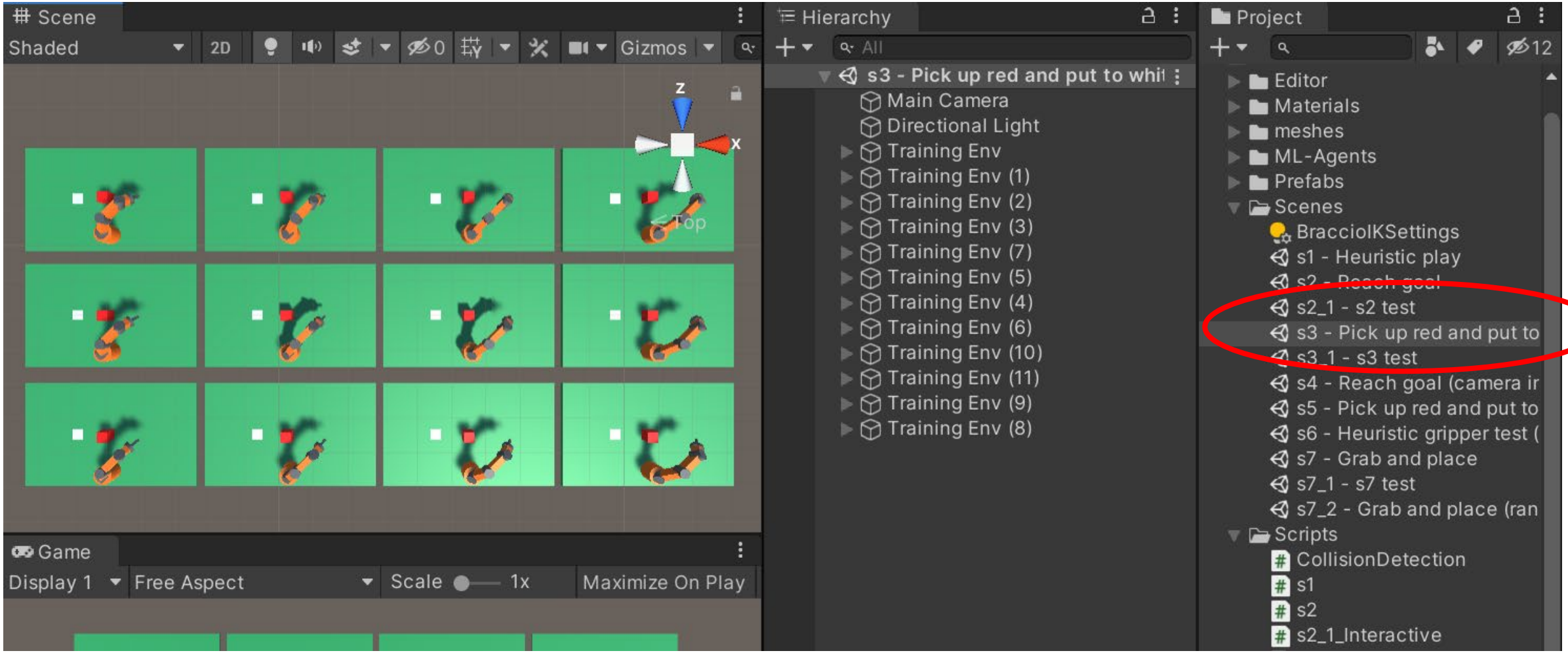
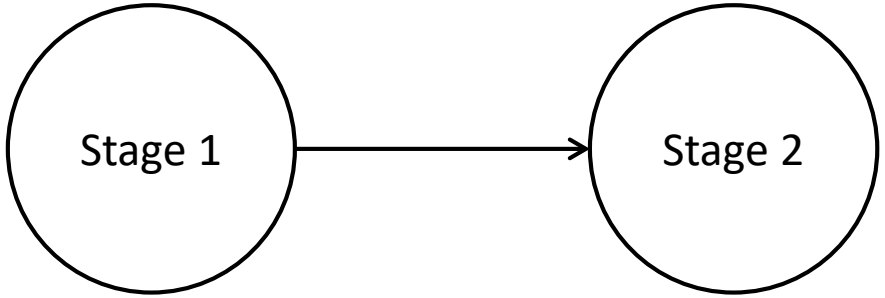
(2) Pick up goal and place it to white area using Δd information

Open "s3 – Pick up red and put to white"



Training setting



$s = (sNo, \Delta_1 x, \Delta_1 y, \Delta_1 z, \theta_{\mathrm{B}}, \theta_{\mathrm{U}}, \theta_{\mathrm{L}}, \theta_{\mathrm{W}})$ $s = (sNo, \Delta_2 x, \Delta_2 y, \Delta_2 z, \theta_{\mathrm{B}}, \theta_{\mathrm{U}}, \theta_{\mathrm{L}}, \theta_{\mathrm{W}})$

$$r = \begin{cases} -0.005 \\ -5 \\ +20 \quad d_1 \leq 0.5 \end{cases} \qquad r = \begin{cases} -0.005 & \text{per step} \\ -5 & \text{collision, out of range} \\ +20 & \text{goal,} d_2 \leq 0.5 \end{cases}$$

NN: 8-512-512-512-4

Time horizon = 2000

Buffer size = 20480

Batch size = 2048

$a = (\Delta\theta_B, \Delta\theta_U, \Delta\theta_L, \Delta\theta_W)$

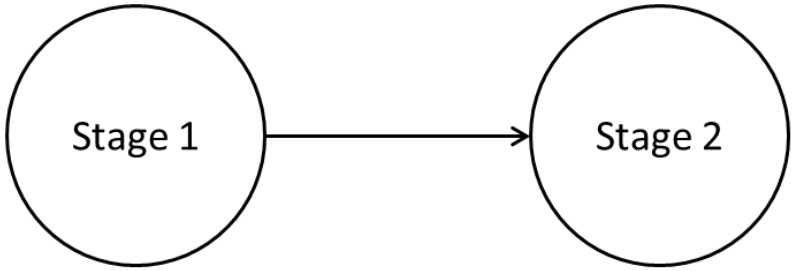
No. of training environment = 9

Goal initialize = randomly positioned in polar system θ=-80~80, r=0.8~1.5

Goal2 initialize = same as goal 1

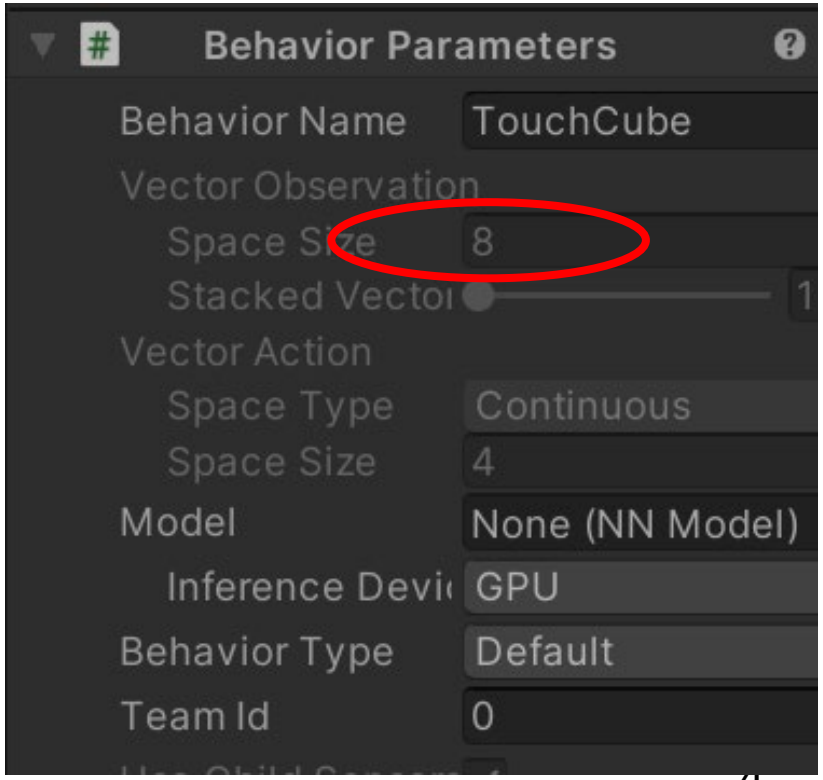
Arm initialize: $(\theta_B = 0, \theta_U = 45, \theta_L = 45, \theta_W = 45)$

State

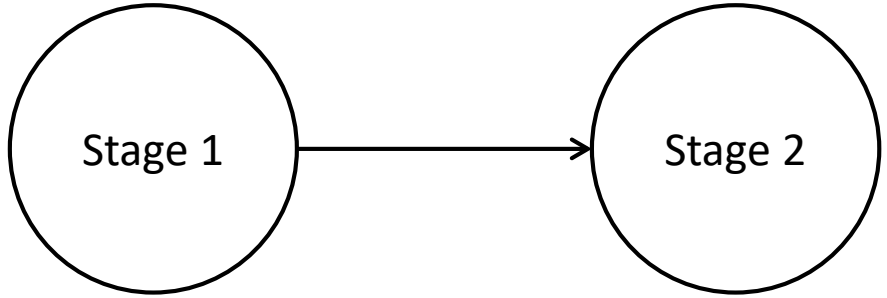


$s = (sNo, \Delta_1 x, \Delta_1 y, \Delta_1 z, \theta_B, \theta_U, \theta_L, \theta_W)$ $s = (sNo, \Delta_2 x, \Delta_2 y, \Delta_2 z, \theta_B, \theta_U, \theta_L, \theta_W)$

```
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);
```



Rewards

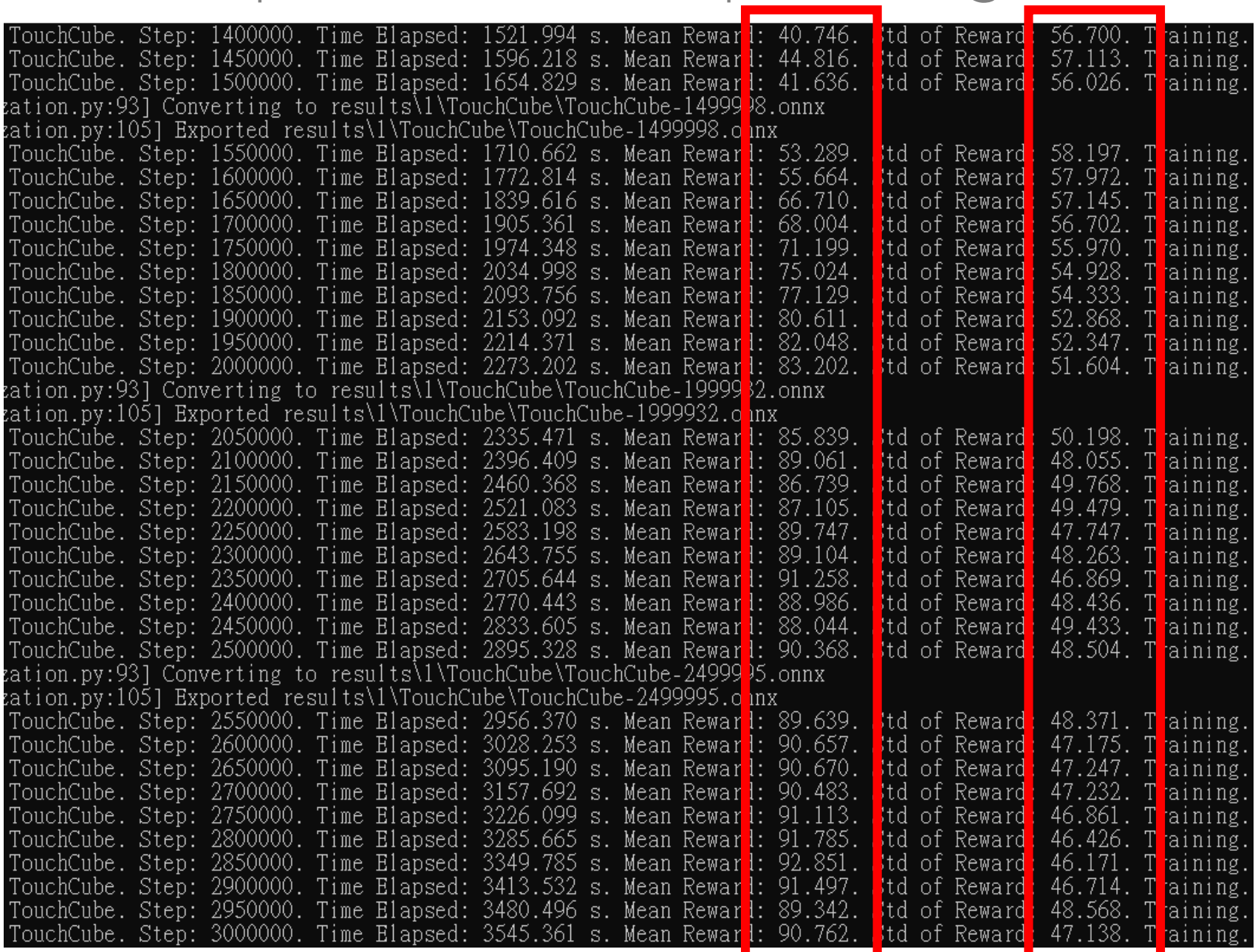


$$r = \begin{cases} -0.005 & \\ -5 & \\ +20 & d_1 \leq 0.5 \end{cases} \qquad r = \begin{cases} -0.005 & \text{per step} \\ -5 & \text{collision, out of range} \\ +20 & \text{goal}, d_2 \leq 0.5 \end{cases}$$

```
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
}
```

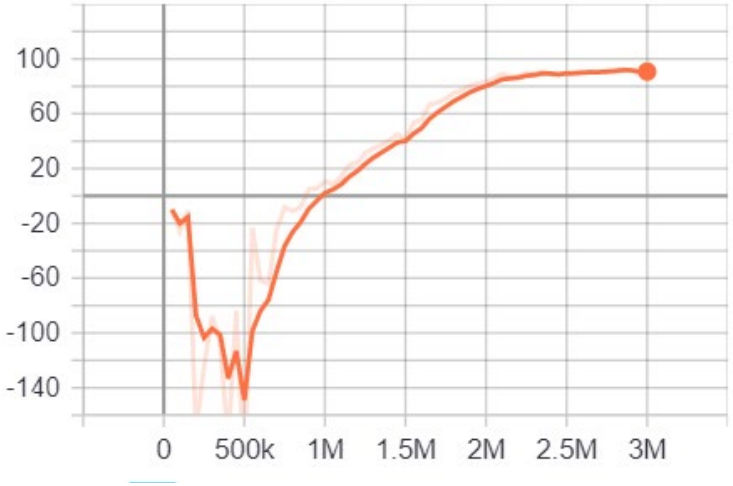
```
else if (PointTouch(goalDownTouchPt, goal2UpTouchPt, 0.3f))
{
    msg = trainingVE.name + " Goal 2! \n";
    Debug.Log(msg);
    AddReward(100.0f);
    EndEpisode();
}
```

I quit at 3M, looks promising

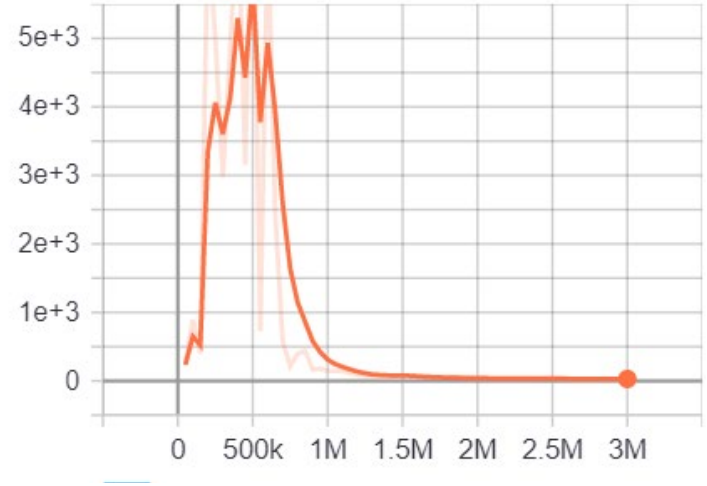


I quit at 3M, looks promising

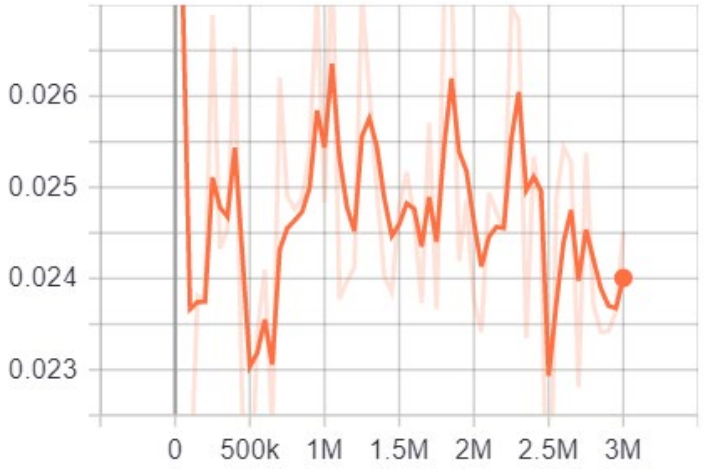
Cumulative Reward
tag: Environment/Cumulative Reward



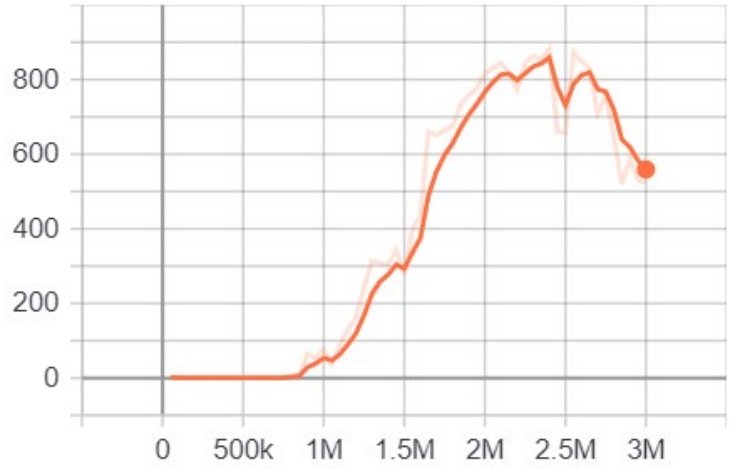
Episode Length
tag: Environment/Episode Length



Policy Loss
tag: Losses/Policy Loss



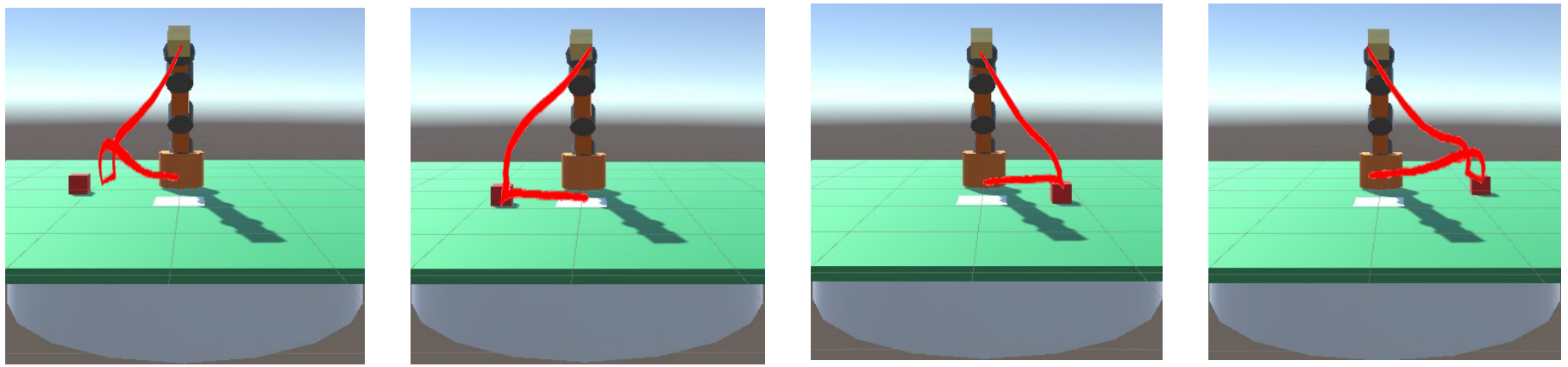
Value Loss
tag: Losses/Value Loss



Test 1 – Performance data

$s = (\Delta x, \Delta y, \Delta z)$

NN: 3-512-512-512-4



	rewards		**steps**	
	mean	std	mean	std
80 degree	39.342	0.005	131.7	0.943
45 degree	39.609	0.006	78.26	1.11
-45 degree	39.618	0.006	76.43	1.16
-80 degree	39.337	0.005	132.64	1.054

Test 2 – Interact with dynamic, adversial environment

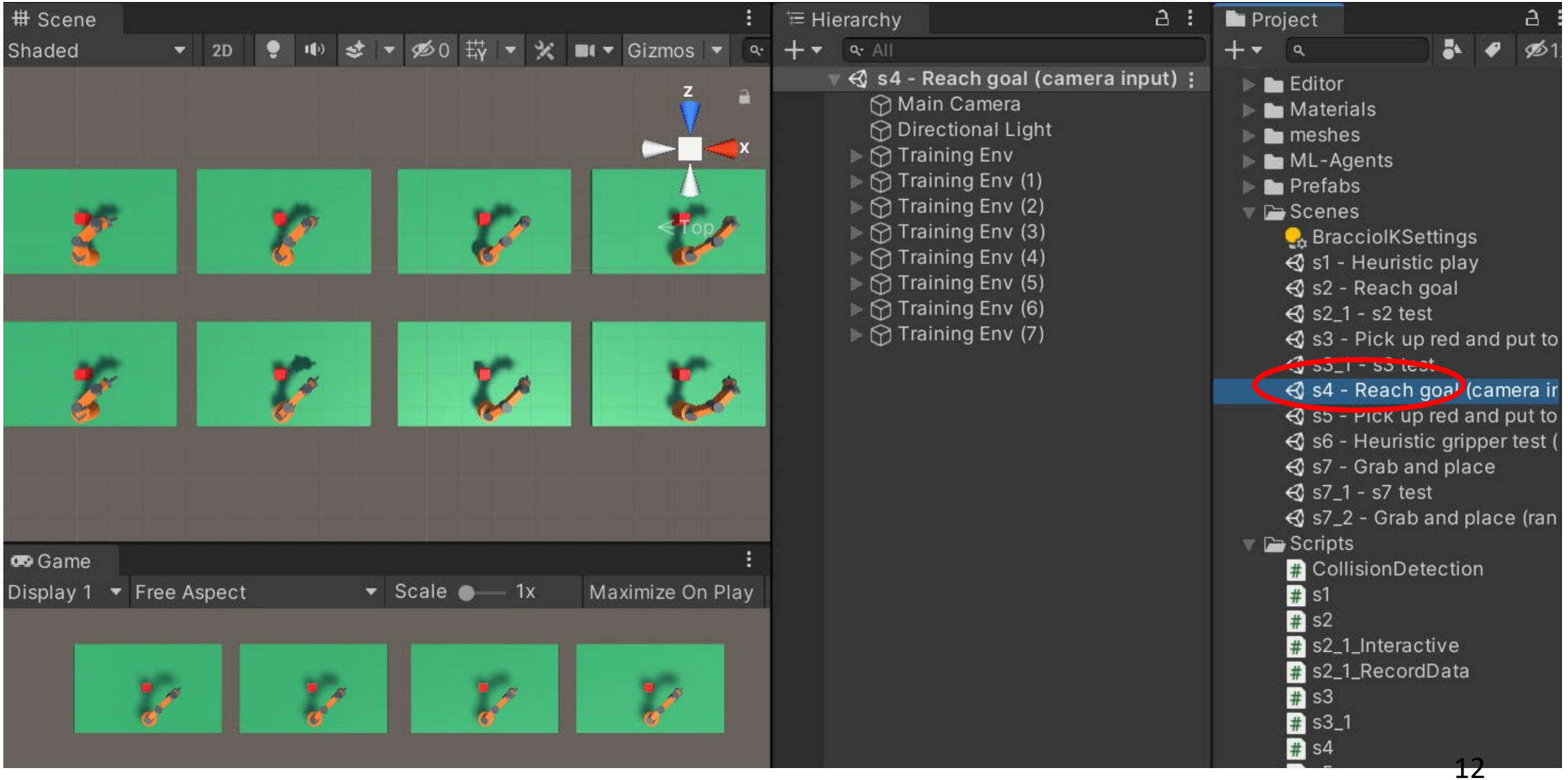
Video showing users interacts and plays with the intelligent robot arm

HW4(2)

- Describe training setting
- Show tensor board plots and discuss your training performance
- Describe test performance (recorded data, interactive test)

(3) Reach goal using camera image information

Open s4


Scene
Shaded
2D
Gizmos
Top
Game
Display 1
Free Aspect
Scale
1x
Maximize On Play
Hierarchy
All
s4 - Reach goal (camera input)
Main Camera
Directional Light
Training Env
Training Env (1)
Training Env (2)
Training Env (3)
Training Env (4)
Training Env (5)
Training Env (6)
Training Env (7)
Project
Editor
Materials
meshes
ML-Agents
Prefabs
Scenes
BraccioIKSettings
s1 - Heuristic play
s2 - Reach goal
s2_1 - s2 test
s3 - Pick up red and put to
s4 - Reach goal (camera ir
s6 - Heuristic gripper test (
s7 - Grab and place
s7_1 - s7 test
s7_2 - Grab and place (ran
Scripts
CollisionDetection
s1
s2
s2_1_Interactive
s2_1_RecordData
s3
s3_1
s4

Training setting

s = feature map vector from a CNN, size = 2592

Input image to the CNN is captured by a camera from top, size = 84x84x3

$$a = (\Delta\theta_B, \Delta\theta_U, \Delta\theta_L, \Delta\theta_W)$$

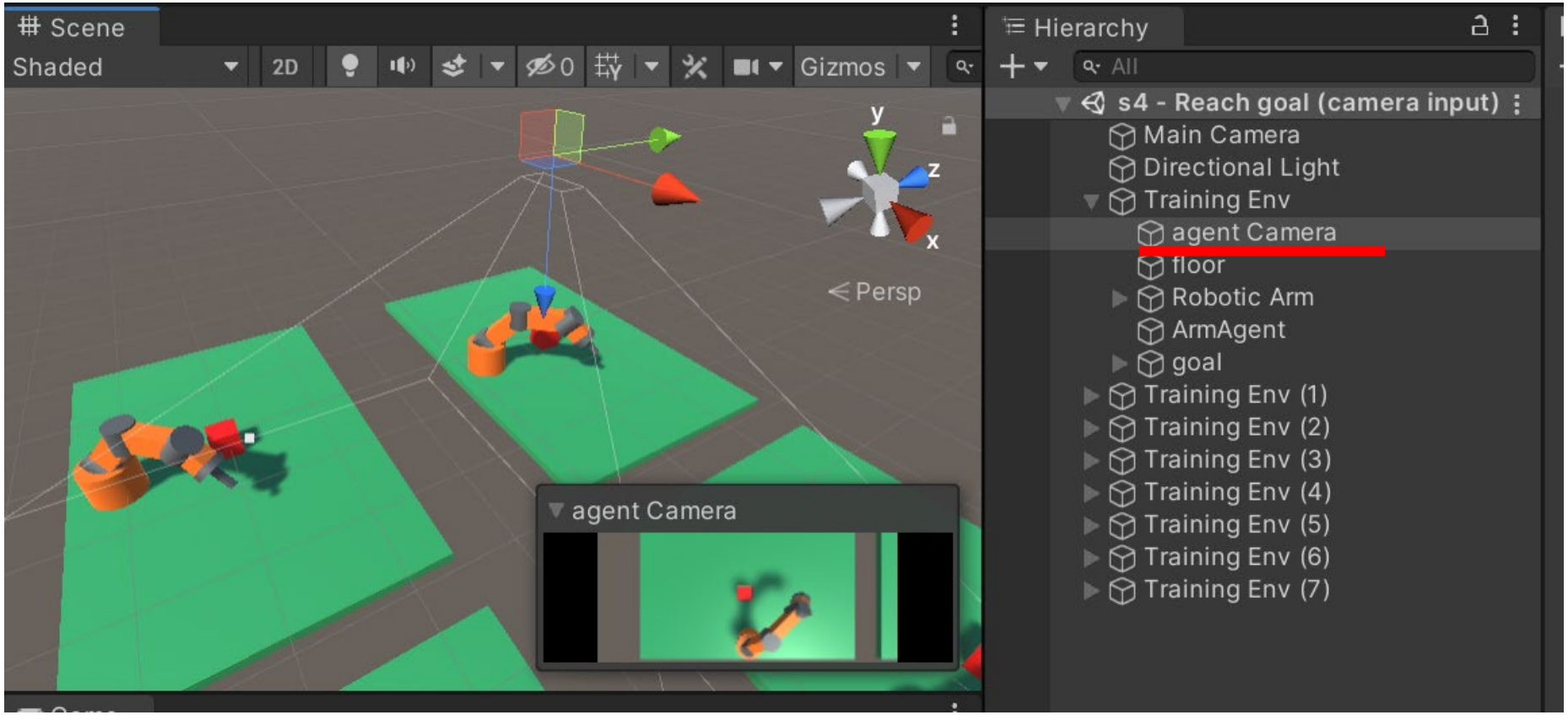
$$r = \begin{cases} -0.005 & \text{per step} \\ -5 & \text{collision, out of range} \\ +20 & \text{goal, } d \leq 0.5 \end{cases}$$

No. of training environment = 9

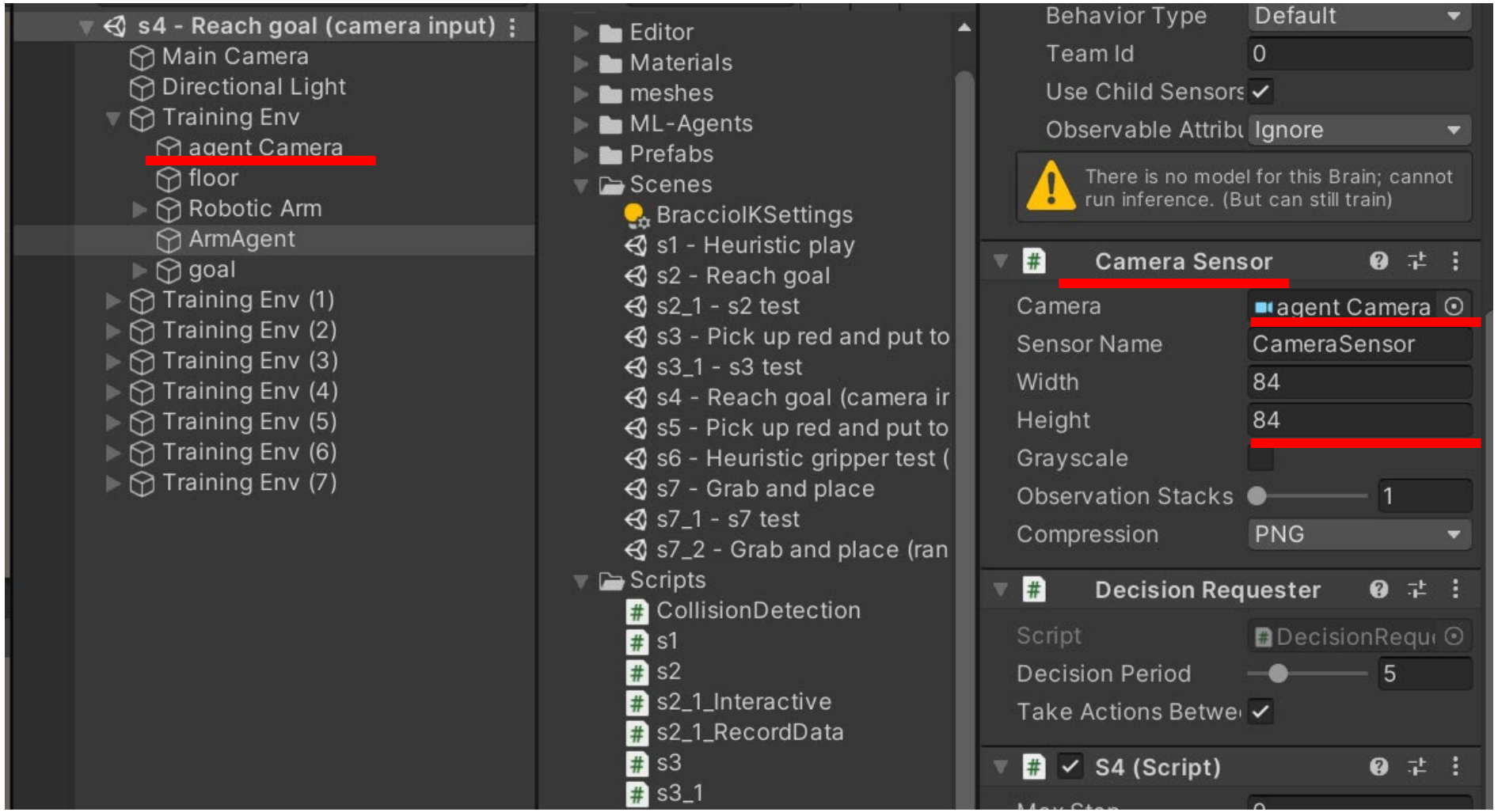
Goal initialize = randomly positioned in polar system θ=-80~80, r=0.8~1.5

Arm initialize: $(\theta_b = 0, \theta_u = 45, \theta_l = 45, \theta_w = 45)$

Add camera sensor to the robot agent

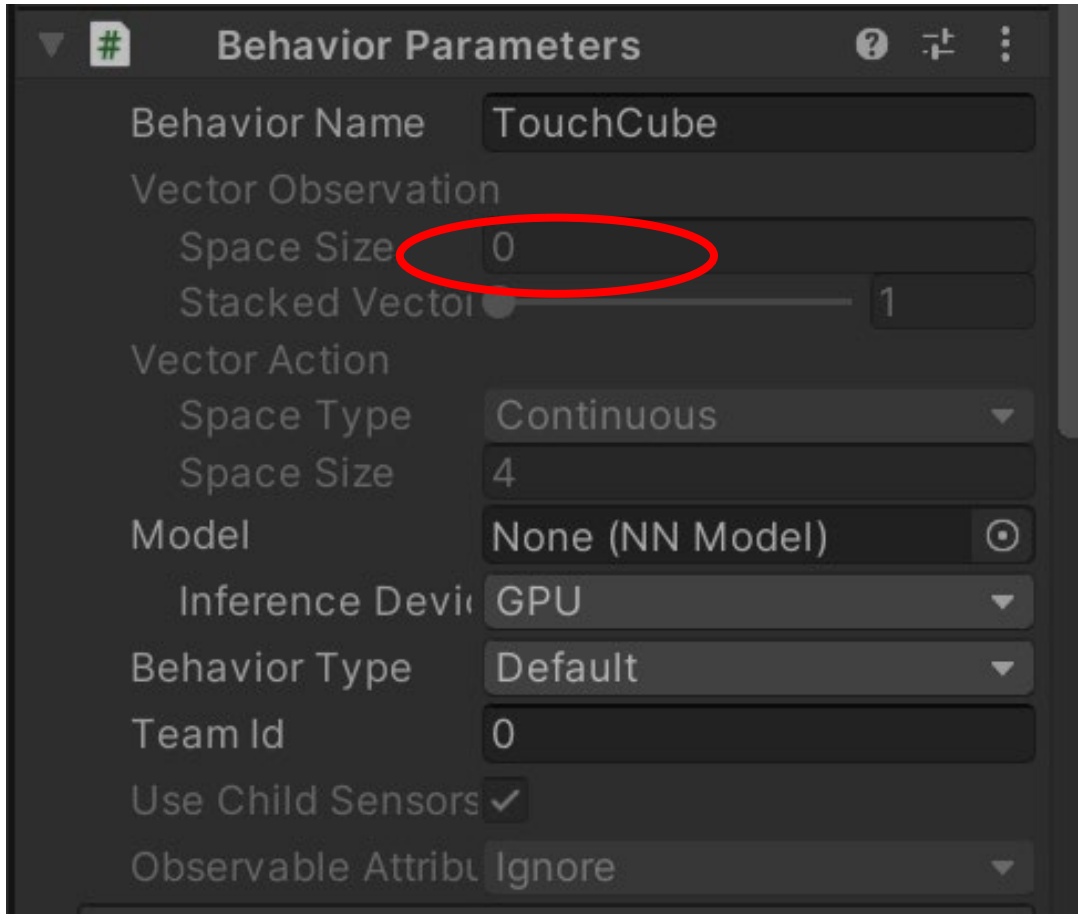


Add camera sensor to the robot agent



Vector observation = 0

```
public override void CollectObservations(VectorSensor sensor)
{

}
```

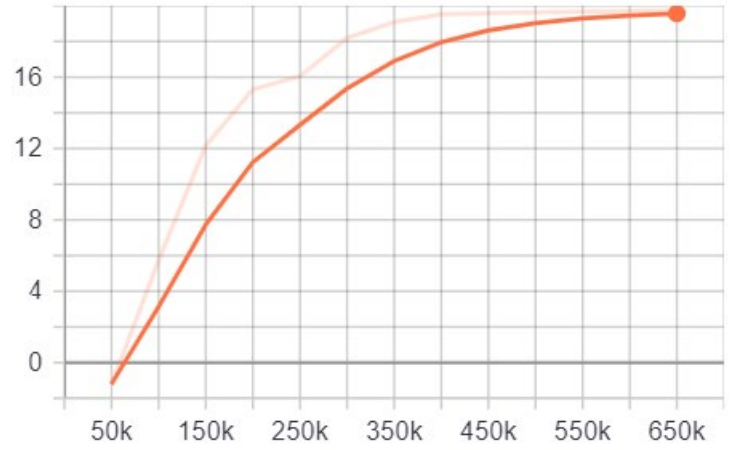


Looks good shortly (600K only)

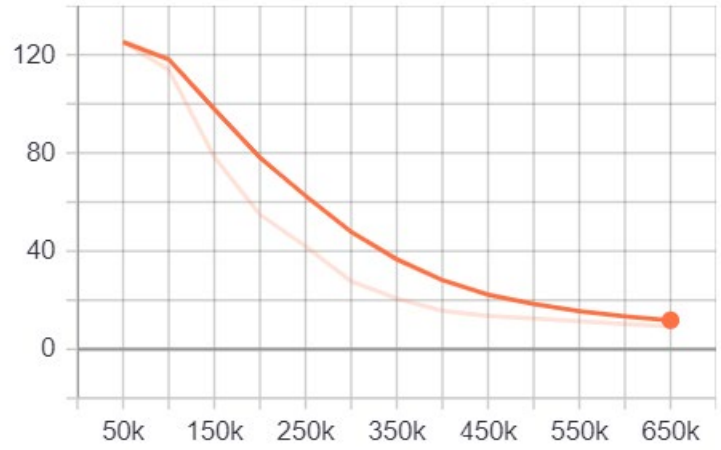
```
TouchCube. Step: 50000. Time Elapsed: 222.493 s. Mean Reward: -1.205. Std of Reward: 12.602. Training.
TouchCube. Step: 100000. Time Elapsed: 417.921 s. Mean Reward: 5.743. Std of Reward: 13.921. Training.
TouchCube. Step: 150000. Time Elapsed: 618.958 s. Mean Reward: 12.166. Std of Reward: 11.773. Training.
TouchCube. Step: 200000. Time Elapsed: 819.051 s. Mean Reward: 15.312. Std of Reward: 9.549. Training.
TouchCube. Step: 250000. Time Elapsed: 1027.214 s. Mean Reward: 16.052. Std of Reward: 9.106. Training.
TouchCube. Step: 300000. Time Elapsed: 1230.887 s. Mean Reward: 18.197. Std of Reward: 5.736. Training.
TouchCube. Step: 350000. Time Elapsed: 1446.545 s. Mean Reward: 19.086. Std of Reward: 3.321. Training.
TouchCube. Step: 400000. Time Elapsed: 1662.651 s. Mean Reward: 19.512. Std of Reward: 1.555. Training.
TouchCube. Step: 450000. Time Elapsed: 1899.574 s. Mean Reward: 19.595. Std of Reward: 1.210. Training.
TouchCube. Step: 500000. Time Elapsed: 2154.839 s. Mean Reward: 19.635. Std of Reward: 0.976. Training.
zation.py:93] Converting to results\1\TouchCube\TouchCube-499992.onnx
ges\mlagents\trainers\torch\distributions.py:163: TracerWarning: Converting a tensor to a Python index mig
n't record the data flow of Python values, so this value will be treated as a constant in the future. This
e to other inputs!
] * inputs.shape[0], axis=0)
ges\mlagents\trainers\torch\networks.py:352: TracerWarning: torch.Tensor results are registered as constar
is warning if you use this function to create tensors out of constant variables that would be the same eve
her case, this might cause the trace to be incorrect.
y_size]),
zation.py:105] Exported results\1\TouchCube\TouchCube-499992.onnx
TouchCube. Step: 550000. Time Elapsed: 2391.675 s. Mean Reward: 19.671. Std of Reward: 0.946. Training.
TouchCube. Step: 600000. Time Elapsed: 2647.336 s. Mean Reward: 19.707. Std of Reward: 0.785. Training.
```

Good results only after 650K

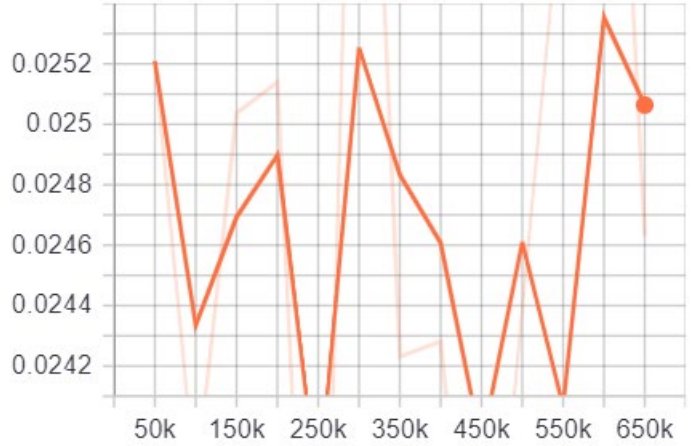
Cumulative Reward
tag: Environment/Cumulative Reward



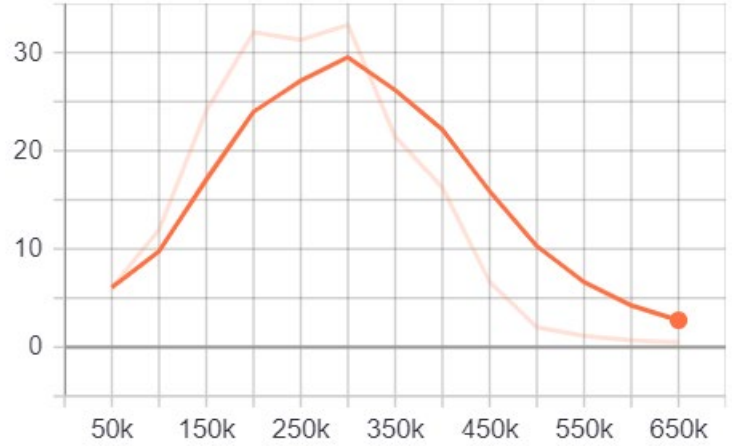
Episode Length
tag: Environment/Episode Length



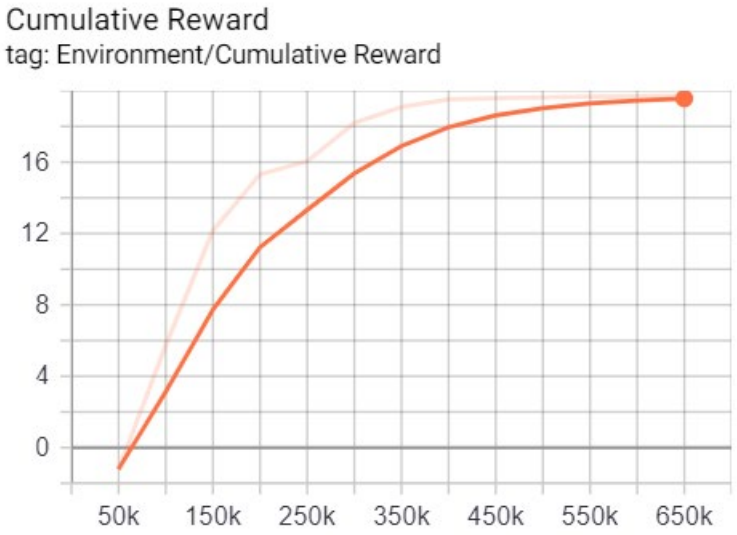
Policy Loss
tag: Losses/Policy Loss

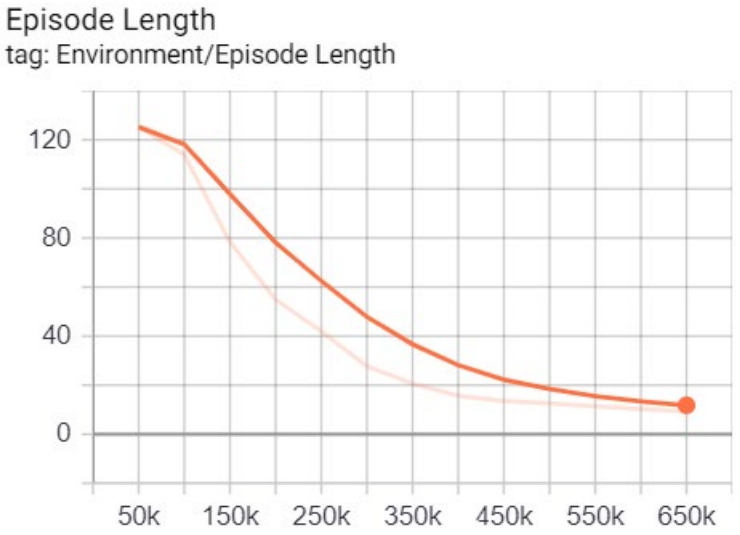


Value Loss
tag: Losses/Value Loss

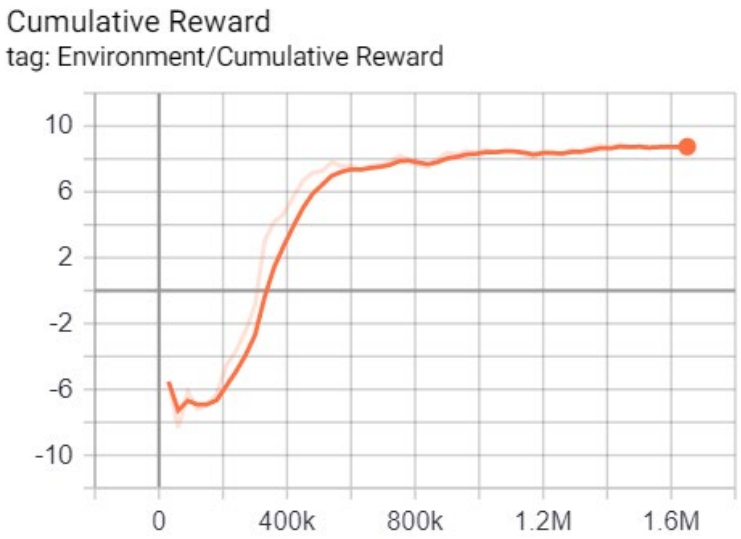


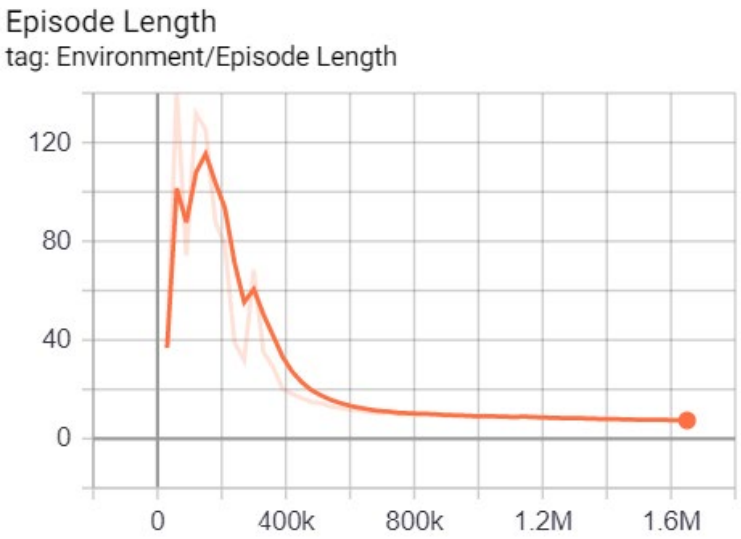
For simple environment, image input is easier than vector input





s = feature map vector from a CNN, size = ?
Input image to the CNN is captured by a camera from top, size = 84x84x3





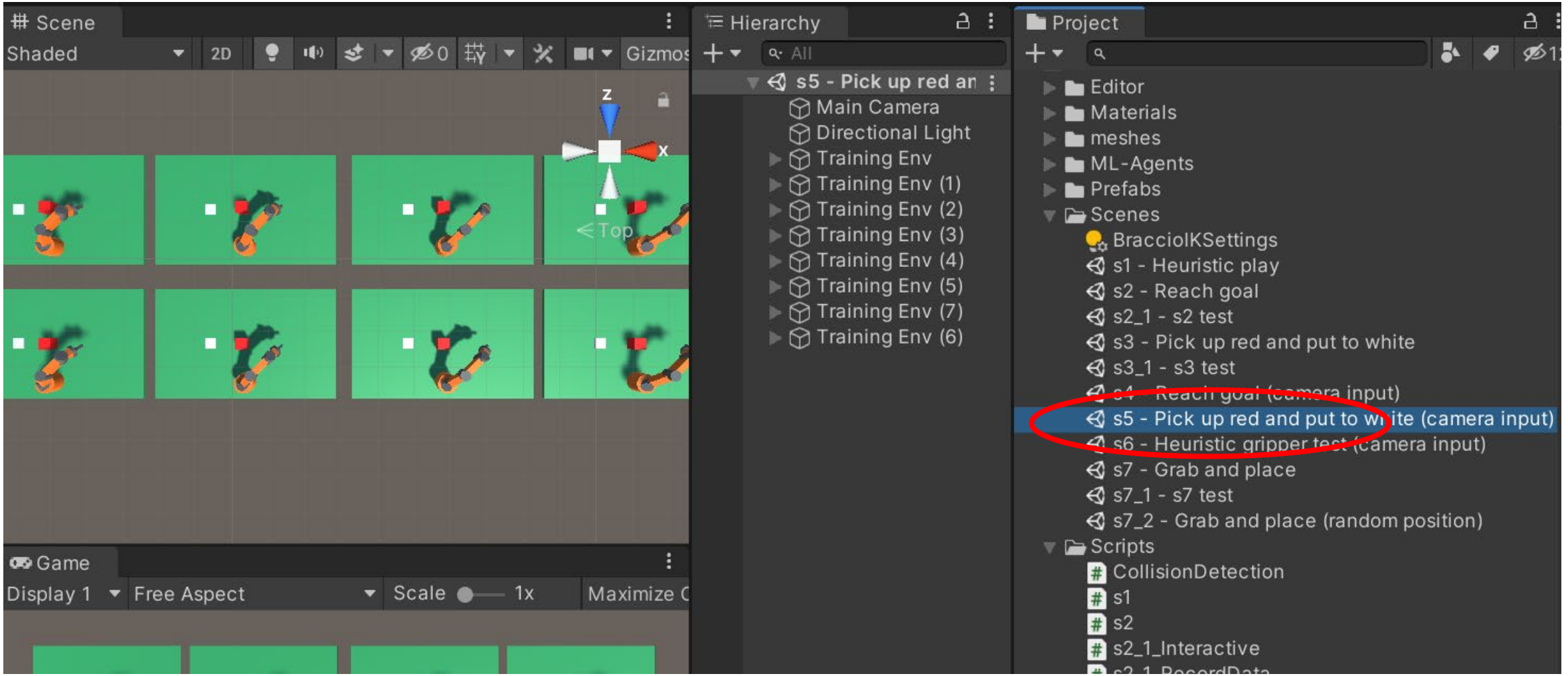
$$s = (\Delta x, \Delta y, \Delta z, \theta_{\mathrm{B}}, \theta_{\mathrm{U}}, \theta_{\mathrm{L}}, \theta_{\mathrm{W}})$$

HW5(1)

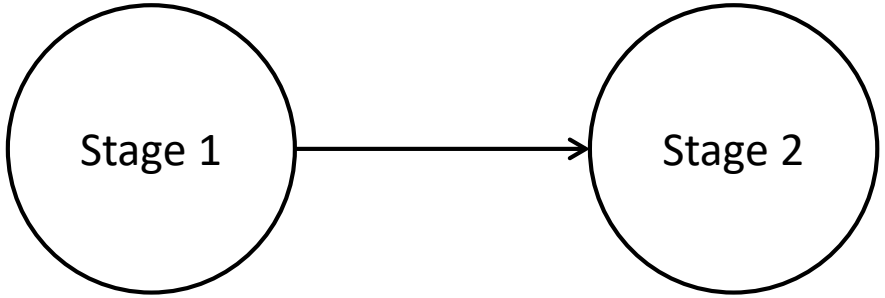
- Describe training setting
- Show tensor board plots and discuss your training performance
- Describe test performance (recorded data, interactive test)

(4) Pick red cube and place it on top of the white cube using camera image information

Open s5


Scene
Shaded
2D
Gizmos
Top
Game
Display 1
Free Aspect
Scale
1x
Hierarchy
All
s5 - Pick up red an
Main Camera
Directional Light
Training Env
Training Env (1)
Training Env (2)
Training Env (3)
Training Env (4)
Training Env (5)
Training Env (7)
Training Env (6)
Project
Editor
Materials
meshes
ML-Agents
Prefabs
Scenes
BracciolKSettings
s1 - Heuristic play
s2 - Reach goal
s2_1 - s2 test
s3 - Pick up red and put to white
s3_1 - s3 test
s5 - Pick up red and put to white (camera input)
s7 - Grab and place
s7_1 - s7 test
s7_2 - Grab and place (random position)
Scripts
CollisionDetection
s1
s2
s2_1_Interactive

Training setting



s = feature map vector from a CNN, size = 2592

Input image to the CNN is captured by a camera from top, size = 84x84x3

$$r = \begin{cases} -0.005 \\ -5 \\ +20 \quad d_1 \leq 0.5 \end{cases} \qquad r = \begin{cases} -0.005 & \text{per step} \\ -5 & \text{collision, out of range} \\ +20 & \text{goal}, d_2 \leq 0.5 \end{cases}$$

NN: ?-512-512-512-4

Time horizon = 2000

Buffer size = 20480

Batch size = 2048

$$a = (\Delta\theta_B, \Delta\theta_U, \Delta\theta_L, \Delta\theta_W)$$

No. of training environment = 9

Goal initialize = randomly positioned in polar system θ=-80~80, r=0.8~1.5

Goal2 initialize = same as goal 1

Arm initialize: $(\theta_B = 0, \theta_U = 45, \theta_L = 45, \theta_W = 45)$

HW5(2)

- Describe training setting
- Show tensor board plots and discuss your training performance
- Describe test performance (recorded data, interactive test)