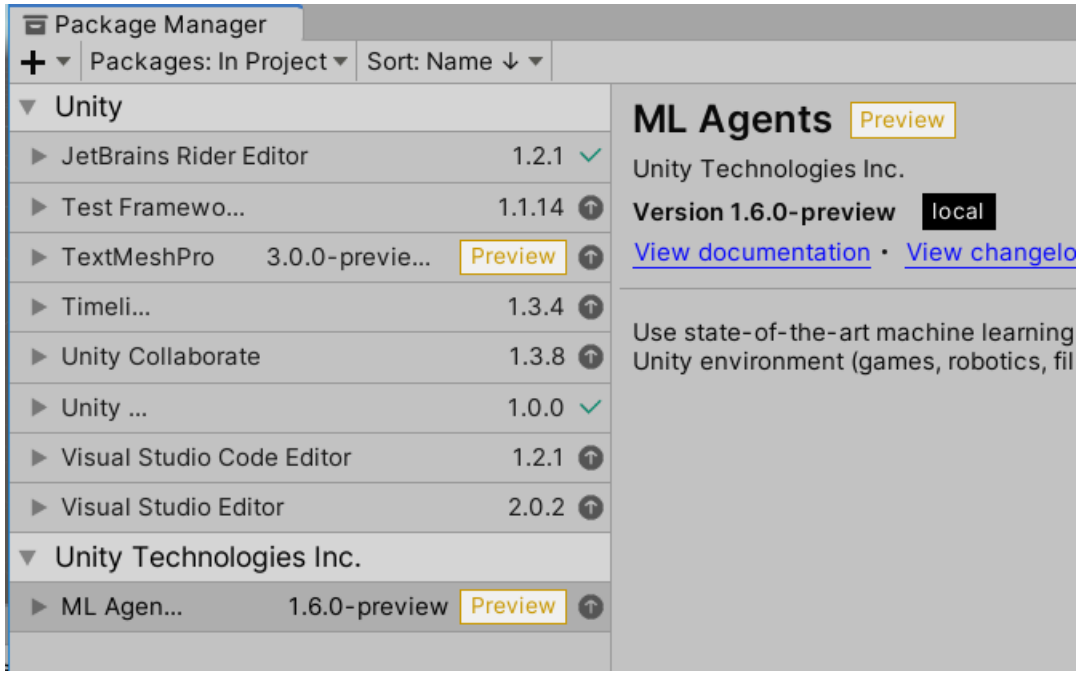
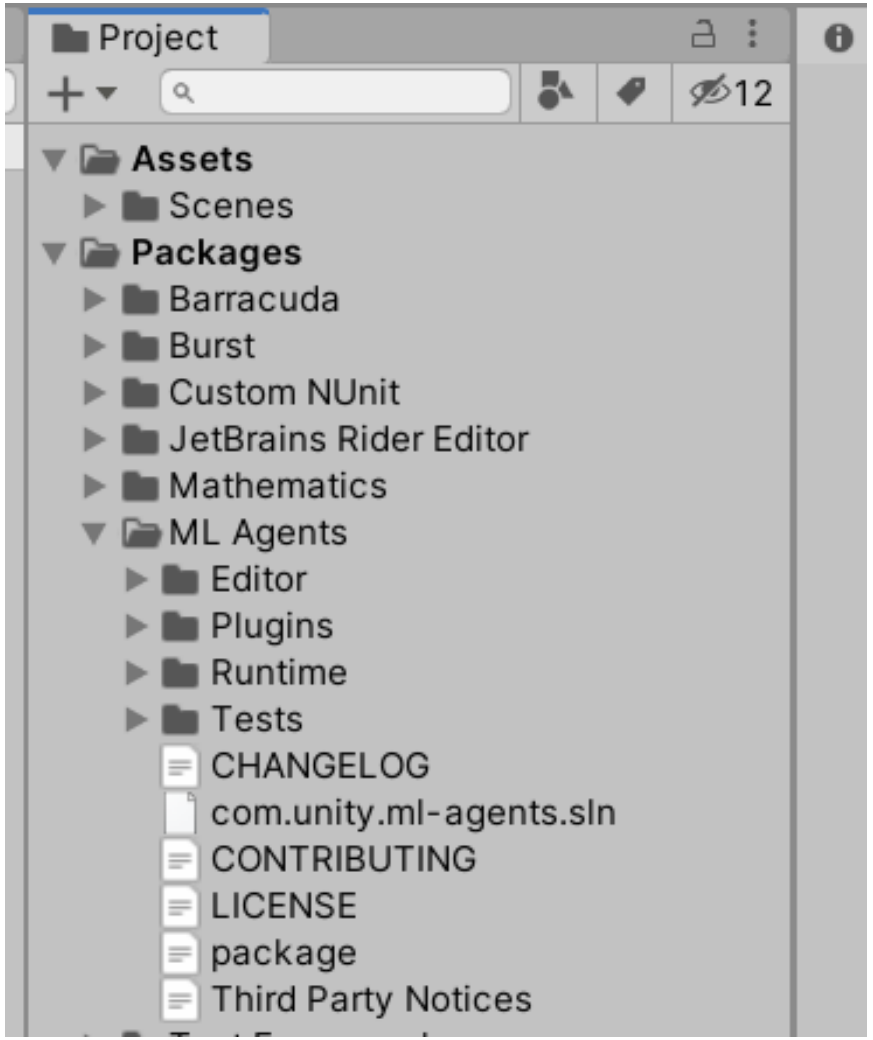
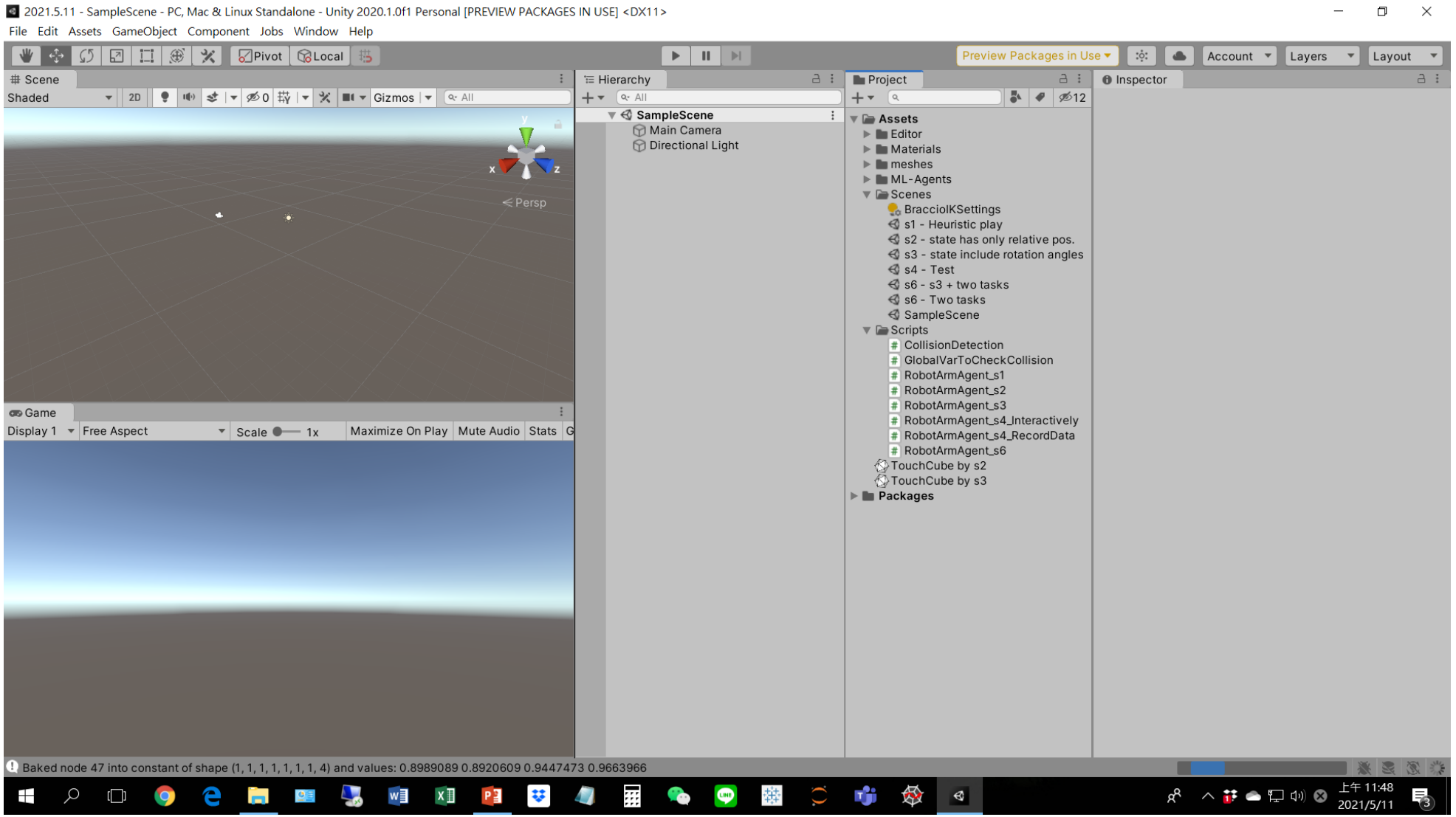
Create a new Unity project and import ML Agent package

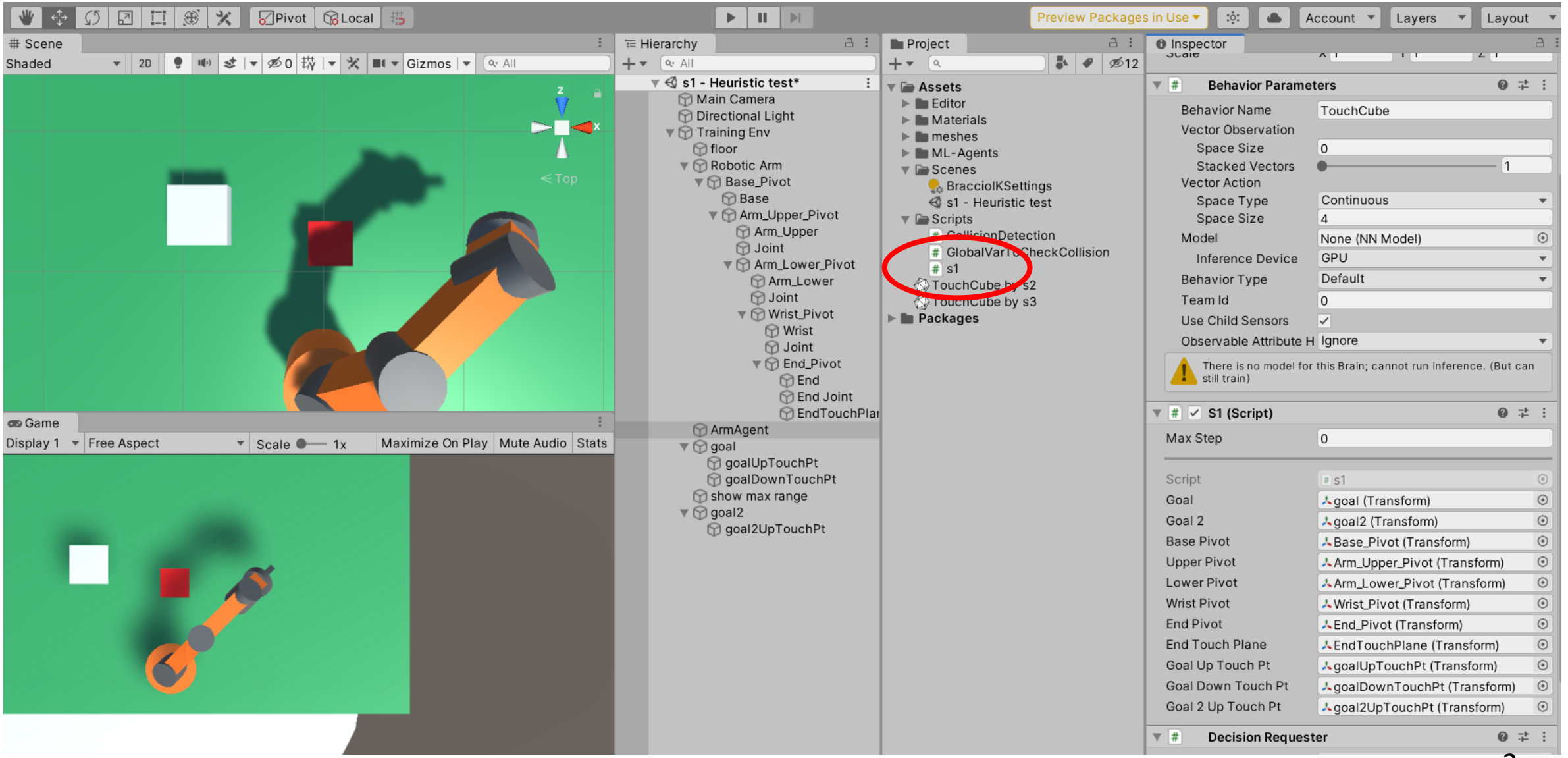




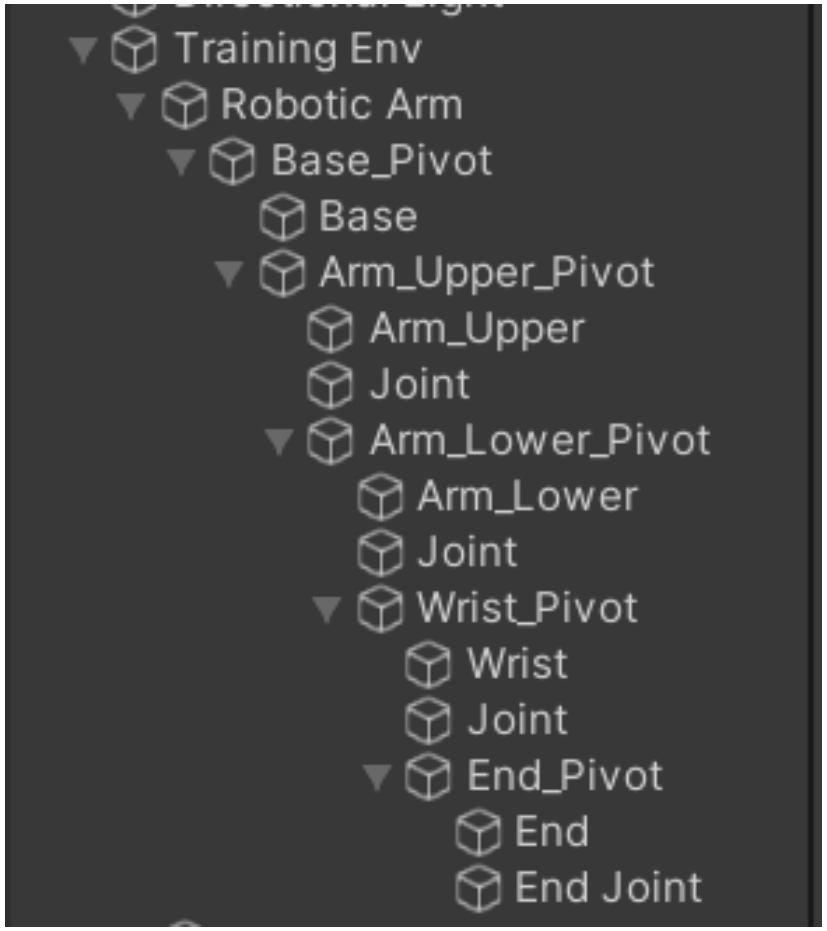
Import Robot arm package to this Unity project



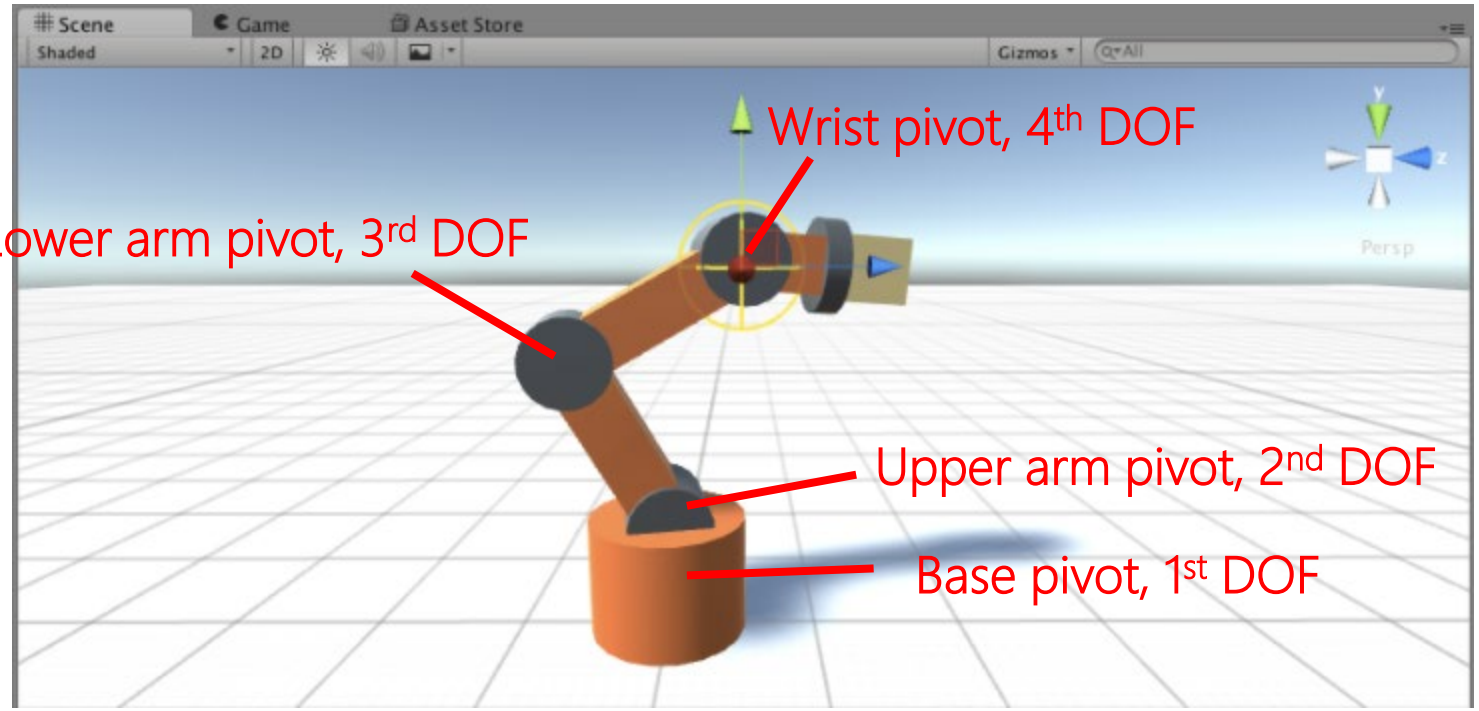
Open scene "s1 - Heuristic play"



This Unity project contains a Braccio robot arm



Alt + [▼] lets you expand all hierarchies of the robot arm

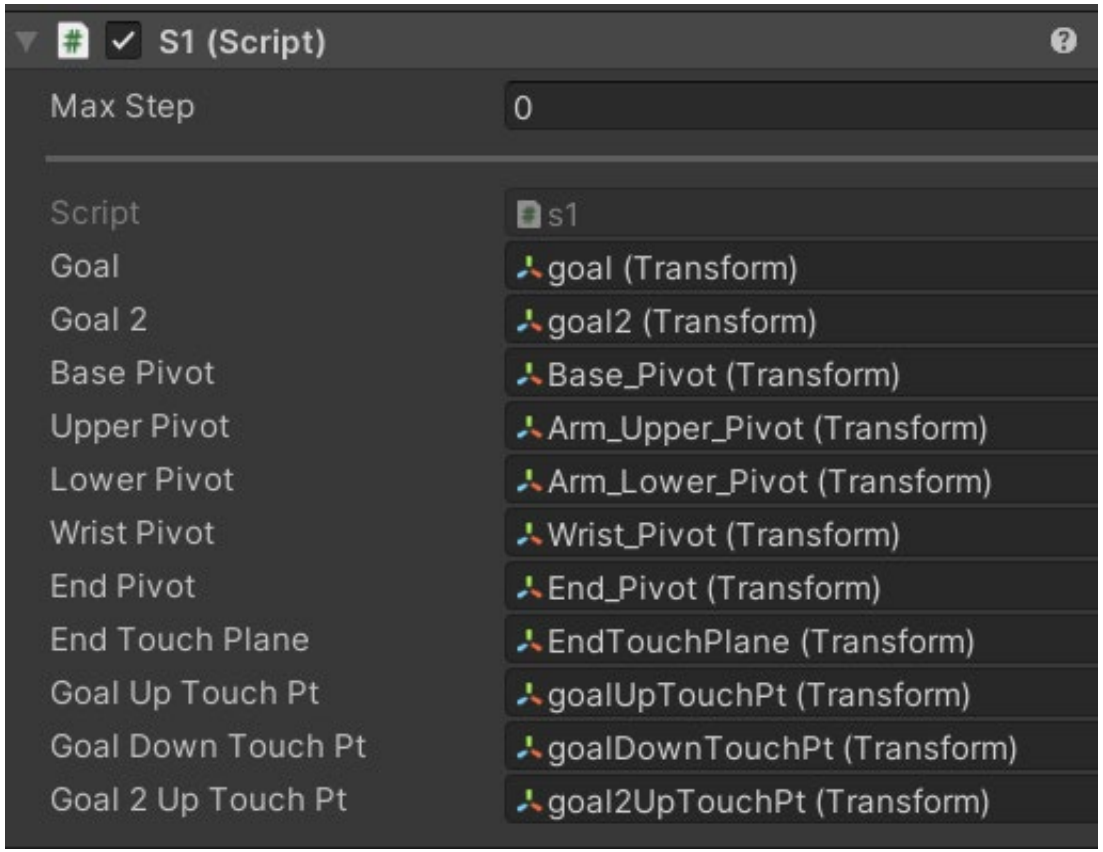


https://github.com/tanyuan/braccio-ik-unity

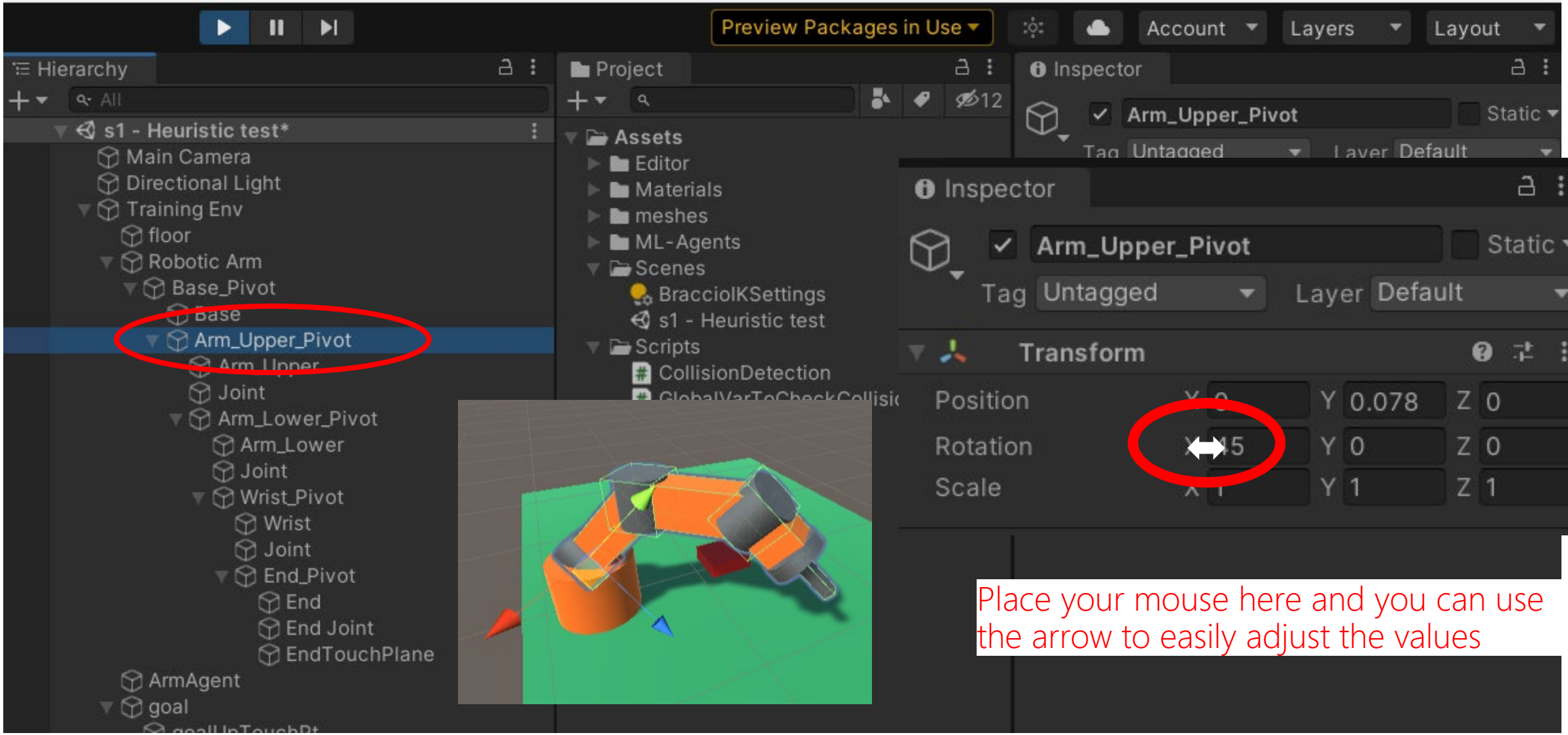
Manually operates the robot arm in VE

Public variables to link agent script with scene objects

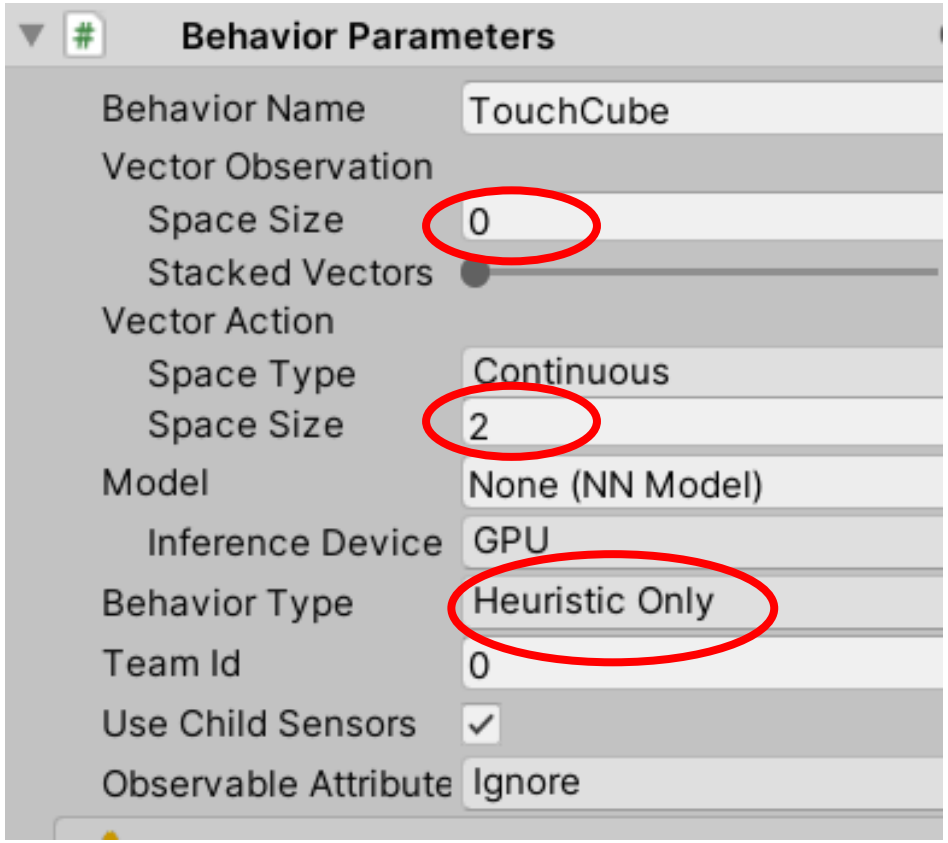
```
public Transform goal, goal2;
public Transform BasePivot, UpperPivot, LowerPivot, WristPivot, EndPivot;
public Transform EndTouchPlane, goalUpTouchPt, goalDownTouchPt, goal2UpTouchPt;
int stage = 1;
```



Play and rotate robot arm by changing "Rotation" angle in the Inspector window



Behavior parameters

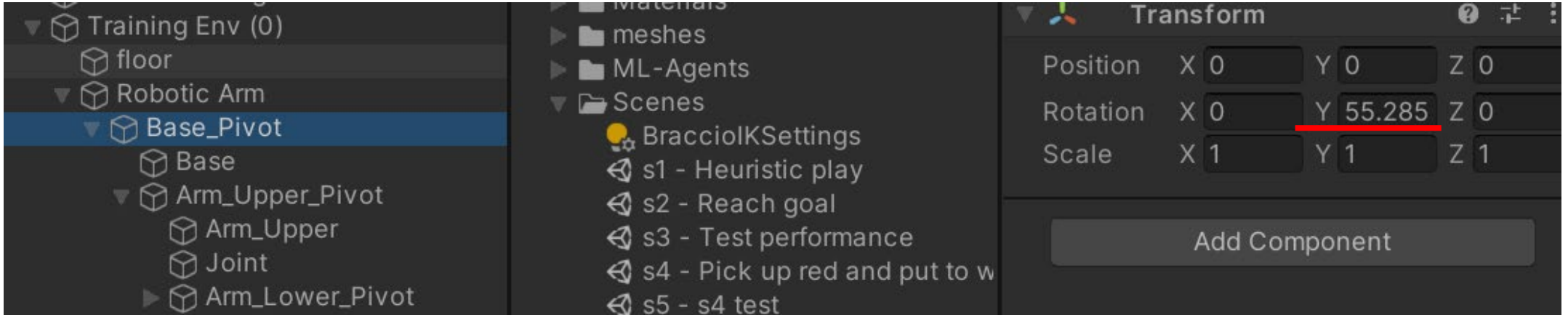


Play and use Up/Down, L/R keys to rotate arm

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

```
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;
}
```

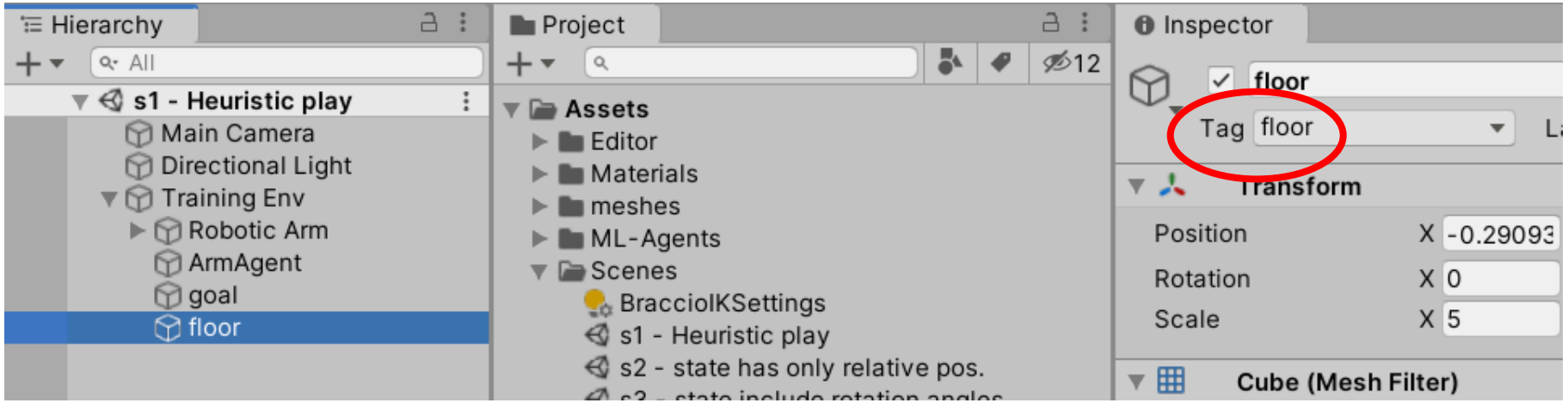
Use OnTriggerEnter/Exist to record collisions between robot arm and other scene objects

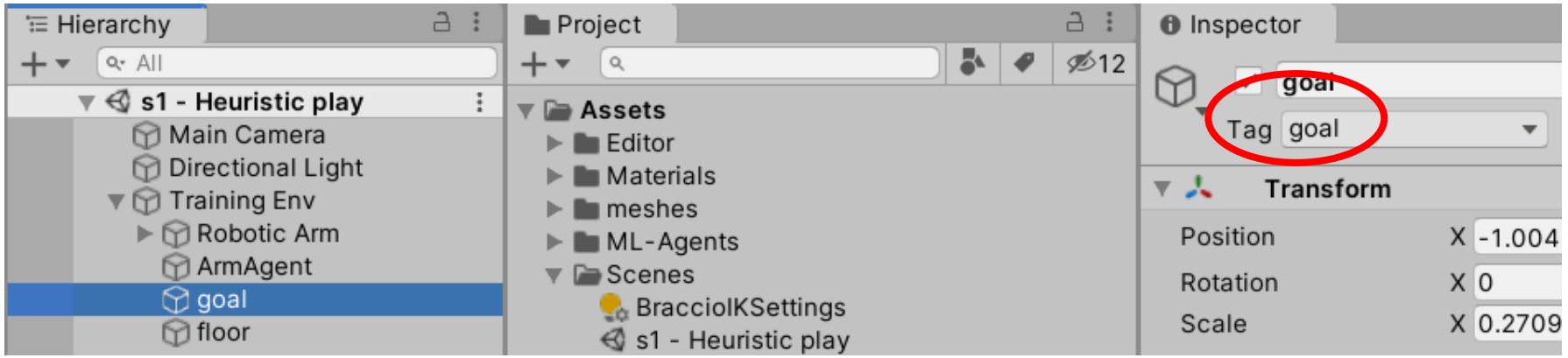
```
public class CollisionDetection : MonoBehaviour
{
    public bool CollisionHappen;
    Unity Message | 0 個參考
    void OnTriggerEnter (Collider other)
    {
        if (other.gameObject.tag == "floor" || o
        {
            CollisionHappen = true;
        }
    }
```

```
void OnTriggerExit(Collider other)
{
    if (other.gameObject.tag == "floor" || o
    {
        CollisionHappen = false;
    }
}
```

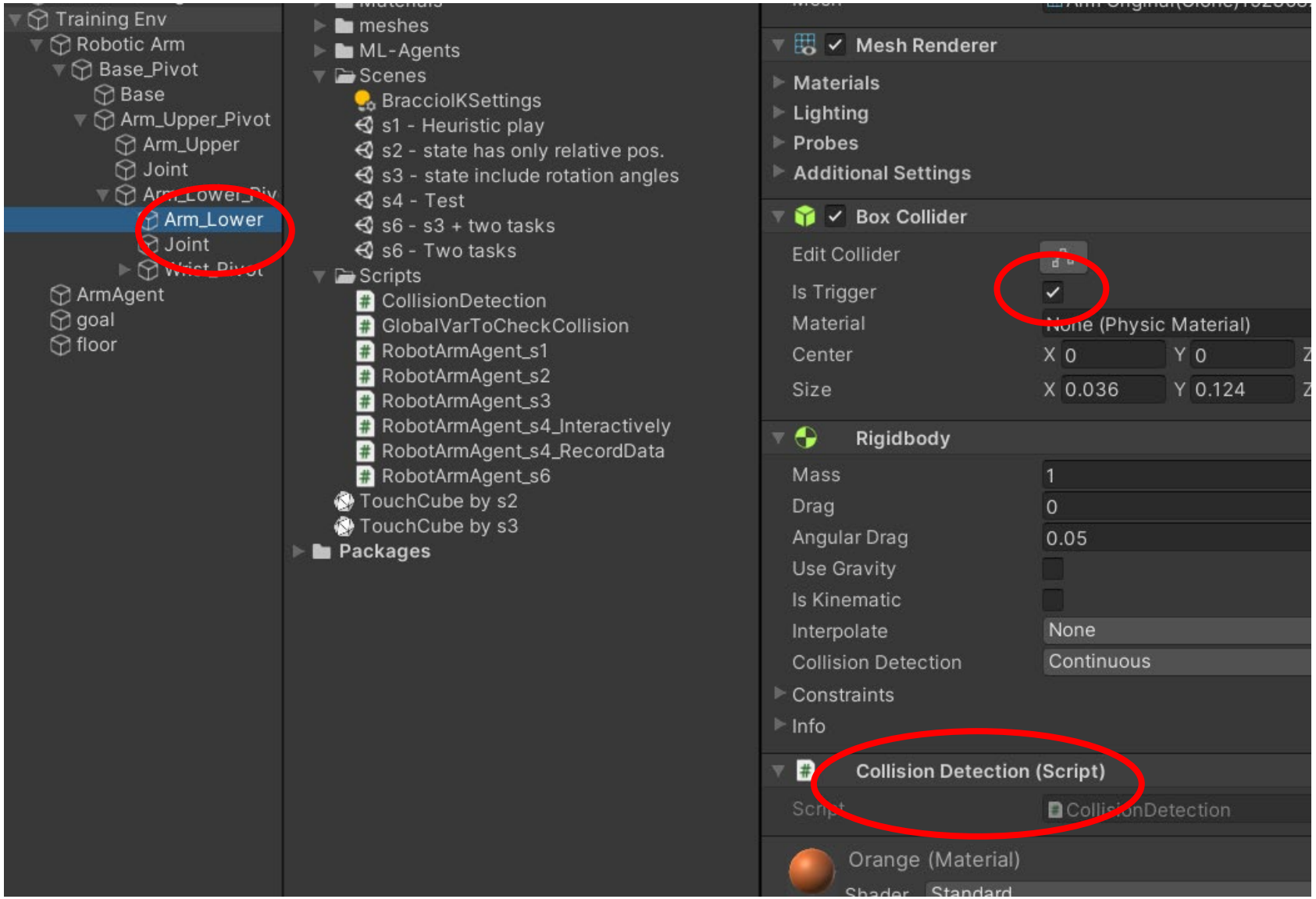
Add tags to floor and goal object

```
if (other.gameObject.tag == "floor" ||
{
    CollisionHappen = true;
```

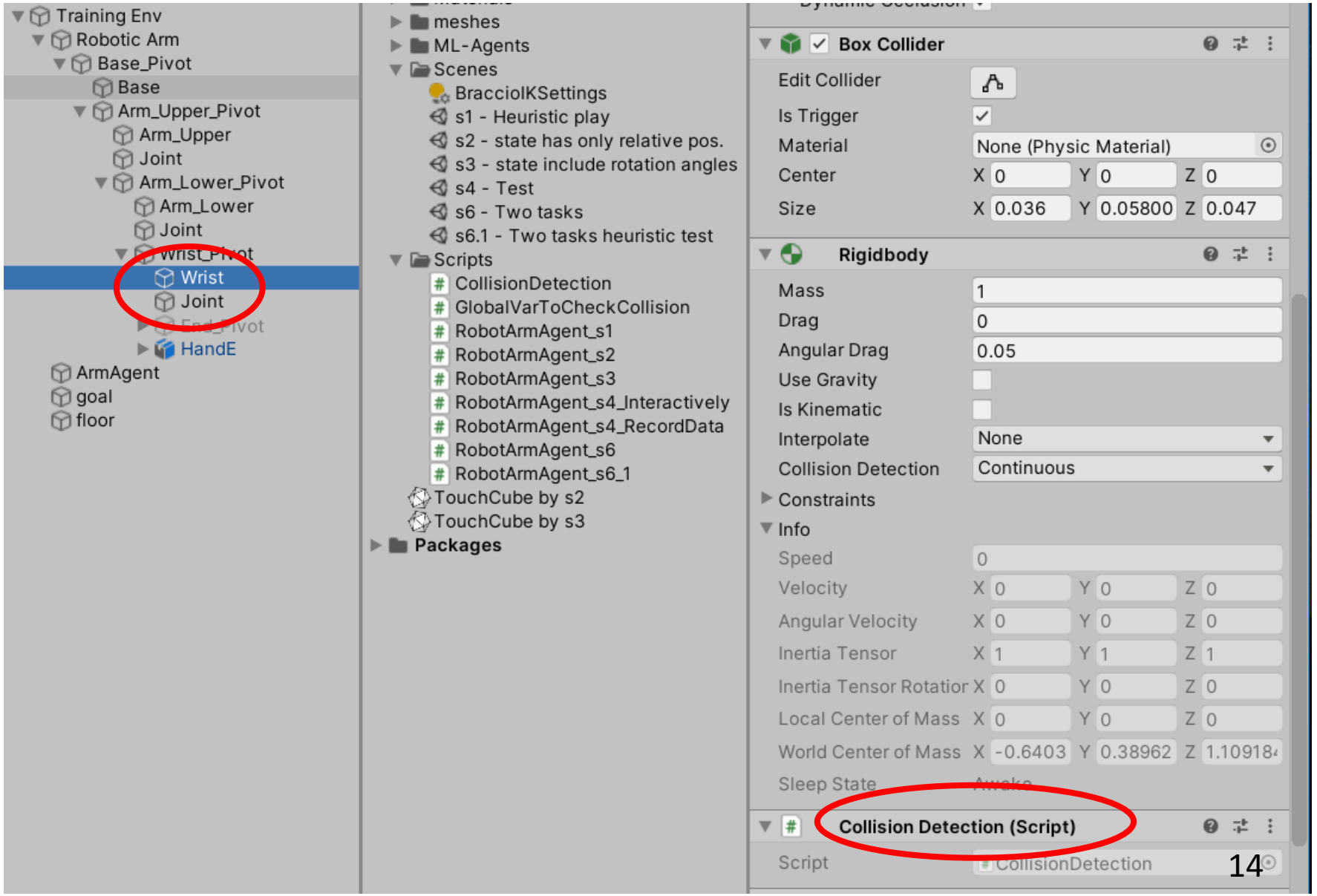




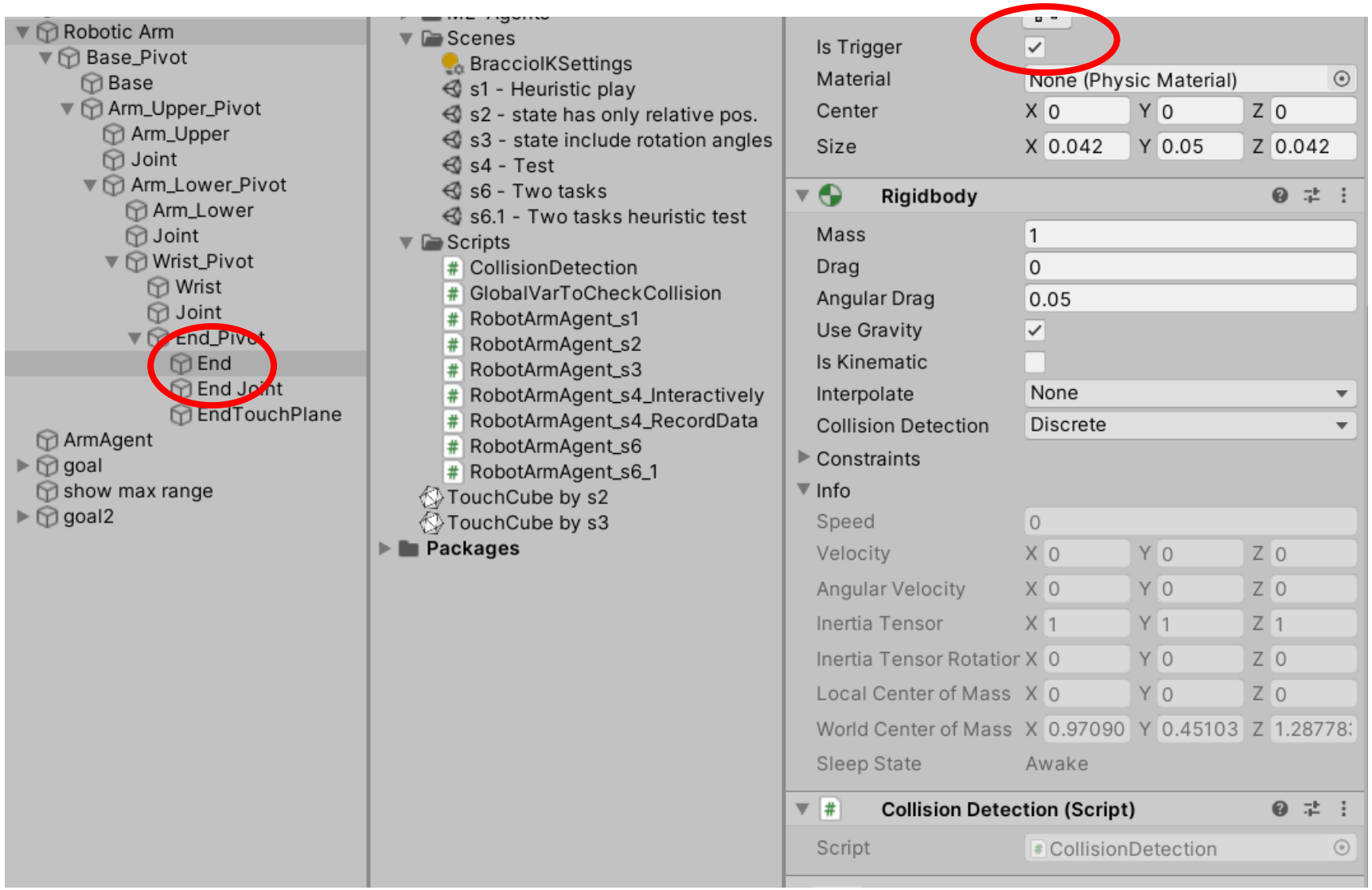
Add trigger collider, Rigid body, and collision detection script to Lower Arm



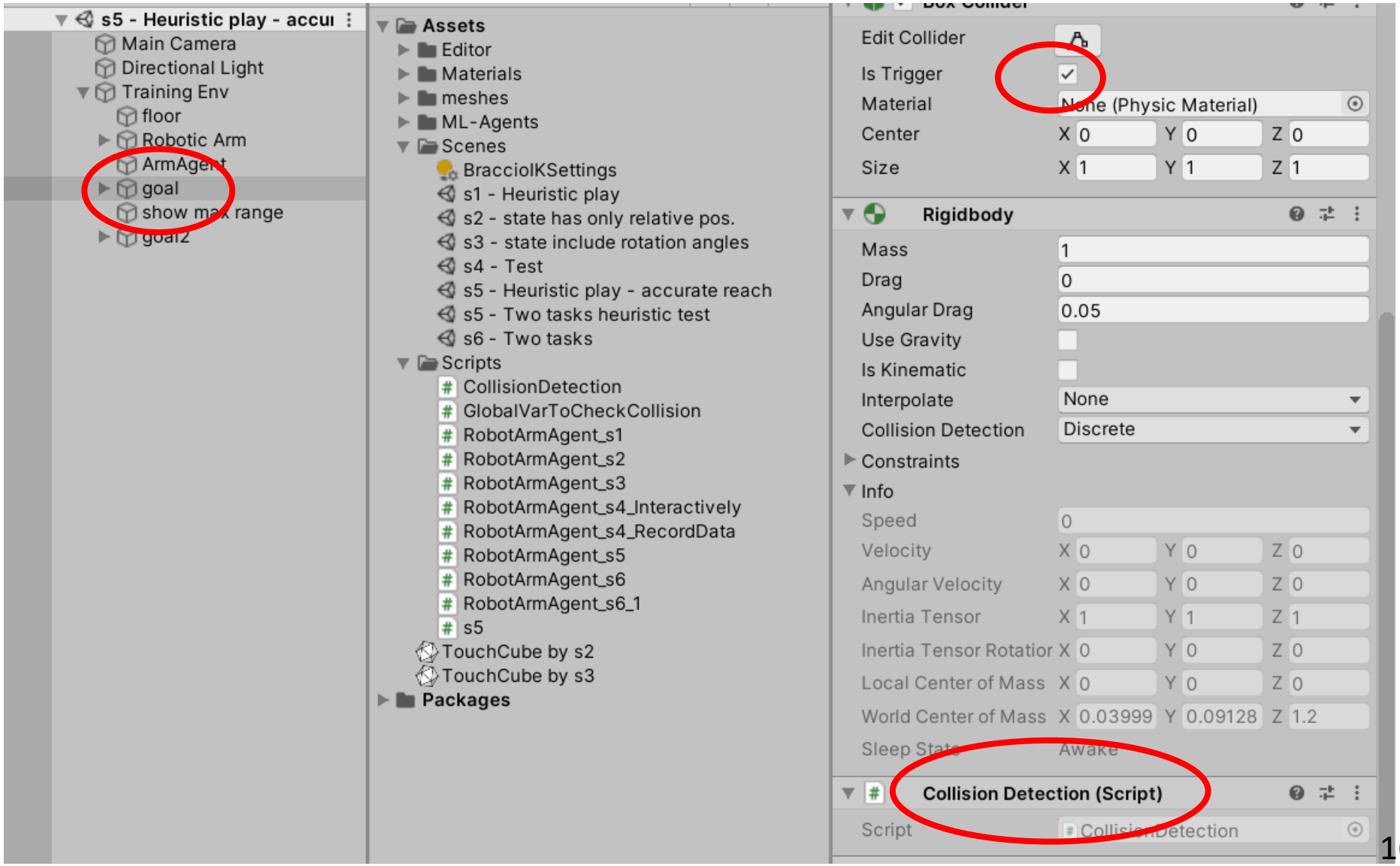
Add trigger collider, Rigid body, and collision detection script to Wrist



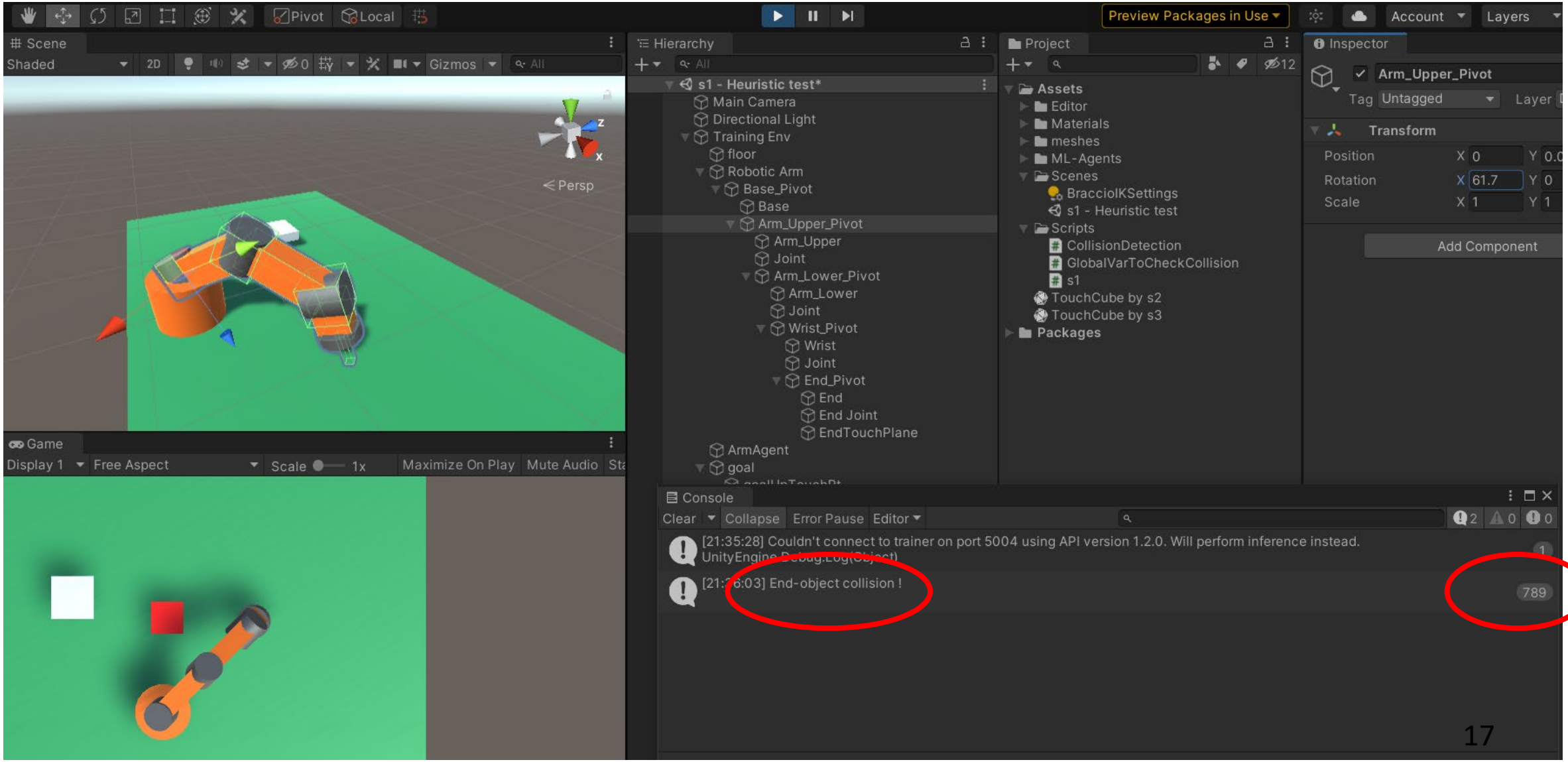
Add trigger collider, Rigid body, and collision detection script to Robot End



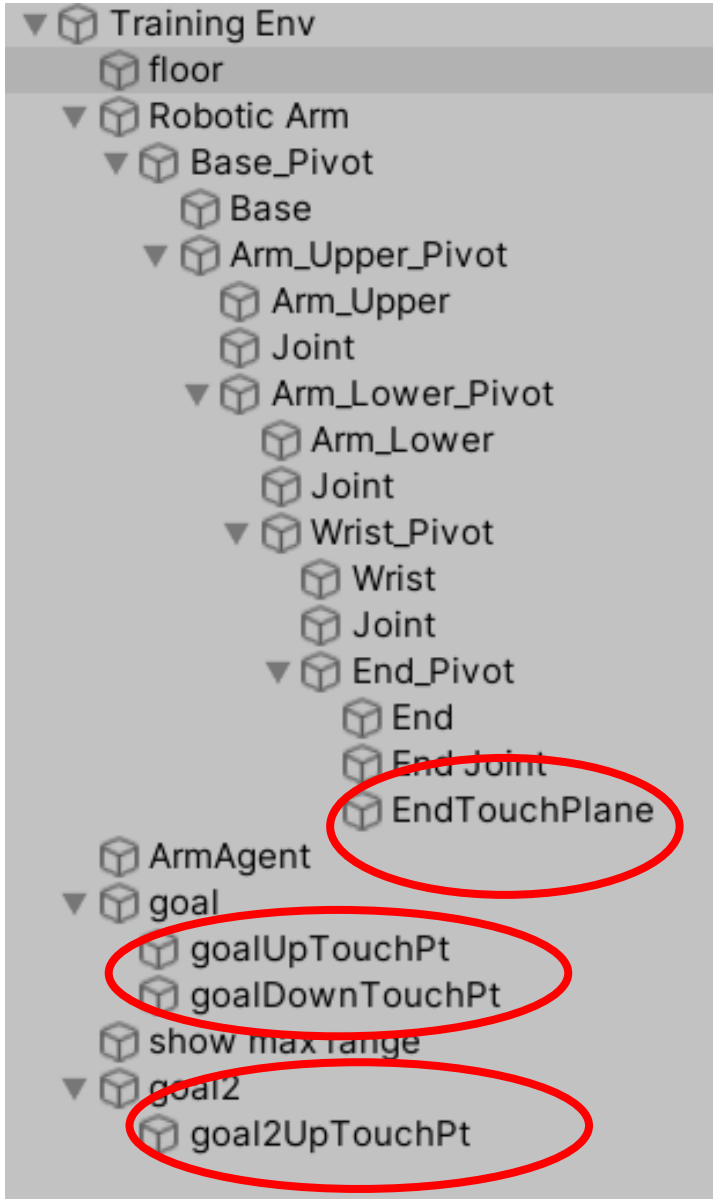
Add trigger collider, Rigid body, and collision detection script to goal object

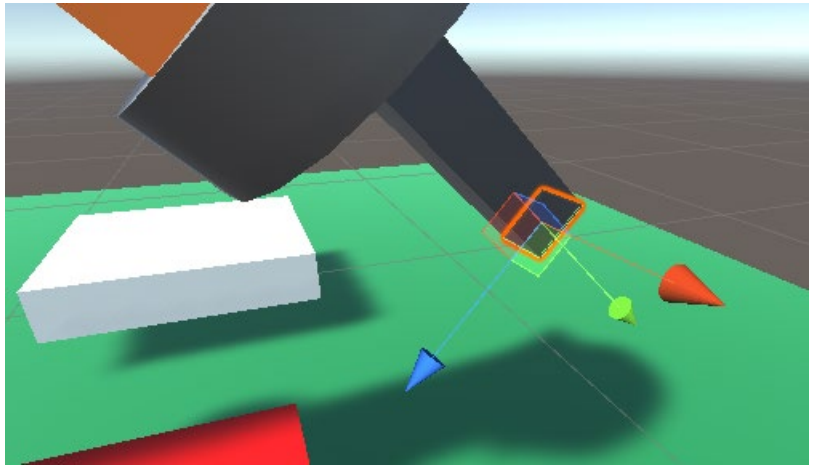


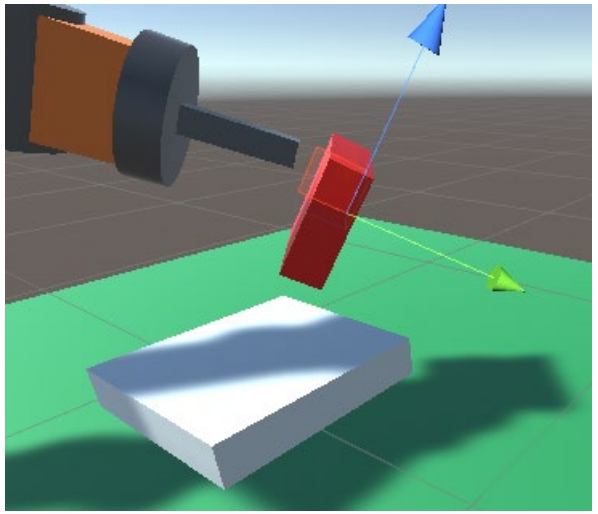
Test collision

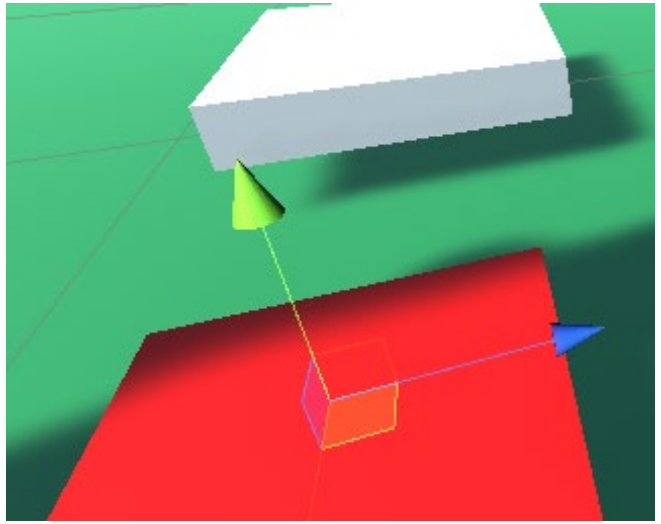


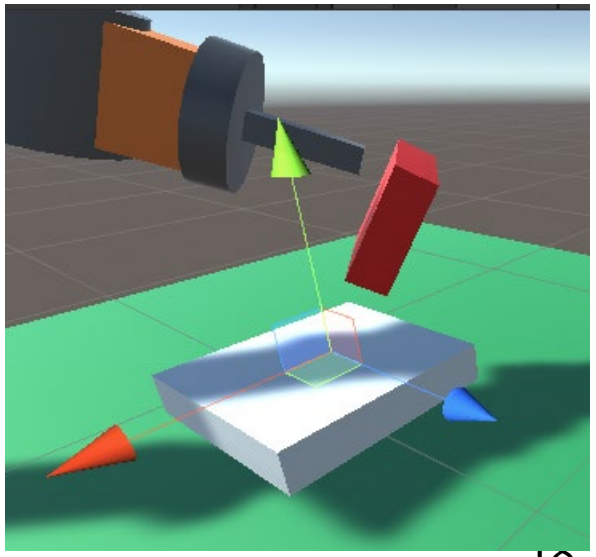
Assign points for reach-detection







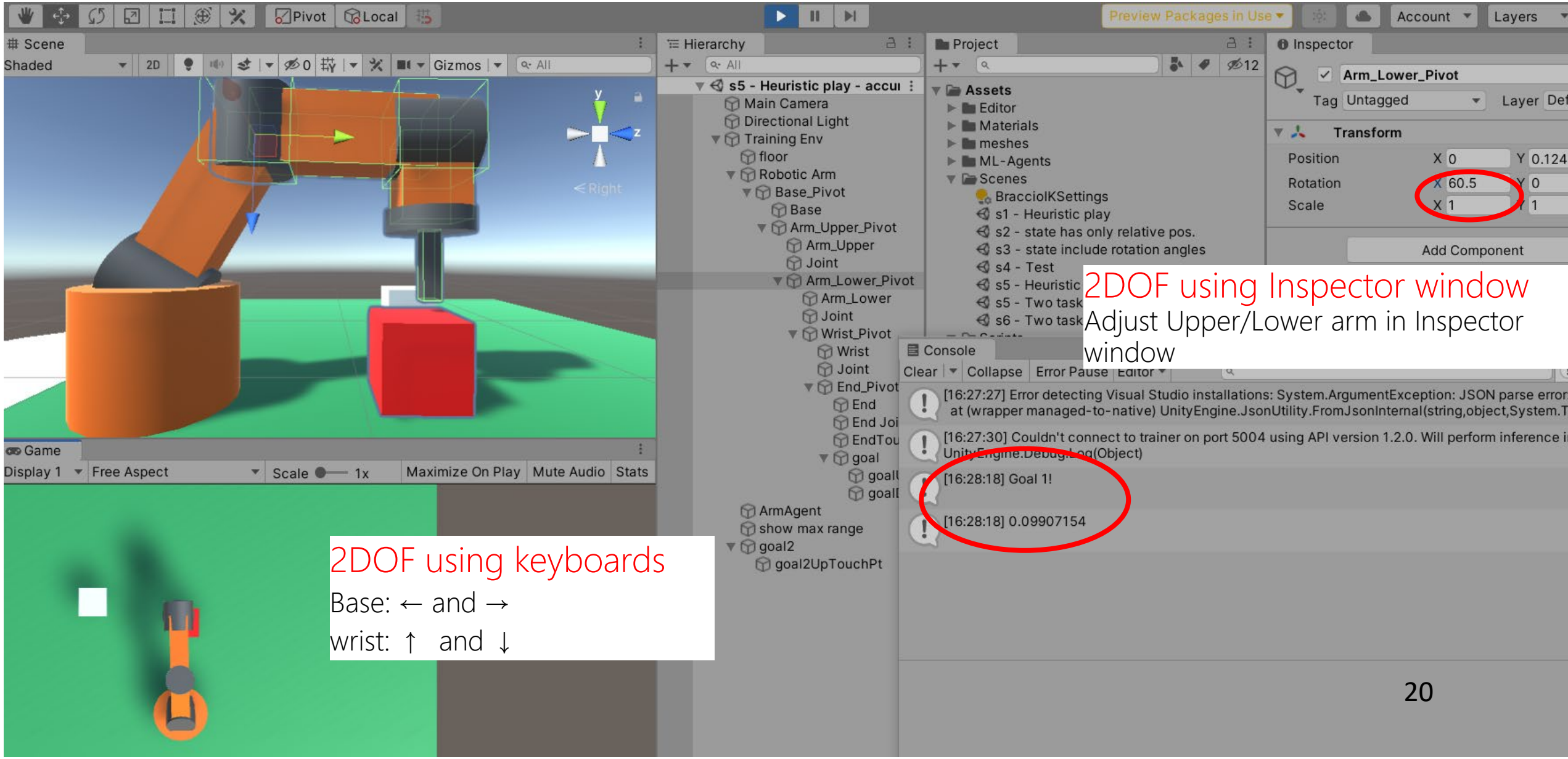




Point-based touch detection

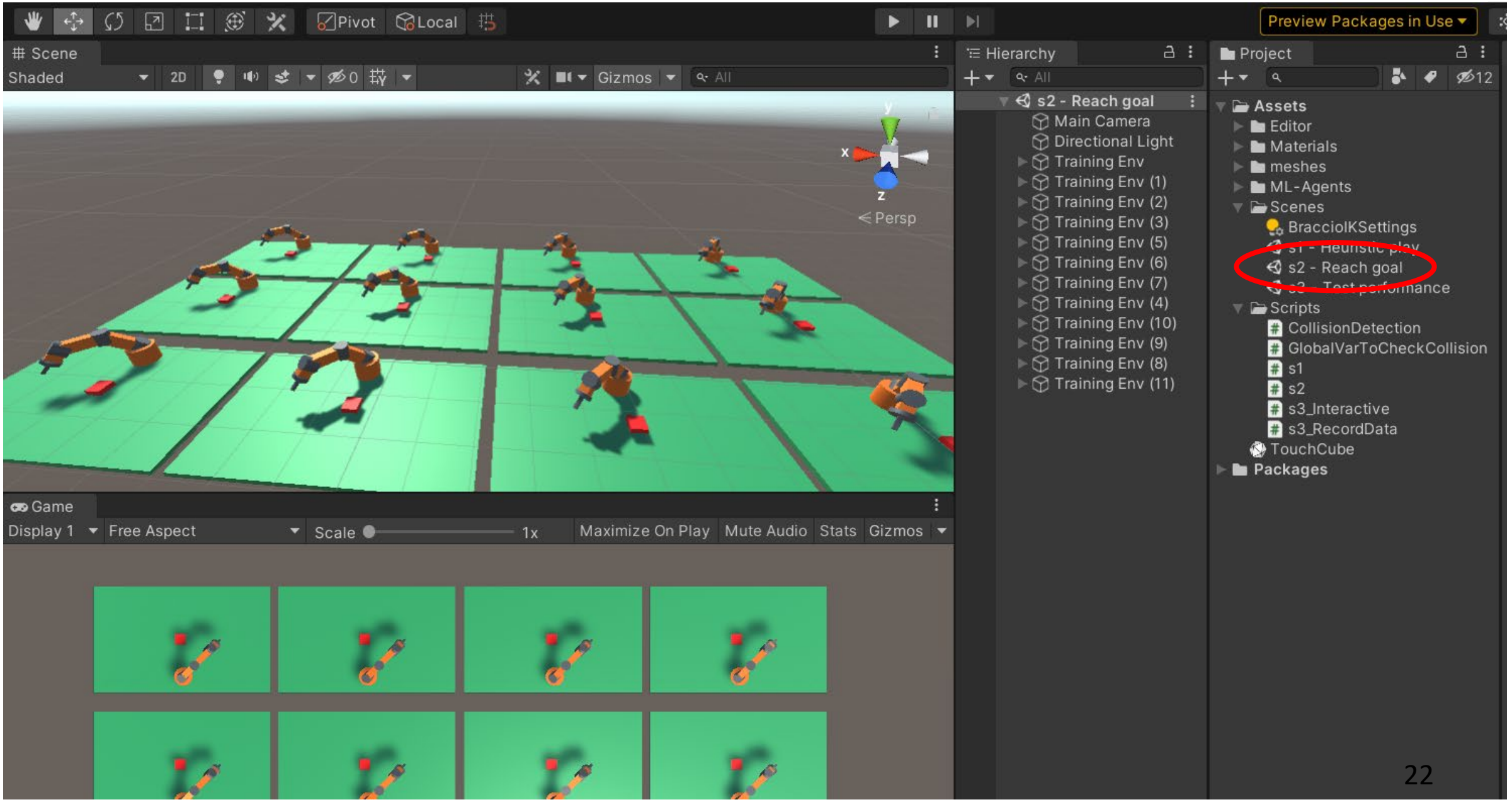
```
bool PointTouch(Transform pt1, Transform pt2, float threshold)
{
    //string msg;
    float dx, dy, dz;

    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 touch goal (avoid collision!)



(1) Reach goal

4. Open "s2 – Reach goal"

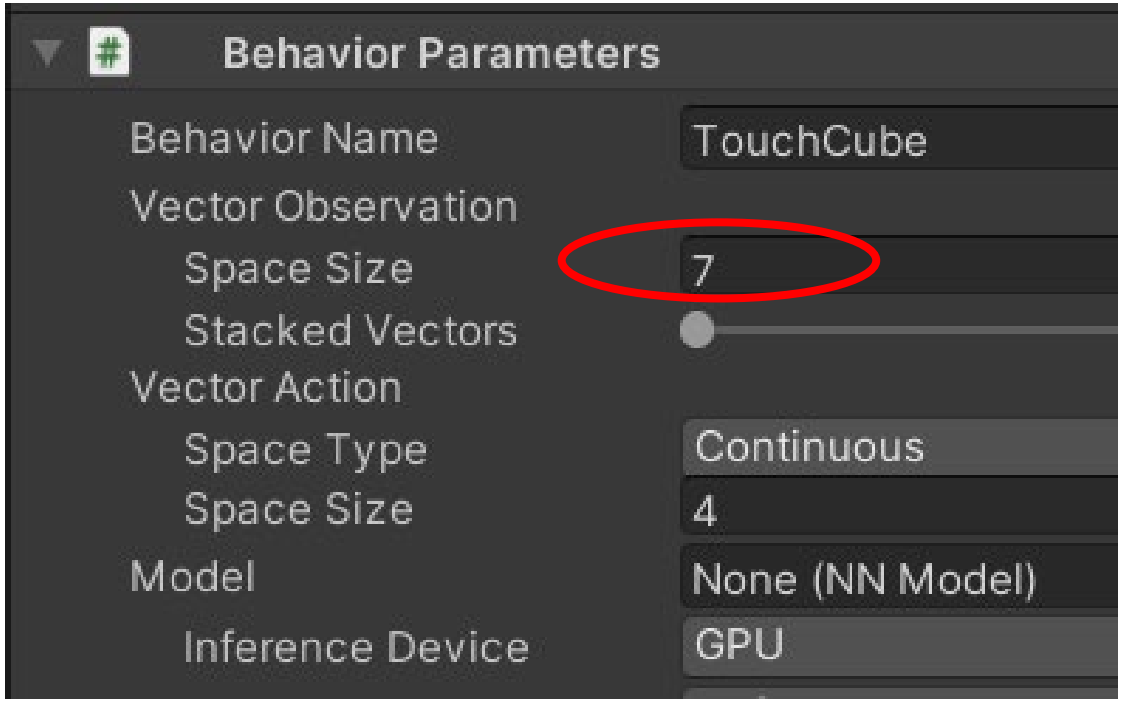


Use polar system to randomly place goals

```
float radius = UnityEngine.Random.Range(radiusMin, radiusMax);
float theta = (UnityEngine.Random.Range(thetaMin, thetaMax) / 180.0f) * Mathf.PI;
  to radians
float x = radius * Mathf.Sin(theta);
float z = radius * Mathf.Cos(theta);
return new Vector3(x, y, z);
```

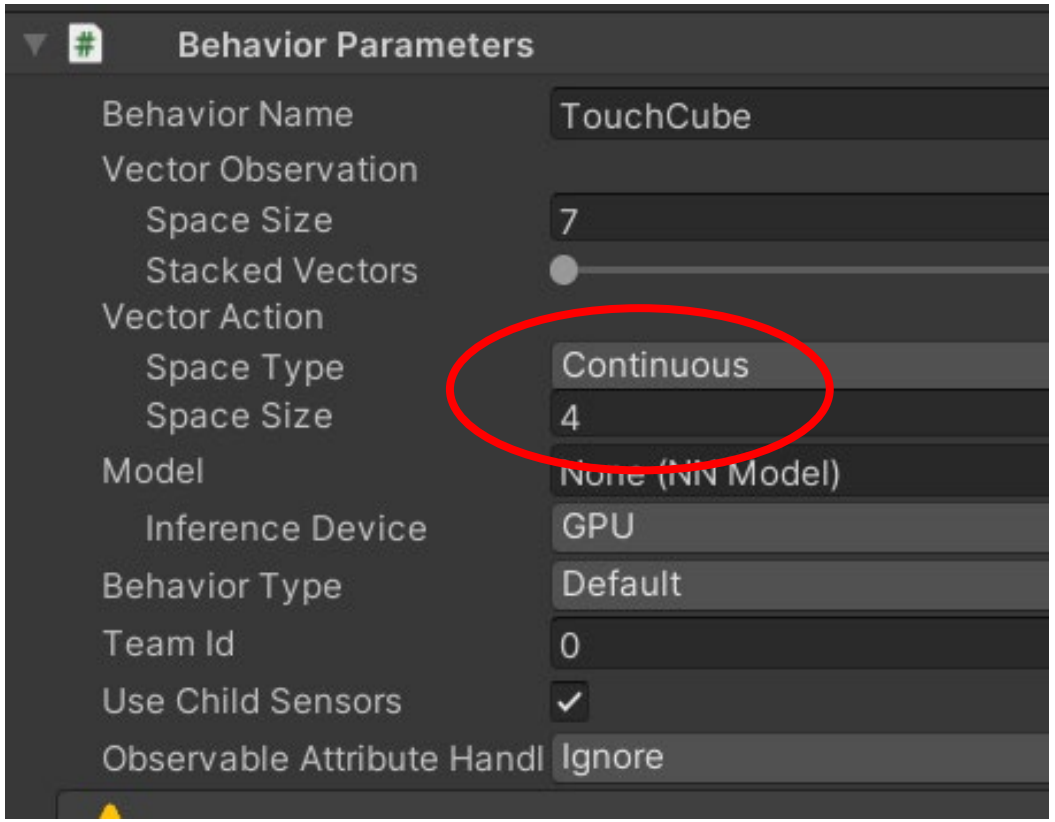
State has 7 variables

```
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);
```



Reward

Punish every step to avoid 耍廢

```
float speed = 1.0f;
AddReward(-0.005f);
string msg;
```

```
if (!Rotation_in_range())
{
    /*
    msg = System.DateTime.Now.ToShortTimeString();
    msg = msg + trainingVE.name + "Angle out of range error ! \n";
    print(msg); */
    AddReward(-5.0f);
    EndEpisode();
}

if (LowerArmObj.GetComponent<CollisionDetection>().CollisionHappen ||
        WristObj.GetComponent<CollisionDetection>().CollisionHappen ||
        EndObj.GetComponent<CollisionDetection>().CollisionHappen)
{
    /*
    msg = System.DateTime.Now.ToShortTimeString();
    msg = msg + trainingVE.name + MyGlobalVar.LowerArmCollisionHappens.
    msg = msg + MyGlobalVar.WristCollisionHappens.ToString() + ", " +
      MyGlobalVar.EndCollisionHappens.ToString() + ", " +
      MyGlobalVar.goalCollisionHappens.ToString()+"\n";
    print(msg);*/
    AddReward(-5.0f);
    EndEpisode();
```

HW

What will happen if do not end episode ?

Reward when reach goal

HW

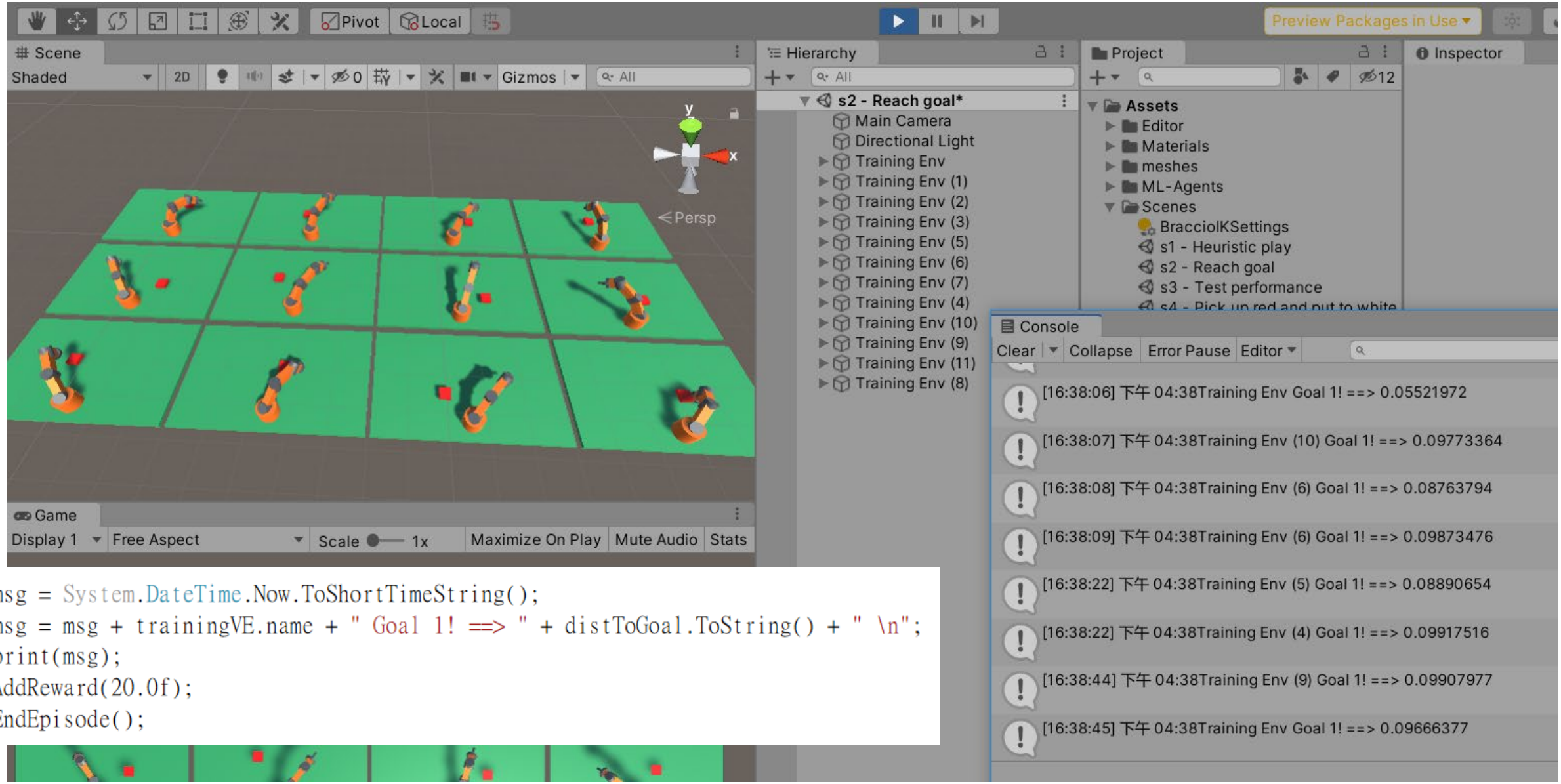
- 0.5 will succeed, but the behavior is too rough
- 0.05 is too difficult and training will fail
- In your HW, try 0.25 or 0.1?

```
if(PointTouch(EndTouchPlane, goalUpTouchPt, 0.5f))
{
    msg = System.DateTime.Now.ToShortTimeString();
    msg = msg + trainingVE.name + " Goal 1! \n";
    print(msg);
    AddReward(20.0f);
    EndEpisode();
}
```

Training configuration file

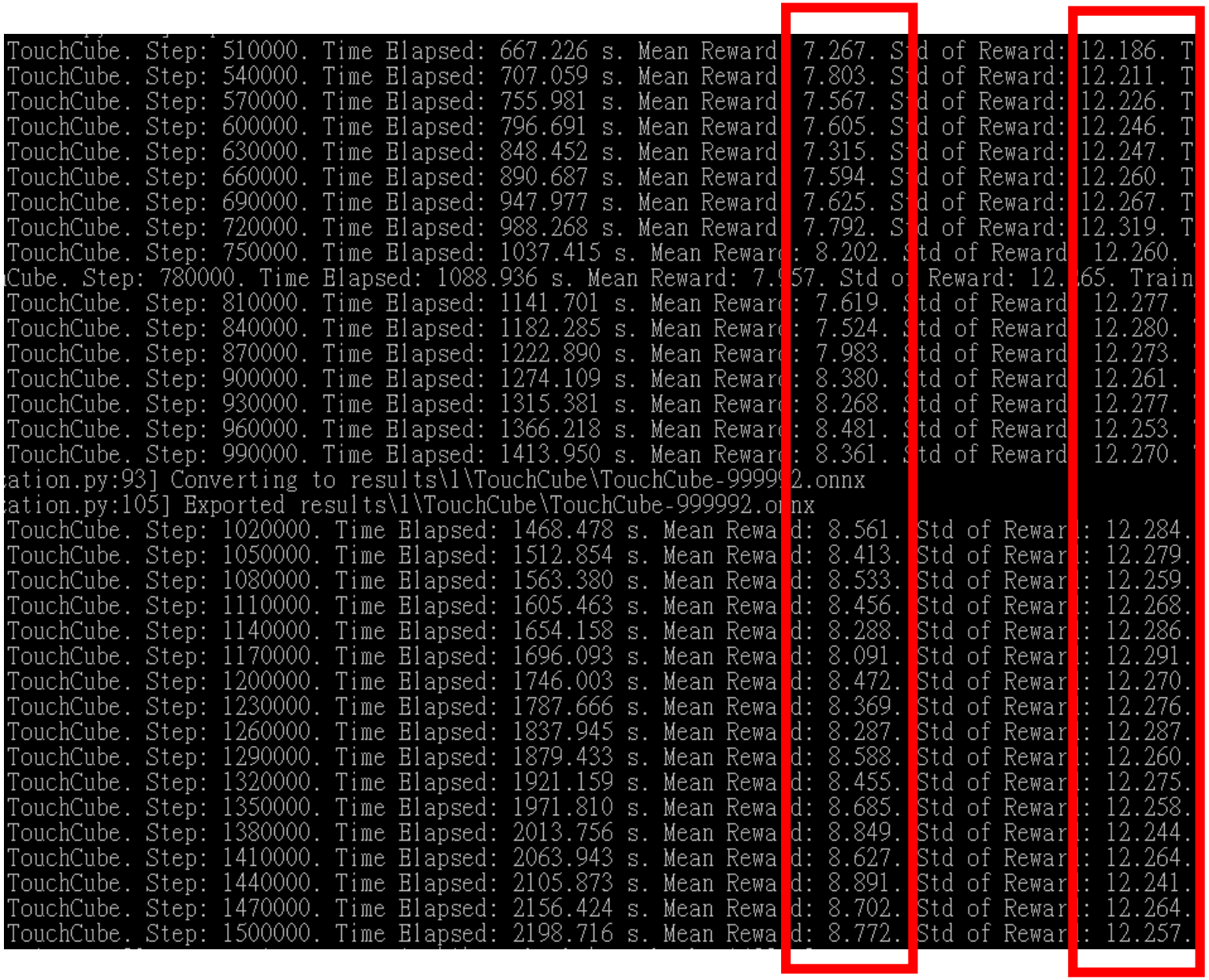
```
TouchCube:
  trainer_type: ppo
  hyperparameters:
    batch_size: 2048
    buffer_size: 20480
    learning_rate: 0.0003
    beta: 0.001
    epsilon: 0.2
    lambd: 0.95
    num_epoch: 3
    learning_rate_schedule
  network_settings:
    normalize: true
    hidden_units: 512
    num_layers: 3
    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
```

Examine environment that reaches goal



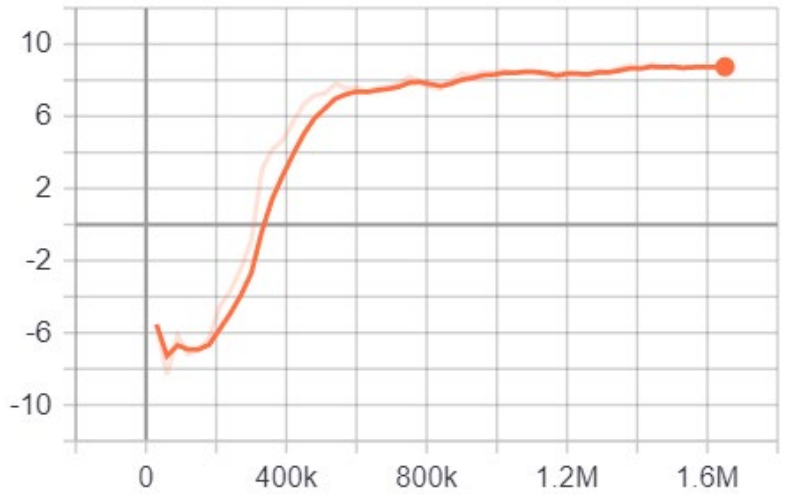

```
msg = System.DateTime.Now.ToShortTimeString();
msg = msg + trainingVE.name + " Goal 1! ==> " + distToGoal.ToString() + " \n";
print(msg);
AddReward(20.0f);
EndEpisode();
```

Results after 1.7M steps, looks promising

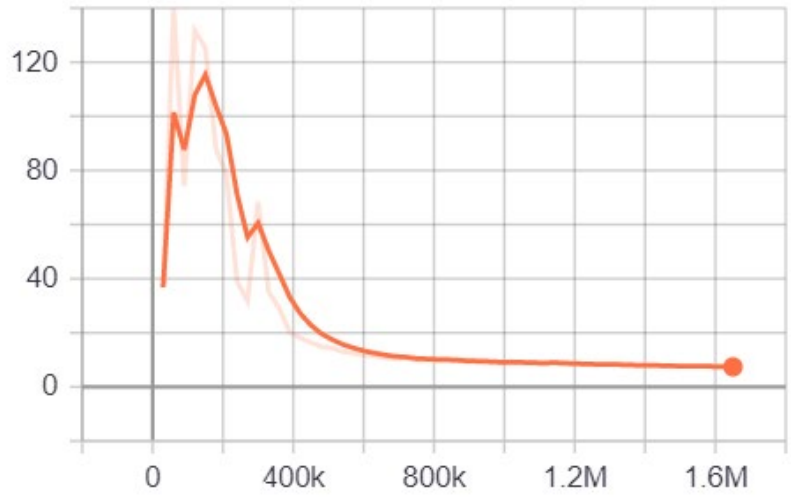


Results after 1.7M steps, looks promising

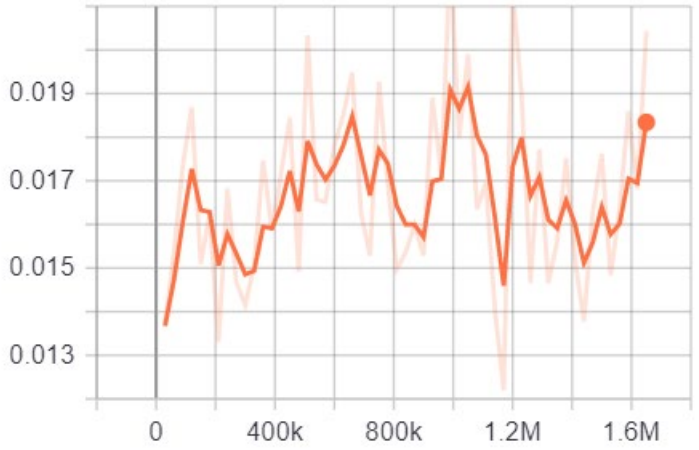
Cumulative Reward
tag: Environment/Cumulative Reward



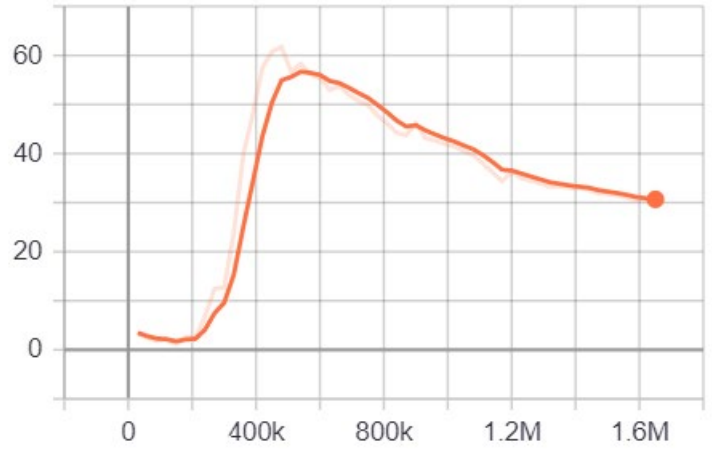
Episode Length
tag: Environment/Episode Length



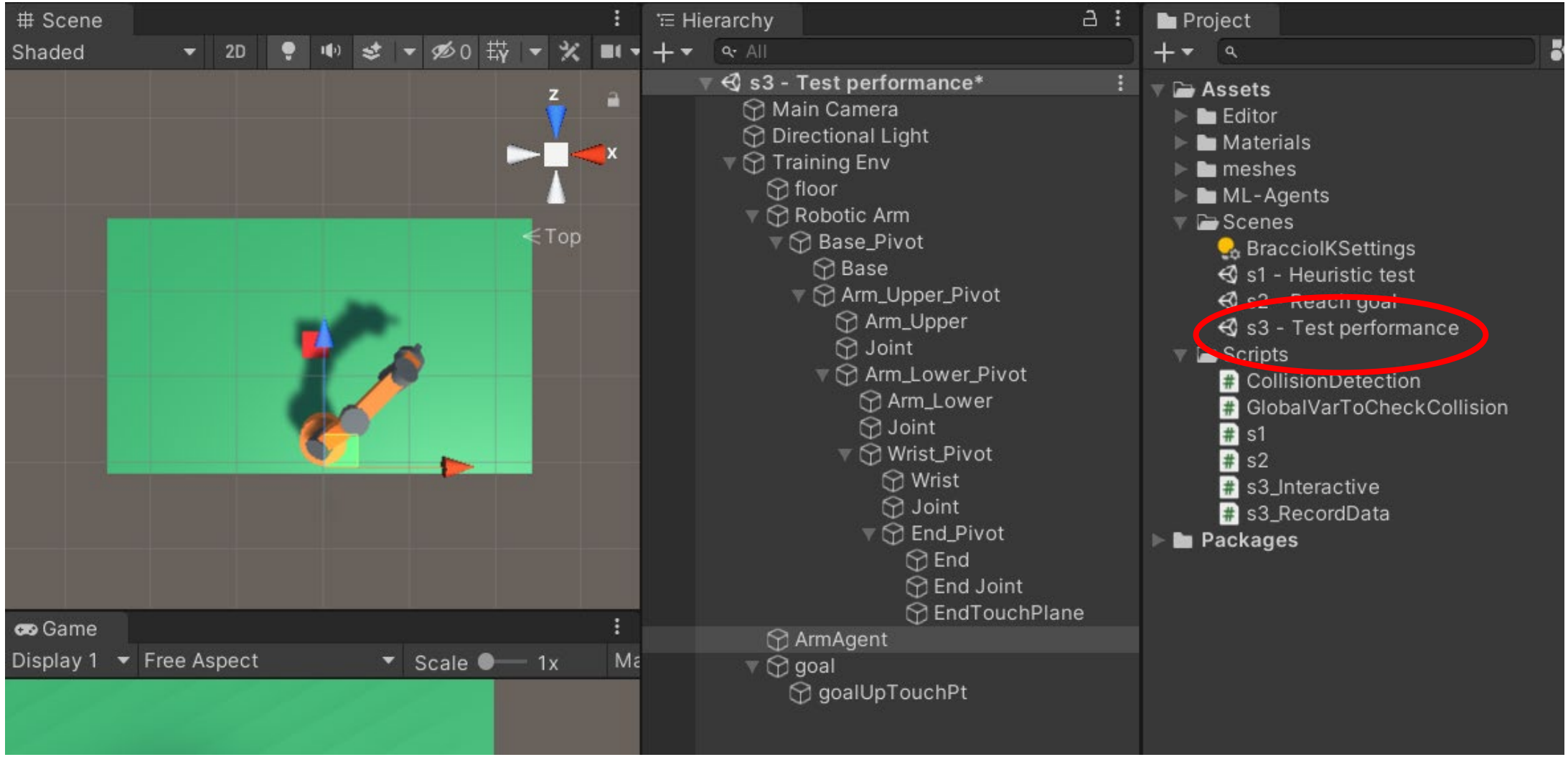
Policy Loss
tag: Losses/Policy Loss



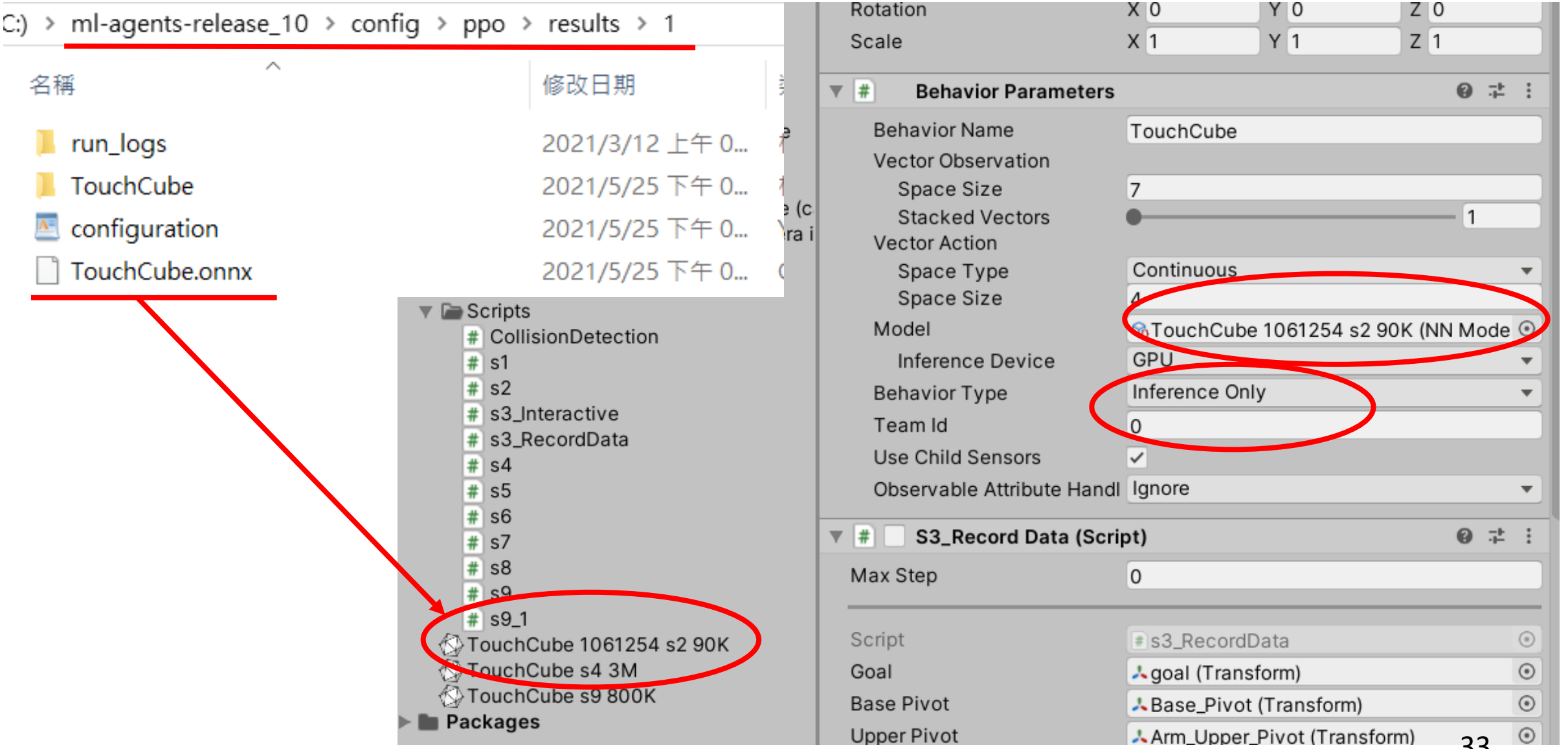
Value Loss
tag: Losses/Value Loss



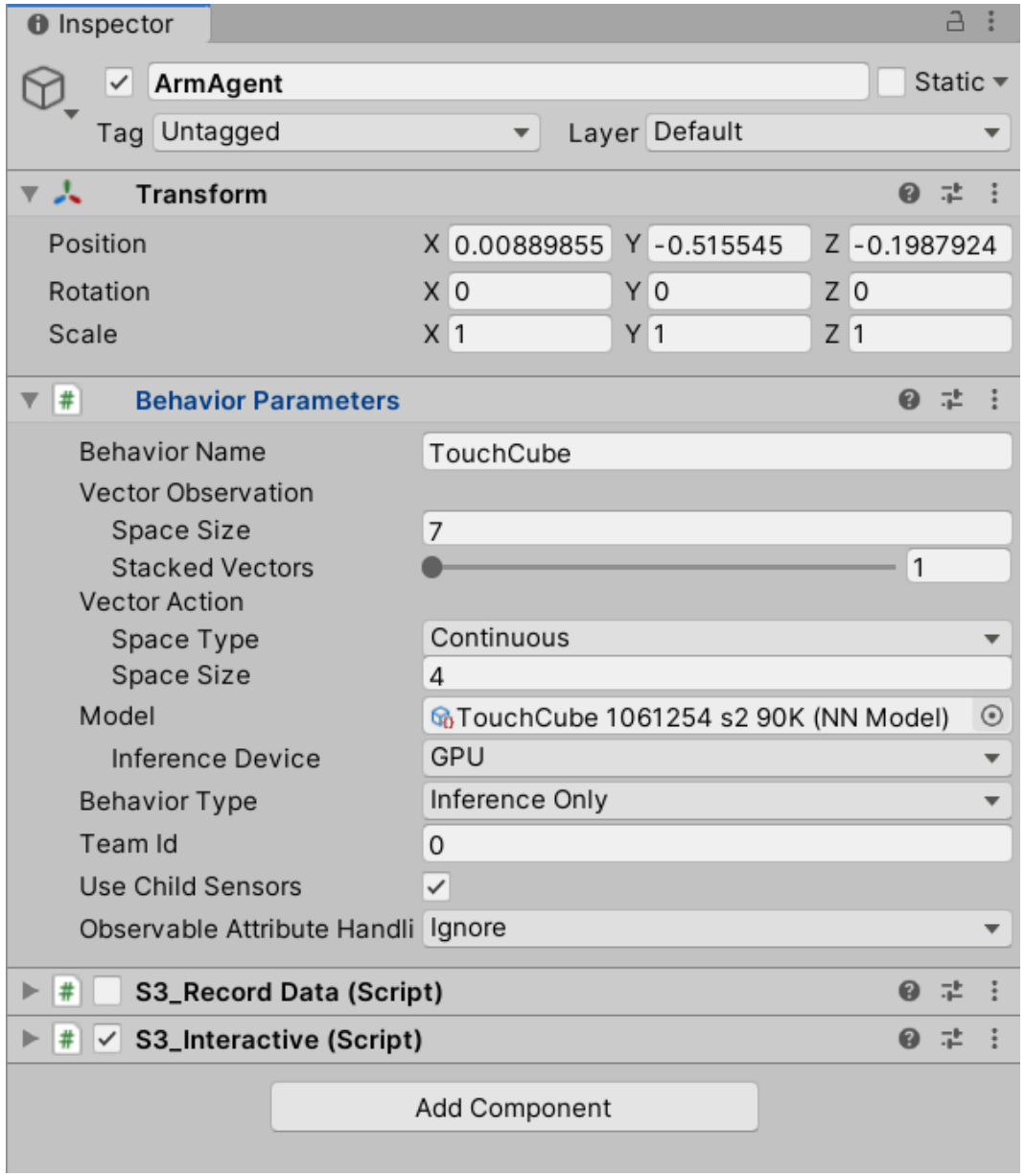
Open "s3 – Test performance"



Assign trained NN



Remove decision requester from Behavior parameter



Assign decision requester in Update function

Record data test

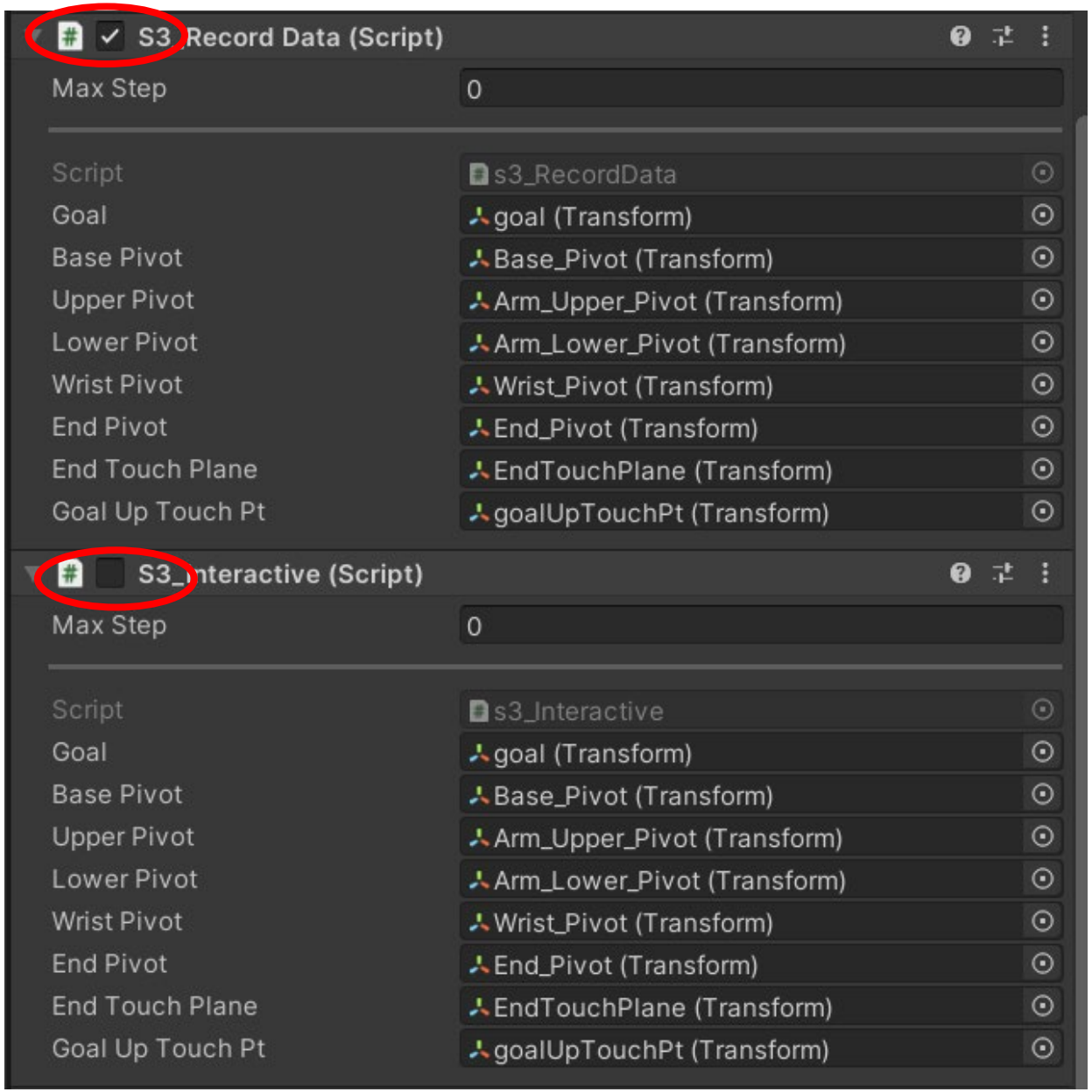
```
void Update()
{
    if (NoTest <= TotalTests)
    {
        if (!PointTouch(EndTouchPlane, goalUpTouchPt, 0.3f))
        {
            RequestDecision();
        }
        else //reach goal
        {
            string s = "Finish No " + NoTest.ToString();
            writer.WriteLine(s);
            NoTest = NoTest + 1;
            EndEpisode(); // Finish this test and start next
        }
    }
    // else NoTest already larger than TotalTests, do nothing
      close the game
}
```

Interactive test

```
void Update()
{
    if (!PointTouch(EndTouchPlane, goalUpTouchPt, 0.3f))
    {
        RequestDecision();
    }
}
```

2 types of tests – (1) data recording

Uncheck interactive test



(1) data recording

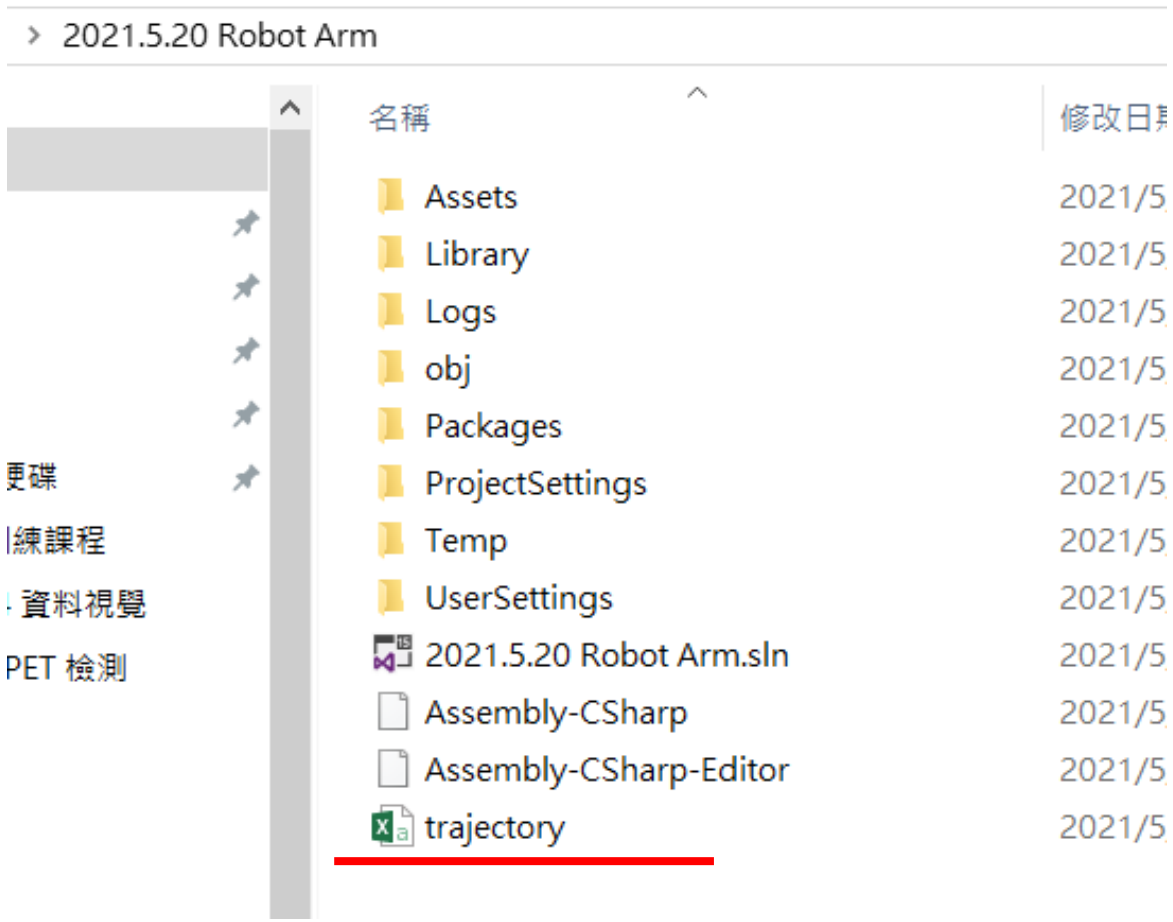
```
void Start()
{
    goalOriginalPos = goal.transform.position;
    BasePivotRoation = BasePivot.rotation;
    UpperPivotRotation = UpperPivot.rotation;
    LowerPivotRotation = LowerPivot.rotation;
    WristPivotRotation = WristPivot.rotation;
    GoalRotation = goal.rotation;

    TotalTests = 6; // test the NN model performance for N times
    NoTest = 1;
    filePath = "trajectory.csv";
    writer = new StreamWriter(filePath);
    writer.WriteLine("time, x, y, z, reward");
}
```

(1) data recording

```
void Update()
{
    if (NoTest <= TotalTests)
    {
        if (!PointTouch(EndTouchPlane, goalUpTouchPt, 0.3f))
        {
            RequestDecision();
        }
        else //reach goal
        {
            string s = "Finish No " + NoTest.ToString();
            writer.WriteLine(s);
            NoTest = NoTest + 1;
            EndEpisode(); // Finish this test and start next test
        }
    }
    // else NoTest already larger than TotalTests, do nothing, wait for user to close the game
}
```

Should be equal to or larger than the threshold used for training, why ?

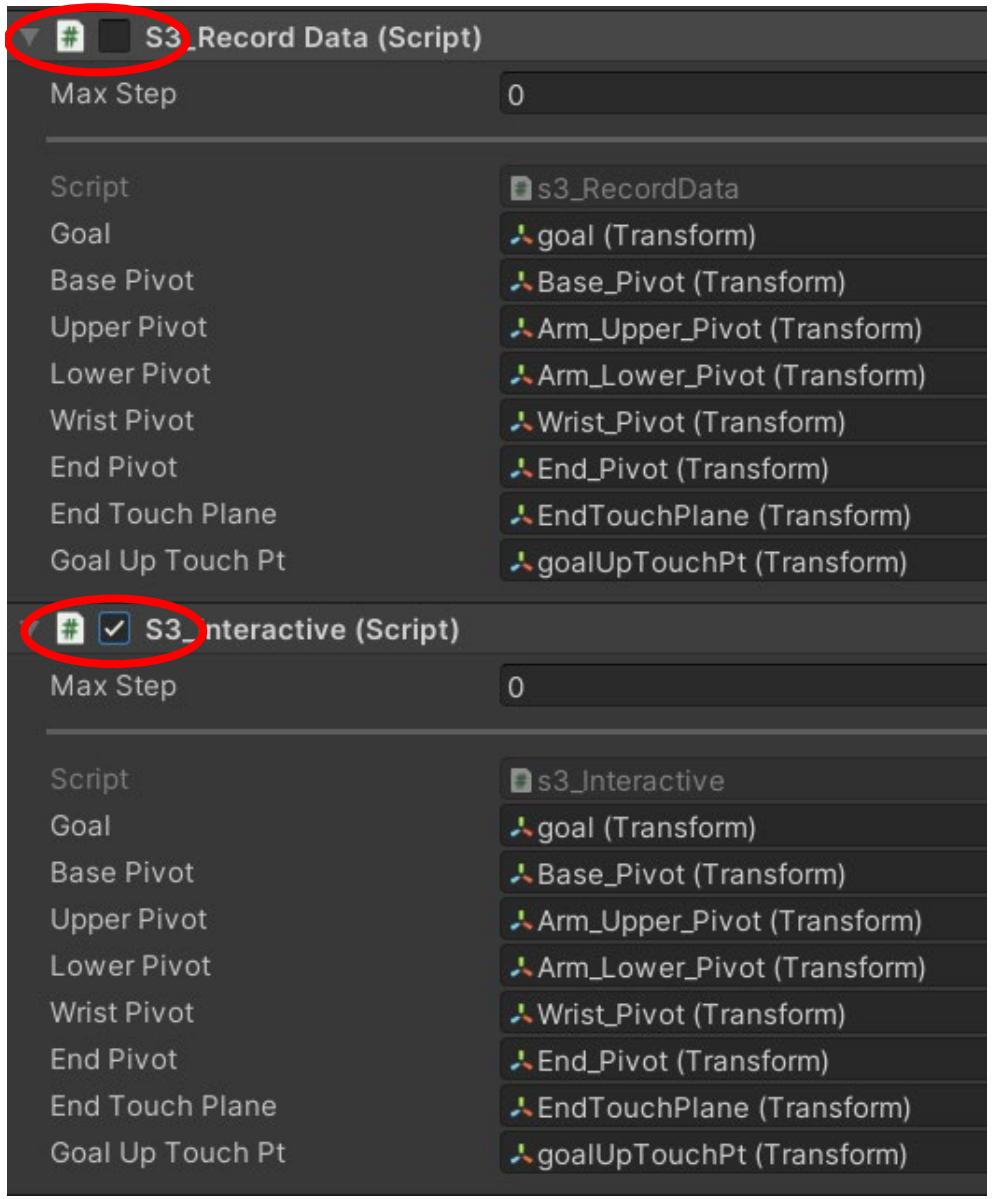
Examine the data file



	A	B	C	D	E	F
	time	x	y	z	reward	
鉢	? 11:33:17	1.038729	0.366309	1.342287	-0.005	
鉢	? 11:33:17	1.046411	0.361419	1.331993	-0.005	
鉢	? 11:33:17	1.052147	0.360803	1.324374	-0.005	
鉢	? 11:33:17	1.060236	0.360774	1.31453	-0.005	
鉢	? 11:33:17	1.066019	0.360026	1.305439	-0.005	
鉢	? 11:33:17	1.072721	0.362663	1.298886	-0.005	
鉢	? 11:33:17	1.079413	0.367661	1.287836	-0.005	
鉢	? 11:33:17	1.088548	0.369737	1.279001	-0.005	
鉢	? 11:33:18	1.095987	0.377341	1.26867	-0.005	
鉢	? 11:33:18	1.098569	0.366547	1.258619	-0.005	
鉢	? 11:33:18	1.102629	0.369689	1.252007	-0.005	
鉢	? 11:33:18	1.109445	0.370761	1.241264	-0.005	
鉢	? 11:33:18	1.114091	0.361341	1.229446	-0.005	
鉢	? 11:33:18	1.119347	0.357583	1.217545	-0.005	
鉢	? 11:33:18	1.126047	0.360188	1.207051	-0.005	
鉢	? 11:33:18	1.133839	0.361506	1.197243	-0.005	
鉢	? 11:33:18	1.138717	0.361016	1.185209	-0.005	
鉢	? 11:33:18	1.142971	0.362387	1.17517	-0.005	
鉢	? 11:33:18	1.146297	0.357001	1.162126	-0.005	
鉢	? 11:33:18	1.151828	0.356633	1.151016	-0.005	
鉢	? 11:33:18	1.155315	0.355944	1.141412	-0.005	

2 types of tests – (2) interactive test

Uncheck data-recording test



(2) interactive test

Should be equal to or larger than the threshold used for training, why ?

```
void Update()
{
    if (!PointTouch(EndTouchPlane, goalUpTouchPt, 0.3f))
    {
        RequestDecision();
    }
}
```