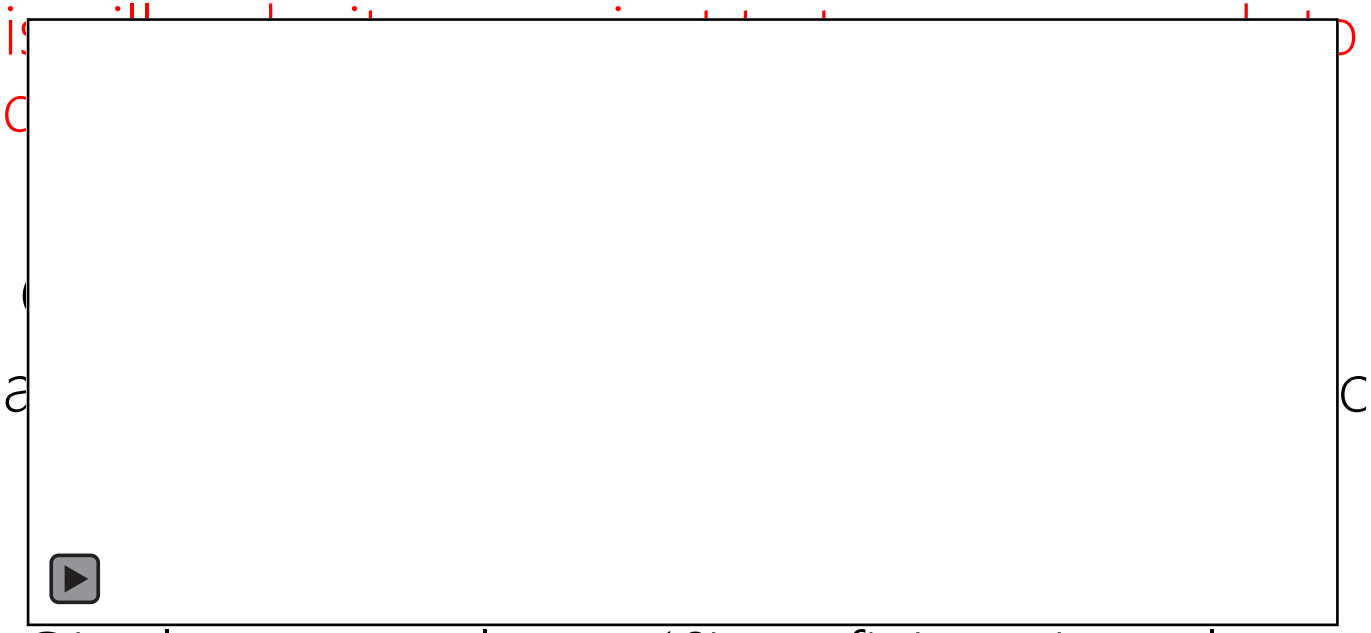
Build intelligent robot that can interact with human

1. Download and save ML Agent to C:\

This will [illegible] to train and

cd [illegible]

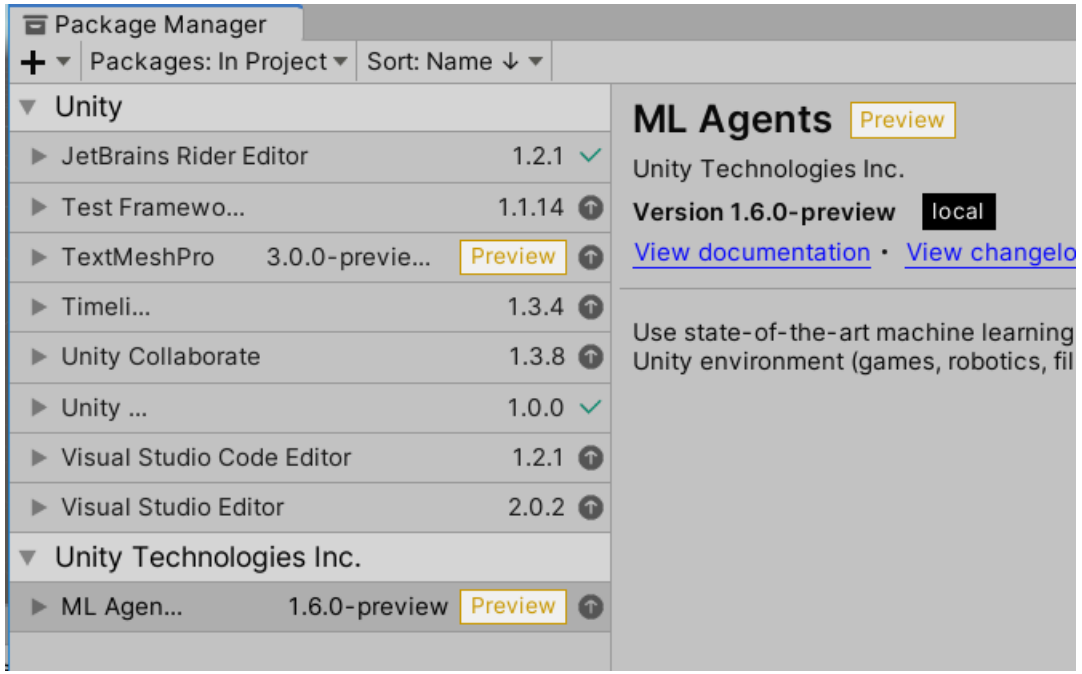
mla[illegible]ce

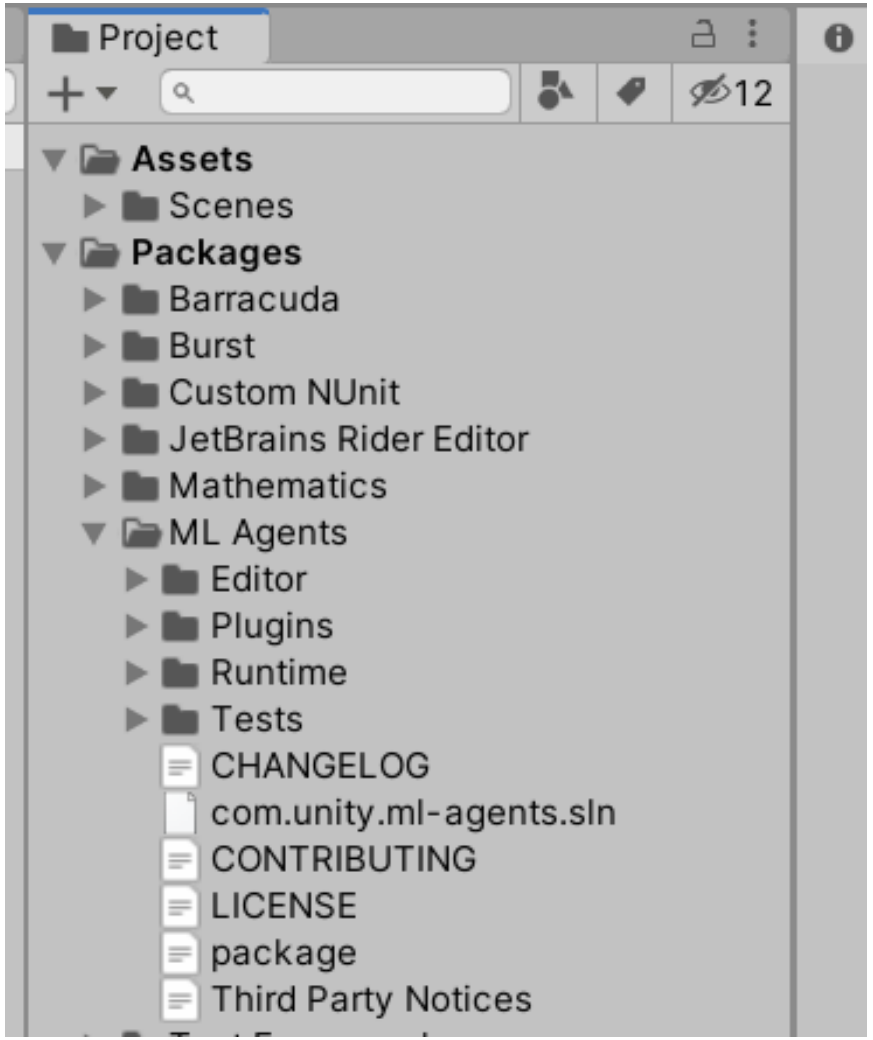


cd C:\ml-agents-release_10\config\ppo\results

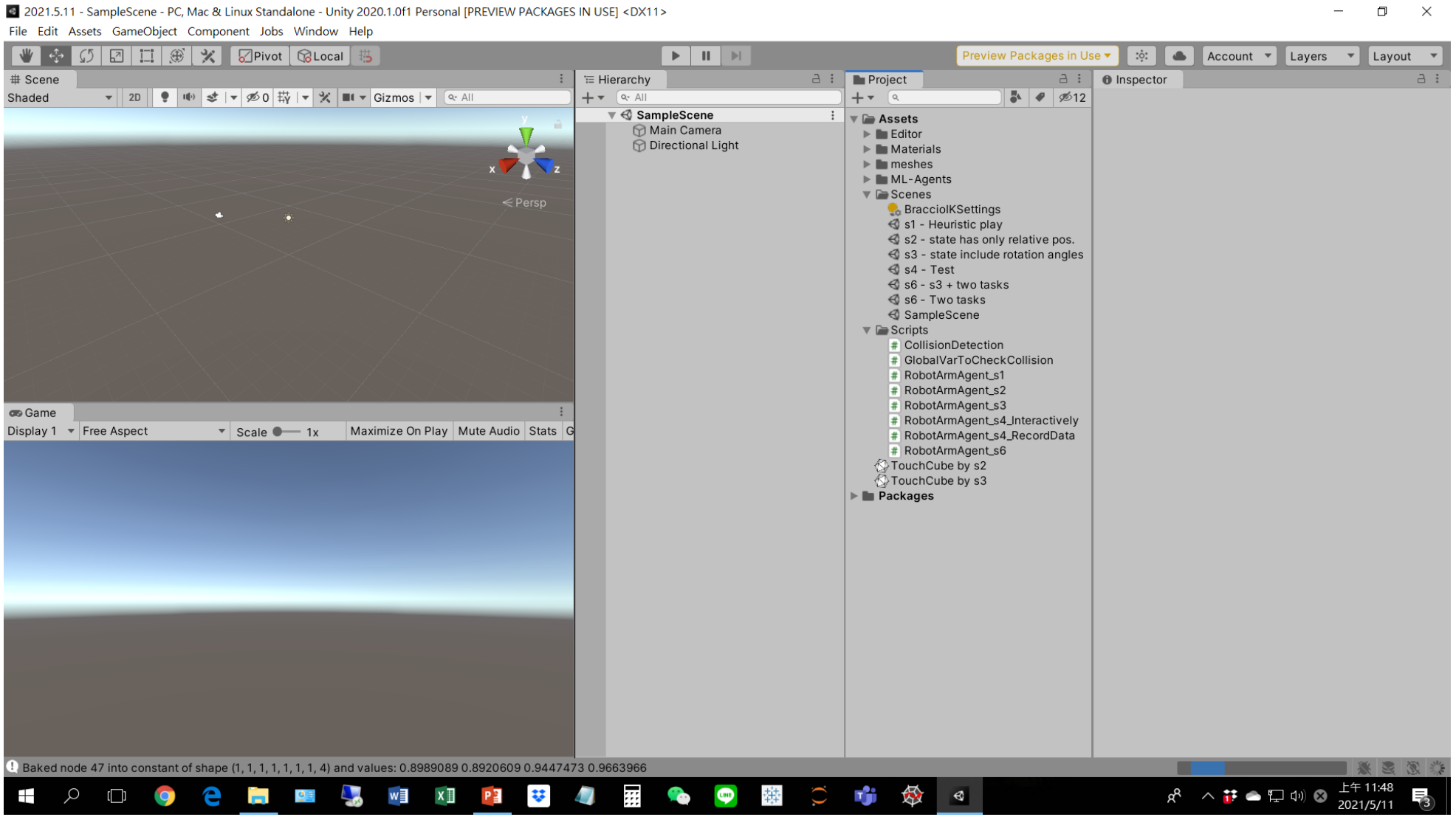
tensorboard --logdir=1

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

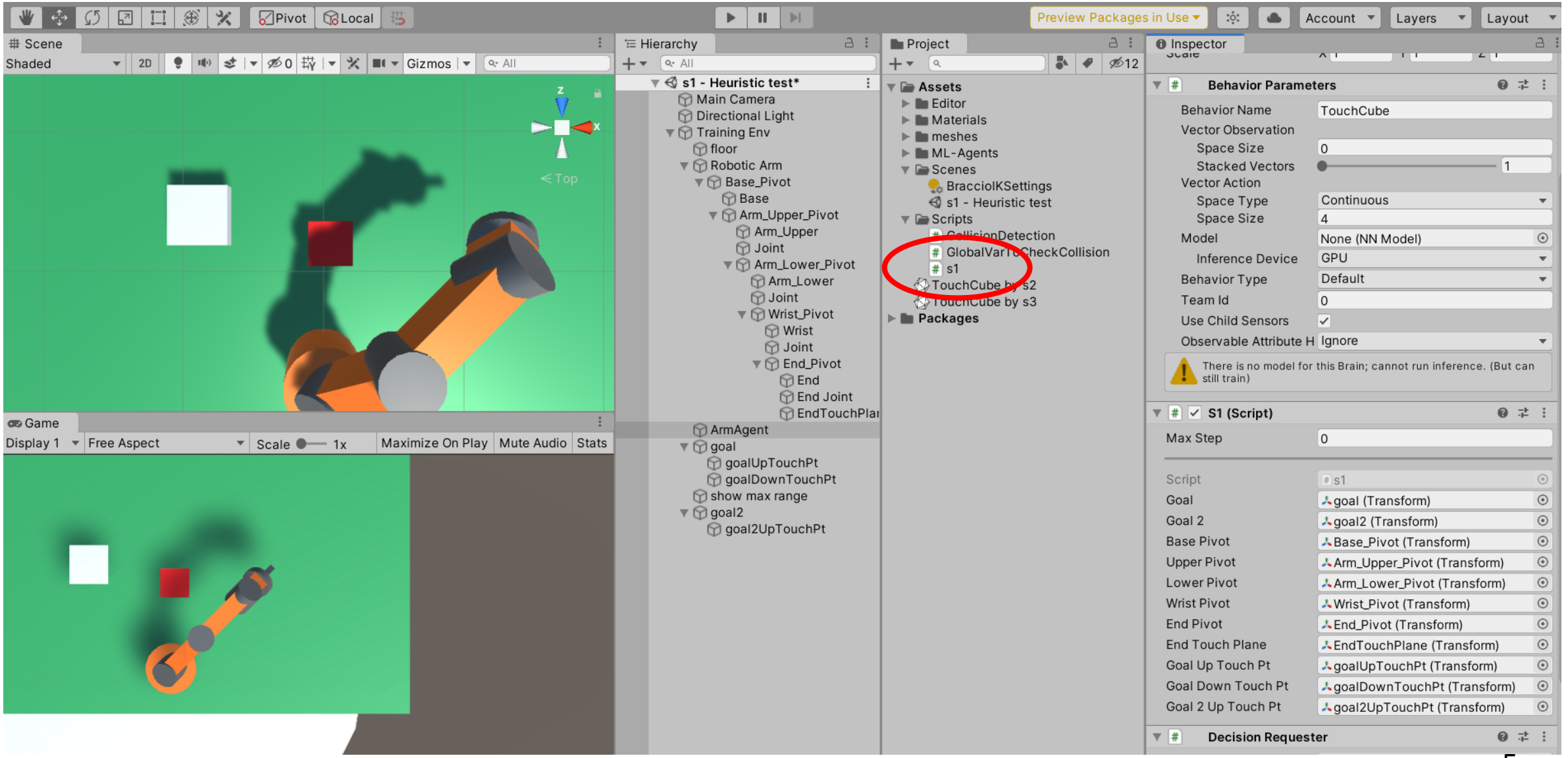




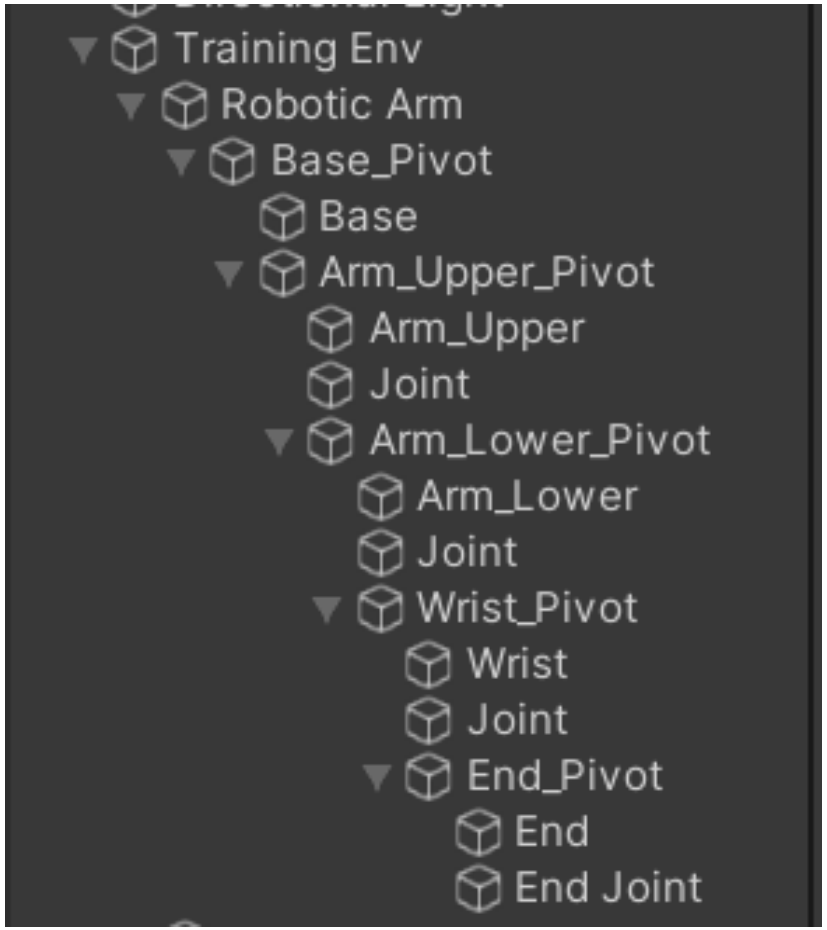
3. Import Robot arm package to Unity project

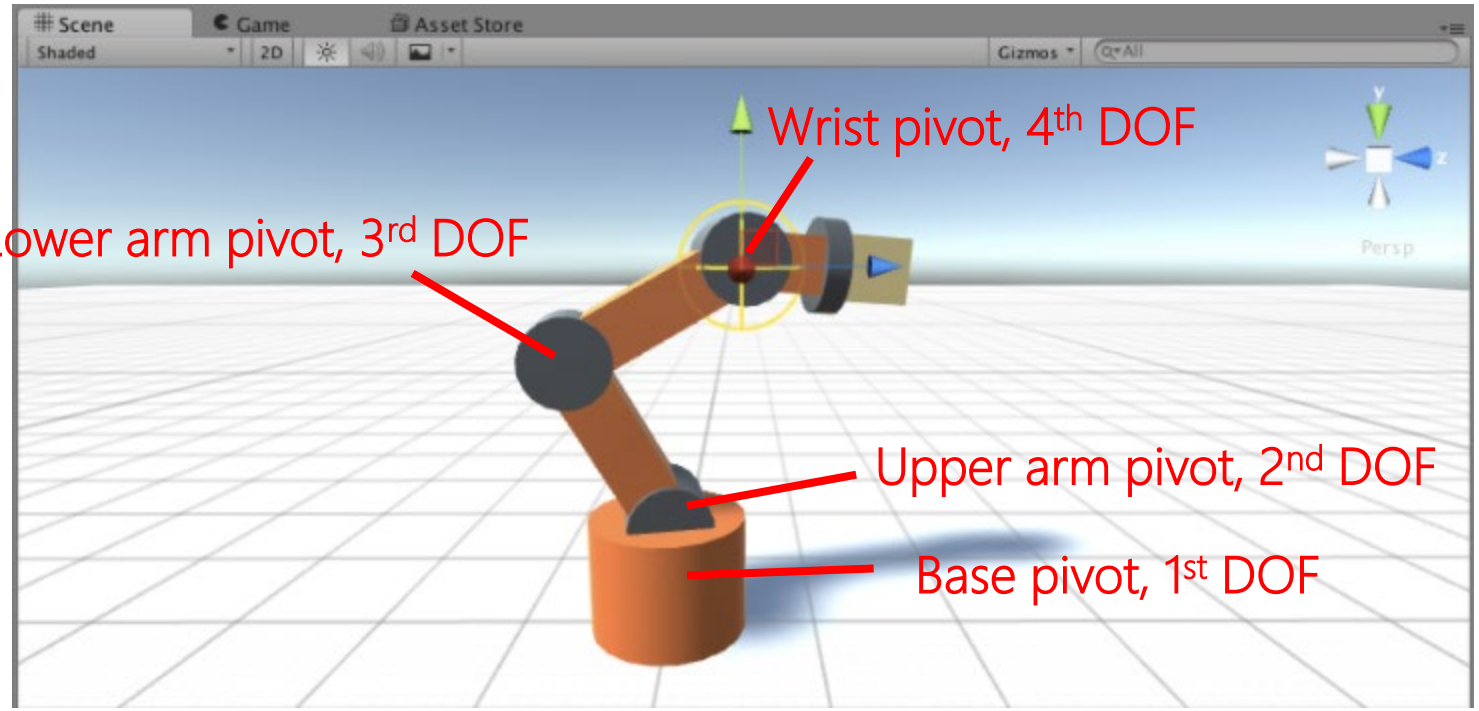


Open scene "s1 - Heuristic play"



This Unity project contains a Braccio robot arm



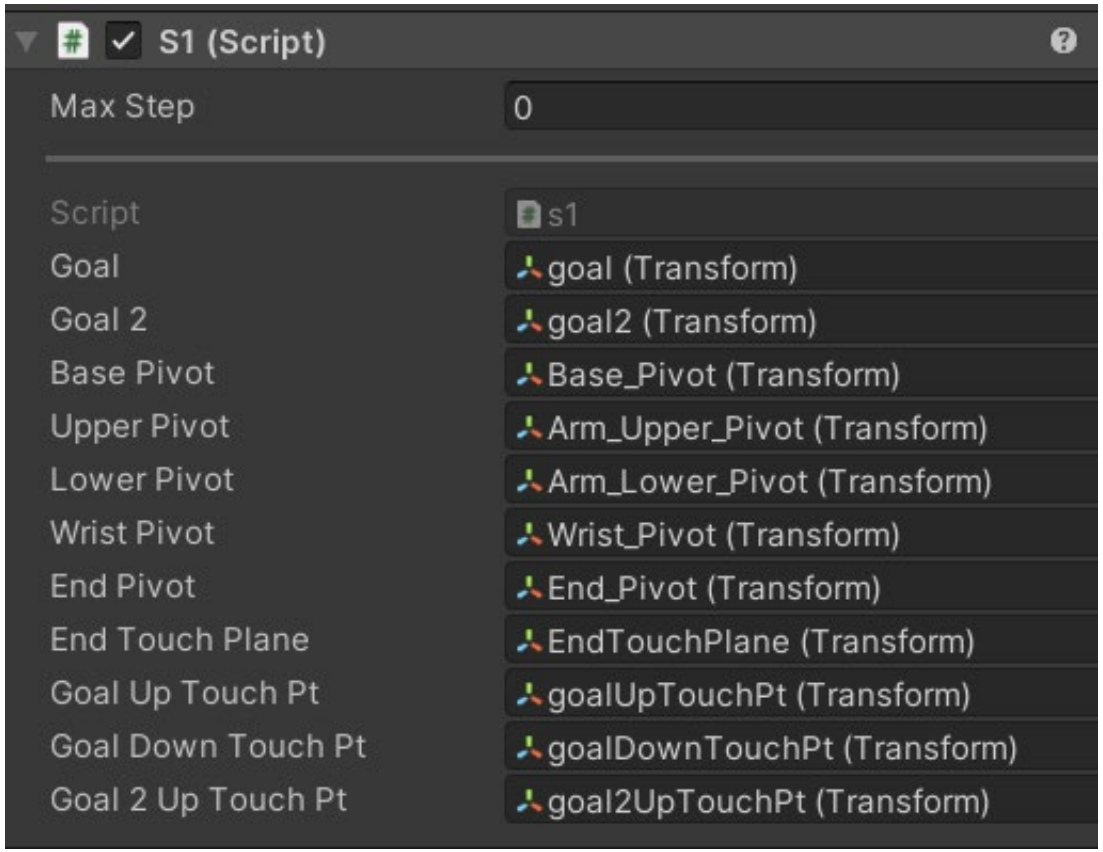


https://github.com/tanyuan/braccio-ik-unity

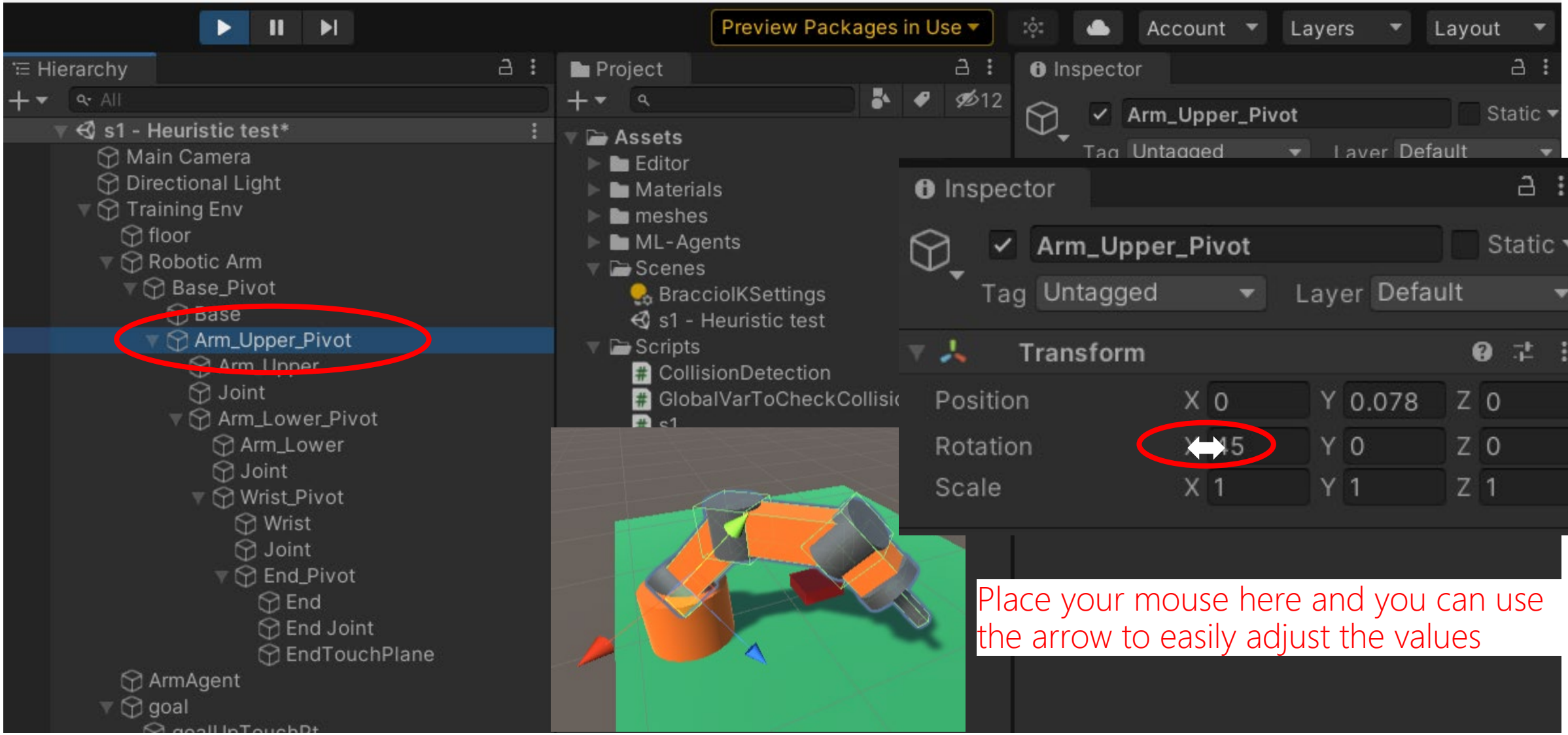
(1) Manually operates the robot arm in VE

Public variables to link agent script with scene objects

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



Play, rotate arm by changing rotation angle in Inspector window

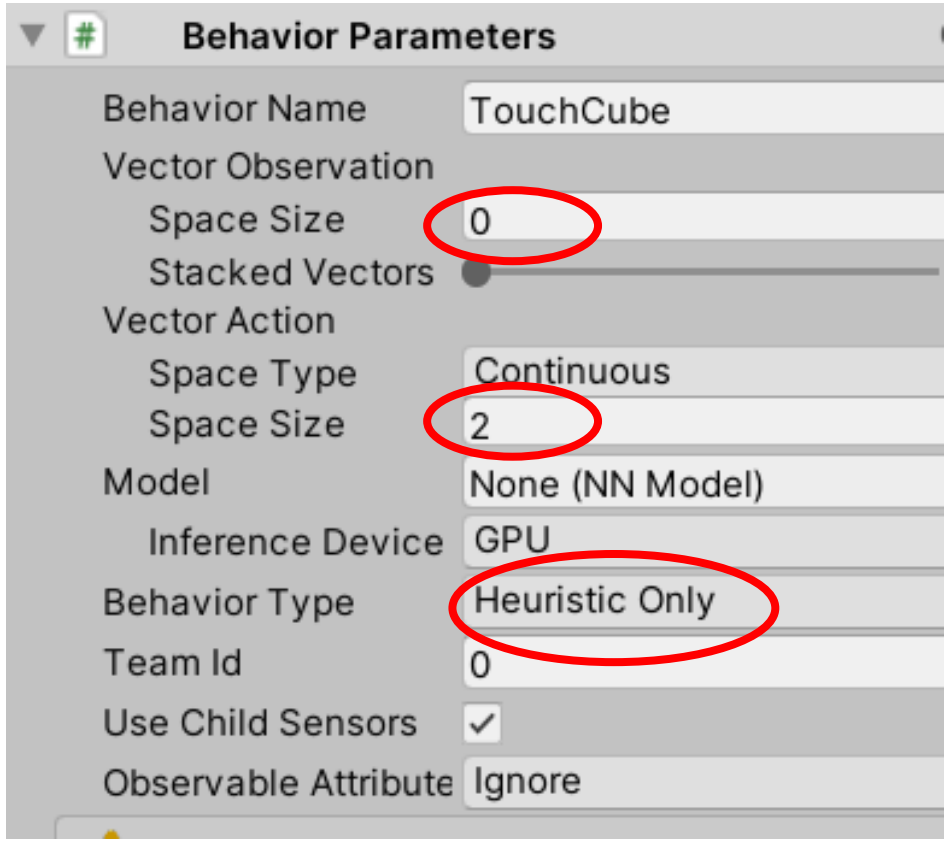


Use Up/Down, L/R keys to rotate arm

← and → key

↑ and ↓ key

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

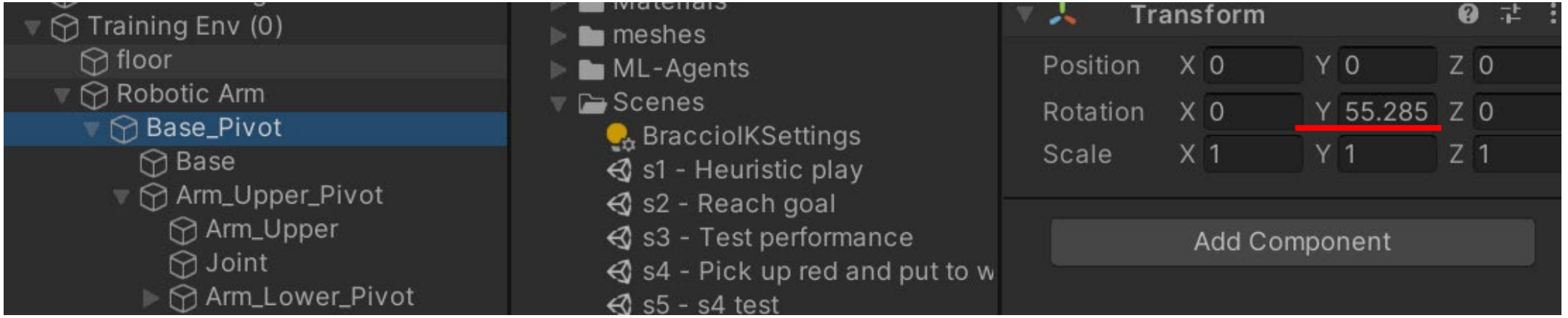


Base -90 ~ 90

U/L arm and wrist : 0 ~ 90

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

UpperPivot.Rotate(vectorAction[1] * speed, 0, 0);
//float RotationAngle = UnityEditor.TransformUtils.GetInspectorRotation(UpperPivot).x;
```

Check whether arm rotation is out of range




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

Collision detections

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

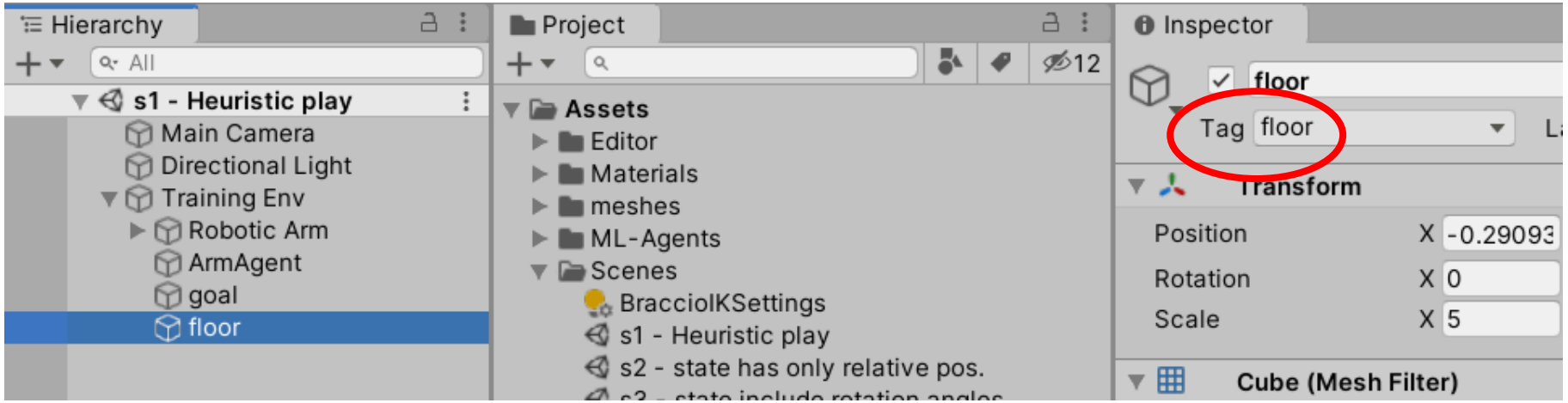
```
public class MyGlobalVar : MonoBehaviour
{
    public static bool LowerArmCollisionHappens = false;
    public static bool WristCollisionHappens = false;
    public static bool EndCollisionHappens = false;
    public static bool goalCollisionHappens = false;
```

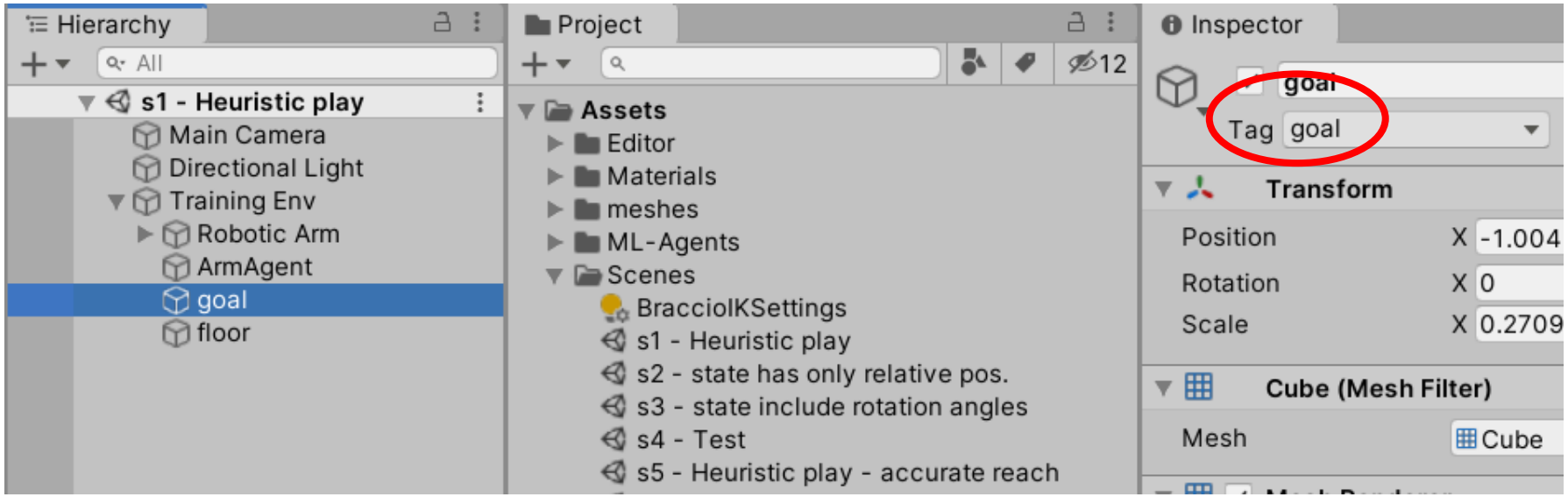
On trigger enter/exist to record collisions of lower arm, wrist, and end with floor and goal objects

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

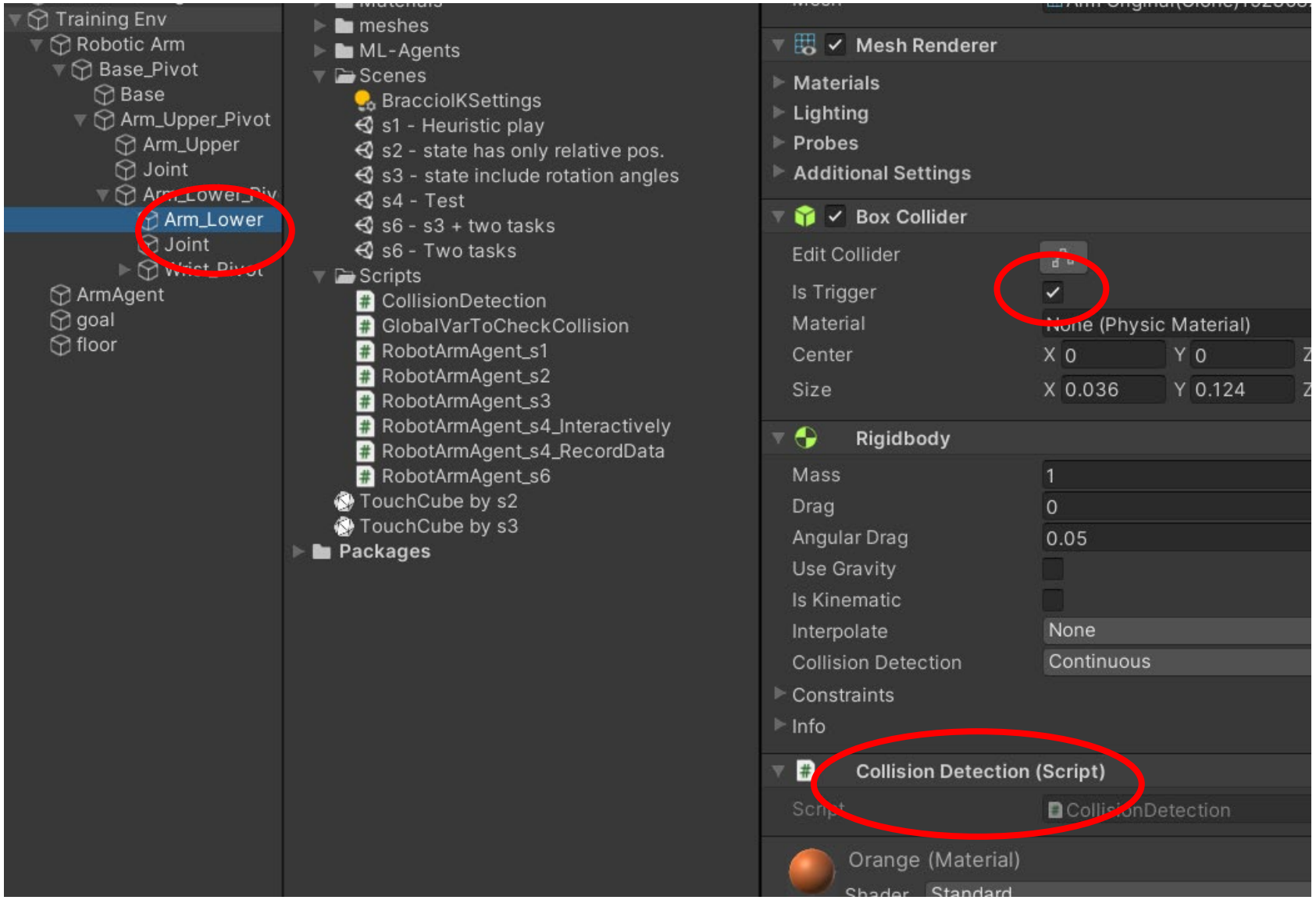
```
void OnTriggerExit(Collider other)
{
    if (other.gameObject.tag == "floor"
    {
        if (this.gameObject.tag == "Lowe
            MyGlobalVar.LowerArmCollisic
        else if (this.gameObject.tag ==
```

Add tags to floor and goal object

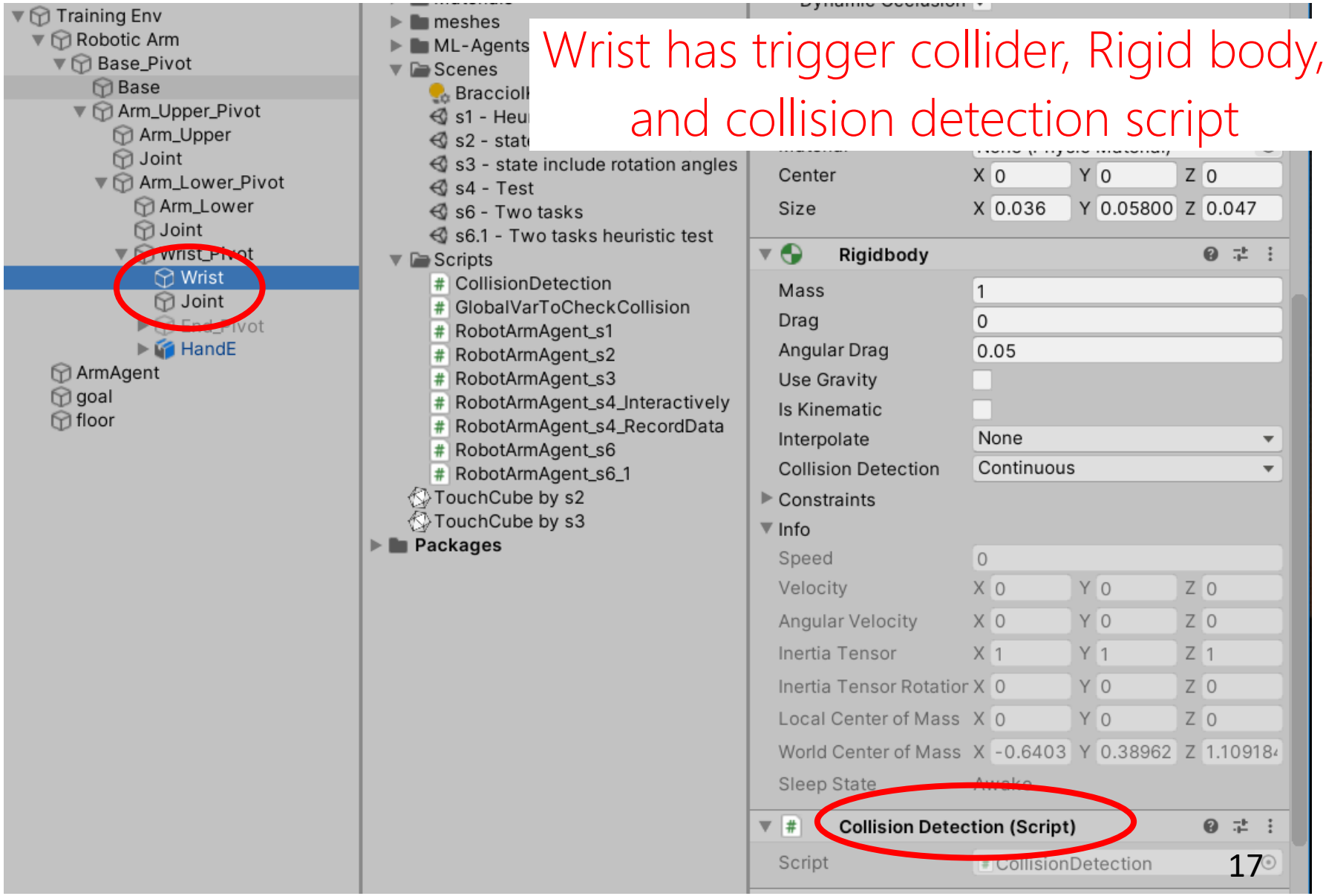




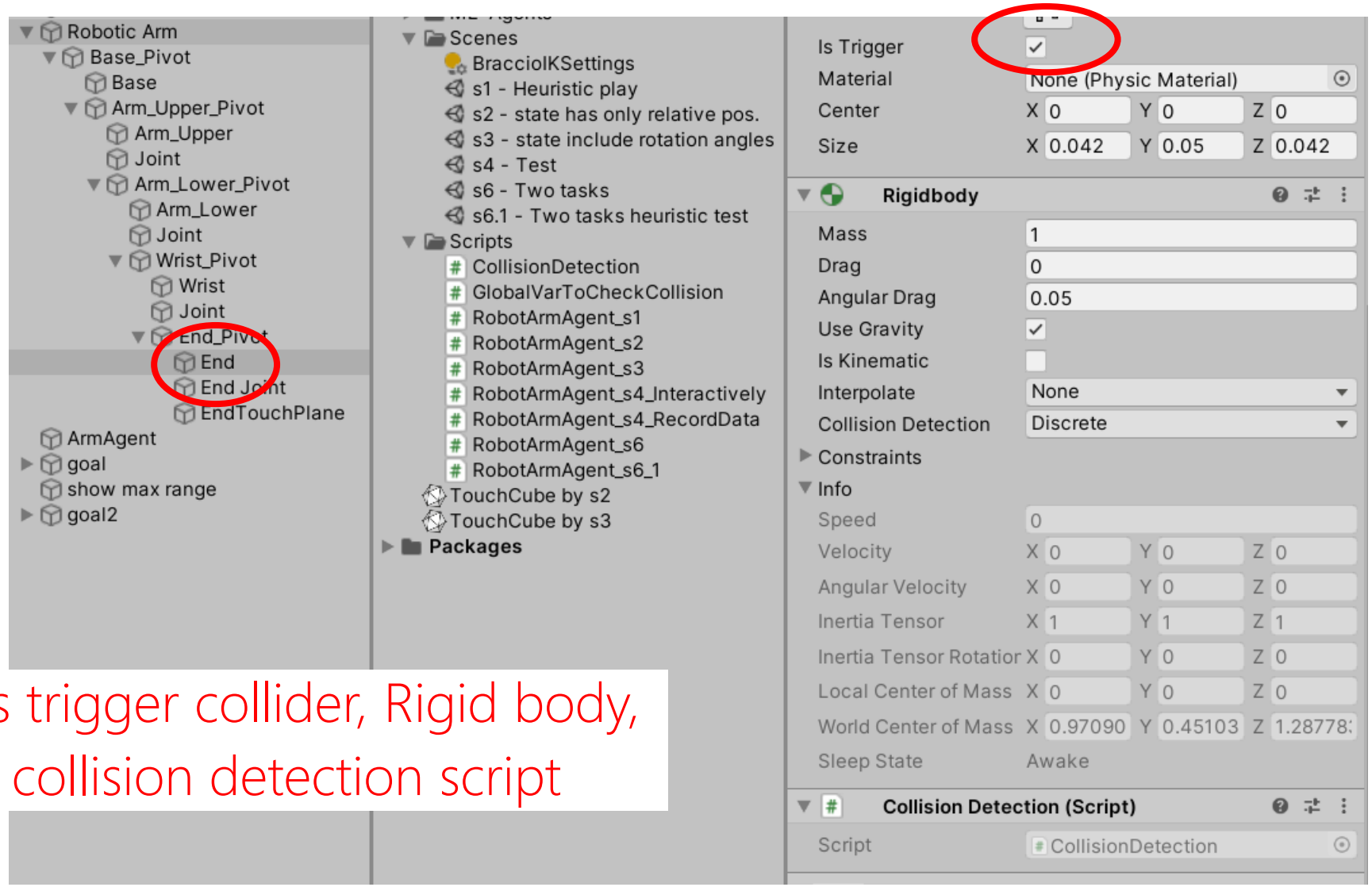
Add trigger collider, Rigid body, and collision detection script to Lower Arm



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

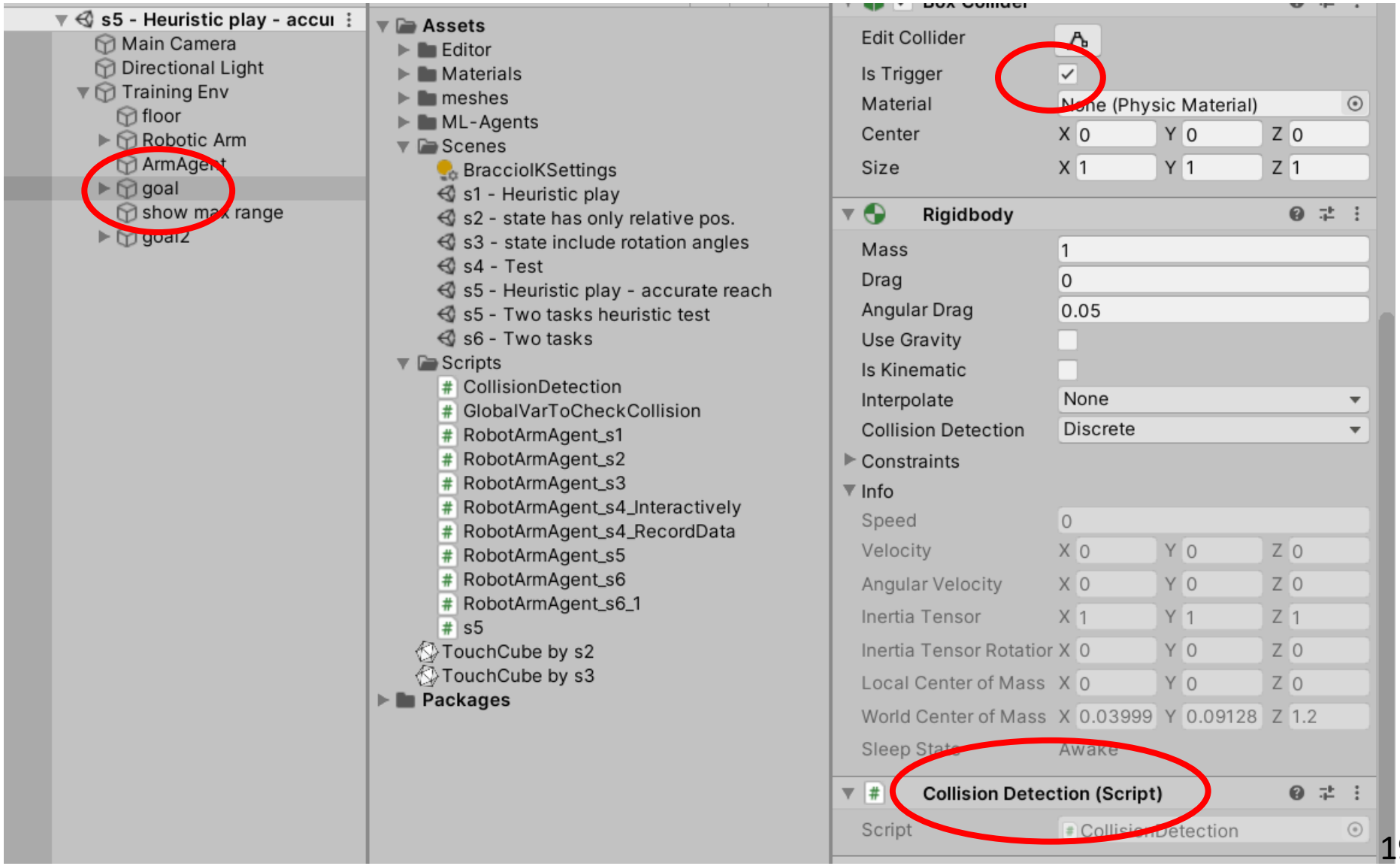


Add trigger collider, Rigid body, and collision detection script to Robot End

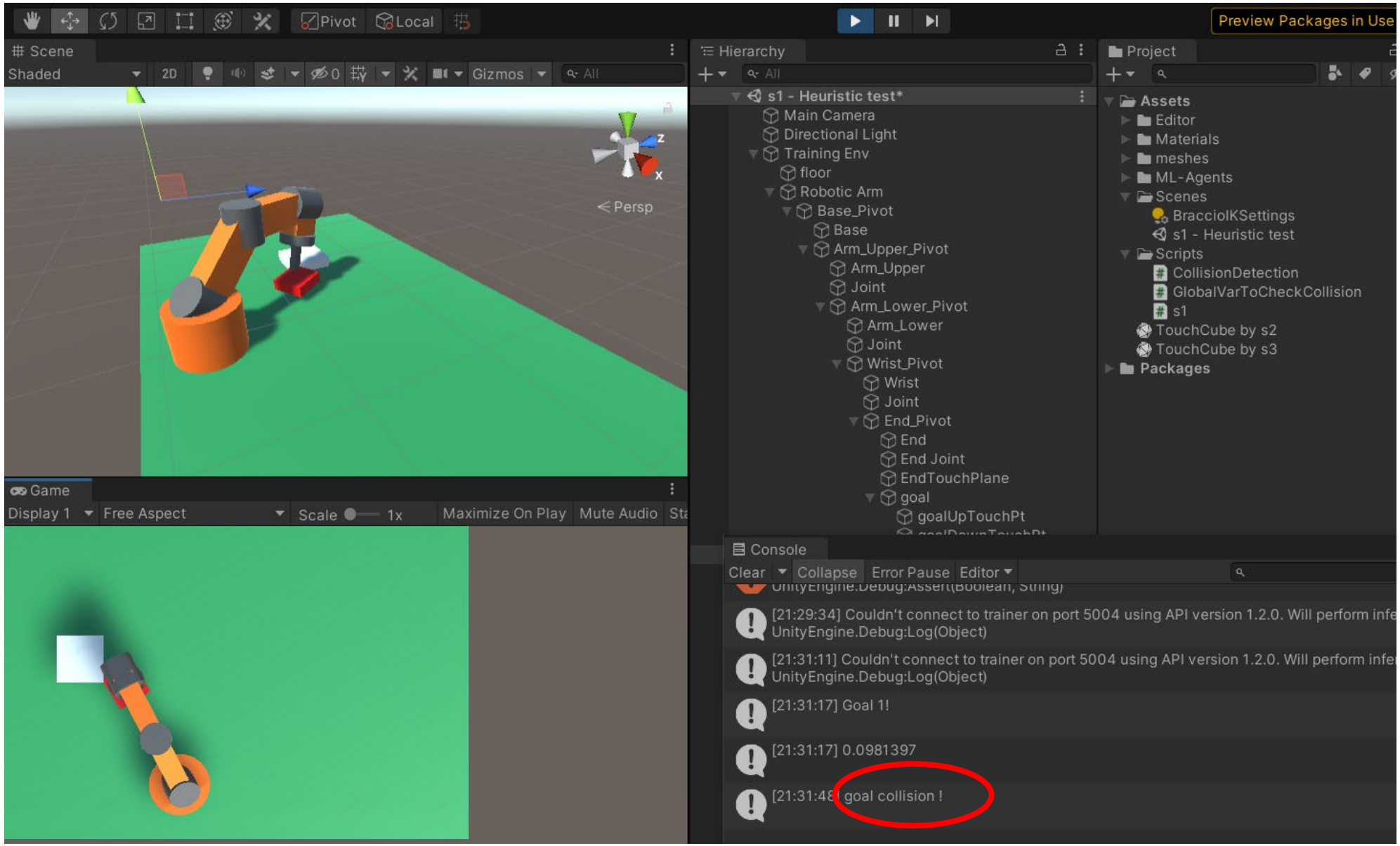


End has trigger collider, Rigid body, and collision detection script

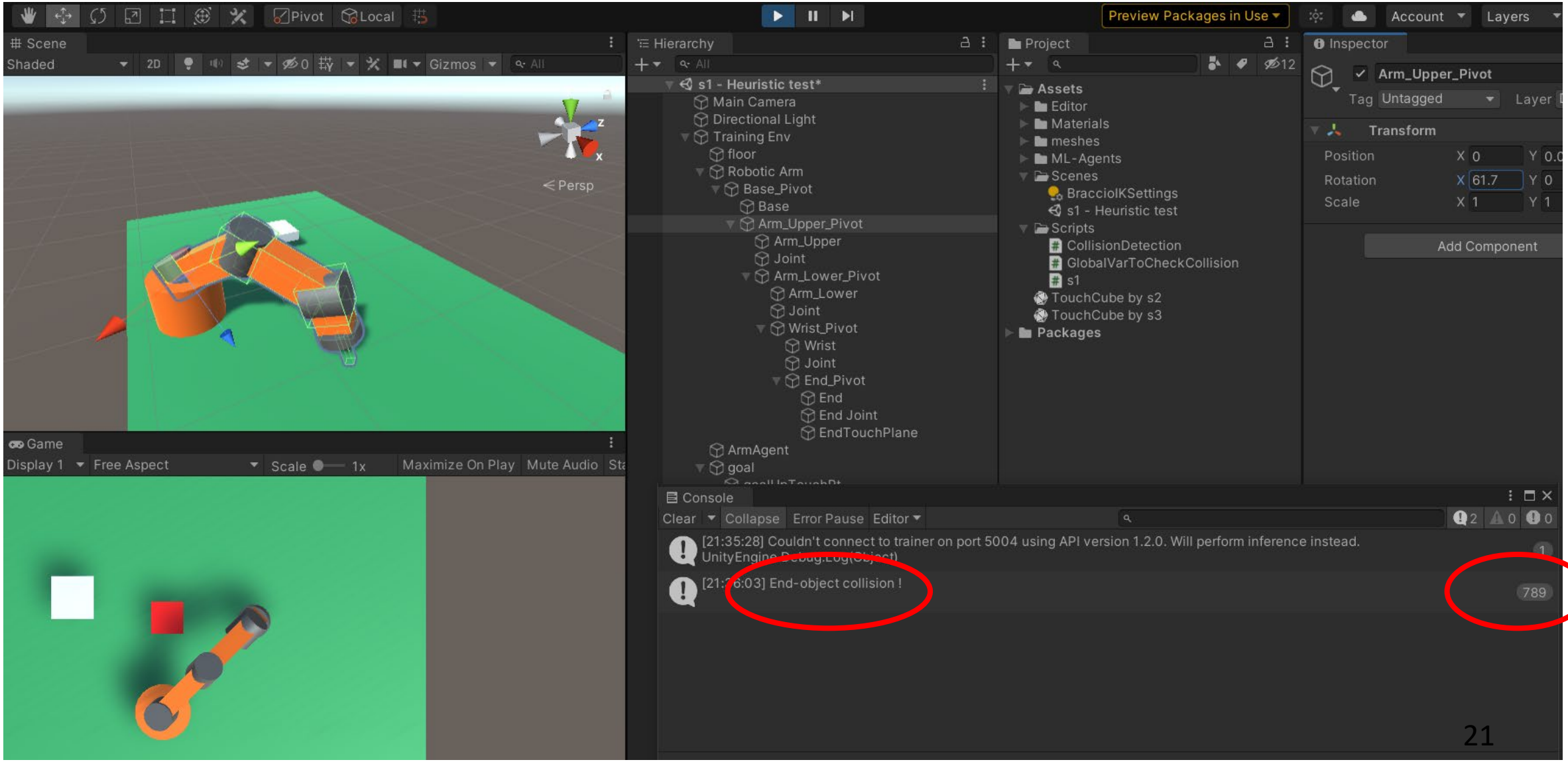
Add trigger collider, Rigid body, and collision detection script to goal object



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

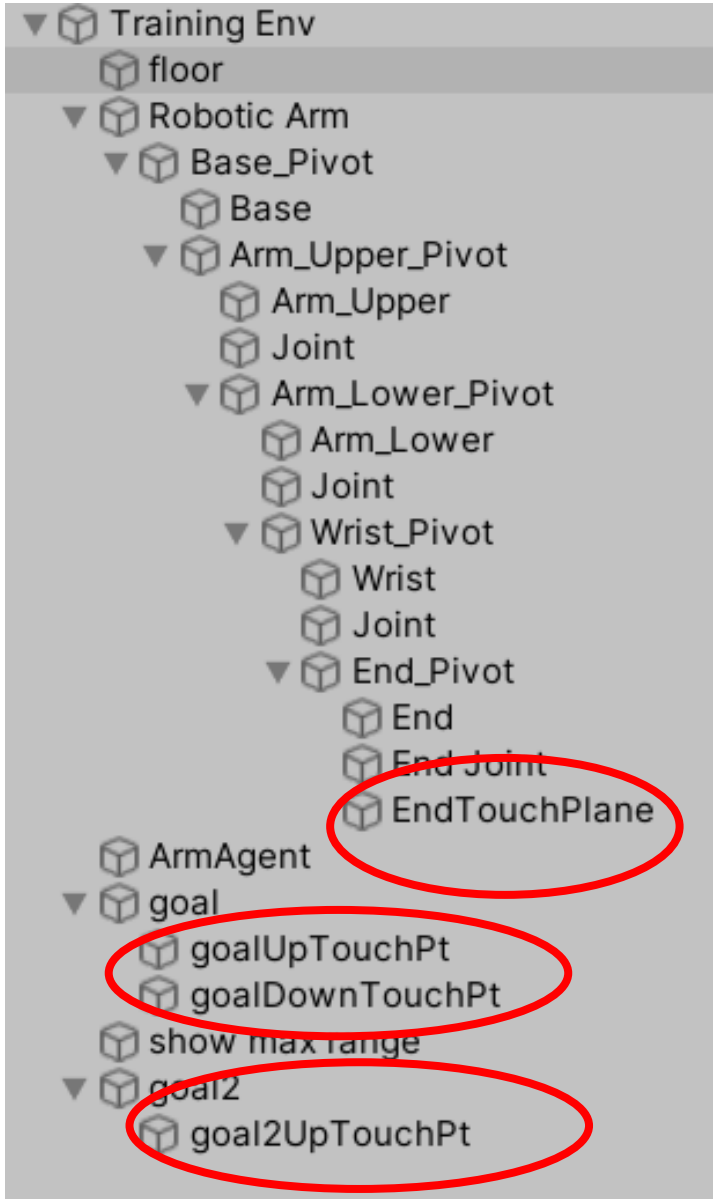


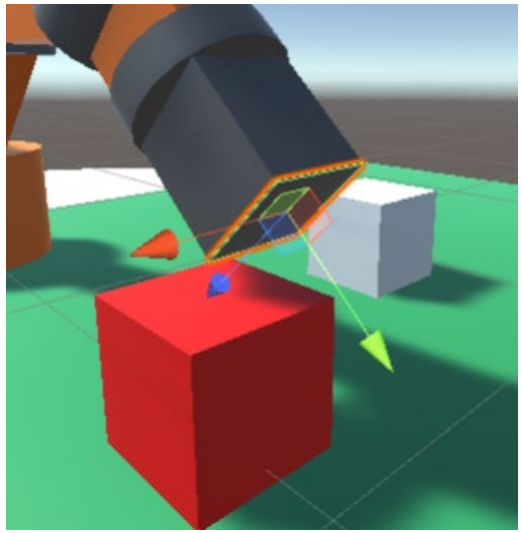
Control the robot arm to collide with the floor or goal and check the error message

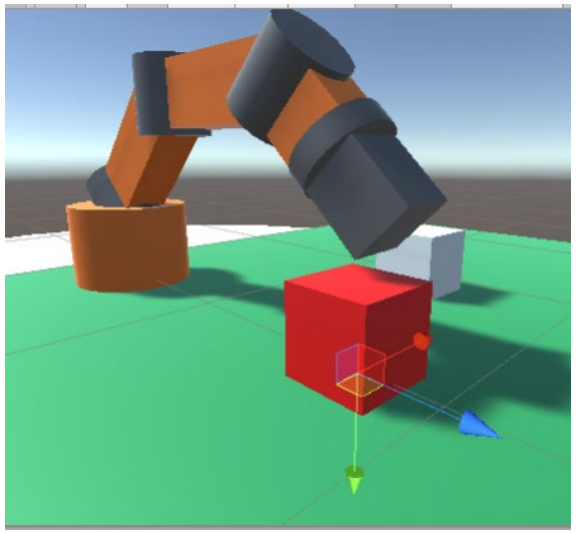


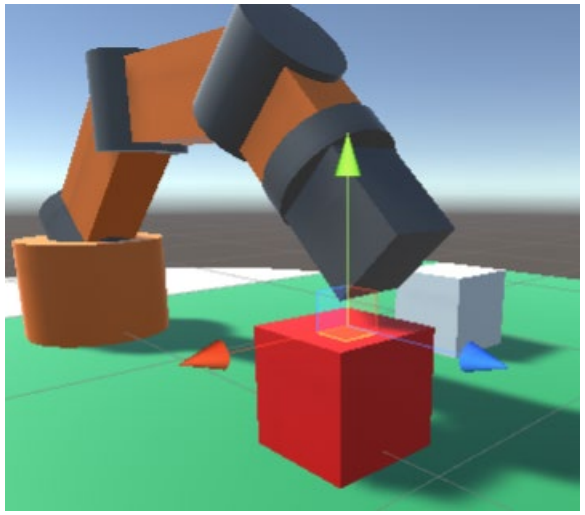
Goal reaching detections

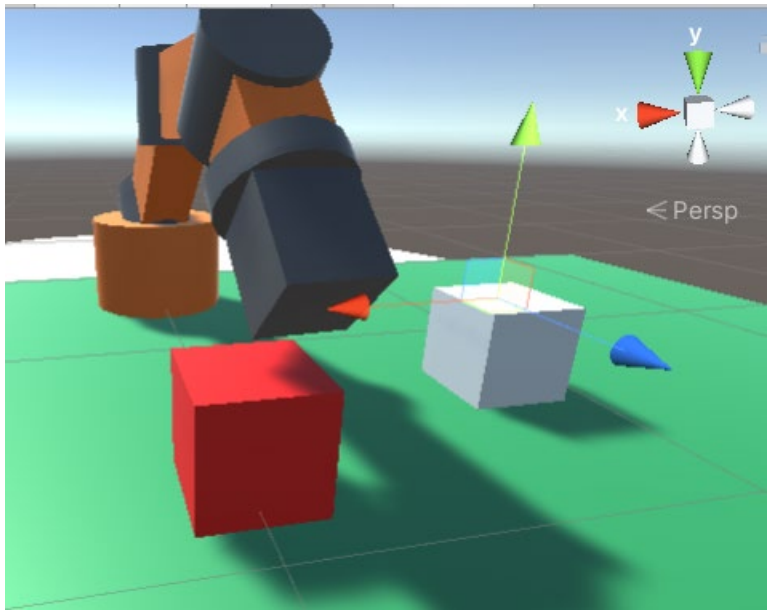
Points for goal-reaching detection







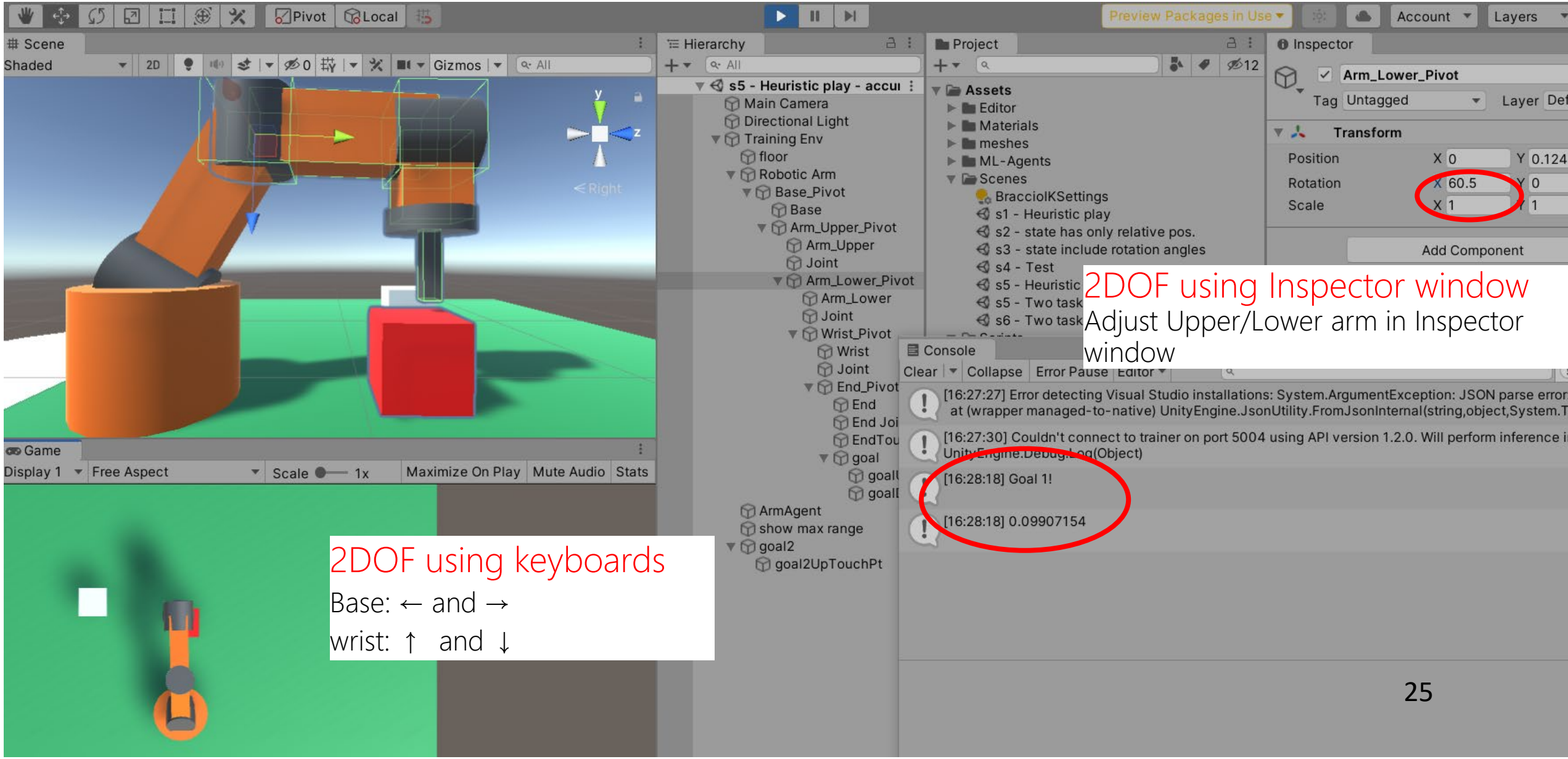




Use dx, dy and dz to determine point touch

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

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



2DOF using Inspector window

Adjust Upper/Lower arm in Inspector window

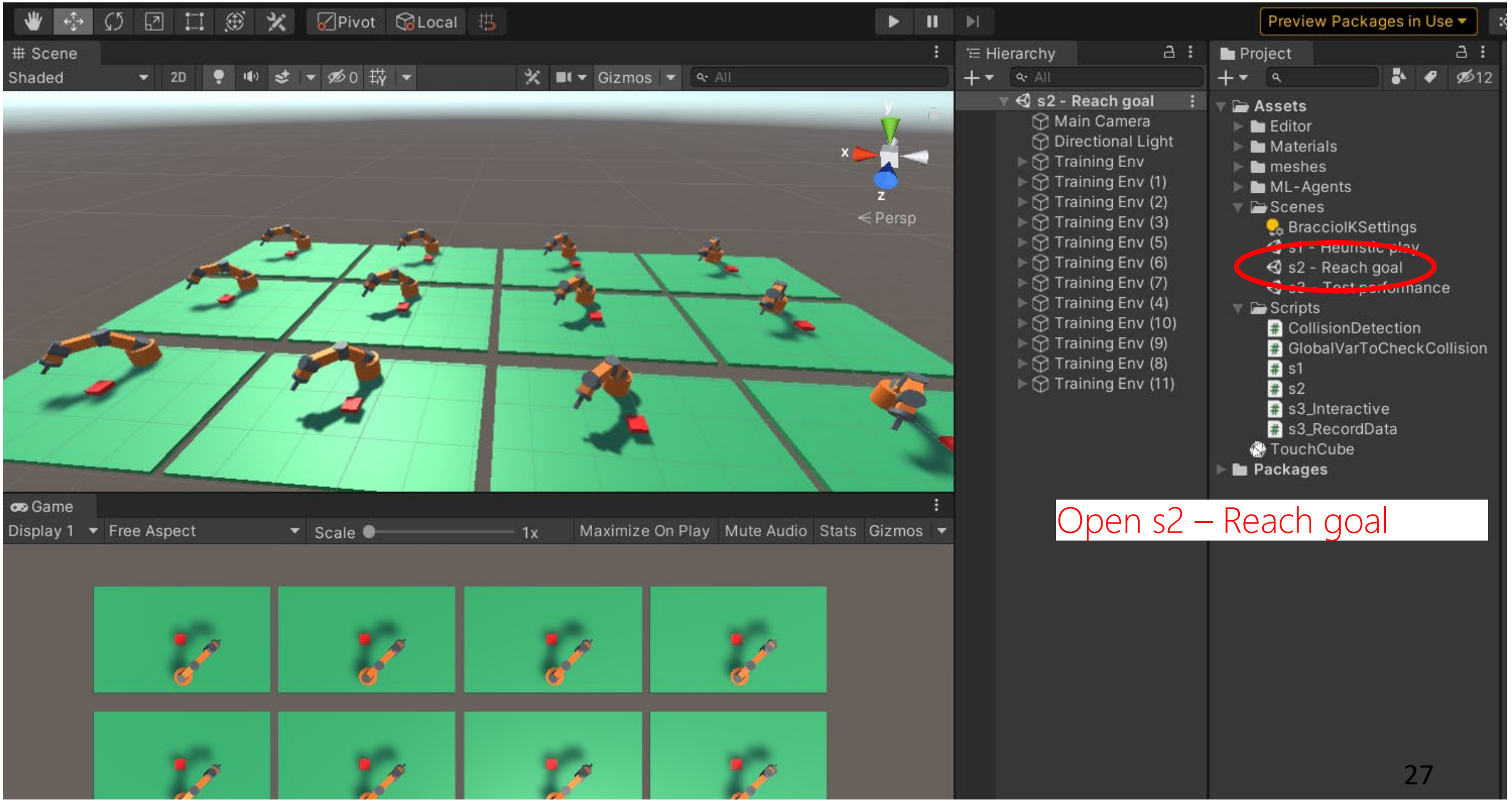
2DOF using keyboards

Base: ← and →

wrist: ↑ and ↓

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

4. Open the training environment



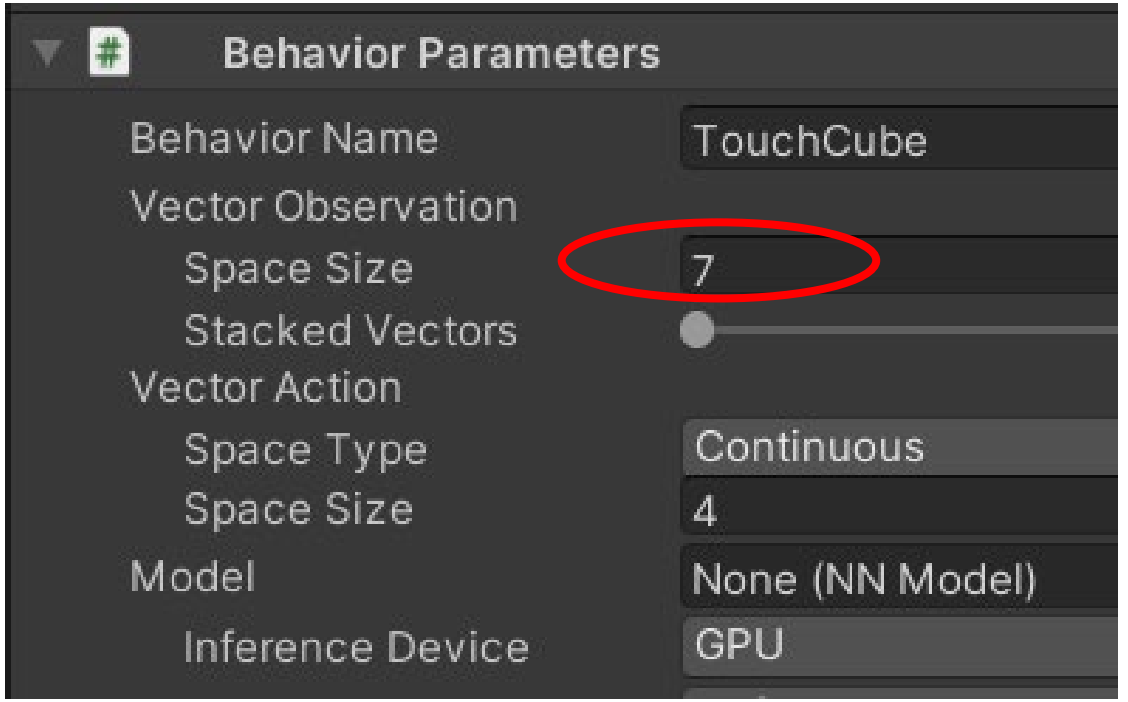
Open s2 – Reach goal

Use polar system to generate initial position

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

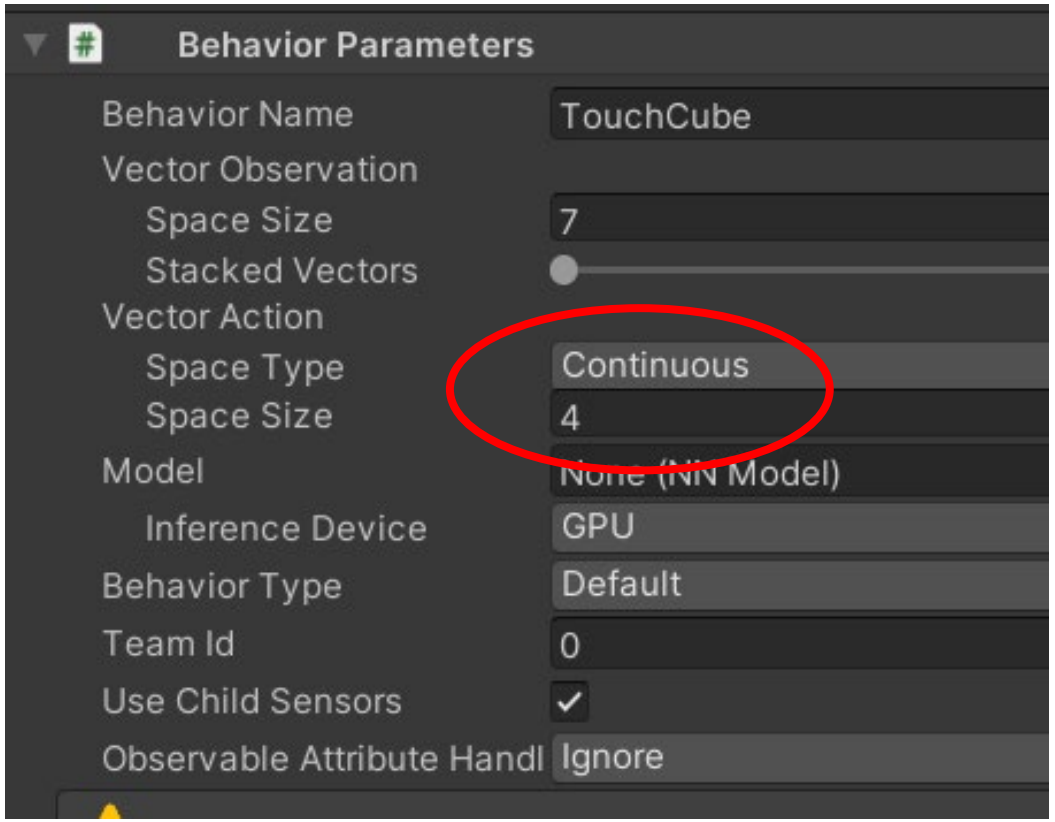
State has 7 variables

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

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

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



Action has 4 values

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



3 types of rewards

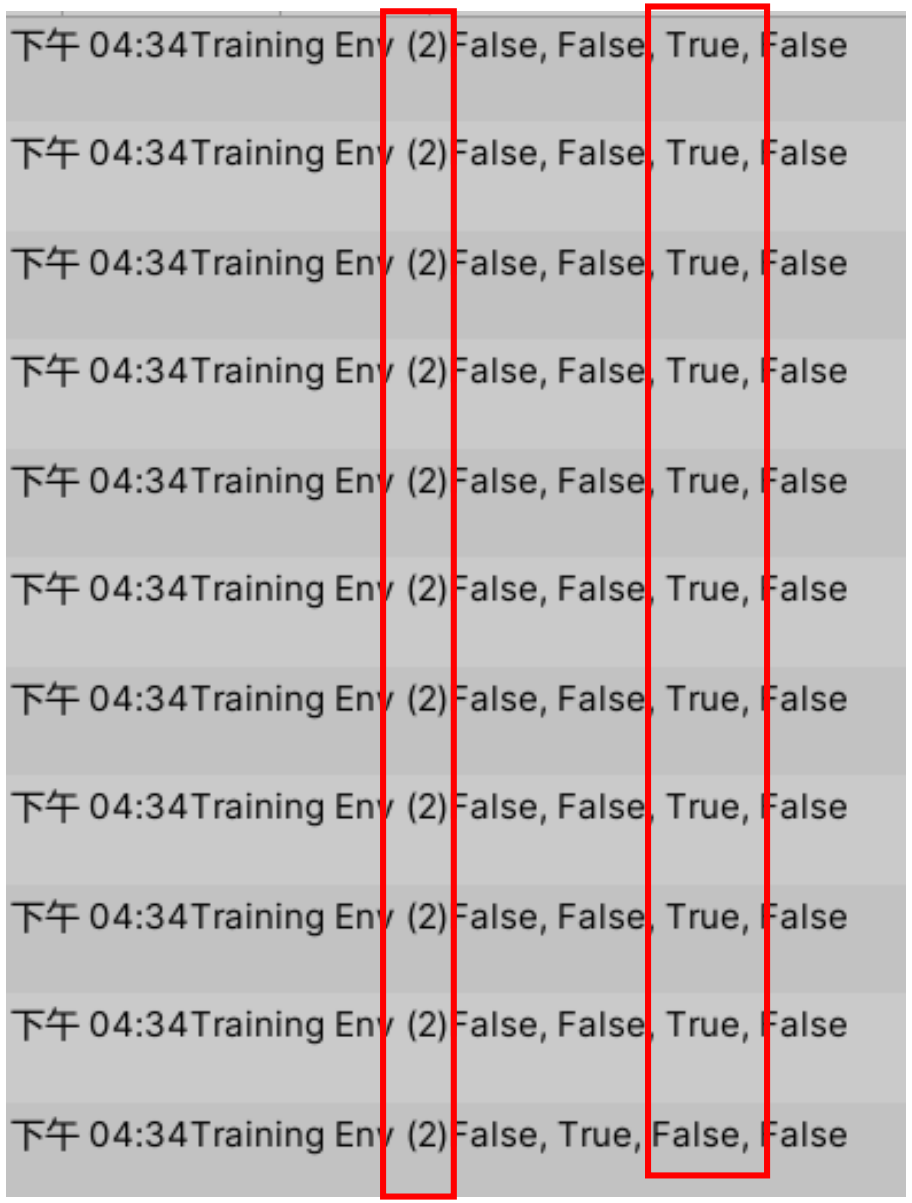
```
AddReward(-0.005f);

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

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

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

Training configuration file

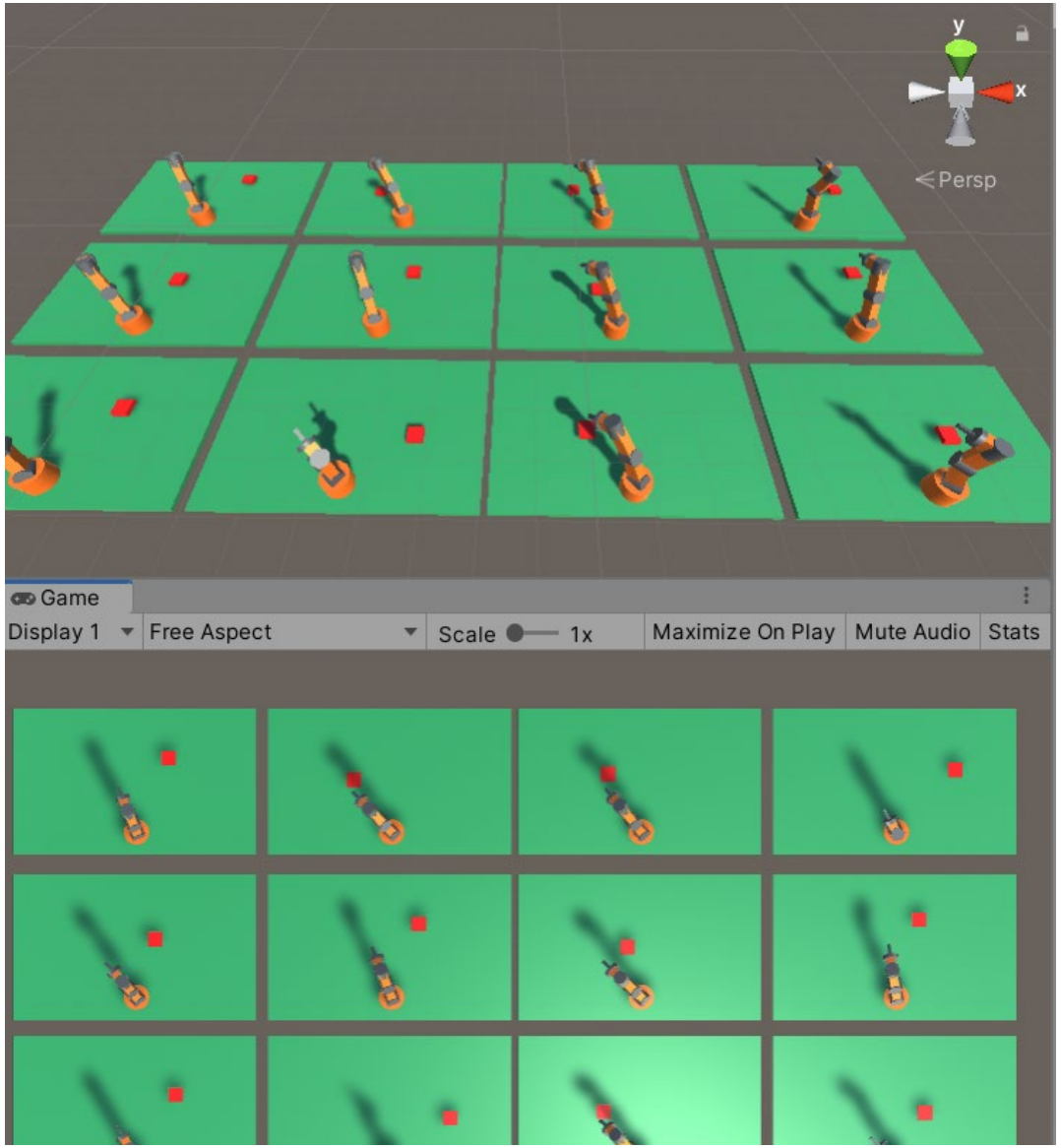
```
TouchCube:
  trainer_type: ppo
  hyperparameters:
    batch_size: 2048
    buffer_size: 20480
    learning_rate: 0.0003
    beta: 0.001
    epsilon: 0.2
    lambd: 0.95
    num_epoch: 3
    learning_rate_schedule
  network_settings:
    normalize: true
    hidden_units: 512
    num_layers: 3
   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

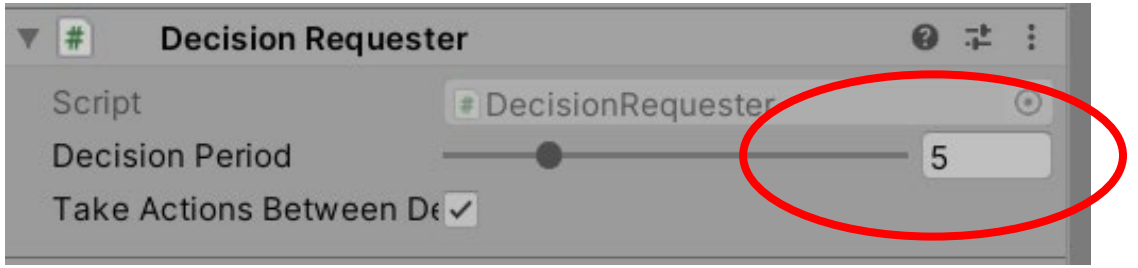
In one environment the red cube runs very fast. Why?

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

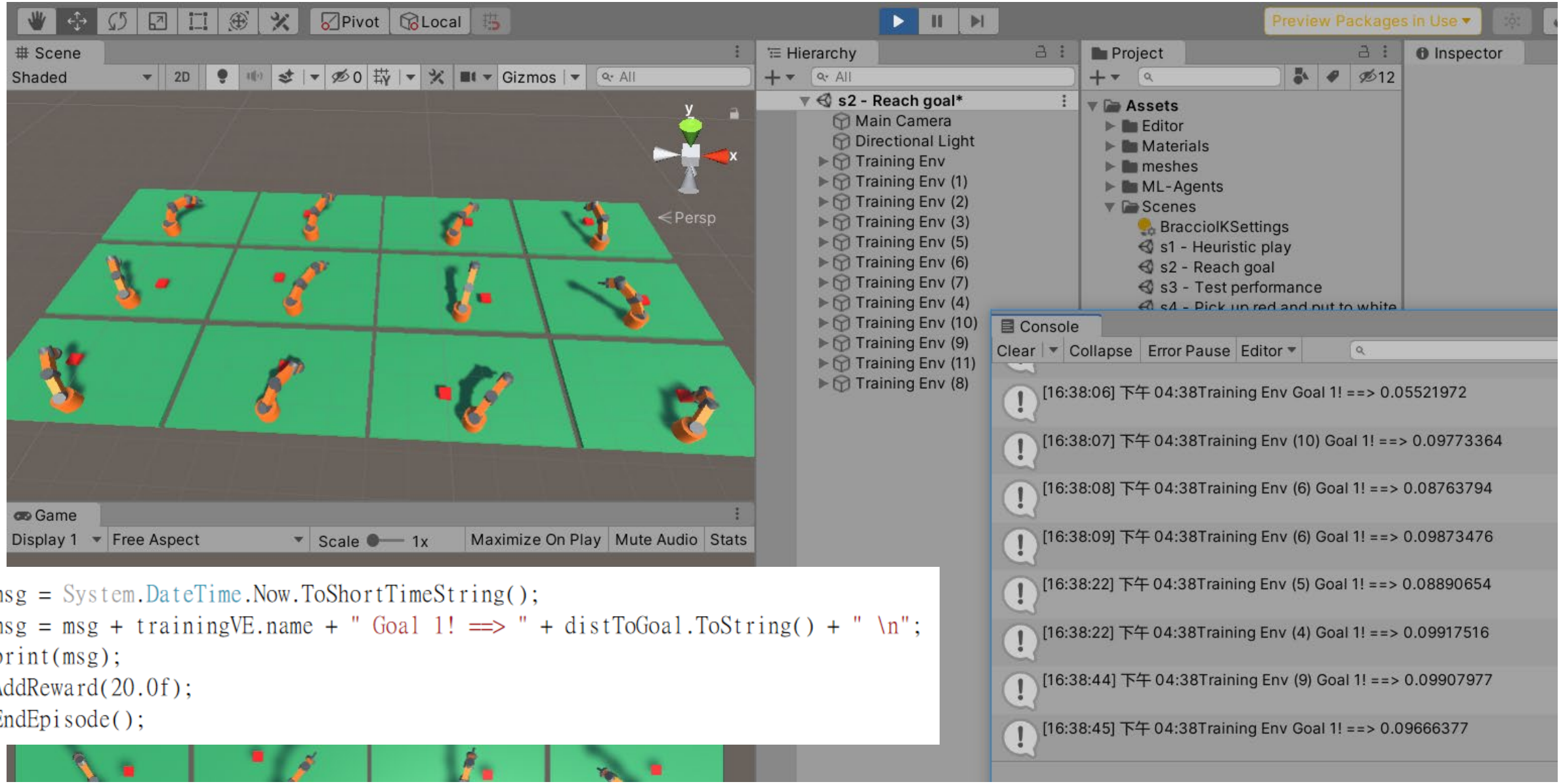


Use default decision period =5, decision period = 1 will result strange arm behavior



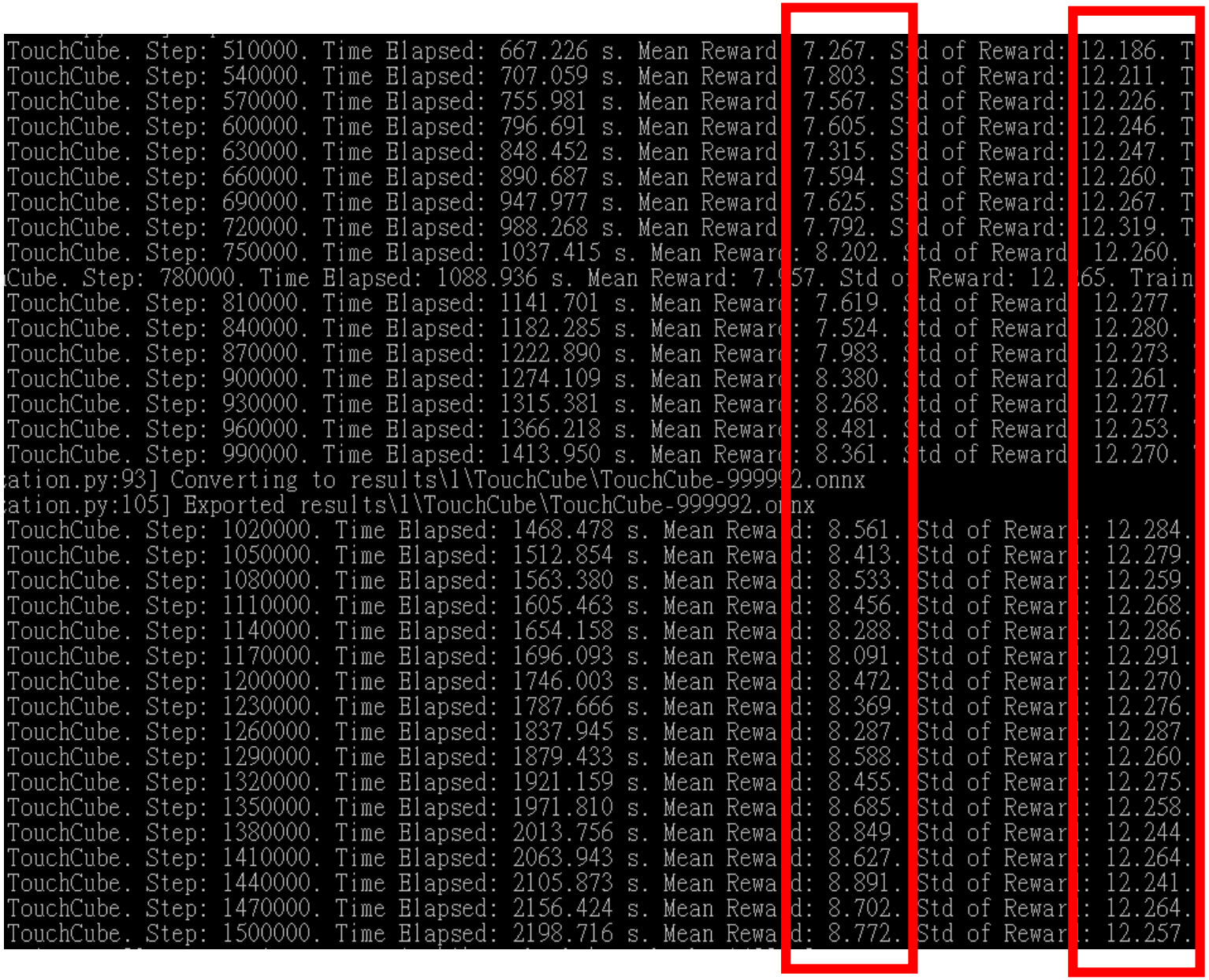


Print environment number that reaches goal



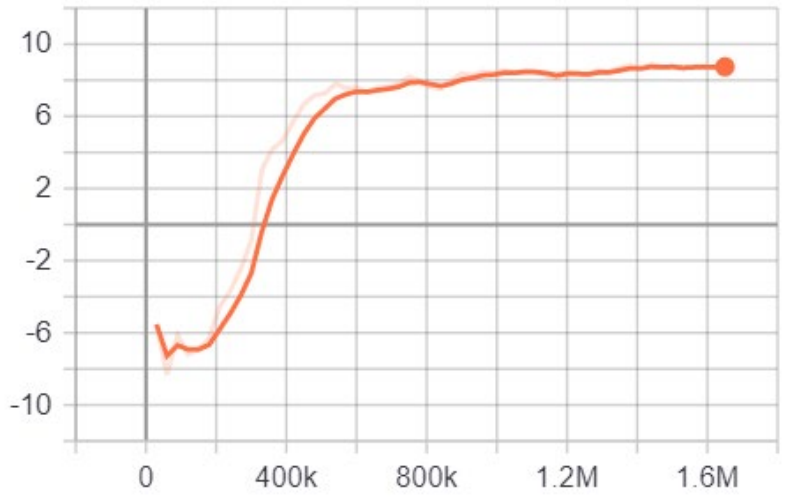

```
msg = System.DateTime.Now.ToShortTimeString();
msg = msg + trainingVE.name + " Goal 1! ==> " + distToGoal.ToString() + " \n";
print(msg);
AddReward(20.0f);
EndEpisode();
```

Results after 1.7M steps, looks promising

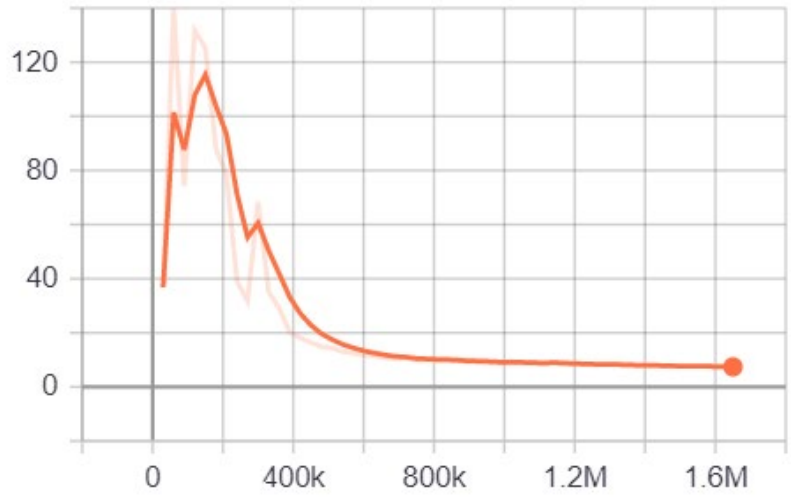


Results after 1.7M steps, looks promising

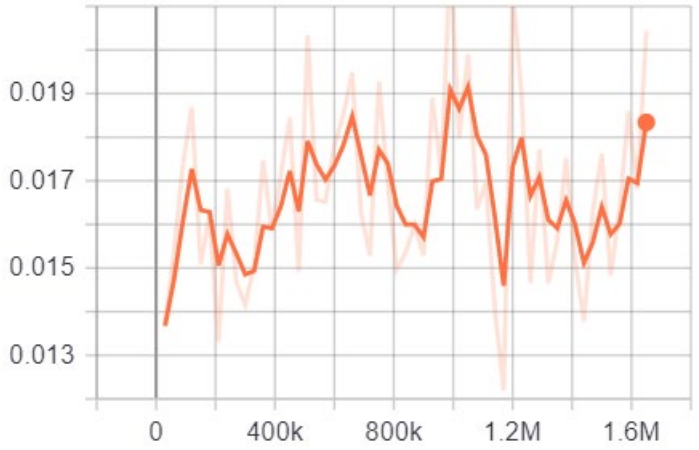
Cumulative Reward
tag: Environment/Cumulative Reward



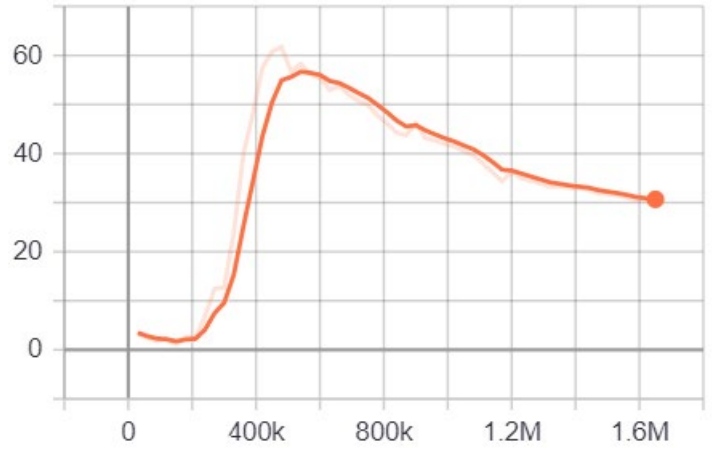
Episode Length
tag: Environment/Episode Length



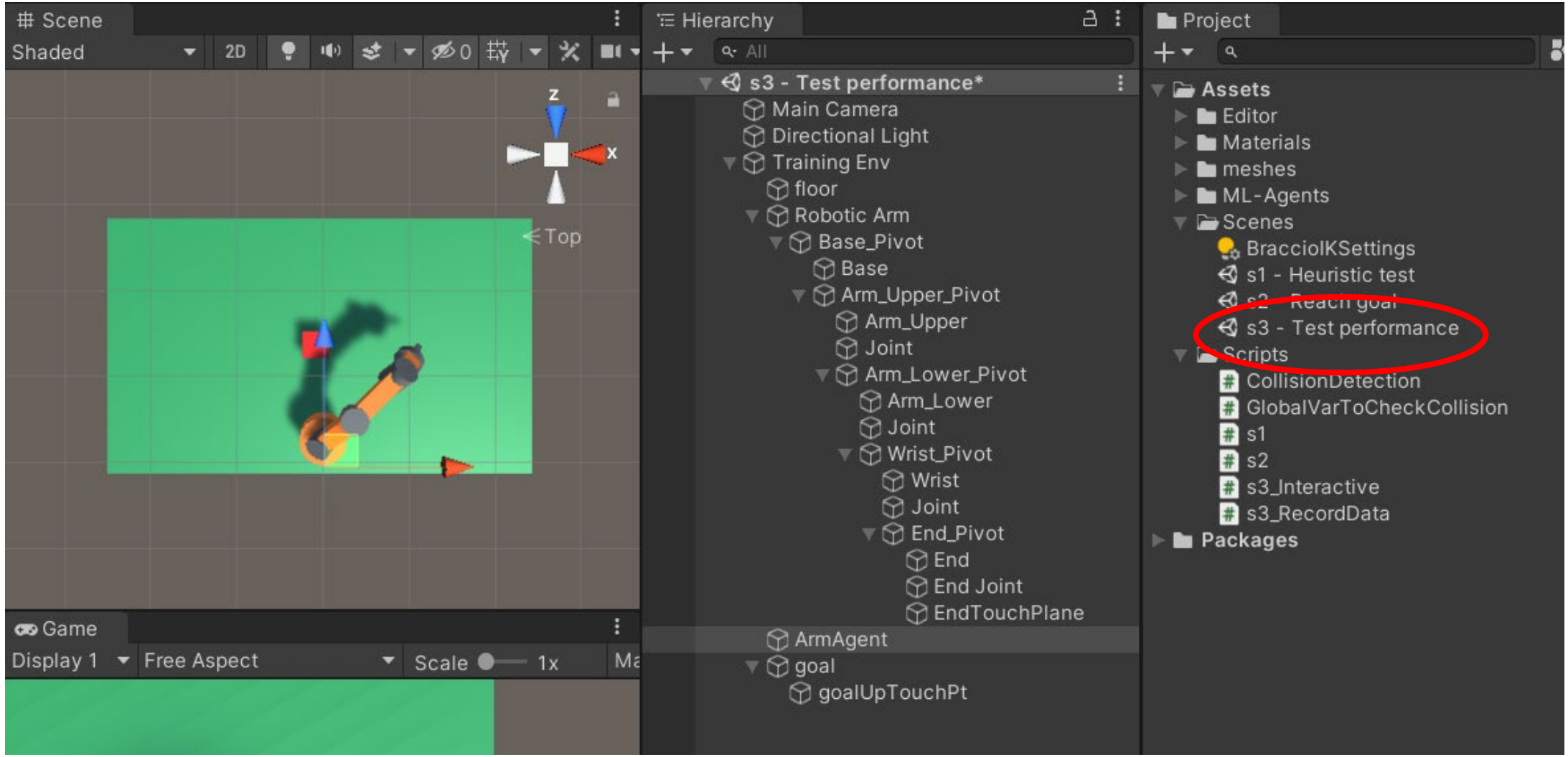
Policy Loss
tag: Losses/Policy Loss



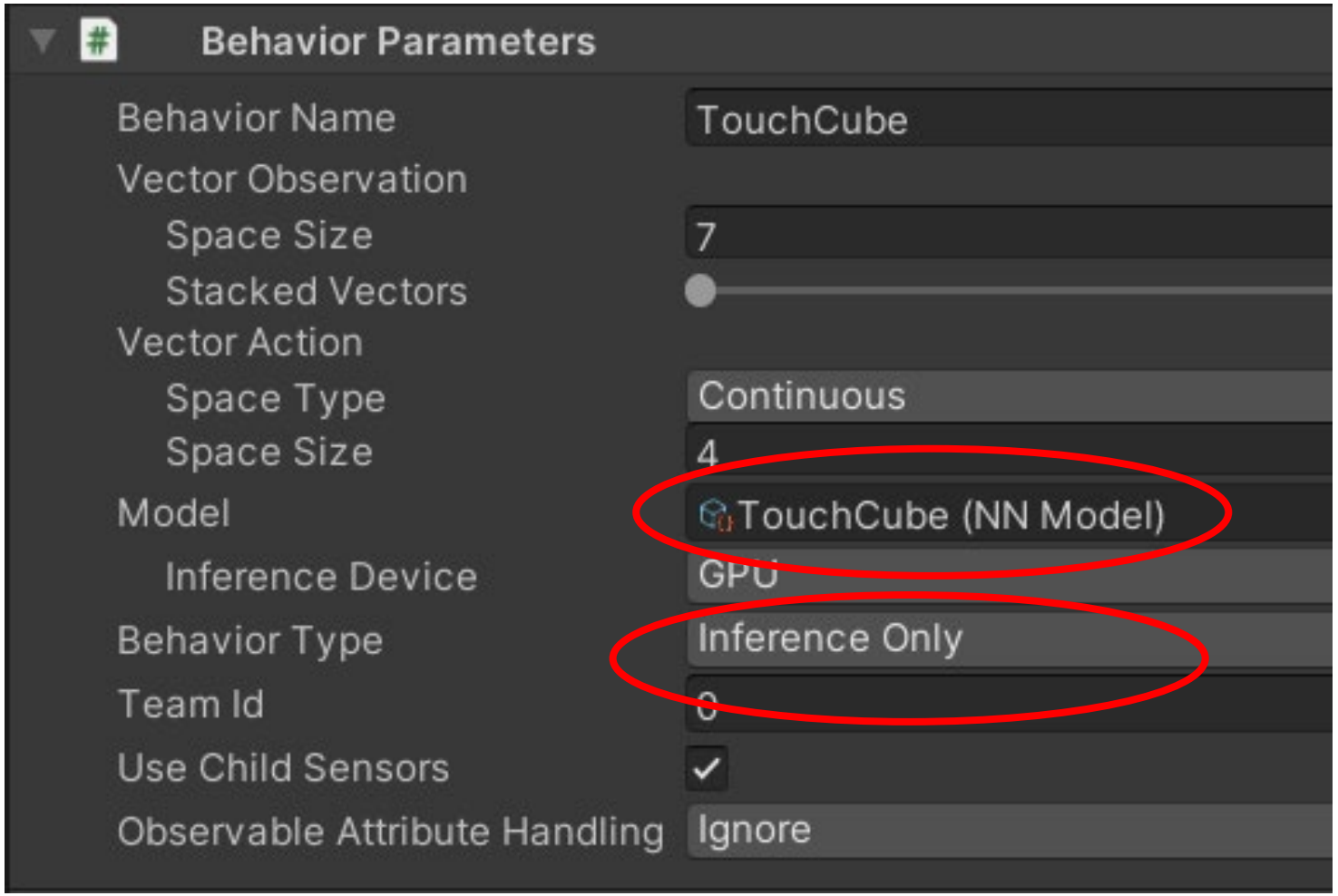
Value Loss
tag: Losses/Value Loss



5. Test performance

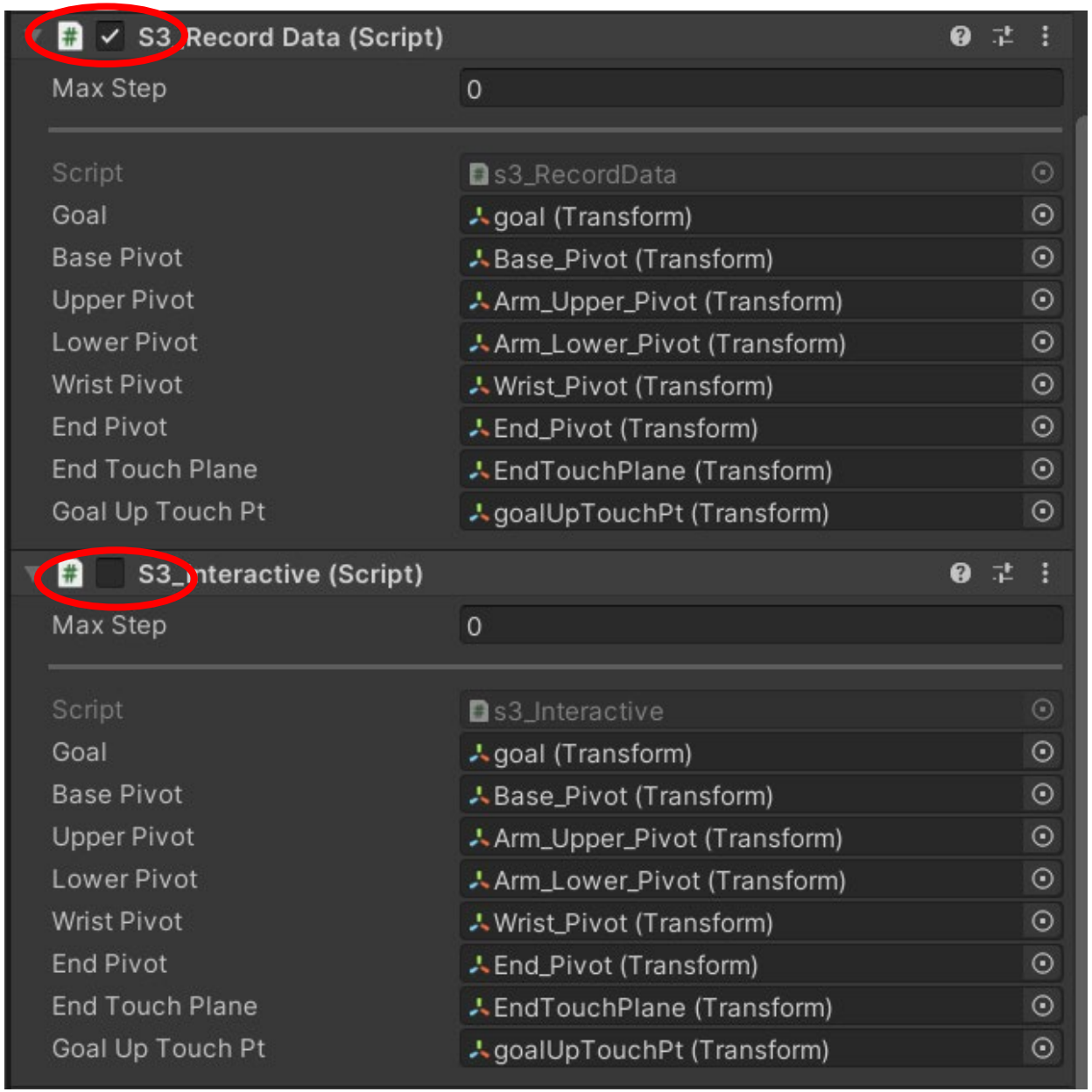


Assign trained NN



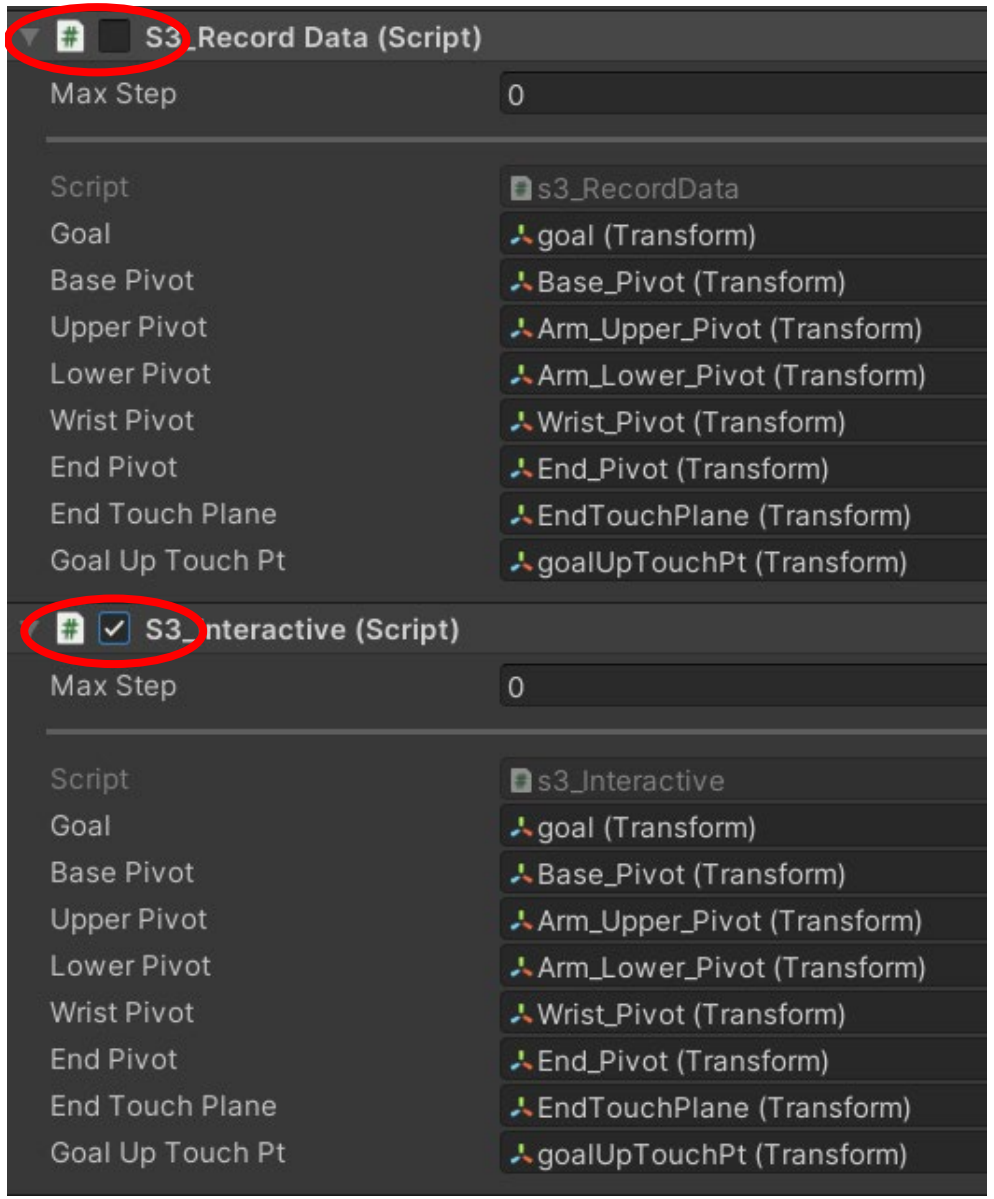
2 types of tests – (1) data recording

Uncheck interactive test



2 types of tests – (2) interactive test

Uncheck data-recording test



Practice

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

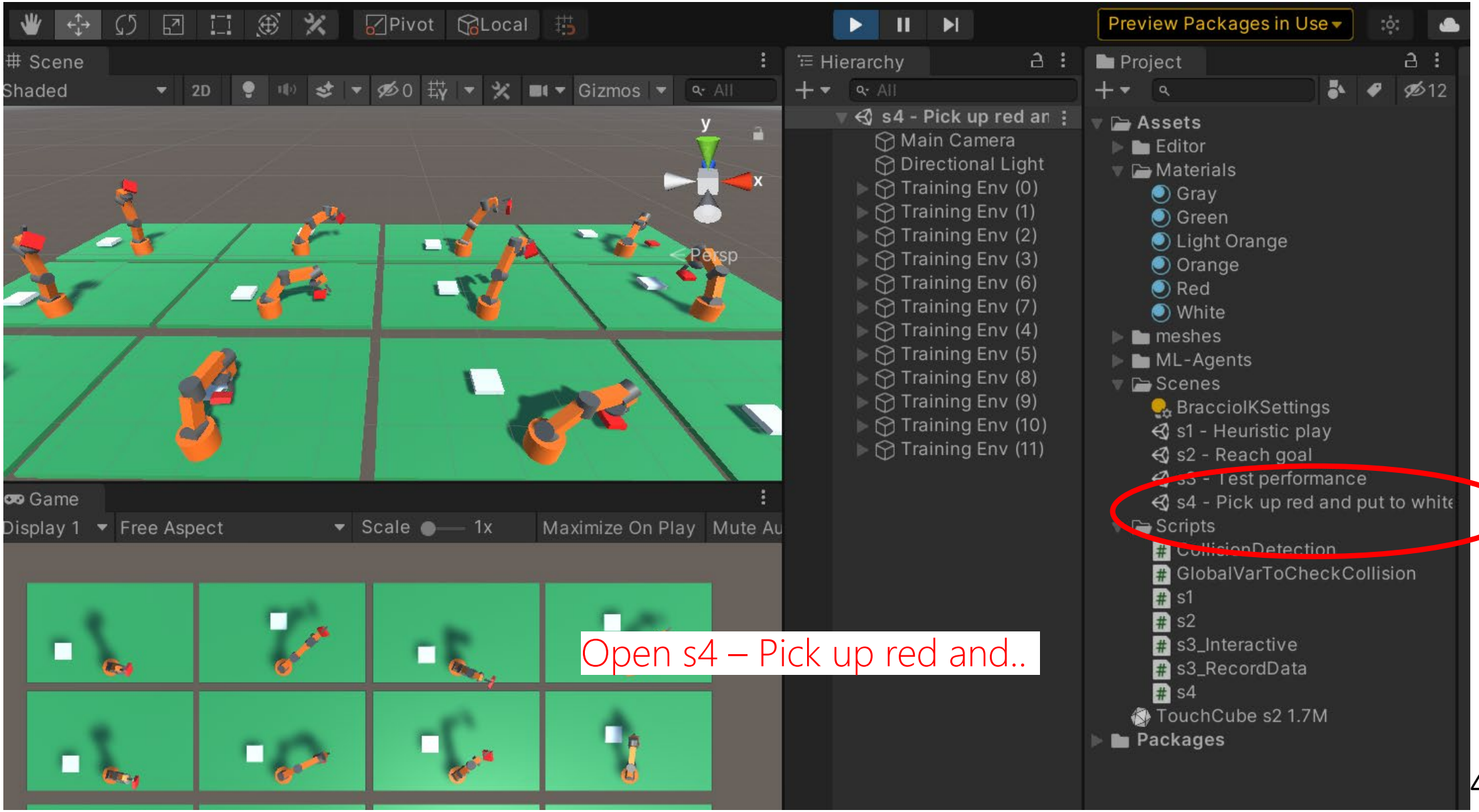
Base pivot (0, 45, 0)
Upper pivot (45, 0, 0)
Lower pivot (45, 0, 0)
Wrist pivot (45, 0, 0)

Base pivot (0, 0, 0)
Upper pivot (0, 0, 0)
Lower pivot (0, 0, 0)
Wrist pivot (0, 0, 0)

Base pivot (0, y, 0)
Upper pivot (x1, 0, 0)
Lower pivot (x2, 0, 0)
Wrist pivot (x3, 0, 0)

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

4. Open the training environment



Use polar system to generate initial position

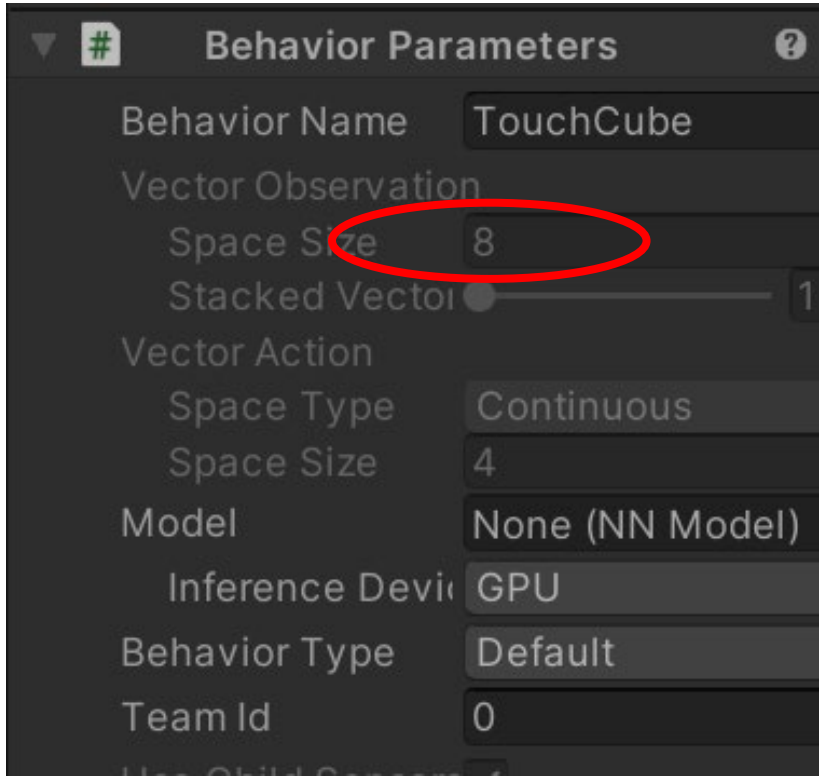
```
//use polar coordinate to calculate x, z to place goal1
float radius = UnityEngine.Random.Range(0.8f, 1.5f);
float theta = (UnityEngine.Random.Range(5.0f, 80.0f) / 180.0f) * Mathf.I
float x = radius * Mathf.Sin(theta);
float z = radius * Mathf.Cos(theta);
goal.transform.localPosition = new Vector3(x,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

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

red cube is generated at right side

white cube is generated at left side

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 * (3 - stage)); // punish more for sta

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

//if rotation angle is out of range or collision happens
if (!Rotation_in_range() || MyGlobalVar.LowerArmCollisic
 MyGlobalVar.EndCollisionHappens || MyGlobalVar.goalCol
{
    AddReward(-5.0f);
    EndEpisode();
}
```

4 types of rewards

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

Training configuration file

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