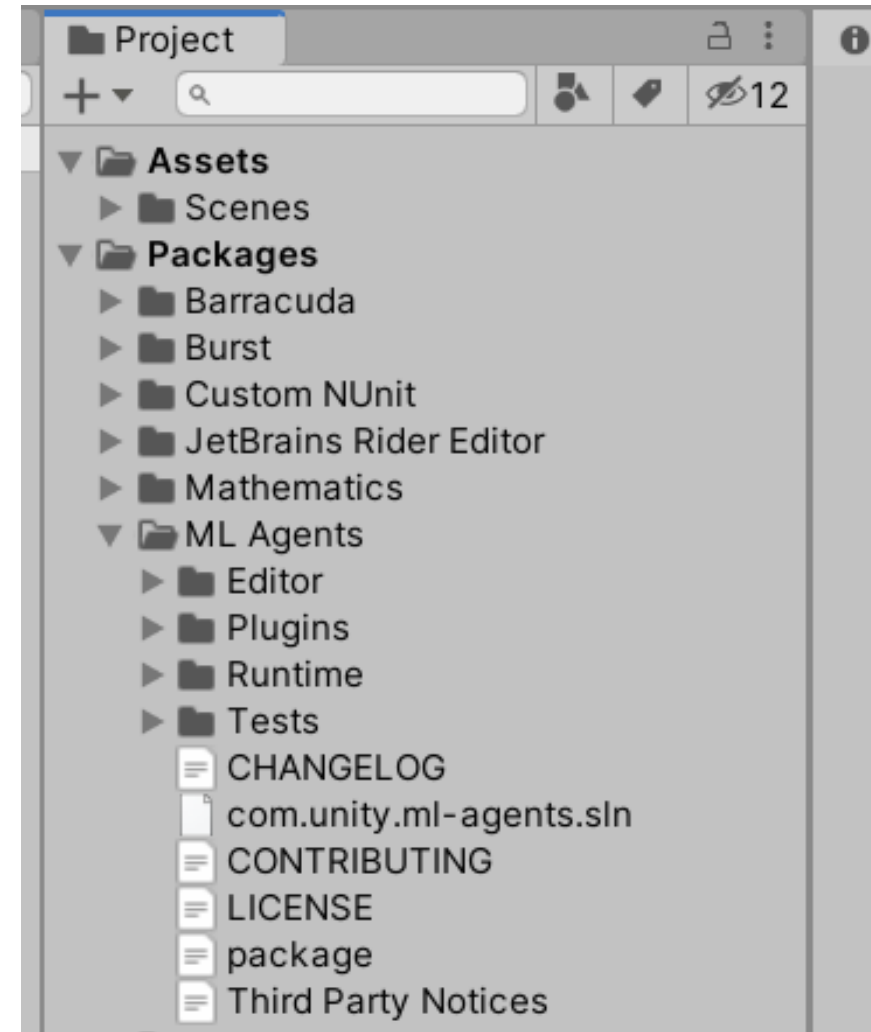
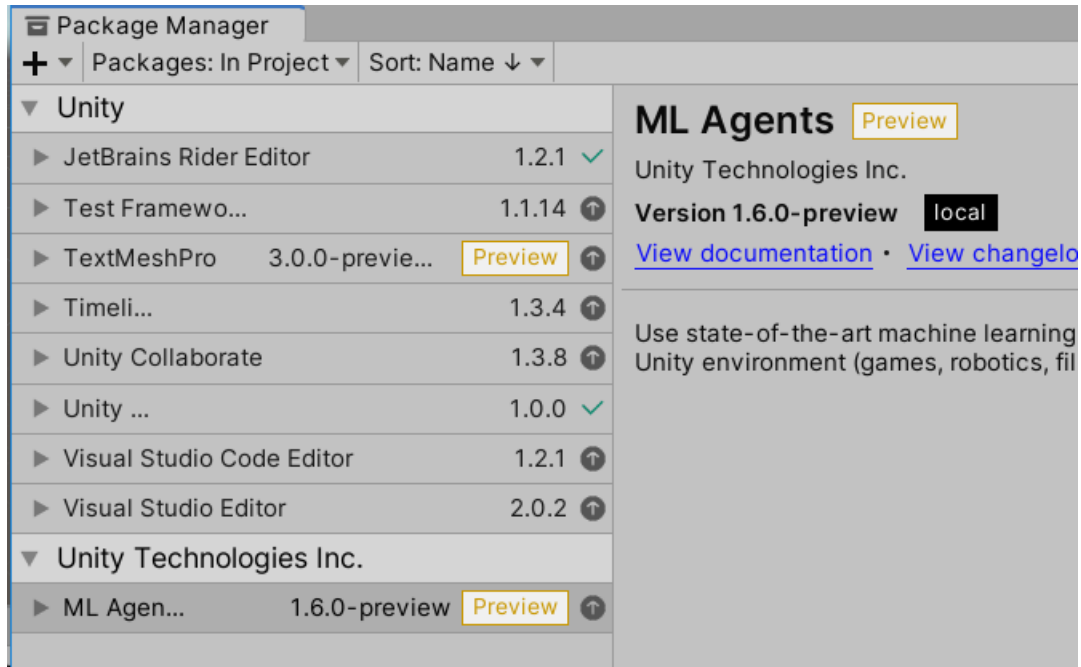
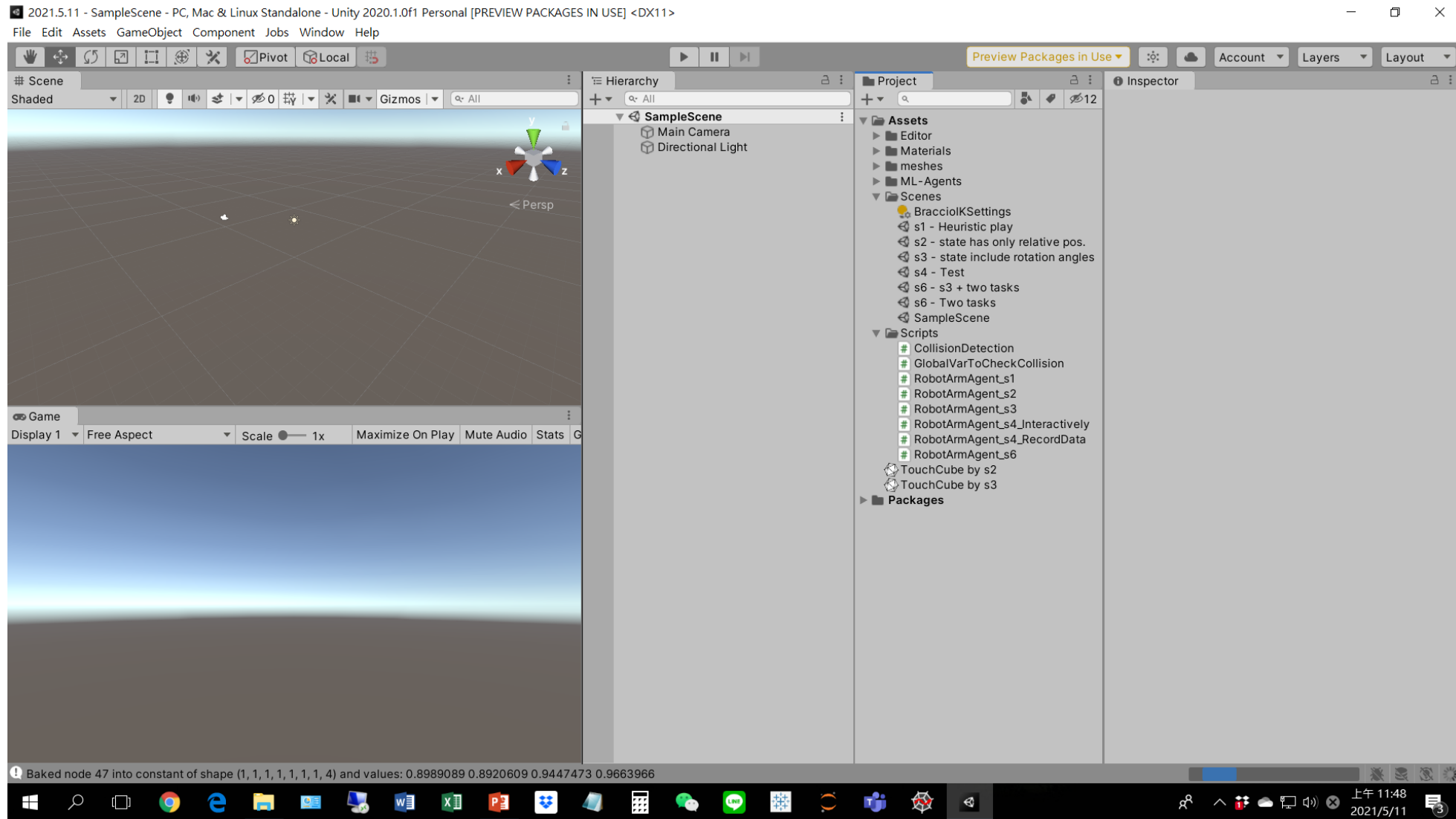


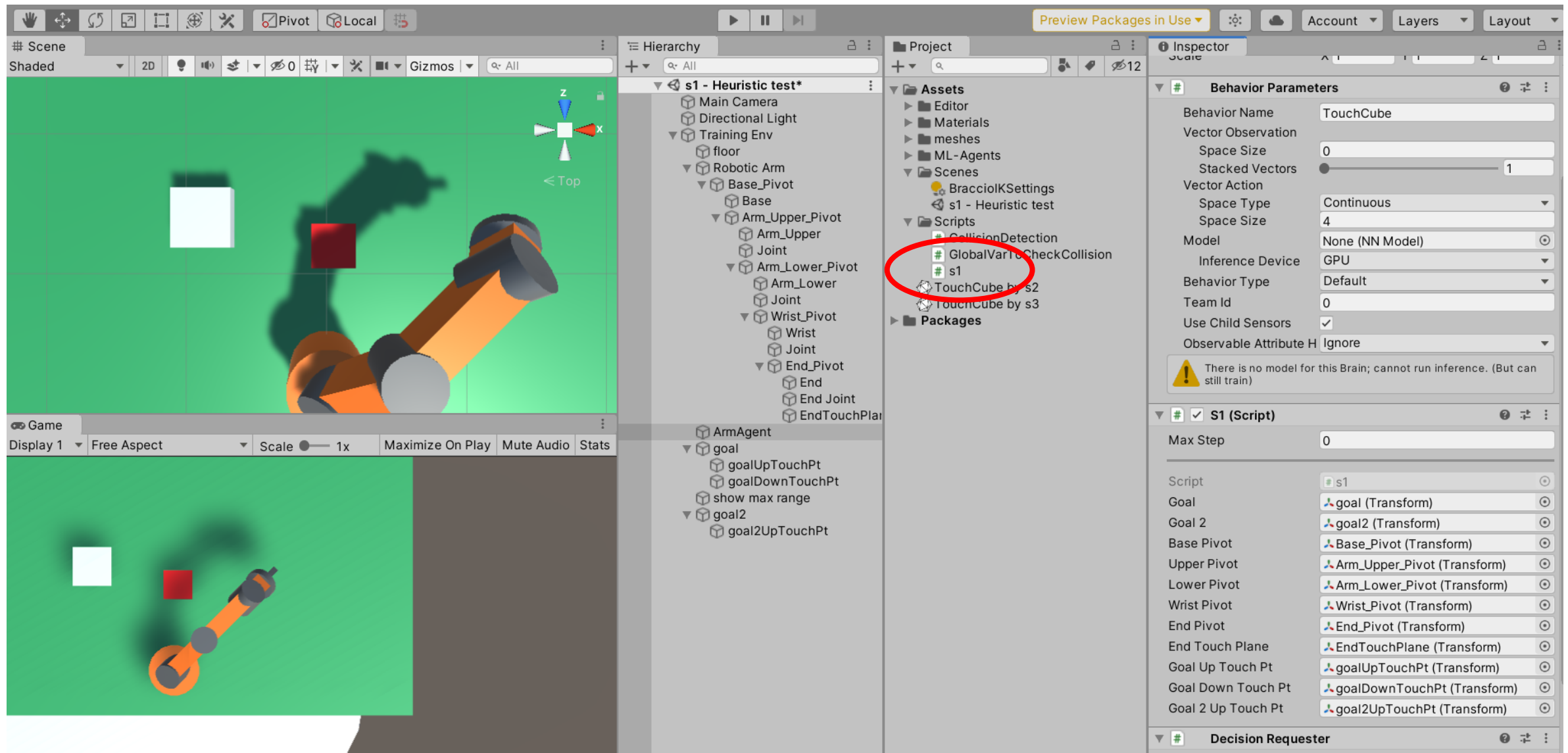
Create a new Unity project and import ML Agent package



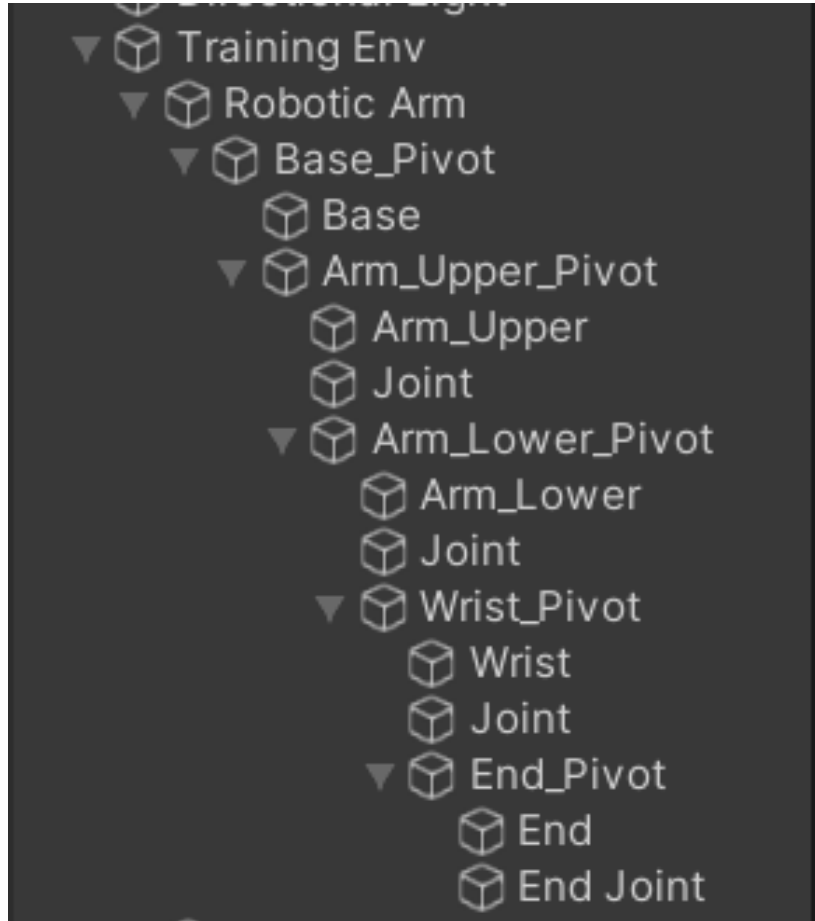
Import Robot arm package to this Unity project




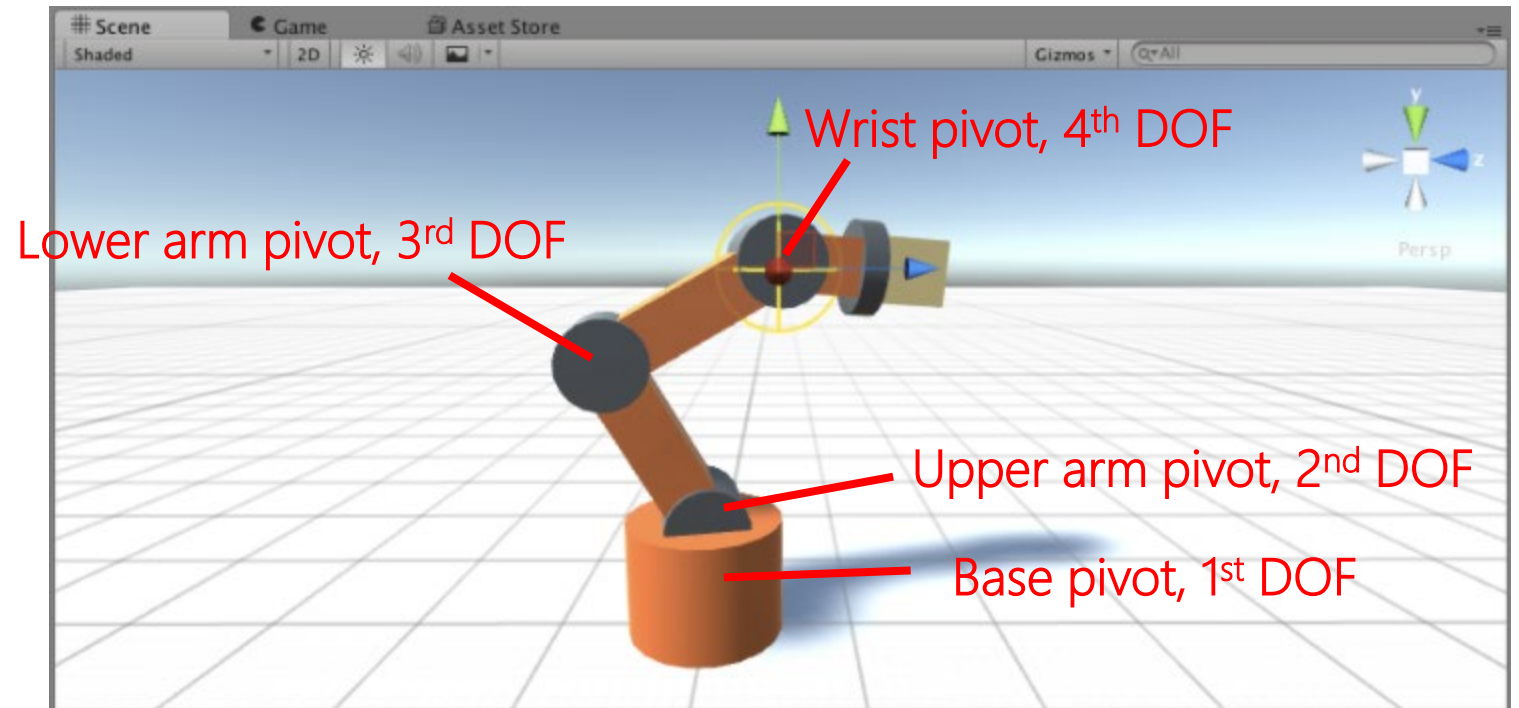
Open scene "s1 - Heuristic play"



This Unity project contains a Braccio robot arm



Alt +  lets you expand all hierarchies of the robot arm

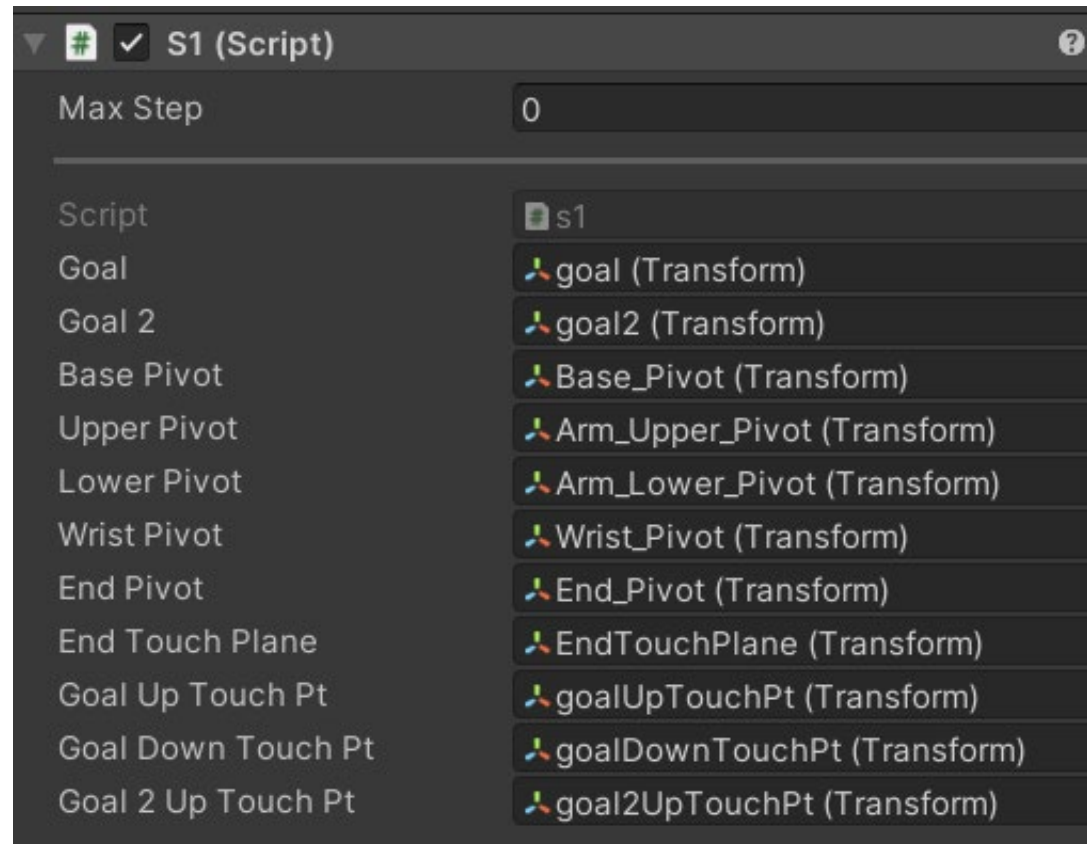


<https://github.com/tanyuan/braccio-ik-unity>

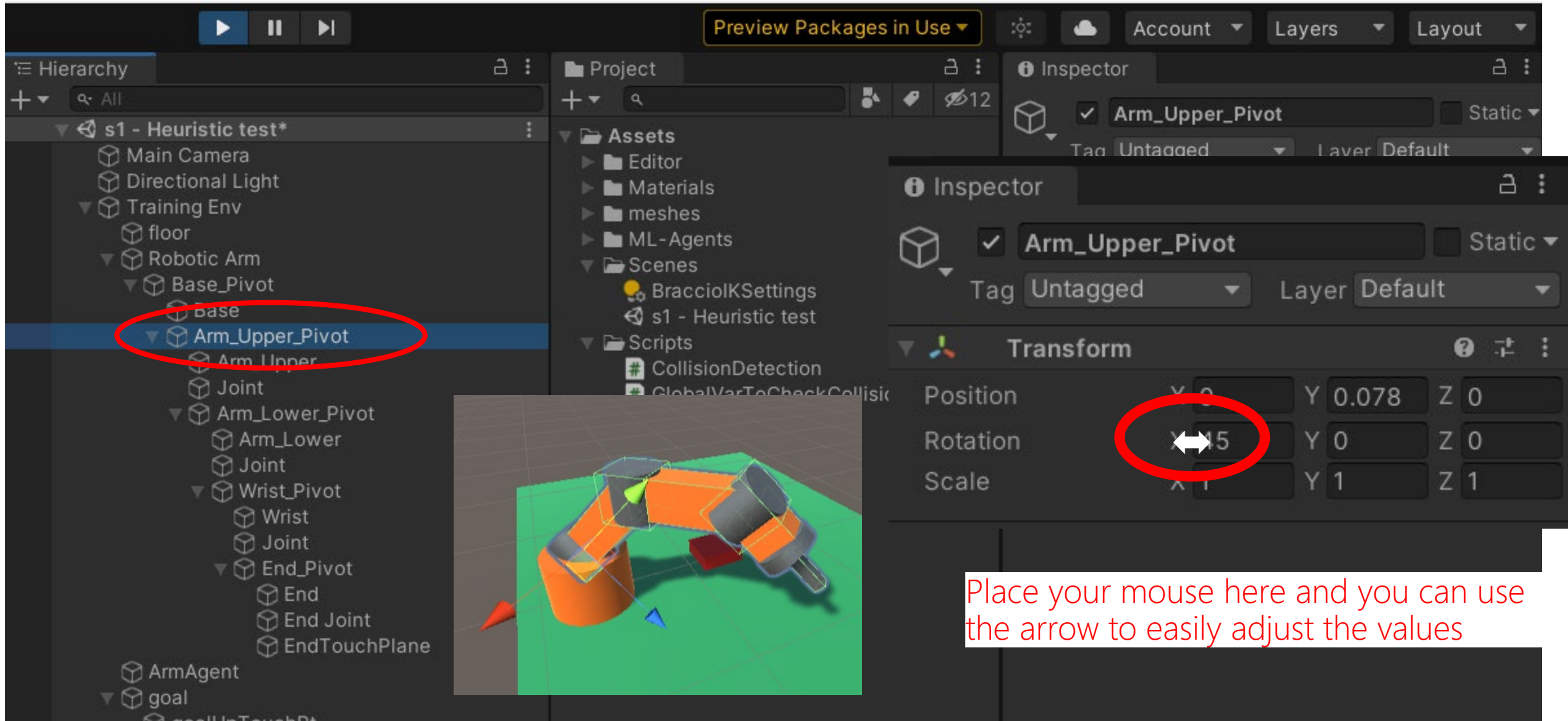
Manually operates the robot arm in VE

Public variables to link agent script with scene objects

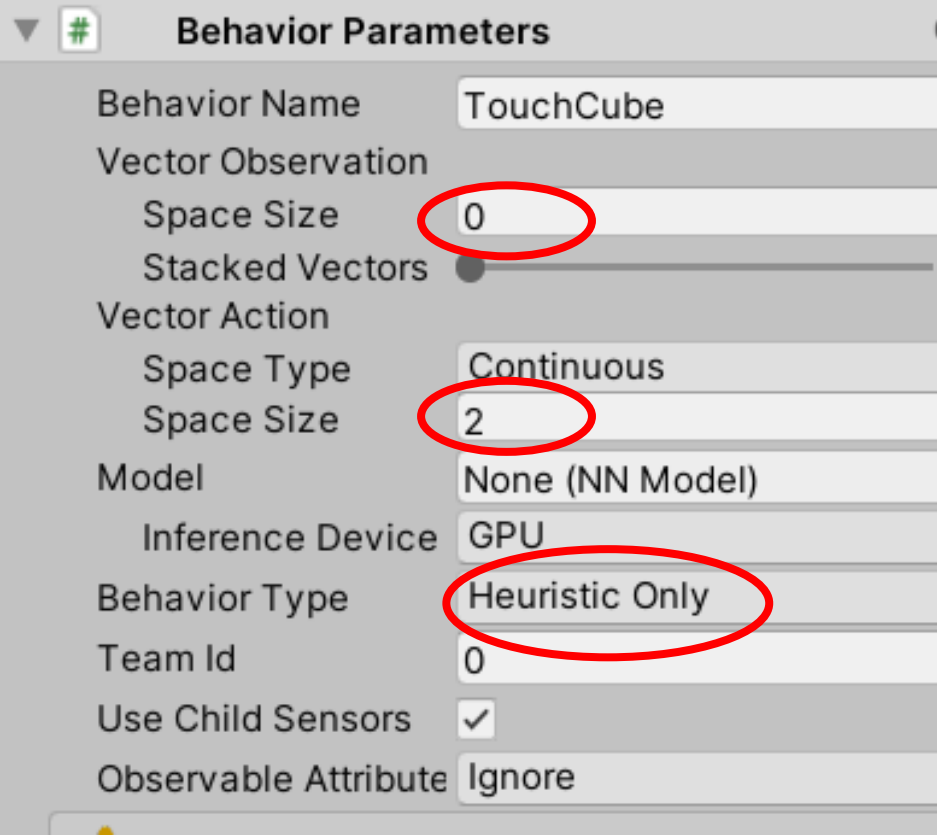
```
public Transform goal, goal2;  
public Transform BasePivot, UpperPivot, LowerPivot, WristPivot, EndPivot;  
public Transform EndTouchPlane, goalUpTouchPt, goalDownTouchPt, goal2UpTouchPt;  
int stage = 1;
```



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



Behavior parameters



The screenshot shows a configuration window titled "Behavior Parameters" with a green icon and a dropdown arrow. The parameters are as follows:

Parameter	Value
Behavior Name	TouchCube
Vector Observation	
Space Size	0
Stacked Vectors	<input type="radio"/>
Vector Action	
Space Type	Continuous
Space Size	2
Model	None (NN Model)
Inference Device	GPU
Behavior Type	Heuristic Only
Team Id	0
Use Child Sensors	<input checked="" type="checkbox"/>
Observable Attribute	Ignore

Red circles highlight the values for Space Size (0), Space Size (2), Behavior Type (Heuristic Only), and Team Id (0).

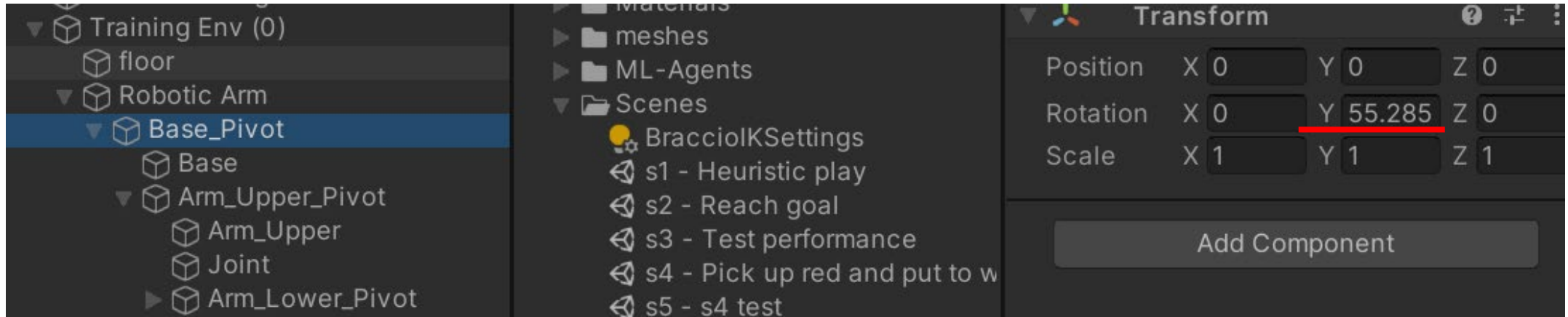
Play and use Up/Down, L/R keys to rotate arm

```
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;
}
```

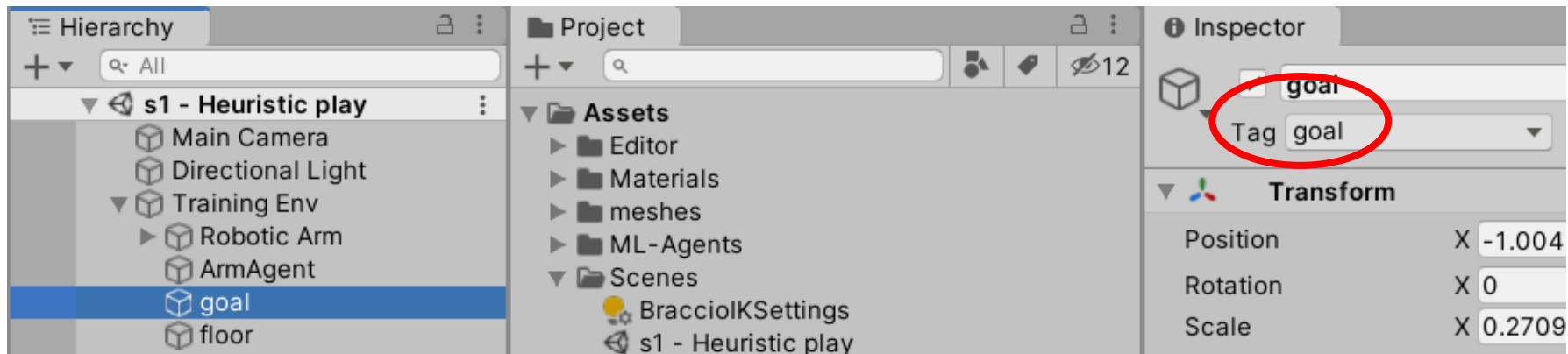
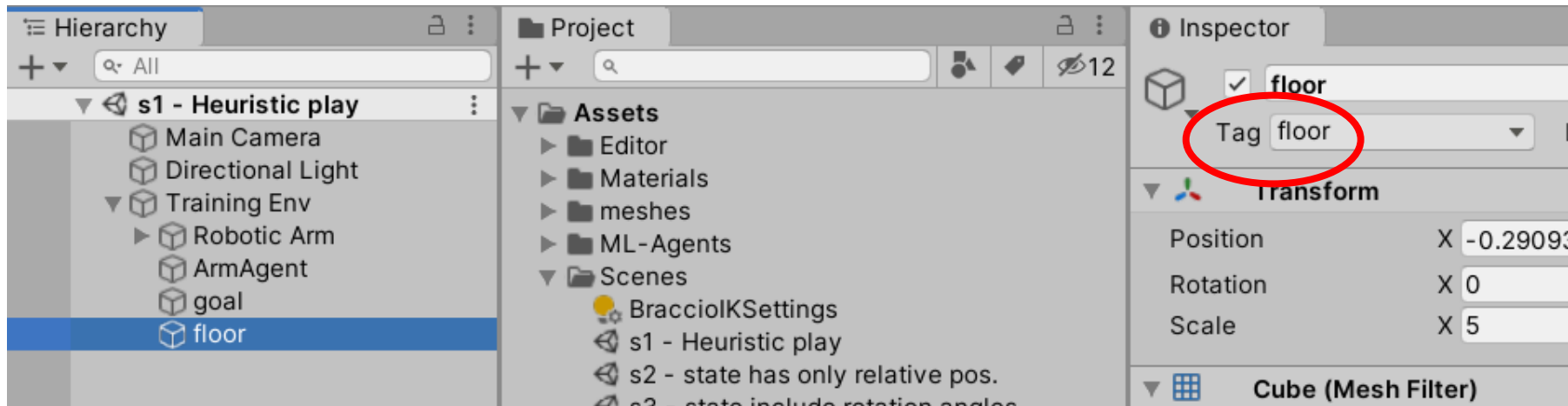
Use OnTriggerEnter/Exist to record collisions between robot arm and other scene objects

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

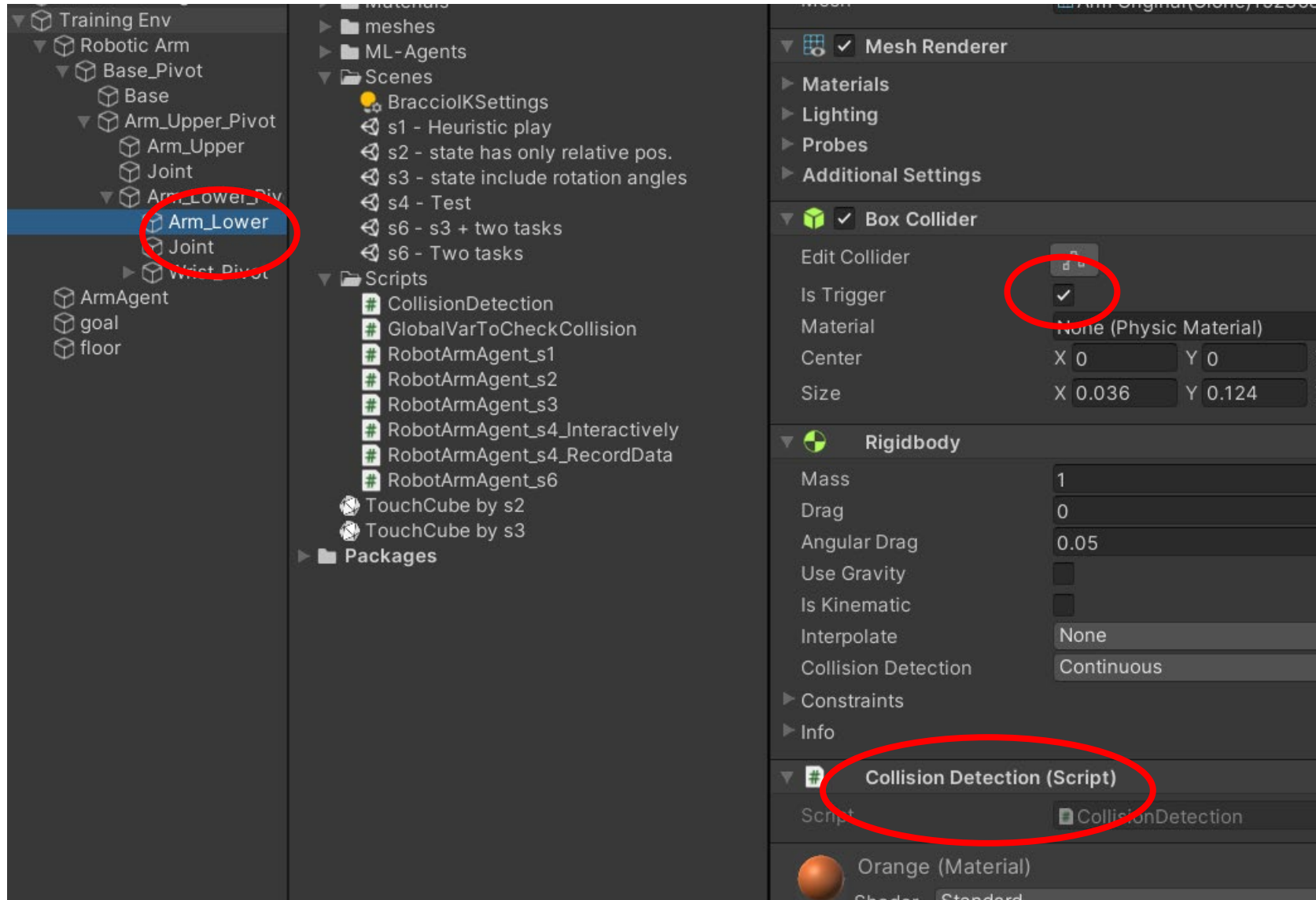
    void OnTriggerExit(Collider other)
    {
        if (other.gameObject.tag == "floor" || o
        {
            CollisionHappen = false;
        }
    }
}
```

Add tags to floor and goal object

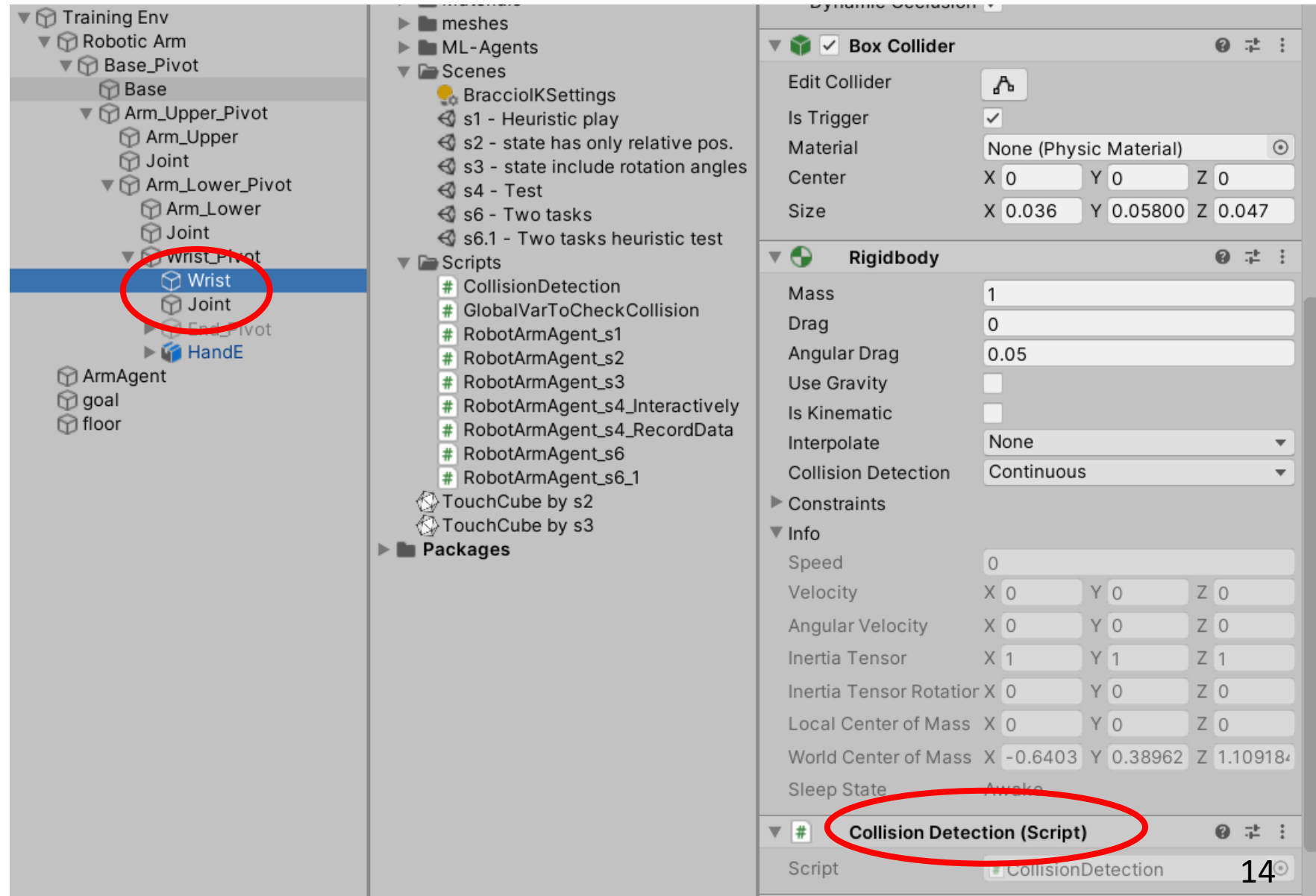
```
if (other.gameObject.tag == "floor" ||  
{  
    CollisionHappen = true;
```



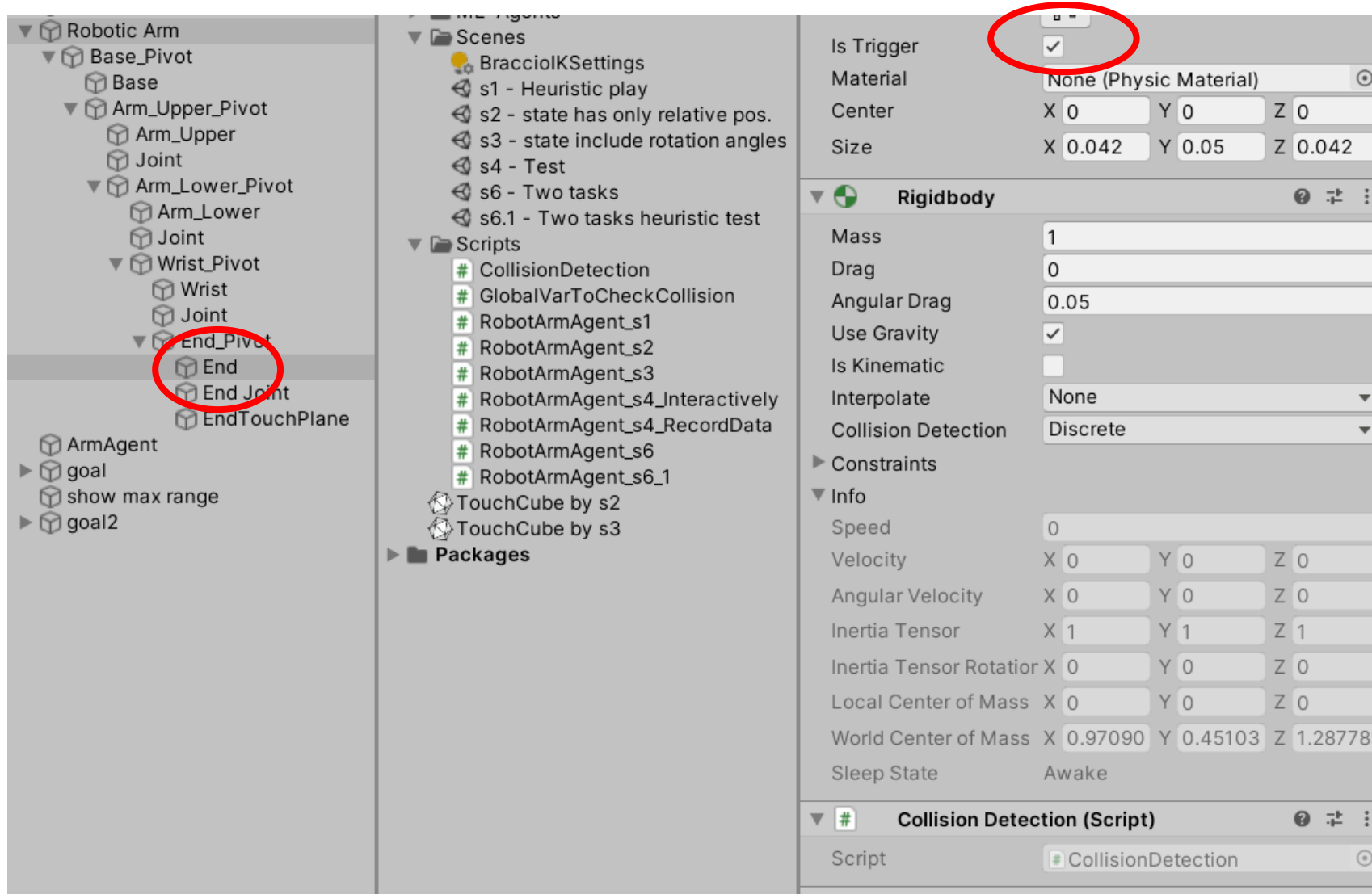
Add trigger collider, Rigid body, and collision detection script to Lower Arm



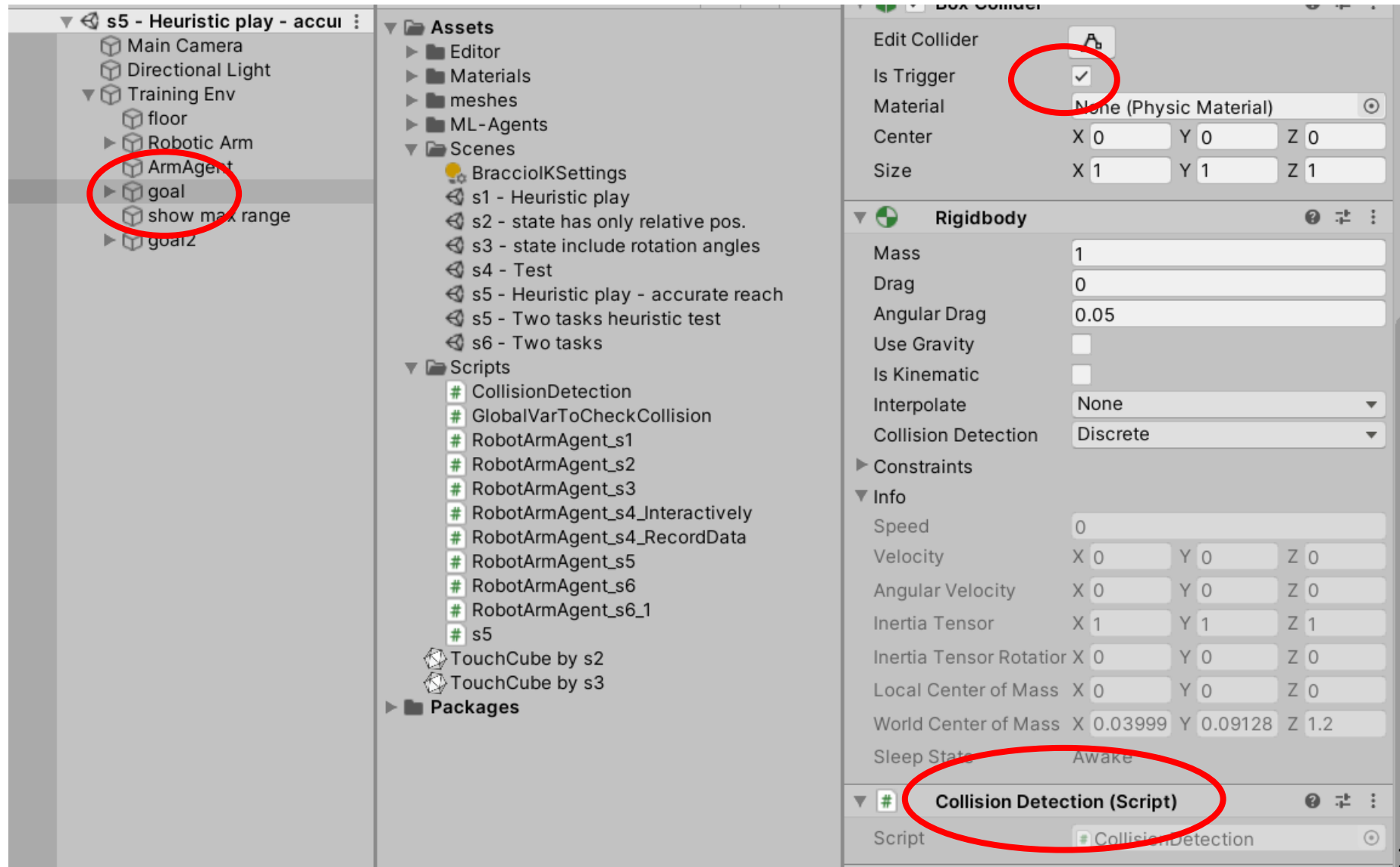
Add trigger collider, Rigid body, and collision detection script to Wrist



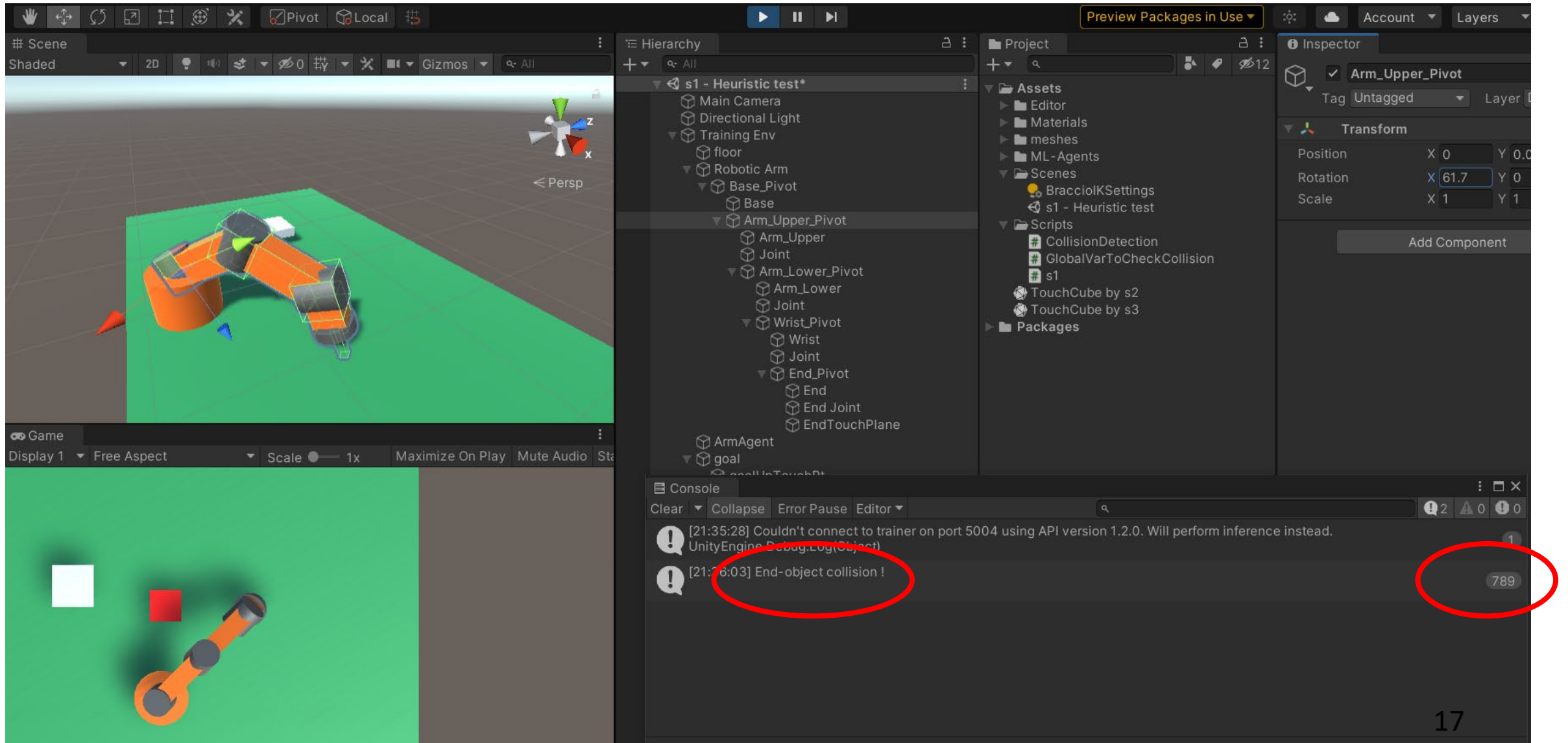
Add trigger collider, Rigid body, and collision detection script to Robot End



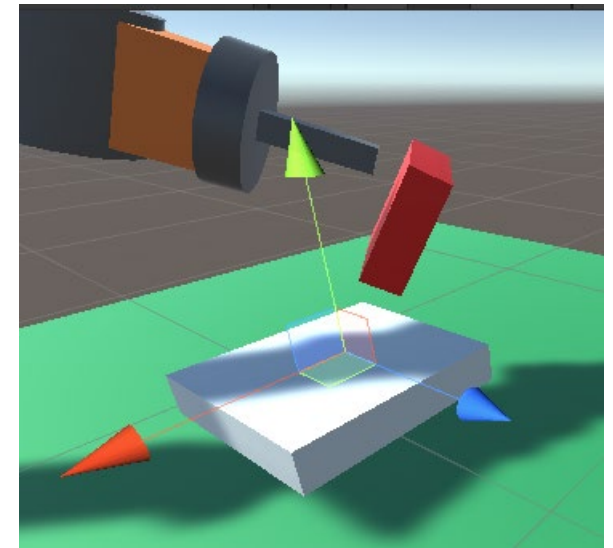
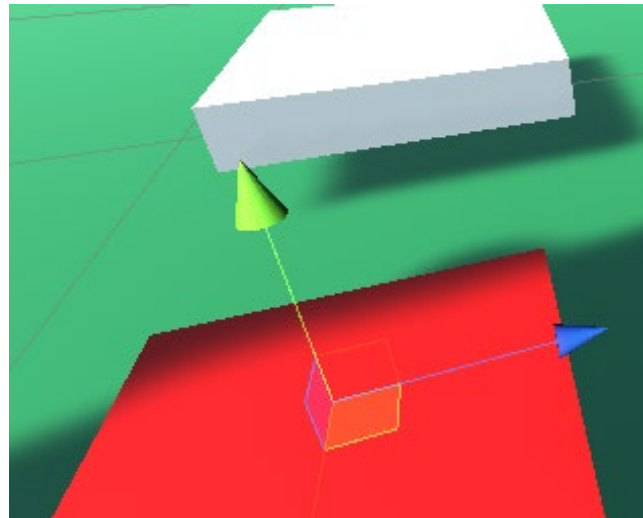
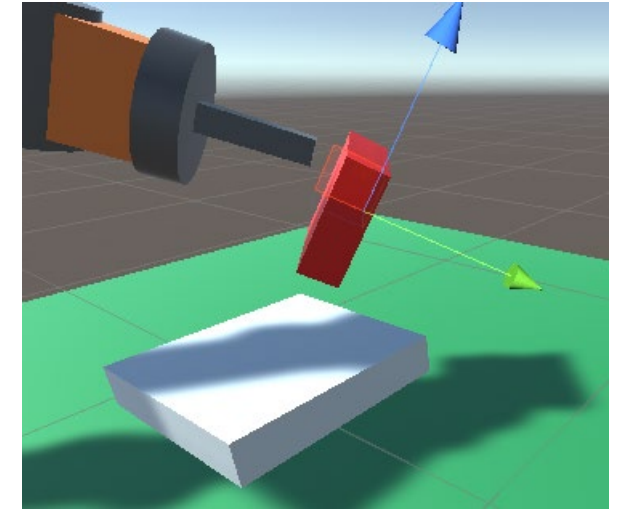
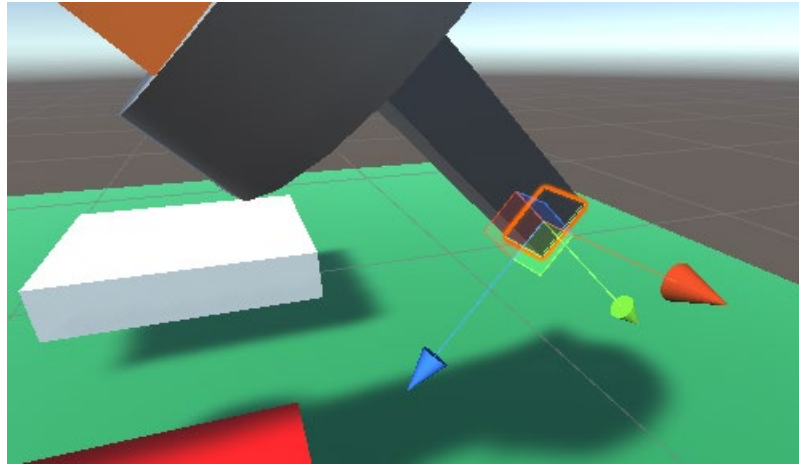
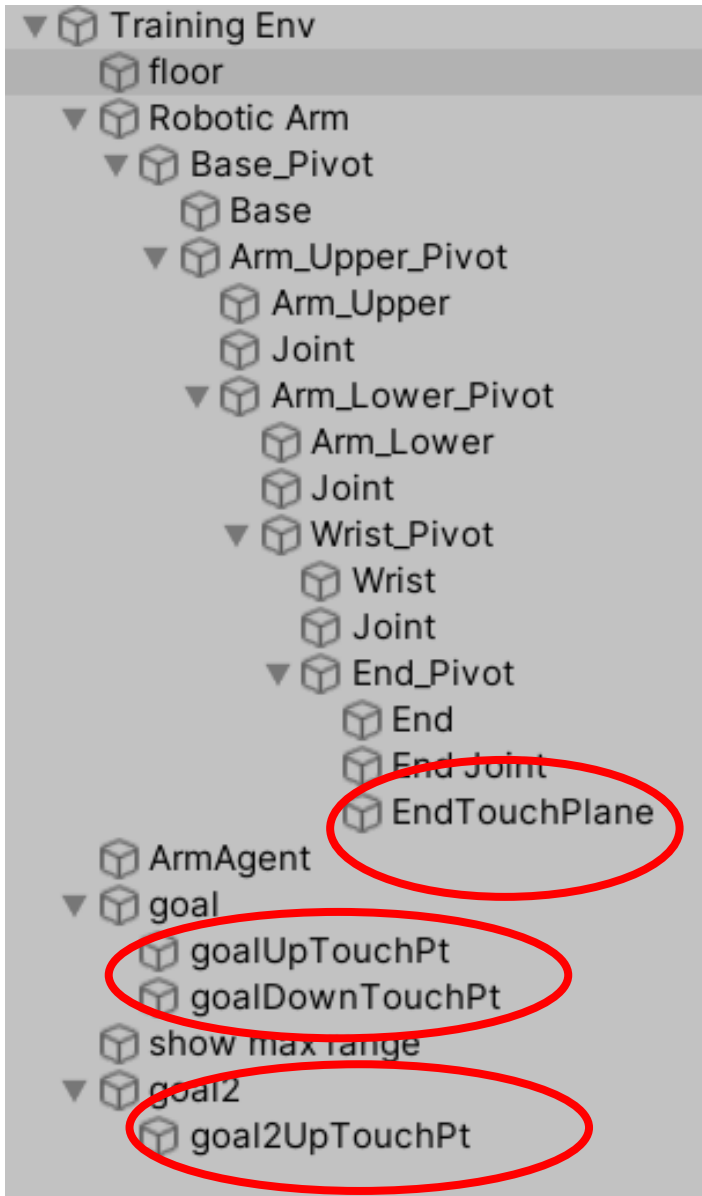
Add trigger collider, Rigid body, and collision detection script to goal object



Test collision



Assign points for reach-detection

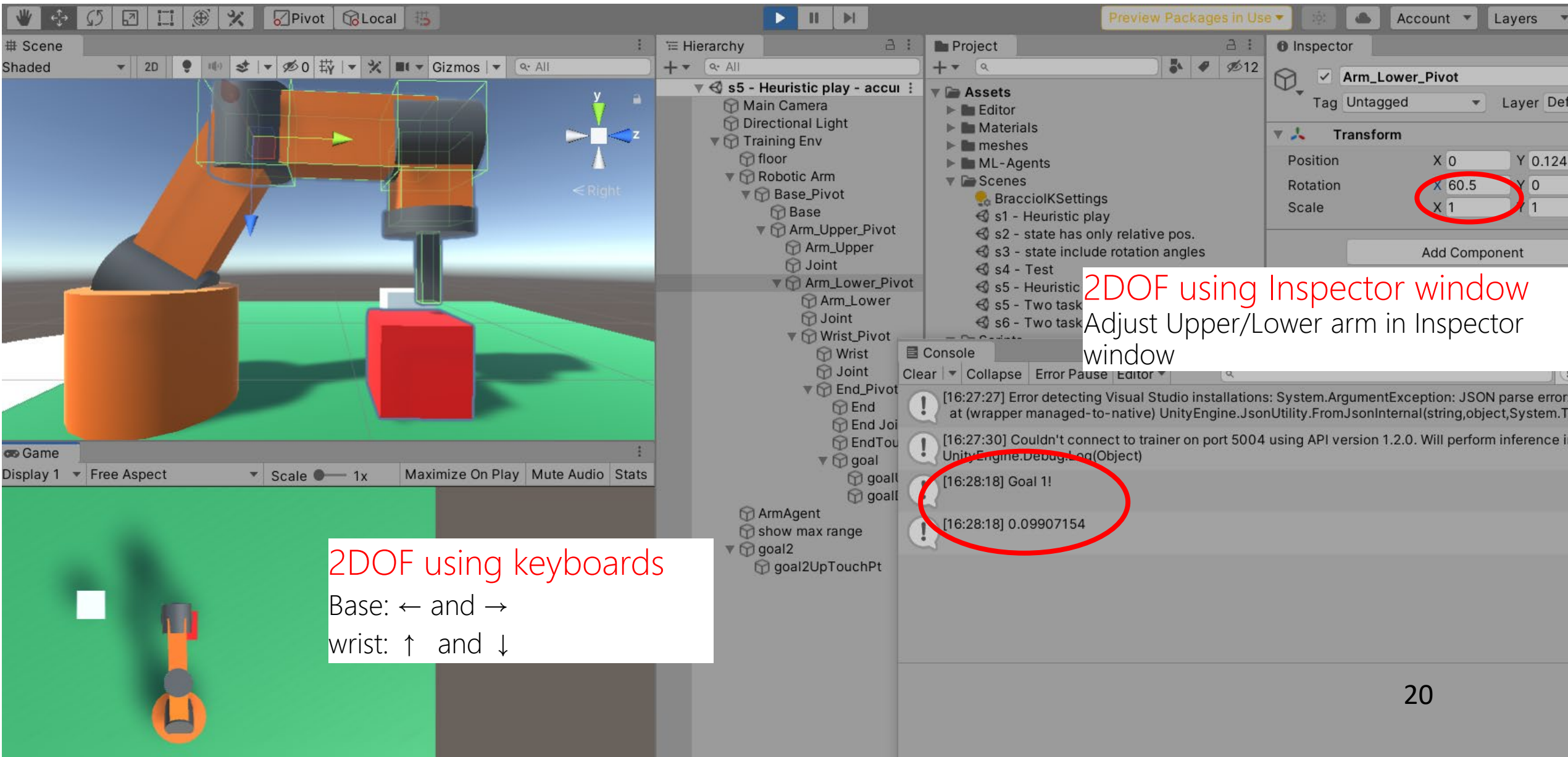


Point-based touch detection

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

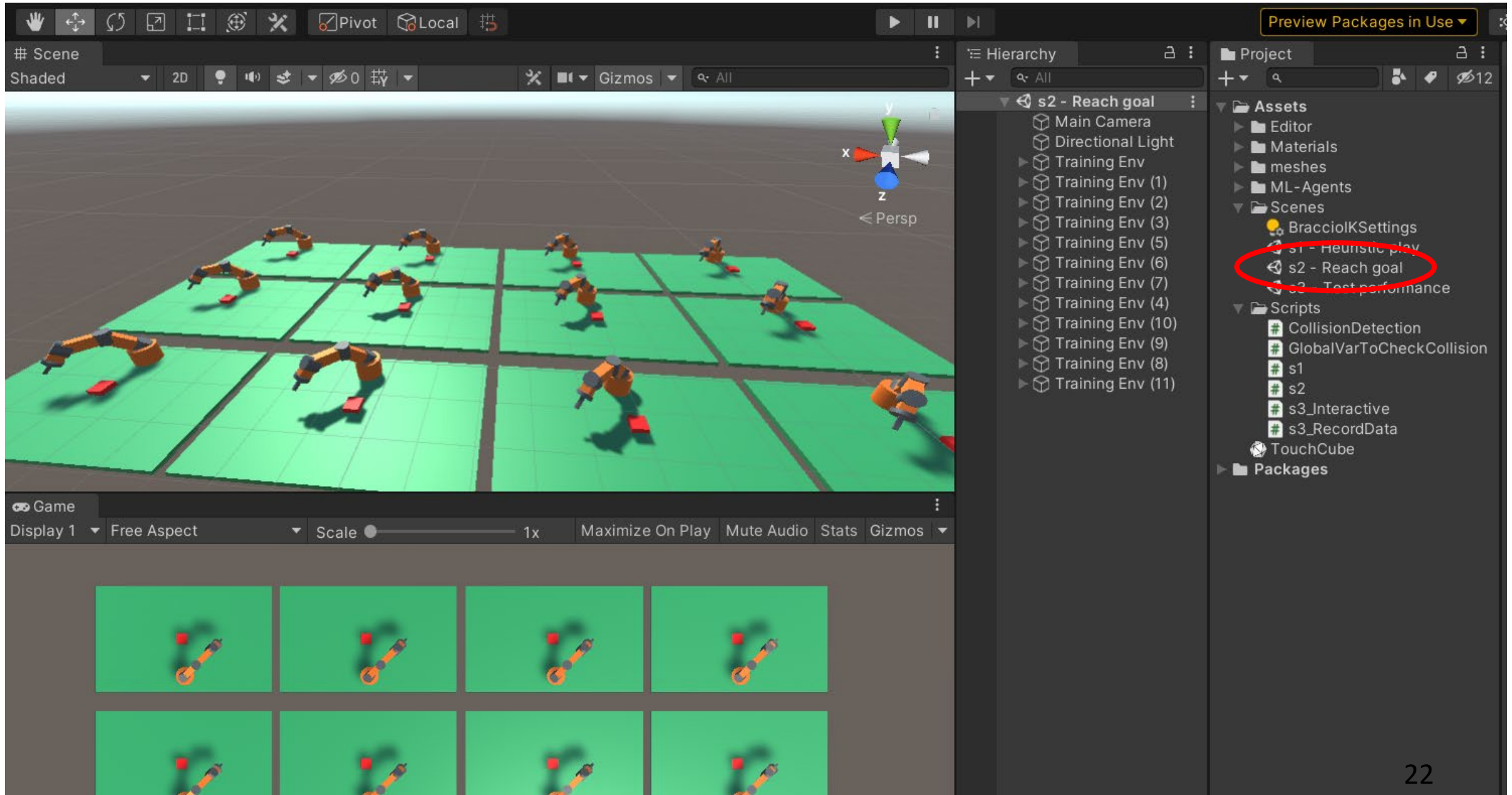
    dx = Mathf.Abs(pt1.position.x - pt2.position.x);
    dz = Mathf.Abs(pt1.position.z - pt2.position.z);
    dy = pt1.position.y - pt2.position.y;
    //msg = dx.ToString() + ", " + dy.ToString() + ", " + dz.ToString(
    //print(msg);
    if (dy > 0 && dy < threshold && dx < threshold && dz < threshold)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

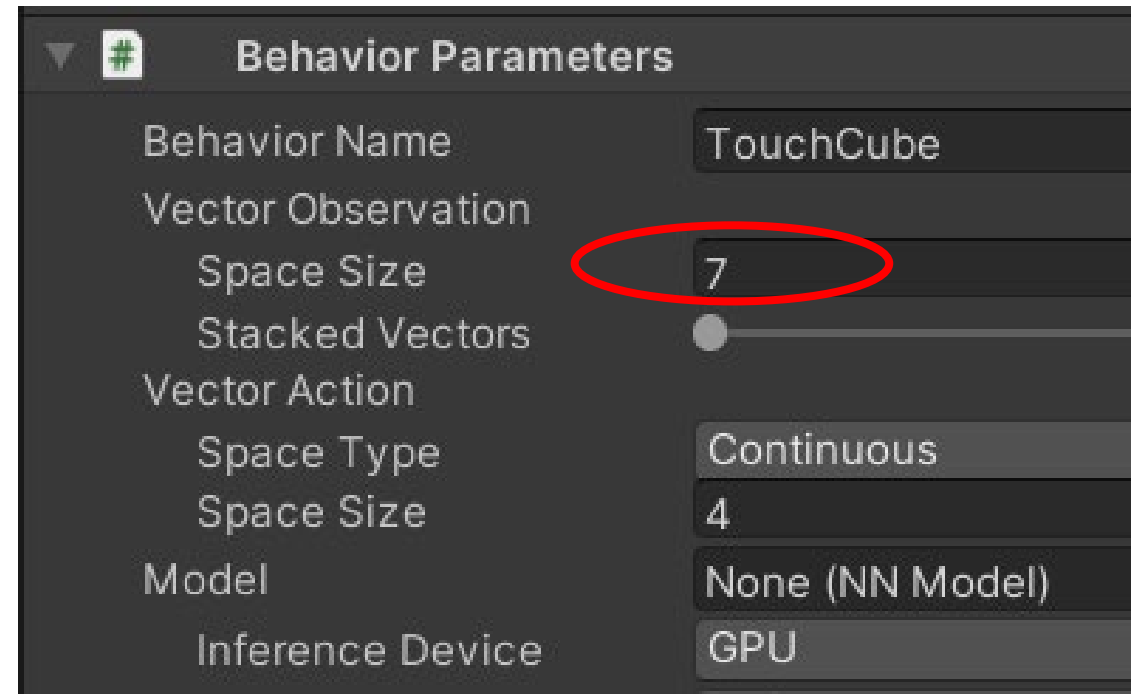
```
float radius = UnityEngine.Random.Range(radiusMin, radiusMax);  
float theta = (UnityEngine.Random.Range(thetaMin, thetaMax) / 180.0f) * Mathf.PI;  
    to radians  
float x = radius * Mathf.Sin(theta);  
float z = radius * Mathf.Cos(theta);  
return new Vector3(x, y, z);
```

State has 7 variables

```
sensor.AddObservation(EndTouchPlane.position - goalUpTouchPt.transform.position);
```

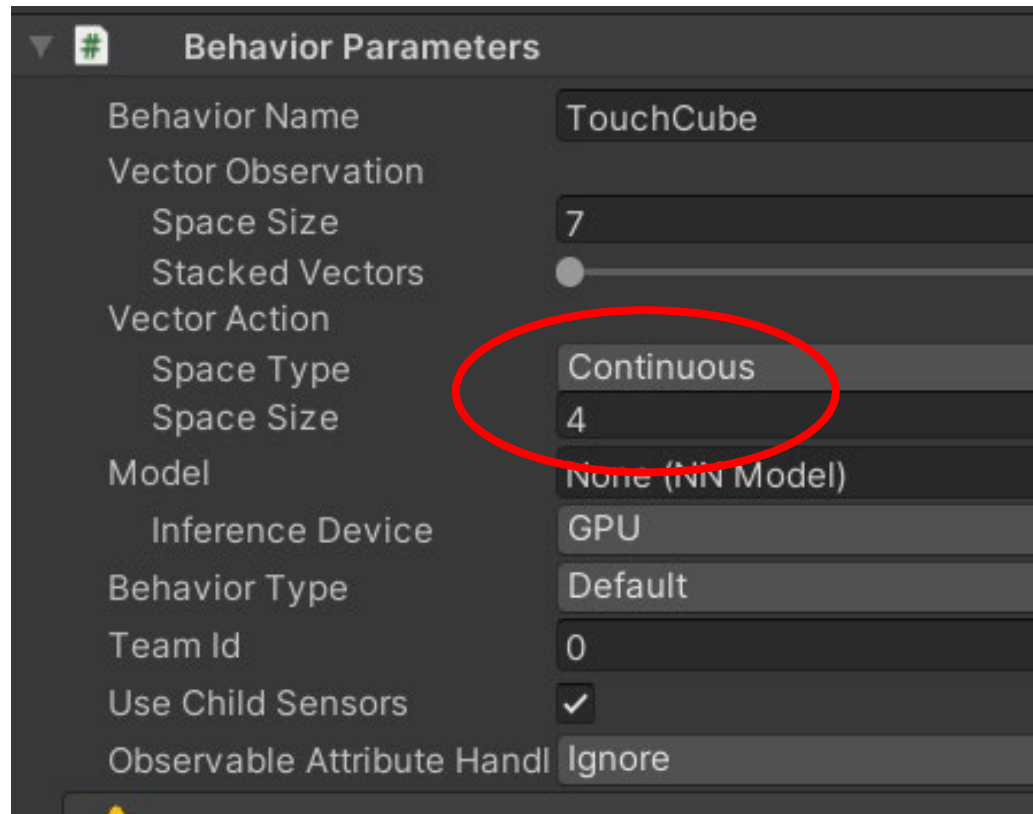
```
float BaseRotationAngle = UnityEditor.TransformUtils.GetInspectorRotation(BasePivot).y;  
float UArmRotationAngle = UnityEditor.TransformUtils.GetInspectorRotation(UpperPivot).x;  
float LArmRotationAngle = UnityEditor.TransformUtils.GetInspectorRotation(LowerPivot).x;  
float WRotationAngle = UnityEditor.TransformUtils.GetInspectorRotation(WristPivot).x;
```

```
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);
```



The screenshot shows a 'Behavior Parameters' panel with the following settings:

Parameter	Value
Behavior Name	TouchCube
Vector Observation	
Space Size	7
Stacked Vectors	<input type="checkbox"/>
Vector Action	
Space Type	Continuous
Space Size	4
Model	None (NN Model)
Inference Device	GPU
Behavior Type	Default
Team Id	0
Use Child Sensors	<input checked="" type="checkbox"/>
Observable Attribute Handl	Ignore

Reward

Punish every step to
avoid 耍廢

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

HW

What will happen if do
not end episode ?

```
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();  
}
```

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

lambda: 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

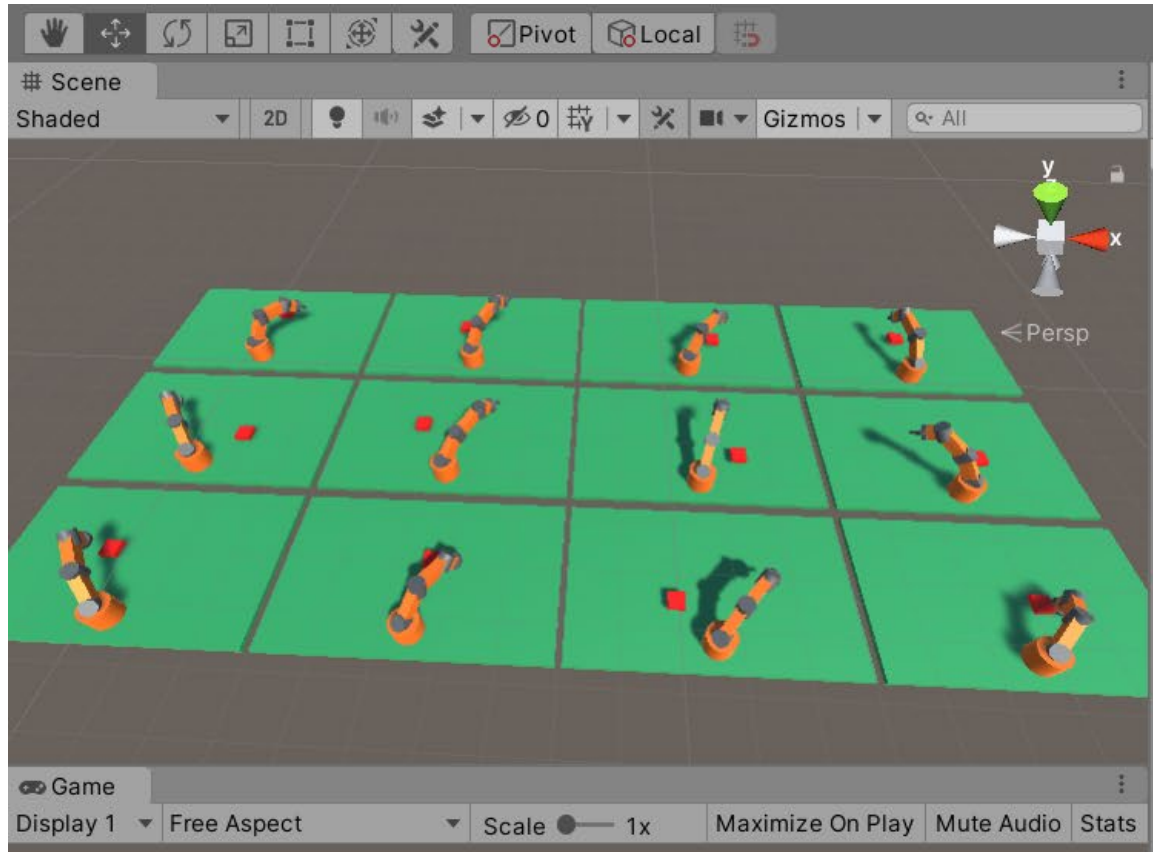
max_steps: 5000000

time_horizon: 2000

summary_freq: 30000

threaded: true

Examine environment that reaches goal



The image shows a Unity 3D environment with a 4x4 grid of 16 training environments. Each environment contains a robotic arm and a red cube. The environment is viewed from a perspective camera. The interface includes a Hierarchy panel, a Project panel, and a Console panel.

Hierarchy Panel:

- s2 - Reach goal*
 - Main Camera
 - Directional Light
 - Training Env
 - Training Env (1)
 - Training Env (2)
 - Training Env (3)
 - Training Env (5)
 - Training Env (6)
 - Training Env (7)
 - Training Env (4)
 - Training Env (10)
 - Training Env (9)
 - Training Env (11)
 - Training Env (8)

Project Panel:

- Assets
 - Editor
 - Materials
 - meshes
 - ML-Agents
 - Scenes
 - BracciolKSettings
 - s1 - Heuristic play
 - s2 - Reach goal
 - s3 - Test performance
 - s4 - Pick up red and put to white

Console Panel:

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

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

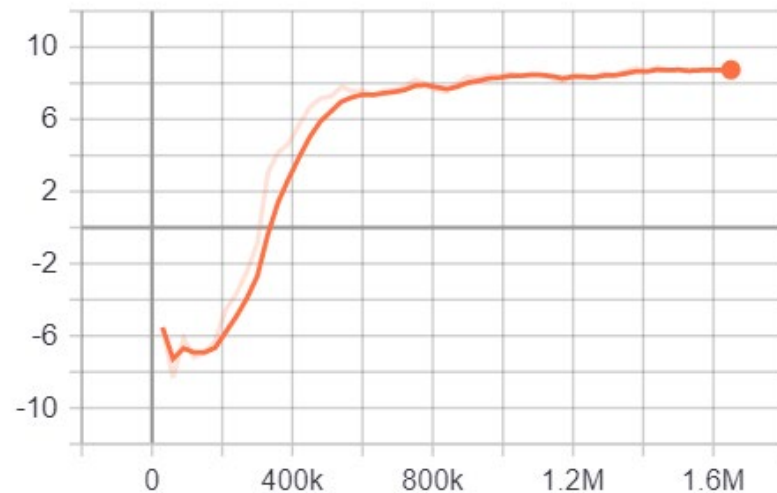

Results after 1.7M steps, looks promising

```
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: 12.265. 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
[TouchCube.py:93] Converting to results\1\TouchCube\TouchCube-999992.onnx
[TouchCube.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

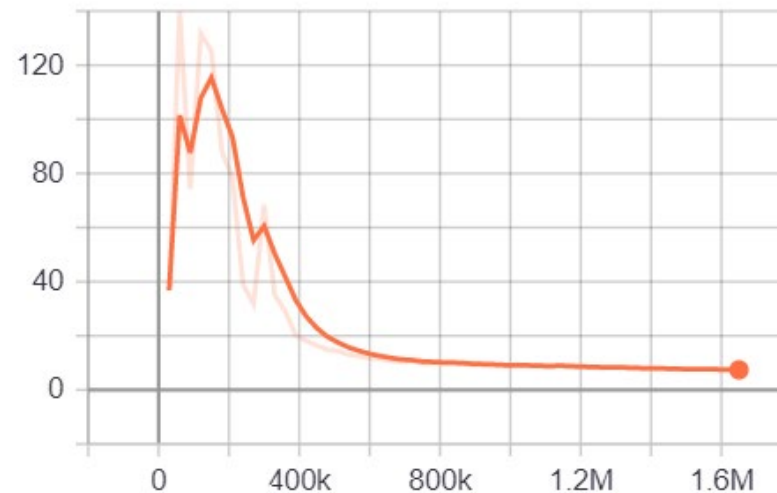
Cumulative Reward

tag: Environment/Cumulative Reward



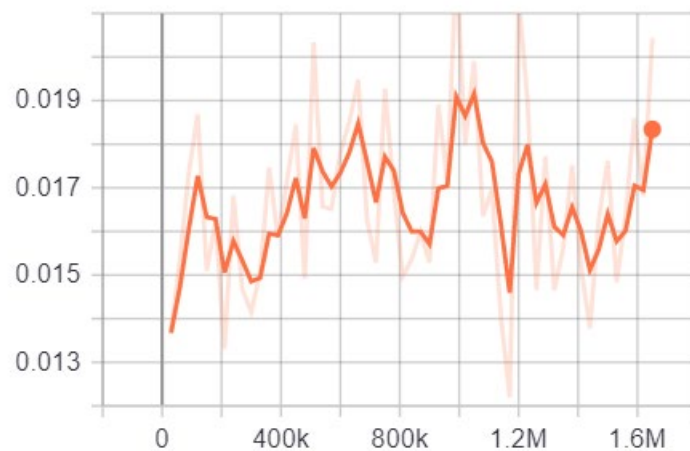
Episode Length

tag: Environment/Episode Length



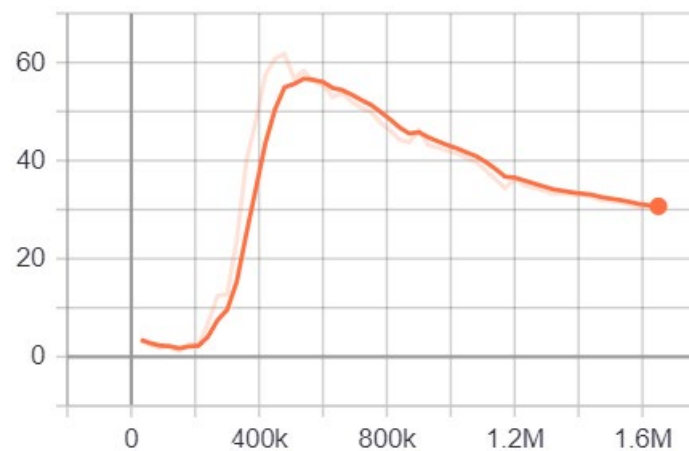
Policy Loss

tag: Losses/Policy Loss

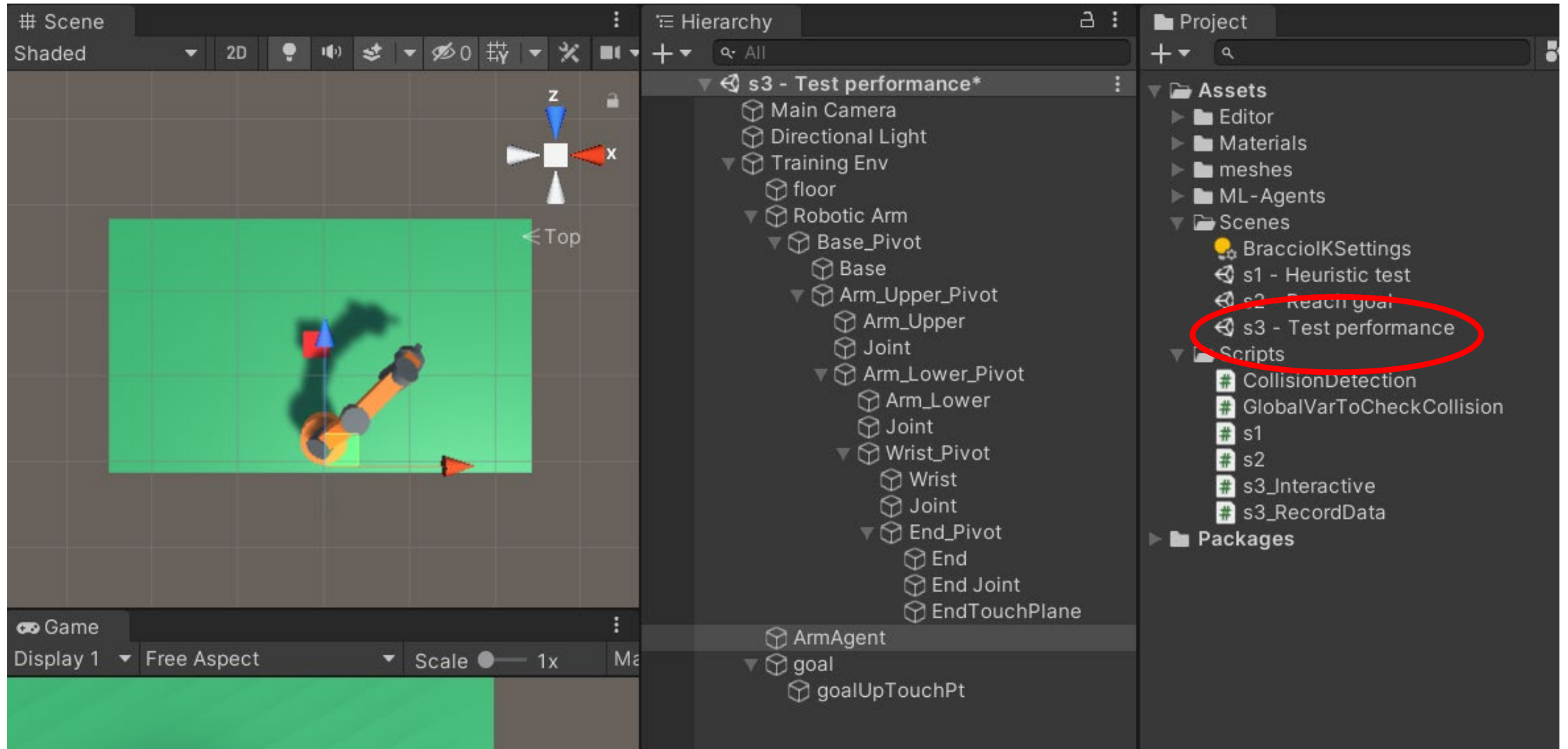


Value Loss

tag: Losses/Value Loss



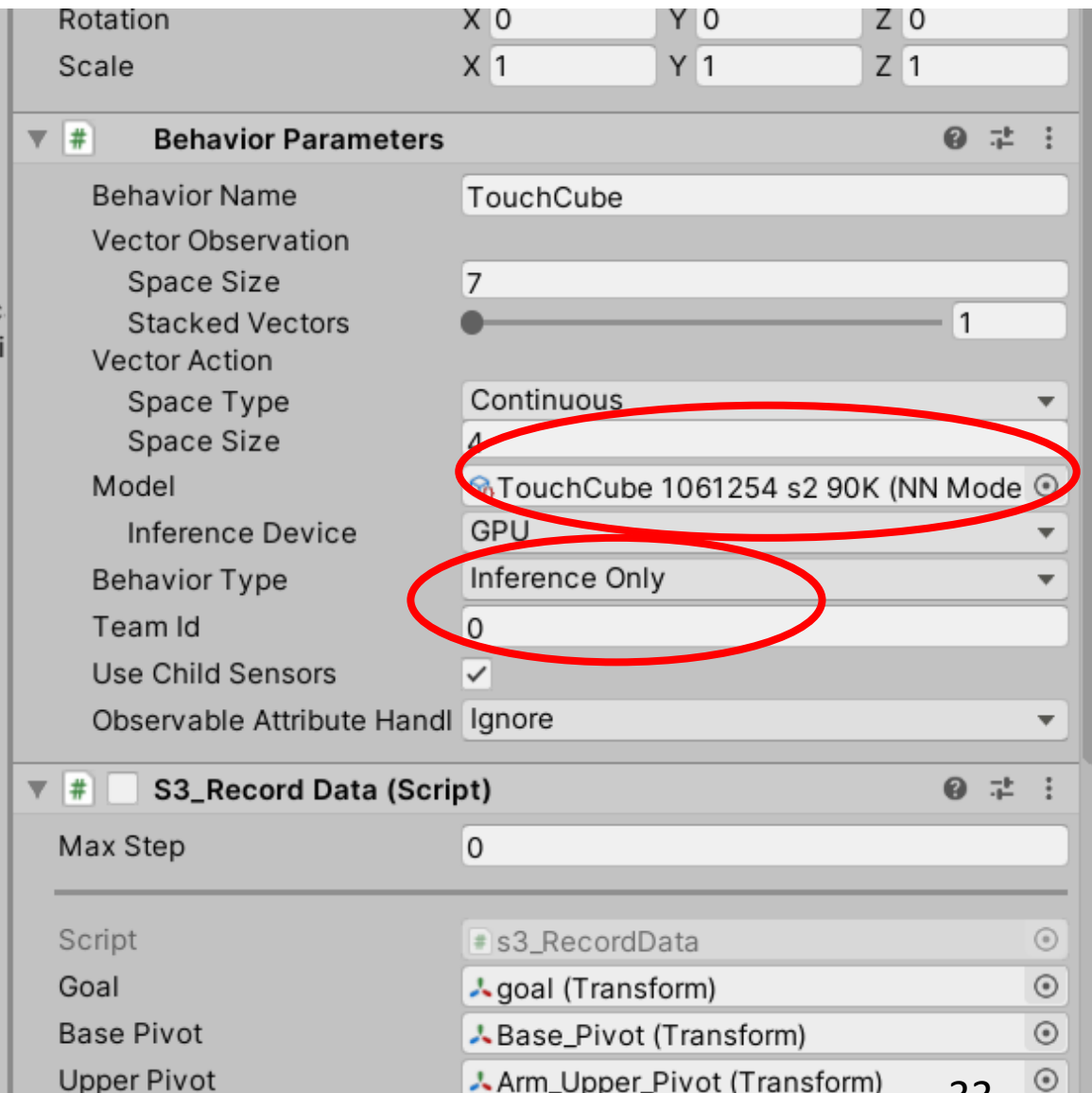
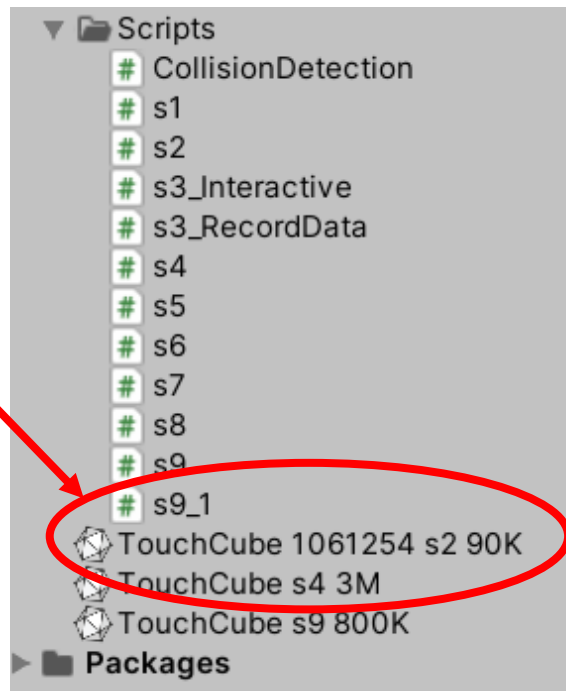
Open "s3 – Test performance"



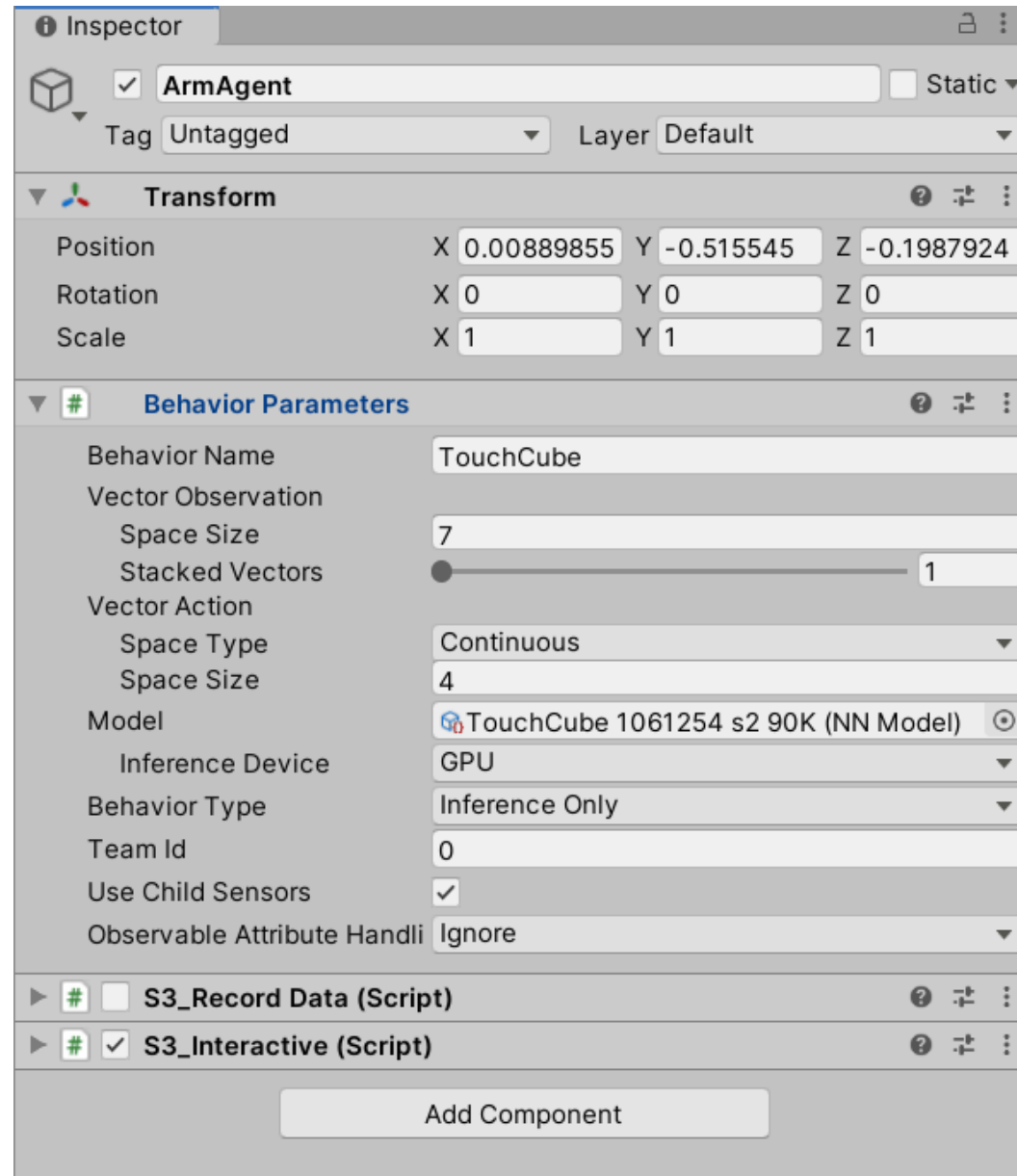
Assign trained NN

C:) > ml-agents-release_10 > config > ppo > results > 1

名稱	修改日期
run_logs	2021/3/12 上午 0...
TouchCube	2021/5/25 下午 0...
configuration	2021/5/25 下午 0...
<u>TouchCube.onnx</u>	2021/5/25 下午 0...



Remove decision requester from Behavior parameter



Assign decision requester in Update function

Record data test

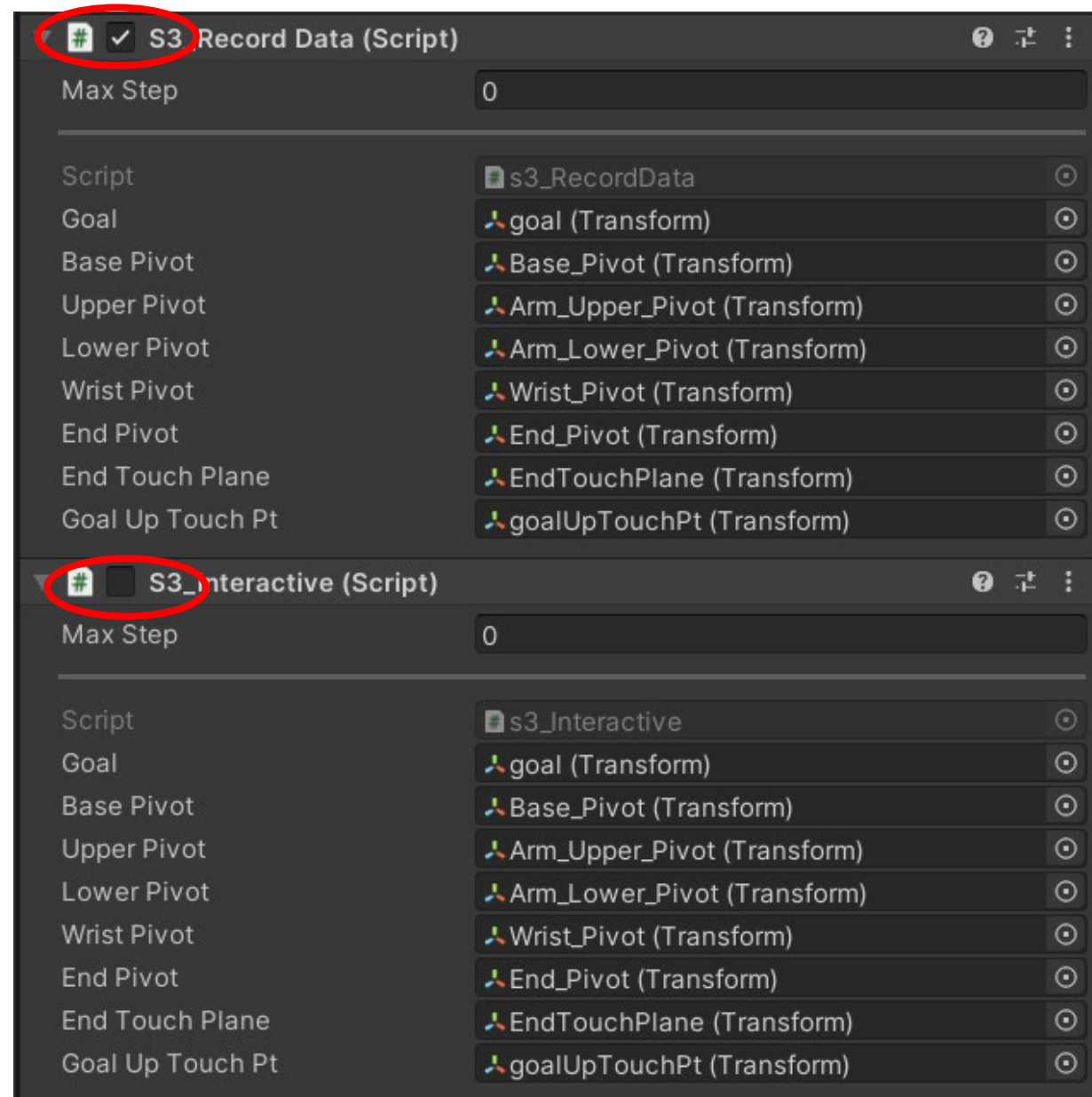
```
void Update()
{
    if (NoTest <= TotalTests)
    {
        if (!PointTouch(EndTouchPlane, goalUpTouchPt, 0.3f))
        {
            RequestDecision();
        }
        else //reach goal
        {
            string s = "Finish No " + NoTest.ToString();
            writer.WriteLine(s);
            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 ?

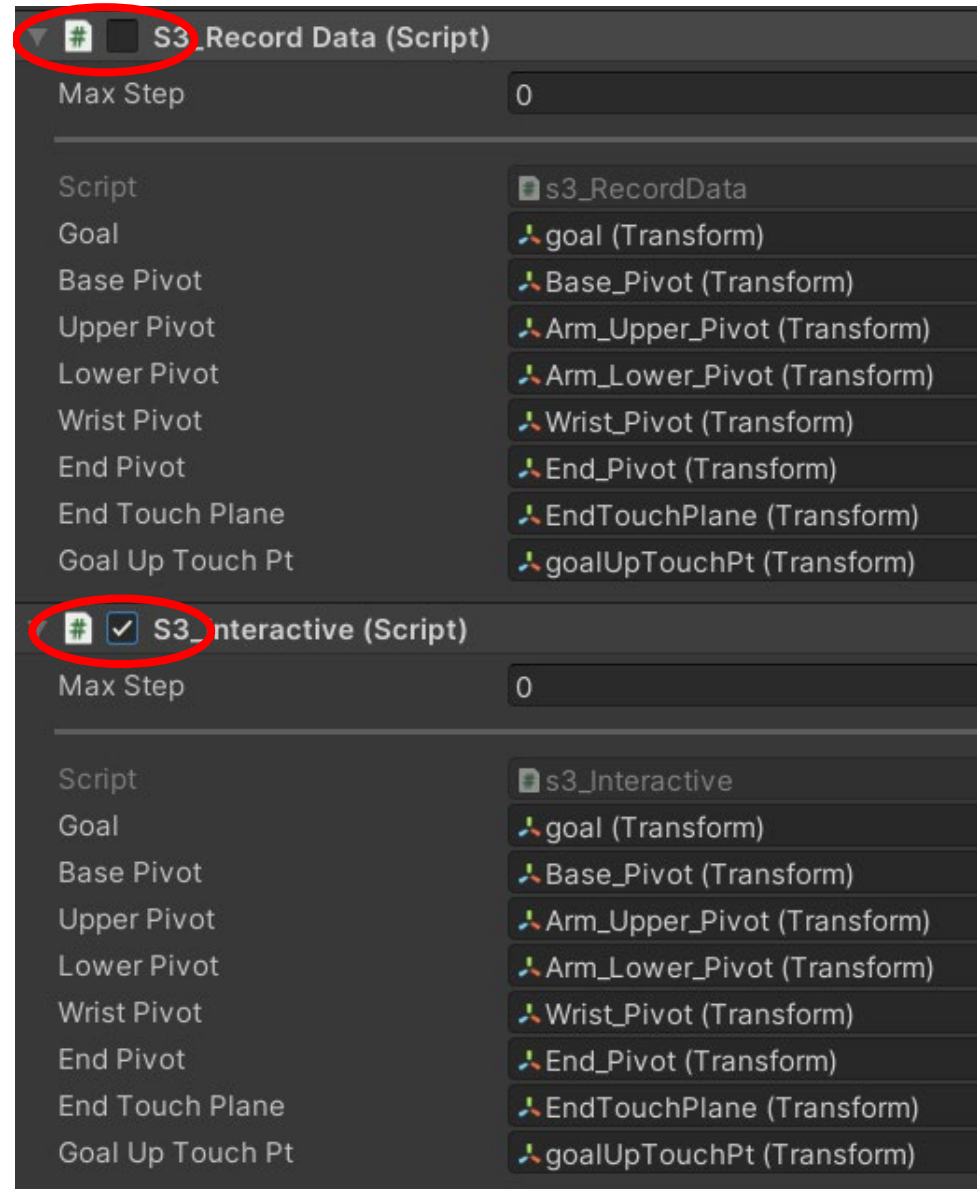
Examine the data file

2021.5.20 Robot Arm		
	名稱	修改日期
	Assets	2021/5
	Library	2021/5
	Logs	2021/5
	obj	2021/5
	Packages	2021/5
	ProjectSettings	2021/5
	Temp	2021/5
	UserSettings	2021/5
	2021.5.20 Robot Arm.sln	2021/5
	Assembly-CSharp	2021/5
	Assembly-CSharp-Editor	2021/5
	<u>trajectory</u>	2021/5

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