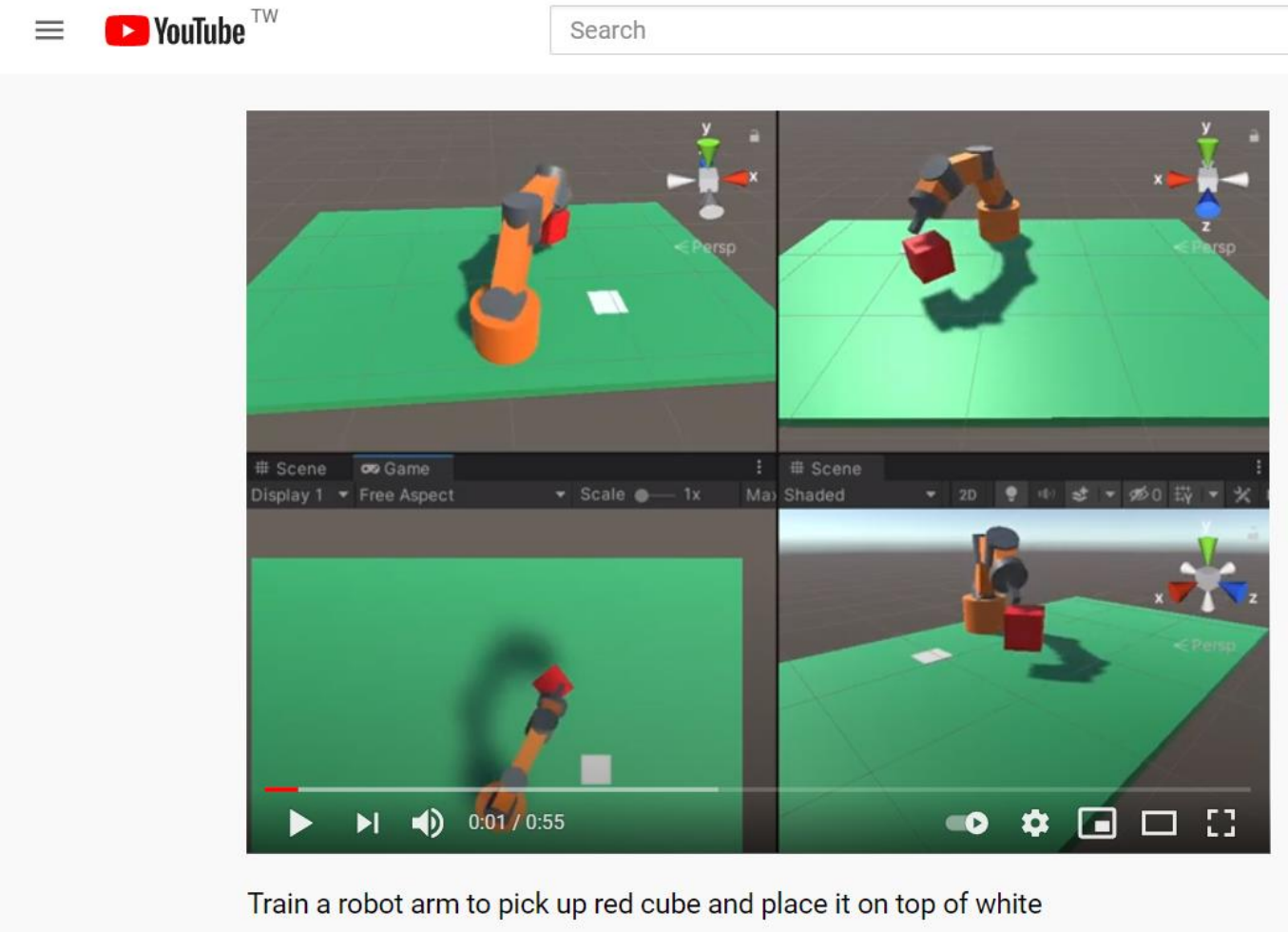
Youtube demo video

Youtube search: Train a robot arm to pick up red cube and place it on top of white



https://youtu.be/-NVNT-YTv1s

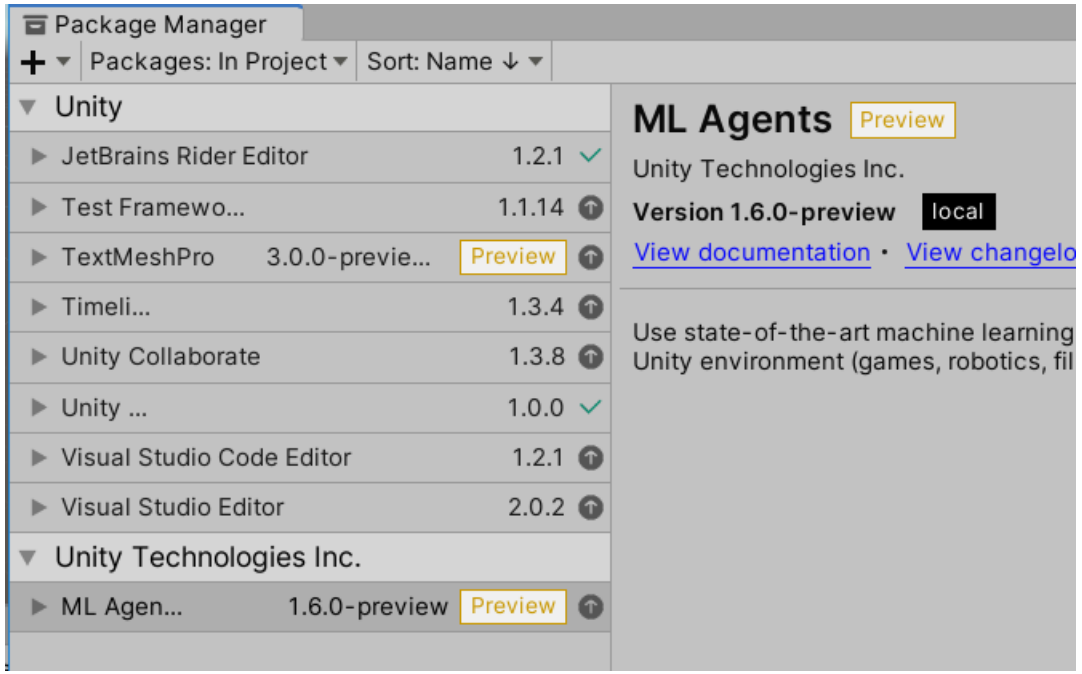
Download and save ML Agent to C:\

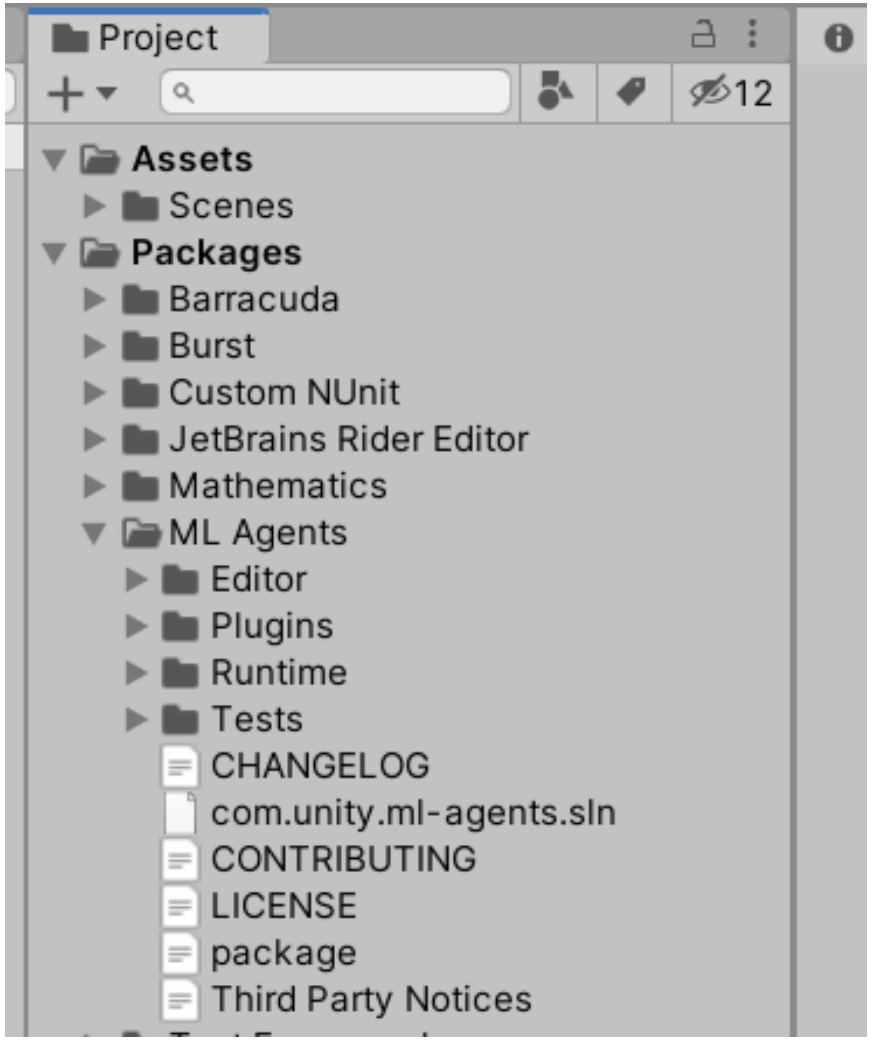
This will make it convenient to type commands to train and monitor performance

```
cd C:\ml-agents-release_10\config\ppo
mlagents-learn TouchCube.yaml --run-id=1 --force
```

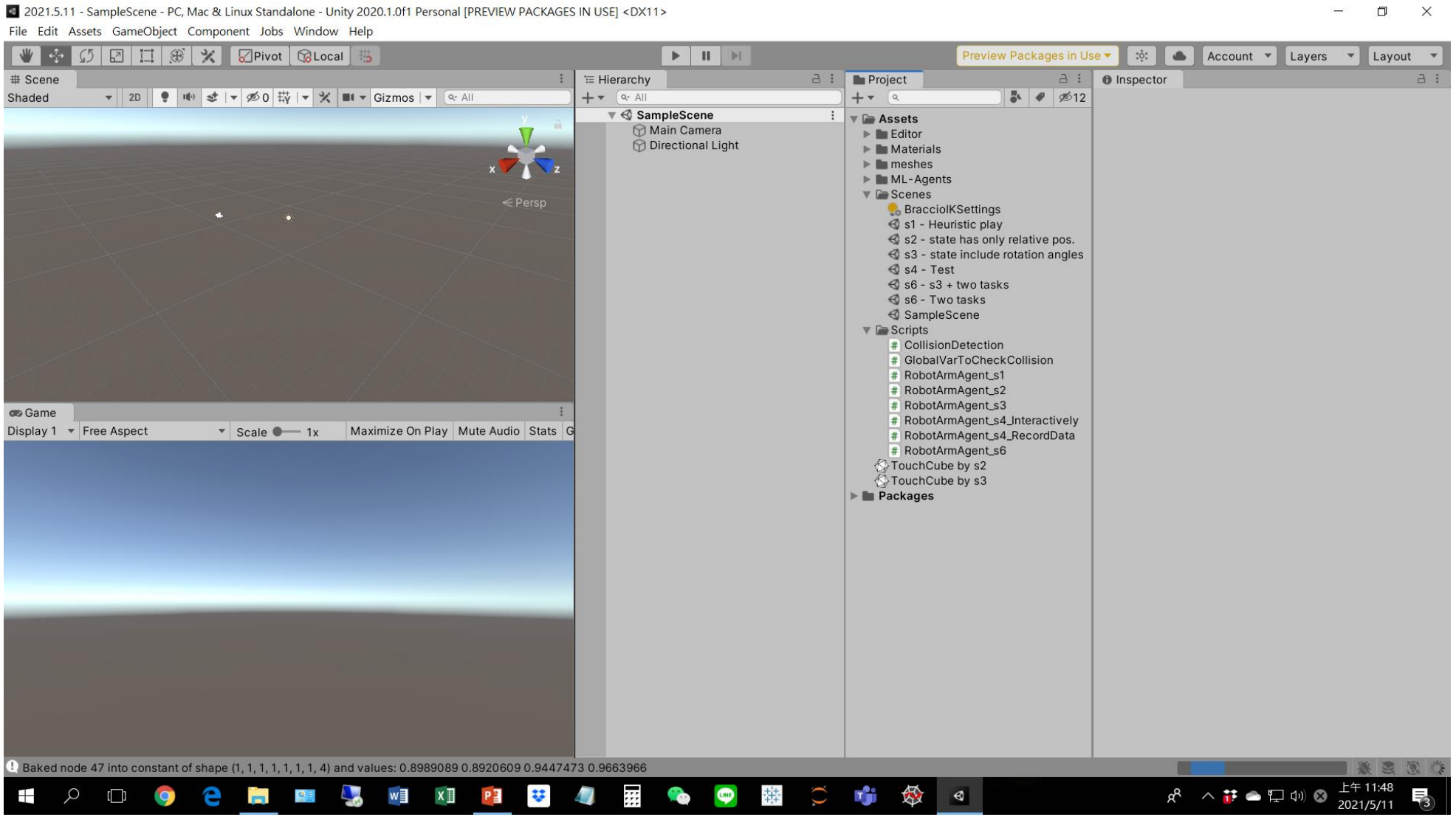
```
cd C:\ml-agents-release_10\config\ppo\results
tensorboard --logdir=1
```

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

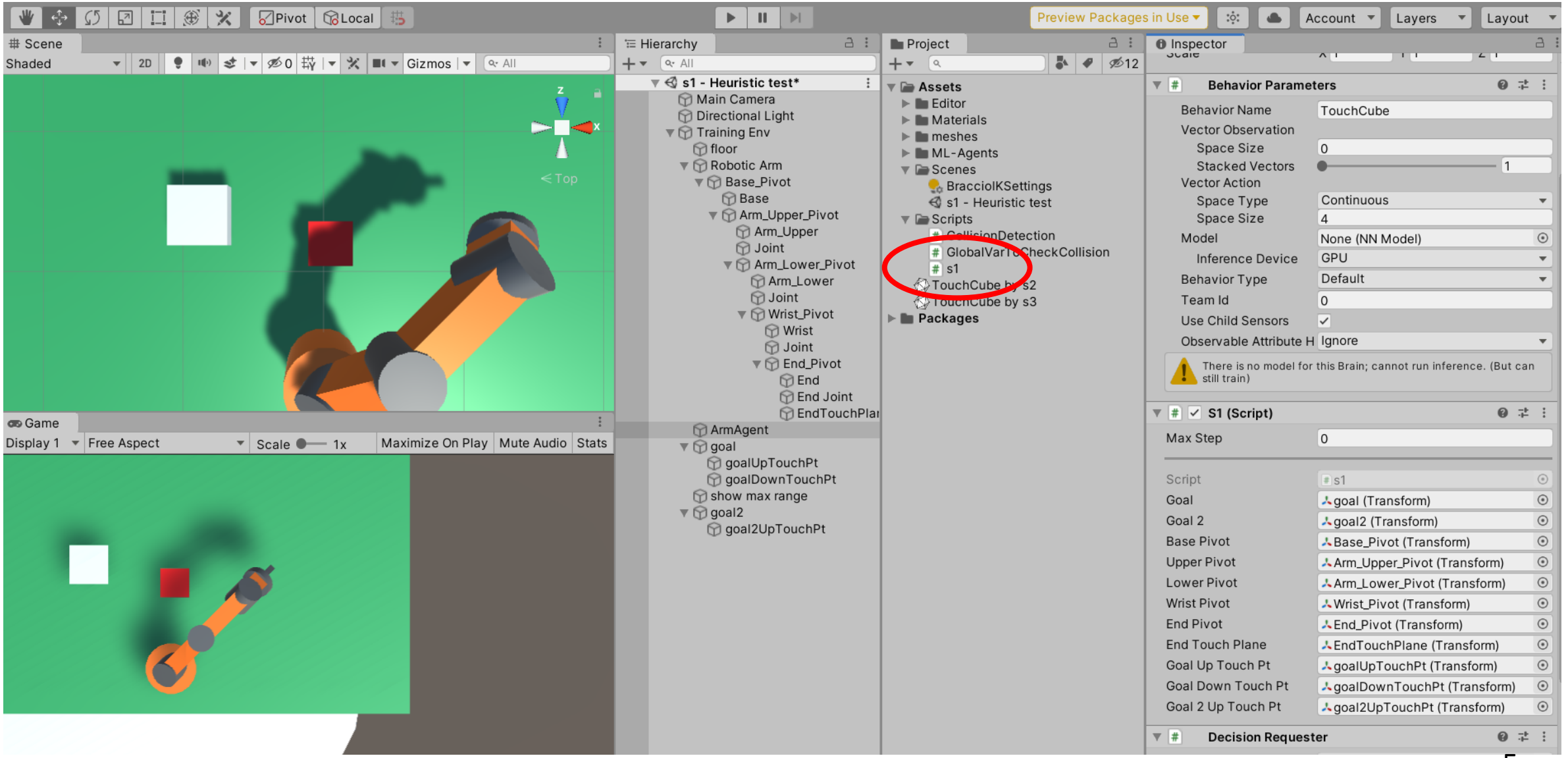




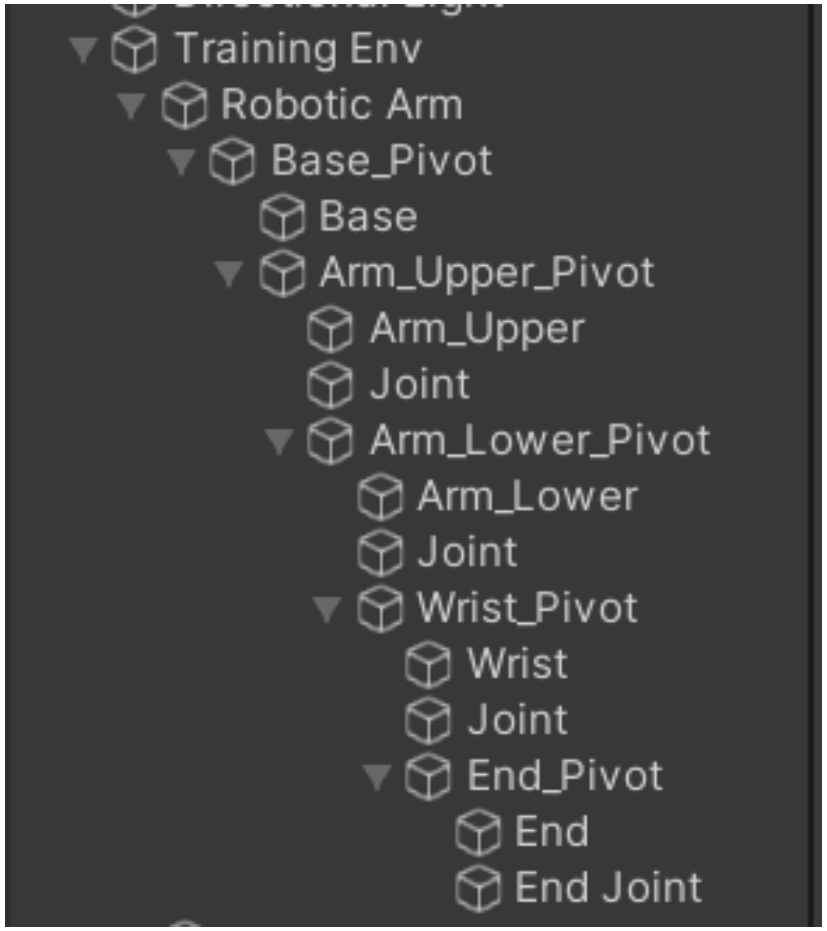
Import Robot arm package to Unity project



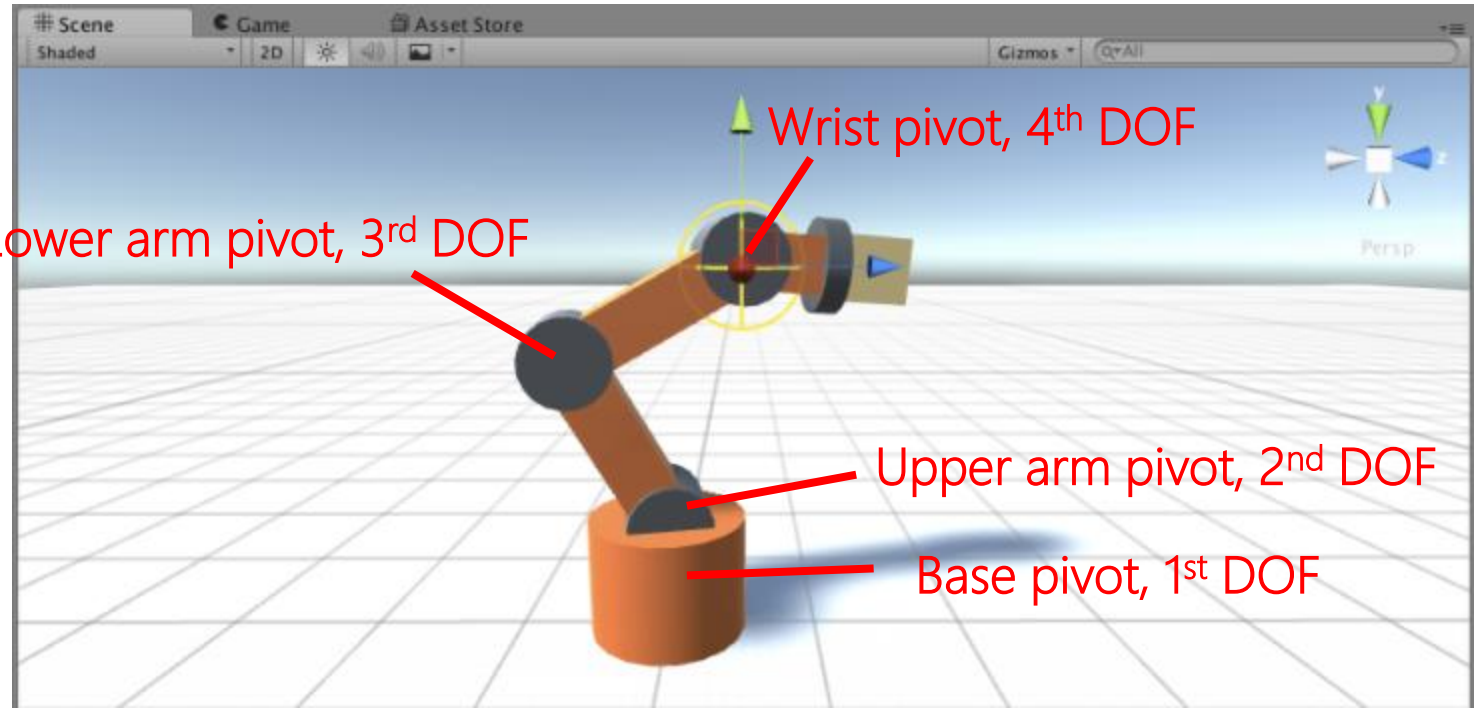
Open scene "s1 - Heuristic play"



This Unity project contains a Braccio robot arm



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

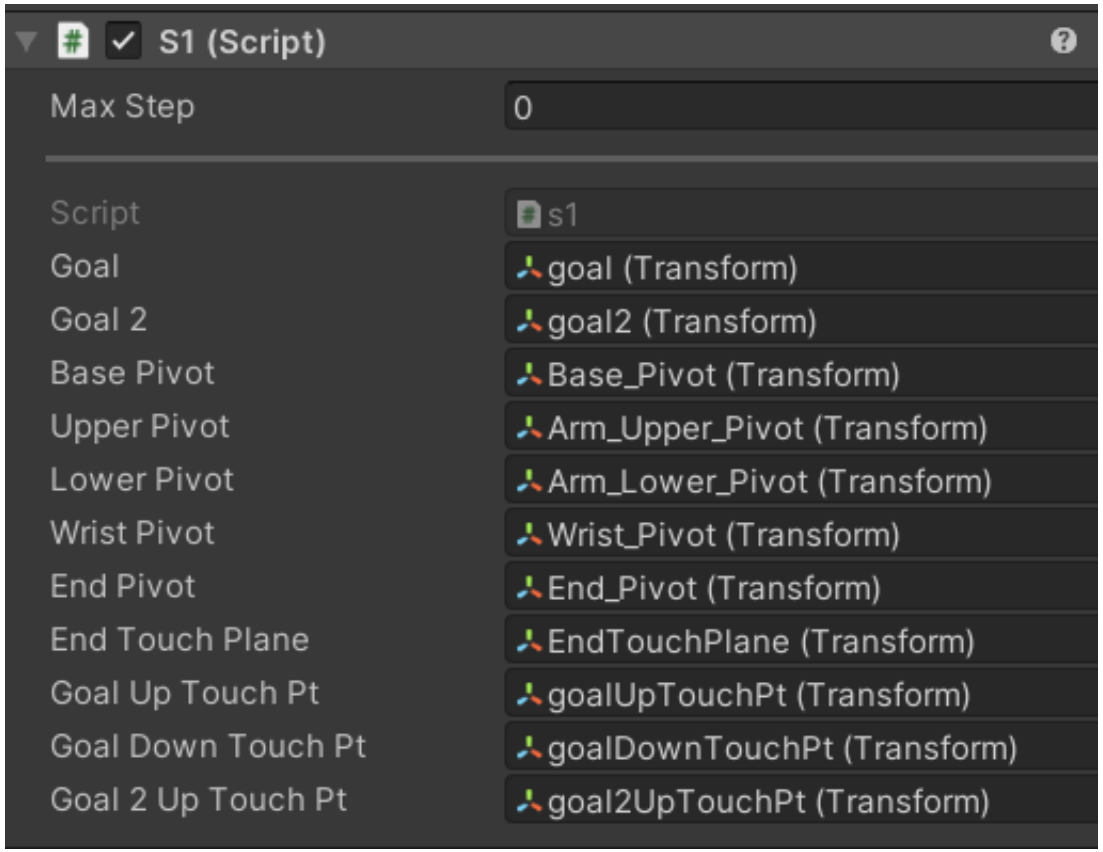


https://github.com/tanyuan/braccio-ik-unity

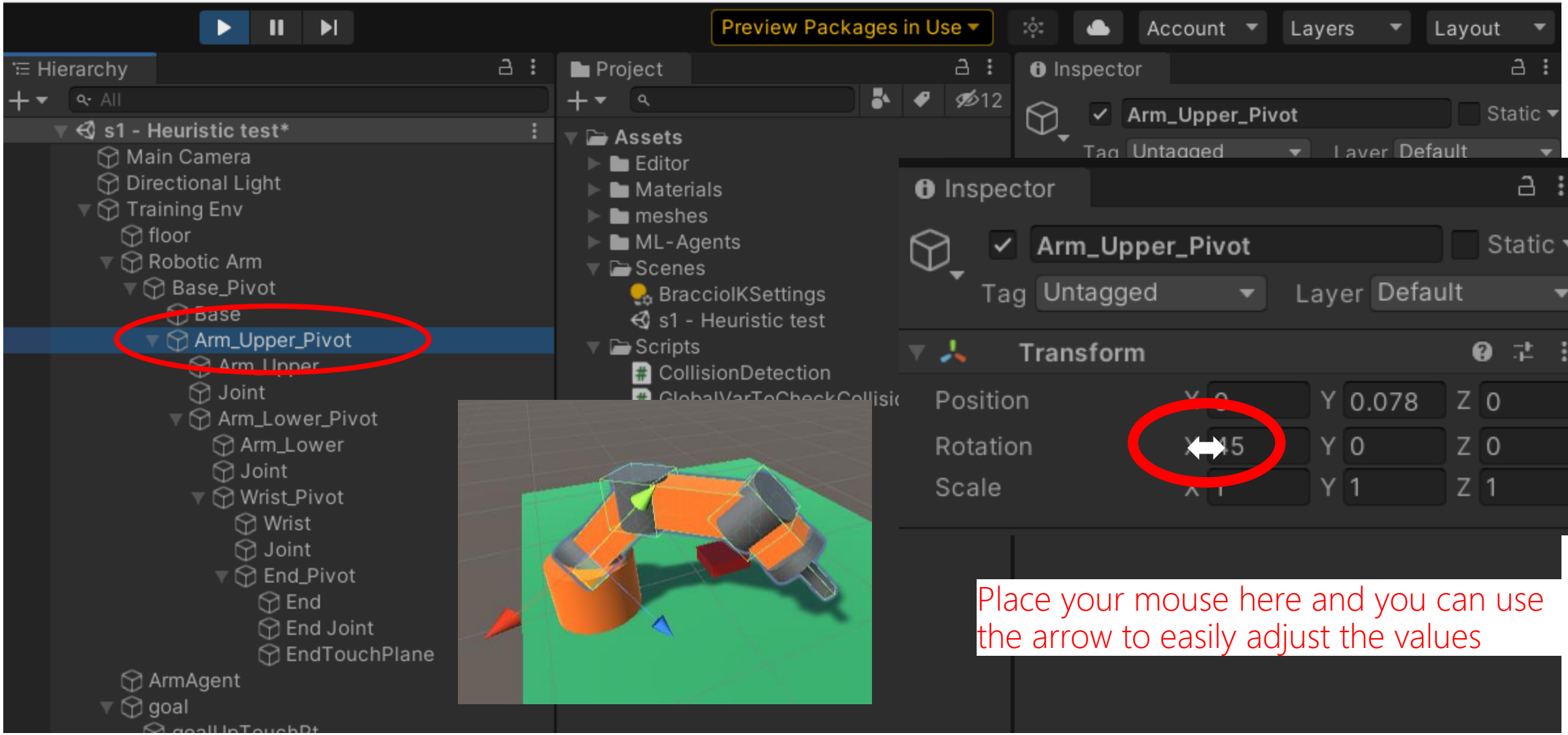
(1) Manually operates the robot arm in VE

Public variables to link agent script with scene objects

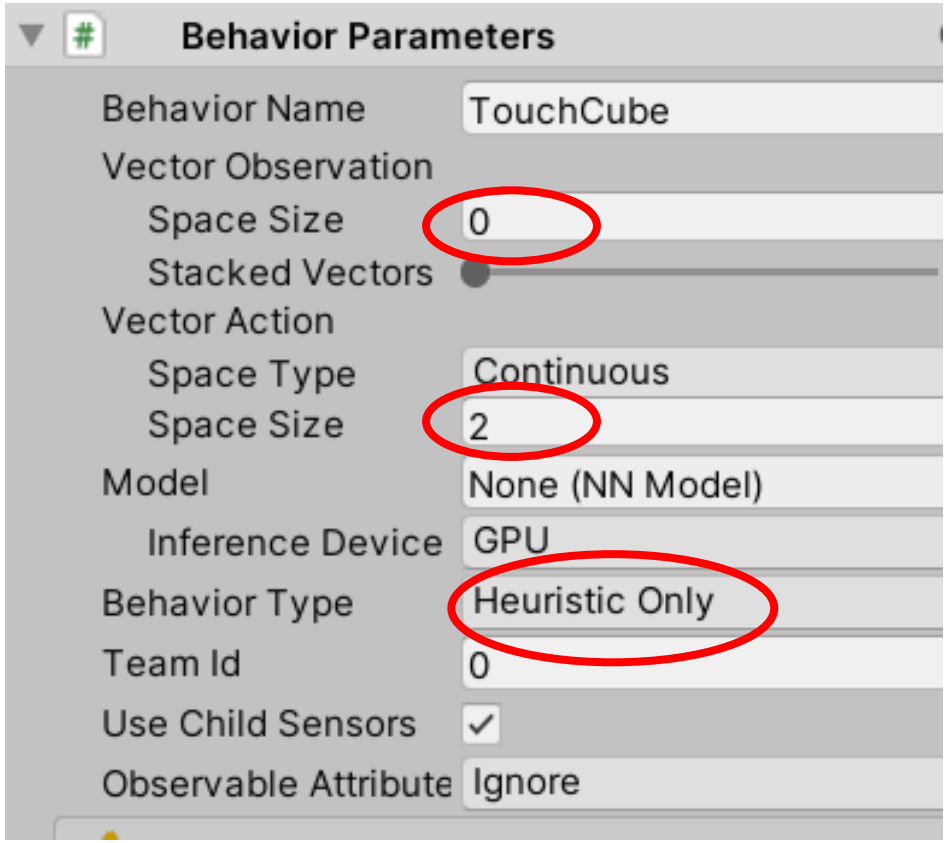
```
public Transform goal, goal2;
public Transform BasePivot, UpperPivot, LowerPivot, WristPivot, EndPivot;
public Transform EndTouchPlane, goalUpTouchPt, goalDownTouchPt, goal2UpTouchPt;
int stage = 1;
```



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



Behavior parameters

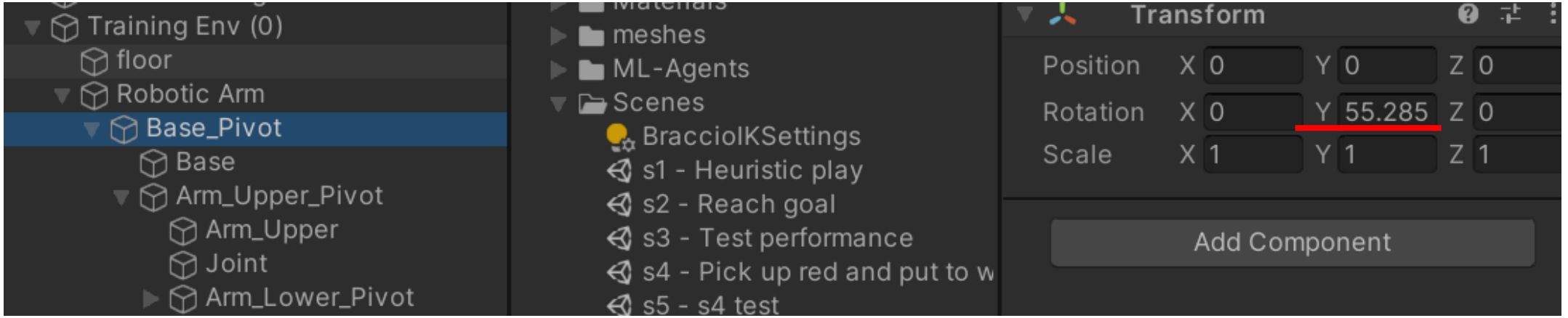


Play and use Up/Down, L/R keys to rotate arm

```
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;
}
```

Collision detections

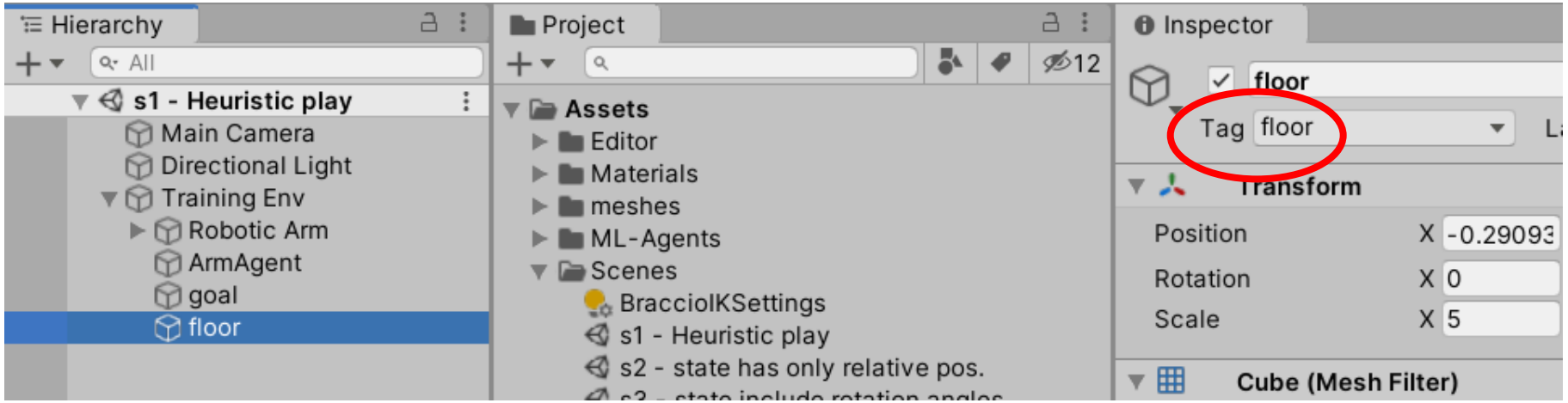
Use OnTriggerEnter/Exist to record collisions between robot arm and other scene objects

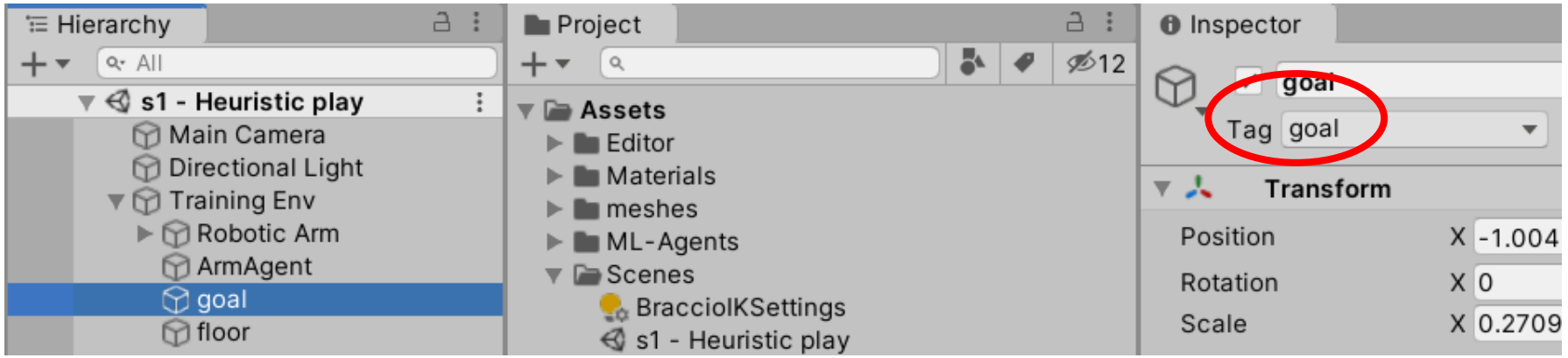
```
public class CollisionDetection : MonoBehaviour
{
    public bool CollisionHappen;
    Unity Message | 0 個參考
    void OnTriggerEnter (Collider other)
    {
        if (other.gameObject.tag == "floor" || o
        {
            CollisionHappen = true;
        }
    }
```

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

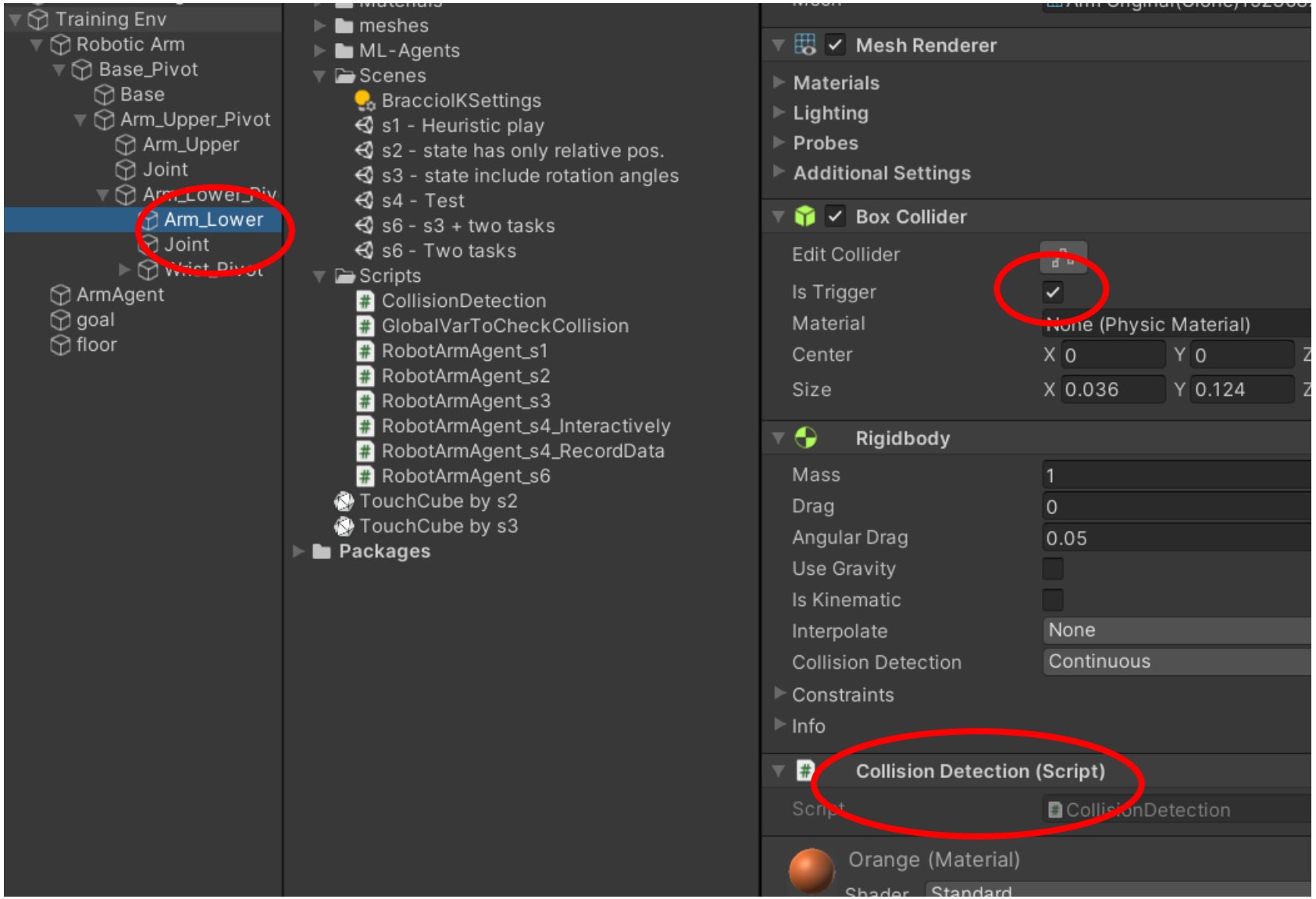
Add tags to floor and goal object

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

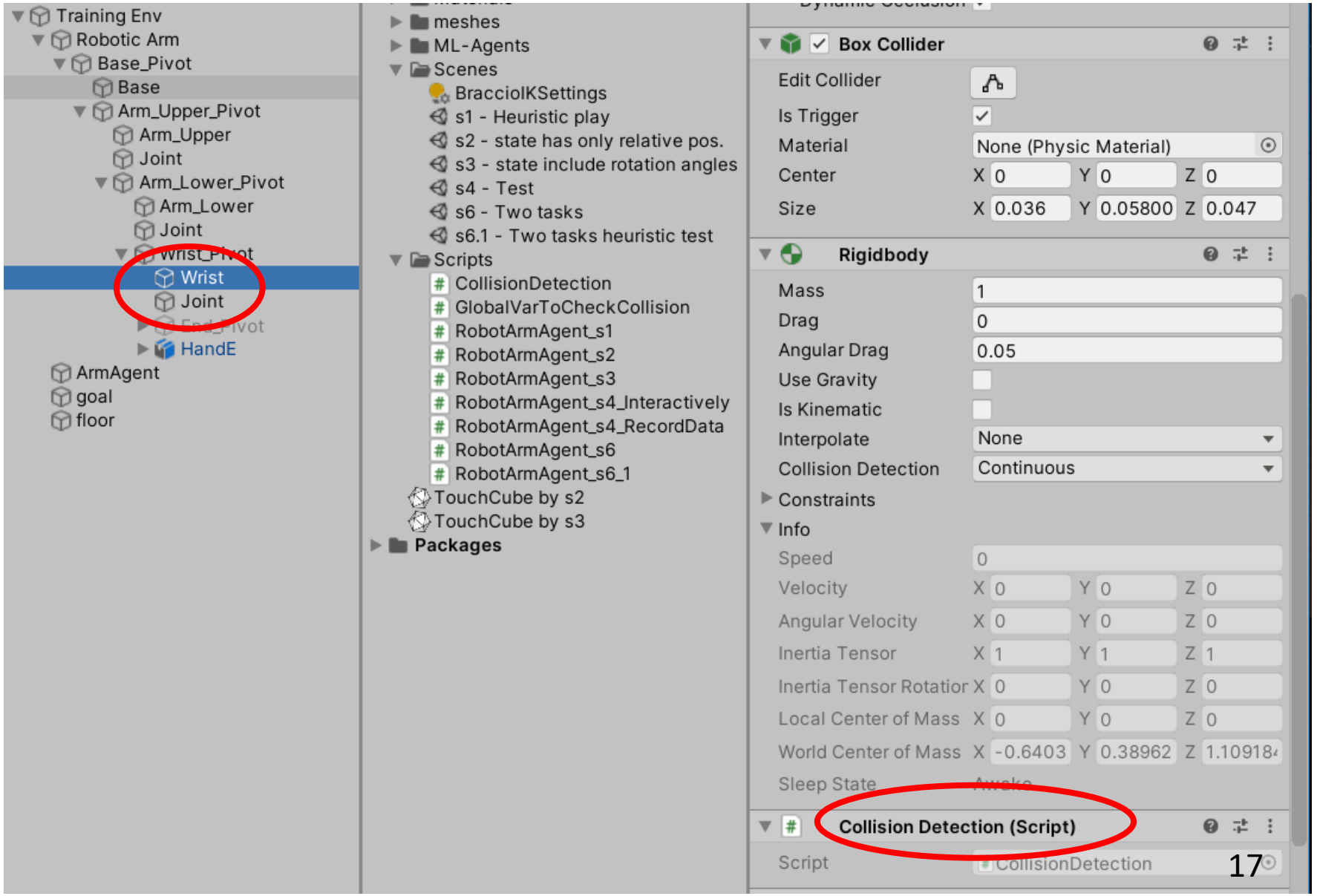




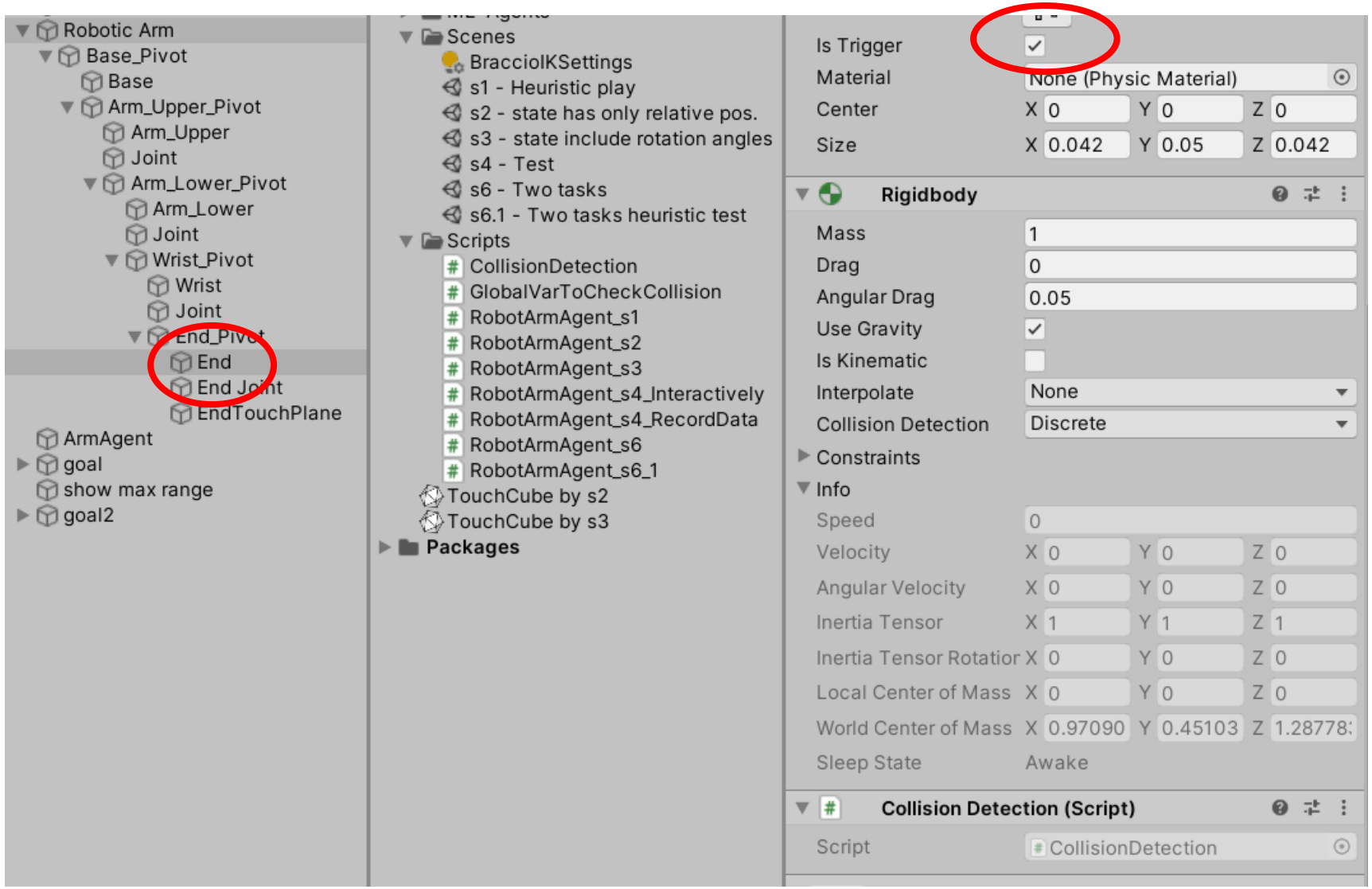
Add trigger collider, Rigid body, and collision detection script to Lower Arm



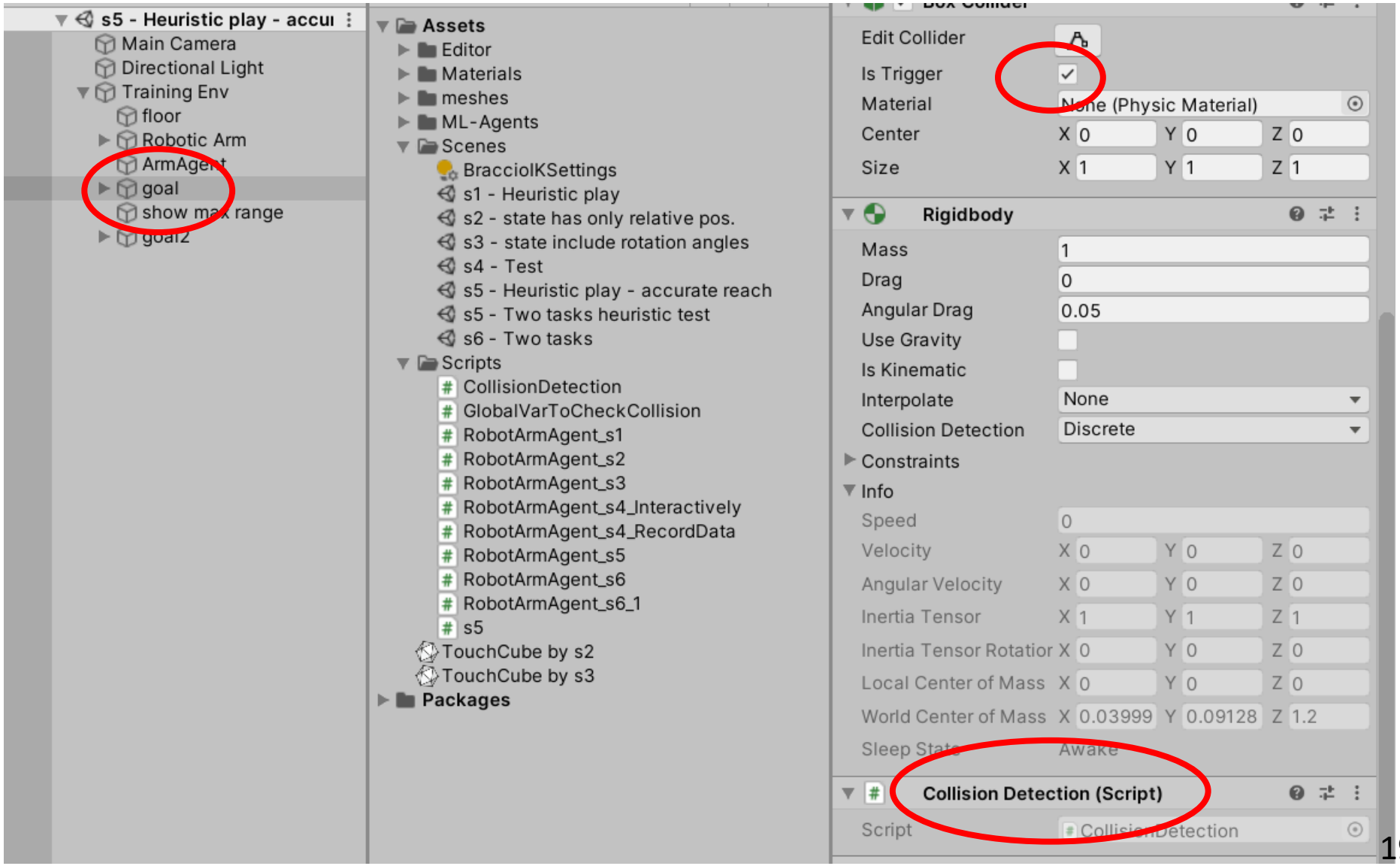
Add trigger collider, Rigid body, and collision detection script to Wrist



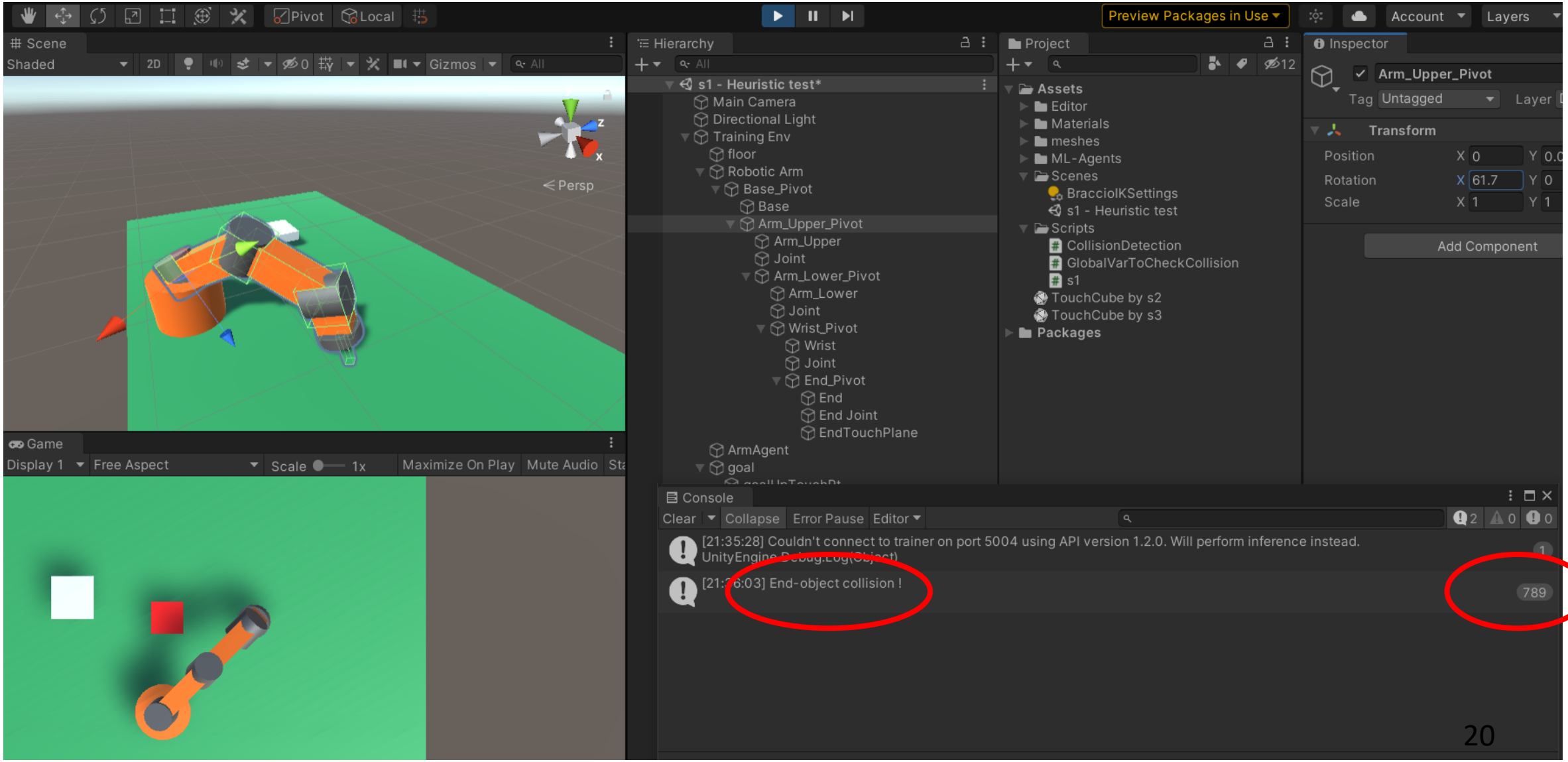
Add trigger collider, Rigid body, and collision detection script to Robot End



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

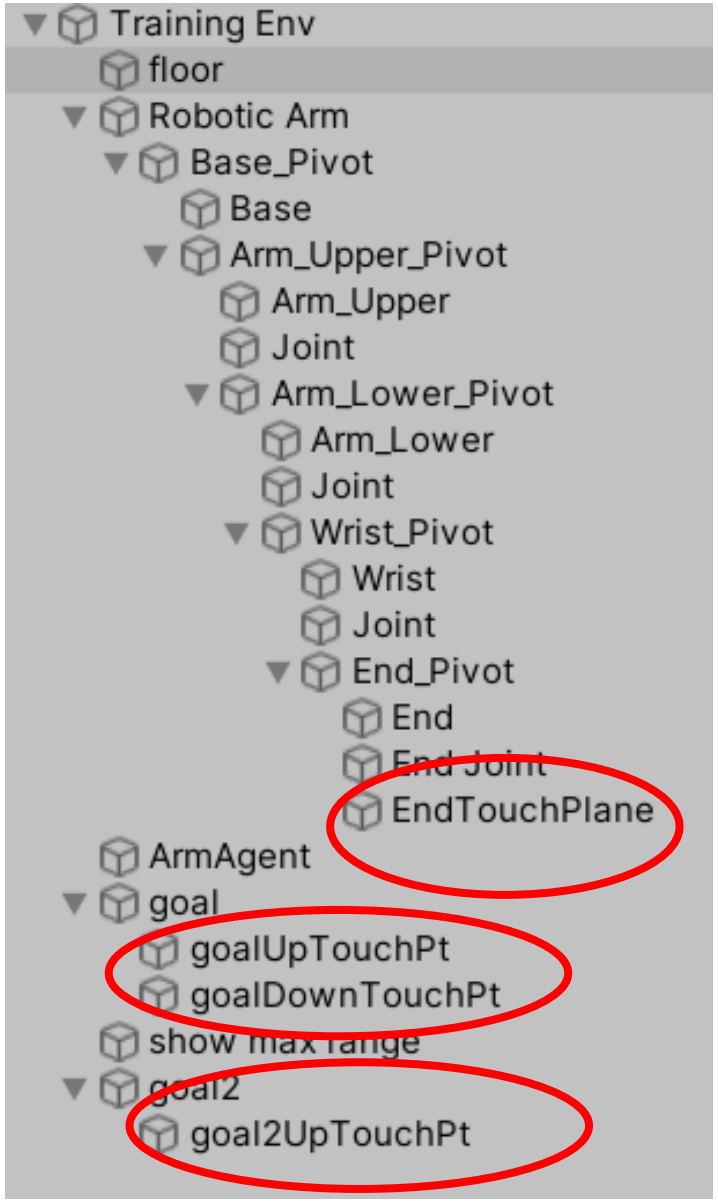


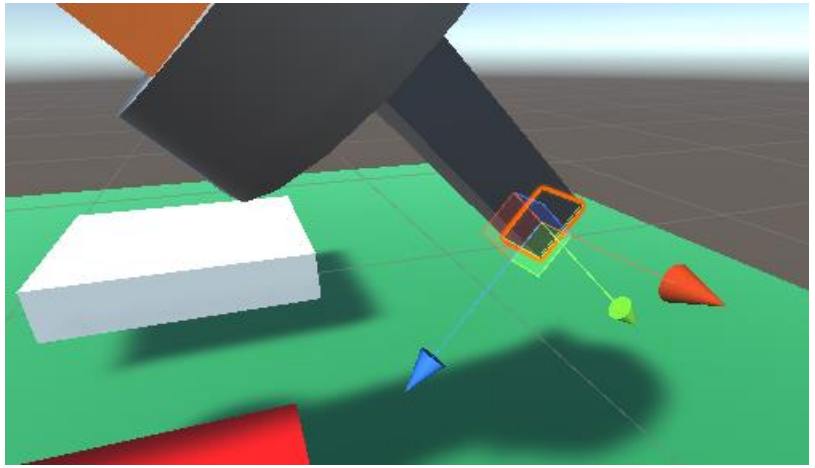
Control the robot arm to collide with the floor or goal and check the error message

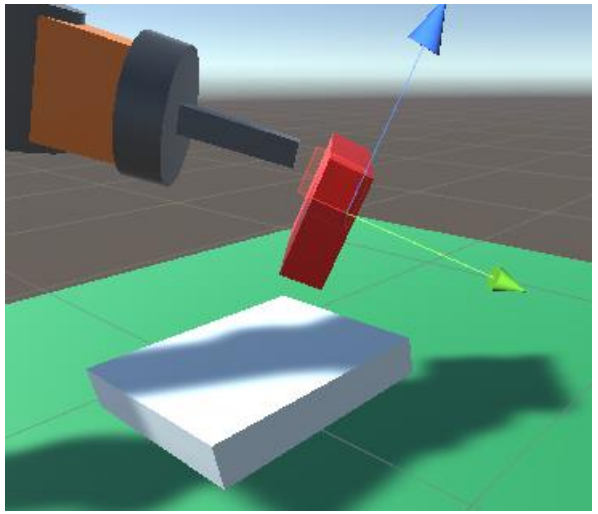


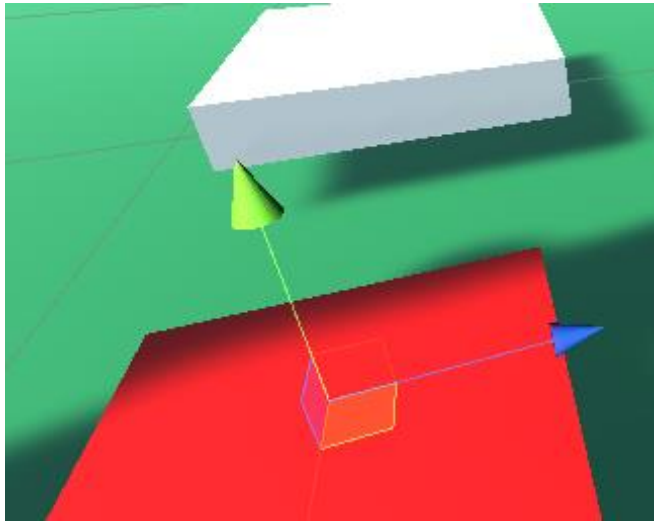
Point-based touch detection

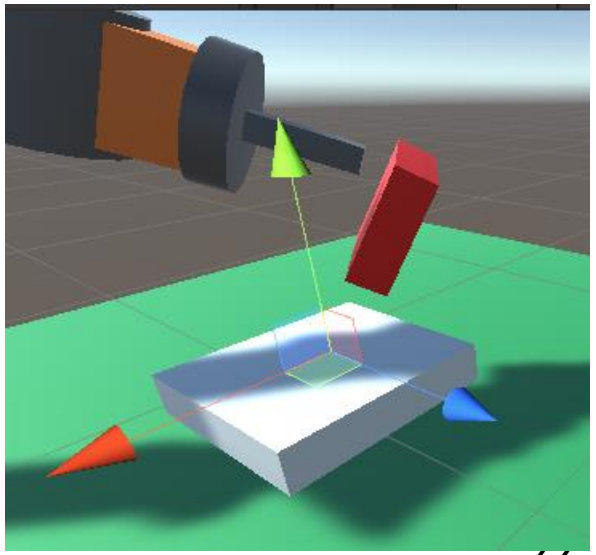
Assign points for reach-detection







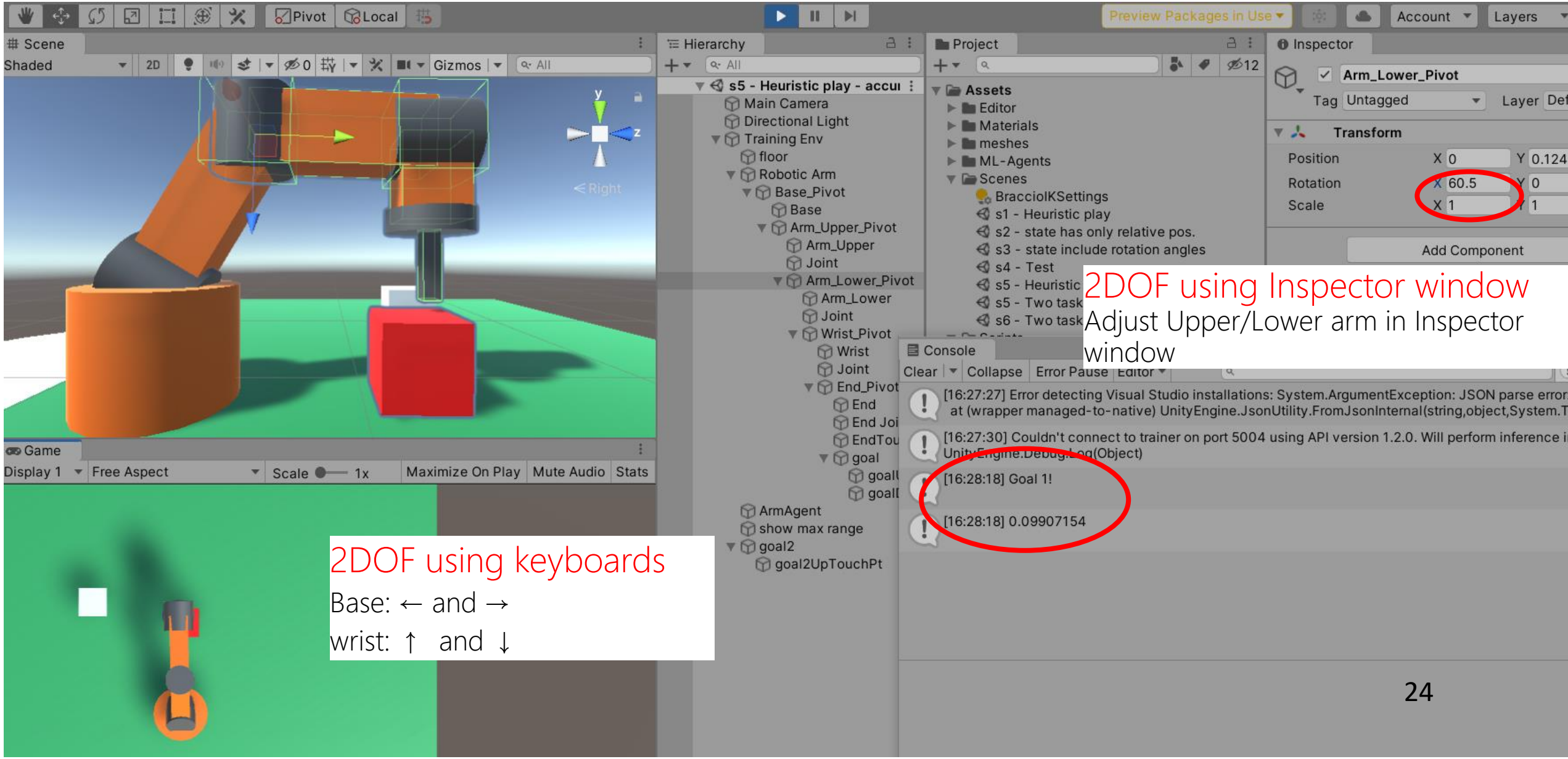




Point-based touch detection

```
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!)



2DOF using keyboards

Base: ← and →

wrist: ↑ and ↓

2DOF using Inspector window

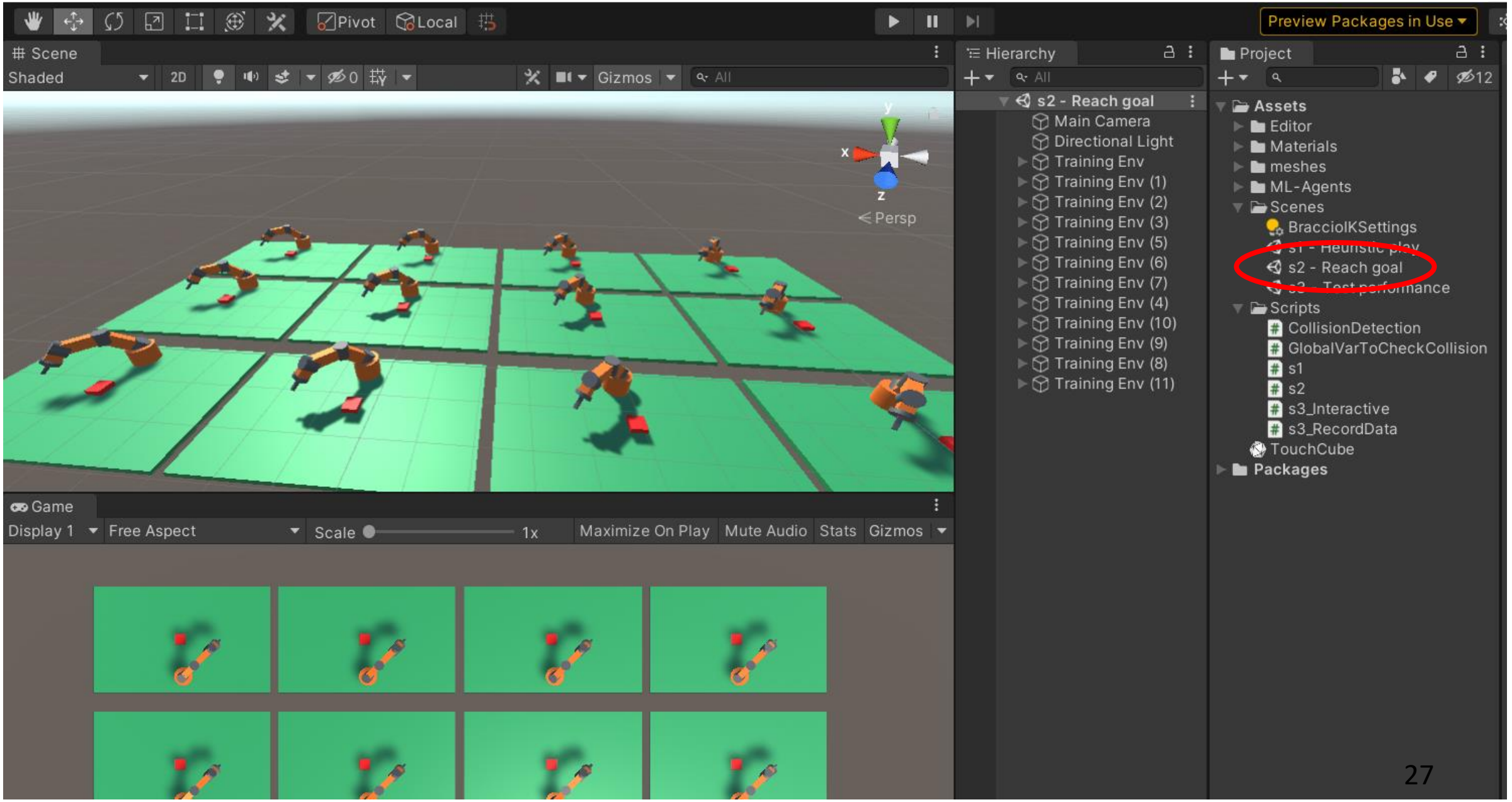
Adjust Upper/Lower arm in Inspector window

(2) Train robot arm to reach goal

HW4(1)

- Describe the training environment ("s2 – Reach goal")
- Describe the agent script (s, a, r)
- Show tensor board plots and discuss your training performance
- Describe your test performance

4. Open "s2 – Reach goal"

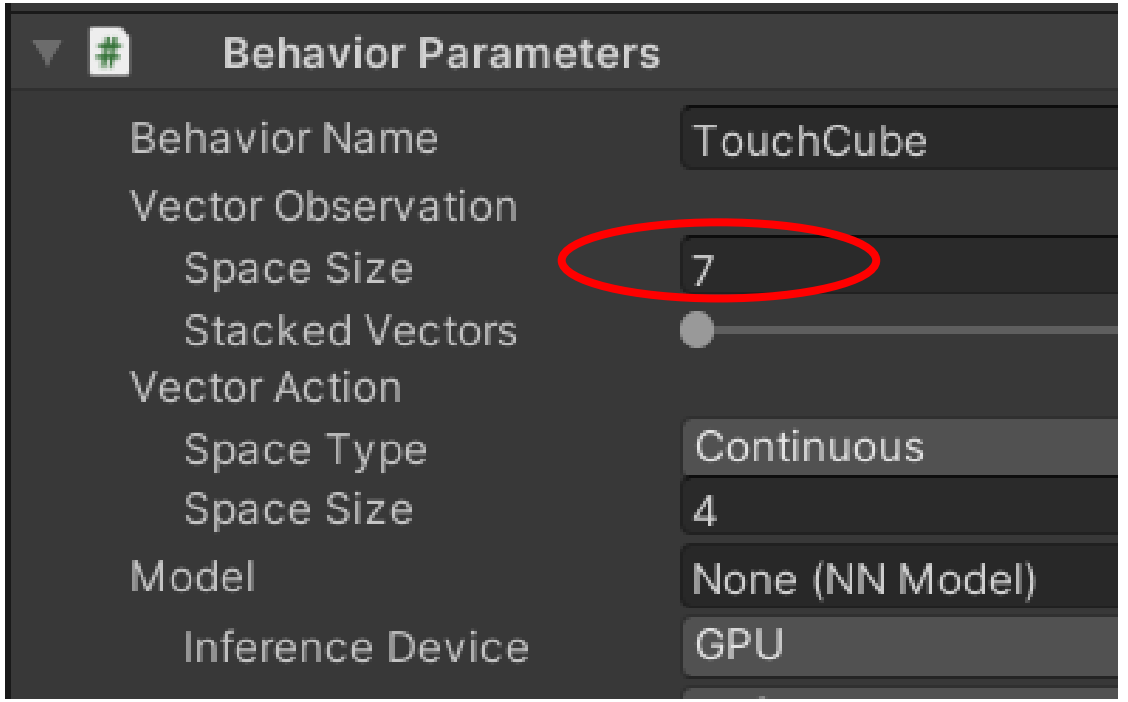


Use polar system to randomly place goals

```
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);
```

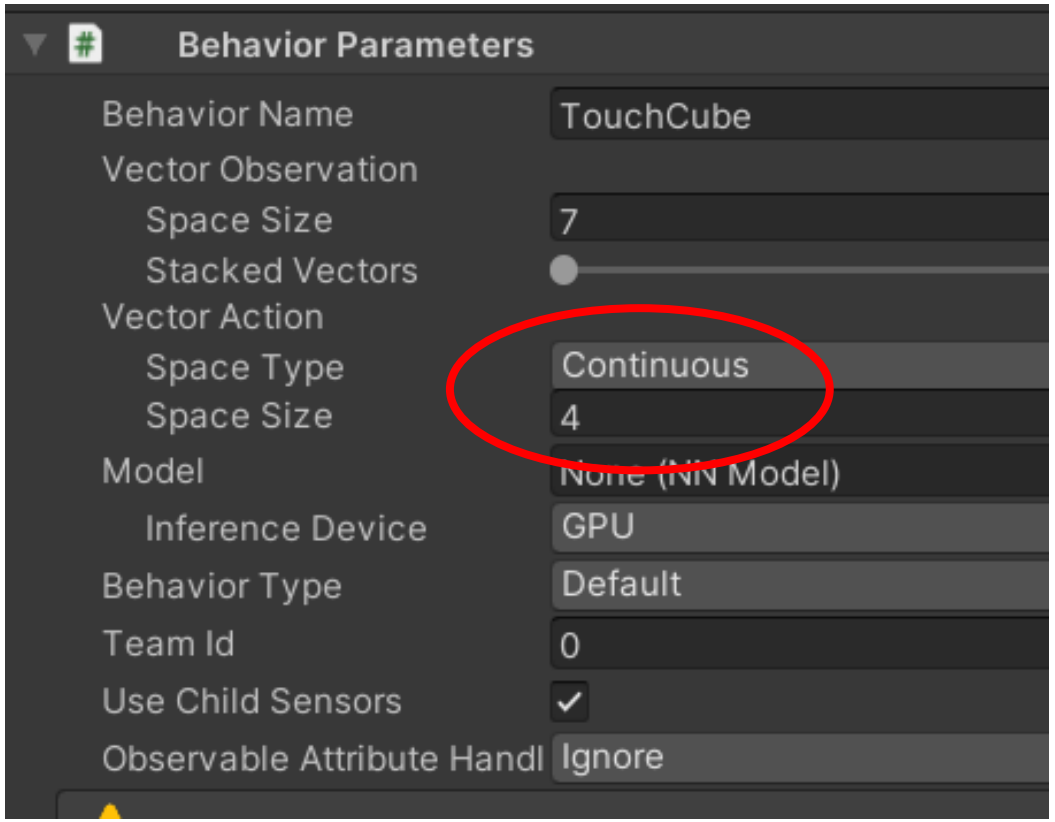
State has 7 variables

```
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

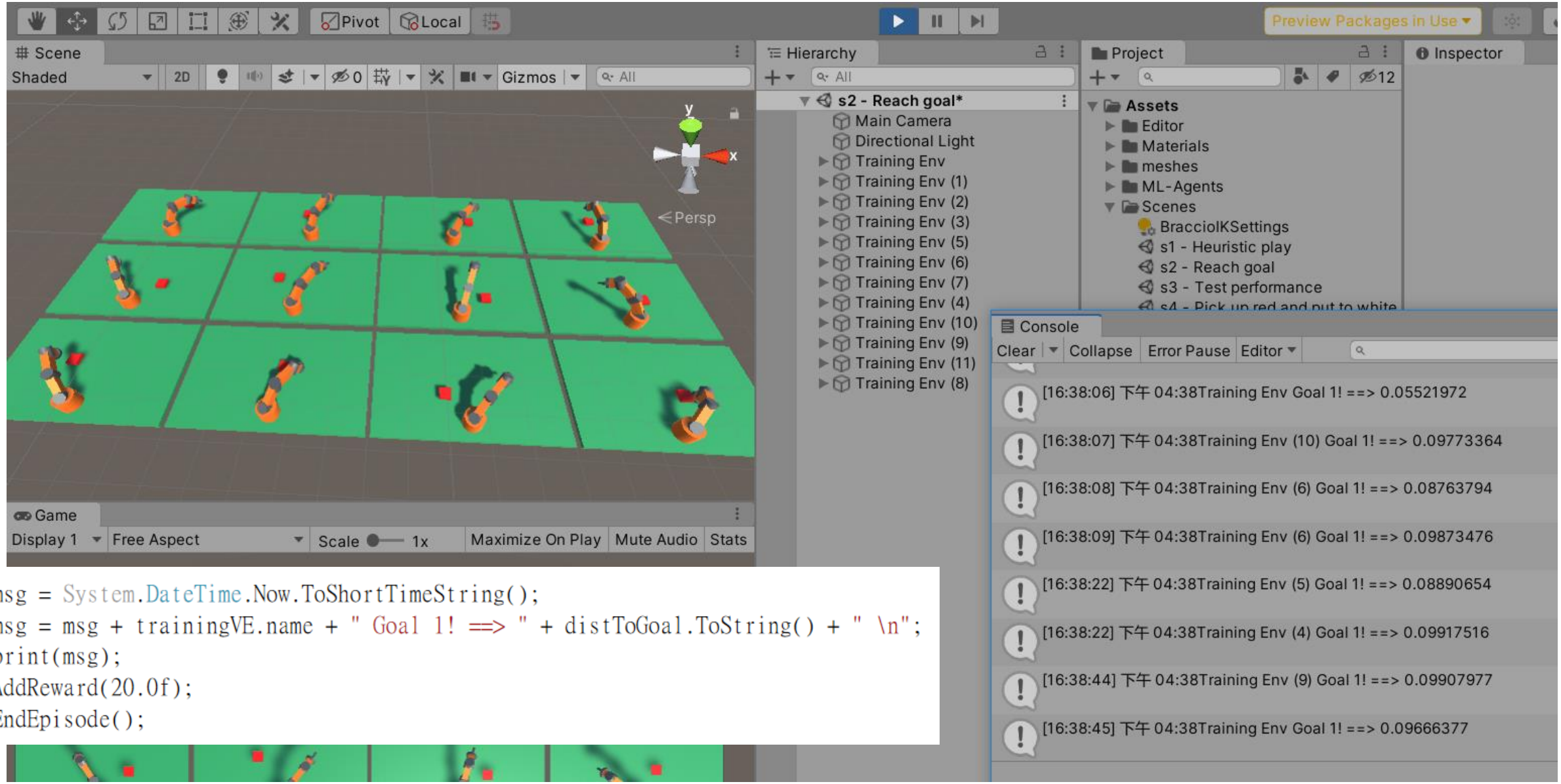
- 0.5 will succeed, but the behavior is too rough
- 0.05 is too difficult and training will fail
- In your HW, try 0.25 or 0.1?

```
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

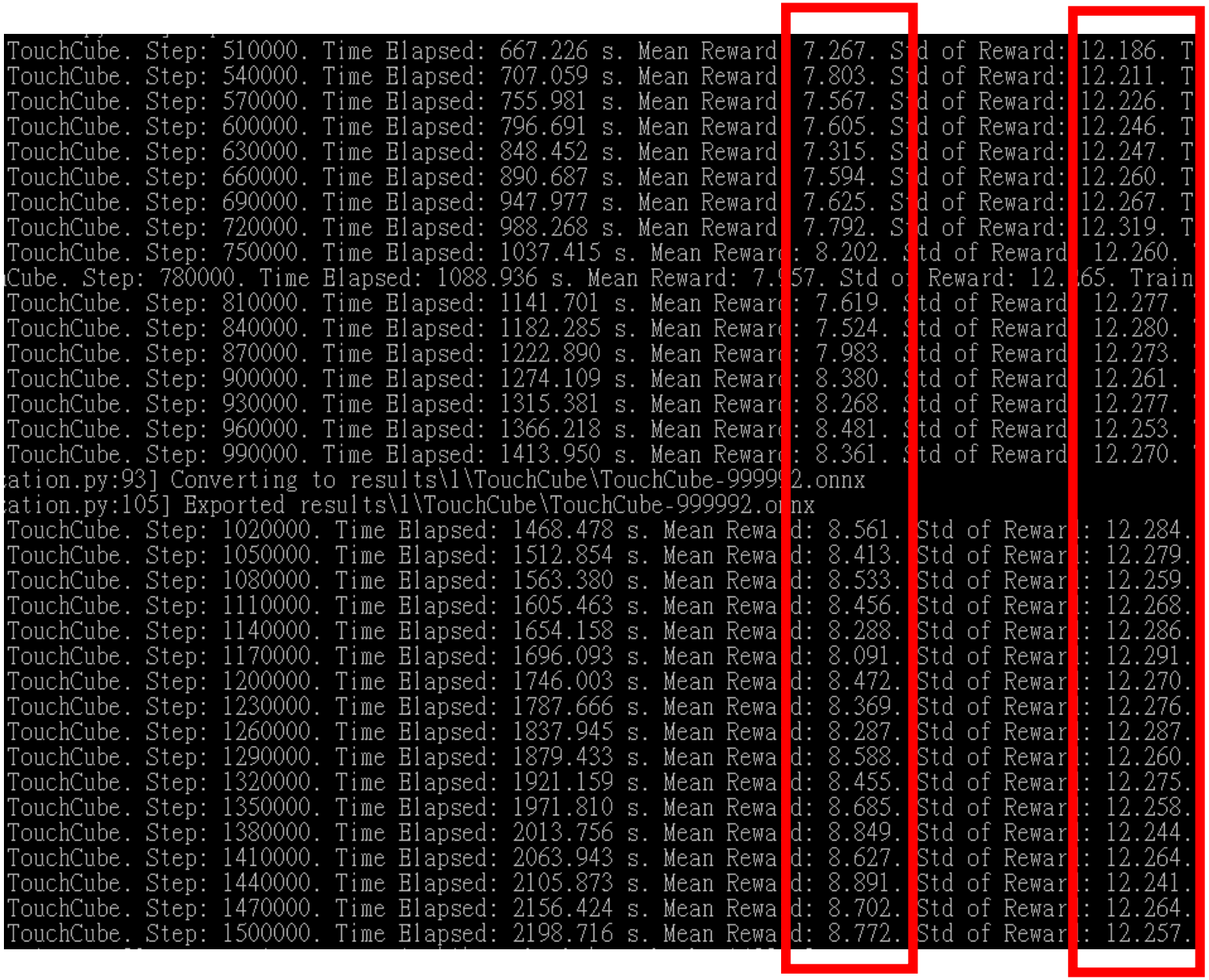
```
TouchCube:
  trainer_type: ppo
  hyperparameters:
    batch_size: 2048
    buffer_size: 20480
    learning_rate: 0.0003
    beta: 0.001
    epsilon: 0.2
    lambd: 0.95
    num_epoch: 3
    learning_rate_schedule
  network_settings:
    normalize: true
    hidden_units: 512
    num_layers: 3
   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

Print environment number that reaches goal



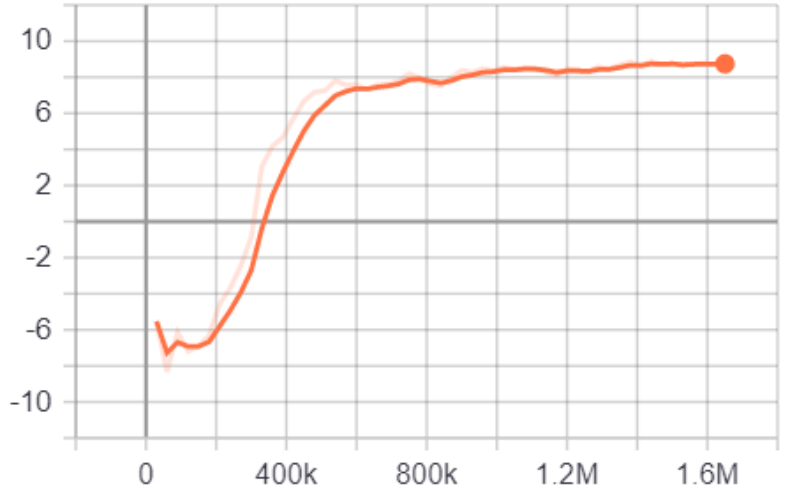

```
msg = System.DateTime.Now.ToShortTimeString();
msg = msg + trainingVE.name + " Goal 1! ==> " + distToGoal.ToString() + " \n";
print(msg);
AddReward(20.0f);
EndEpisode();
```

Results after 1.7M steps, looks promising

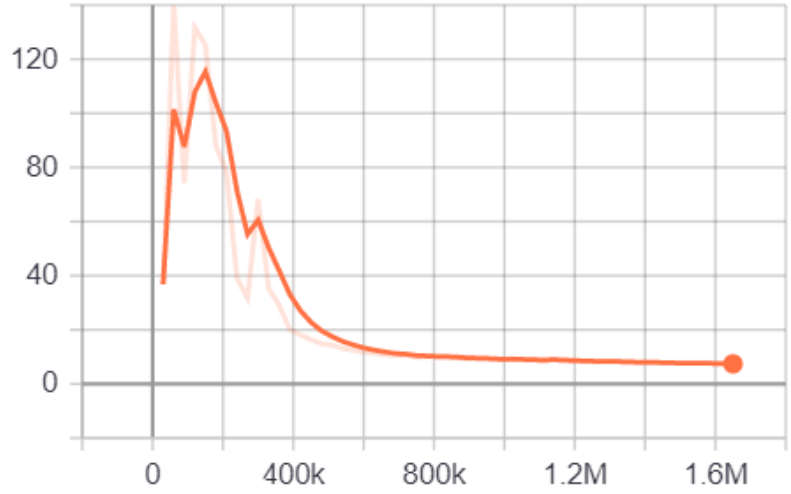


Results after 1.7M steps, looks promising

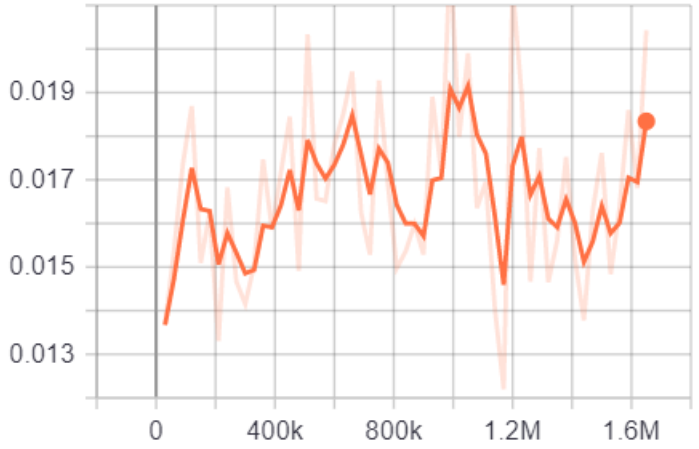
Cumulative Reward
tag: Environment/Cumulative Reward



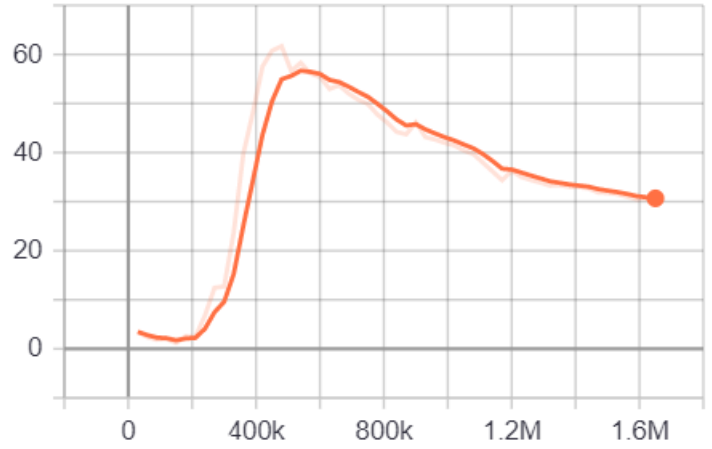
Episode Length
tag: Environment/Episode Length



Policy Loss
tag: Losses/Policy Loss



Value Loss
tag: Losses/Value Loss

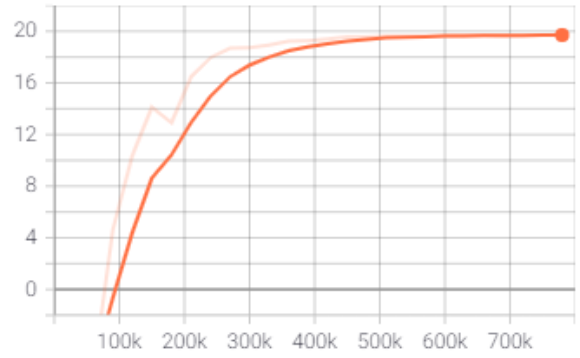


Training results from students

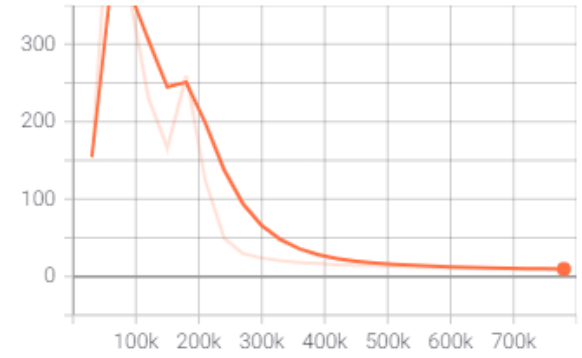
Typical successful training experiences

point reach threshold = 0.25

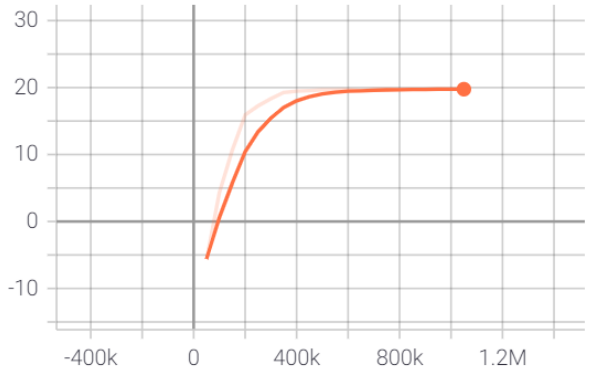
Cumulative Reward
tag: Environment/Cumulative Reward



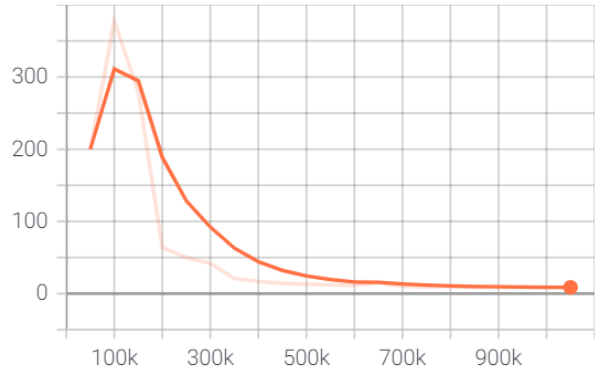
Episode Length
tag: Environment/Episode Length



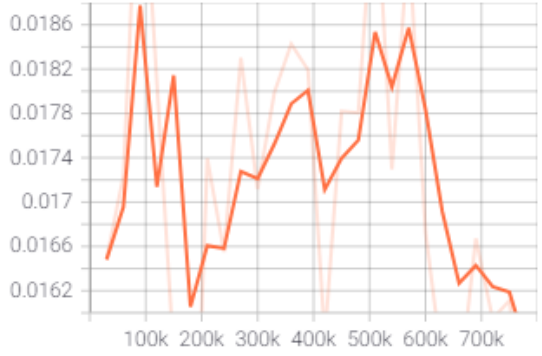
Cumulative Reward
tag: Environment/Cumulative Reward



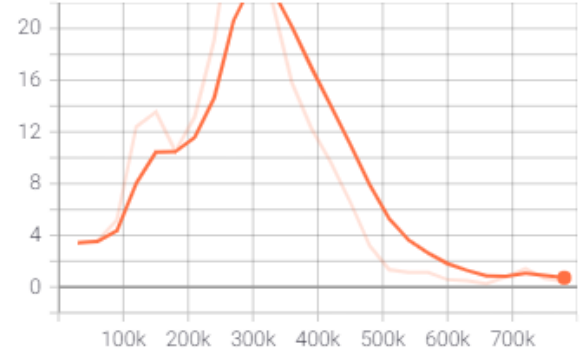
Episode Length
tag: Environment/Episode Length



Policy Loss
tag: Losses/Policy Loss



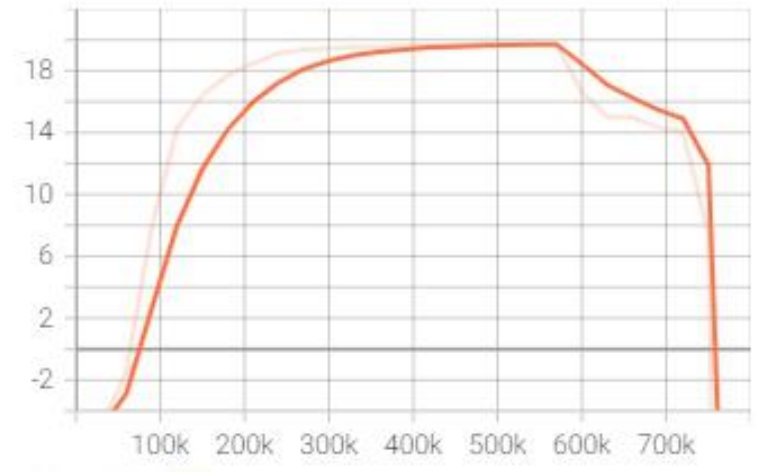
Value Loss
tag: Losses/Value Loss



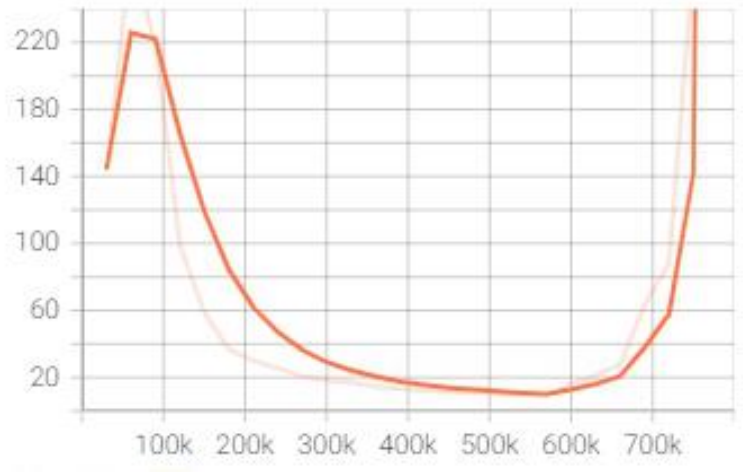
A disaster training experience

point reach threshold = 0.25

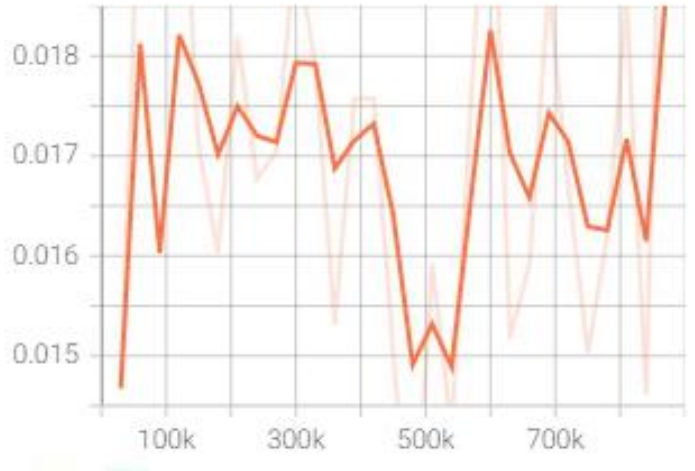
Cumulative Reward
tag: Environment/Cumulative Reward



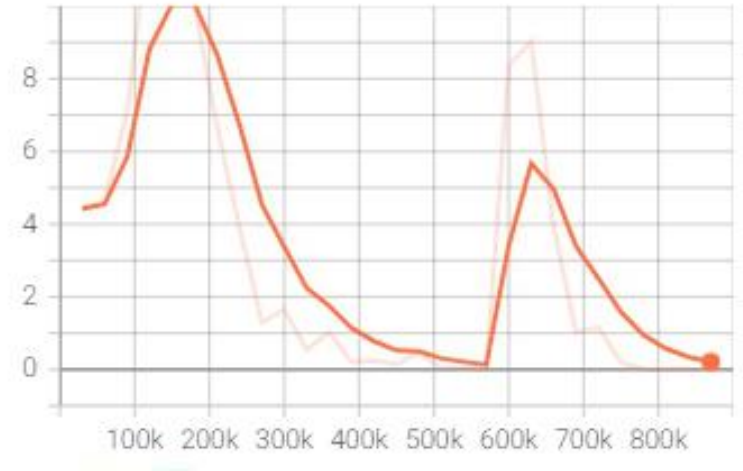
Episode Length
tag: Environment/Episode Length



Policy Loss
tag: Losses/Policy Loss

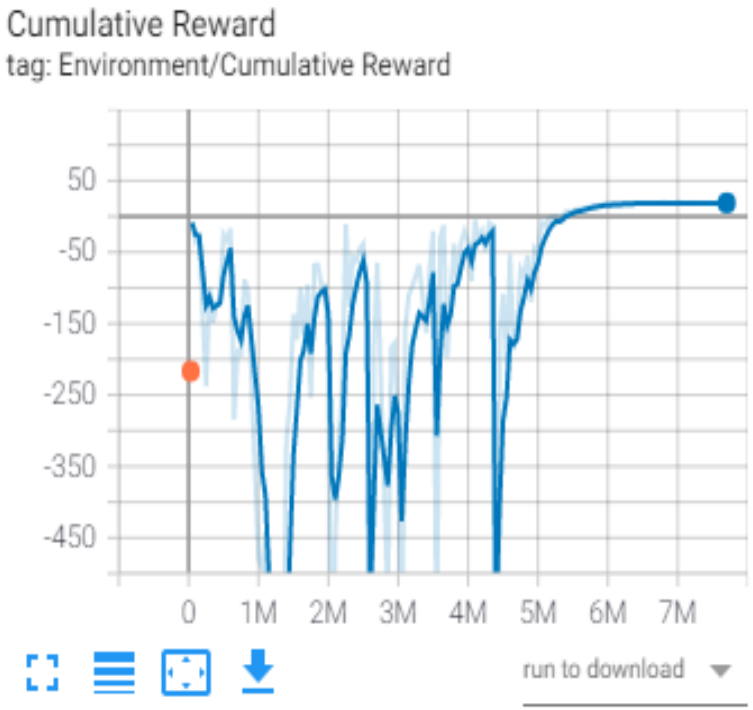


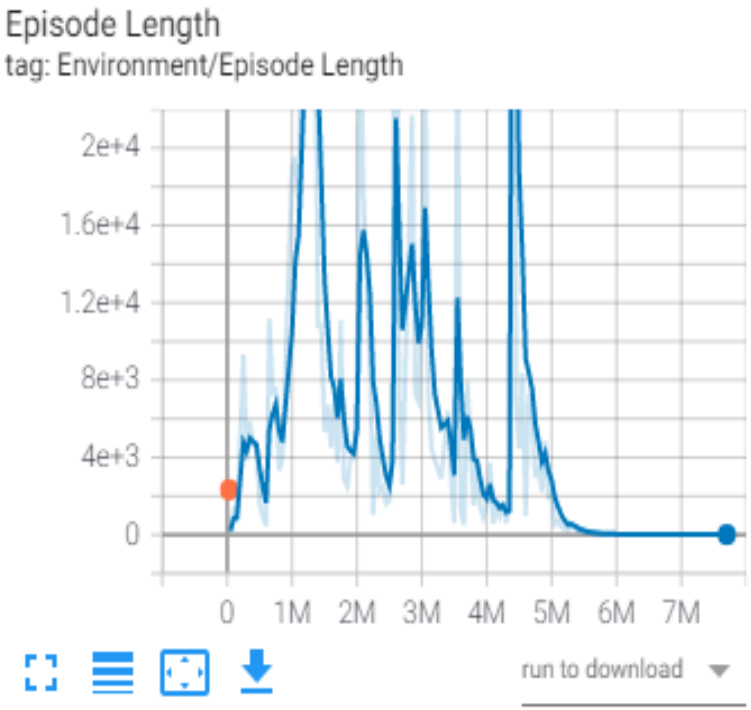
Value Loss
tag: Losses/Value Loss

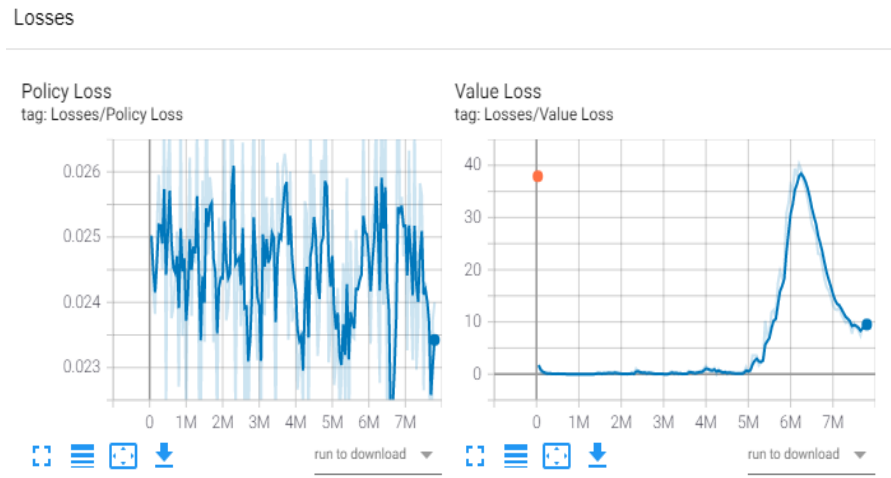


A patient training experience

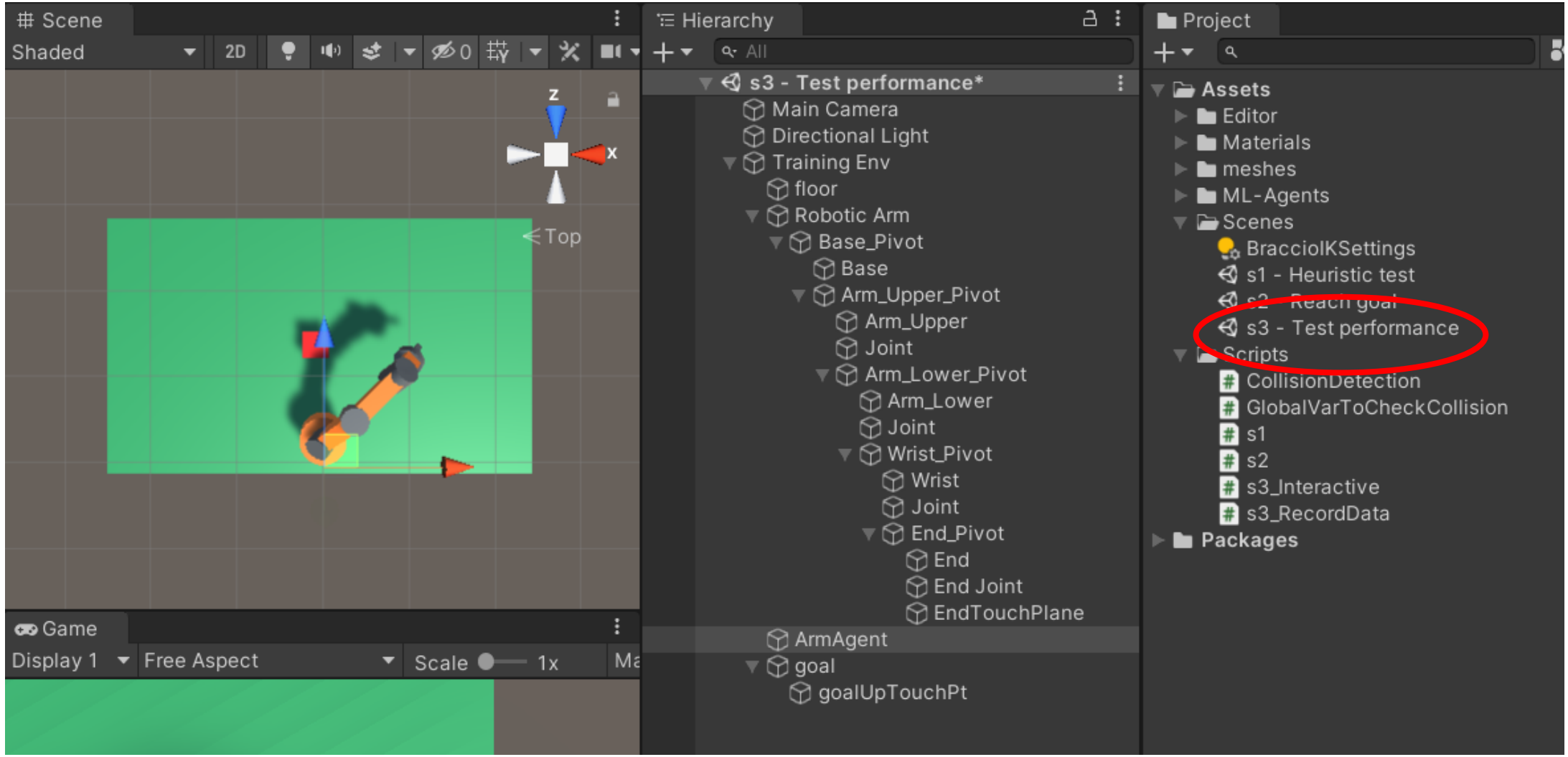
In this training, the point reach threshold is set to be 0.1, which makes it very difficult for AI to learn to reach goal. For the first 5M, the reward performance looks un-hope and most people will give up. But luckily the reward goes to maximum and become steady after 6M.



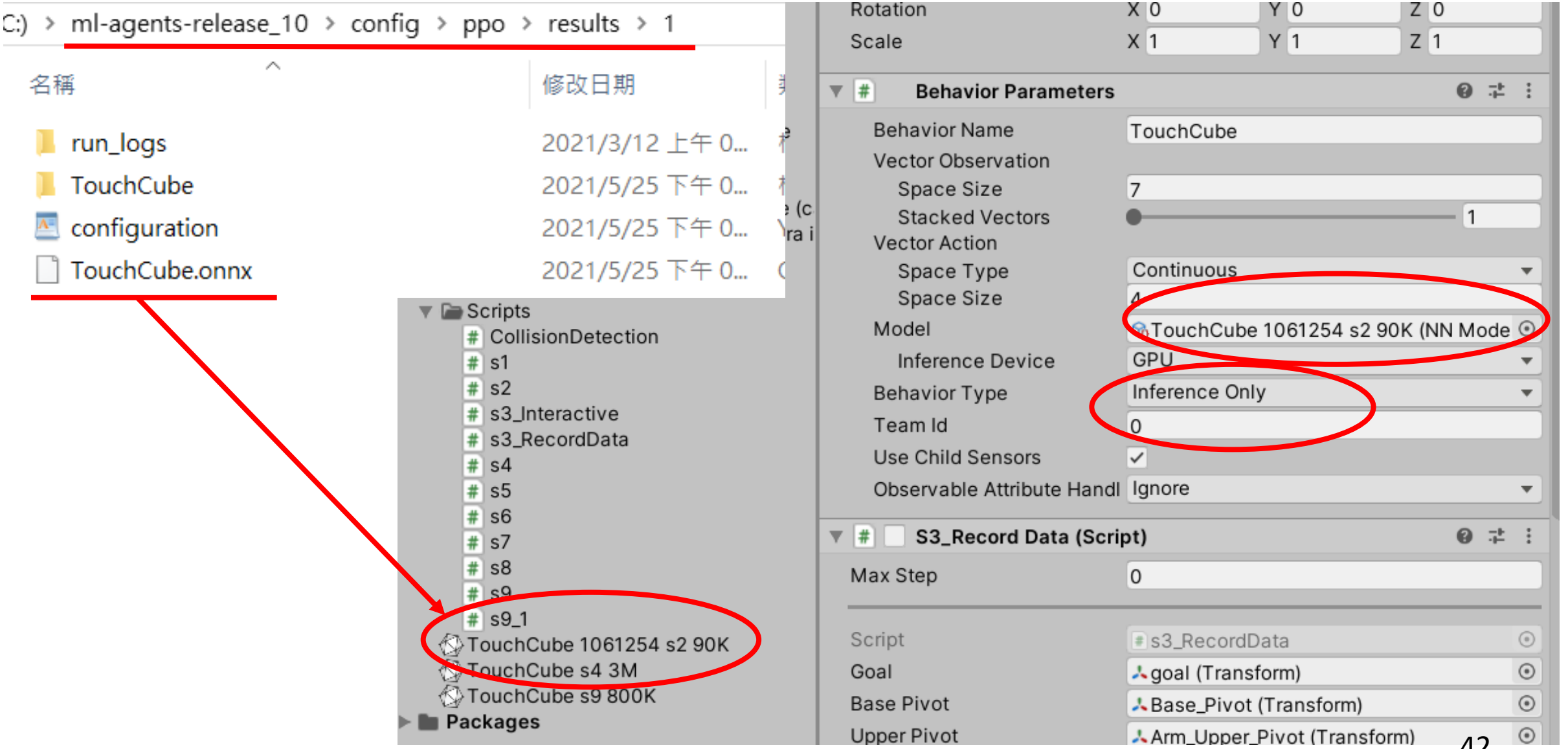




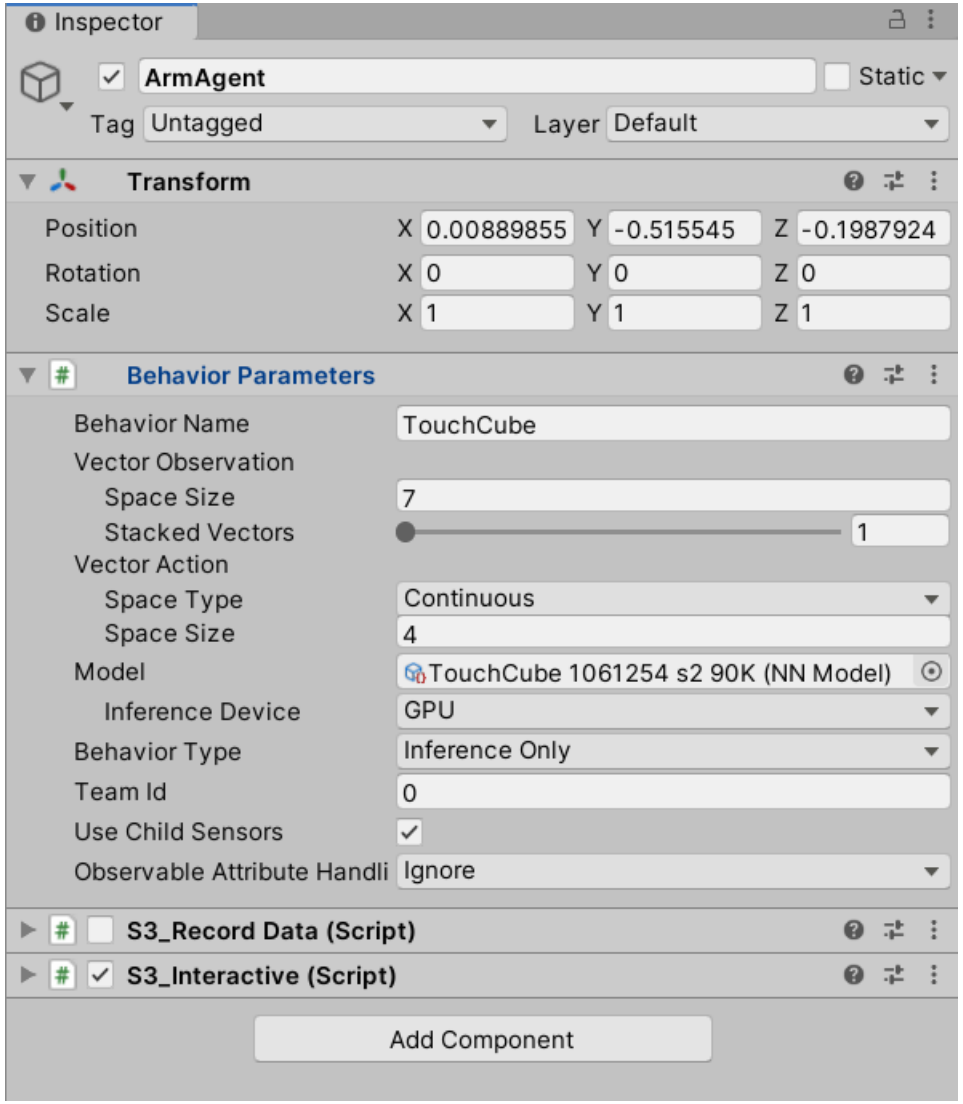
Open "s3 – Test performance"



Assign trained NN

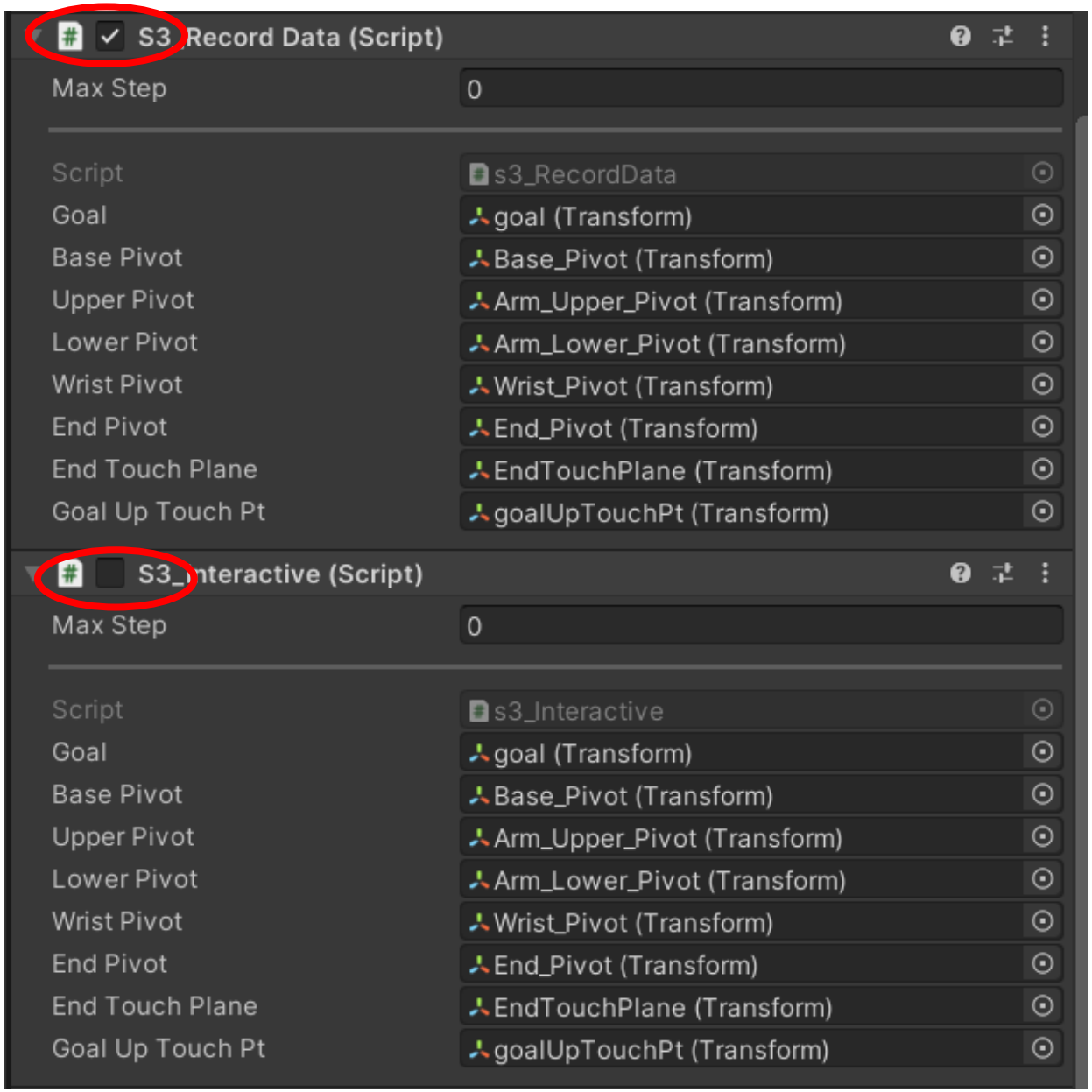


No decision requester in Behavior parameter



2 types of tests – (1) data recording

Uncheck interactive test



(1) data recording

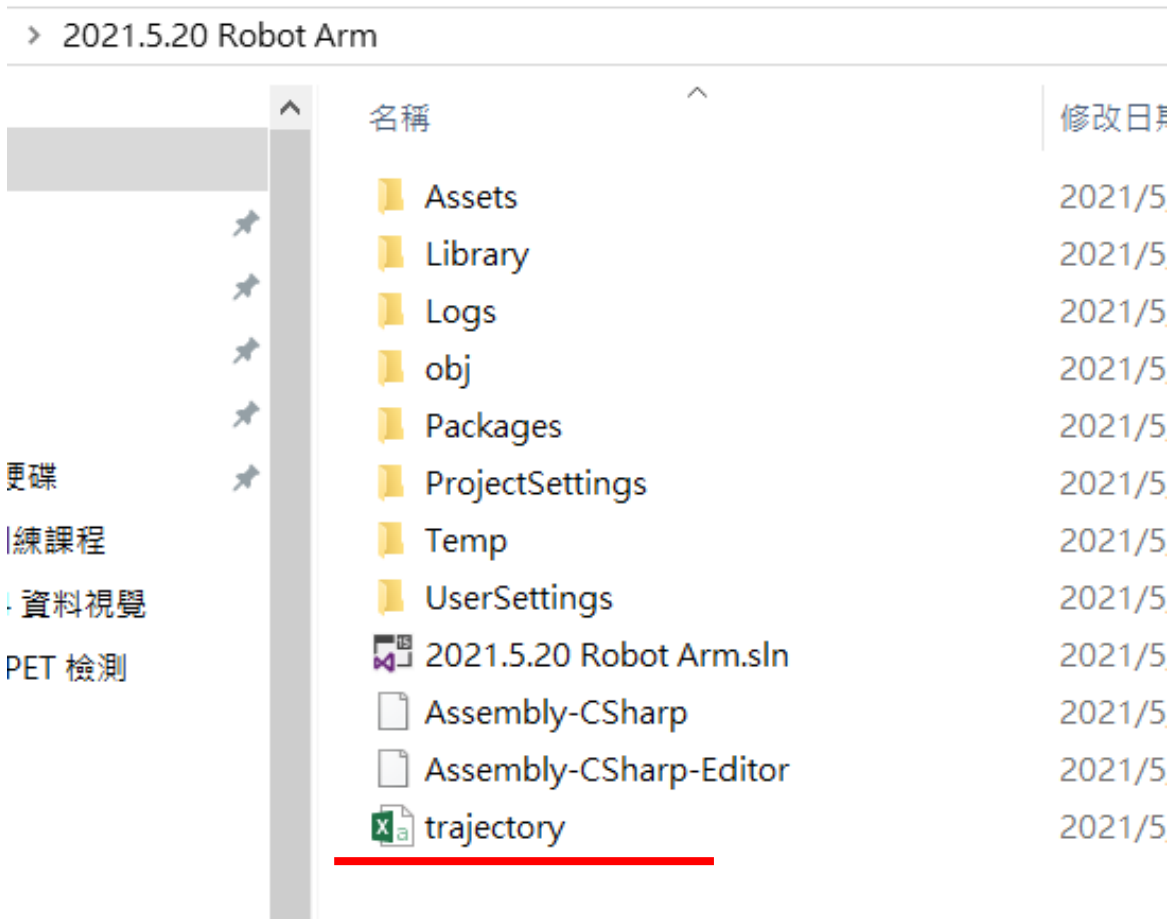
```
void Start()
{
    goalOriginalPos = goal.transform.position;
    BasePivotRoation = BasePivot.rotation;
    UpperPivotRotation = UpperPivot.rotation;
    LowerPivotRotation = LowerPivot.rotation;
    WristPivotRotation = WristPivot.rotation;
    GoalRotation = goal.rotation;

    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 ?

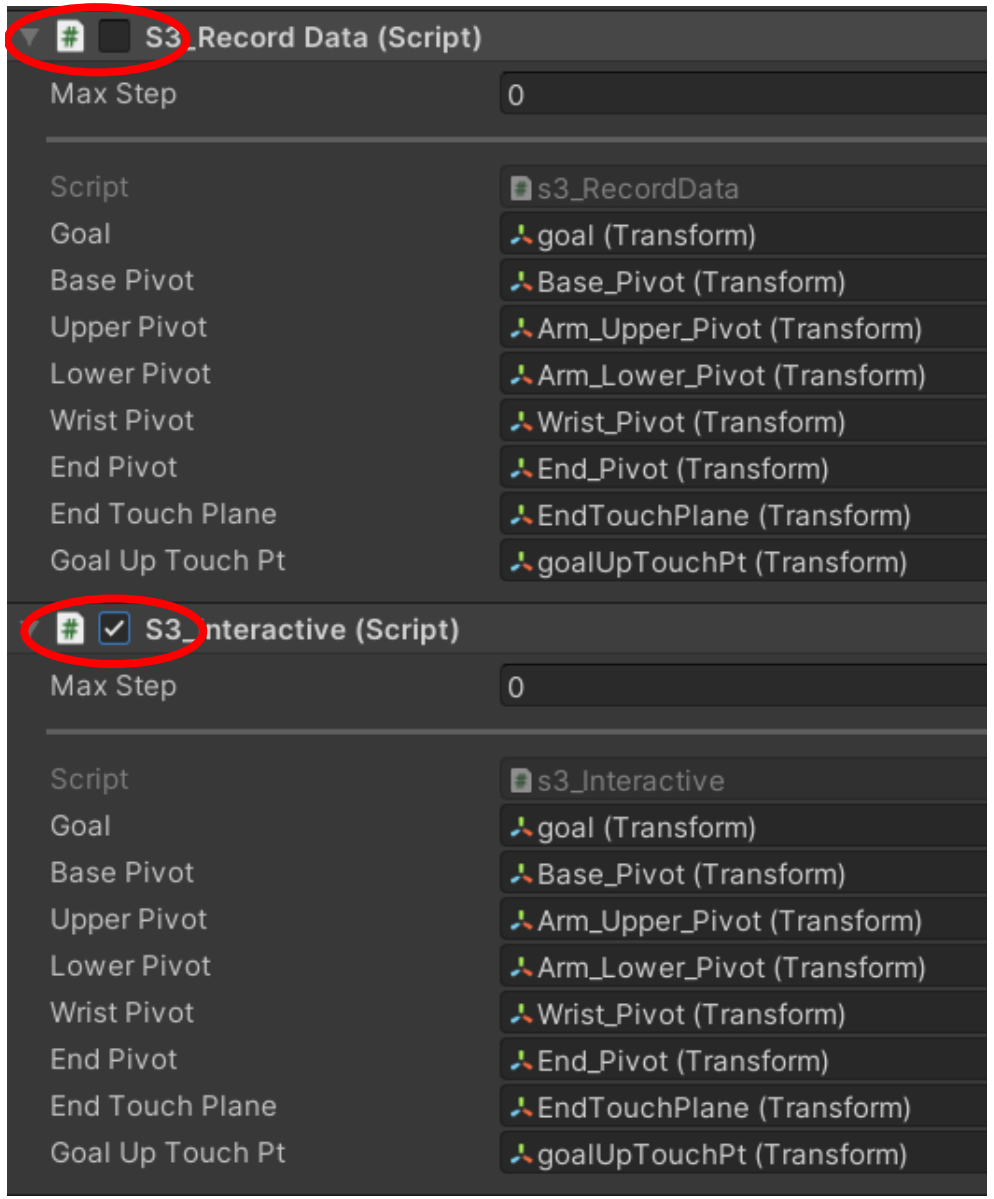
Examine the data file



	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



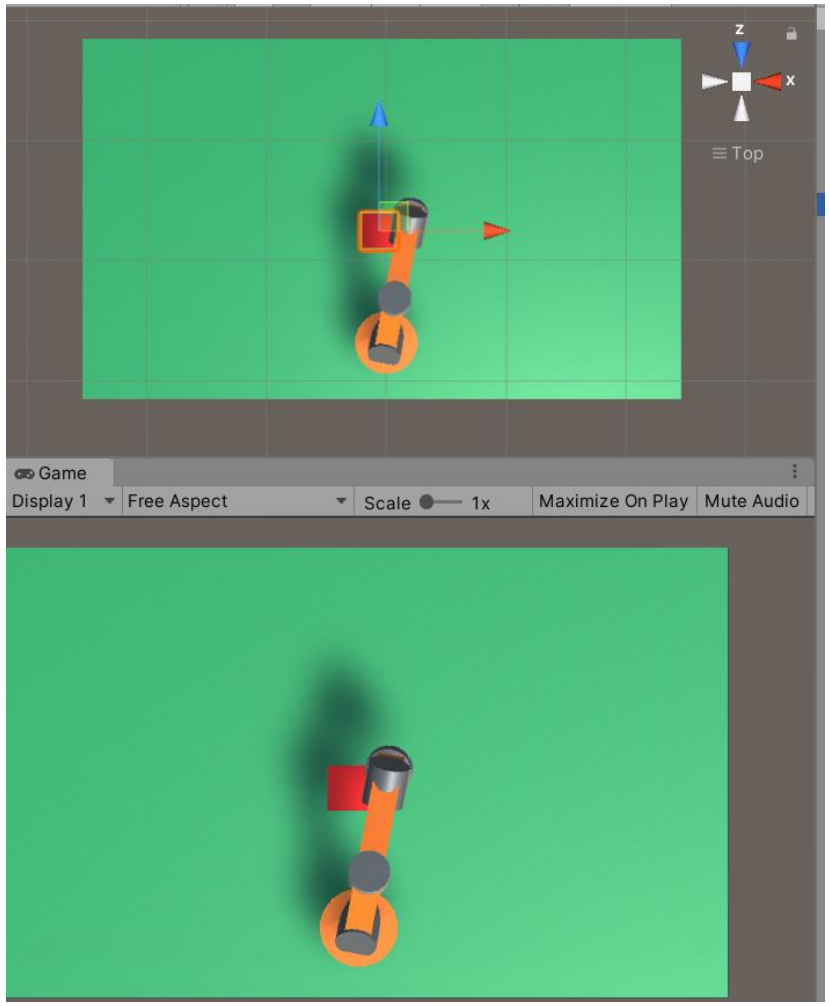
(2) interactive test

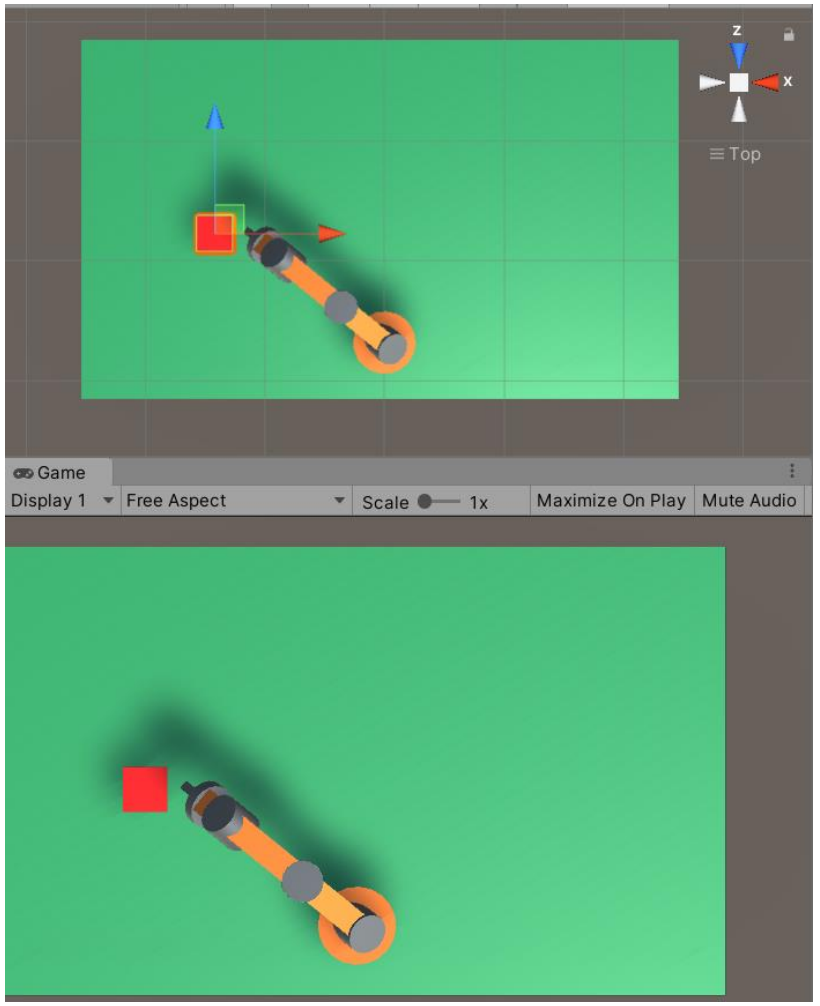
Should be equal to or larger than the threshold used for training, why ?

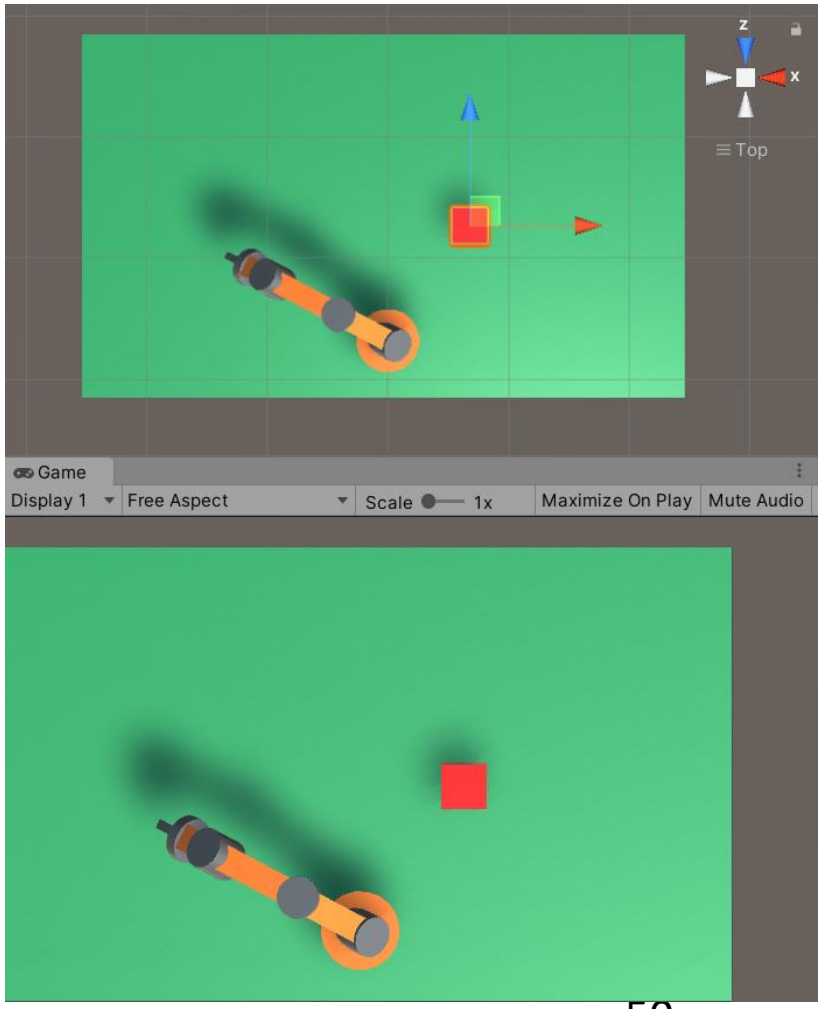
```
void Update()
{
    if (!PointTouch(EndTouchPlane, goalUpTouchPt, 0.3f))
    {
        RequestDecision();
    }
}
```

Why only succeed the first time?

Hint: during training, what is fixed ? What is randomly changed?

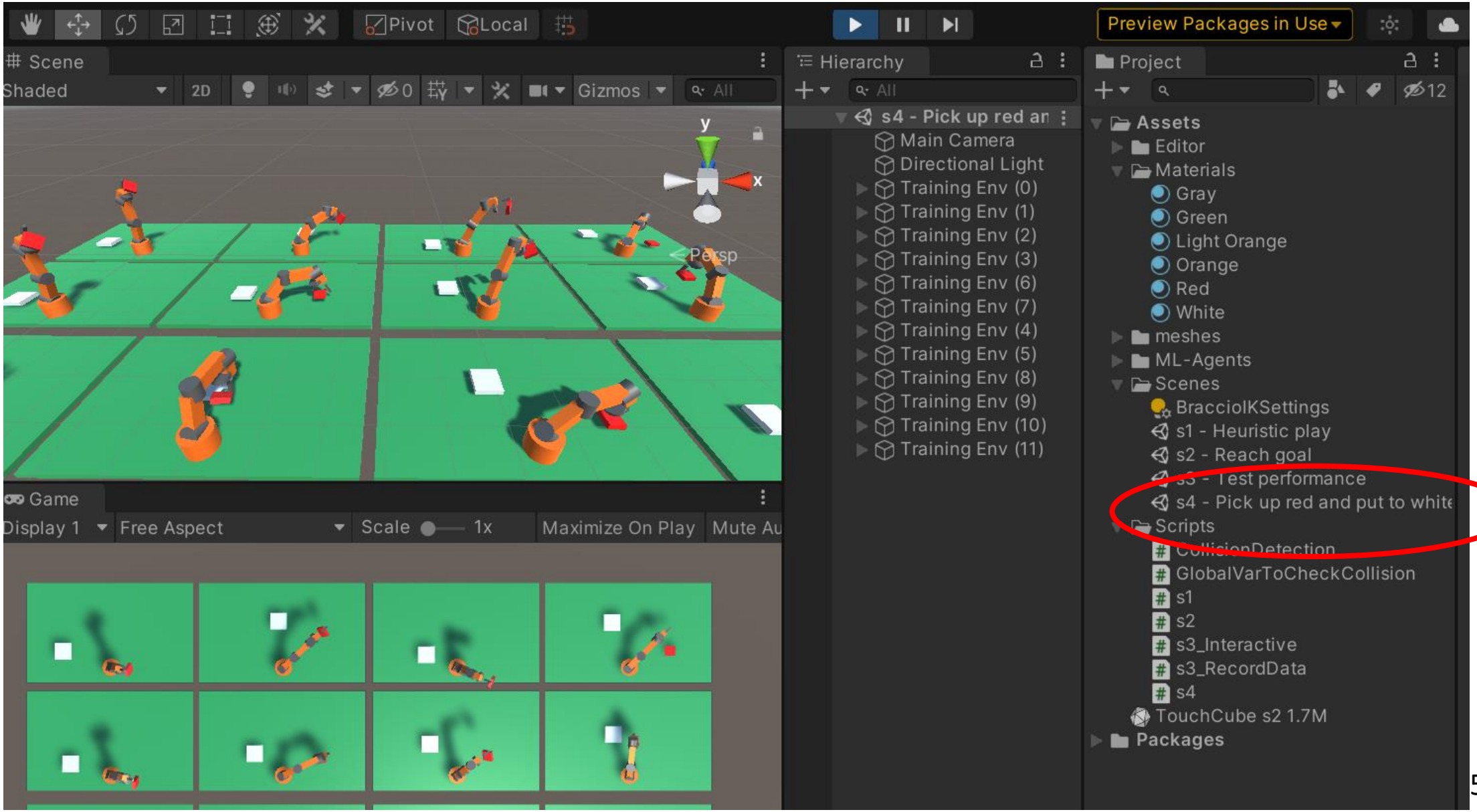






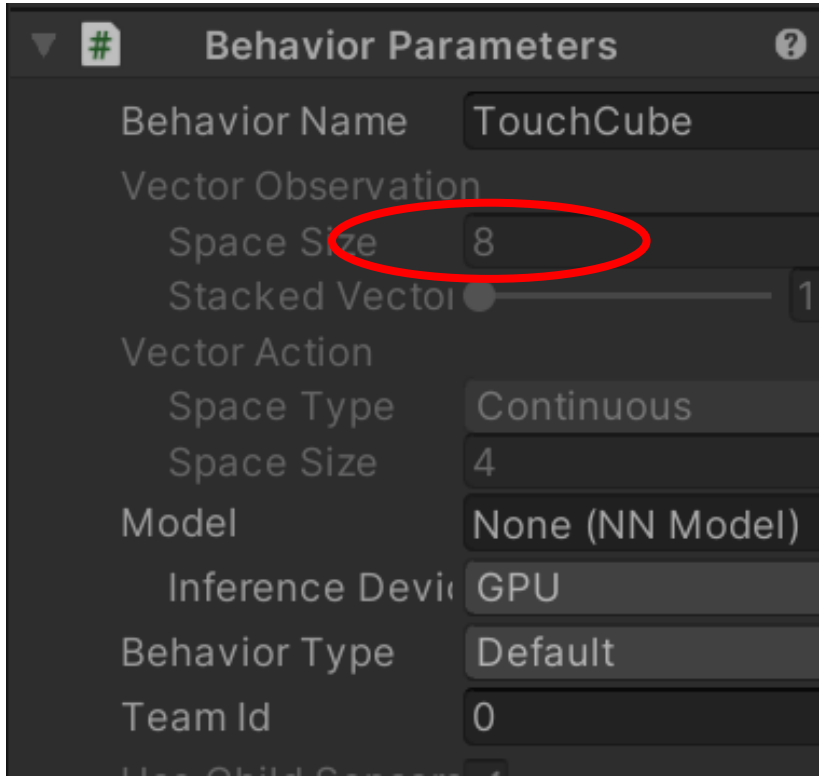
(3) Train robot arm to pick up the red cube and place it on top of the white cube

Open "s4 – Pick up red and put to white"



State has 8 variables

```
sensor.AddObservation(stage);

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

float BaseRotationAngle = UnityEditor.TransformUtils.GetInspectorRota
float UArmRotationAngle = UnityEditor.TransformUtils.GetInspectorRota
float LArmRotationAngle = UnityEditor.TransformUtils.GetInspectorRota
float WRotationAngle = UnityEditor.TransformUtils.GetInspectorRotatio

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);  //avoid 要廢
```

```
if (!Rotation_in_range())
{
    /*
    msg = trainingVE.name +
    print(msg); */
    AddReward(-5.0f);
    EndEpisode();
}
```

```
else if (LowerArmObj.GetComponent<CollisionDetectio
        WristObj.GetComponent<CollisionDetection>(
        EndObj.GetComponent<CollisionDetection>()
        GoalObj.GetComponent<CollisionDetection>(
{
    msg = trainingVE.name + "Collision" +
        LowerArmObj.GetComponent<CollisionDetectio
        WristObj.GetComponent<CollisionDetection>(
        EndObj.GetComponent<CollisionDetection>()
        GoalObj.GetComponent<CollisionDetection>(
    print(msg);
    AddReward(-5.0f);
    EndEpisode();
}
```

4 types of rewards

```
// if no rotation out of range and no collision, give awards
if (stage ==1 && PointTouch(EndTouchPlane, goalUpTouchPt, 0.1f))
{
    msg = trainingVE.name + " Goal 1! \n";
    Debug.Log(msg);
    stage = 2;
    AddReward(15.0f);
    goal.transform.parent = EndPivot.transform; //grab goal
}
else if (PointTouch(goalDownTouchPt, goal2UpTouchPt, 0.3f))
{
    msg = trainingVE.name + " Goal 2! \n";
    Debug.Log(msg);
    AddReward(100.0f);
    EndEpisode();
}
```

Training configuration file

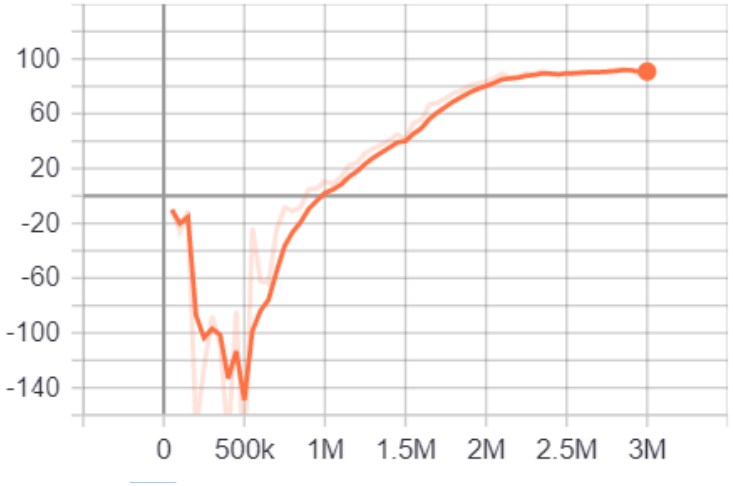
```
TouchCube:
  trainer_type: ppo
  hyperparameters:
    batch_size: 2048
    buffer_size: 20480
    learning_rate: 0.0003
    beta: 0.001
    epsilon: 0.2
    lambd: 0.95
    num_epoch: 3
    learning_rate_schedule
  network_settings:
    normalize: true
    hidden_units: 512
    num_layers: 3
   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

I quit at 3M, looks promising

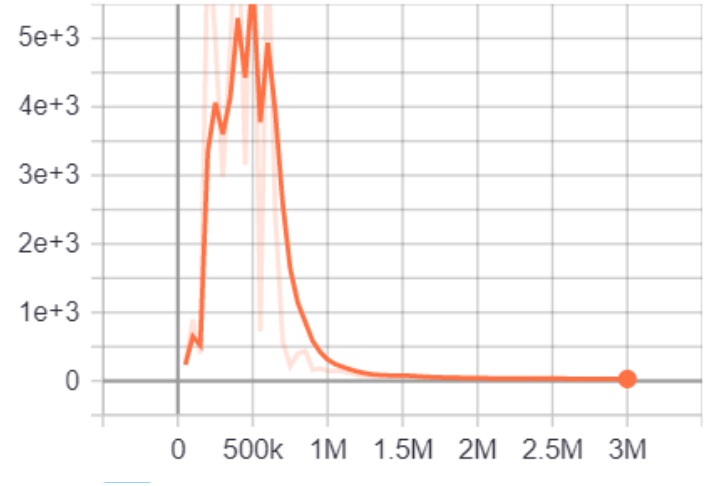
```
TouchCube. Step: 1400000. Time Elapsed: 1521.994 s. Mean Reward: 40.746. Std of Reward: 56.700. Training.
TouchCube. Step: 1450000. Time Elapsed: 1596.218 s. Mean Reward: 44.816. Std of Reward: 57.113. Training.
TouchCube. Step: 1500000. Time Elapsed: 1654.829 s. Mean Reward: 41.636. Std of Reward: 56.026. Training.
zation.py:93] Converting to results\1\TouchCube\TouchCube-1499998.onnx
zation.py:105] Exported results\1\TouchCube\TouchCube-1499998.onnx
TouchCube. Step: 1550000. Time Elapsed: 1710.662 s. Mean Reward: 53.289. Std of Reward: 58.197. Training.
TouchCube. Step: 1600000. Time Elapsed: 1772.814 s. Mean Reward: 55.664. Std of Reward: 57.972. Training.
TouchCube. Step: 1650000. Time Elapsed: 1839.616 s. Mean Reward: 66.710. Std of Reward: 57.145. Training.
TouchCube. Step: 1700000. Time Elapsed: 1905.361 s. Mean Reward: 68.004. Std of Reward: 56.702. Training.
TouchCube. Step: 1750000. Time Elapsed: 1974.348 s. Mean Reward: 71.199. Std of Reward: 55.970. Training.
TouchCube. Step: 1800000. Time Elapsed: 2034.998 s. Mean Reward: 75.024. Std of Reward: 54.928. Training.
TouchCube. Step: 1850000. Time Elapsed: 2093.756 s. Mean Reward: 77.129. Std of Reward: 54.333. Training.
TouchCube. Step: 1900000. Time Elapsed: 2153.092 s. Mean Reward: 80.611. Std of Reward: 52.868. Training.
TouchCube. Step: 1950000. Time Elapsed: 2214.371 s. Mean Reward: 82.048. Std of Reward: 52.347. Training.
TouchCube. Step: 2000000. Time Elapsed: 2273.202 s. Mean Reward: 83.202. Std of Reward: 51.604. Training.
zation.py:93] Converting to results\1\TouchCube\TouchCube-1999932.onnx
zation.py:105] Exported results\1\TouchCube\TouchCube-1999932.onnx
TouchCube. Step: 2050000. Time Elapsed: 2335.471 s. Mean Reward: 85.839. Std of Reward: 50.198. Training.
TouchCube. Step: 2100000. Time Elapsed: 2396.409 s. Mean Reward: 89.061. Std of Reward: 48.055. Training.
TouchCube. Step: 2150000. Time Elapsed: 2460.368 s. Mean Reward: 86.739. Std of Reward: 49.768. Training.
TouchCube. Step: 2200000. Time Elapsed: 2521.083 s. Mean Reward: 87.105. Std of Reward: 49.479. Training.
TouchCube. Step: 2250000. Time Elapsed: 2583.198 s. Mean Reward: 89.747. Std of Reward: 47.747. Training.
TouchCube. Step: 2300000. Time Elapsed: 2643.755 s. Mean Reward: 89.104. Std of Reward: 48.263. Training.
TouchCube. Step: 2350000. Time Elapsed: 2705.644 s. Mean Reward: 91.258. Std of Reward: 46.869. Training.
TouchCube. Step: 2400000. Time Elapsed: 2770.443 s. Mean Reward: 88.986. Std of Reward: 48.436. Training.
TouchCube. Step: 2450000. Time Elapsed: 2833.605 s. Mean Reward: 88.044. Std of Reward: 49.433. Training.
TouchCube. Step: 2500000. Time Elapsed: 2895.328 s. Mean Reward: 90.368. Std of Reward: 48.504. Training.
zation.py:93] Converting to results\1\TouchCube\TouchCube-2499995.onnx
zation.py:105] Exported results\1\TouchCube\TouchCube-2499995.onnx
TouchCube. Step: 2550000. Time Elapsed: 2956.370 s. Mean Reward: 89.639. Std of Reward: 48.371. Training.
TouchCube. Step: 2600000. Time Elapsed: 3028.253 s. Mean Reward: 90.657. Std of Reward: 47.175. Training.
TouchCube. Step: 2650000. Time Elapsed: 3095.190 s. Mean Reward: 90.670. Std of Reward: 47.247. Training.
TouchCube. Step: 2700000. Time Elapsed: 3157.692 s. Mean Reward: 90.483. Std of Reward: 47.232. Training.
TouchCube. Step: 2750000. Time Elapsed: 3226.099 s. Mean Reward: 91.113. Std of Reward: 46.861. Training.
TouchCube. Step: 2800000. Time Elapsed: 3285.665 s. Mean Reward: 91.785. Std of Reward: 46.426. Training.
TouchCube. Step: 2850000. Time Elapsed: 3349.785 s. Mean Reward: 92.851. Std of Reward: 46.171. Training.
TouchCube. Step: 2900000. Time Elapsed: 3413.532 s. Mean Reward: 91.497. Std of Reward: 46.714. Training.
TouchCube. Step: 2950000. Time Elapsed: 3480.496 s. Mean Reward: 89.342. Std of Reward: 48.568. Training.
TouchCube. Step: 3000000. Time Elapsed: 3545.361 s. Mean Reward: 90.762. Std of Reward: 47.138. Training.
```

I quit at 3M, looks promising

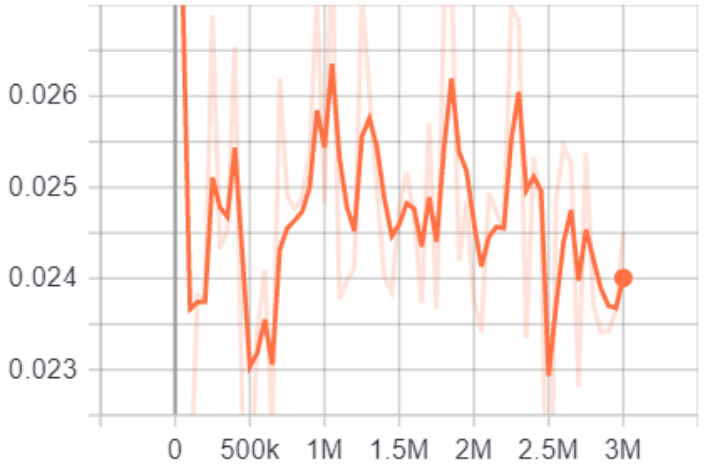
Cumulative Reward
tag: Environment/Cumulative Reward



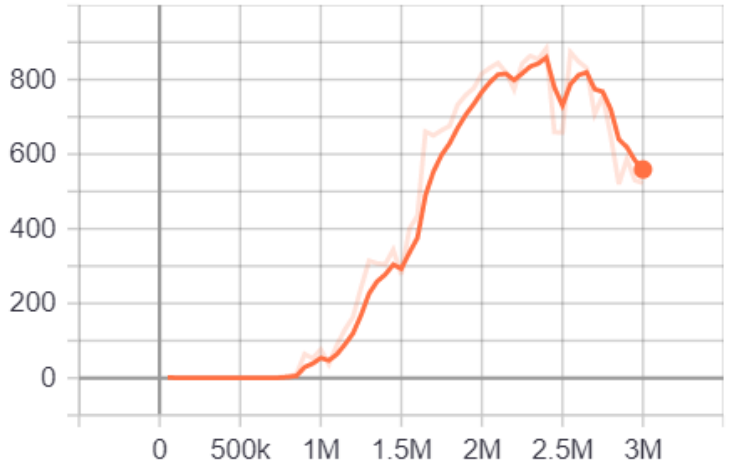
Episode Length
tag: Environment/Episode Length



Policy Loss
tag: Losses/Policy Loss



Value Loss
tag: Losses/Value Loss



HW4(2)

- Describe the training environment ("s4 – Pick up red and put to white")
- Describe the agent script (s, a, r)
- Show tensor board plots and discuss your training performance
- Describe your test performance