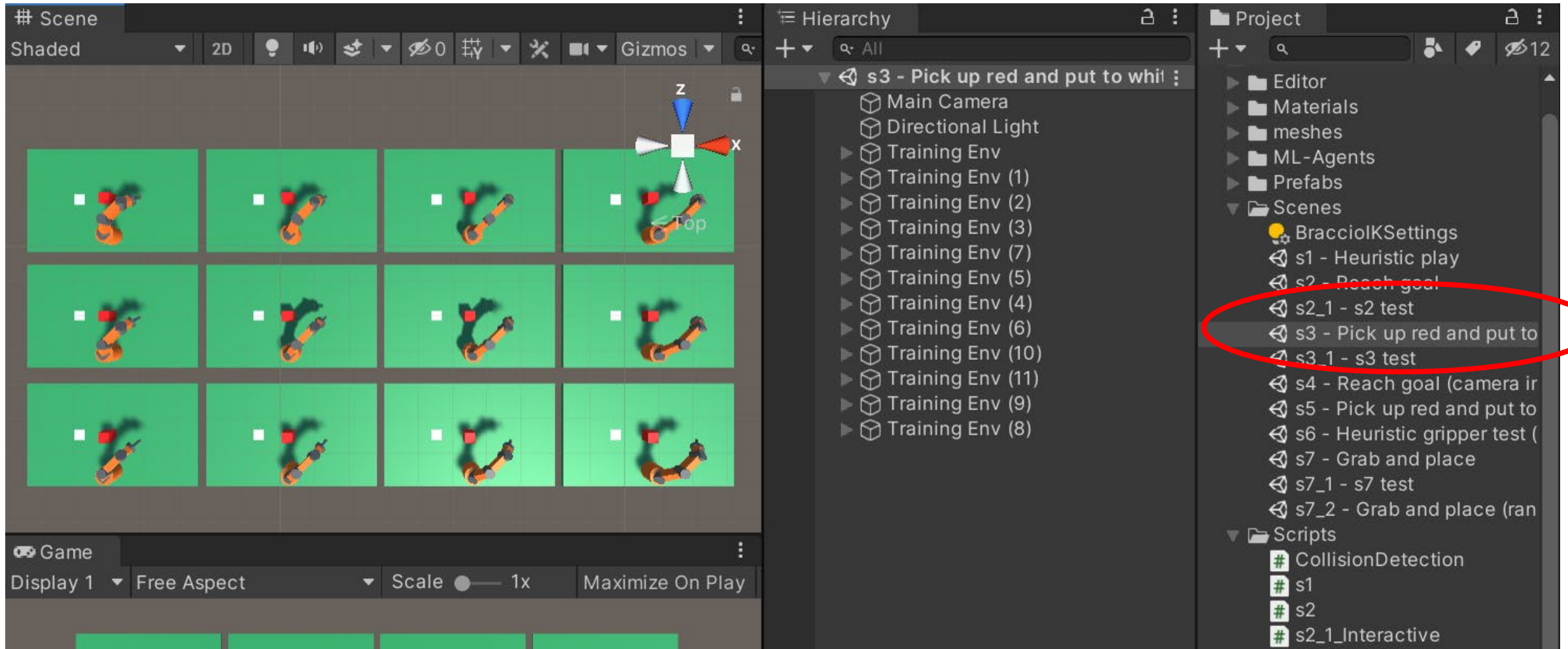
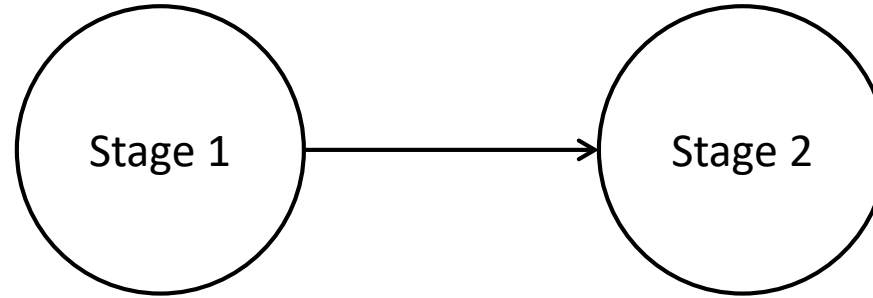


(2) Pick up goal and place it to white area using  $\Delta d$  information

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



# Training setting



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

$$r = \begin{cases} -0.005 \\ -5 \\ +20 \end{cases} \quad d_1 \leq 0.5$$

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

NN: 8-512-512-512-4

Time horizon = 2000

Buffer size = 20480

Batch size = 2048

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

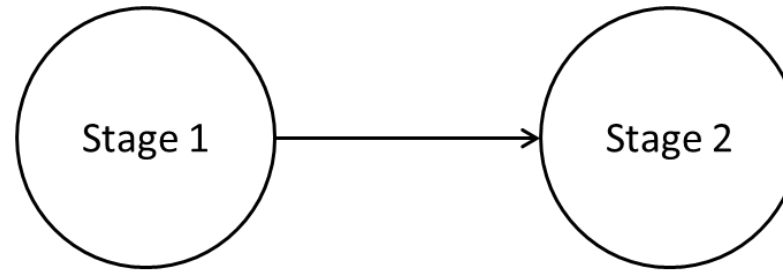
No. of training environment = 9

Goal initialize = randomly positioned in polar system  $\theta = -80 \sim 80$ ,  $r = 0.8 \sim 1.5$

Goal2 initialize = same as goal 1

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

# State



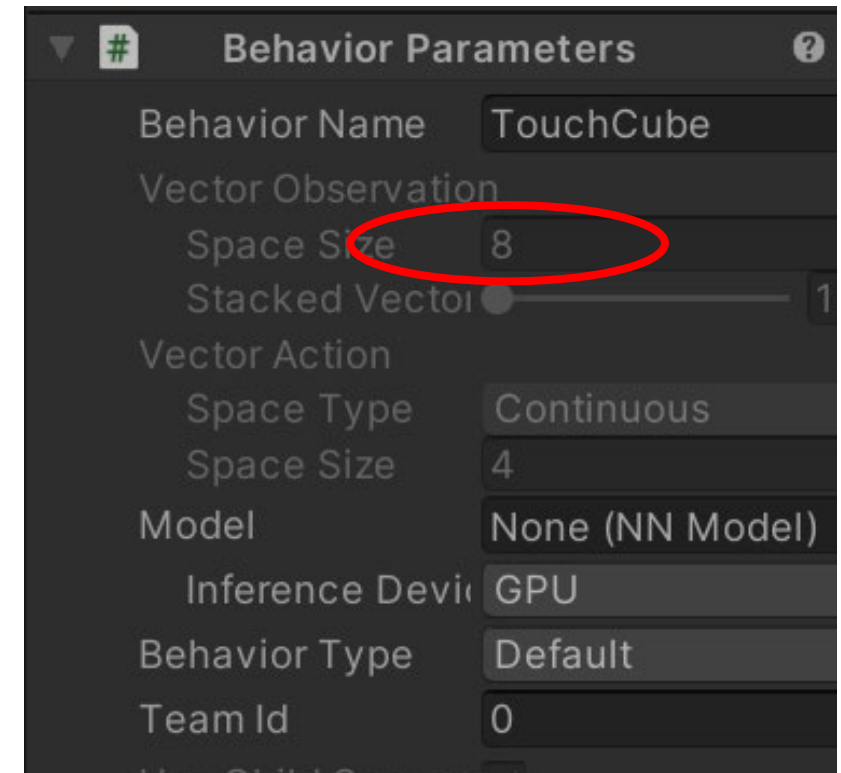
$$s = (sNo, \Delta_1 x, \Delta_1 y, \Delta_1 z, \theta_B, \theta_U, \theta_L, \theta_W) \quad s = (sNo, \Delta_2 x, \Delta_2 y, \Delta_2 z, \theta_B, \theta_U, \theta_L, \theta_W)$$

```
sensor.AddObservation(stage);
```

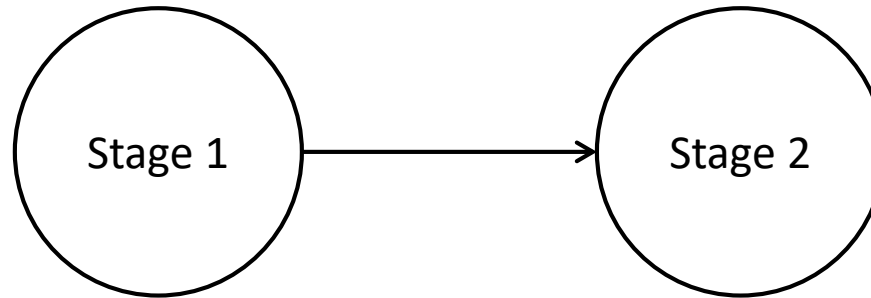
```
if (stage == 1)
    sensor.AddObservation(EndTouchPlane.position - goalUpTouchPt.posi
else //stage =2
    sensor.AddObservation(goalDownTouchPt.position - goal2UpTouchPt.p
```

```
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



$$r = \begin{cases} -0.005 \\ -5 \\ +20 \end{cases} \quad d_1 \leq 0.5$$

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

```
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
}
```

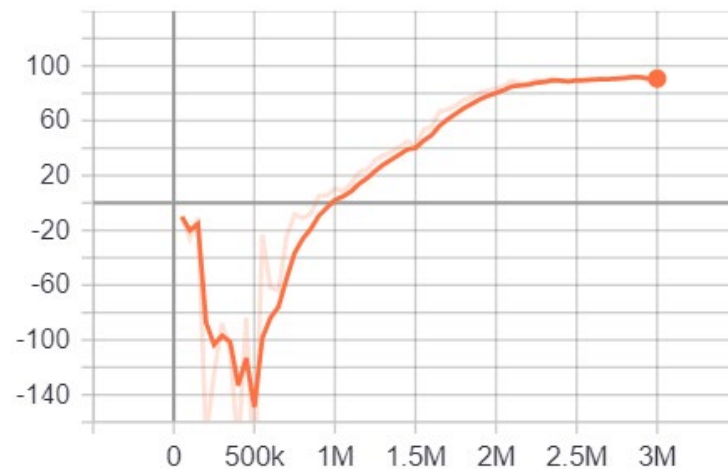
```
else if (PointTouch(goalDownTouchPt, goal2UpTouchPt, 0.3f))
{
    msg = trainingVE.name + " Goal 2! \n";
    Debug.Log(msg);
    AddReward(100.0f);
    EndEpisode();
}
```

# I quit at 3M, looks promising

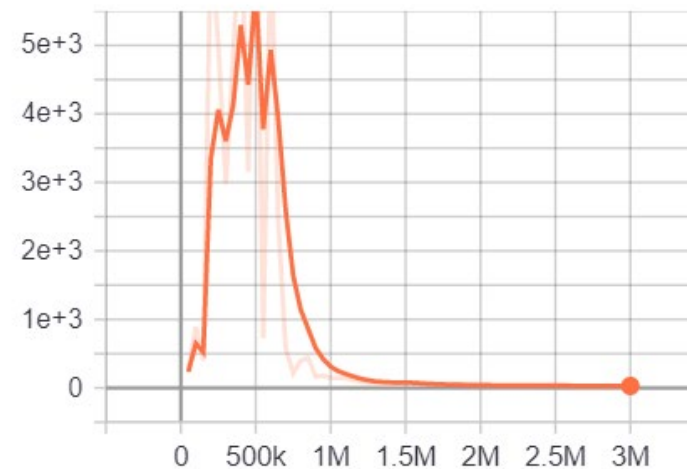
```
TouchCube. Step: 1400000. Time Elapsed: 1521.994 s. Mean Reward: 40.746. Std of Reward: 56.700. Training.
TouchCube. Step: 1450000. Time Elapsed: 1596.218 s. Mean Reward: 44.816. Std of Reward: 57.113. Training.
TouchCube. Step: 1500000. Time Elapsed: 1654.829 s. Mean Reward: 41.636. Std of Reward: 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

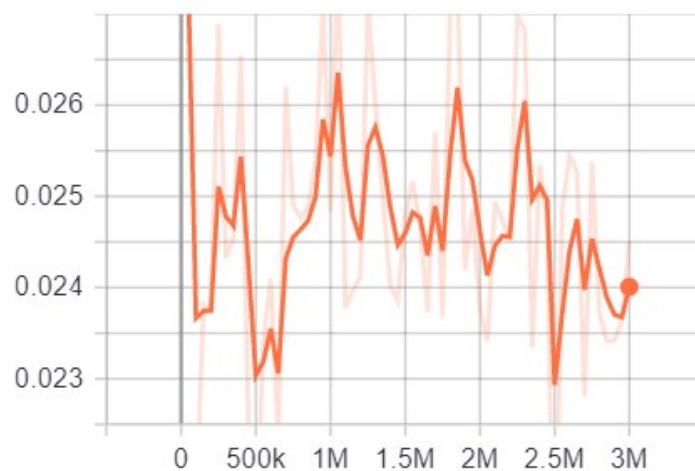
Cumulative Reward  
tag: Environment/Cumulative Reward



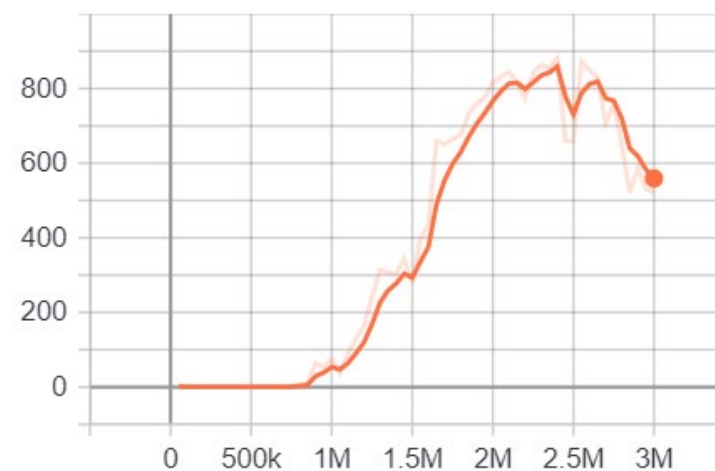
Episode Length  
tag: Environment/Episode Length



Policy Loss  
tag: Losses/Policy Loss



Value Loss  
tag: Losses/Value Loss

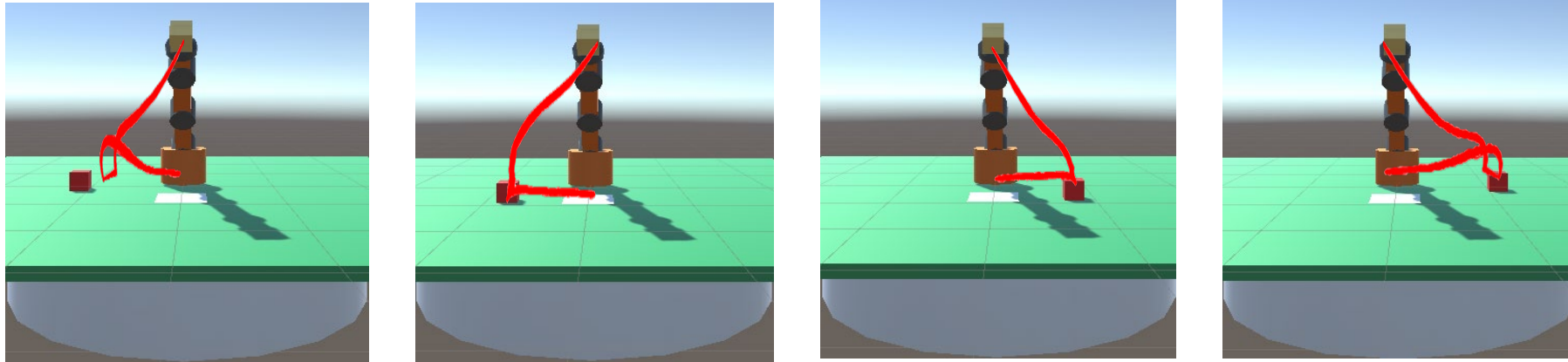




# Test 1 – Performance data

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

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, adversarial environment

Video showing users interacts and plays  
with the intelligent robot arm

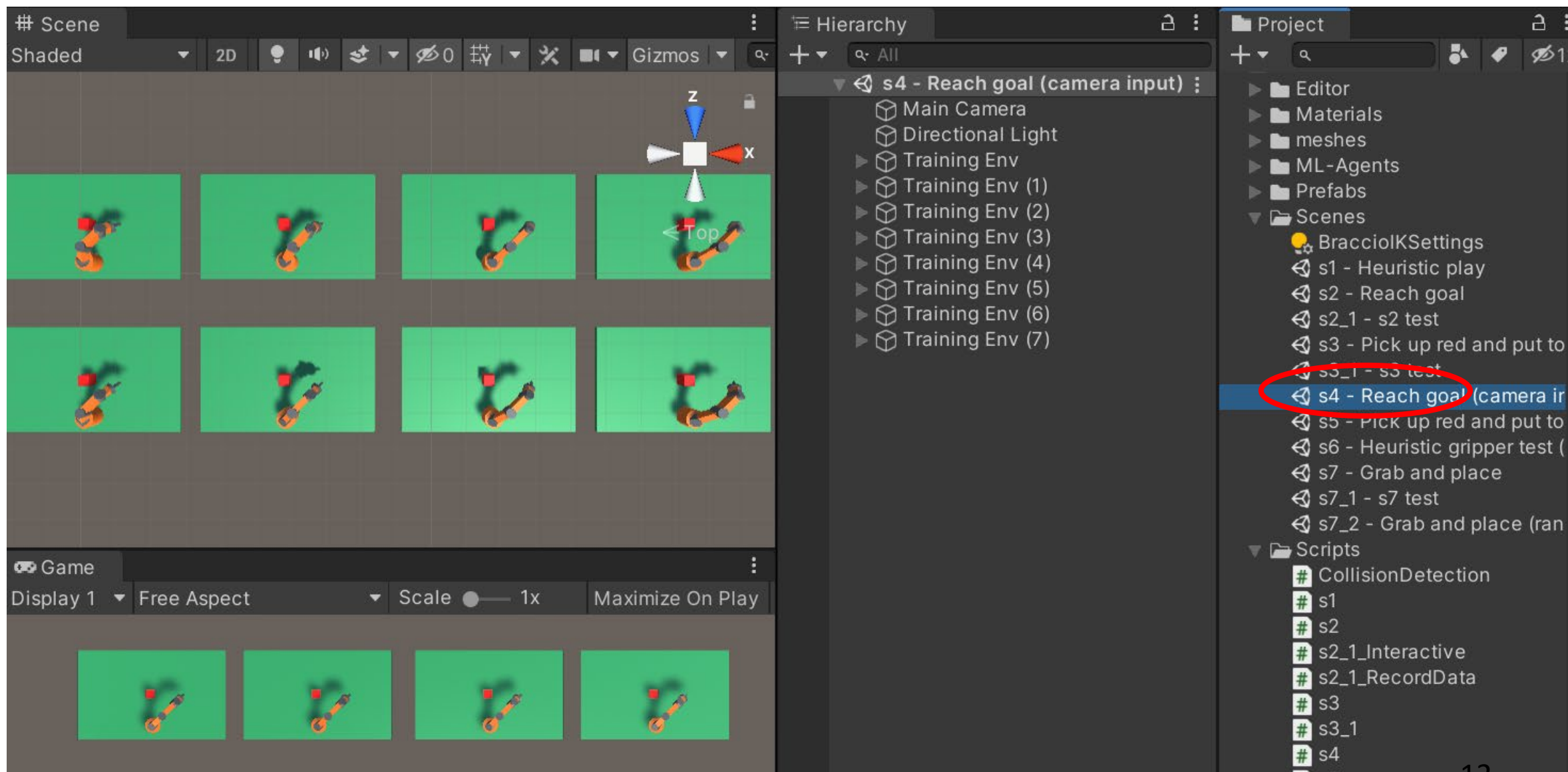
# HW4(2)

- 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



# 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

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

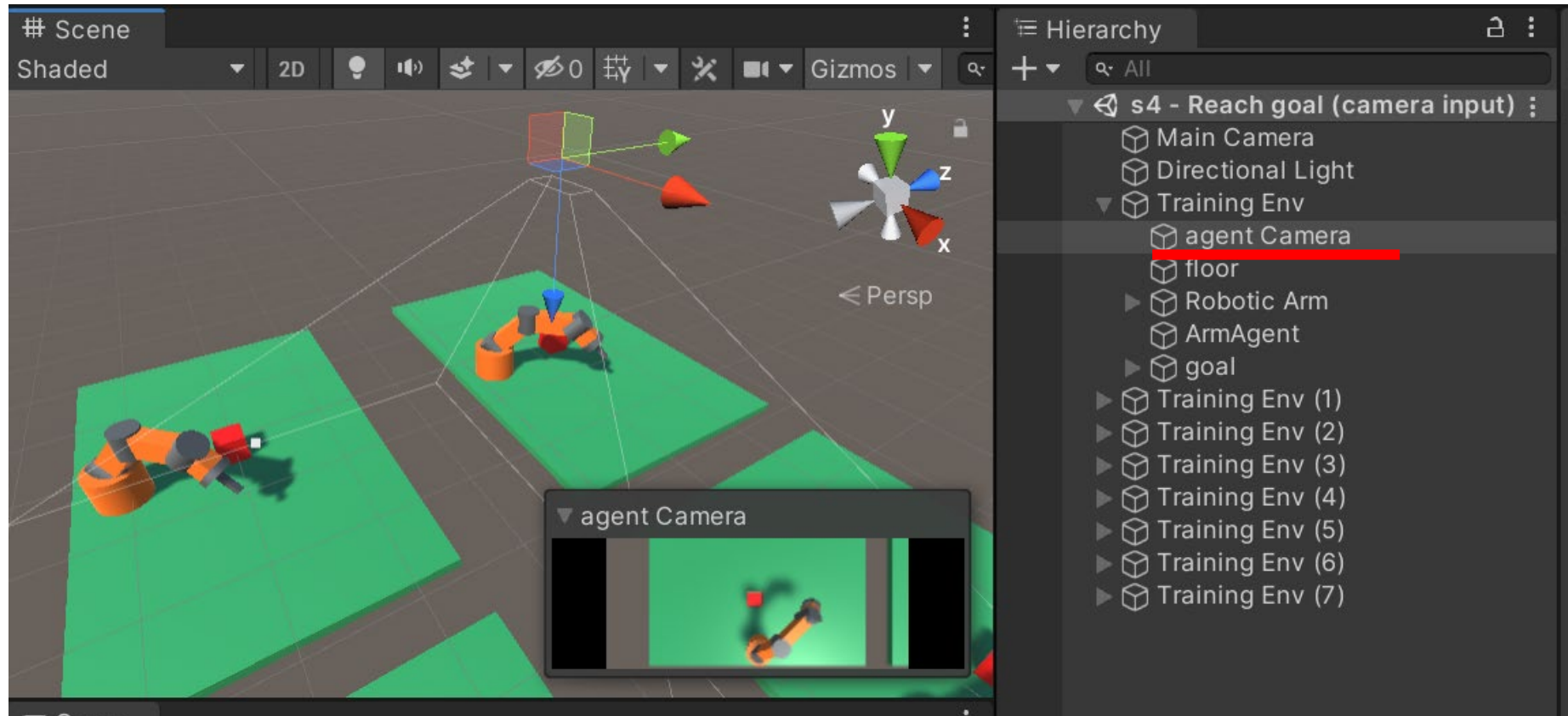
$$r = \begin{cases} -0.005 & \text{per step} \\ -5 & \text{collision, out of range} \\ +20 & \text{goal, } d \leq 0.5 \end{cases}$$

No. of training environment = 9

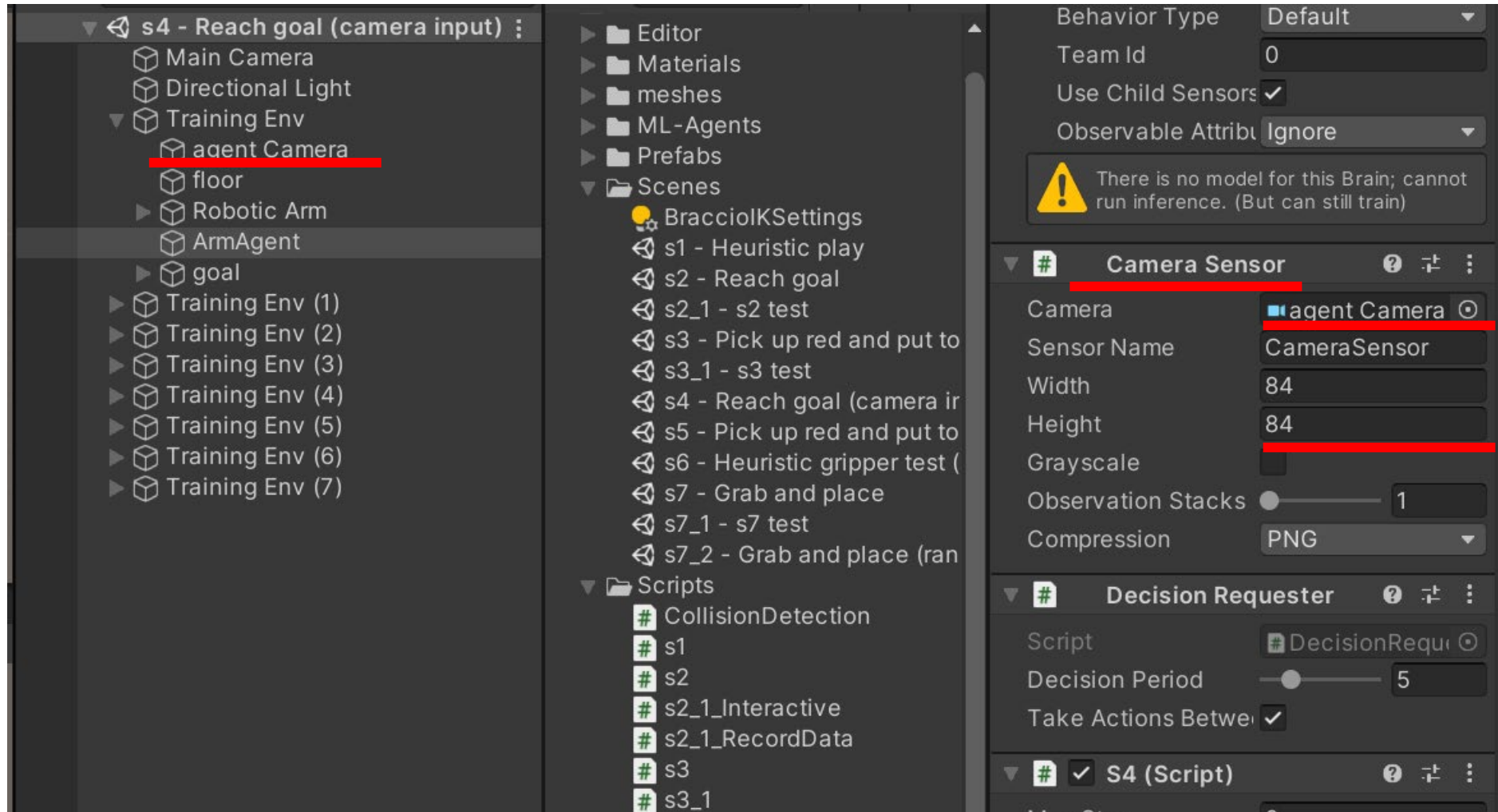
Goal initialize = randomly positioned in polar system  $\theta = -80 \sim 80$ ,  $r = 0.8 \sim 1.5$

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

# Add camera sensor to the robot agent



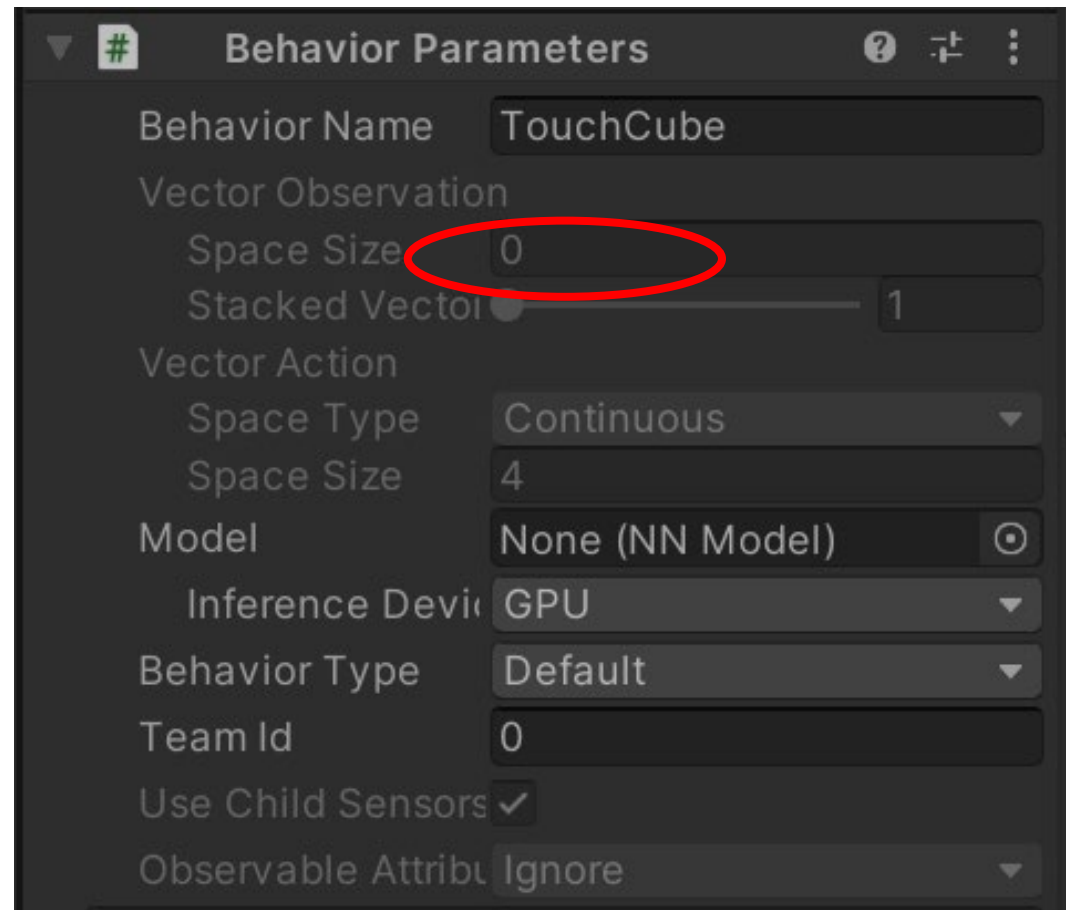
# Add camera sensor to the robot agent





# Vector observation = 0

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

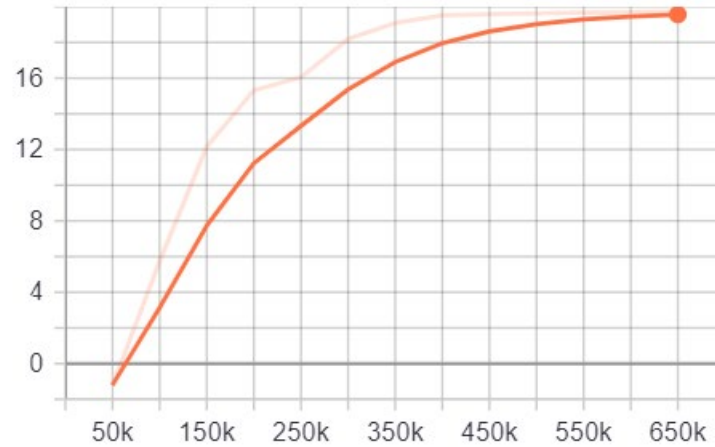


Looks good shortly (600K only)

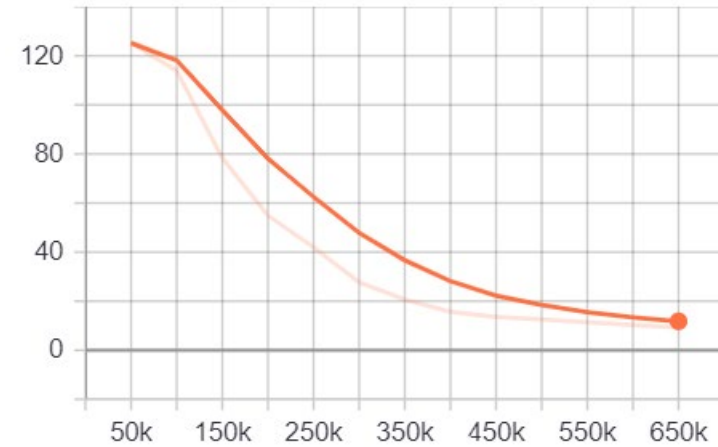
```
TouchCube. Step: 50000. Time Elapsed: 222.493 s. Mean Reward: -1.205. Std of Reward: 12.602. Training.
TouchCube. Step: 100000. Time Elapsed: 417.921 s. Mean Reward: 5.743. Std of Reward: 13.921. Training.
TouchCube. Step: 150000. Time Elapsed: 618.958 s. Mean Reward: 12.166. Std of Reward: 11.773. 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 might
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 constant
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

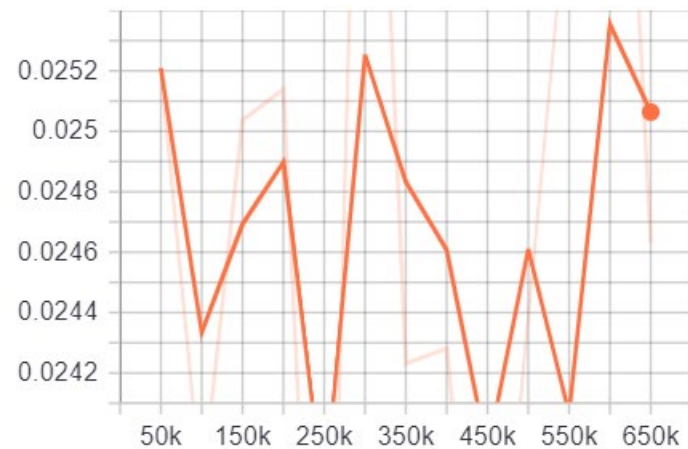
Cumulative Reward  
tag: Environment/Cumulative Reward



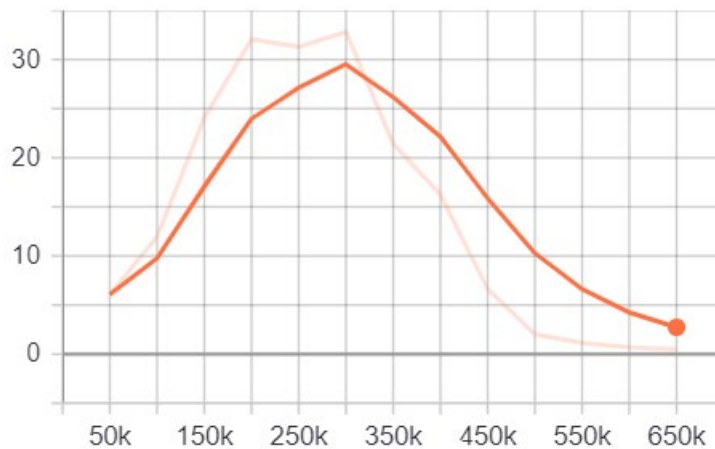
Episode Length  
tag: Environment/Episode Length



Policy Loss  
tag: Losses/Policy Loss



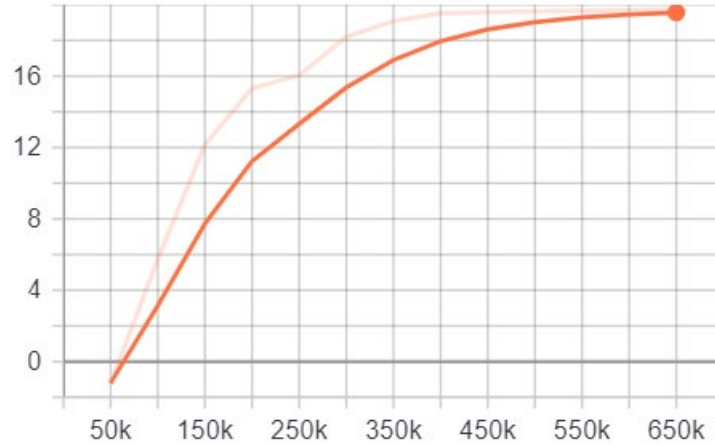
Value Loss  
tag: Losses/Value Loss



# For simple environment, image input is easier than vector input

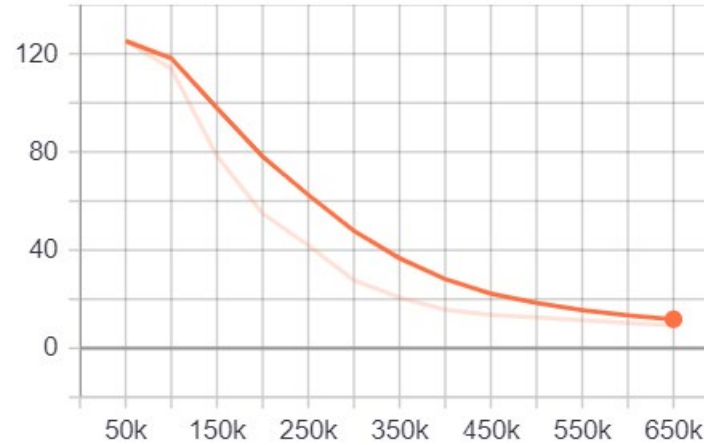
Cumulative Reward

tag: Environment/Cumulative Reward



Episode Length

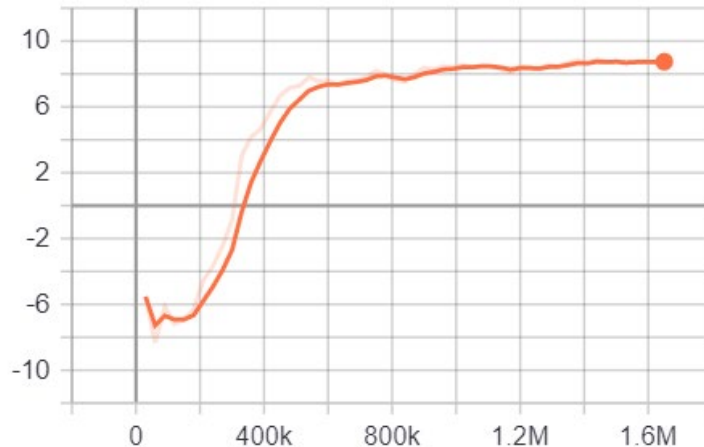
tag: Environment/Episode Length



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

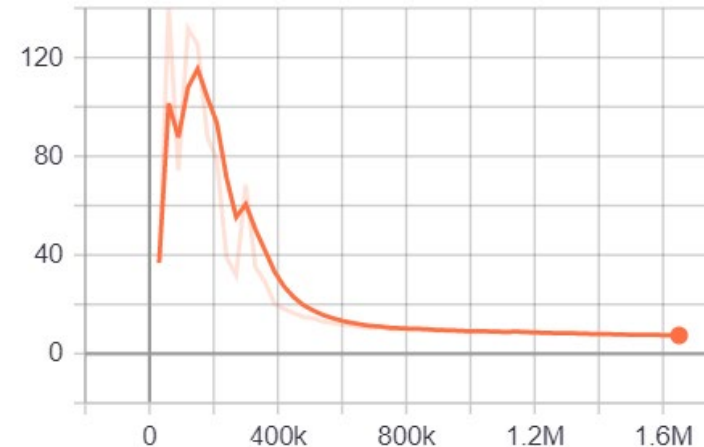
Cumulative Reward

tag: Environment/Cumulative Reward



Episode Length

tag: Environment/Episode Length



$s = (\Delta x, \Delta y, \Delta z, \theta_B, \theta_U, \theta_L, \theta_W)$

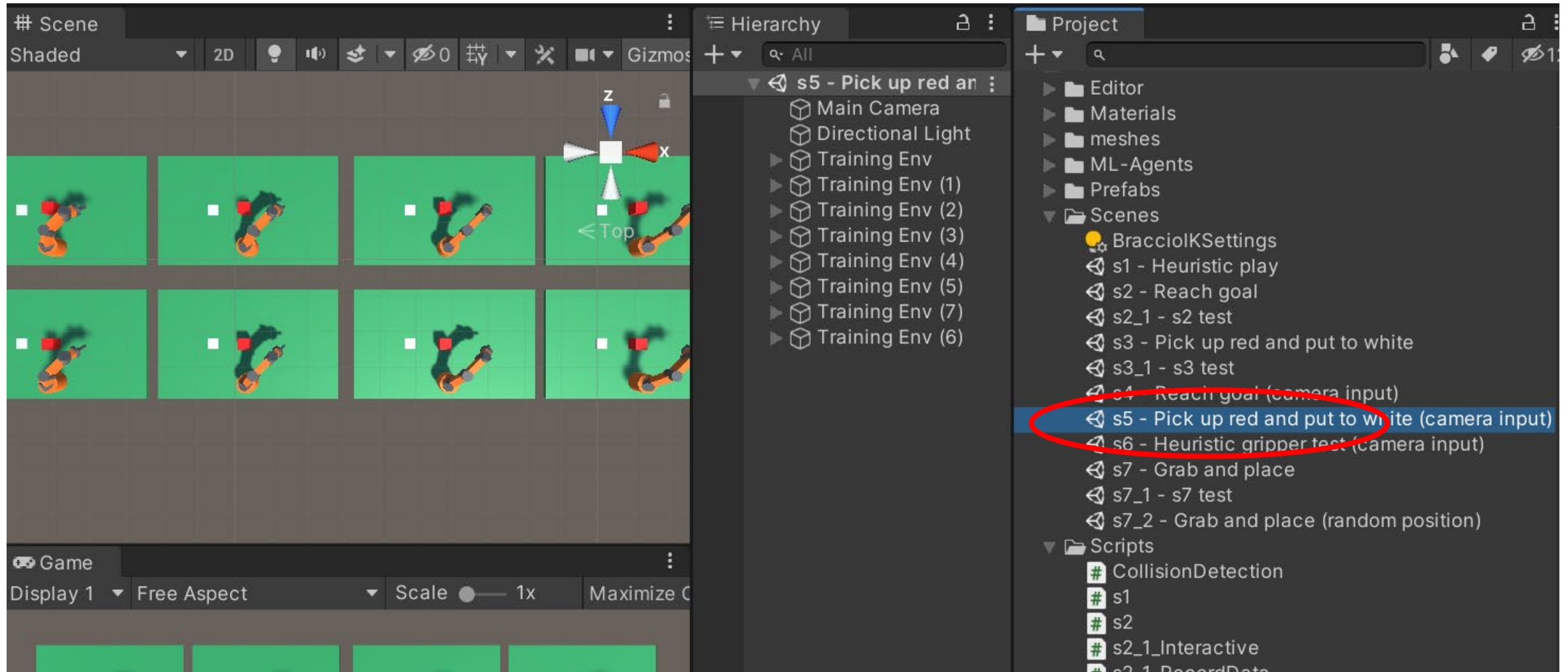
# HW5(1)

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



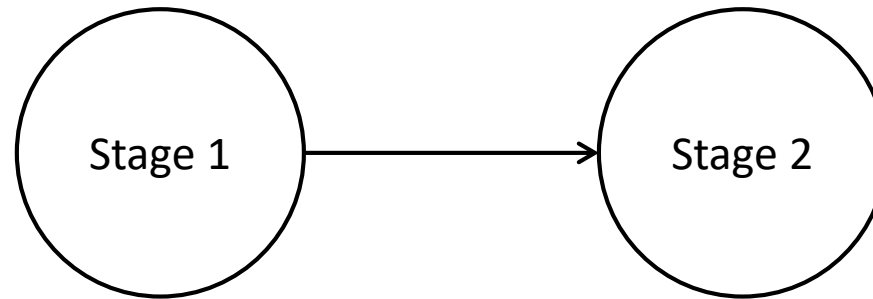
(4) Pick red cube and place it on top of the white cube using camera image information

# Open s5





# 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

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

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

No. of training environment = 9

Goal initialize = randomly positioned in polar system  $\theta = -80 \sim 80$ ,  $r = 0.8 \sim 1.5$

Goal2 initialize = same as goal 1

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

NN: ?-512-512-512-4

Time horizon = 2000

Buffer size = 20480

Batch size = 2048

# HW5(2)

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

