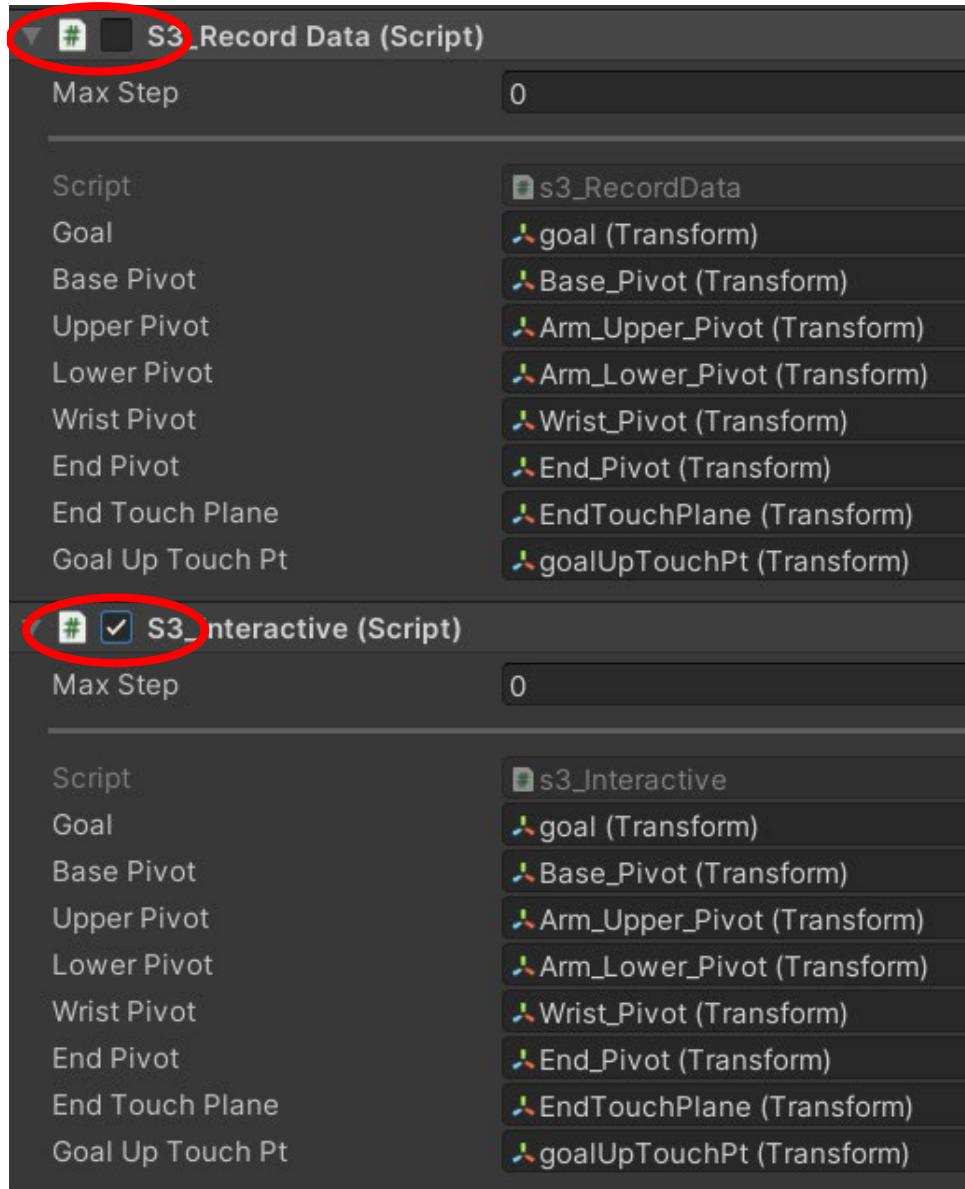
2 types of tests – (1) data recording



Uncheck interactive test

2 types of tests – (2) interactive test

Uncheck data-recording
test



Practice

1. Initial position, all rotation angles =0
2. Different robot has different initial positions.

Base pivot (0, 45, 0)

Upper pivot (45, 0, 0)

Lower pivot (45, 0, 0)

Wrist pivot (45, 0, 0)

Base pivot (0, 0, 0)

Upper pivot (0, 0, 0)

Lower pivot (0, 0, 0)

Wrist pivot (0, 0, 0)

Base pivot (0, y, 0)

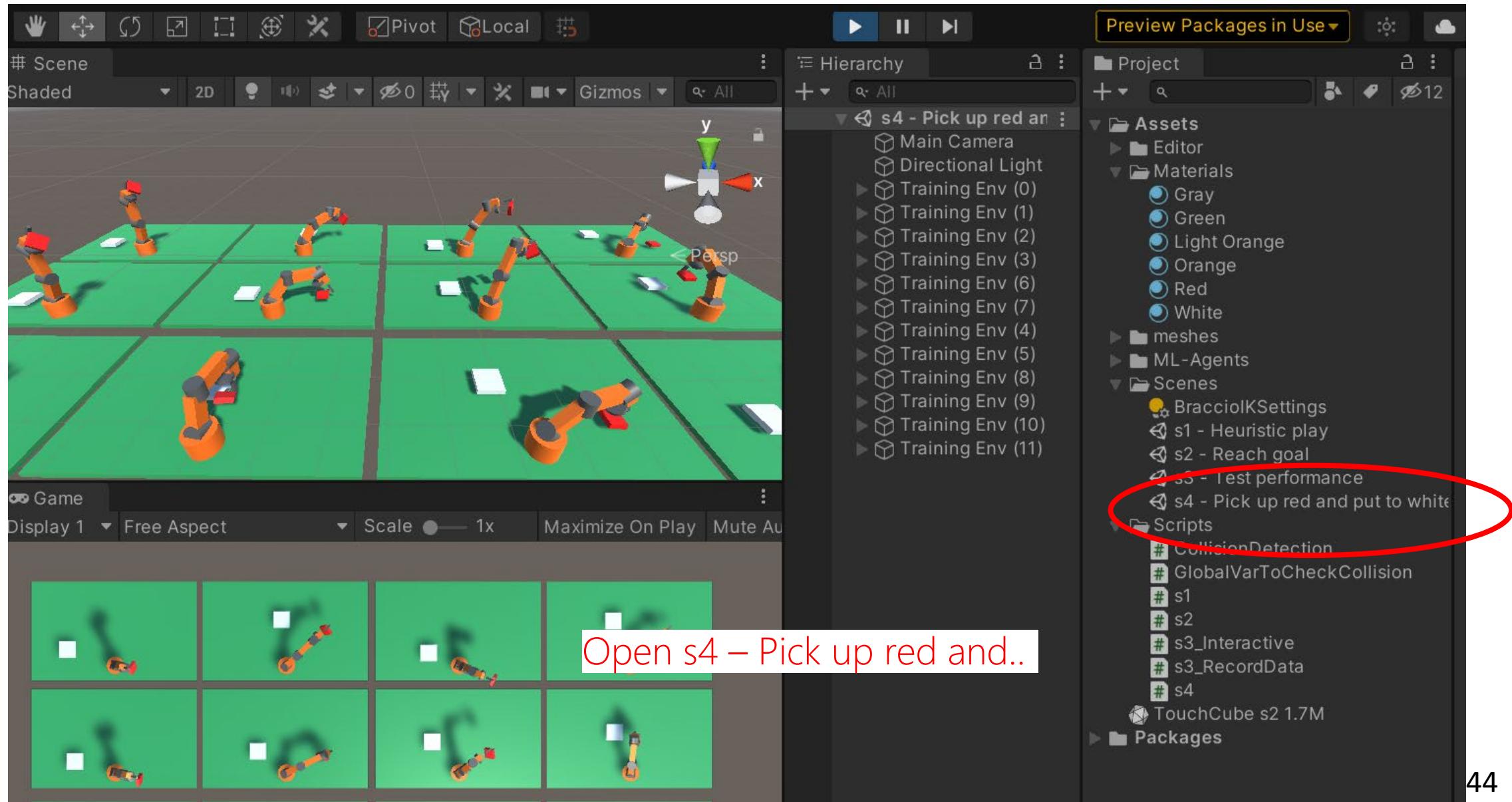
Upper pivot (x1, 0, 0)

Lower pivot (x2, 0, 0)

Wrist pivot (x3, 0, 0)

(3) Train AI to operate robot arm to do two tasks

4. Open the training environment



Use polar system to generate initial position

```
//use polar coordinate to calculate x, z to place goal1
float radius = UnityEngine.Random.Range(0.8f, 1.5f);
float theta = (UnityEngine.Random.Range(5.0f, 80.0f) / 180.0f) * Mathf.PI;
float x = radius * Mathf.Sin(theta); red cube is generated at right side
float z = radius * Mathf.Cos(theta);
goal.transform.localPosition = new Vector3(x, -1.46f, z);
goal.rotation = GoalRotation;

radius = UnityEngine.Random.Range(0.8f, 1.5f);
theta = (UnityEngine.Random.Range(-80.0f, -5.0f) / 180.0f) * Mathf.PI;
x = radius * Mathf.Sin(theta); white cube is generated at left side
z = radius * Mathf.Cos(theta);
goal2.transform.localPosition = new Vector3(x, -1.46f, z);
goal2.rotation = Goal2Rotation;
```

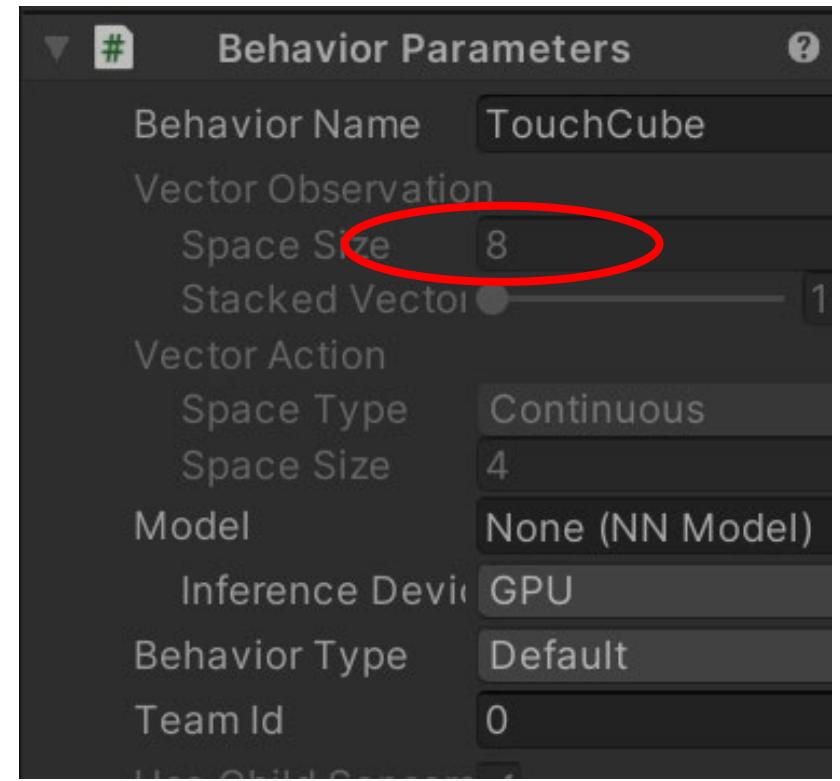
State has 8 variables

```
sensor.AddObservation(stage);

if (stage == 1)
    sensor.AddObservation(EndTouchPlane.position - goalUpTouchPt.position)
else //stage =2
    sensor.AddObservation(goalDownTouchPt.position - goal2UpTouchPt.position)

float BaseRotationAngle = UnityEditor.TransformUtils.GetInspectorRotationAngle();
float UArmRotationAngle = UnityEditor.TransformUtils.GetInspectorRotationAngle();
float LArmRotationAngle = UnityEditor.TransformUtils.GetInspectorRotationAngle();
float WRotationAngle = UnityEditor.TransformUtils.GetInspectorRotationAngle();

sensor.AddObservation(BaseRotationAngle);
sensor.AddObservation(UArmRotationAngle);
sensor.AddObservation(LArmRotationAngle);
sensor.AddObservation(WRotationAngle);
```



Action has 4 values

```
BasePivot.Rotate(0, vectorAction[0] * speed, 0);  
UpperPivot.Rotate(vectorAction[1] * speed, 0, 0);  
LowerPivot.Rotate(vectorAction[2] * speed, 0, 0);  
WristPivot.Rotate(vectorAction[3] * speed, 0, 0);
```

4 types of rewards

```
AddReward(-0.005f * (3 - stage)); // punish more for sta  
  
BasePivot.Rotate(0, vectorAction[0] * speed, 0);  
UpperPivot.Rotate(vectorAction[1] * speed, 0, 0);  
LowerPivot.Rotate(vectorAction[2] * speed, 0, 0);  
WristPivot.Rotate(vectorAction[3] * speed, 0, 0);  
  
//if rotation angle is out of range or collision happens  
if (!Rotation_in_range() || MyGlobalVar.LowerArmCollisic  
| MyGlobalVar.EndCollisionHappens || MyGlobalVar.goalCol  
{  
| AddReward(-5.0f);  
| EndEpisode();  
}
```

4 types of rewards

```
if (stage == 1)
{
    distToGoal = Vector3.Distance(EndTouchPlane.position, goalUpTouchPt.position);
    if (distToGoal <= 0.1f && (EndTouchPlane.position.y > goal2UpTouchPt.position.y))
    {
        stage = 2;
        AddReward(50.0f);
        goal.transform.parent = EndPivot.transform; //grab goal
    }
}
else //stage =2
{
    distToGoal = Vector3.Distance(goalDownTouchPt.position, goal2UpTouchPt.position);
    if (distToGoal <= 0.1f && (goalDownTouchPt.position.y > goal2UpTouchPt.position.y))
    {
        msg = System.DateTime.Now.ToShortTimeString();
        msg = msg + trainingVE.name + " Goal 2! ==> " + distToGoal.ToString() + " \n";
        print(msg);
        AddReward(100.0f);
        EndEpisode();
    }
}
```

Training configuration file

```
TouchCube:  
    trainer_type: ppo  
    hyperparameters:  
        batch_size: 2048  
        buffer_size: 20480  
        learning_rate: 0.0003  
        beta: 0.001  
        epsilon: 0.2  
        lambd: 0.95  
        num_epoch: 3  
        learning_rate_schedule:  
            network_settings:  
                normalize: true  
                hidden_units: 512  
                num_layers: 3  
                vis_encode_type: s  
            reward_signals:  
                extrinsic:  
                    gamma: 0.995  
                    strength: 1.0  
    keep_checkpoints: 5  
    max_steps: 5000000  
    time_horizon: 2000  
    summary_freq: 30000  
    threaded: true
```