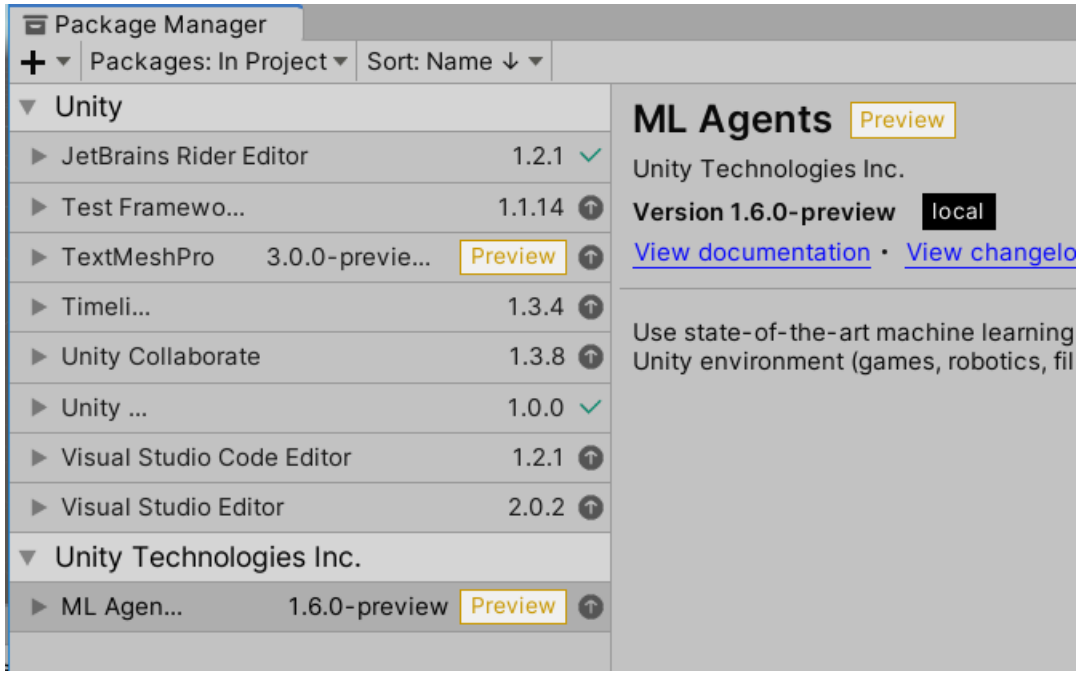
1. Download and save ML Agent to C:\

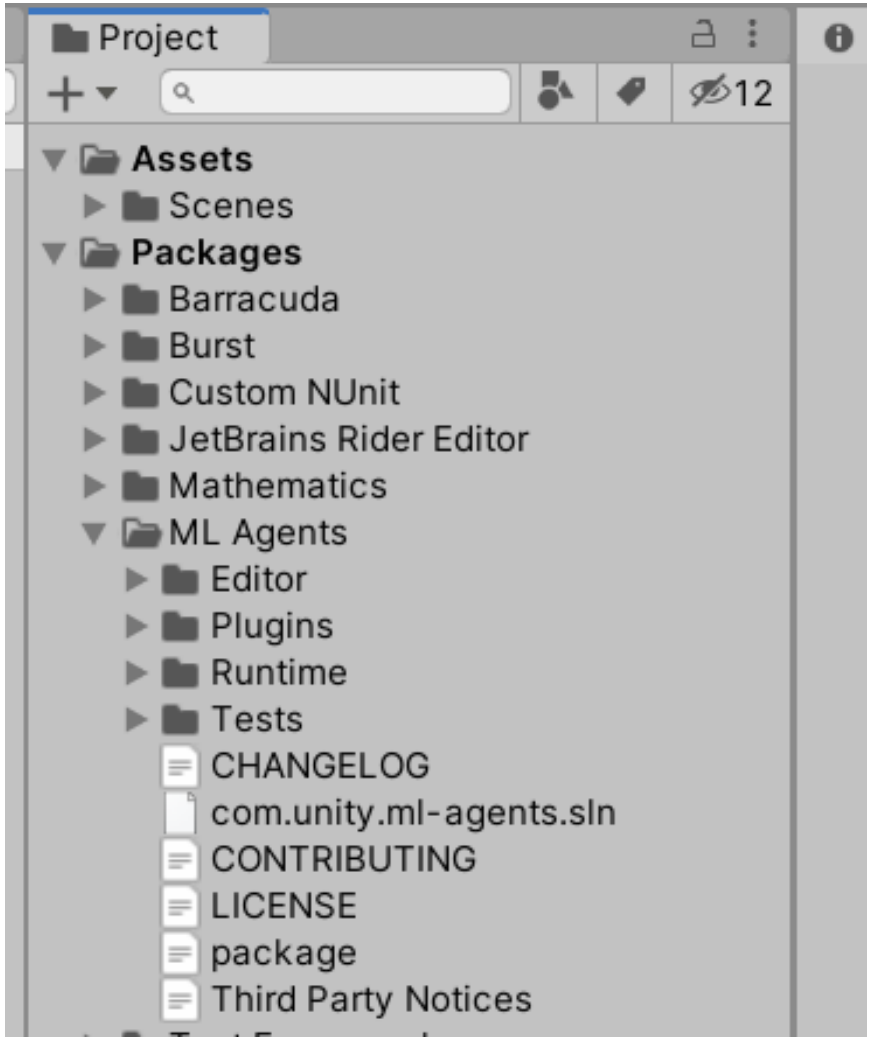
This will make it convenient to type commands to train and monitor performance

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

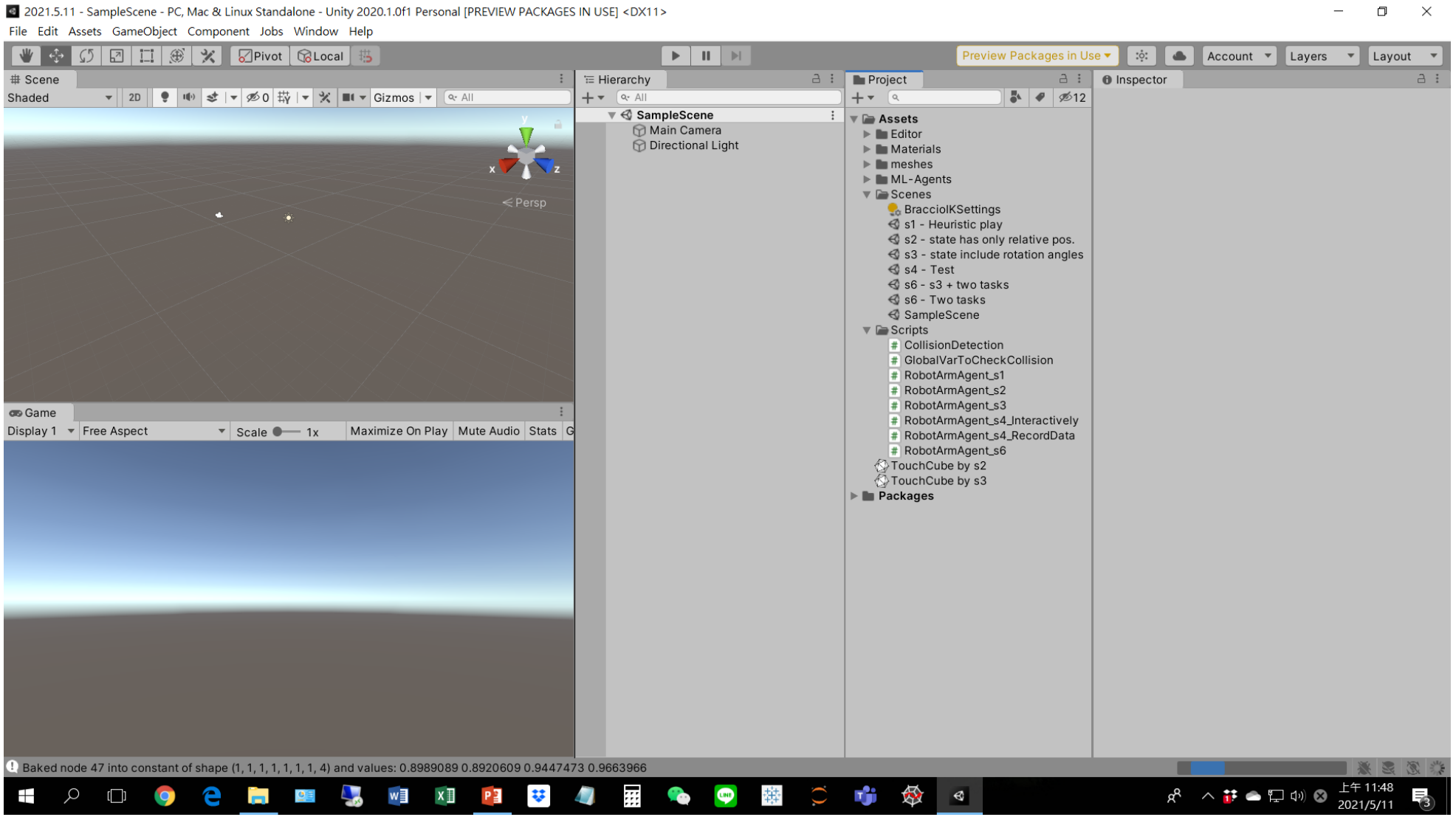
```
cd C:\ml-agents-release_10\config\ppo\results
tensorboard --logdir=1
```

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

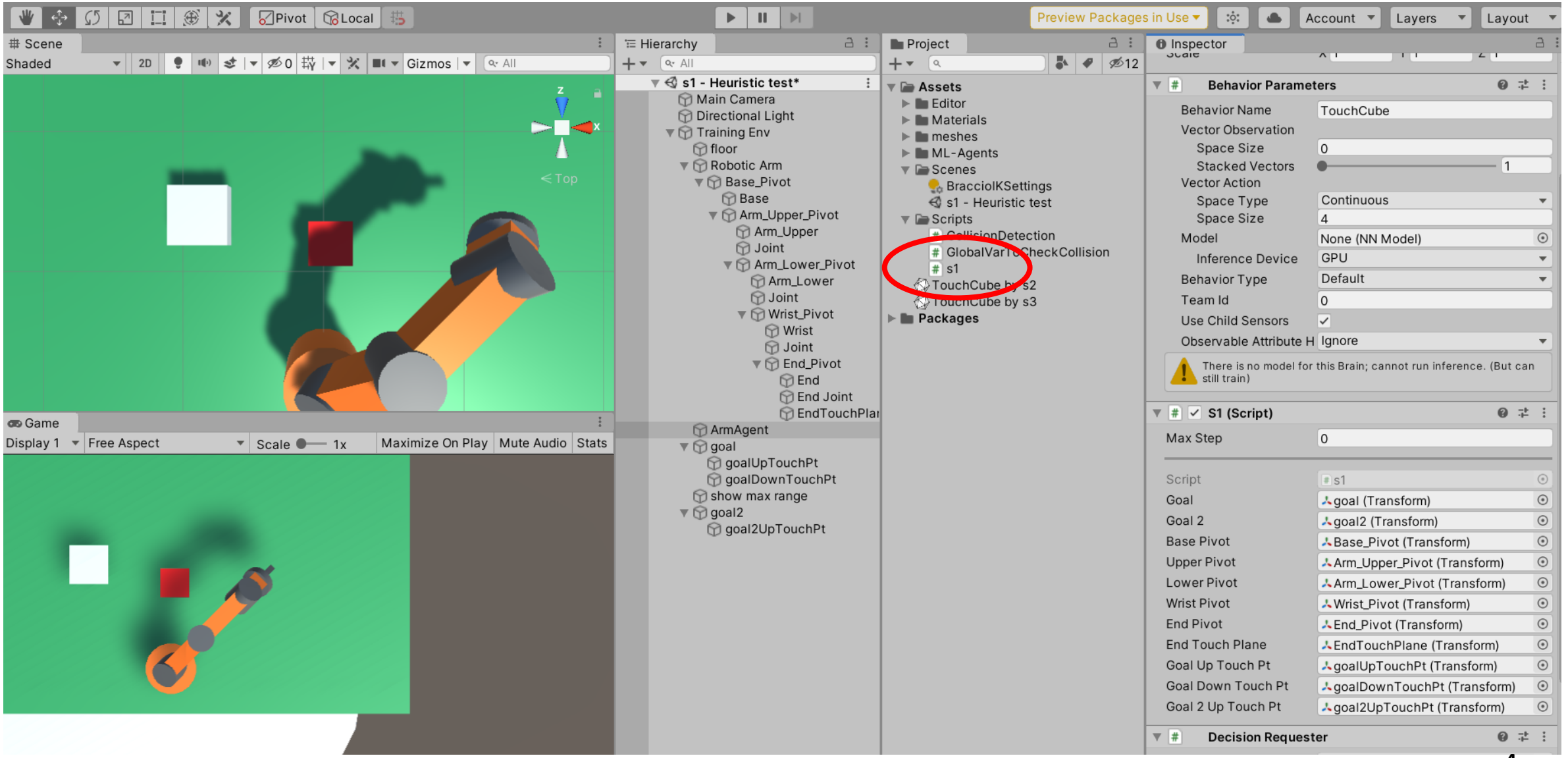




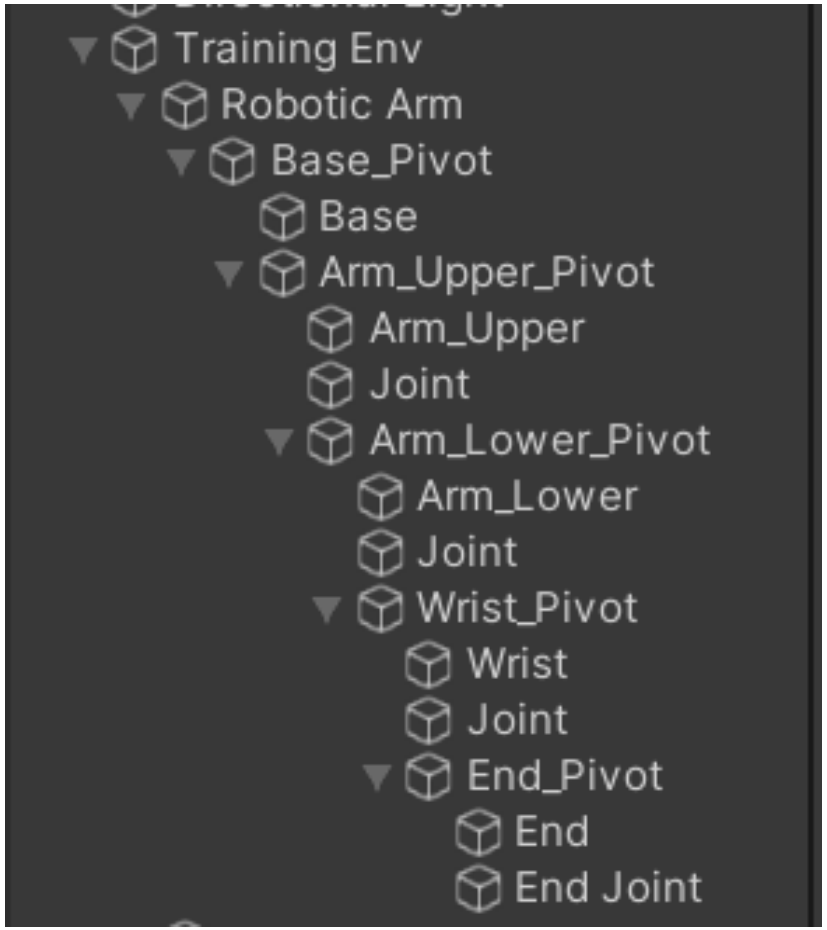
3. Import Robot arm package to Unity project

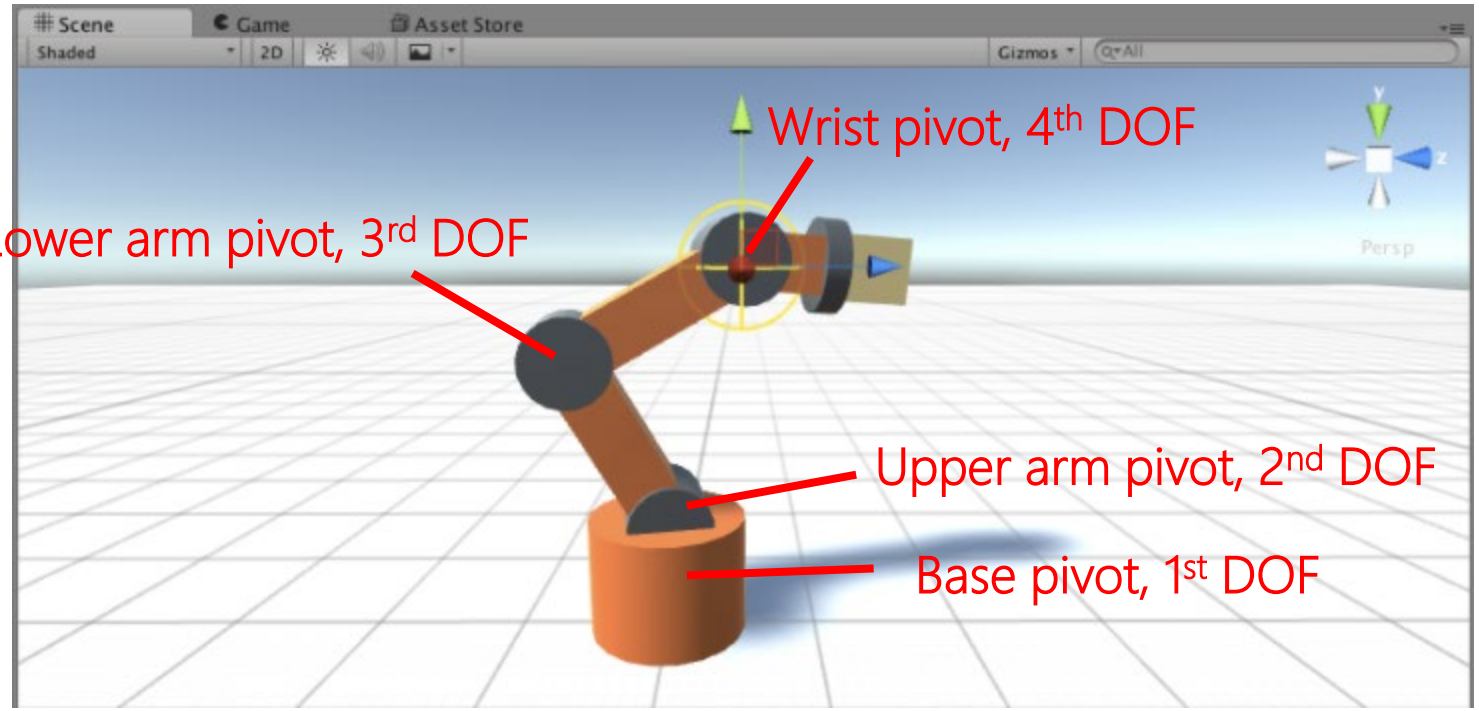


Open scene "s1 - Heuristic play"



This Unity project contains a Braccio robot arm



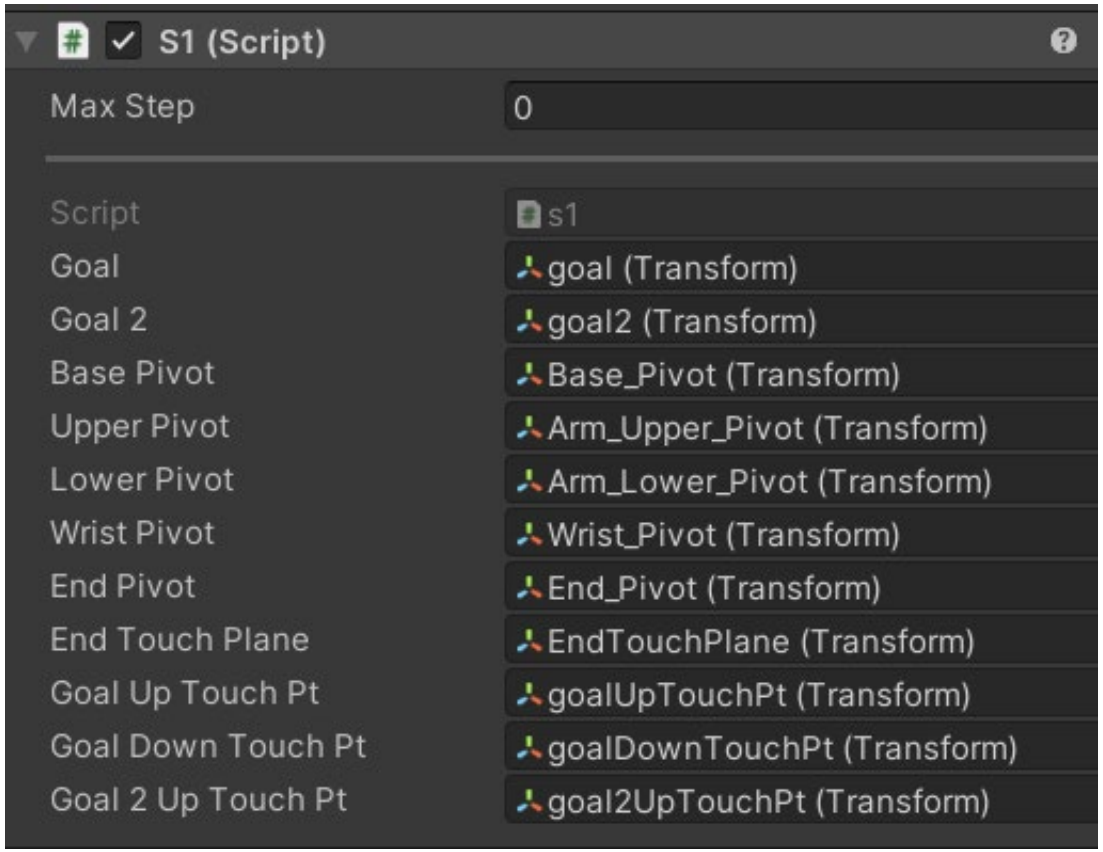


https://github.com/tanyuan/braccio-ik-unity

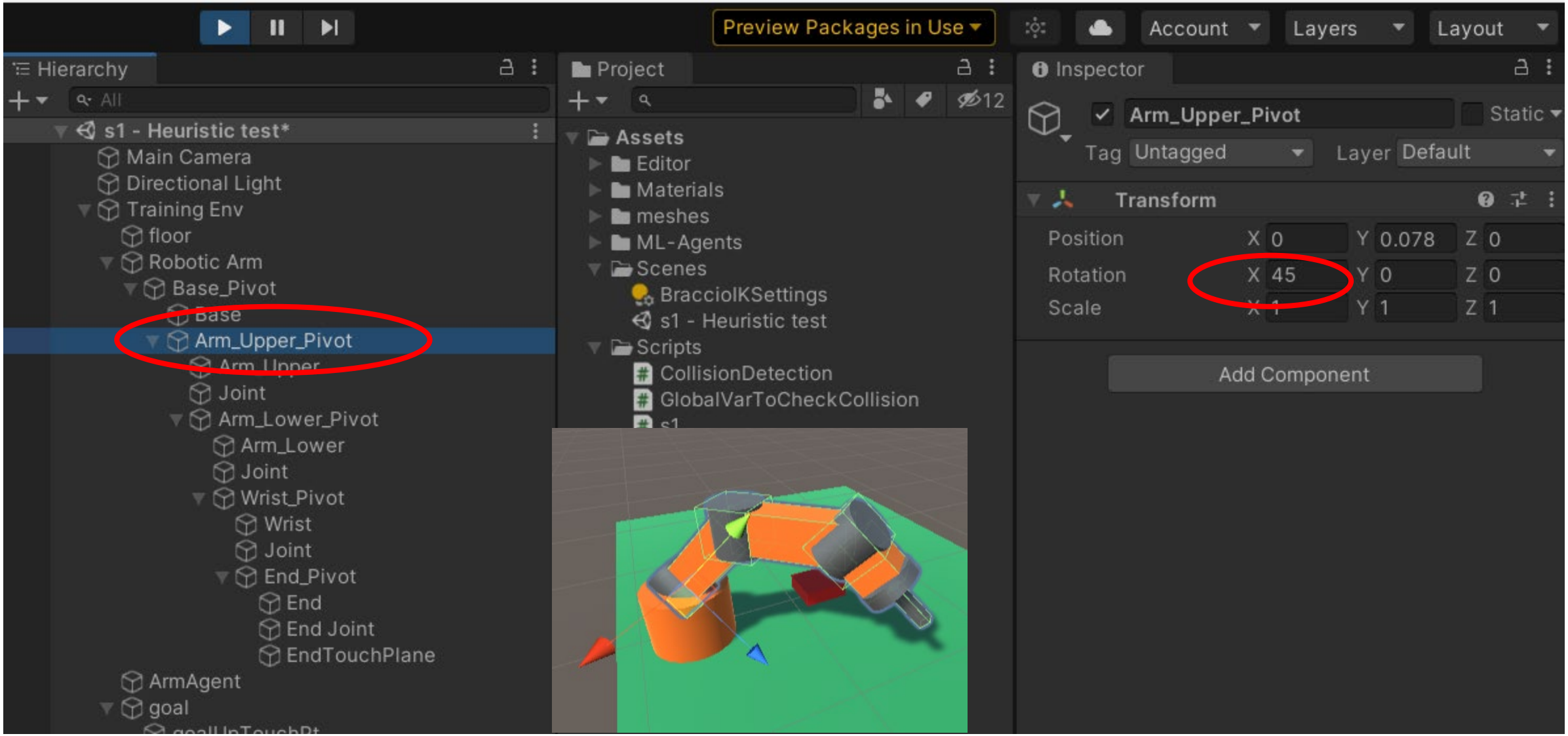
Manually manipulate the 4DOF robot arm

Public variables

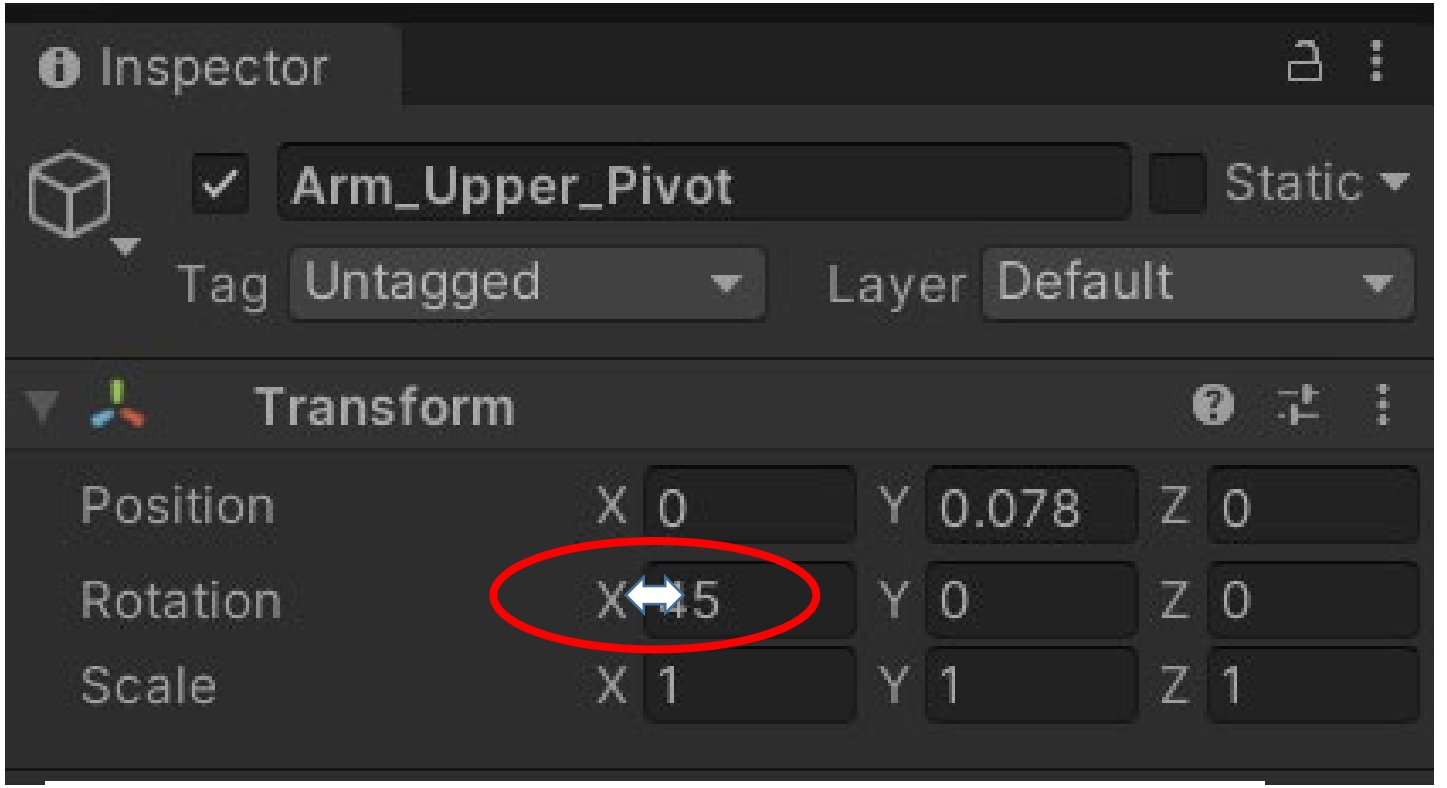
```
public Transform goal, goal2;
public Transform BasePivot, UpperPivot, LowerPivot, WristPivot, EndPivot;
public Transform EndTouchPlane, goalUpTouchPt, goalDownTouchPt, goal2UpTouchPt;
int stage = 1;
```



Play, rotate arm by changing rotation angle in Inspector window



Play, rotate arm by changing rotation angle in Inspector window



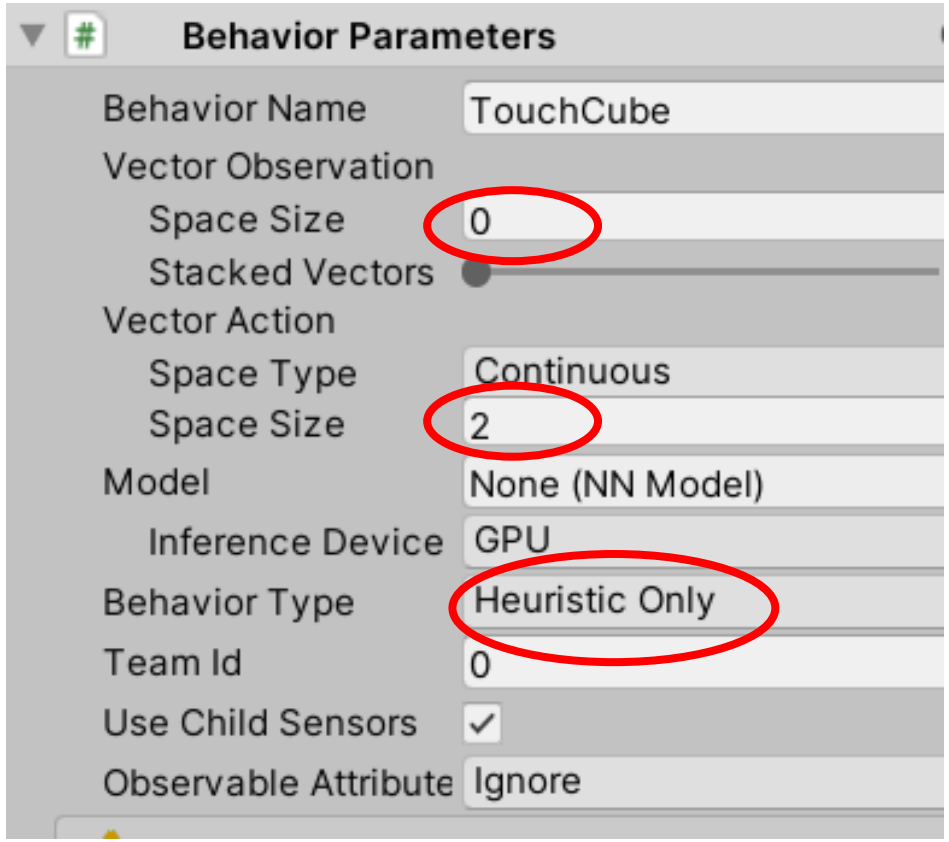
Place your mouse here and you can use the arrow to easily adjust the values

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 arm range: 0 ~ 90

L arm, Wrist: -90~90

```
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;
```

Collision detection

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

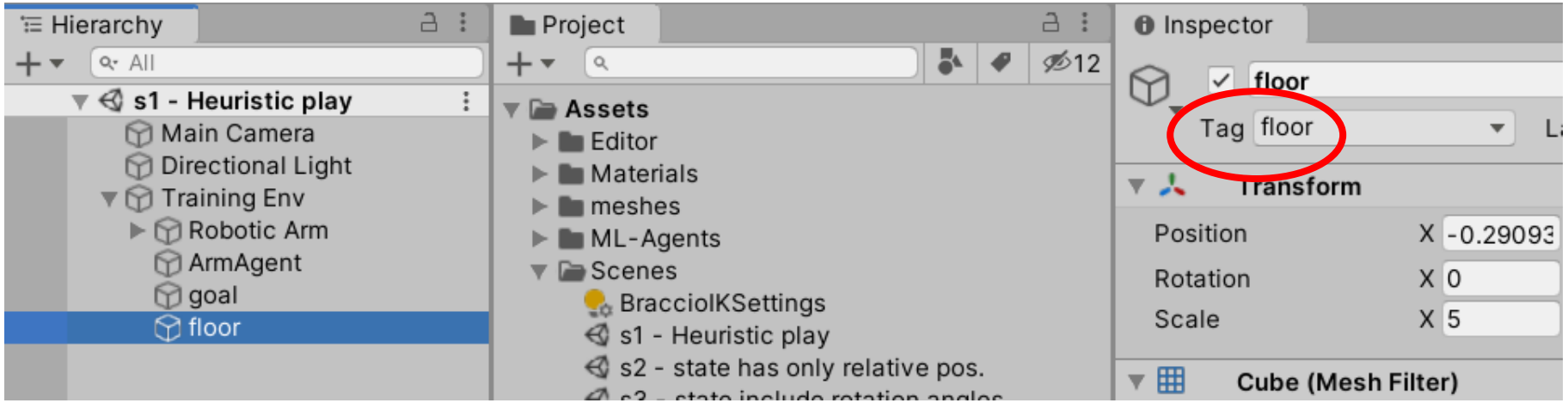
```
public class MyGlobalVar : MonoBehaviour
{
    public static bool LowerArmCollisionHappens = false;
    public static bool WristCollisionHappens = false;
    public static bool EndCollisionHappens = false;
    public static bool goalCollisionHappens = false;
```

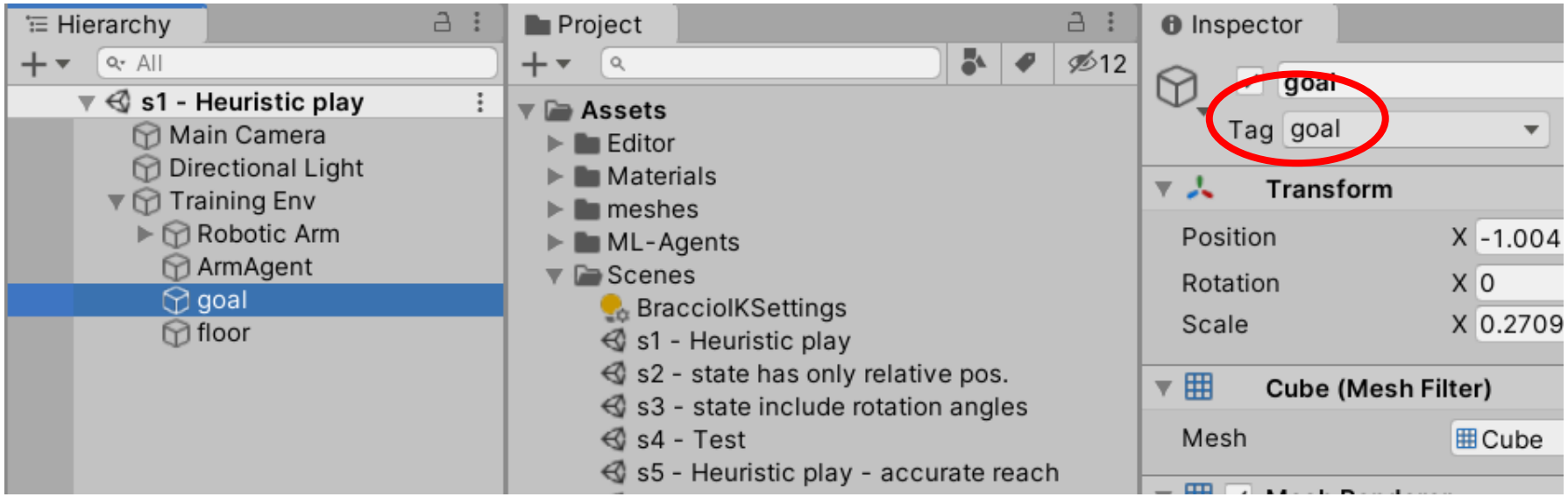
On trigger enter/exist to record collisions of lower arm, wrist, and end with floor and goal objects

```
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")
```

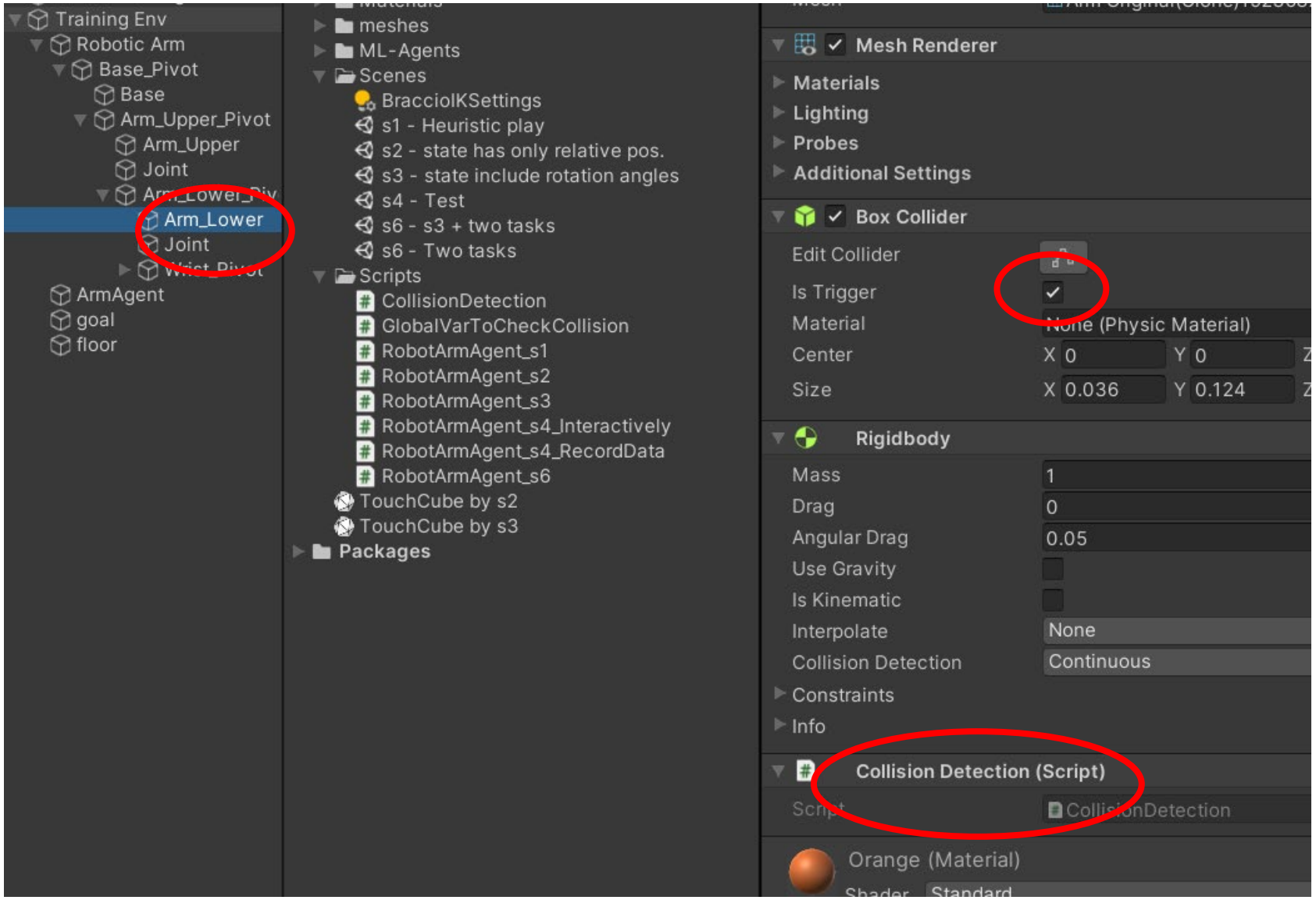
```
void OnTriggerExit(Collider other)
{
    if (other.gameObject.tag == "floor"
    {
        if (this.gameObject.tag == "Lowe
            MyGlobalVar.LowerArmCollisic
        else if (this.gameObject.tag ==
```

Add tags to floor and goal object

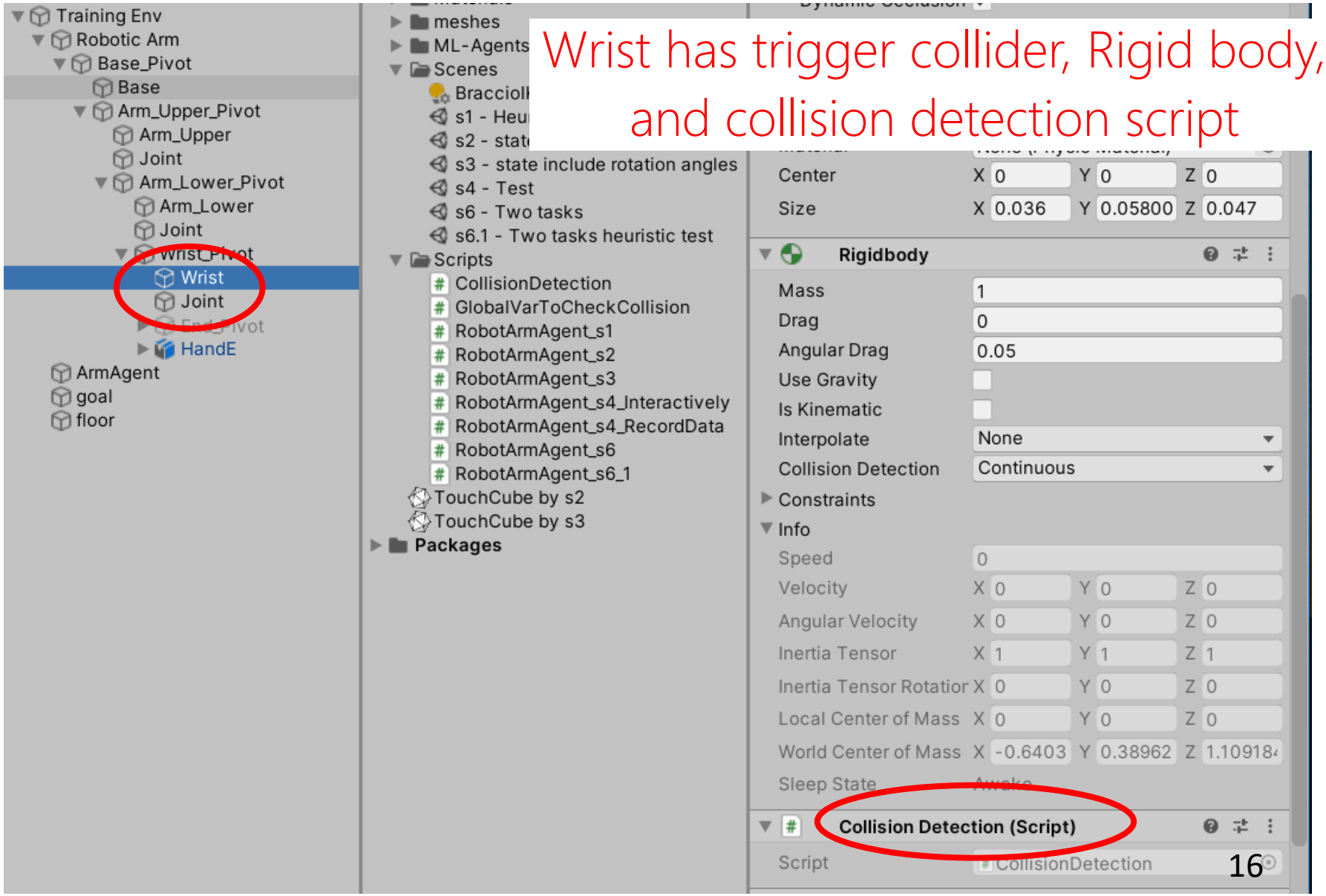




Lower arm has trigger collider, Rigid body, and collision detection script

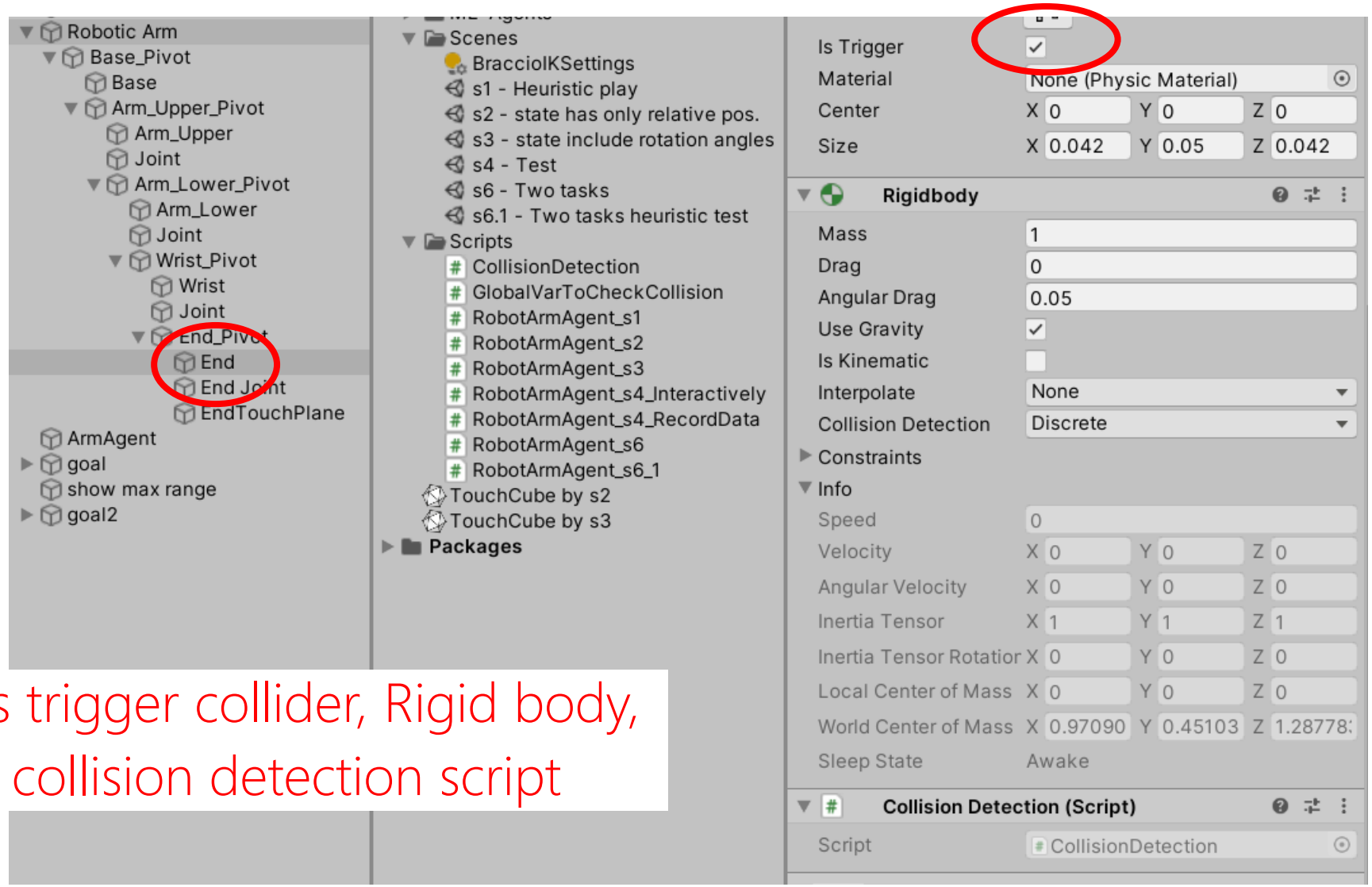


Wrist has trigger collider, Rigid body, and collision detection script



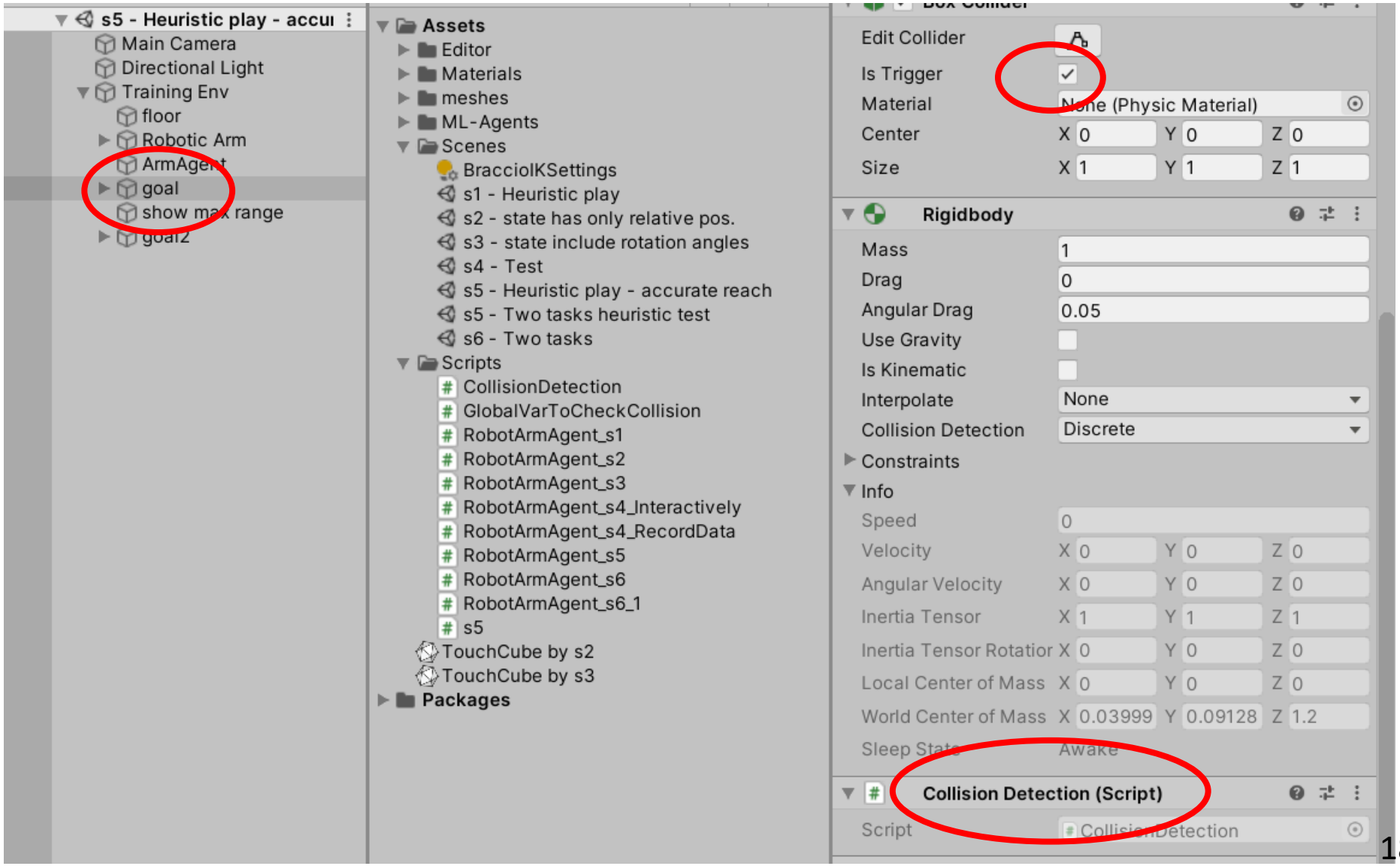
Wrist has trigger collider, Rigid body, and collision detection script

End has trigger collider, Rigid body, and collision detection script

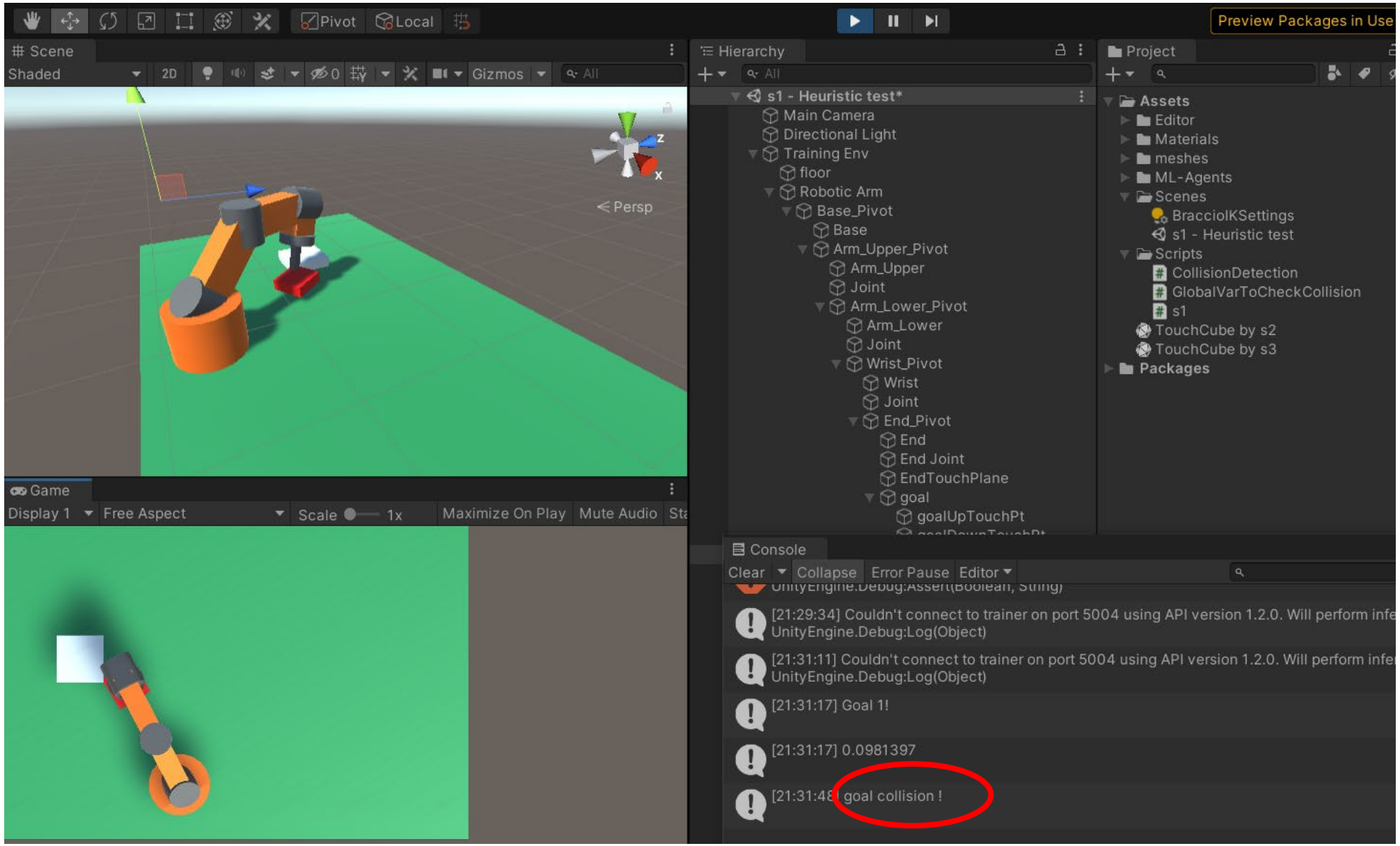


End has trigger collider, Rigid body, and collision detection script

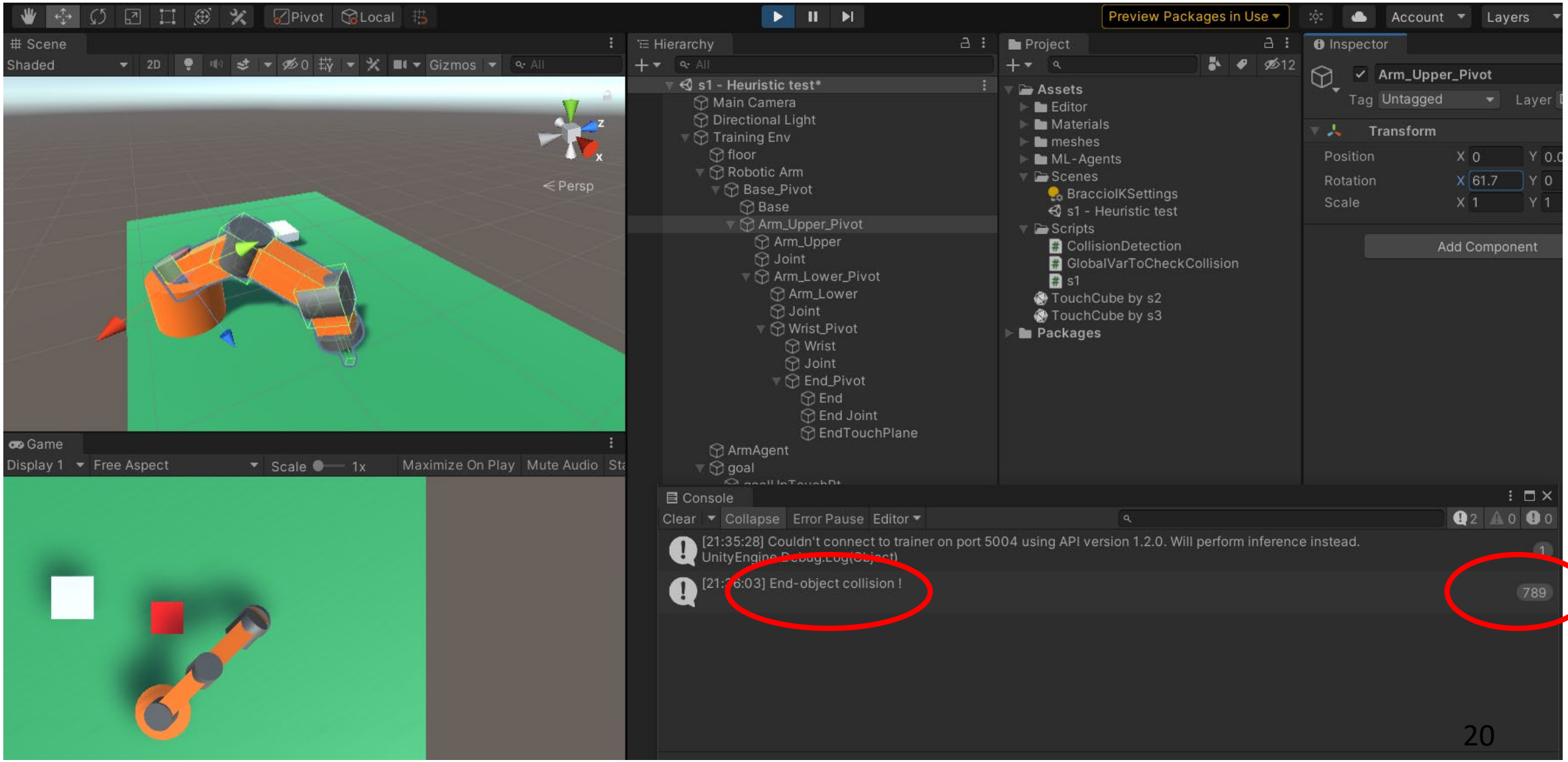
Goal has trigger collider, Rigid body, and collision detection script



Examples of collisions

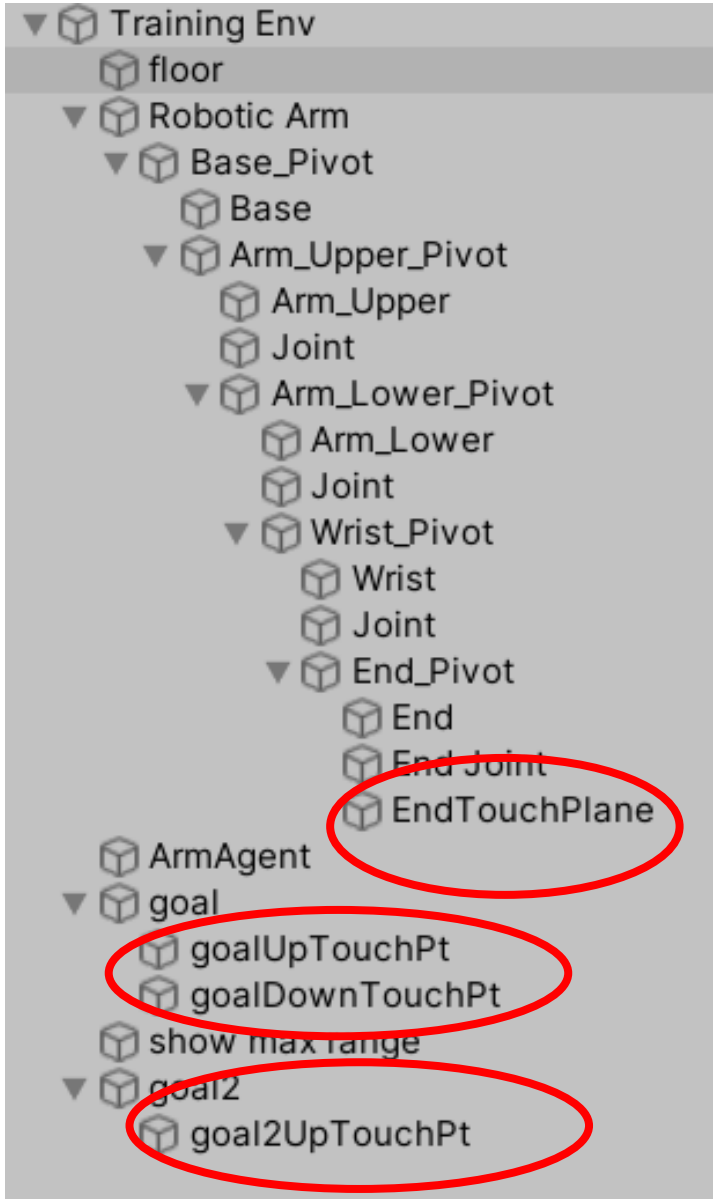


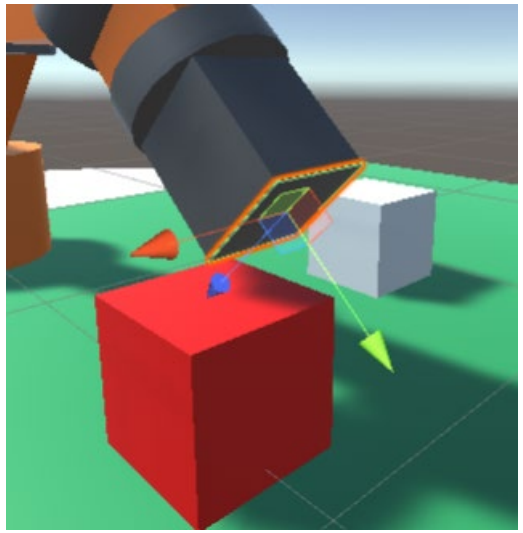
Examples of collisions

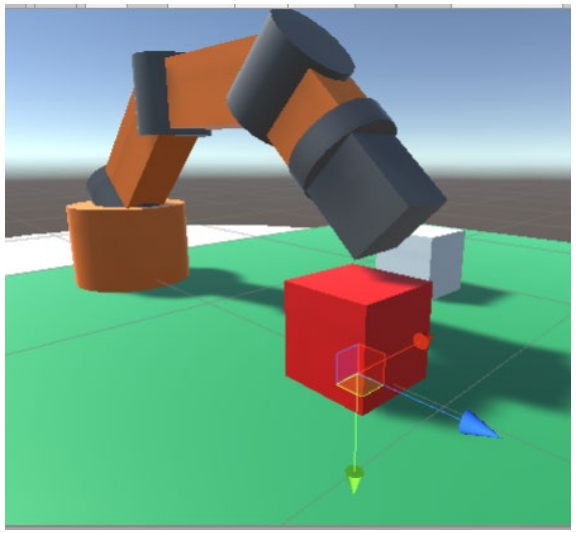


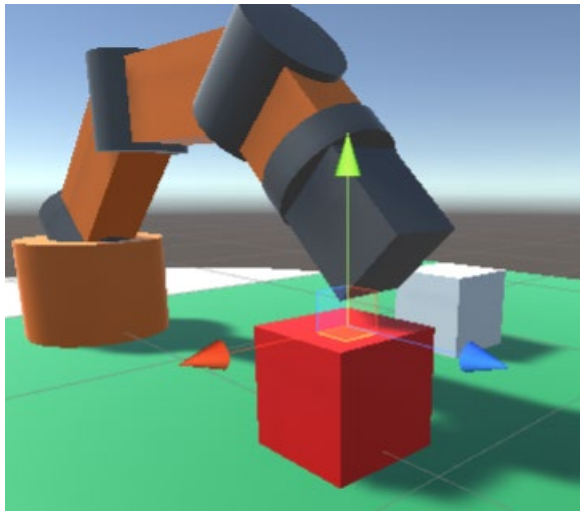
Determine goal-reach

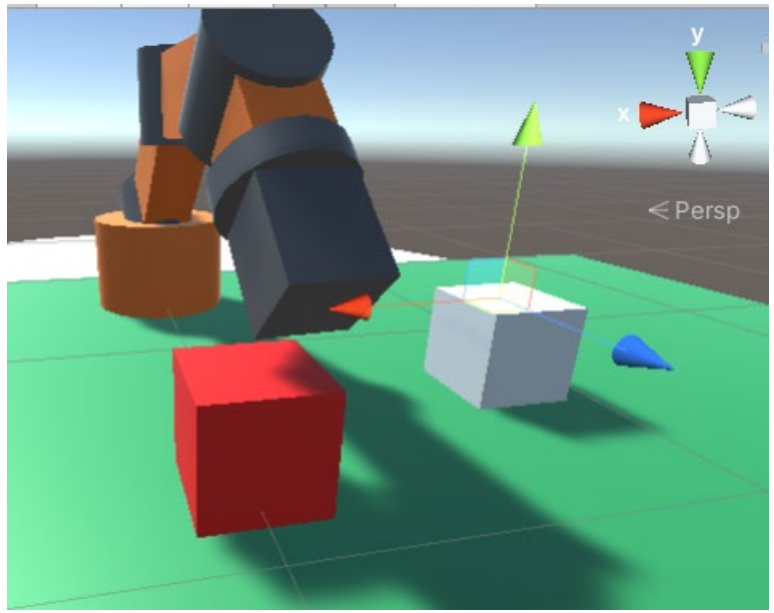
Points to detect goal-reach







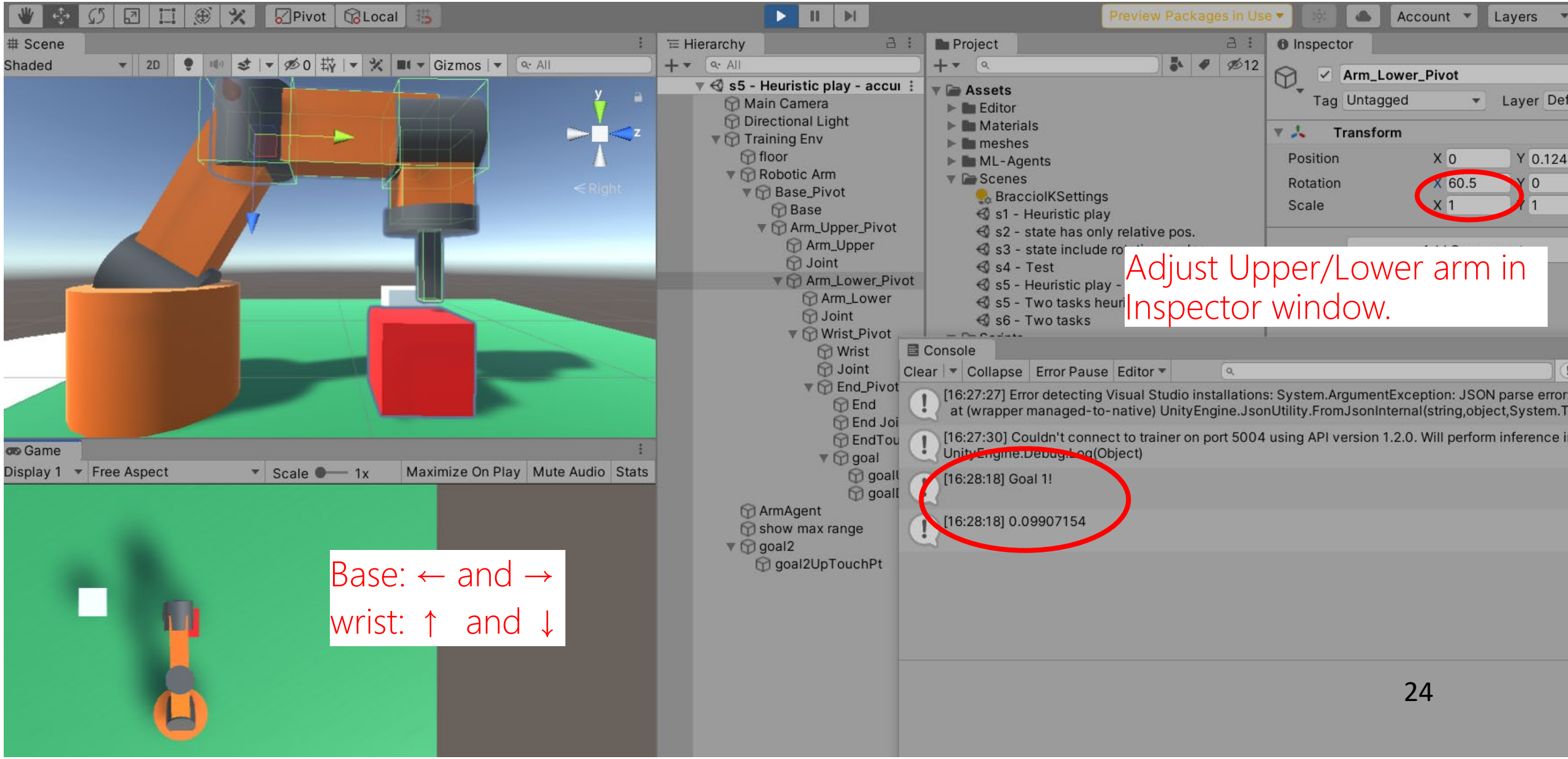




Use distance and y value to determine goal reach

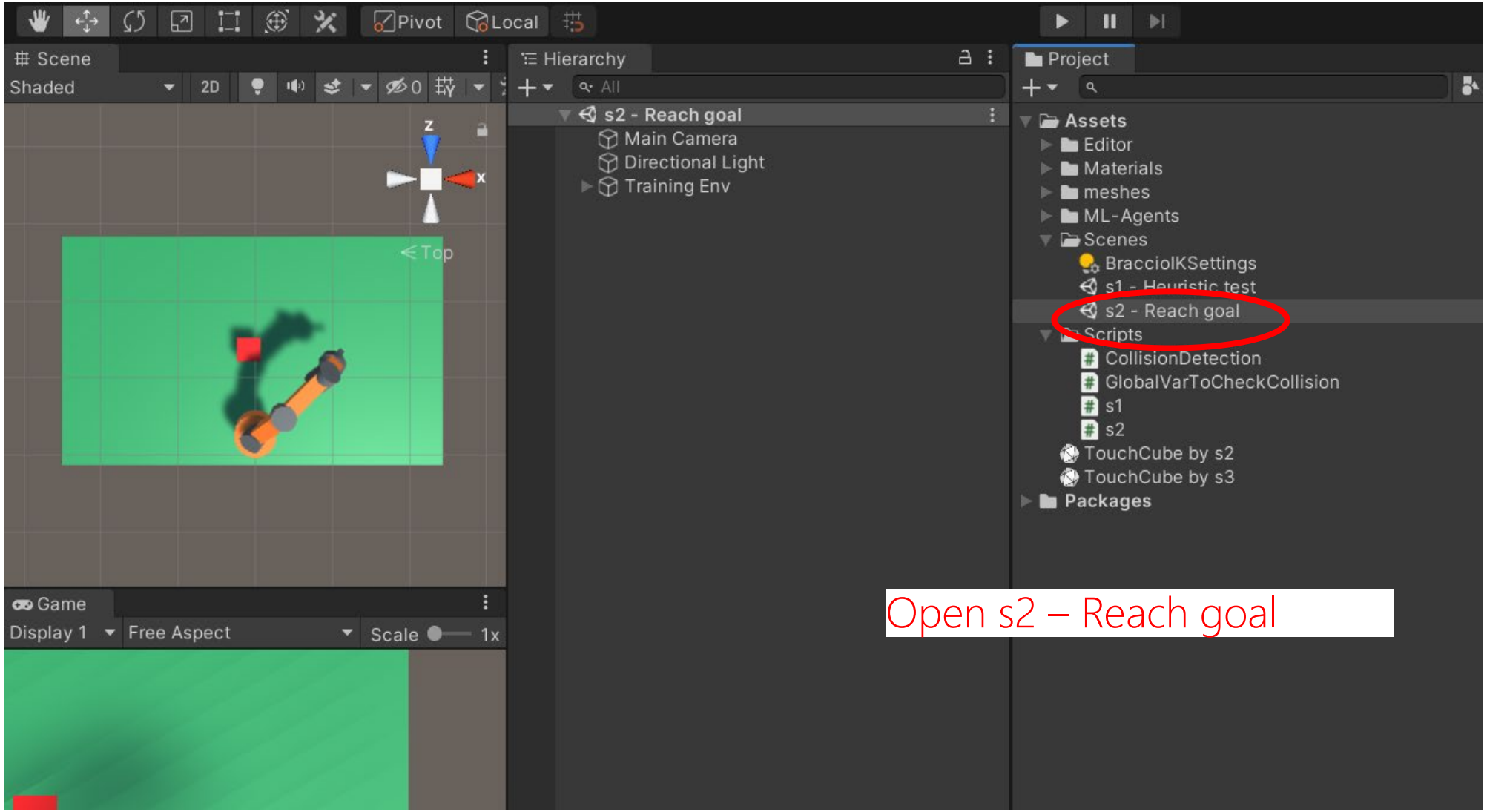
```
if (stage == 1)
{
    distToGoal = Vector3.Distance(EndTouchPlane.position, goalUpTouchPt.position);
    if (distToGoal <= 0.1f && (EndTouchPlane.position.y > goal2UpTouchPt.position.y))
    {
        stage = 2;
        print("Goal 1!\n");
        print(distToGoal + "\n");
        goal.transform.parent = EndPivot.transform; //grab goal
    }
}
else //stage =2
{
    distToGoal = Vector3.Distance(goalDownTouchPt.position, goal2UpTouchPt.position);
    if (distToGoal <= 0.1f && (goalDownTouchPt.position.y > goal2UpTouchPt.position.y))
    {
        print("Goal 2!\n");
    }
}
```

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



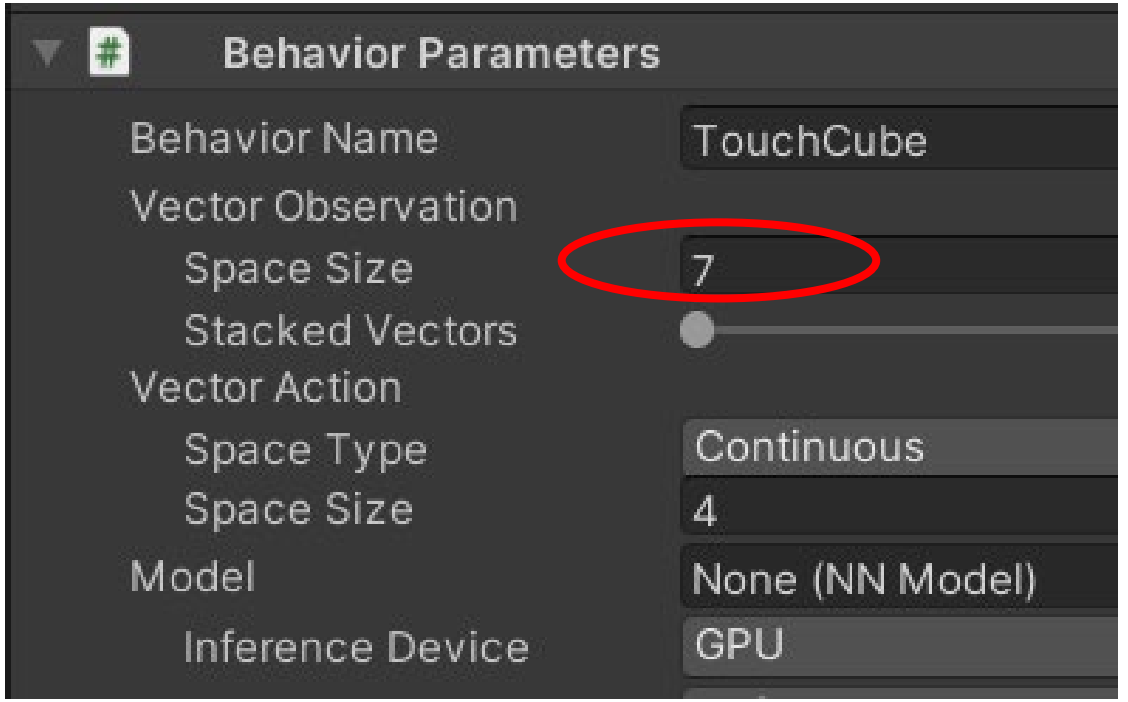
Training to reach goal

4. Train to reach goal



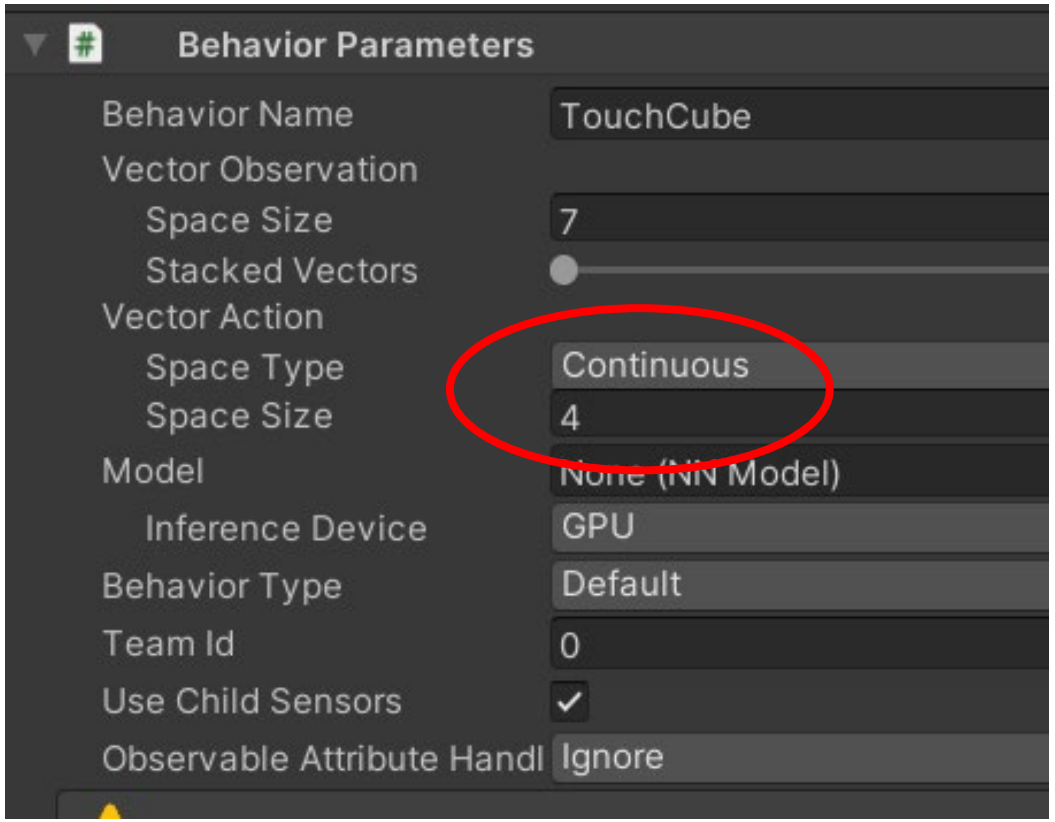
State

```
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

```
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

```
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: 1024
    buffer_size: 20480
    learning_rate: 0.0003
    beta: 0.001
    epsilon: 0.2
    lambd: 0.95
    num_epoch: 3
    learning_rate_schedule: linear
  network_settings:
    normalize: true
    hidden_units: 512
    num_layers: 3
    vis_encode_type: simple
  reward_signals:
    extrinsic:
      gamma: 0.995
      strength: 1.0
  keep_checkpoints: 5
  max_steps: 2000000
  time_horizon: 1000
  summary_freq: 50000
  threaded: true
```

Train and watch performance

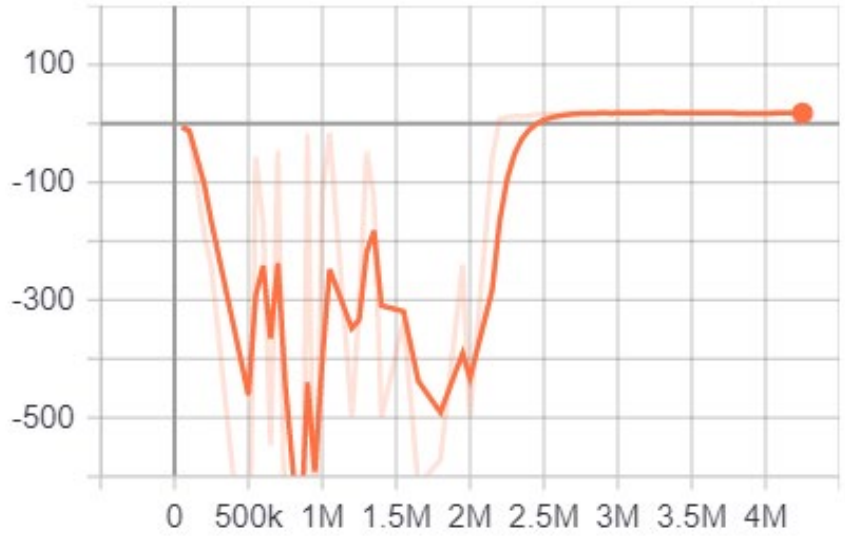
Assume ML Agent folder is placed at C:\

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

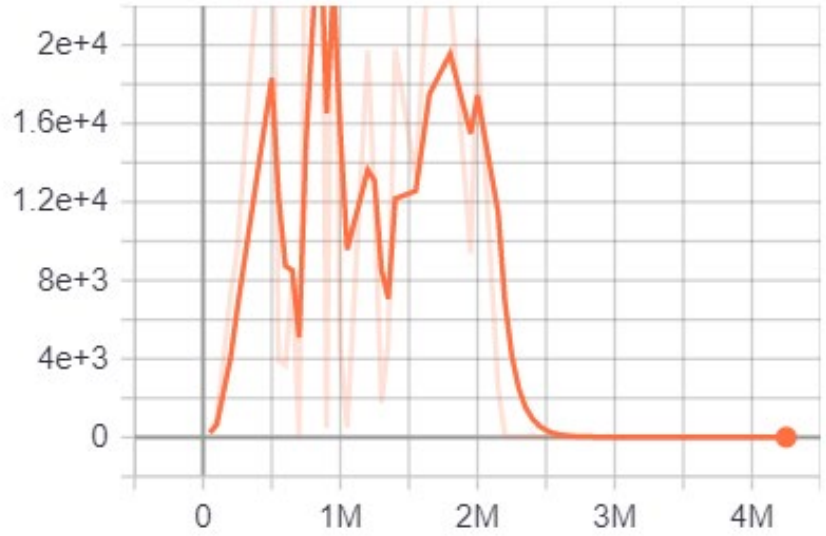
```
cd C:\ml-agents-release_10\config\ppo\results
tensorboard --logdir=1
```

```
TouchCube. Step: 2650000. Time Elapsed: 3658.100 s. Mean Reward:  17.614. Std of Reward:  6.833. Training.
TouchCube. Step: 2700000. Time Elapsed: 3733.506 s. Mean Reward:  17.876. Std of Reward:  6.707. Training.
TouchCube. Step: 2750000. Time Elapsed: 3807.903 s. Mean Reward:  18.228. Std of Reward:  5.854. Training.
TouchCube. Step: 2800000. Time Elapsed: 3884.155 s. Mean Reward:  17.309. Std of Reward:  52.468. Training.
TouchCube. Step: 2850000. Time Elapsed: 3964.688 s. Mean Reward:  17.973. Std of Reward:  6.358. Training.
TouchCube. Step: 2900000. Time Elapsed: 4041.246 s. Mean Reward:  18.081. Std of Reward:  6.204. Training.
TouchCube. Step: 2950000. Time Elapsed: 4123.769 s. Mean Reward:  17.064. Std of Reward:  44.831. Training.
TouchCube. Step: 3000000. Time Elapsed: 4203.710 s. Mean Reward:  17.803. Std of Reward:  6.677. Training.
ation.py:93] Converting to results\1\TouchCube\TouchCube-2999987.onnx
ation.py:105] Exported results\1\TouchCube\TouchCube-2999987.onnx
ager.py:43] Removed checkpoint model results\1\TouchCube\TouchCube-4997[illegible].onnx.
TouchCube. Step: 3050000. Time Elapsed: 4287.124 s. Mean Reward:  17.977. Std of Reward:  6.411. Training.
TouchCube. Step: 3100000. Time Elapsed: 4367.629 s. Mean Reward:  18.150. Std of Reward:  6.118. Training.
TouchCube. Step: 3150000. Time Elapsed: 4448.134 s. Mean Reward:  18.524. Std of Reward:  5.416. Training.
TouchCube. Step: 3200000. Time Elapsed: 4532.615 s. Mean Reward:  18.634. Std of Reward:  5.199. Training.
TouchCube. Step: 3250000. Time Elapsed: 4614.547 s. Mean Reward:  18.582. Std of Reward:  5.335. Training.
TouchCube. Step: 3300000. Time Elapsed: 4698.999 s. Mean Reward:  18.367. Std of Reward:  5.753. Training.
TouchCube. Step: 3350000. Time Elapsed: 4780.770 s. Mean Reward:  18.247. Std of Reward:  5.985. Training.
TouchCube. Step: 3400000. Time Elapsed: 4865.399 s. Mean Reward:  18.335. Std of Reward:  5.818. Training.
TouchCube. Step: 3450000. Time Elapsed: 4947.306 s. Mean Reward:  17.813. Std of Reward:  6.726. Training.
TouchCube. Step: 3500000. Time Elapsed: 5029.465 s. Mean Reward:  17.731. Std of Reward:  6.854. Training.
ation.py:93] Converting to results\1\TouchCube\TouchCube-3499993.onnx
ation.py:105] Exported results\1\TouchCube\TouchCube-3499993.onnx
ager.py:43] Removed checkpoint model results\1\TouchCube\TouchCube-9997[illegible].onnx.
TouchCube. Step: 3550000. Time Elapsed: 5114.902 s. Mean Reward:  17.368. Std of Reward:  7.377. Training.
TouchCube. Step: 3600000. Time Elapsed: 5197.616 s. Mean Reward:  17.534. Std of Reward:  7.143. Training.
TouchCube. Step: 3650000. Time Elapsed: 5287.537 s. Mean Reward:  17.599. Std of Reward:  7.053. Training.
TouchCube. Step: 3700000. Time Elapsed: 5371.427 s. Mean Reward:  17.704. Std of Reward:  6.902. Training.
TouchCube. Step: 3750000. Time Elapsed: 5459.356 s. Mean Reward:  17.503. Std of Reward:  7.198. Training.
TouchCube. Step: 3800000. Time Elapsed: 5548.941 s. Mean Reward:  17.260. Std of Reward:  7.530. Training.
TouchCube. Step: 3850000. Time Elapsed: 5637.050 s. Mean Reward:  17.444. Std of Reward:  7.284. Training.
TouchCube. Step: 3900000. Time Elapsed: 5722.583 s. Mean Reward:  17.374. Std of Reward:  7.380. Training.
TouchCube. Step: 3950000. Time Elapsed: 5809.752 s. Mean Reward:  17.124. Std of Reward:  7.701. Training.
TouchCube. Step: 4000000. Time Elapsed: 5901.880 s. Mean Reward:  17.342. Std of Reward:  7.424. Training.
ation.py:93] Converting to results\1\TouchCube\TouchCube-3999991.onnx
ation.py:105] Exported results\1\TouchCube\TouchCube-3999991.onnx
ager.py:43] Removed checkpoint model results\1\TouchCube\TouchCube-1499[illegible]4.onnx.
TouchCube. Step: 4050000. Time Elapsed: 5985.685 s. Mean Reward:  17.703. Std of Reward:  6.913. Training.
TouchCube. Step: 4100000. Time Elapsed: 6072.875 s. Mean Reward:  17.631. Std of Reward:  7.014. Training.
TouchCube. Step: 4150000. Time Elapsed: 6157.968 s. Mean Reward:  17.791. Std of Reward:  6.771. Training.
TouchCube. Step: 4200000. Time Elapsed: 6245.044 s. Mean Reward:  17.452. Std of Reward:  7.270. Training.
TouchCube. Step: 4250000. Time Elapsed: 6329.007 s. Mean Reward:  17.412. Std of Reward:  7.326. Training.
```

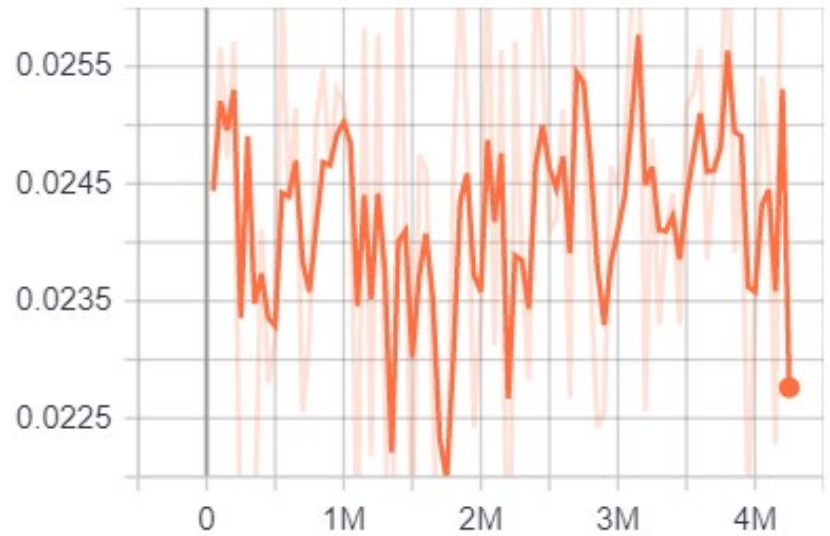
Cumulative Reward
tag: Environment/Cumulative Reward



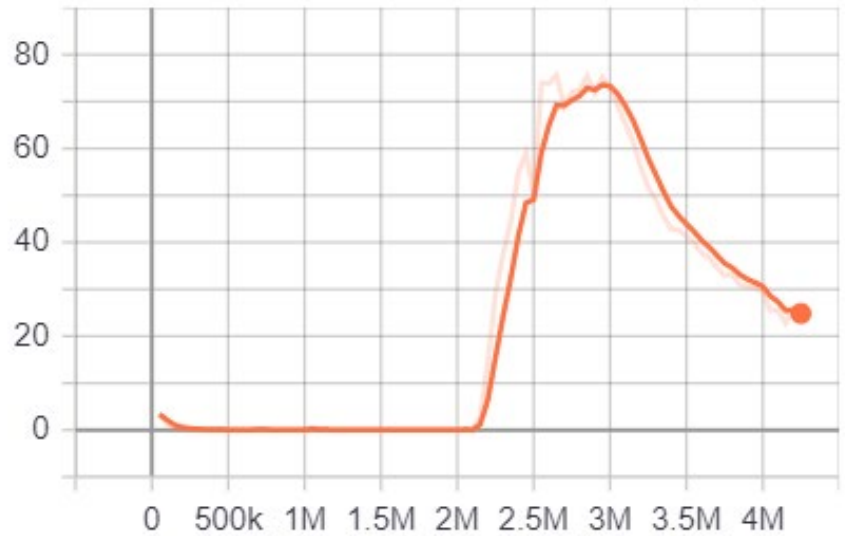
Episode Length
tag: Environment/Episode Length



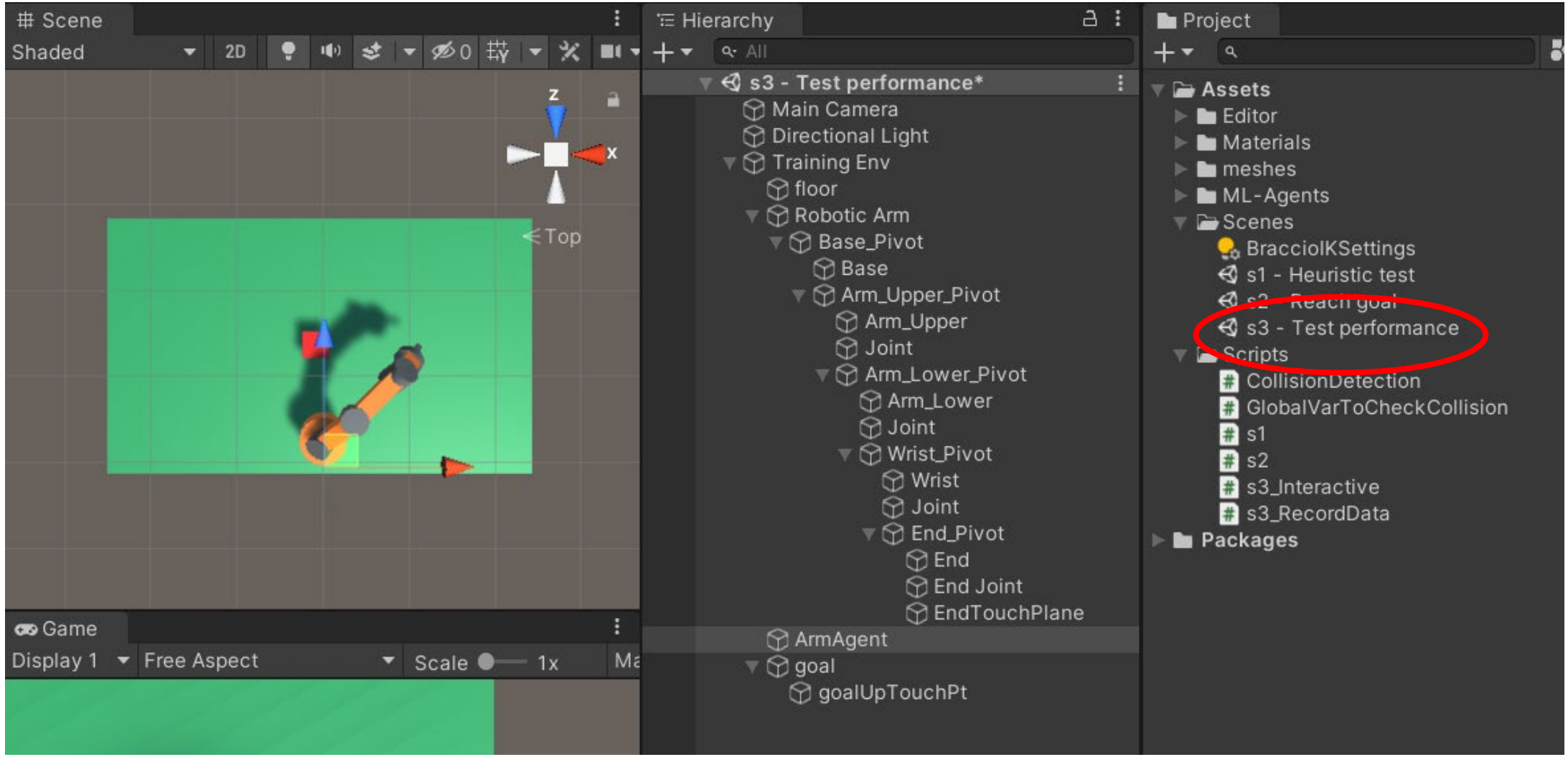
Policy Loss
tag: Losses/Policy Loss



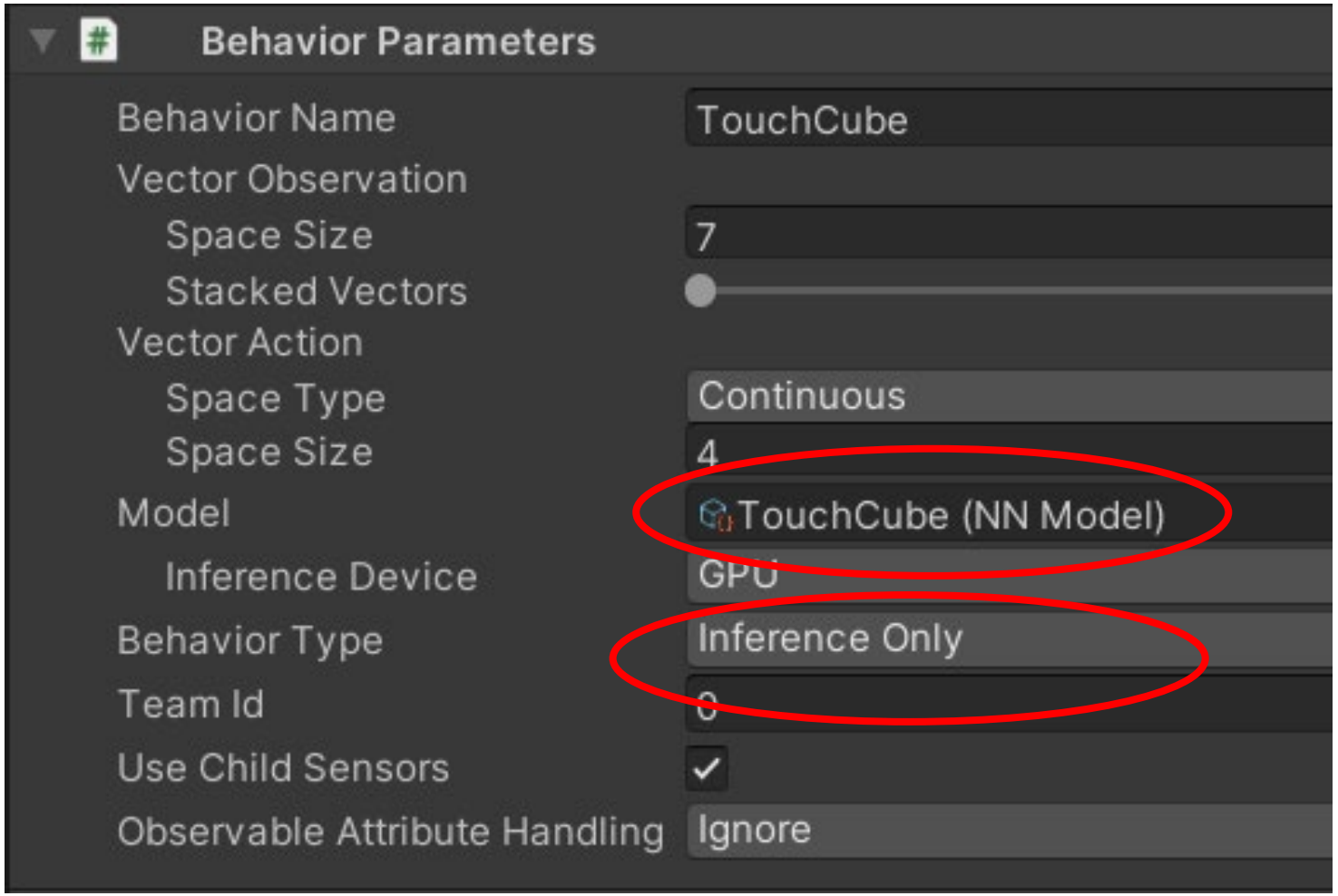
Value Loss
tag: Losses/Value Loss



5. Test performance

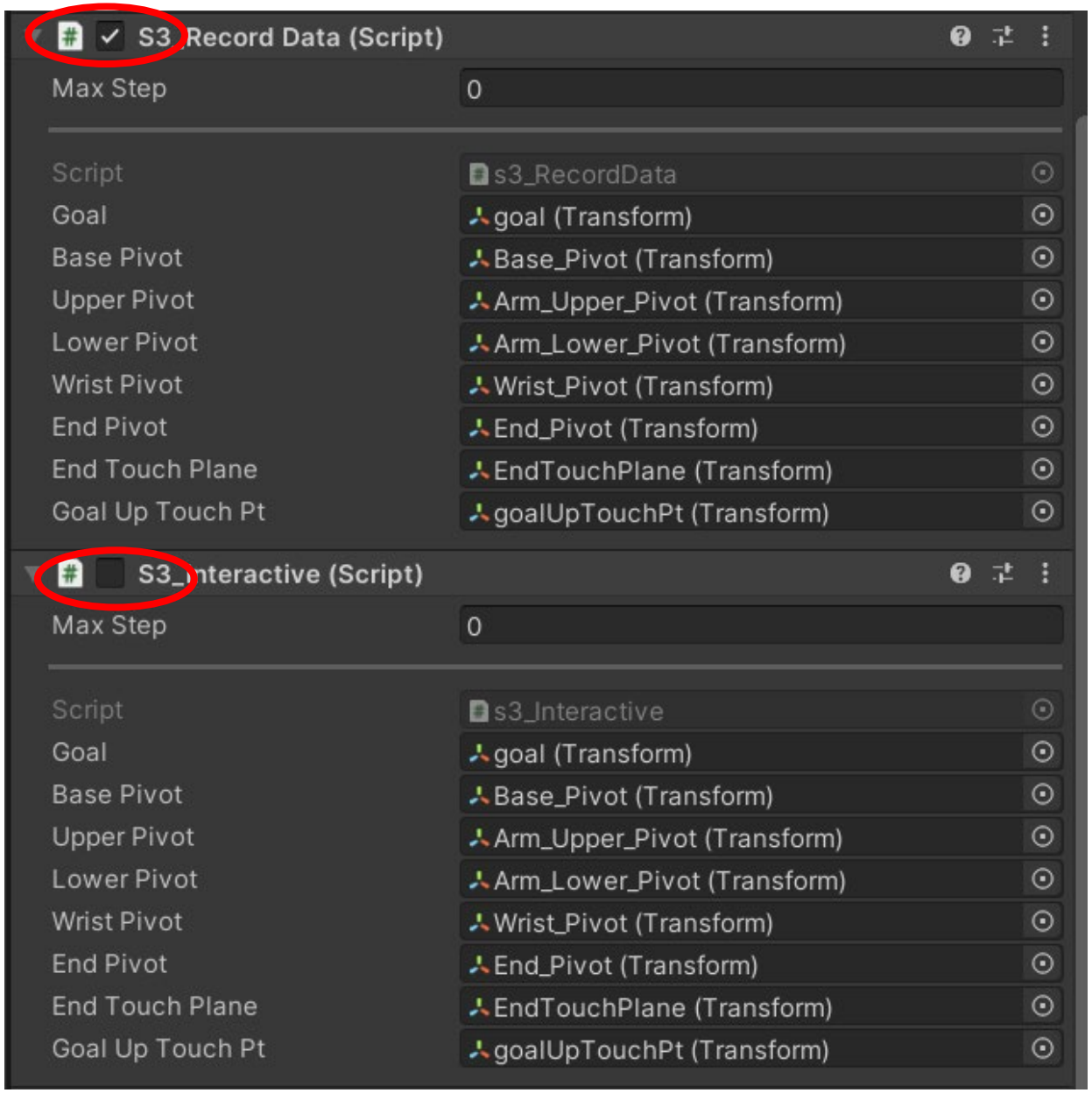


Assign trained NN



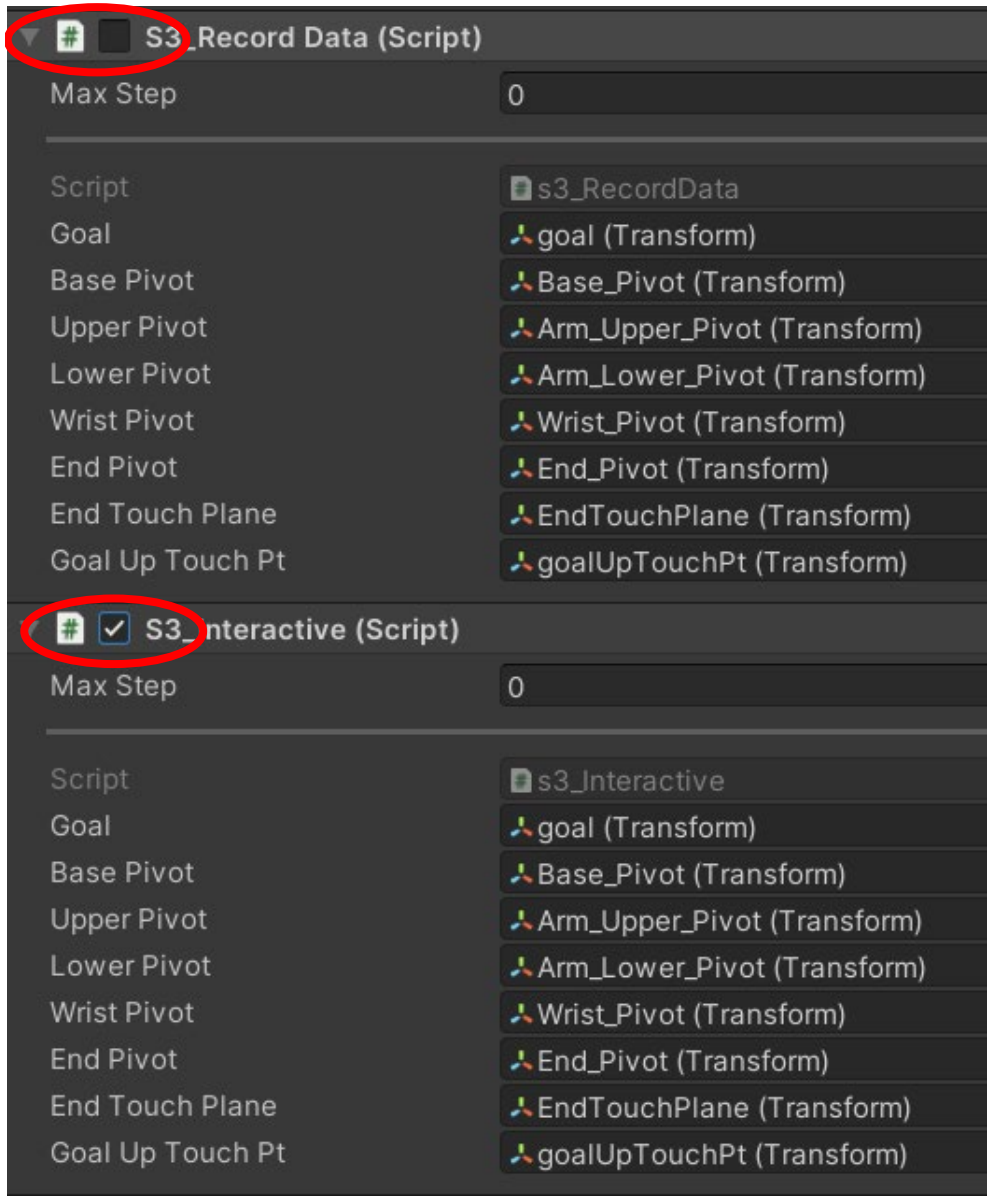
Data-recording test

Uncheck interactive test



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)

Training to reach goal 1 → goal 2