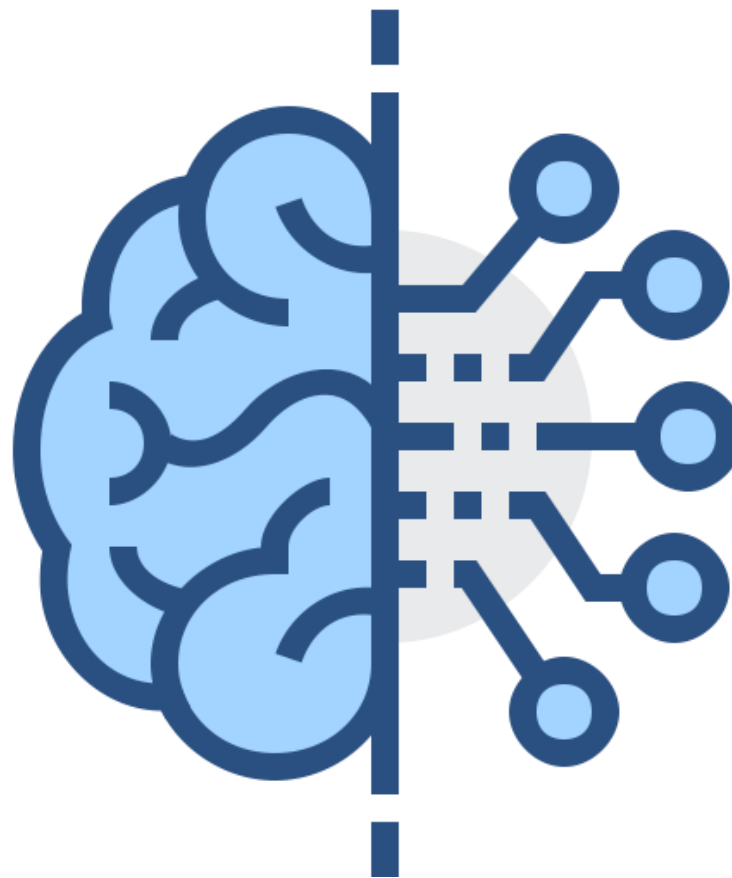


Unity ML-Agents Pong 만들기

2018. 10. 26 민규식



유니티

- 2D/3D 게임 개발 환경을 제공하는 게임 엔진
- 자동차, 애니메이션, 건축, 인공지능 등 다양한 분야를 위한 솔루션 제공
- 전세계 게임 엔진 시장의 절반 정도를 차지하며 등록 개발자 수가 500만명 이상
- C#과 자바스크립트를 스크립트 언어로 사용



Unity ML-agents

~ Reinforcement Learning Korea ~

RL KOREA



유니티의 인터페이스



Unity ML-agents

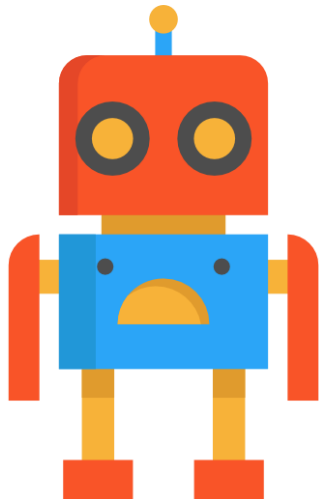
~ Reinforcement Learning Korea ~

RL KOREA



Unity ML-agents

- 강화학습을 구성하는 요소: 강화학습 알고리즘, 환경
- 환경과 에이전트가 서로 정보를 주고 받으며 학습 수행



Agent

Action



State, Reward

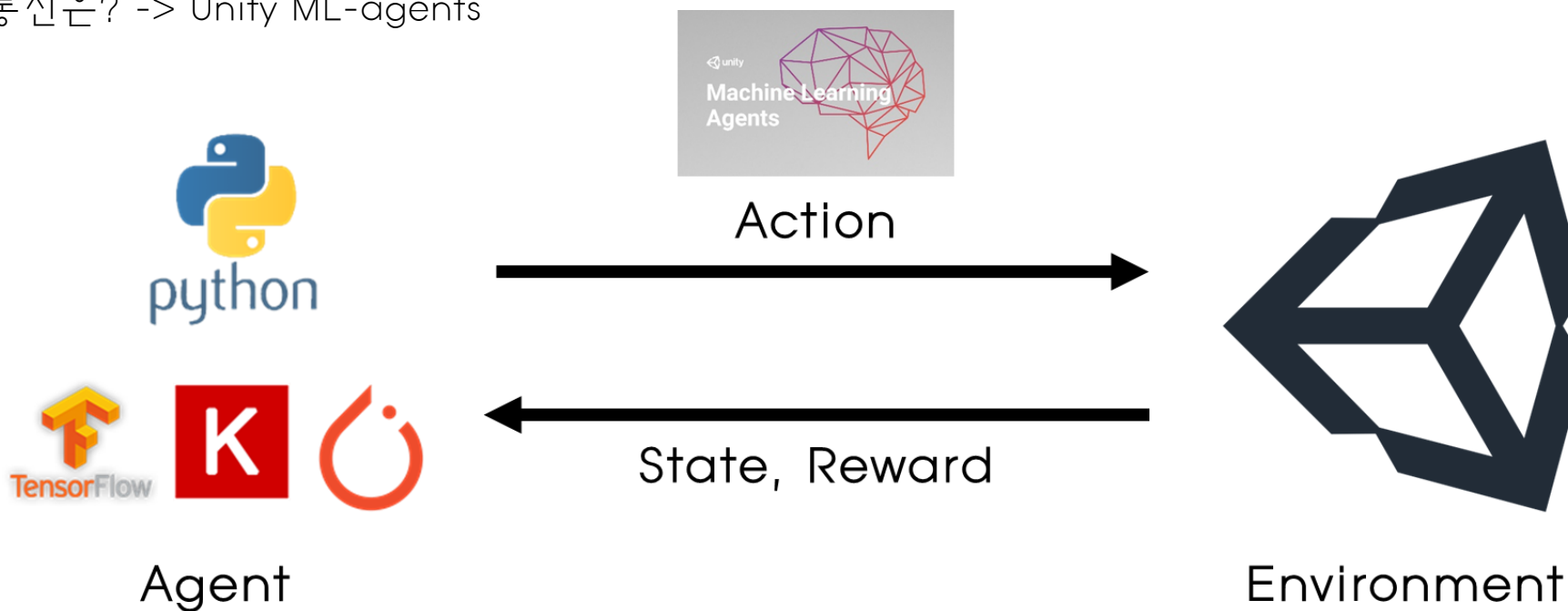


Environment

Unity ML-agents

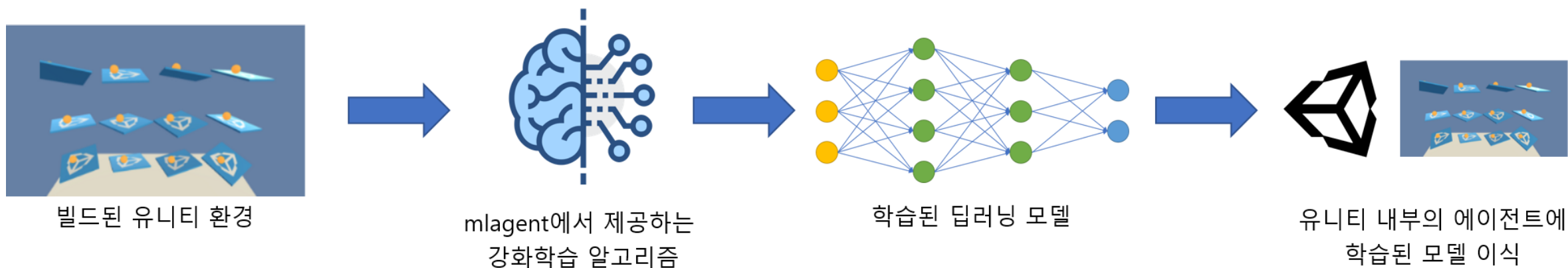
Unity ML-agents

- 일반적으로 딥강화학습 알고리즘은 python을 이용하여 구성
 - 환경은 유니티를 이용하여 제작
- => 둘 간의 통신은? -> Unity ML-agents



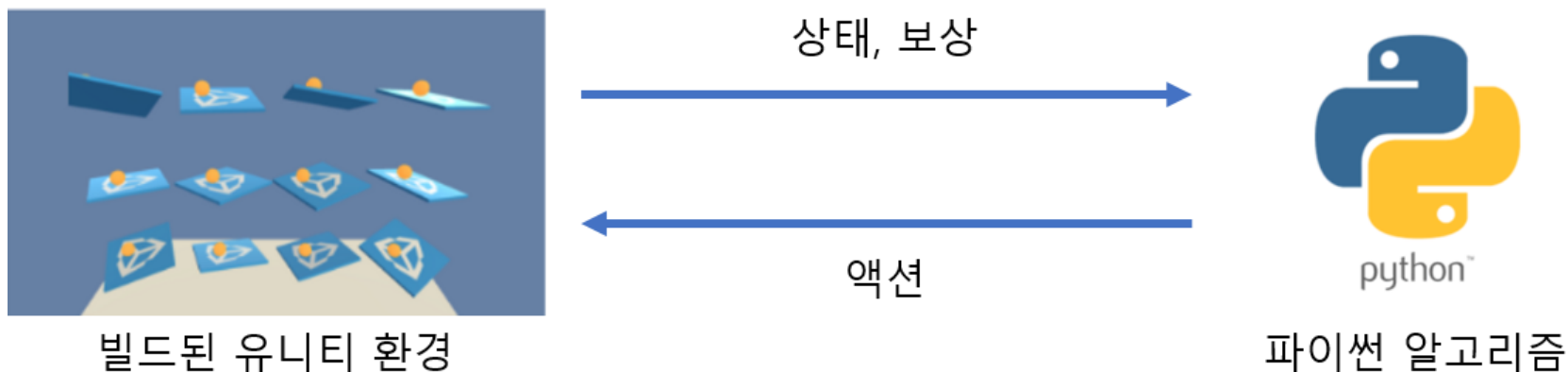
Unity ML-agents

- 학습 방법 1
 - 빌드된 유니티 환경을 mlagent에서 제공하는 알고리즘을 통해 학습
 - 학습된 딥러닝 모델을 유니티 내부의 에이전트에 이식하여 환경을 다시 빌드
=> 학습된대로 에이전트가 행동 (다른 플랫폼에 적용 가능)



Unity ML-agents

- 학습 방법 2
 - 빌드된 유니티 환경과 구성한 파이썬 코드가 서로 통신하며 에이전트를 학습
=> 이번 강의에서는 이 방법을 이용할 예정입니다!



Unity ML-agents

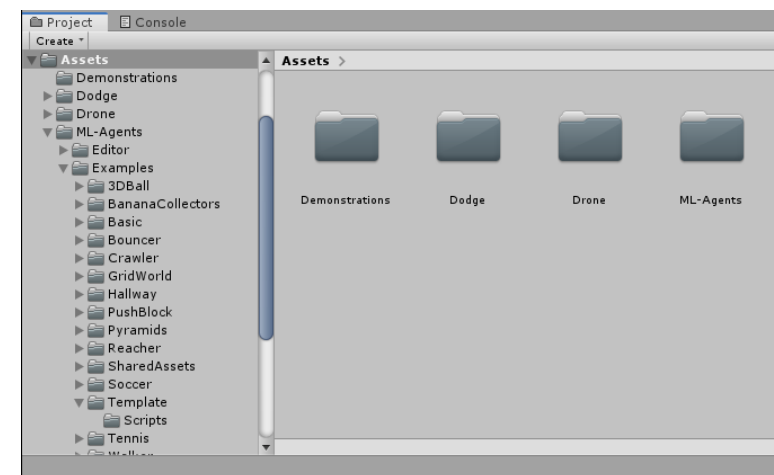
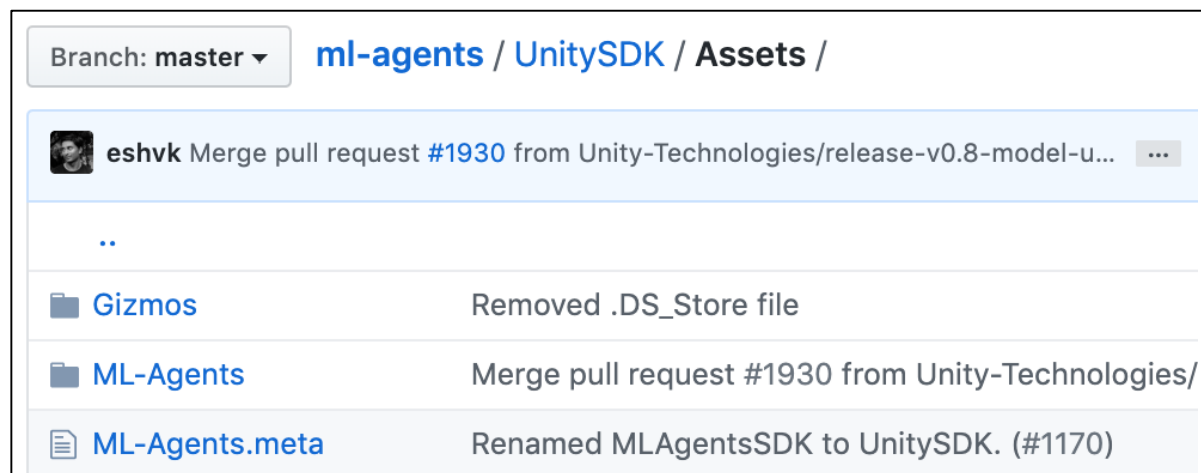
- 유니티 환경 내에 강화학습을 위한 설정을 할 수 있음
- Python과 Unity 환경 간 통신 가능 (state, action, reward)
- Agent, Brain, Academy로 구성
 - Agent: Agent에 대한 코드 작성, Observation, reward, done에 대한 코드 작성
 - Brain: Brain을 통해 agent를 제어하는 방법 결정 및 Observation, action에 대한 설정 가능
 - Player: 직접 사람이 플레이 할 수 있는 brain 설정
 - Heuristic: 사람이 설정한 규칙을 통해 action을 선택하는 brain 설정
 - Internal: 학습된 모델을 통해 Unity 내부에서 환경을 플레이하는 brain 설정
 - External: 외부 Python 코드와 Unity 환경 간 통신을 할 수 있게 하는 설정
 - Academy: Brain을 통합 관리하며 환경에 대한 다양한 설정 가능

Unity ML-agents



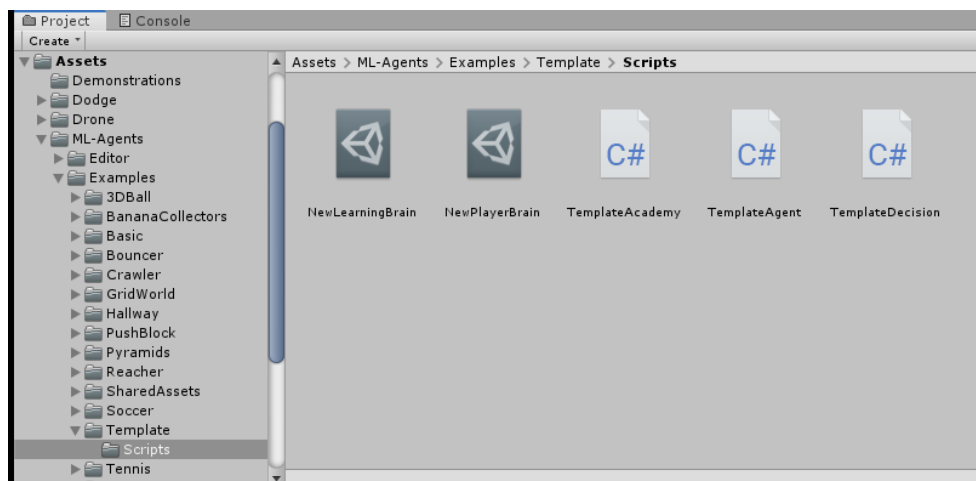
Unity ML-agents

- ML-agents 깃허브에서 ML-agents 프로젝트 받기
 - <https://github.com/Unity-Technologies/ml-agents/releases> (beta 0.8.1)
- ML-agents 프로젝트의 [UnitySDK 폴더 -> Assets 폴더] 내부의 폴더를 유니티의 Assets 내부에 복사



Unity ML-agents

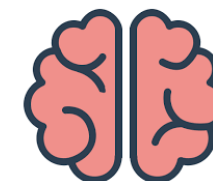
- Assets 내부의 [ML-Agents 폴더 -> Examples 폴더 -> Template 폴더]의 코드들을 이용하면 편해요!
 - Agent 스크립트는 게임 내에서 제어할 대상에게 연결
 - 각 Agent 스크립트에 해당하는 Brain 할당
 - 빈 오브젝트를 생성 후 Academy 스크립트를 연결



Agent 코드



Agent



Brain

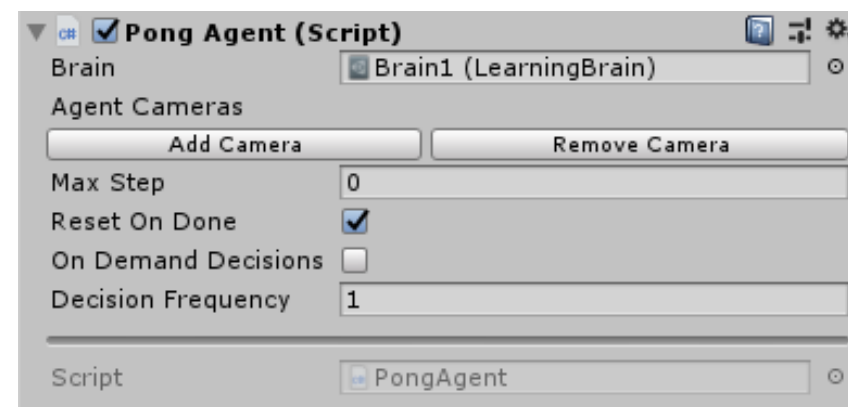
빈 오브젝트



Academy

Agent

- 스크립트 내부는 4개의 함수로 구성
 - CollectObservations 함수: 현재의 상태로 이용할 정보들을 저장 (ex. 위치 좌표 (x, y, z), 각도 정보 (x, y, z), 속도 등)
 - AgentAction 함수: Action을 입력으로 받고 해당 Action에 따라 어떻게 에이전트를 제어할지 결정 -> 이동, 회전 등등 행동을 코딩
 - AgentReset 함수: 게임 한판이 끝난 경우 (Done) 에이전트에 대한 설정을 코딩 (ex. 위치 초기화, 회전 초기화)
 - AgentOnDone 함수: 에이전트를 더이상 사용하지 않을 때 해당 함수 내부에 Destroy를 사용하여 에이전트를 제거
- Inspector View
 - Brain: 해당 에이전트를 위해 사용할 브레인을 설정
 - Agent Cameras: 이미지 입력을 받을 경우 사용할 카메라를 설정
 - Max Step: 게임이 Max Step만큼 진행되면 게임 한판을 끝냄!
 - Reset On Done: 이 설정이 체크되어야 Done이 된 경우 AgentReset 함수 호출
 - On Demand Decisions: 특정 이벤트가 발생했을 때만 액션을 취하도록 설정
 - Decision Frequency: 해당 스텝마다 한번씩 액션을 취하도록 설정





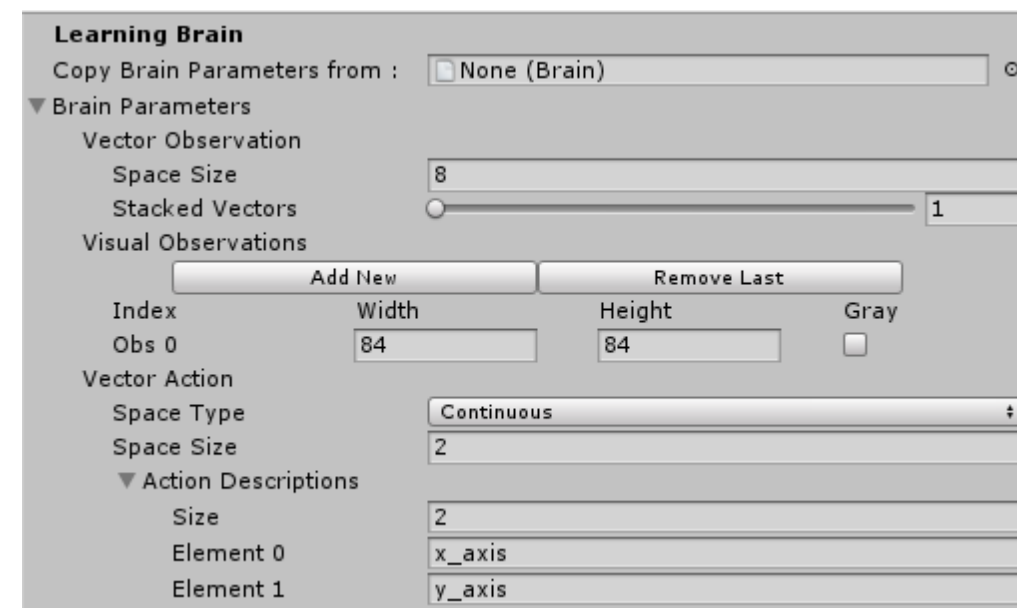
Brain

■ 3가지 종류의 브레인

- Player: 직접 사람이 플레이 할 수 있는 brain 설정
 - 키보드와 액션을 직접 설정해서 에이전트를 키보드를 제어할 수 있음
- Heuristic: 사람이 설정한 규칙을 통해 action을 선택하는 brain 설정
 - Decisions 스크립트에 상태에 대한 규칙을 설정하고 해당 규칙에 따라 액션을 반환하도록 설정
- Training (Internal): Unity에 학습된 모델을 내장하고 환경을 플레이하는 brain 설정
 - ML-agents에서 기본적으로 제공하는 코드로 환경을 학습하면 nn 파일 생성
 - nn 파일을 training brain에 연결해주면 학습된 대로 에이전트가 행동
- Training (External): 외부 Python 코드와 Unity 환경 간 통신을 할 수 있게 하는 설정
 - Python 코드에 observation, reward, done 정보를 전달하고 python 코드로부터 action 정보를 받음

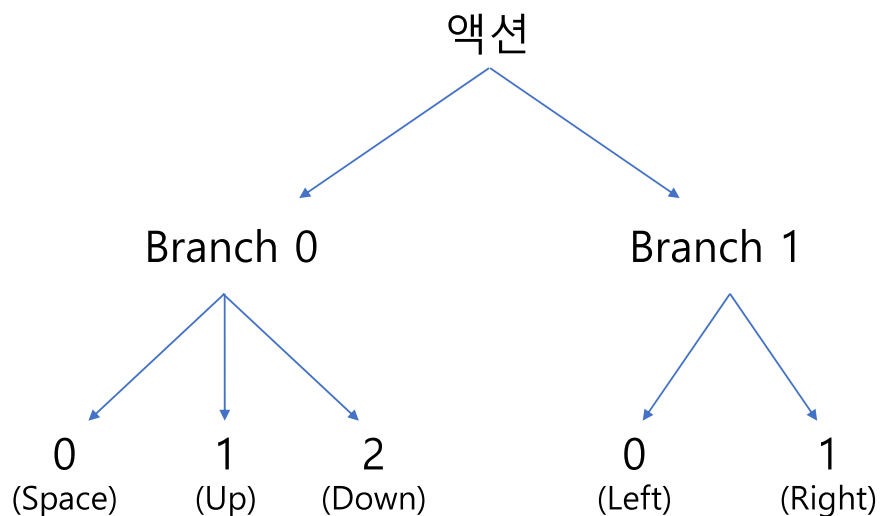
Brain

- 브레인 설정 (모든 브레인에 공통적으로 적용)
- Inspector View
 - Copy Brain Parameters from: 브레인의 파라미터 복사
 - Vector Observation
 - Space Size: Vector Observation의 크기 결정
 - Stacked Vectors: 설정한 스텝만큼 vector 정보를 stack
 - Visual Observation
 - Visual observation의 크기와 grayscale 여부 결정
 - Vector Action
 - 액션의 종류 및 개수 결정 -> Action Descriptions에 액션에 대한 설명 작성



Brain

■ Player 브레인 설정



Edit the discrete inputs for your actions

▼ Discrete Player Actions

Size: 5

▼ Element 0

Key: Space

Branch Index: 0

Value: 0

▼ Element 1

Key: Up Arrow

Branch Index: 0

Value: 1

▼ Element 2

Key: Down Arrow

Branch Index: 0

Value: 2

▼ Element 3

Key: Left Arrow

Branch Index: 1

Value: 0

▼ Element 4

Key: Right Arrow

Branch Index: 0

Value: 1

Edit the continuous inputs for your actions

▼ Key Continuous Player Actions

Size: 4

▼ Element 0

Key: Up Arrow

Index: 0

Value: 0.1

▼ Element 1

Key: Down Arrow

Index: 0

Value: -0.1

▼ Element 2

Key: Left Arrow

Index: 1

Value: -0.1

▼ Element 3

Key: Right Arrow

Index: 1

Value: 0.1

▼ Axis Continuous Player Actions

Size: 2

▼ Vertical

Axis: Vertical

Index: 0

Scale: 0.1

▼ Horizontal

Axis: Horizontal

Index: 1

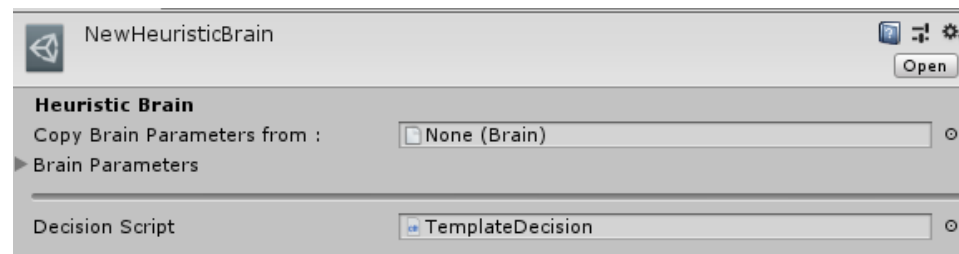
Scale: 0.1

You can change axis settings from Edit->Project Settings->Input

Brain

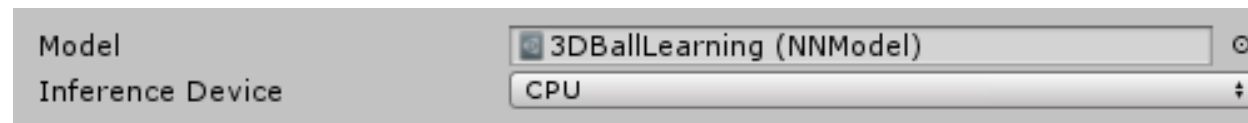
■ Heuristic 브레인 설정

- Decision Script에 코딩된 Decision 스크립트 파일 연결



■ Training 브레인 설정

- 유니티 내부에서 사용할 nn 모델 연결 및 연산 장치 결정



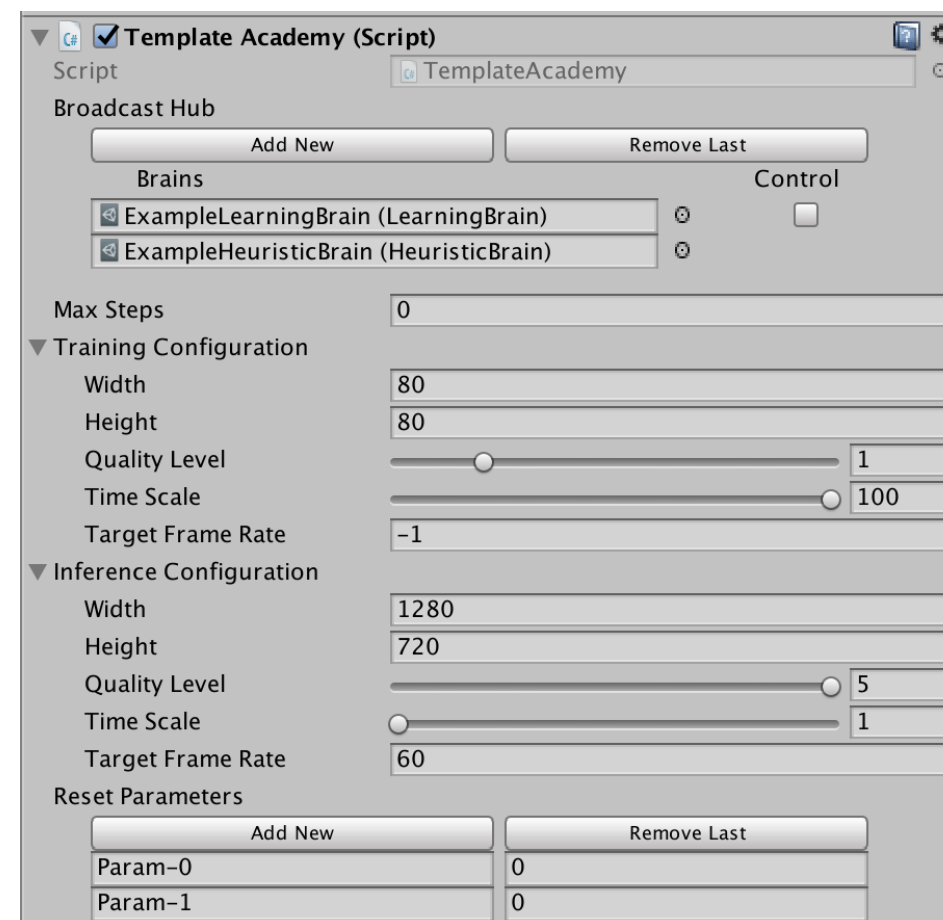
Academy

■ 스크립트 내부는 3개의 함수로 구성

- InitializeAcademy 함수: 환경이 처음 실행될 때 딱 한번 호출되는 함수
- AcademyReset 함수: 매 에피소드가 끝날 때마다 한번씩 호출되는 함수
- AcademyStep 함수: 매 스텝마다 호출되는 함수

■ Inspector View

- Broadcast Hub: 해당 환경에서 사용할 브레인을 추가
 - Learning brain의 control을 체크하면 external, 체크하지 않으면 internal
- Max Steps: Global done을 위한 최대 스텝 (모든 에이전트를 초기화)
- Configuration: 화면의 크기, 그래픽 품질, 화면 업데이트 주기 등 결정
 - Training Configuration: 학습할 때 환경에 대한 설정
 - Inference Configuration: 학습이 끝나고 테스트 할 때 환경에 대한 설정
- Reset Parameters: 외부 (python)에서 조절할 수 있는 파라미터 설정

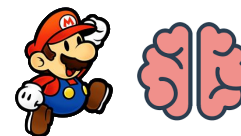


Unity ML-agents

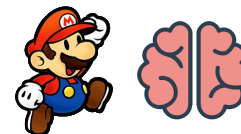
Unity ML-agents



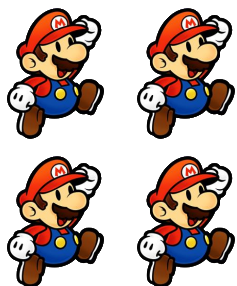
: Single Agent



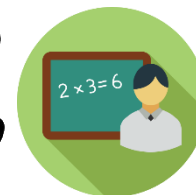
vs



: Adversarial Agents



: Multi-agents

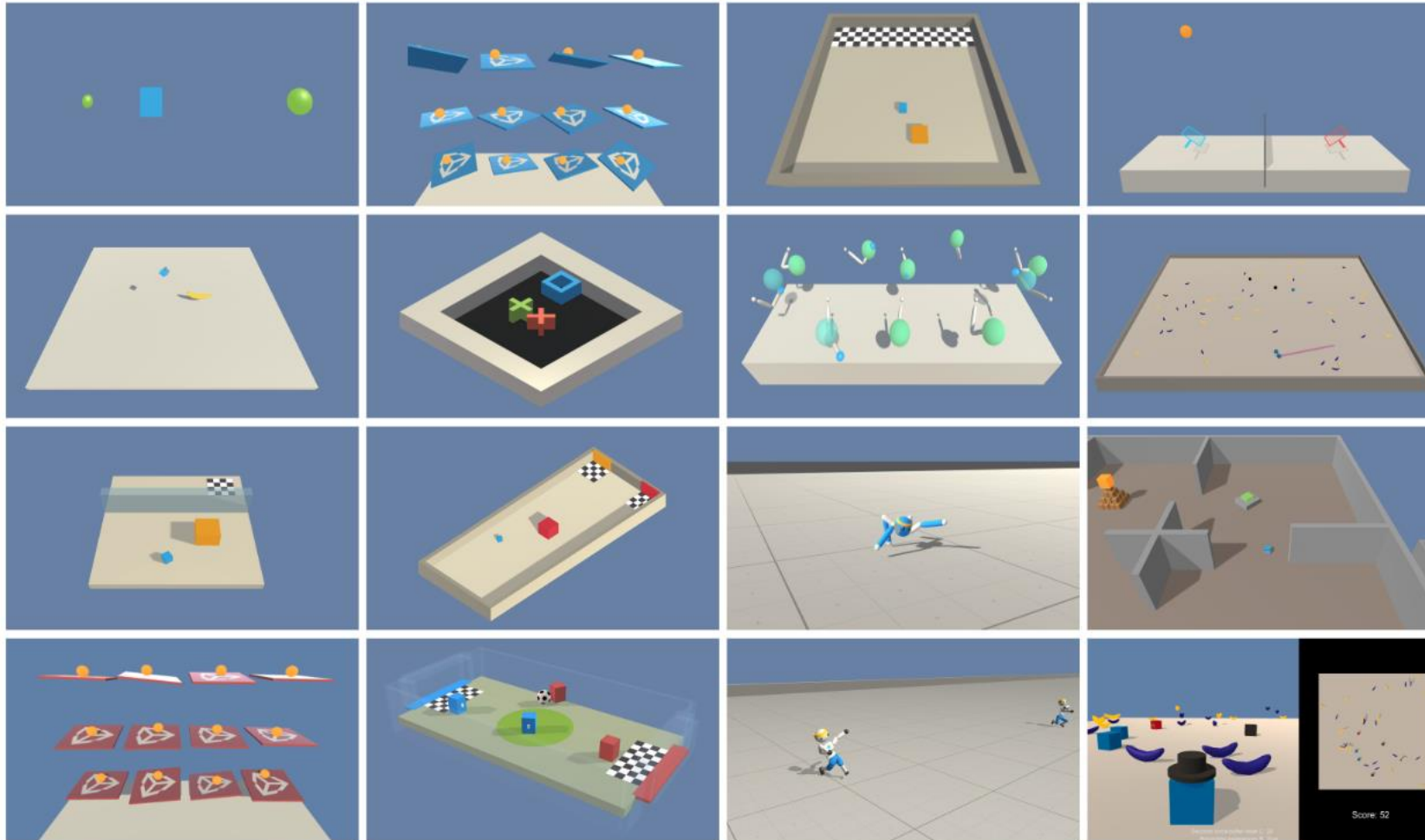


: Imitation Learning

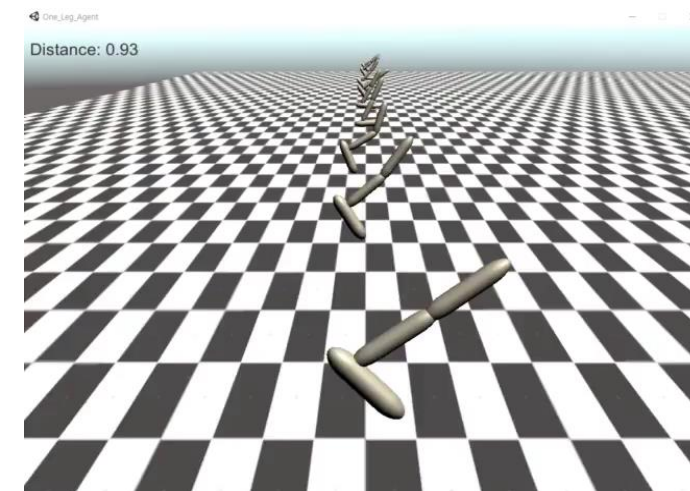
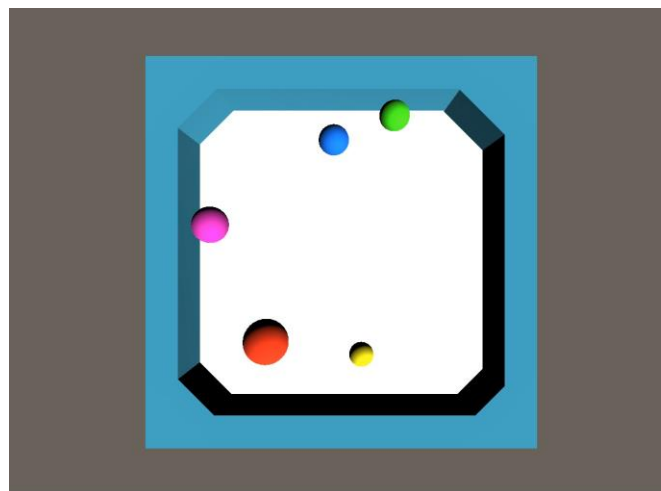
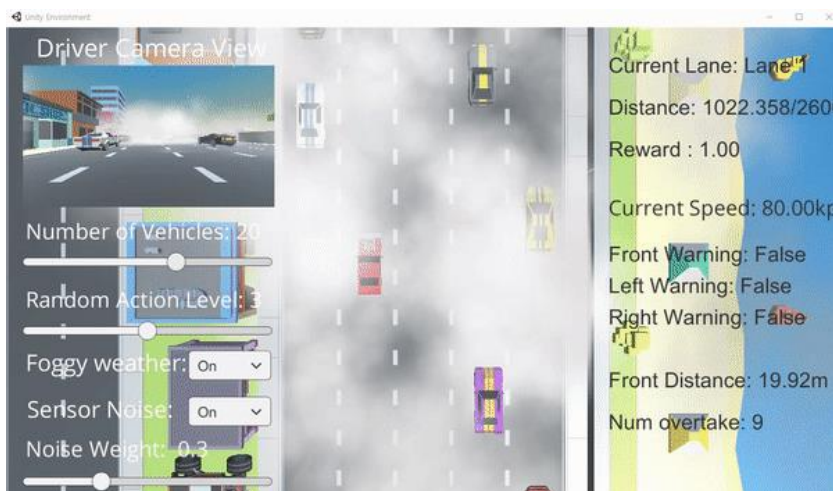
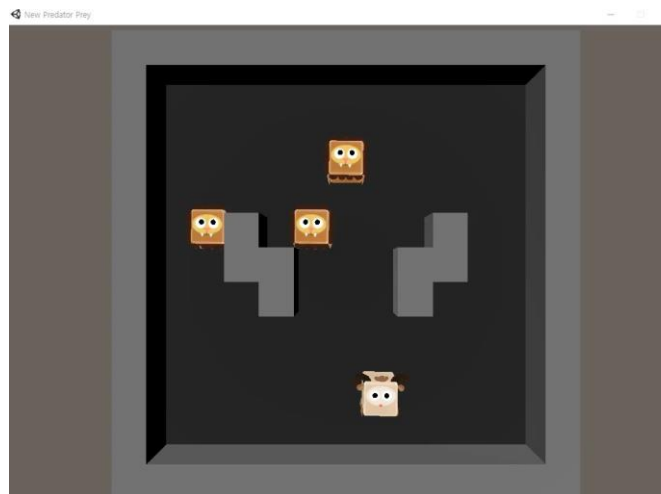
Unity ML-agents

~ Reinforcement Learning Korea ~

RL KOREA



Unity ML-agents

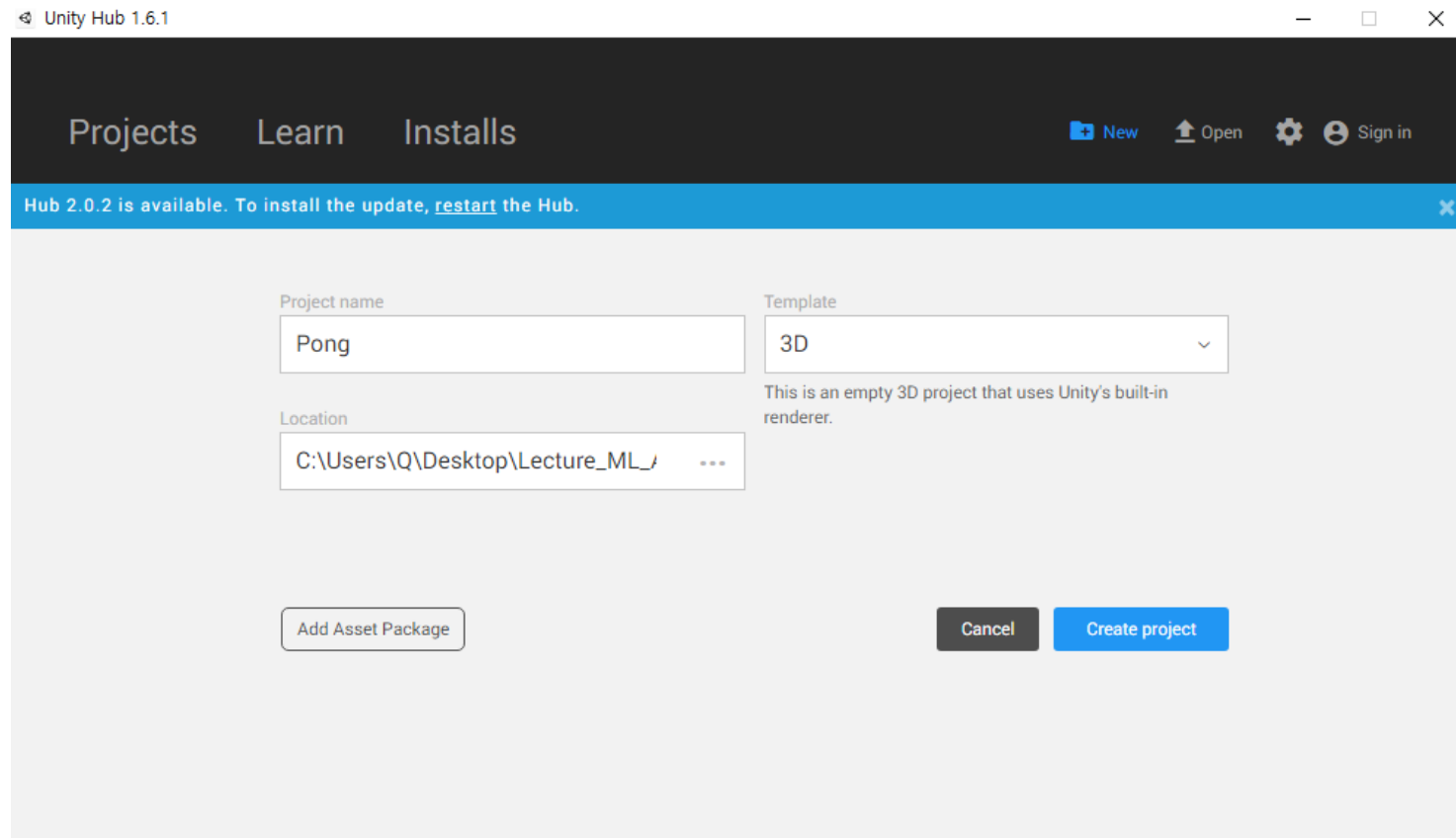


Unity ML-agents 예제



Learning (External)

- Pong 환경 만들기!!
- 새 프로젝트 제작



Unity ML-agents 예제

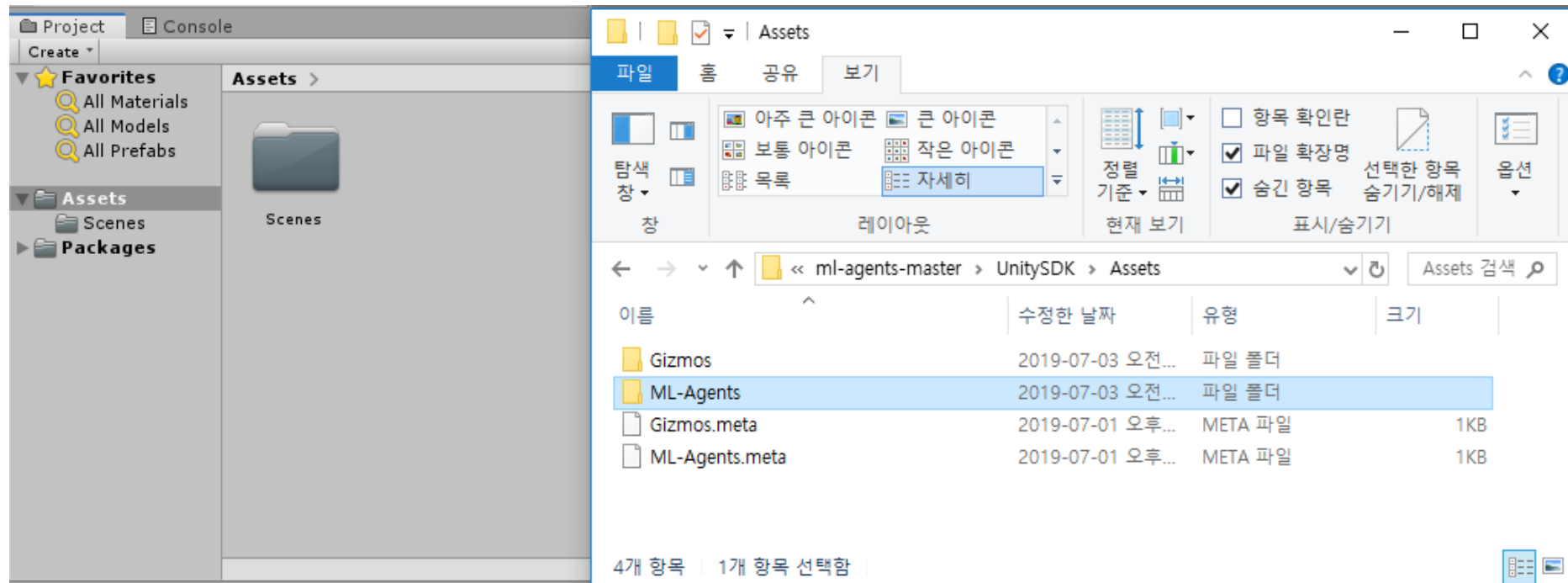
~ Reinforcement Learning Korea ~

RL KOREA



Learning (External)

- UnitySDK -> Assets -> ML-Agents를 유니티 프로젝트 뷰의 Assets에 복사



Unity ML-agents 예제

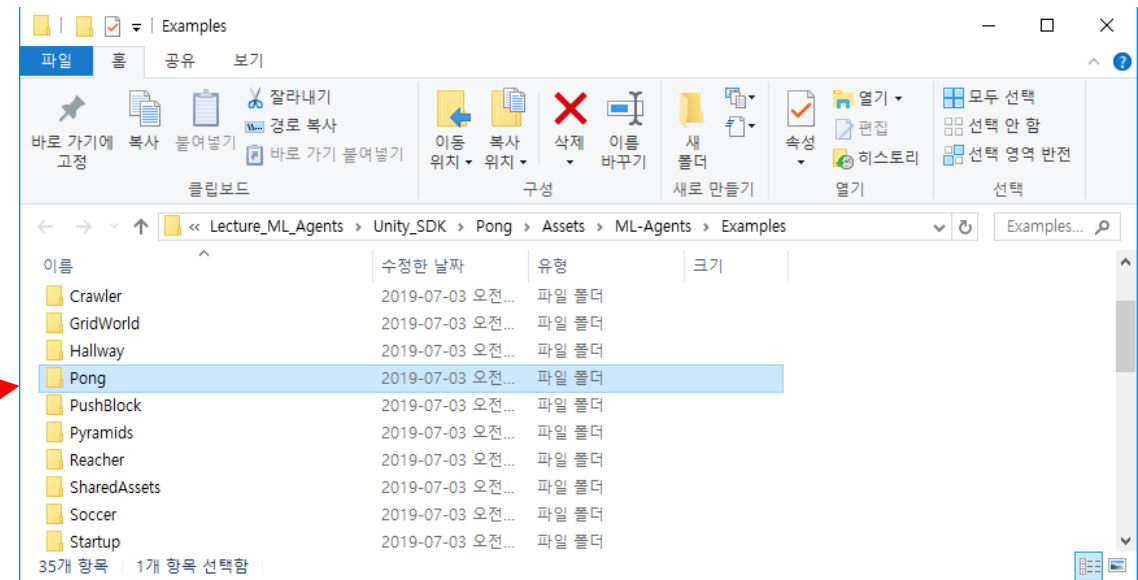
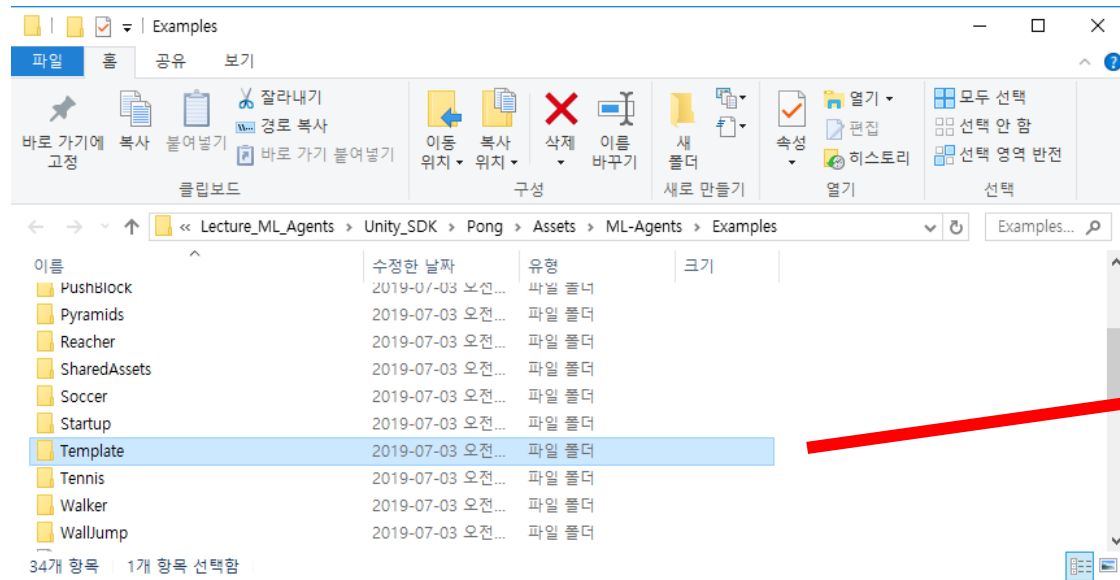
~ Reinforcement Learning Korea ~

RL KOREA



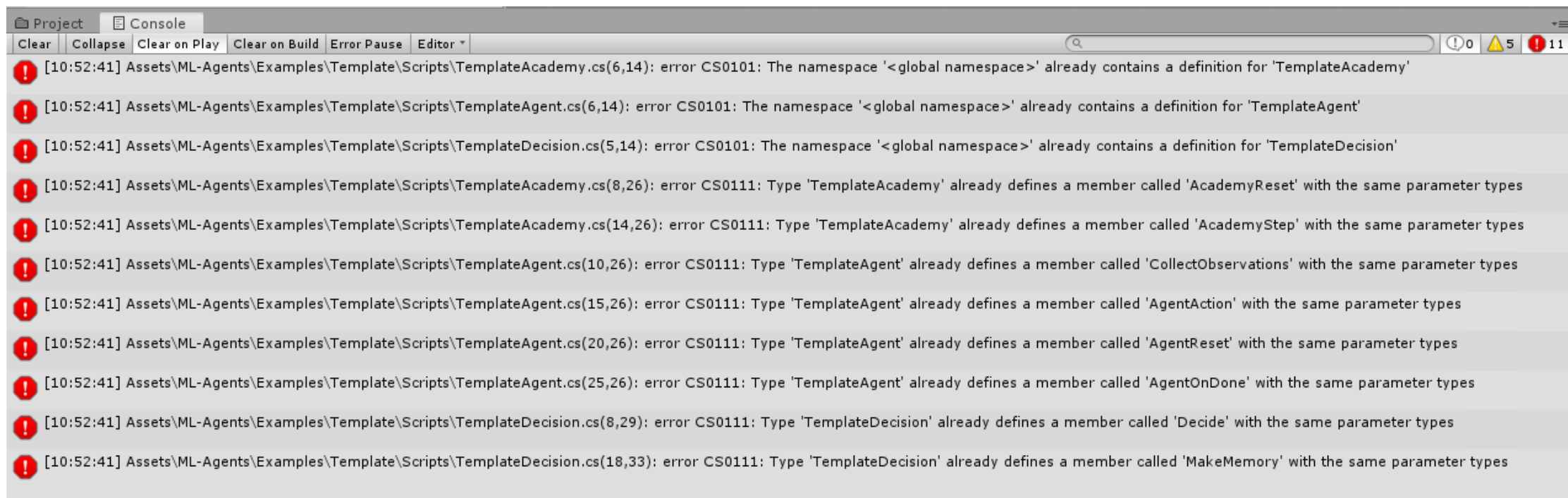
Learning (External)

- Assets -> ML-agents 내부의 Template을 Pong으로 폴더명 변경



Learning (External)

- 다시 유니티로 돌아오면 에러가 한가득!! -> 스크립트 파일들의 namespace 변경 필요!

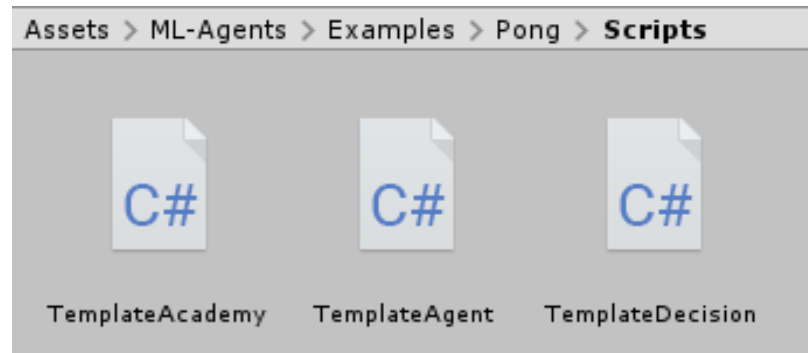
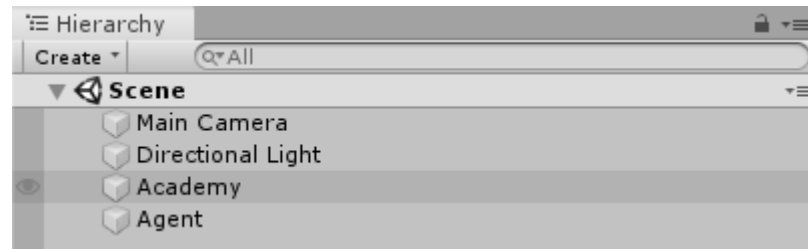
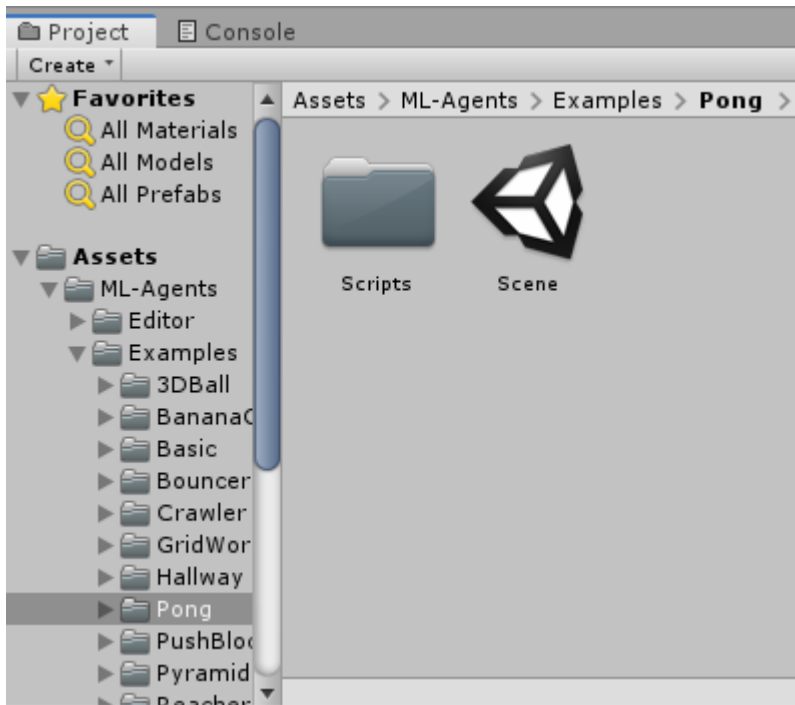


Unity ML-agents 예제



Learning (External)

- 우선 [ML-Agents -> Examples -> Pong] 내부의 Scene 파일을 열어줌! (기본적인 Template 파일들)



Unity ML-agents 예제

~ Reinforcement Learning Korea ~

RL KOREA

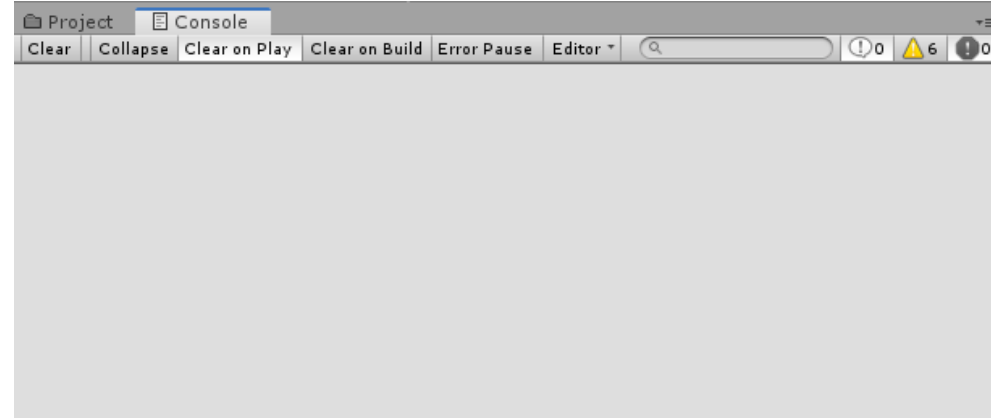
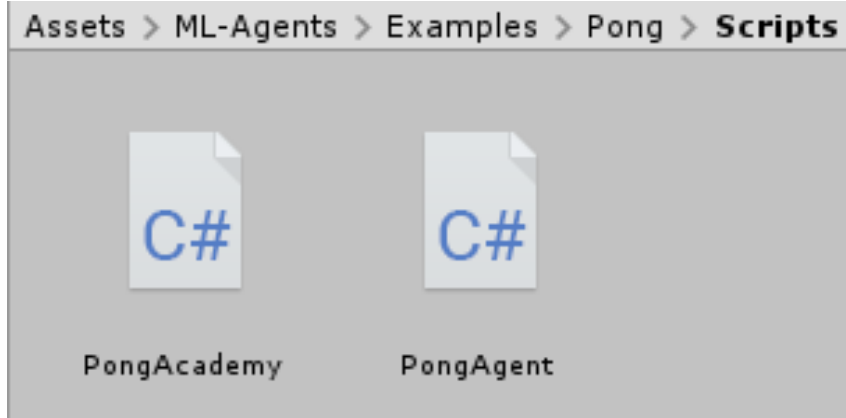


Learning (External)

- Decision은 삭제하고 각 스크립트 파일을 이름 변경 후 열어서 namespace 변경 -> 에러가 사라졌어요~

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using MLAgents;
5
6 public class PongAcademy : Academy {
```

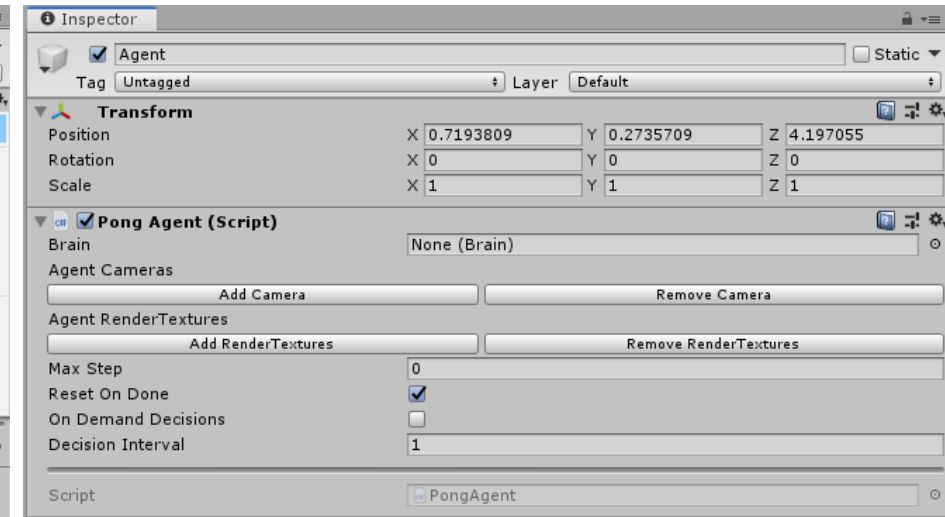
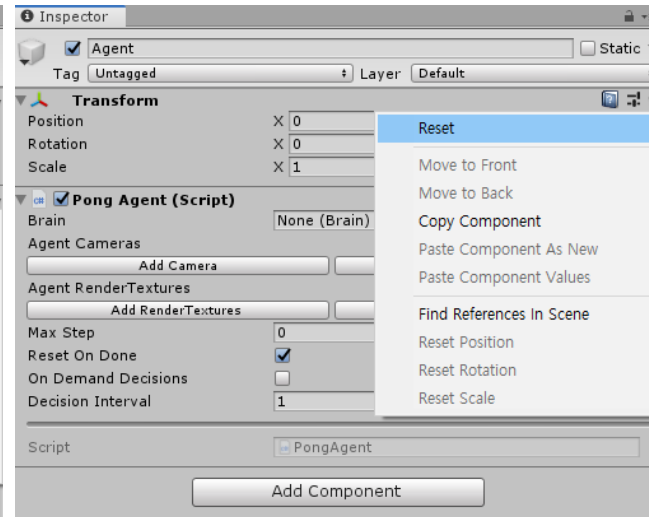
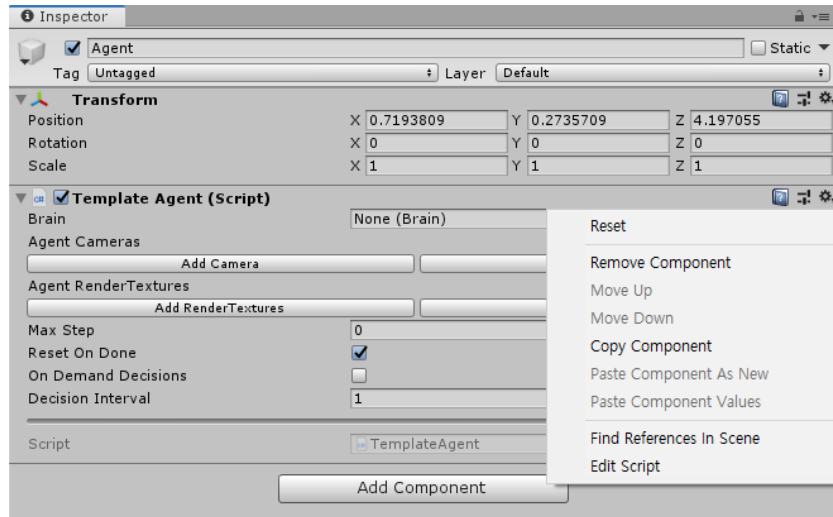
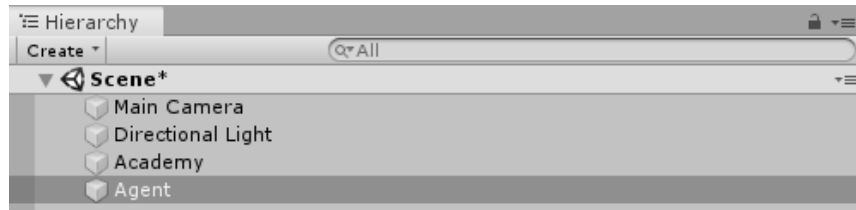
```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using MLAgents;
5
6 public class PongAgent : Agent {
```



Unity ML-agents 예제

Learning (External)

- Agent에 있던 스크립트 파일을 없애고 PongAgent로 변경 (Transform 리셋)

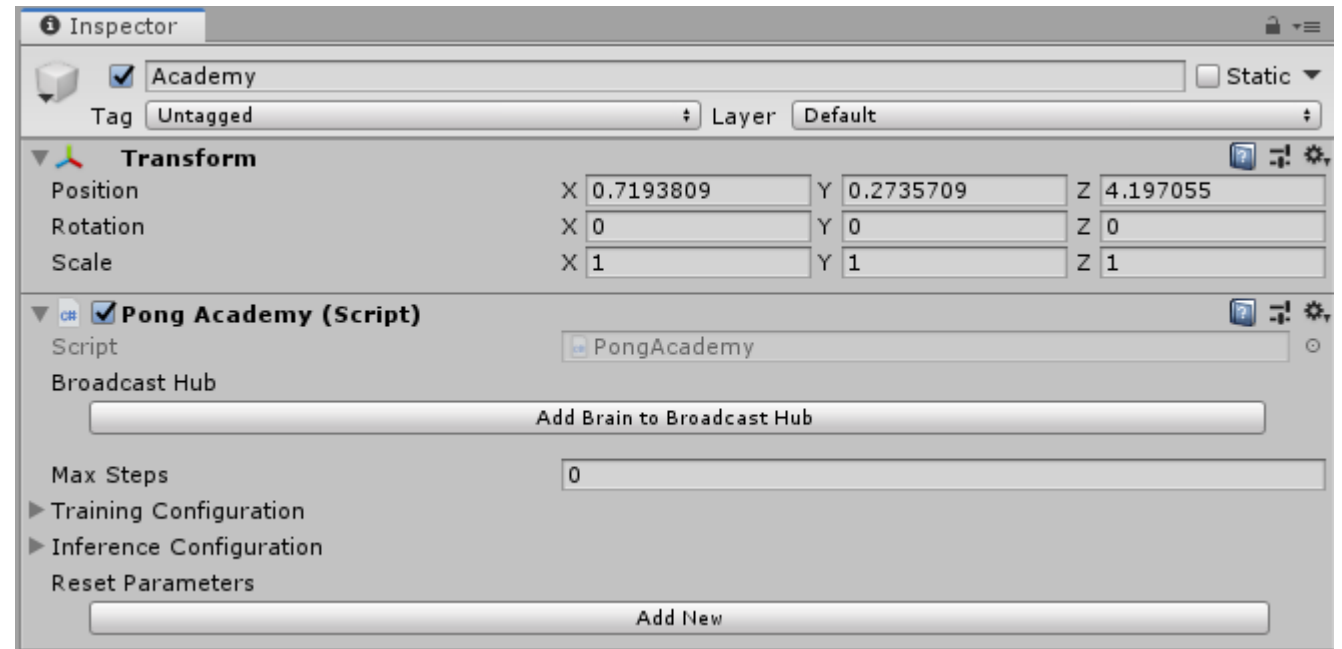
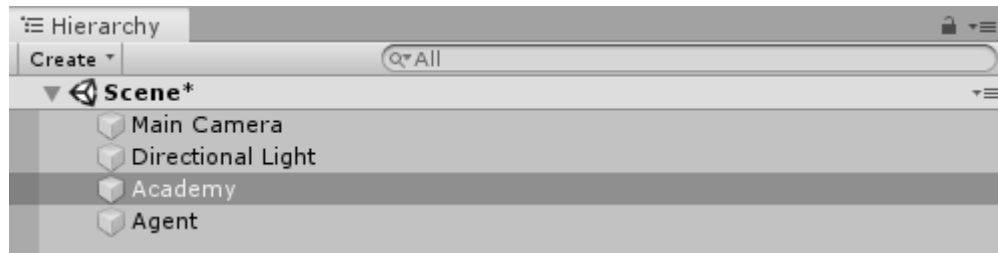


Unity ML-agents 예제



Learning (External)

- Academy도 스크립트 파일을 PongAcademy로 변경

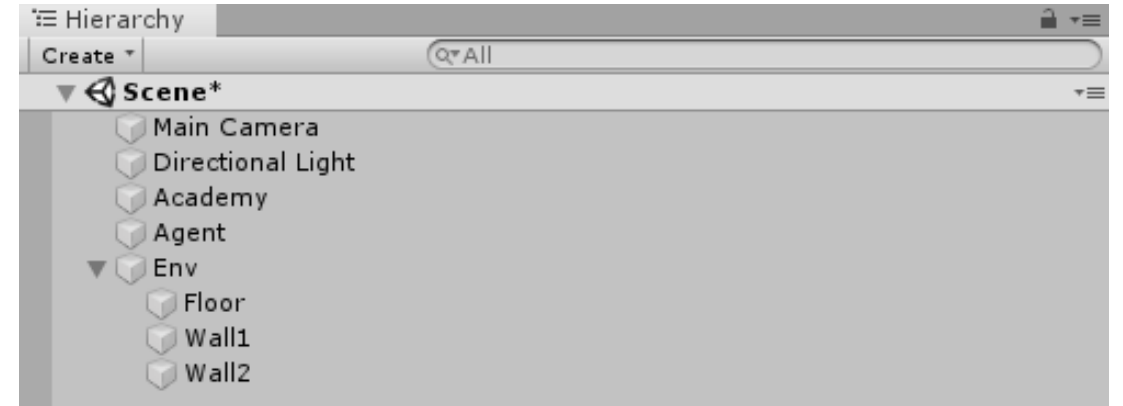
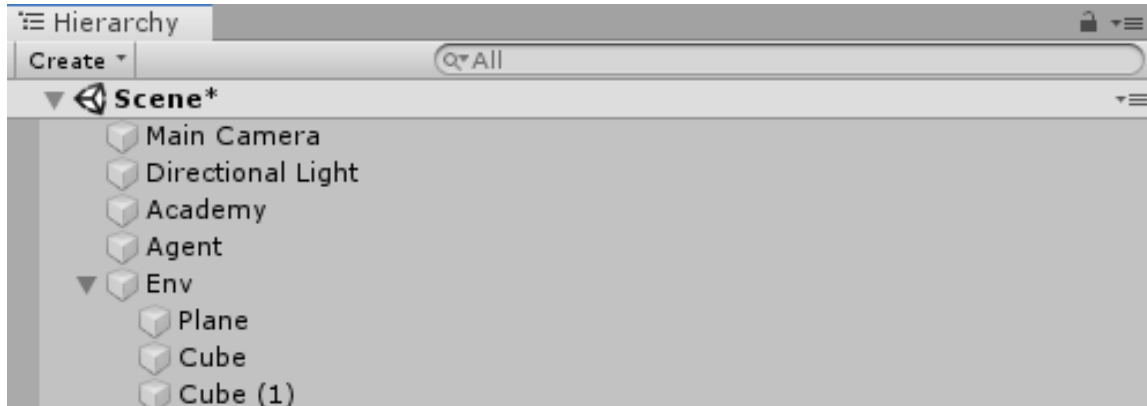


Unity ML-agents 예제



Learning (External)

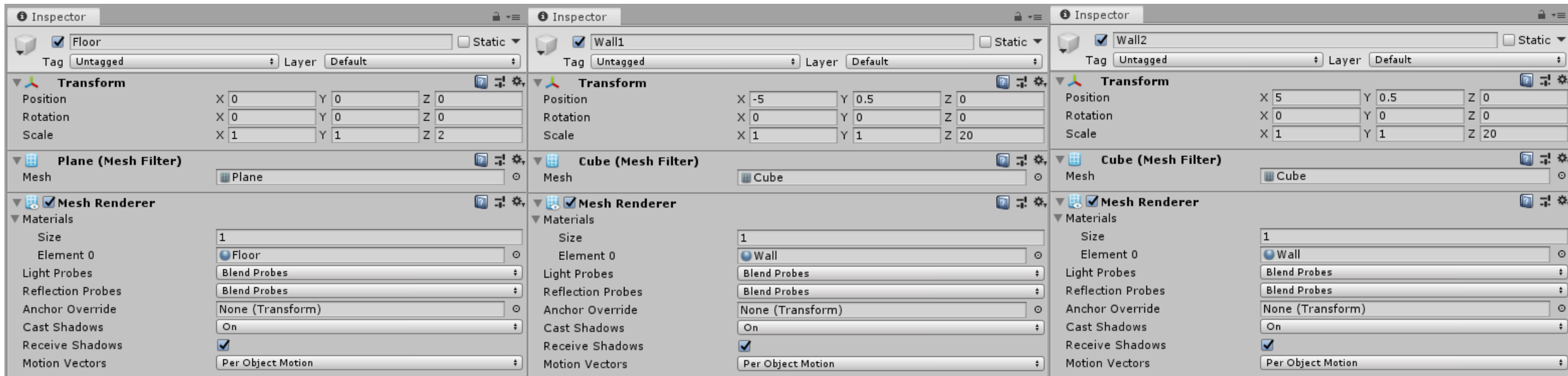
- 빈 오브젝트인 Env 생성 후 내부에 Plane 1개와 Cube 2개 생성 후 이름 변경
 - Plane -> Floor
 - Cube -> Wall1
 - Cube -> Wall2



Unity ML-agents 예제

Learning (External)

- 각 Object의 Inspector view 설정 (Transform, Mesh renderer -> element)

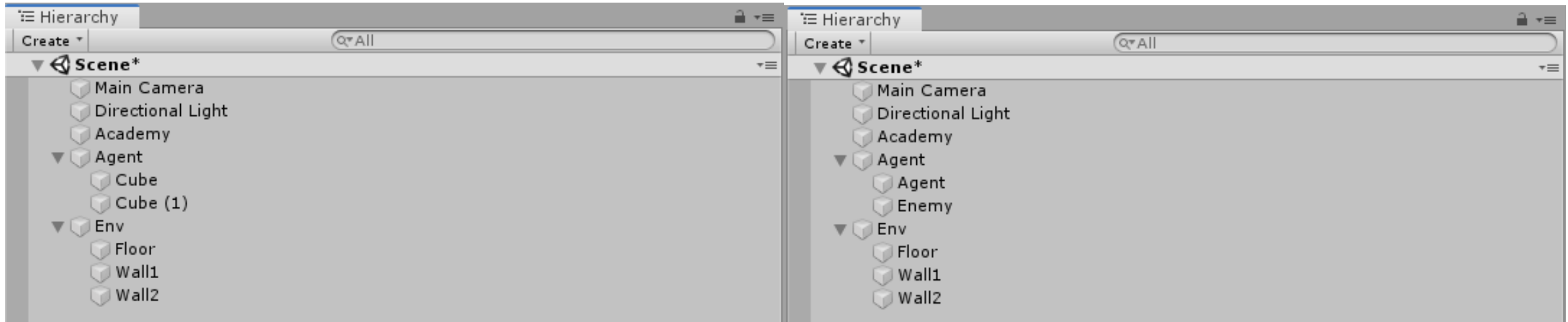


Unity ML-agents 예제



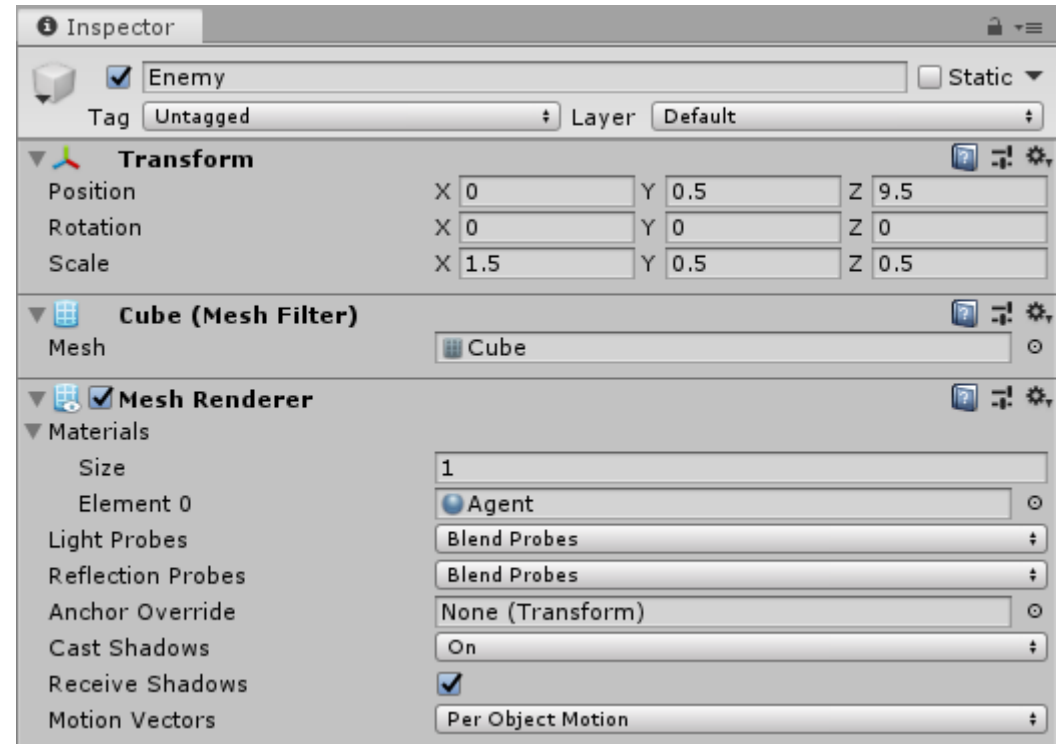
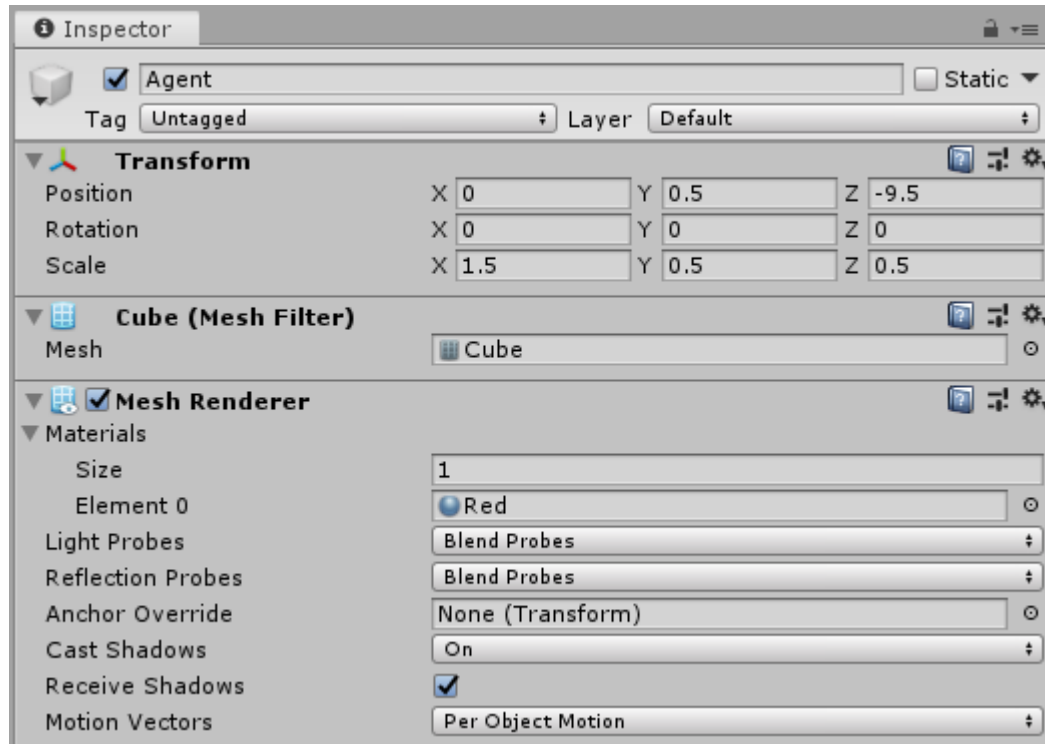
Learning (External)

- Agent 내부에 2개의 Cube 생성 후 이름 변경
 - Cube -> Agent
 - Cube -> Enemy



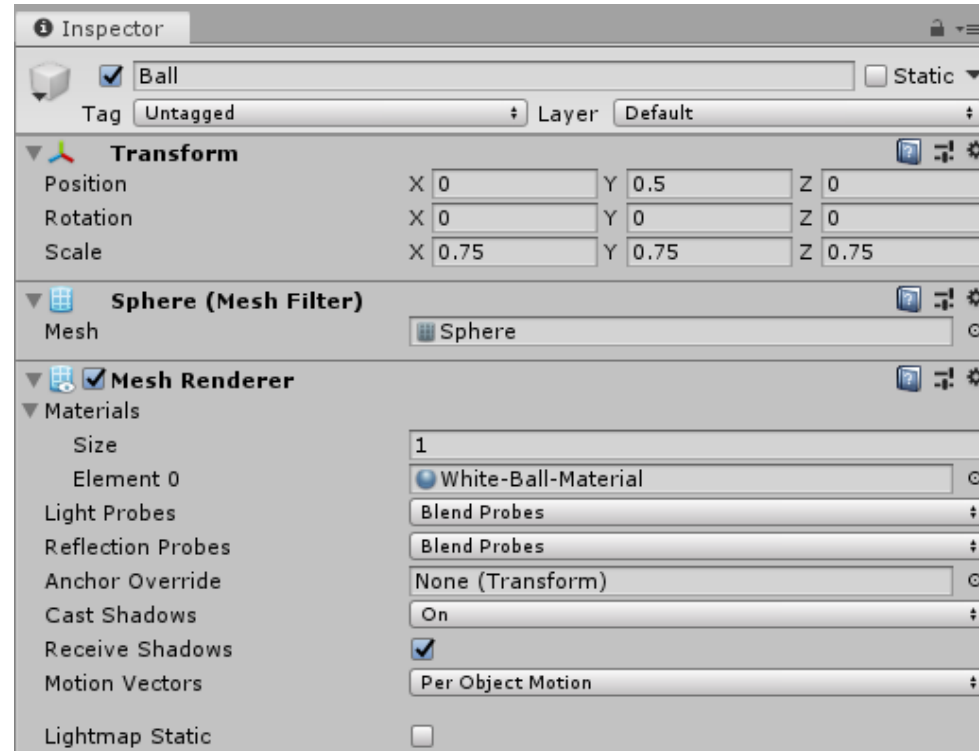
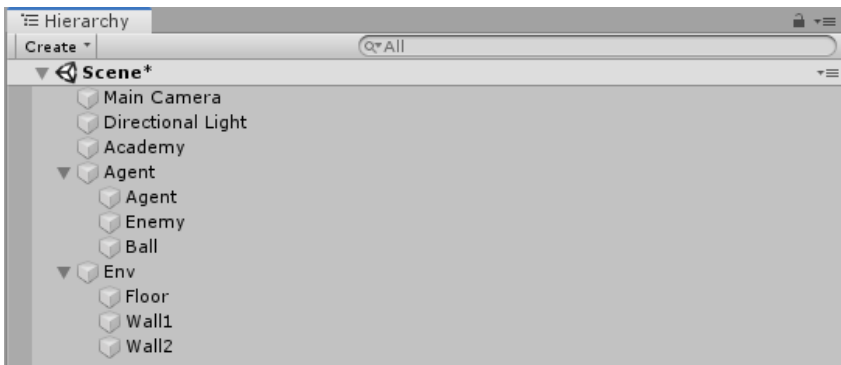
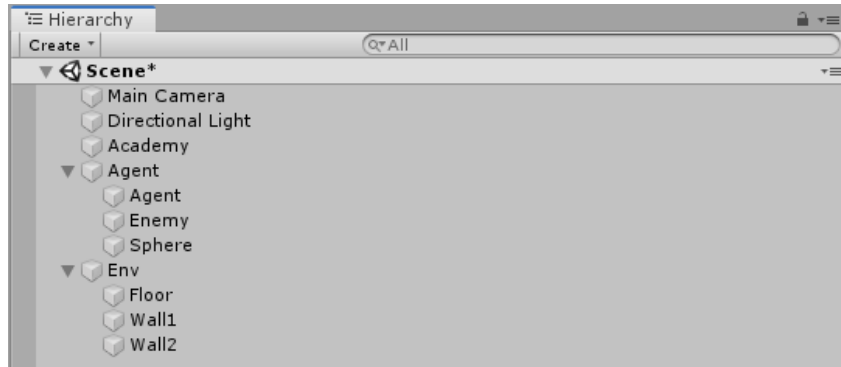
Learning (External)

- 2개의 오브젝트의 인스펙터뷰 설정 (Transform, Mesh Renderer -> Element)



Learning (External)

- Agent 내부에 Sphere 추가 -> 이름을 Ball로 변경 후 인스펙터뷰 수정 (Transform, Mesh Renderer -> Element

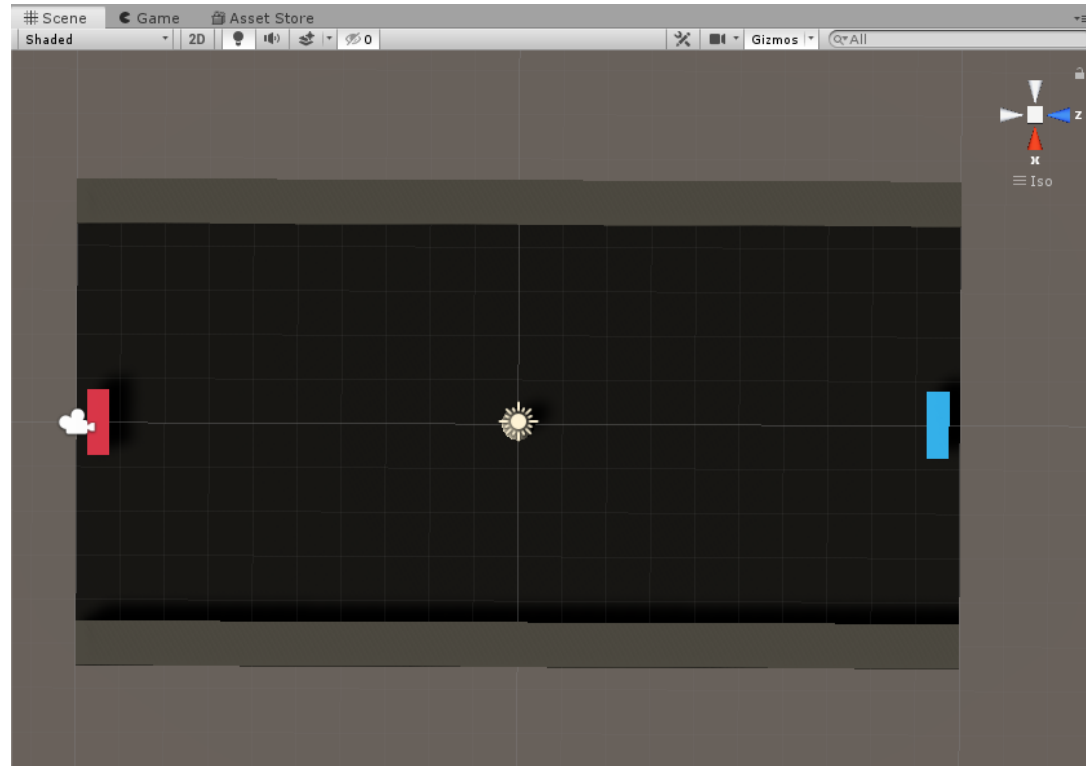


Unity ML-agents 예제



Learning (External)

- 환경 완성!!

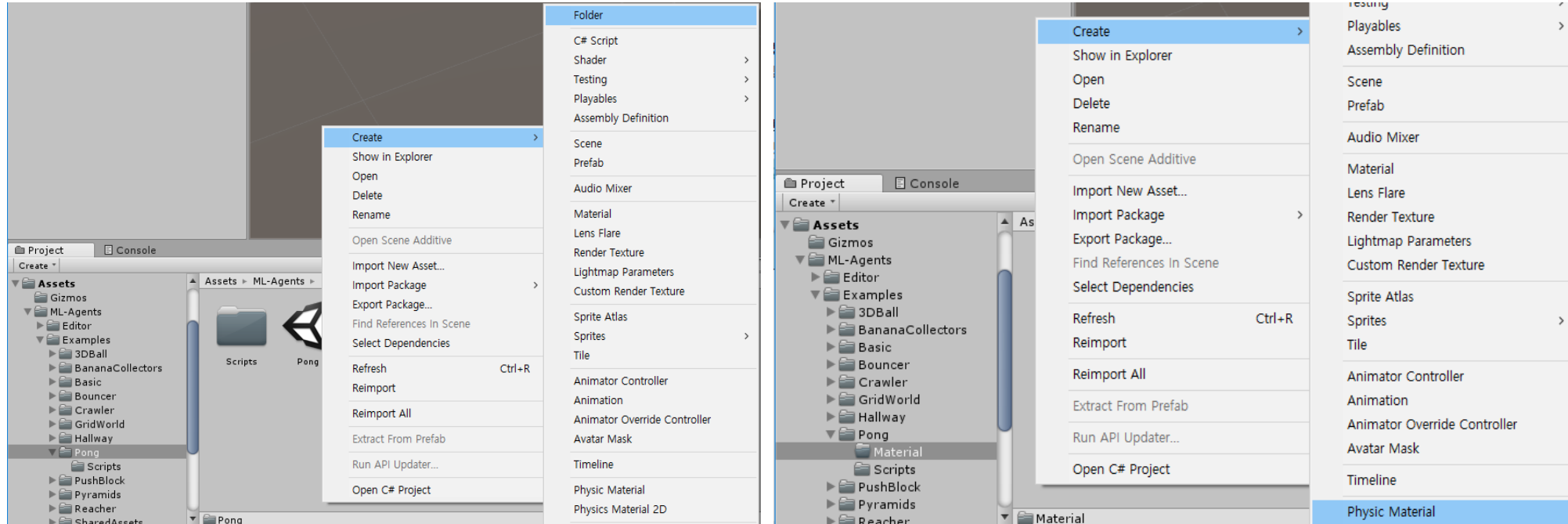
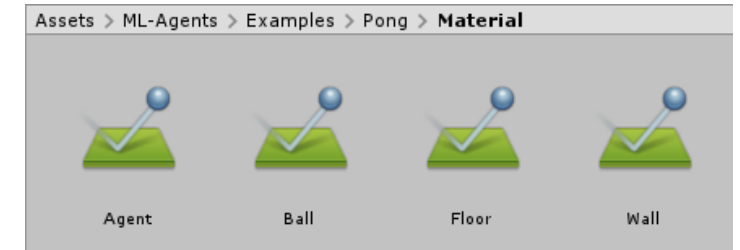


Unity ML-agents 예제



Learning (External)

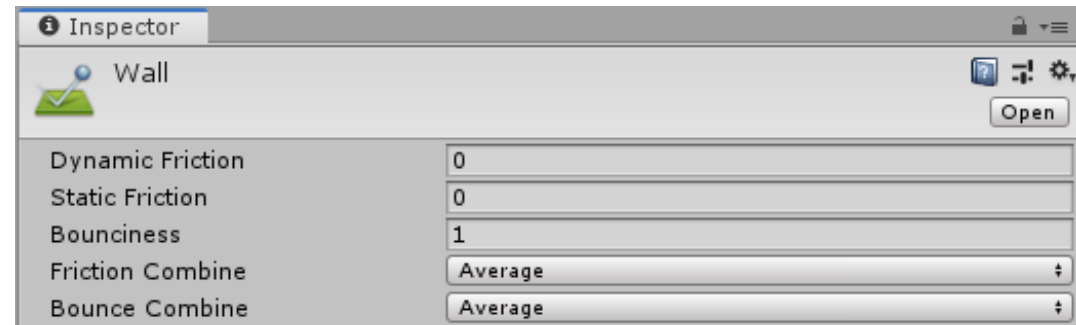
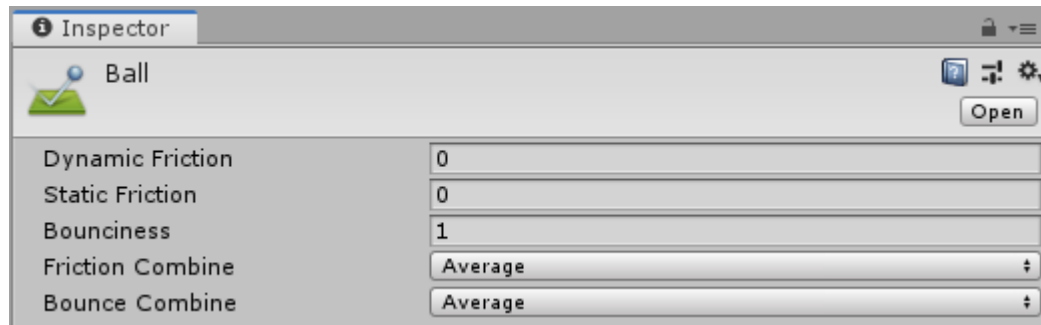
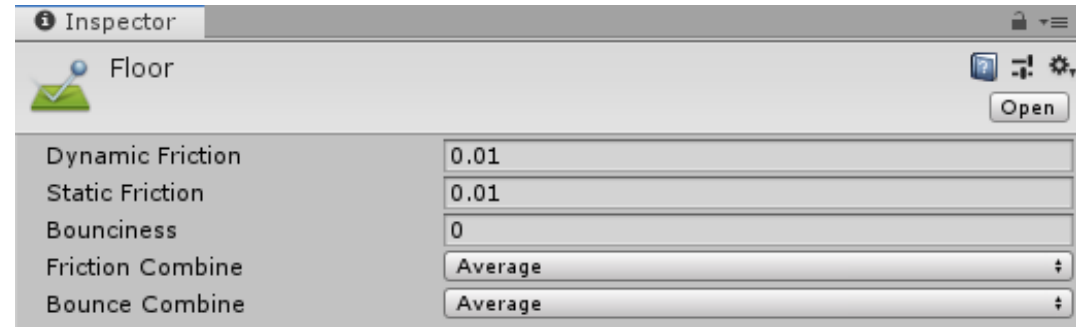
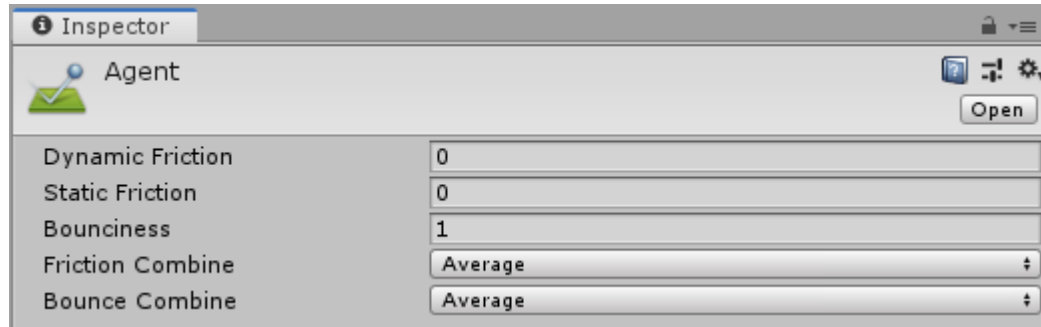
- 물리적 특성 부여 -> Pong 내부에 Material 폴더 생성 후 Physic Material 4개 생성 (Agent, Ball, Floor, Wall)



Unity ML-agents 예제

Learning (External)

- 각 Physic Material의 인스펙터뷰 설정

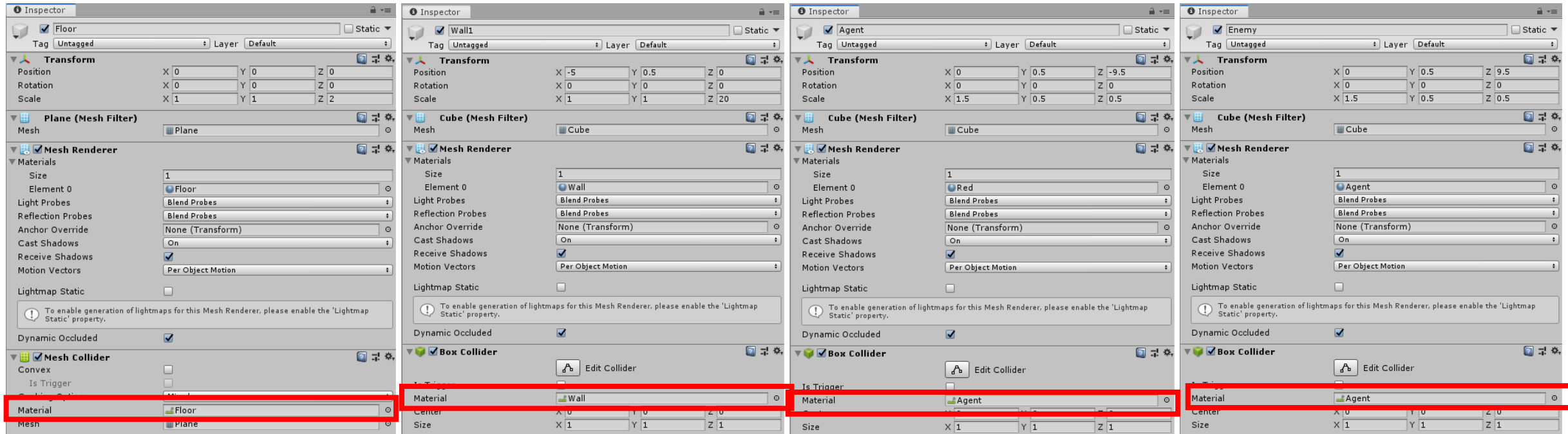


Unity ML-agents 예제



Learning (External)

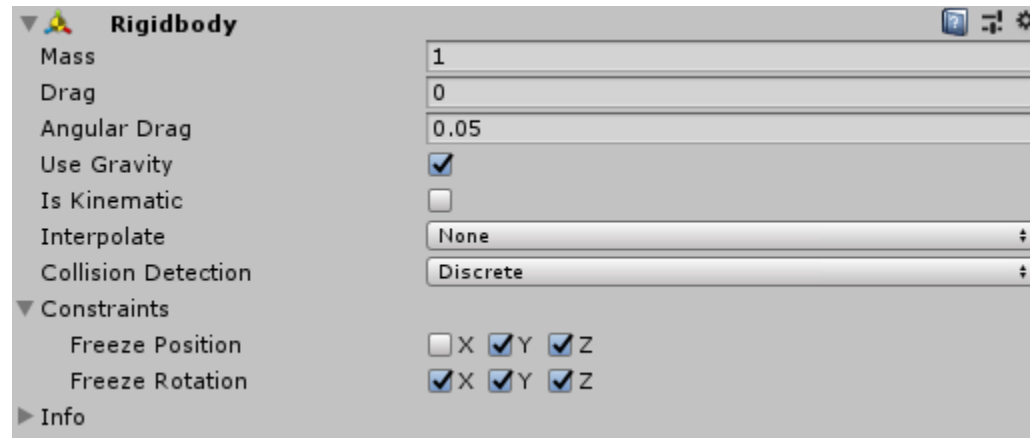
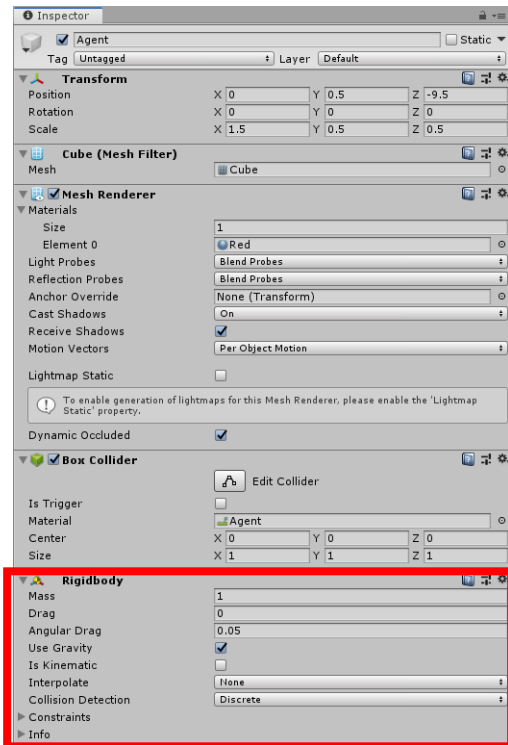
- 각 Physic Material을 각각 대응하는 오브젝트에 연결 (Mesh Collider -> Material) (Agent, Enemy, Wall1, Wall2, Ball)



Unity ML-agents 예제

Learning (External)

- Rigidbody 생성 및 Constraints 설정 (Agent)



Unity ML-agents 예제

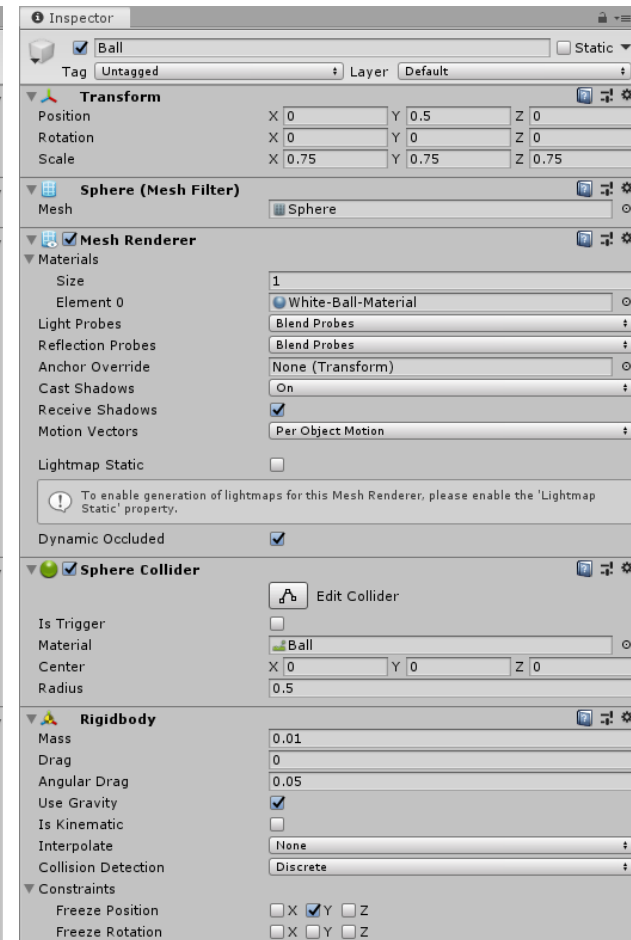
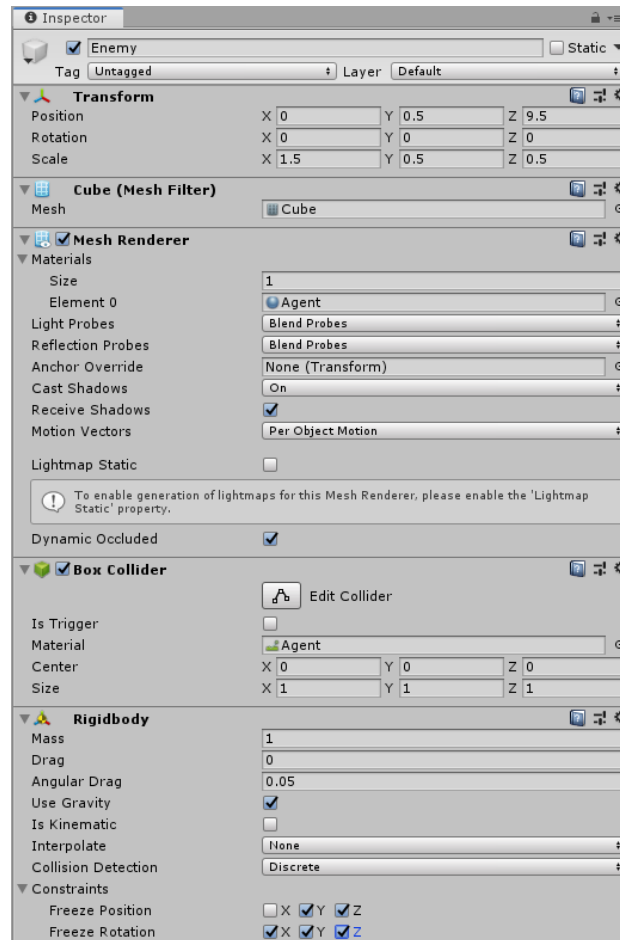
~ Reinforcement Learning Korea ~

RL KOREA



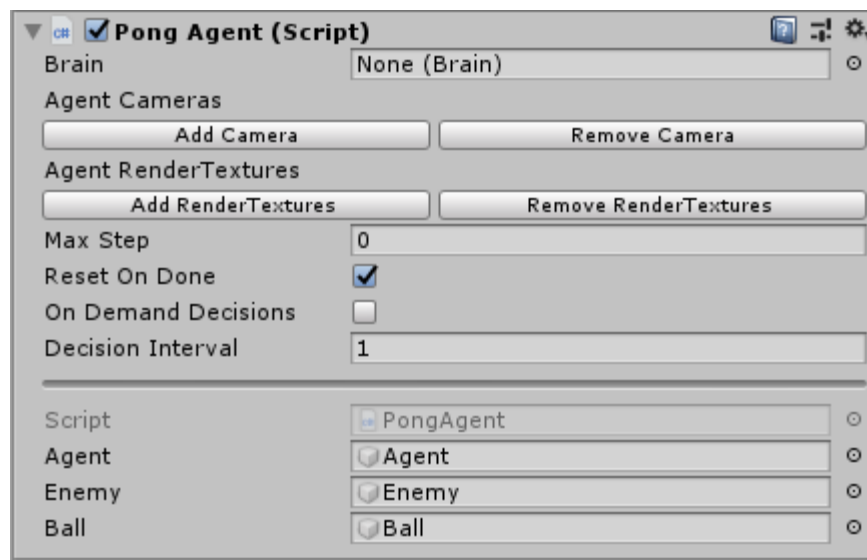
Learning (External)

- Rigidbody 생성 및 Constraints 설정 (Enemy, Ball)
- Ball의 mass는 0.01로 설정



Learning (External)

- PongAgent 스크립트
 - 파라미터 설정 및 오브젝트 선언



```
6 public class PongAgent : Agent {
7
8     public GameObject agent;
9     public GameObject enemy;
10    public GameObject ball;
11
12    private Rigidbody RbAgent;
13    private Rigidbody RbEnemy;
14    private Rigidbody RbBall;
15
16    private const int Stay = 0;
17    private const int Up = 1;
18    private const int Down = 2;
19
20    private Vector3 ResetPos;
21    private Vector3 velocity;
22
23    private float ball_vel_z = 0f;
24    private float ball_vel_z_old = 0f;
25
26    private float max_ball_speed = 10f;
27    private float min_ball_speed = 7f;
28
29    private Vector3 ResetPosBall;
30    private Vector3 ResetPosAgent;
31    private Vector3 ResetPosEnemy;
```

Learning (External)

- PongAgent 스크립트
 - InitializeAgent: 에이전트가 가장 처음 초기화 될때 한번 호출되는 함수
 - CollectObservations: 수치적 관측값들을 저장할 때 사용하는 함수

```
30 public override void InitializeAgent()  
31 {  
32     ResetPosBall = ball.transform.position;  
33     ResetPosAgent = agent.transform.position;  
34     ResetPosEnemy = enemy.transform.position;  
35  
36     RbAgent = agent.GetComponent<Rigidbody>();  
37     RbEnemy = enemy.GetComponent<Rigidbody>();  
38     RbBall = ball.GetComponent<Rigidbody>();  
39 }
```

```
41 public override void CollectObservations()  
42 {  
43     AddVectorObs(agent.transform.position.x);  
44     AddVectorObs(enemy.transform.position.x);  
45     AddVectorObs(ball.transform.position.x);  
46     AddVectorObs(ball.transform.position.z);  
47     AddVectorObs(RbAgent.velocity.x);  
48     AddVectorObs(RbAgent.velocity.z);  
49     AddVectorObs(RbEnemy.velocity.x);  
50     AddVectorObs(RbEnemy.velocity.z);  
51     AddVectorObs(RbBall.velocity.x);  
52     AddVectorObs(RbBall.velocity.z);  
53 }
```




Learning (External)

- PongAgent 스크립트
 - AgentAction 함수
 - 액션을 받아오고 이에 따라 에이전트를 행동하게 만드는 함수
 - 보상 및 게임 종료 조건 설정

```
58 public override void AgentAction(float[] vectorAction, string textAction)
59 {
60     int action = Mathf.FloorToInt(vectorAction[0]);
61     switch (action)
62     {
63         case Stay:
64             agent.transform.position = agent.transform.position + 0f * Vector3.right;
65             break;
66         case Up:
67             agent.transform.position = agent.transform.position + 0.3f * Vector3.left;
68             break;
69         case Down:
70             agent.transform.position = agent.transform.position + 0.3f * Vector3.right;
71             break;
72     }
73
74     enemy.transform.position = new Vector3(ball.transform.position.x, enemy.transform.position.y, enemy.transform.position.z);
75
76     if (ball.transform.position.z < -10)
77     {
78         AddReward(-1.0f);
79         Done();
80     }
81
82     if (ball.transform.position.z > 10)
83     {
84         AddReward(1.0f);
85         Done();
86     }
87
88     ball_vel_z = RbBall.velocity.z;
89
90     if (ball_vel_z > 0 && ball_vel_z_old < 0)
91     {
92         AddReward(0.5f);
93     }
94
95     ball_vel_z_old = ball_vel_z;
96
97     AddReward(0f);
98 }
```

Learning (External)

- PongAgent 스크립트
 - AgentAction 함수
 - 액션 선택에 따라 에이전트 이동 및 적 이동

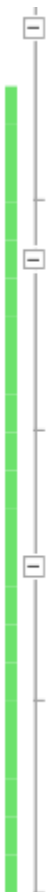
```
58 public override void AgentAction(float[] vectorAction, string textAction)
59 {
60     int action = Mathf.FloorToInt(vectorAction[0]);
61     switch (action)
62     {
63         case Stay:
64             agent.transform.position = agent.transform.position + 0f * Vector3.right;
65             break;
66         case Up:
67             agent.transform.position = agent.transform.position + 0.3f * Vector3.left;
68             break;
69         case Down:
70             agent.transform.position = agent.transform.position + 0.3f * Vector3.right;
71             break;
72     }
73
74     enemy.transform.position = new Vector3(ball.transform.position.x, enemy.transform.position.y, enemy.transform.position.z);
```



Learning (External)

- PongAgent 스크립트
 - AgentAction 함수
 - 보상 및 게임 종료 조건 설정

76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98



```
if (ball.transform.position.z < -10)
{
    AddReward(-1.0f);
    Done();
}

if (ball.transform.position.z > 10)
{
    AddReward(1.0f);
    Done();
}

ball_vel_z = RbBall.velocity.z;

if (ball_vel_z > 0 && ball_vel_z_old < 0)
{
    AddReward(0.5f);
}

ball_vel_z_old = ball_vel_z;

AddReward(0f);
}
```



Learning (External)

- PongAgent 스크립트

- AgentReset 함수: 게임이 새로 시작될 때 호출되는 코드

```
100 public override void AgentReset()
101 {
102     ball.transform.position = ResetPosBall;
103     agent.transform.position = ResetPosAgent;
104     enemy.transform.position = ResetPosEnemy;
105     RbBall.velocity = Vector3.zero;
106     ball.transform.rotation = Quaternion.identity;
107     RbAgent.velocity = Vector3.zero;
108     RbAgent.angularVelocity = Vector3.zero;
109     RbEnemy.velocity = Vector3.zero;
110     RbEnemy.angularVelocity = Vector3.zero;
111
112     float rand_num = Random.Range(-1f, 1f);
113
114     if (rand_num < -0.5f)
115     {
116         // 우상단으로 공을 움직입니다.
117         velocity = new Vector3(Random.Range(min_ball_speed, max_ball_speed), 0, Random.Range(min_ball_speed, max_ball_speed));
118     }
119     else if (rand_num < 0f)
120     {
121         // 우하단으로 공을 움직입니다.
122         velocity = new Vector3(Random.Range(min_ball_speed, max_ball_speed), 0, Random.Range(-max_ball_speed, -min_ball_speed));
123     }
124     else if (rand_num < 0.5f)
125     {
126         // 좌상단으로 공을 움직입니다.
127         velocity = new Vector3(Random.Range(-max_ball_speed, -min_ball_speed), 0, Random.Range(min_ball_speed, max_ball_speed));
128     }
129     else
130     {
131         // 좌하단으로 공을 움직입니다.
132         velocity = new Vector3(Random.Range(-max_ball_speed, -min_ball_speed), 0, Random.Range(-max_ball_speed, -min_ball_speed));
133     }
134     RbBall.AddForce(velocity);
135 }
```

Learning (External)

- PongAgent 스크립트
 - AgentReset 함수
 - 에이전트, 적, 공의 위치 및 속도 초기화

```
100 public override void AgentReset()  
101 {  
102     ball.transform.position = ResetPosBall;  
103     agent.transform.position = ResetPosAgent;  
104     enemy.transform.position = ResetPosEnemy;  
105     RbBall.velocity = Vector3.zero;  
106     ball.transform.rotation = Quaternion.identity;  
107     RbAgent.velocity = Vector3.zero;  
108     RbAgent.angularVelocity = Vector3.zero;  
109     RbEnemy.velocity = Vector3.zero;  
110     RbEnemy.angularVelocity = Vector3.zero;  
111 }
```



Learning (External)

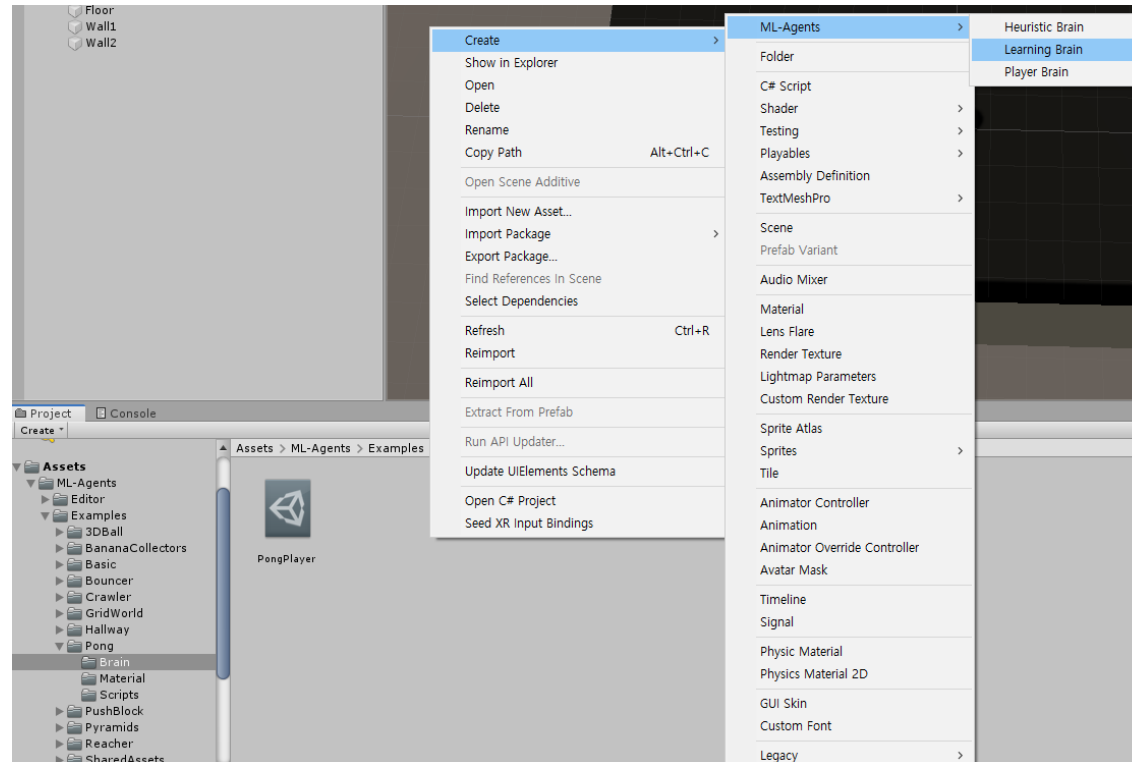
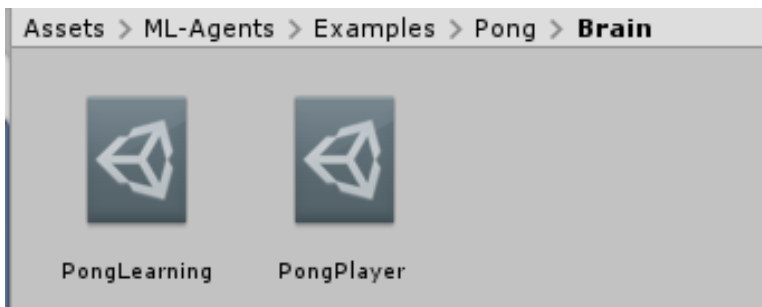
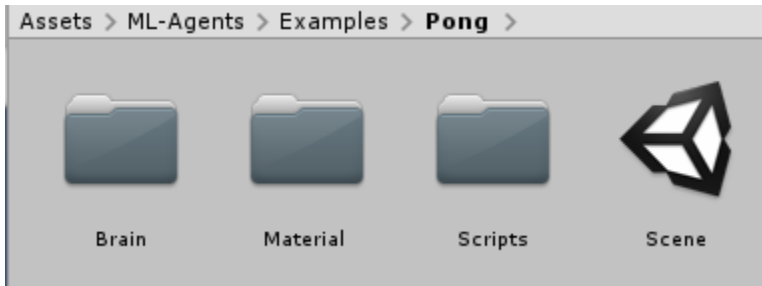
- PongAgent 스크립트
 - AgentReset 함수
 - 초기 공의 방향 및 속도를 임의로 설정

```
114     if (rand_num < -0.5f)
115     {
116         // 우상단으로 공을 움직입니다.
117         velocity = new Vector3(Random.Range(min_ball_speed, max_ball_speed), 0, Random.Range(min_ball_speed, max_ball_speed));
118     }
119     else if (rand_num < 0f)
120     {
121         // 우하단으로 공을 움직입니다.
122         velocity = new Vector3(Random.Range(min_ball_speed, max_ball_speed), 0, Random.Range(-max_ball_speed, -min_ball_speed));
123     }
124     else if (rand_num < 0.5f)
125     {
126         // 좌상단으로 공을 움직입니다.
127         velocity = new Vector3(Random.Range(-max_ball_speed, -min_ball_speed), 0, Random.Range(min_ball_speed, max_ball_speed));
128     }
129     else
130     {
131         // 좌하단으로 공을 움직입니다.
132         velocity = new Vector3(Random.Range(-max_ball_speed, -min_ball_speed), 0, Random.Range(-max_ball_speed, -min_ball_speed));
133     }
134     RbBall.AddForce(velocity);
135 }
```

Unity ML-agents 예제

Learning (External)

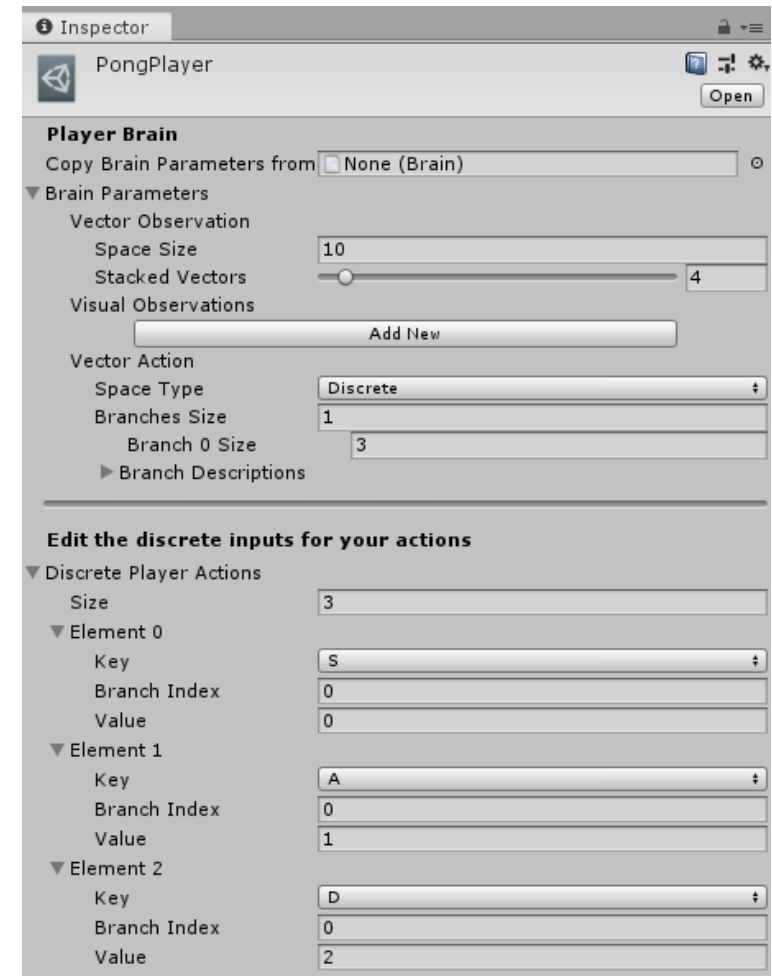
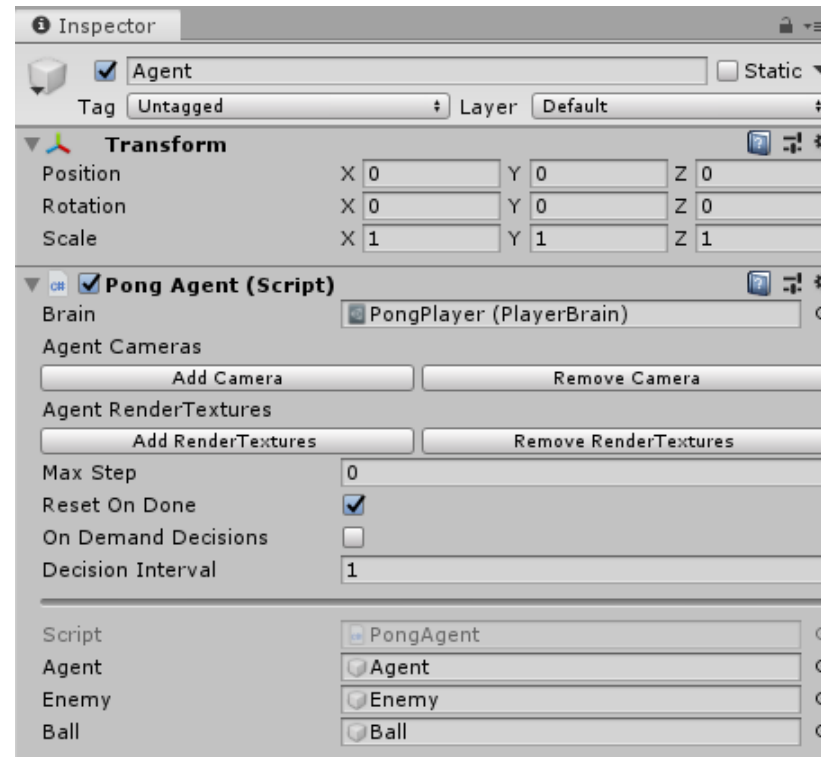
- Pong -> Brain 폴더 생성 및 Learning Brain, Player Brain 생성 (마우스 오른쪽 클릭 -> ML-Agents)



Unity ML-agents 예제

Learning (External)

- Player Brain 설정 및 Agent에 연결 (Agent, Enemy, Ball에 오브젝트 연결)



Unity ML-agents 예제

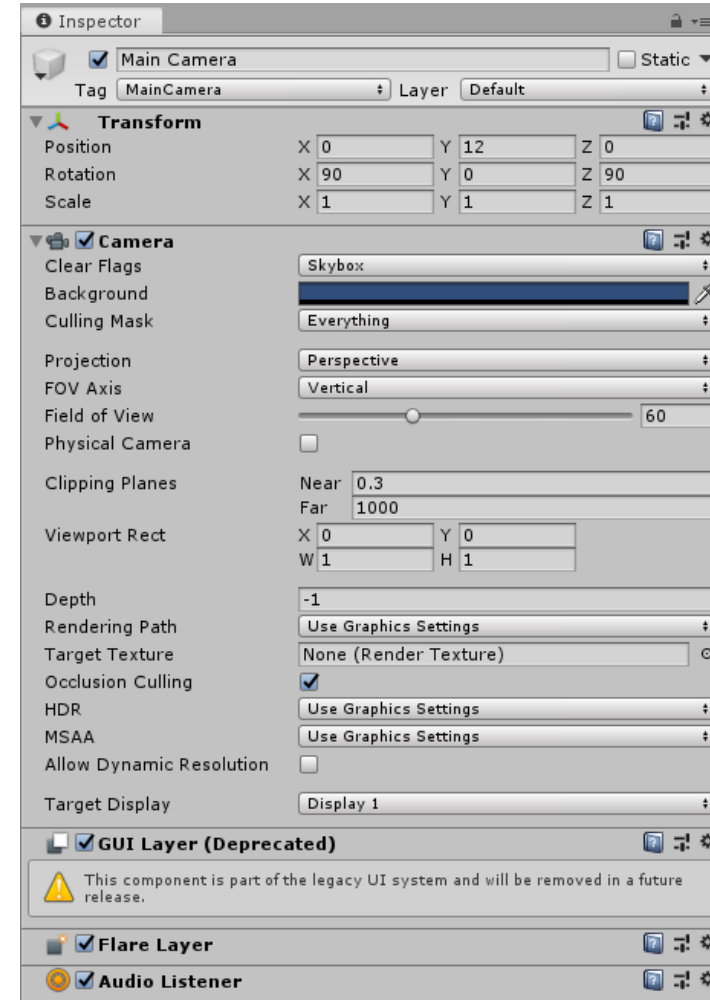
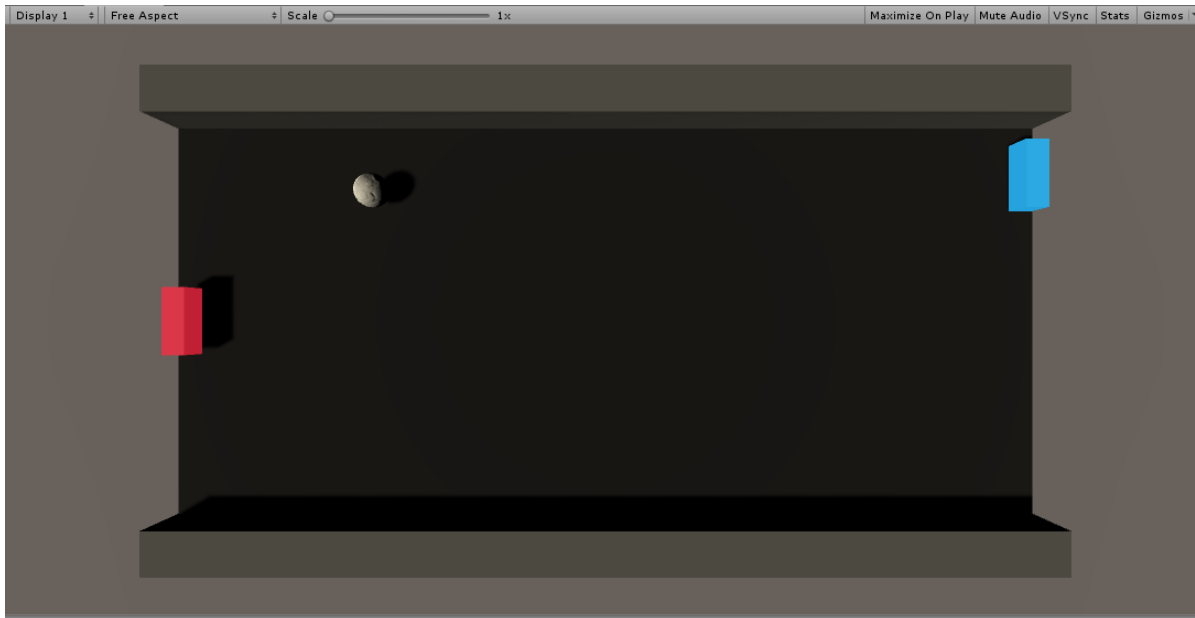
~ Reinforcement Learning Korea ~

RL KOREA



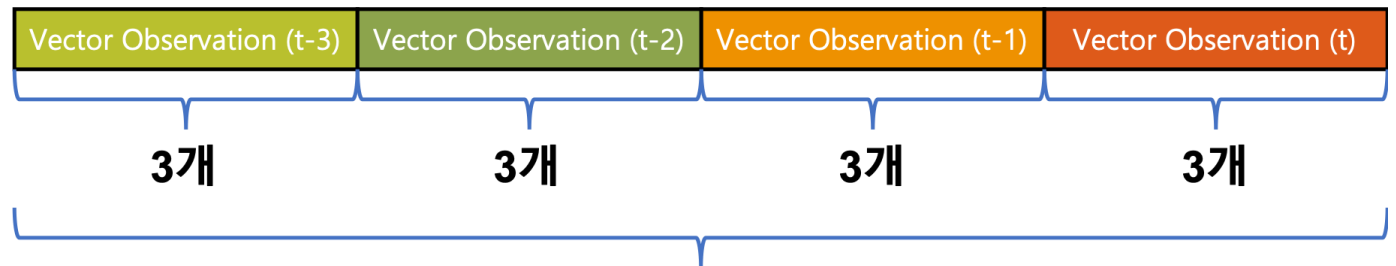
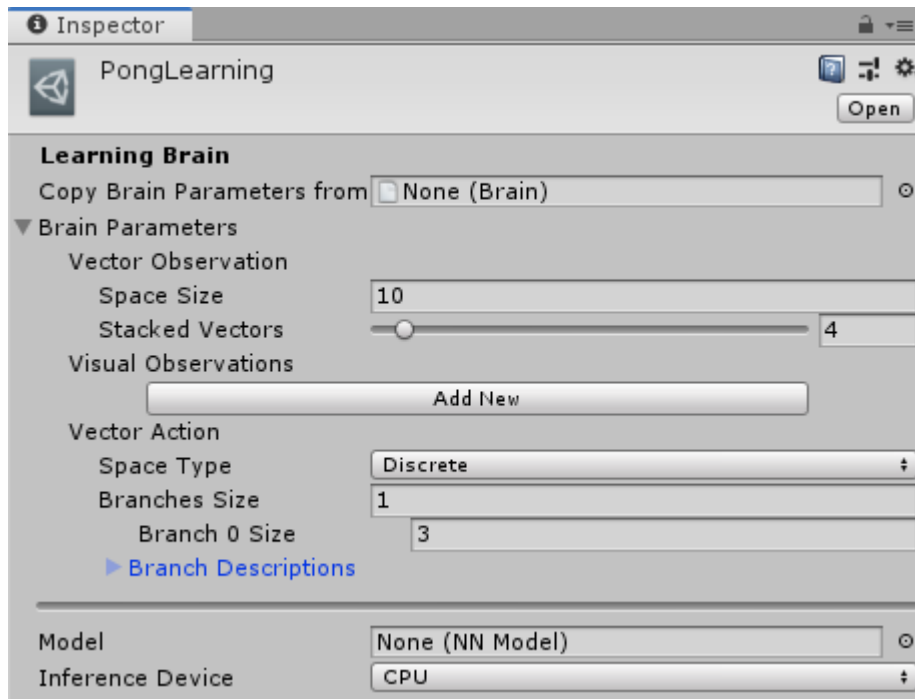
Learning (External)

- Main 카메라 위치 변경 (인스펙터뷰 설정 변경)

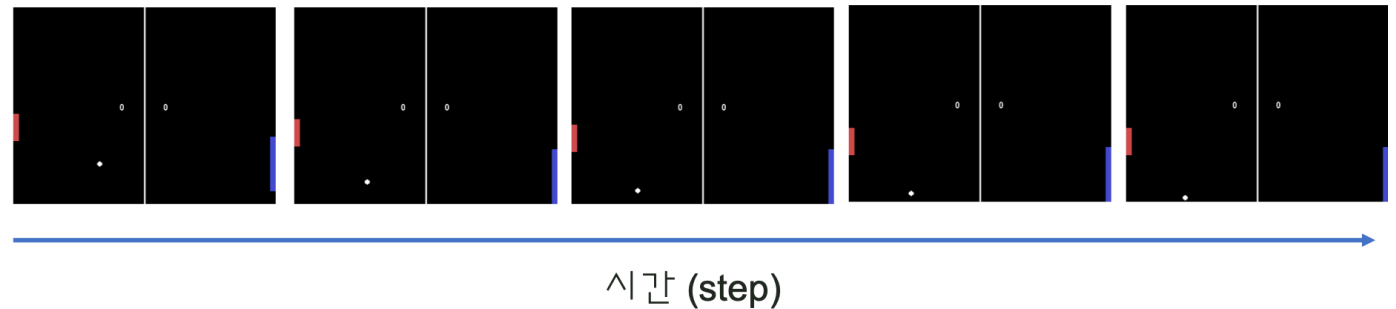


Learning (External)

- Learning Brain 설정



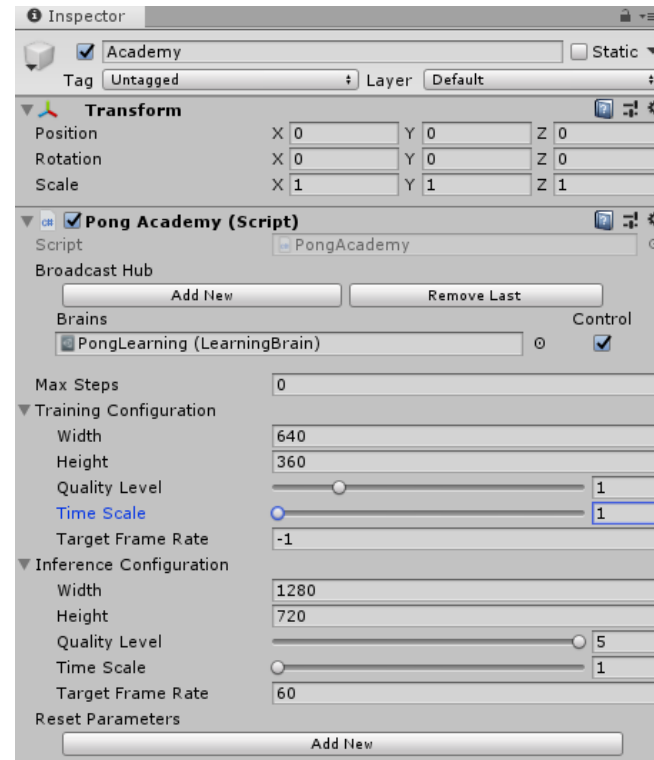
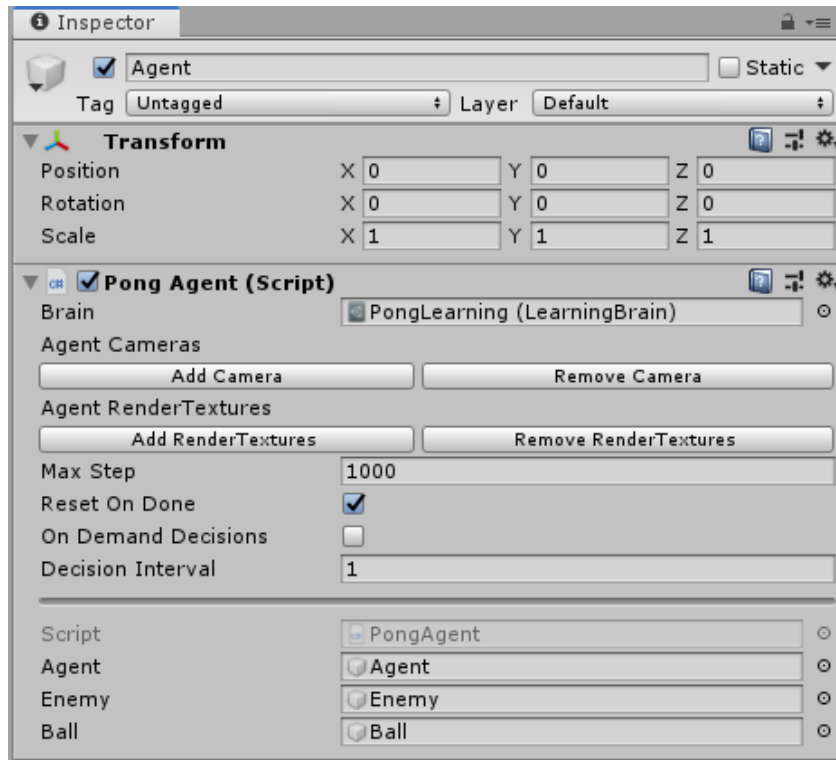
총 12개의 데이터를 하나의 수치적 관측으로!



Unity ML-agents 예제

Learning (External)

- Learning Brain을 Agent에 연결 및 Academy의 Brains에 추가 후 Academy 설정 변경

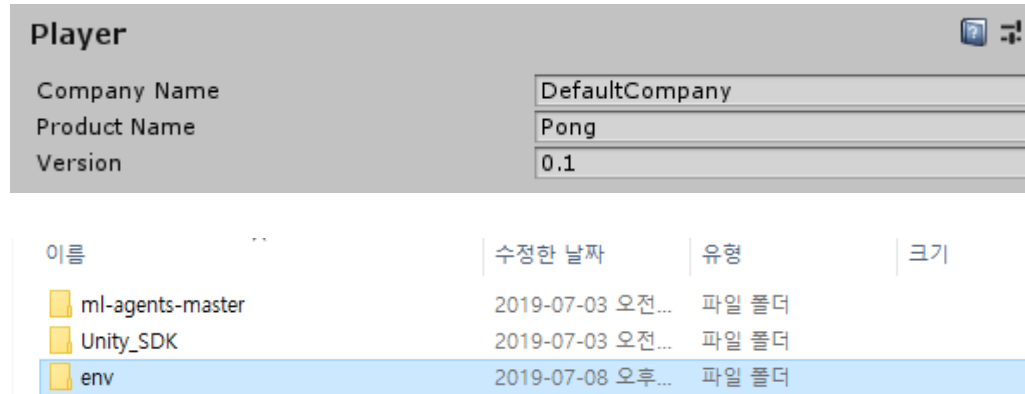
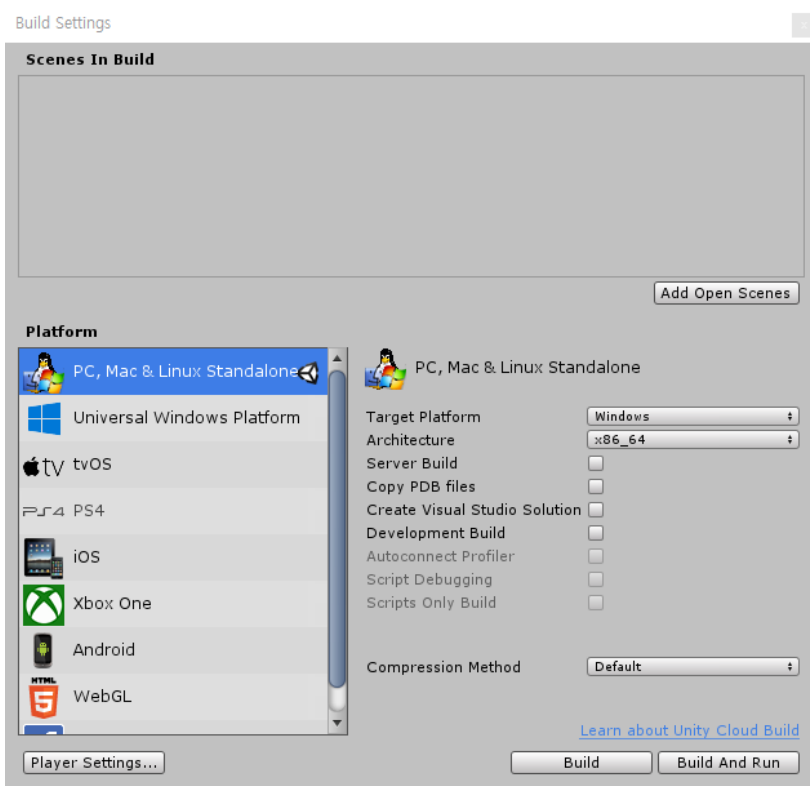


Unity ML-agents 예제



Learning (External)

- 환경 빌드

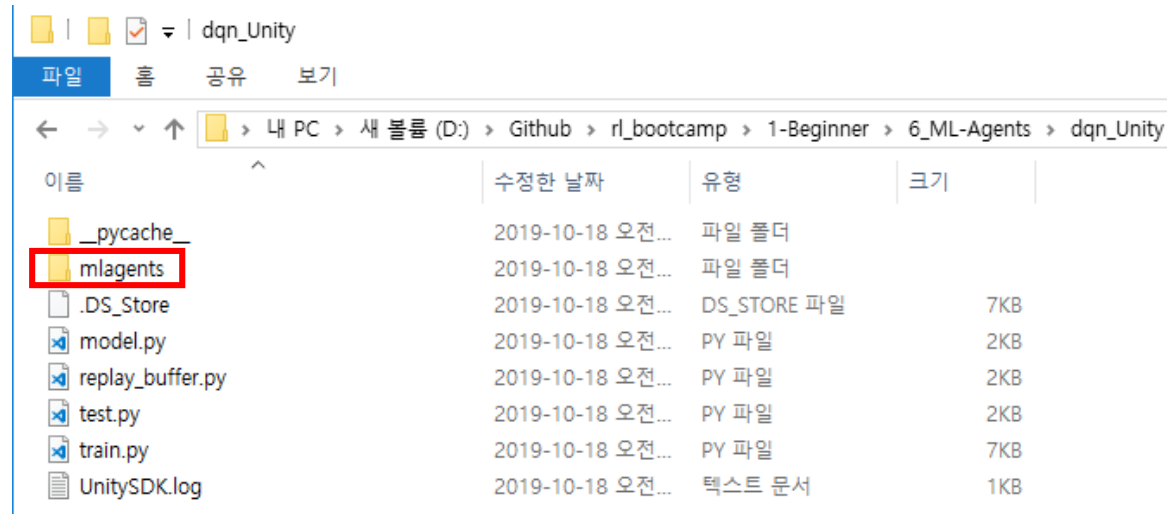
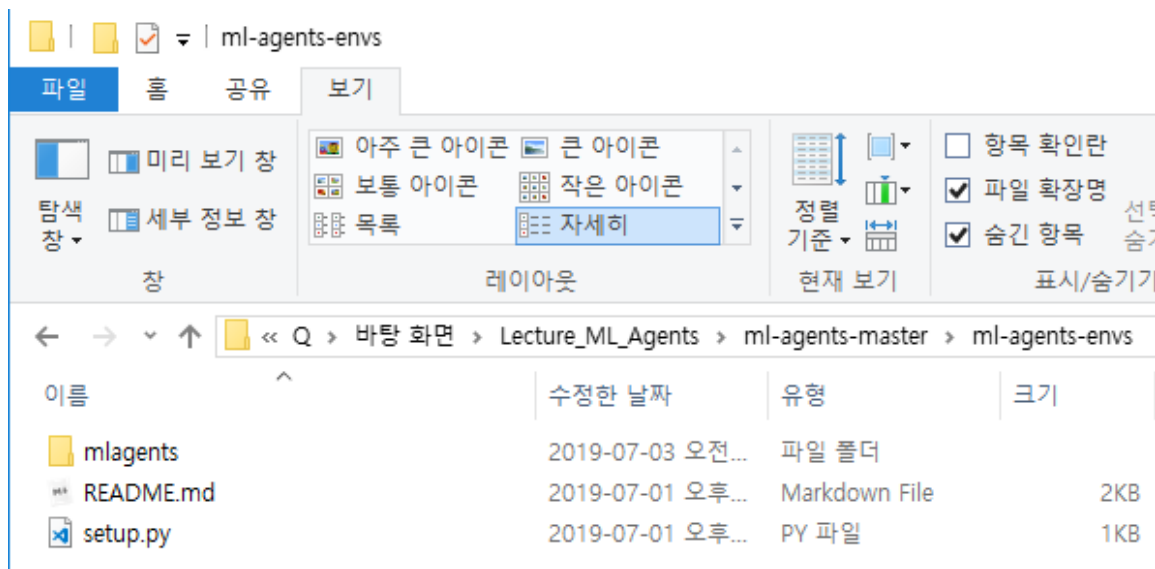


이름	수정한 날짜	유형	크기
ml-agents-master	2019-07-03 오전...	파일 폴더	
Unity_SDK	2019-07-03 오전...	파일 폴더	
env	2019-07-08 오후...	파일 폴더	

Unity ML-agents 예제

Learning (External)

- ML-agents 깃허브 폴더 -> ml-agents-env -> mlagents를 알고리즘과 동일한 폴더로 복사



Unity ML-agents 예제

~ Reinforcement Learning Korea ~

RL KOREA



DQN 코드 변경 (train.py)

```
1 import os
2 import gym
3 import time
4 import argparse
5 from collections import deque
```



```
1 import os
2 import time
3 import argparse
4 from collections import deque
5 from mlagents.envs import UnityEnvironment
```

Unity ML-agents 예제

~ Reinforcement Learning Korea ~

RL KOREA



DQN 코드 변경 (train.py)

```
16 parser = argparse.ArgumentParser()
17 parser.add_argument('--training_eps', type=int, default=500)
18 parser.add_argument('--threshold_return', type=int, default=495)
19 parser.add_argument('--render', action="store_true", default=False)
20 parser.add_argument('--gamma', type=float, default=0.99)
21 parser.add_argument('--epsilon', type=float, default=1.0)
22 parser.add_argument('--epsilon_decay', type=float, default=0.995)
23 parser.add_argument('--buffer_size', type=int, default=10000)
24 parser.add_argument('--batch_size', type=int, default=64)
25 parser.add_argument('--target_update_period', type=int, default=100)
26 args = parser.parse_args()
27 device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
```



```
16 parser = argparse.ArgumentParser()
17 parser.add_argument('--episode_num', type=int, default=500)
18 parser.add_argument('--threshold_return', type=int, default=495)
19 parser.add_argument('--gamma', type=float, default=0.99)
20 parser.add_argument('--epsilon', type=float, default=1.0)
21 parser.add_argument('--epsilon_decay', type=float, default=0.995)
22 parser.add_argument('--buffer_size', type=int, default=10000)
23 parser.add_argument('--batch_size', type=int, default=64)
24 parser.add_argument('--target_update_steps', type=int, default=100)
25 args = parser.parse_args()
26 device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
```

Unity ML-agents 예제

~ Reinforcement Learning Korea ~

RL KOREA



DQN 코드 변경 (train.py)

```
78 def main():
79     # Initialize environment
80     env = gym.make('CartPole-v1')
81     obs_dim = env.observation_space.shape[0]
82     act_num = env.action_space.n
83     print('State dimension:', obs_dim)
84     print('Action number:', act_num)
85
86     # Set a random seed
87     env.seed(0)
88     np.random.seed(0)
89     torch.manual_seed(0)
```



```
77 def main():
78     # Initialize environment
79     env = UnityEnvironment(file_name='../env/Pong/Pong')
80
81     default_brain = env.brain_names[0]
82     brain = env.brains[default_brain]
83
84     env_info = env.reset(train_mode=True)[default_brain]
85
86     obs_dim = env_info.vector_observations[0].shape[0]
87     act_num = brain.vector_action_space_size[0]
88
89     print('State dimension:', obs_dim)
90     print('Action number:', act_num)
91
92     # Set a random seed
93     np.random.seed(0)
94     torch.manual_seed(0)
```


Unity ML-agents 예제

DQN 코드 변경 (train.py)

```
123     while not done:
124         if args.render:
125             env.render()
126
127         step_count += 1
128
129         # Collect experience (s, a, r, s') using some policy
130         action = select_action(torch.Tensor(obs).to(device), act_num, qf)
131         next_obs, reward, done, _ = env.step(action)
```



```
128     while not done:
129         step_count += 1
130
131         # Collect experience (s, a, r, s') using some policy
132         action = select_action(torch.Tensor(obs).to(device), act_num, qf)
133
134         env_info = env.step(int(action))[default_brain]
135
136         next_obs = env_info.vector_observations[0]
137         reward = env_info.rewards[0]
138         done = env_info.local_done[0]
```

Unity ML-agents 예제

~ Reinforcement Learning Korea ~

RL KOREA



DQN 코드 변경 (train.py)

```
173  
174  if __name__ == '__main__':  
175      main()
```



```
180  
181      env.close()  
182  
183  if __name__ == '__main__':  
184      main()
```

Unity ML-agents 예제

~ Reinforcement Learning Korea ~

RL KOREA



DQN 코드 변경 (test.py)

```
1 import os
2 import gym
3 import argparse
4 import numpy as np
5 import torch
6 from model import MLP
```

```
8 # Configurations
9 parser = argparse.ArgumentParser()
10 parser.add_argument('--load', type=str, default=None,
11                     help='load the saved model')
12 parser.add_argument('--render', action="store_true", default=True,
13                     help='if you want to render, set this to True')
14 args = parser.parse_args()
```



```
1 import os
2 import argparse
3 import numpy as np
4 import torch
5 from model import MLP
6 from mlagents.envs import UnityEnvironment
```

```
8 # Configurations
9 parser = argparse.ArgumentParser()
10 parser.add_argument('--load', type=str, default=None,
11                     help='load the saved model')
12 args = parser.parse_args()
```

DQN 코드 변경 (test.py)

```
18 def main():
19     env = gym.make('CartPole-v1')
20     obs_dim = env.observation_space.shape[0]
21     act_num = env.action_space.n
```



```
16 def main():
17     env = UnityEnvironment(file_name='../env/Pong/Pong')
18
19     default_brain = env.brain_names[0]
20     brain = env.brains[default_brain]
21
22     env_info = env.reset(train_mode=False)[default_brain]
23
24     obs_dim = env_info.vector_observations[0].shape[0]
25     act_num = brain.vector_action_space_size[0]
```

Unity ML-agents 예제

DQN 코드 변경 (test.py)

```
33     for episode in range(1, 10001):
34         total_reward = 0.
35
36         obs = env.reset()
37         done = False
38
39         while not done:
40             if args.render:
41                 env.render()
42
43             action = mlp(torch.Tensor(obs).to(device)).argmax().detach().cpu().numpy()
44             next_obs, reward, done, _ = env.step(action)
```



```
37     for episode in range(1, 10001):
38         total_reward = 0.
39
40         obs = env_info.vector_observations[0]
41         done = False
42
43         while not done:
44             action = mlp(torch.Tensor(obs).to(device)).argmax().detach().cpu().numpy()
45             env_info = env.step(int(action))[default_brain]
46
47             next_obs = env_info.vector_observations[0]
48             reward = env_info.rewards[0]
49             done = env_info.local_done[0]
```

