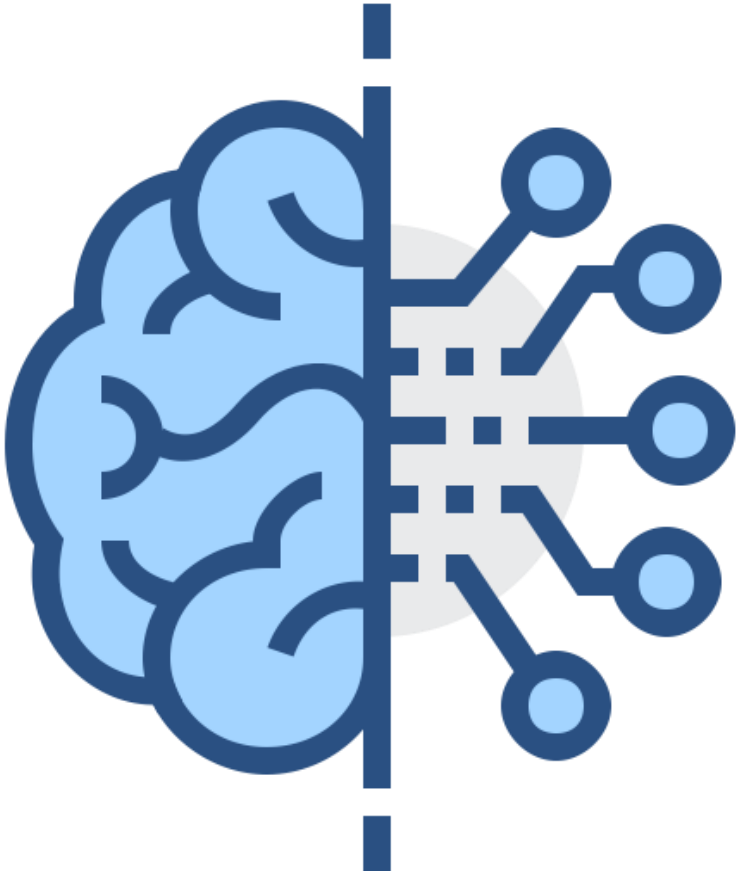
Unity ML-Agents
Hopper 만들기

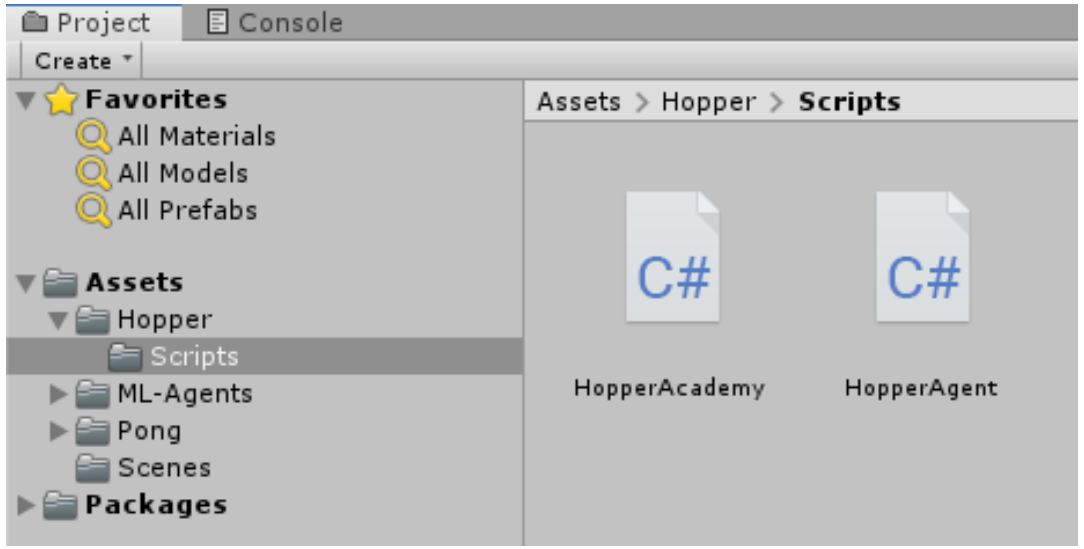
2019. 10. 27 민규식



Unity ML-agents 예제

Learning (External)

- UnitySDK -> Assets -> ML-Agents를 유니티 프로젝트 뷰의 Assets에 복사
- Template 폴더를 다시 복사하여 폴더명을 Hopper로 변경
- 프로젝트뷰의 스크립트 파일명 변경 (HopperAcademy, HopperAgent) 및 Namespace 변경



HopperAcademy.cs

Assembly-CSharp

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

HopperAgent.cs

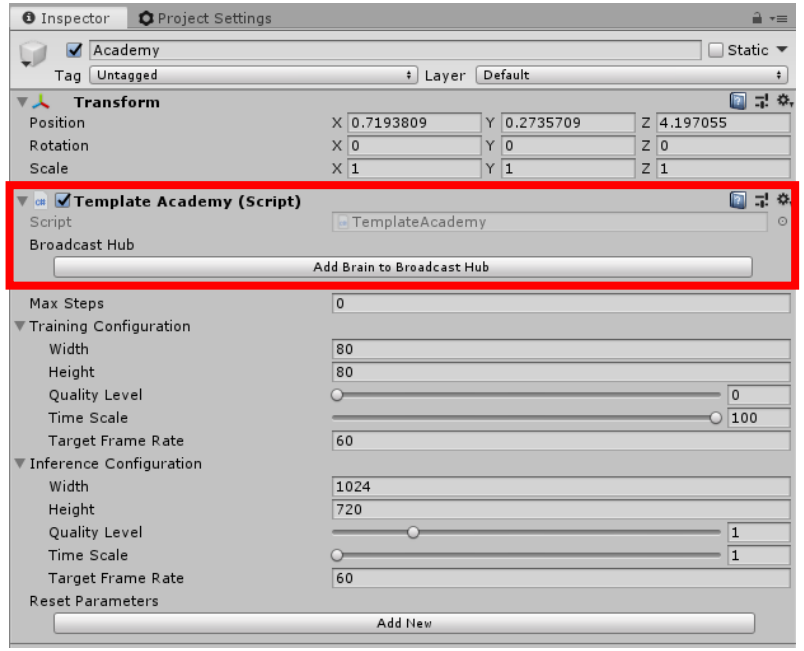
Assembly-CSharp

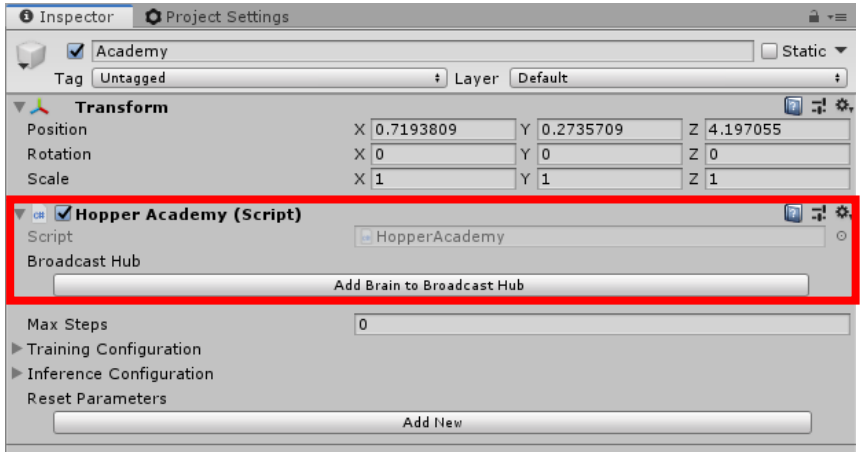
```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using MLAgents;
5
6  public class HopperAgent : Agent {
7
```

Unity ML-agents 예제

Learning (External)

- Scene의 이름을 Hopper로 변경
- Academy를 TemplateAcademy에서 HopperAcademy로 변경





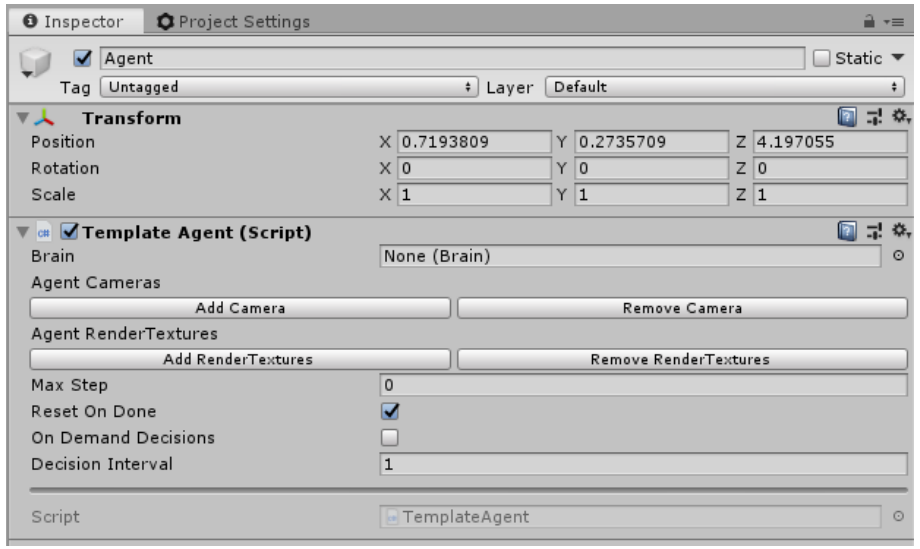
Unity ML-agents 예제

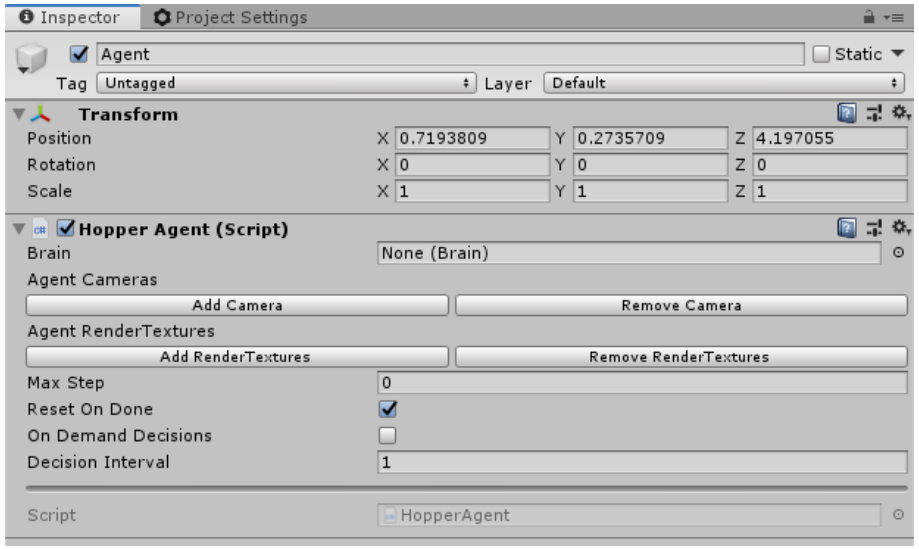
~ Reinforcement Learning Korea ~

RL KOREA

Learning (External)

- Agent를 TemplateAgent에서 HopperAgent로 변경
- Agent의 경우 Transform을 reset!

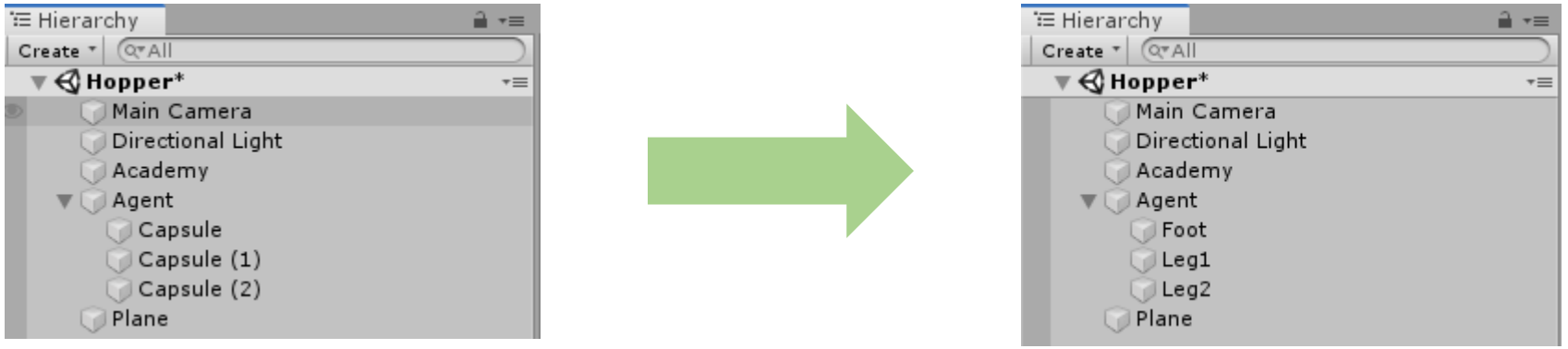




Unity ML-agents 예제

Learning (External)

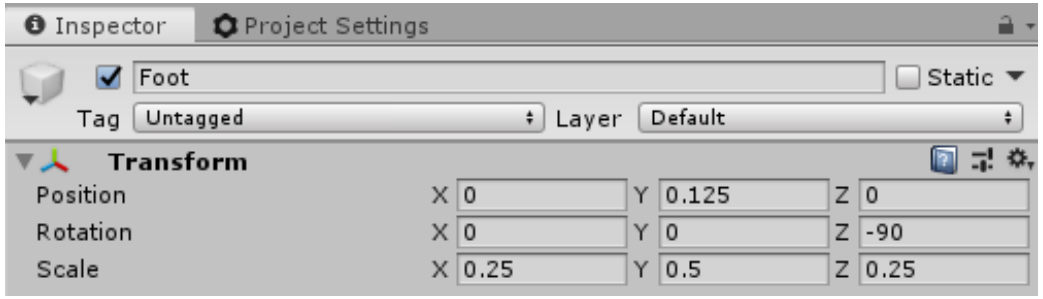
- 하이러키뷰에서 Agent 하위에 3개의 [3D Object -> Capsule]을 만들고 이름을 각각 다음과 같이 변경
  - Foot
  - Leg1
  - Leg2

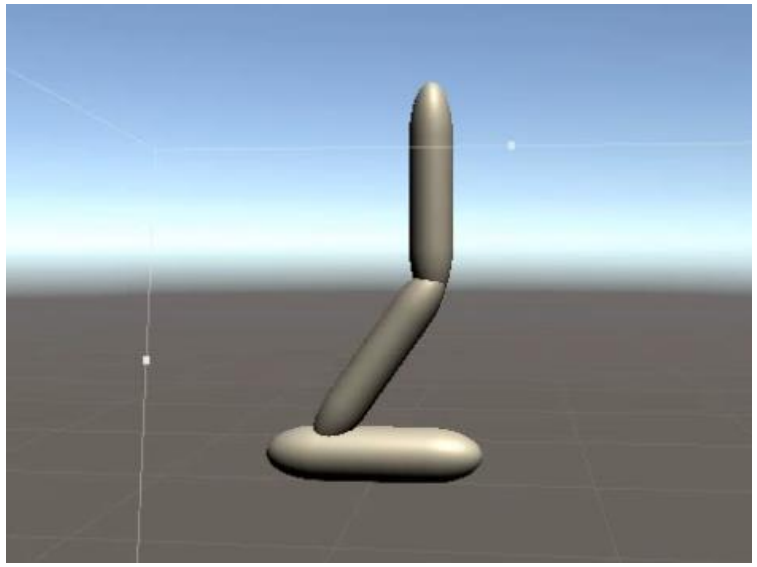


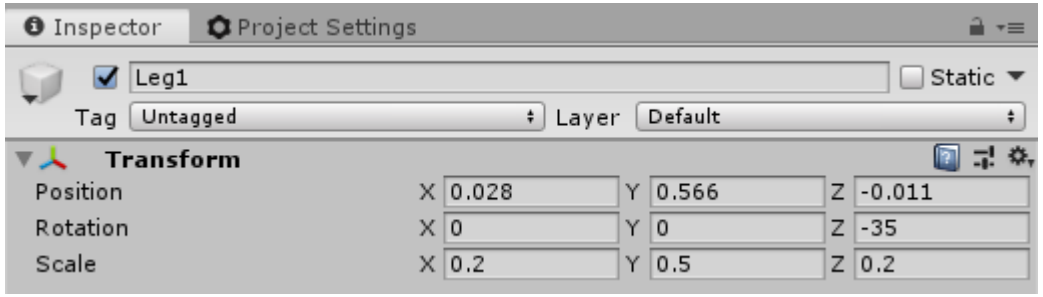
Unity ML-agents 예제

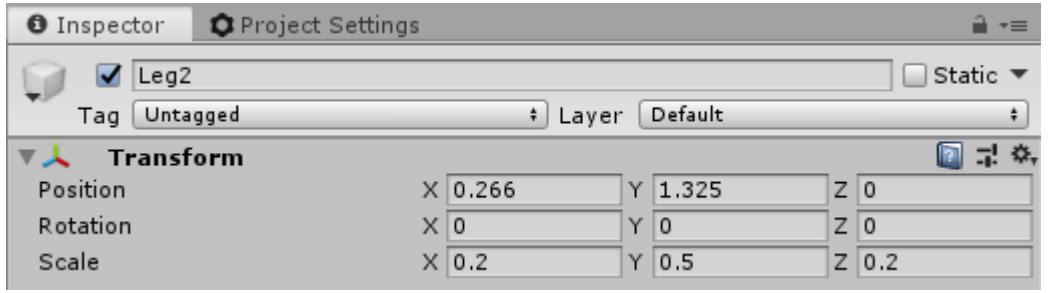
Learning (External)

- Foot, Leg1, Leg2의 Transform 설정을 다음과 같이 변경





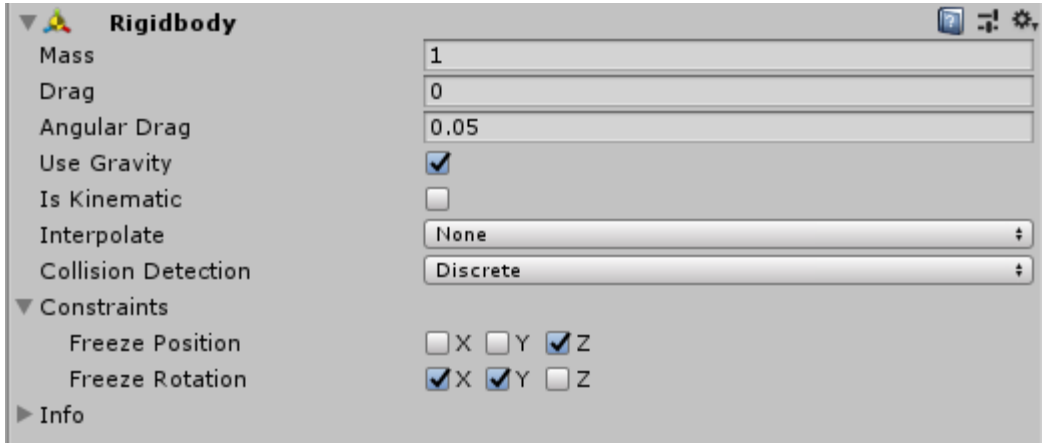




Unity ML-agents 예제

Learning (External)

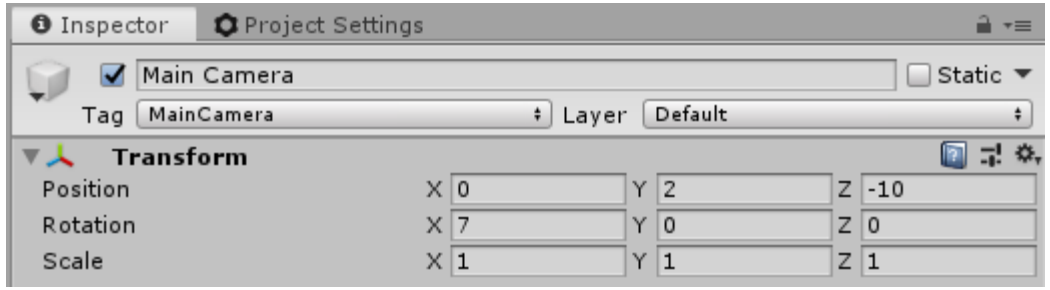
- 그리고 Foot, Leg1, Leg2 각각에 Rigidbody를 추가한 후 모든 object에 대해 다음과 같은 Constraints 설정
  - Z 방향 이동, X, Y 방향 회전 제한

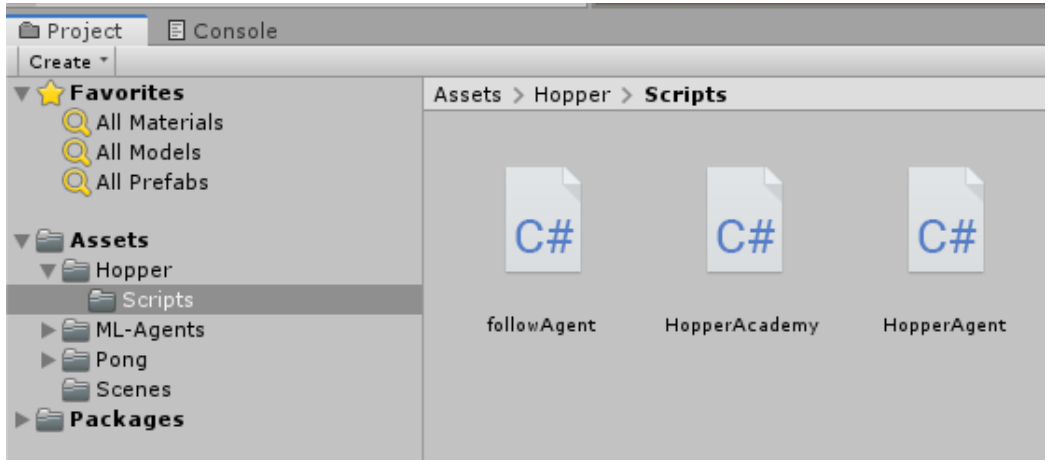


Unity ML-agents 예제

Learning (External)

- 카메라의 Transform 변경 및 [Hopper->Scripts] 폴더에 followAgent 스크립트 추가





Unity ML-agents 예제

Learning (External)

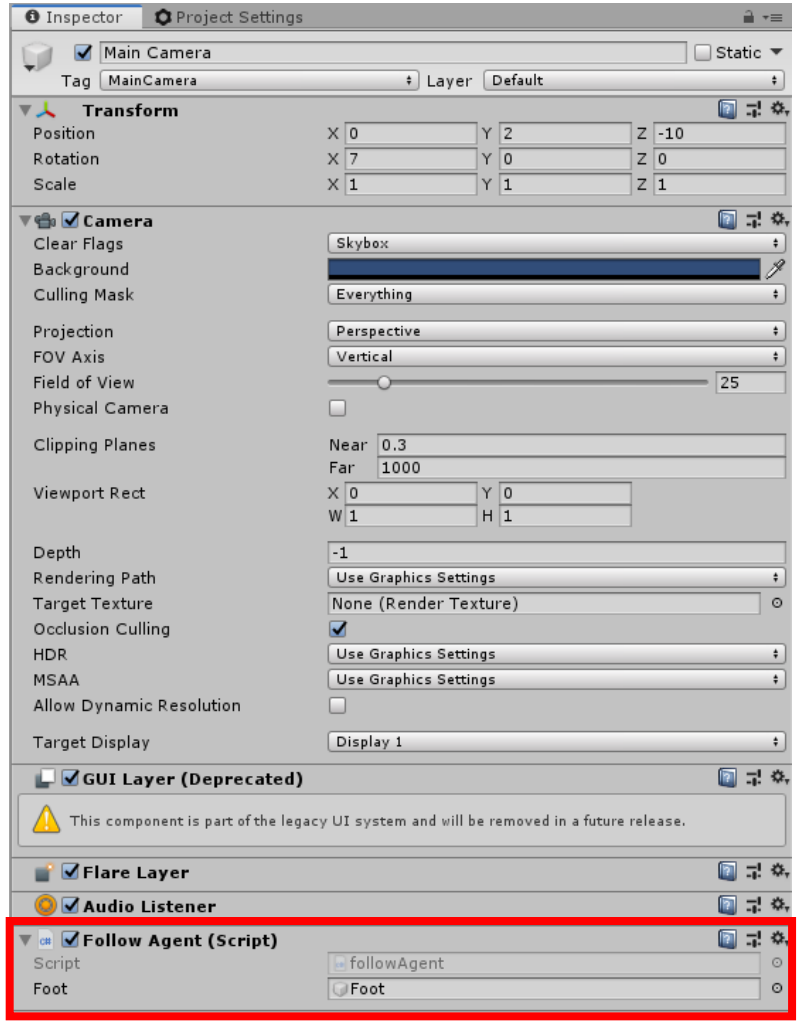
- followAgent 스크립트

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class followAgent : MonoBehaviour
{
	public GameObject foot;

	// Use this for initialization
	void Start()
	{

	}

	// Update is called once per frame
	void Update()
	{
	this.transform.position = new Vector3(foot.transform.position.x, this.transform.position.y, this.transform.position.z);
	}
}
```

Unity ML-agents 예제

Learning (External)

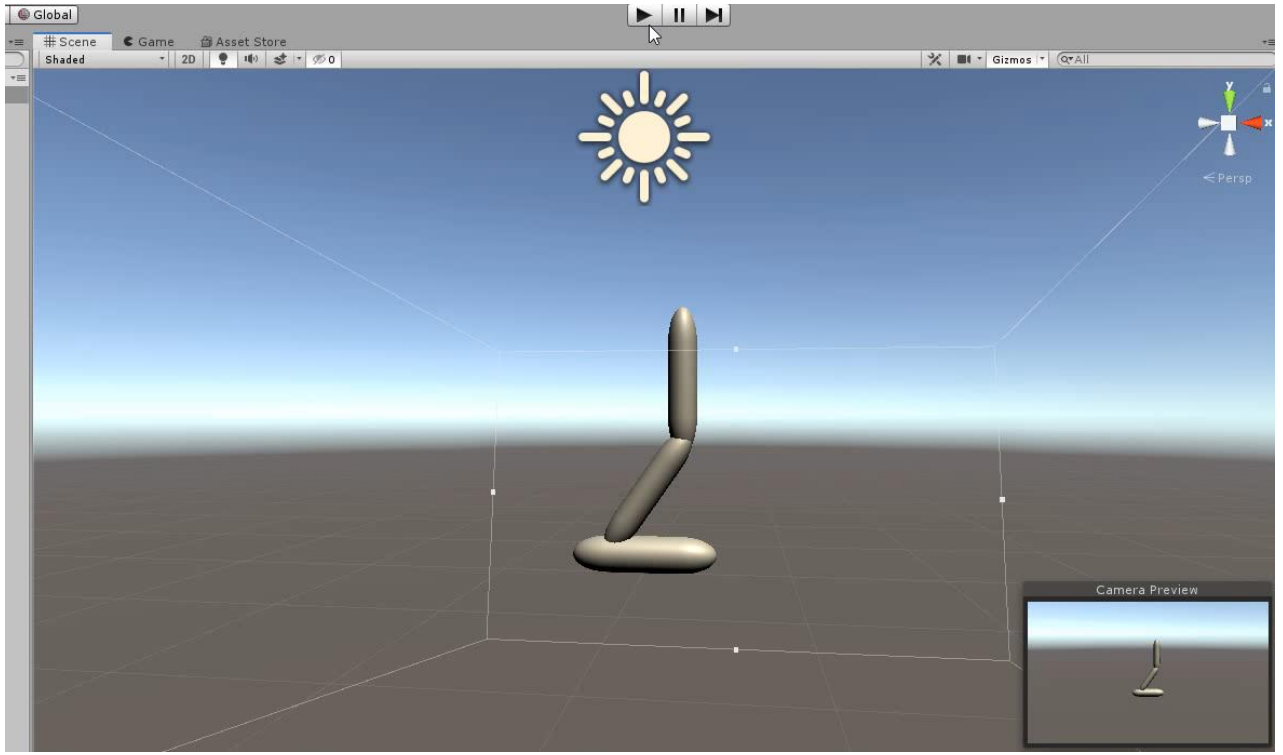
- Main Camera에 followAgent 스크립트 추가
- Foot object에 하이러키뷰의 Foot 드래그 앤 드랍



Unity ML-agents 예제

Learning (External)

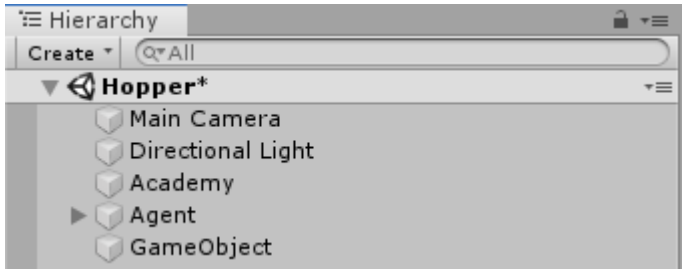
- 여기까지 구현 후 실행한 결과 -> 바닥이 없어서 다 떨어져버림 ㅜㅜ

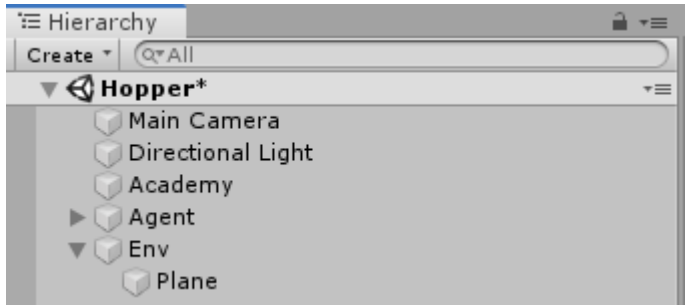


Unity ML-agents 예제

Learning (External)

- 바닥 추가하기
  - Empty Object를 하이러키뷰에 만들고 이름을 Env로 변경
  - Env의 하위에 Plane 추가

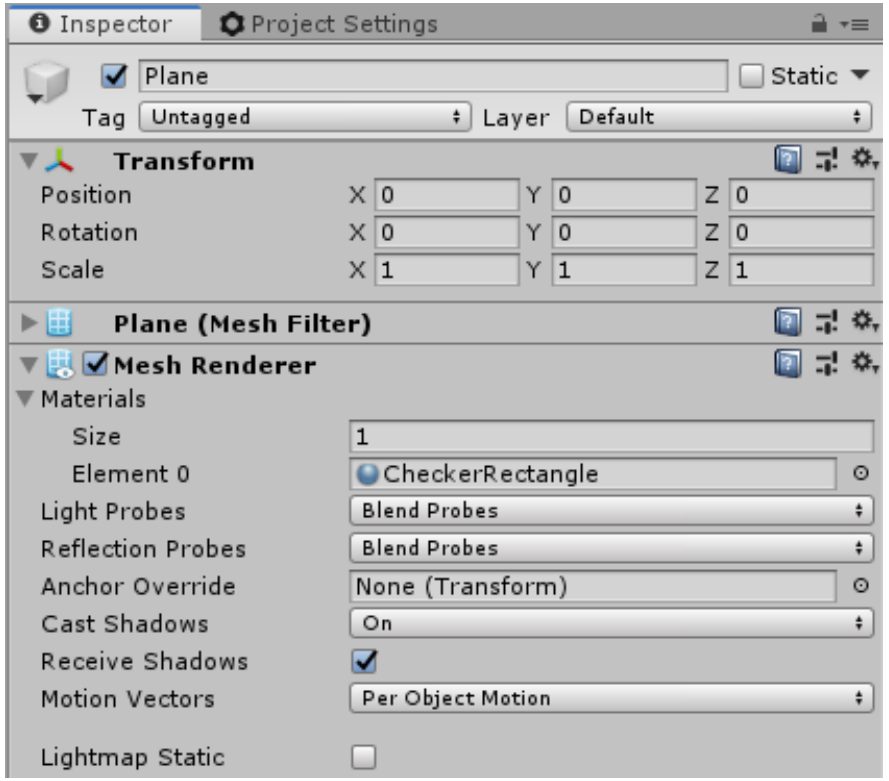




Unity ML-agents 예제

Learning (External)

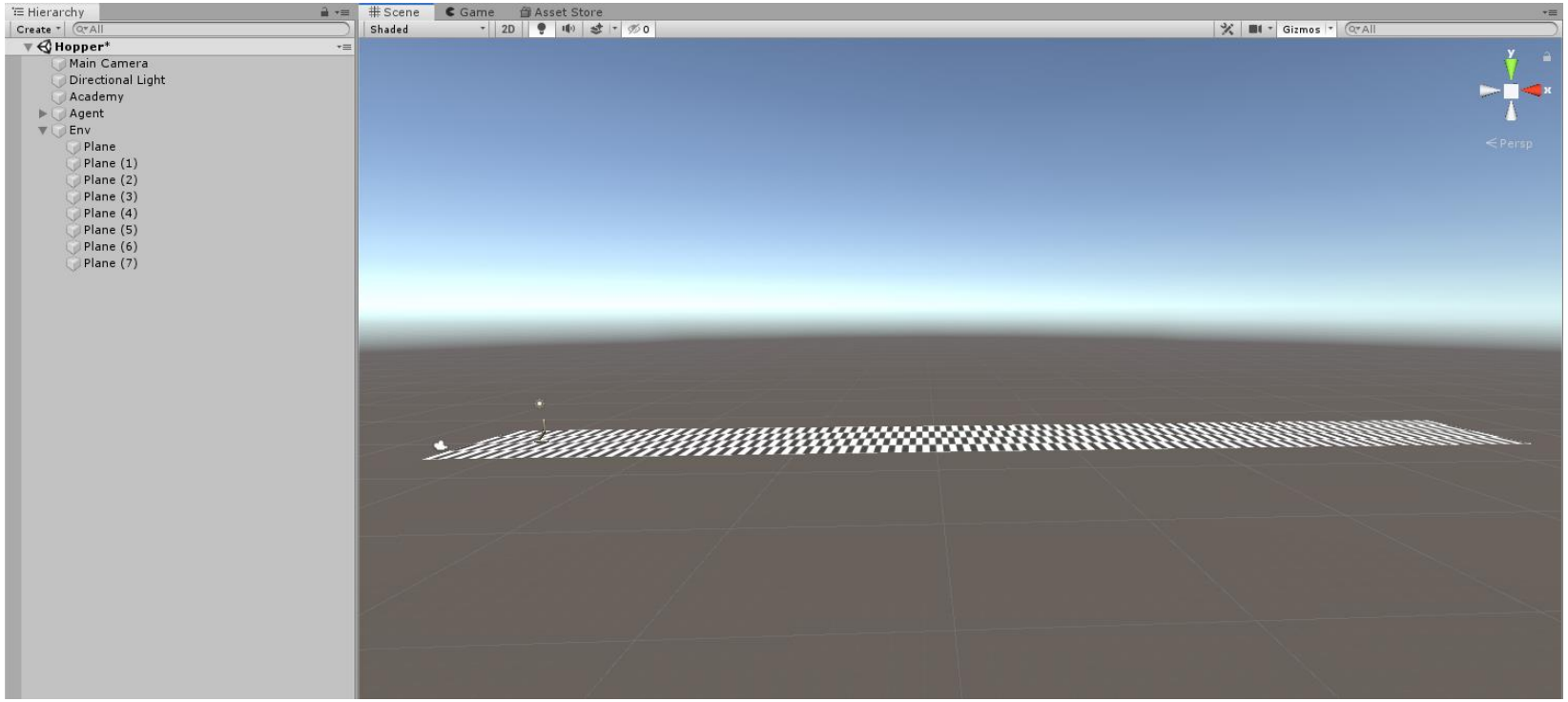
- 바닥 추가하기
  - Plane의 Mesh Renderer -> Materials의 Element 0를 CheckerRectangle로 변경
  - 이후 Control + D를 통해 Plane을 복제하고 이를 이동시켜서 바닥을 길게 만들어줌



Unity ML-agents 예제

Learning (External)

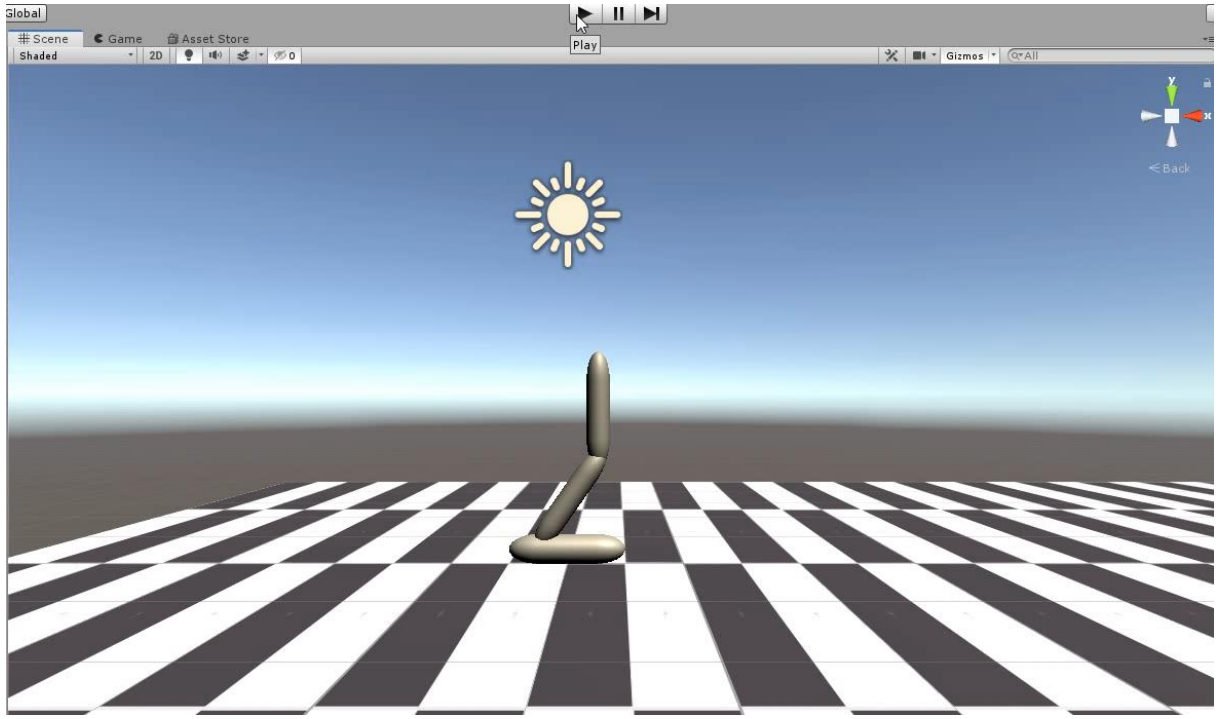
- 바닥 추가하기



Unity ML-agents 예제

Learning (External)

- 여기까지 구현 후 실행한 결과 -> 아래로 떨어지지는 않지만 각 부분이 붙어있지 못하고 분리되어 버림

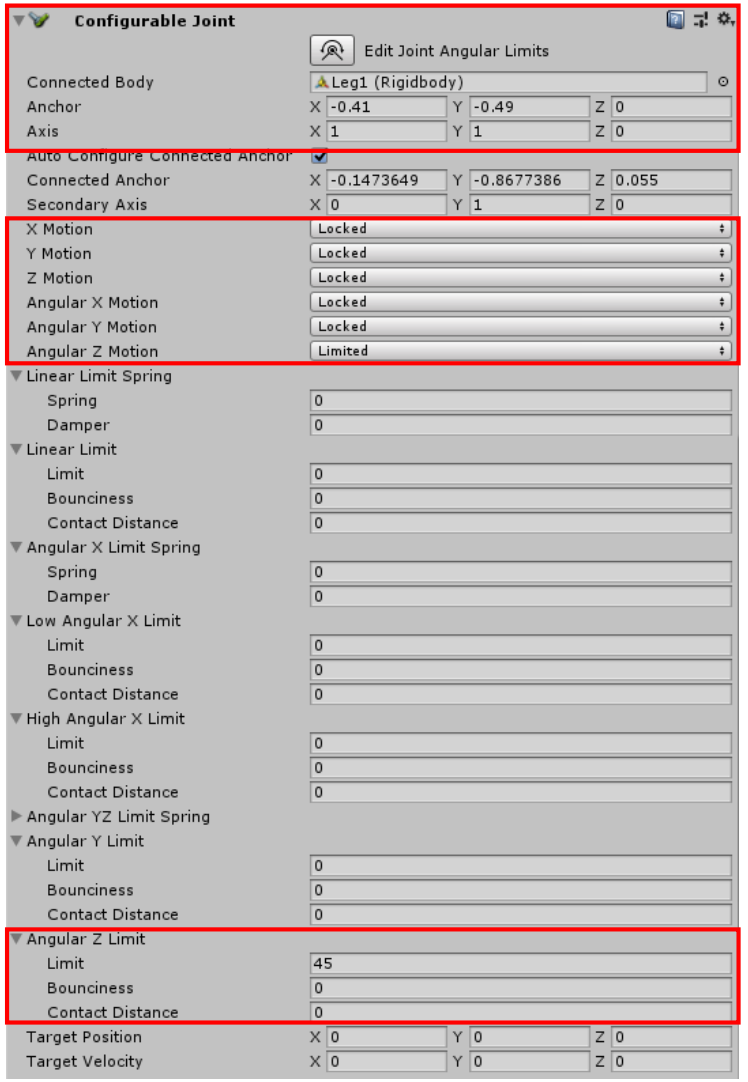


Unity ML-agents 예제

Learning (External)

- 관절을 만들자!

  => Configurable Joint
- Foot의 인스펙터뷰에서 Configurable Joint 추가
- Connected Body에 Leg1 추가
- Anchor 및 Axis 조정
- 각 축으로의 이동과 회전 Locked 및 Limited도 설정
- Z축으로의 회전값을 45도로 제한

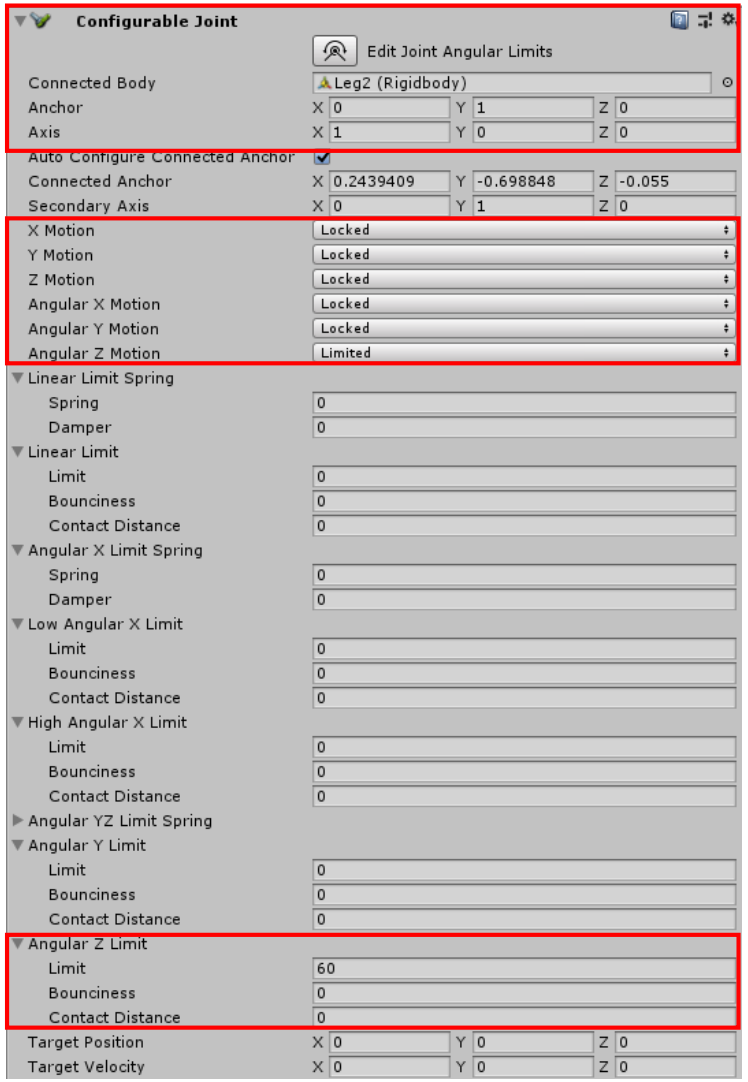


Unity ML-agents 예제

Learning (External)

- 관절을 만들자!

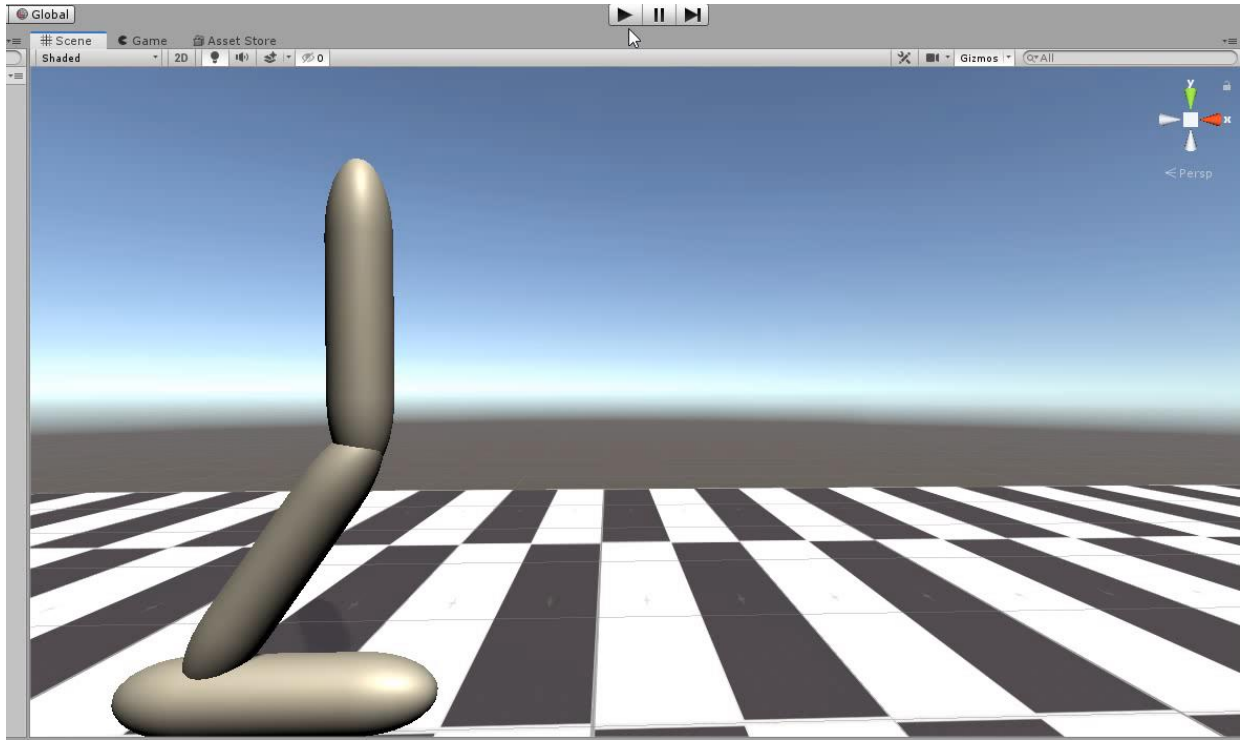
  => Configurable Joint
- Leg1의 인스펙터뷰에서 Configurable Joint 추가
- Connected Body에 Leg2 추가
- Anchor 및 Axis 조정
- 각 축으로의 이동과 회전 Locked 및 Limited도 설정
- Z축으로의 회전값을 60도로 제한



Unity ML-agents 예제

Learning (External)

- 여기까지 실행하면 이제 관절 제작도 끝~



Unity ML-agents 예제

Learning (External)

- Agent 스크립트: Object, RigidBody, Vector3 등 선언

```
public GameObject foot;
public GameObject leg1;
public GameObject leg2;

private Rigidbody foot_rBody;
private Rigidbody leg1_rBody;
private Rigidbody leg2_rBody;

private Vector3 foot_init_pos;
private Quaternion foot_init_rot;
private Vector3 leg1_init_pos;
private Quaternion leg1_init_rot;
private Vector3 leg2_init_pos;
private Quaternion leg2_init_rot;
```

Unity ML-agents 예제


Learning (External)

- Agent 스크립트
  - InitializeAgent 함수: 환경이 시작되었을 때 딱 한번 호출되는 함수
  - 초기 위치 및 회전에 대해 저장하고 Rigidbody 요소 불러오기

```
public override void InitializeAgent()
{
      foot_init_pos = foot.transform.position;
      foot_init_rot = foot.transform.rotation;
      leg1_init_pos = leg1.transform.position;
      leg1_init_rot = leg1.transform.rotation;
      leg2_init_pos = leg2.transform.position;
      leg2_init_rot = leg2.transform.rotation;

      foot_rBody = foot.GetComponent<Rigidbody>();
      leg1_rBody = leg1.GetComponent<Rigidbody>();
      leg2_rBody = leg2.GetComponent<Rigidbody>();
}
```

Unity ML-agents 예제

Learning (External)

- Agent 스크립트
  - CollectObservations 함수: 상태를 추가하는 함수
  - 이동량, 상대 위치, 속도, 각속도 등을 포함: 총 19개

```
public override void CollectObservations()
{
        AddVectorObs(foot.transform.position.x / 50f);
        AddVectorObs(foot.transform.localPosition.x);
        AddVectorObs(foot.transform.localPosition.y);
        AddVectorObs(foot.transform.localRotation.z);
        AddVectorObs(foot_rBody.velocity.x);
        AddVectorObs(foot_rBody.velocity.y);
        AddVectorObs(foot_rBody.angularVelocity.z);

        AddVectorObs(leg1.transform.localPosition.x);
        AddVectorObs(leg1.transform.localPosition.y);
        AddVectorObs(leg1.transform.localRotation.z);
        AddVectorObs(leg1_rBody.velocity.x);
        AddVectorObs(leg1_rBody.velocity.y);
        AddVectorObs(leg1_rBody.angularVelocity.z);

        AddVectorObs(leg2.transform.localPosition.x);
        AddVectorObs(leg2.transform.localPosition.y);
        AddVectorObs(leg2.transform.localRotation.z);
        AddVectorObs(leg2_rBody.velocity.x);
        AddVectorObs(leg2_rBody.velocity.y);
        AddVectorObs(leg2_rBody.angularVelocity.z);
}
```

Unity ML-agents 예제


Learning (External)

- Agent 스크립트
  - AgentAction 함수:
    - 행동 결정
    - 보상 및 게임 종료 설정
  - 액션: 각 오브젝트에 대한 토크
  - 보상
    - 몸통이 너무 내려가지 않도록
    - 발이 앞으로 가는 속도가 빠르도록
    - 최대한 전방으로 많이 이동하도록

```
public override void AgentAction(float[] vectorAction, string textAction)
{
    for (int k = 0; k < vectorAction.Length; k++)
    {
        vectorAction[k] = Mathf.Clamp(vectorAction[k], -1f, 1f);
    }

    float torque = 80f;

    foot_rBody.AddTorque(transform.forward * torque * vectorAction[0]);
    leg1_rBody.AddTorque(transform.forward * torque * vectorAction[1]);
    leg2_rBody.AddTorque(transform.forward * torque * vectorAction[2]);

    if (leg2.transform.position.y < 0.8f)
    {
        AddReward(-1f);
        Done();
    }

    if (foot.transform.position.x > 50f)
    {
        AddReward(1f);
        Done();
    }

    AddReward(0.01f*(foot_rBody.velocity.x) + 0.00001f*(foot.transform.position.x / 50f));
}
```

Unity ML-agents 예제


Learning (External)

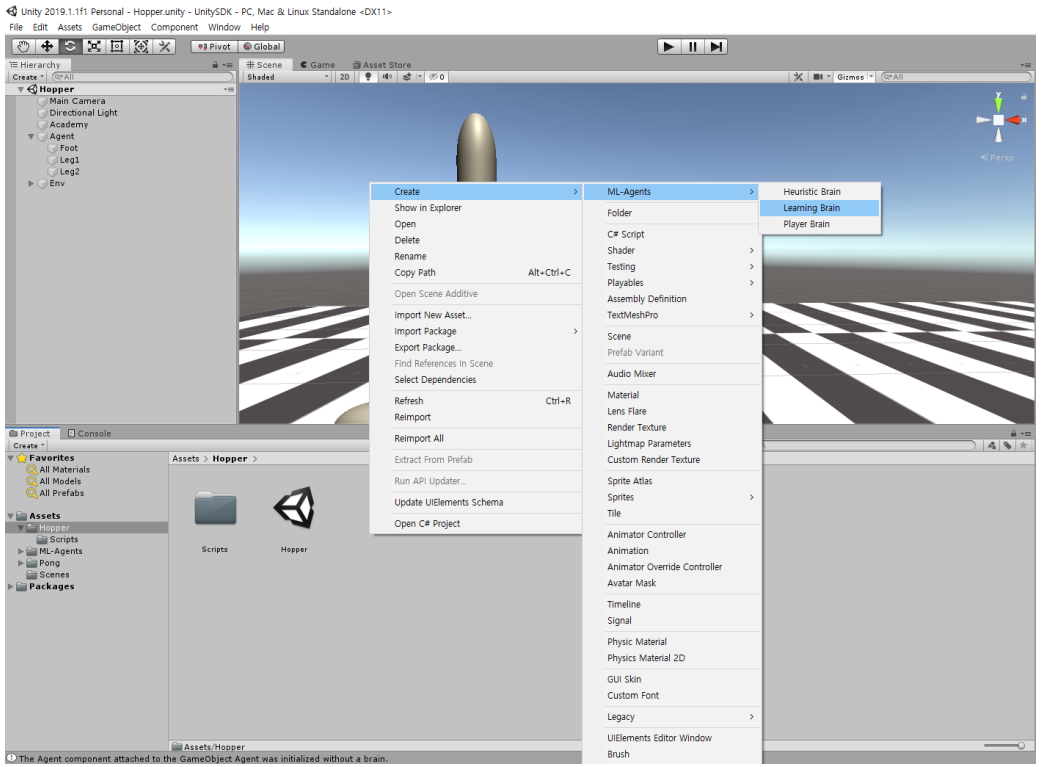
- Agent 스크립트
  - AgentReset 함수:
    - 한 에피소드 종료시 설정
    - 위치 및 회전 초기화

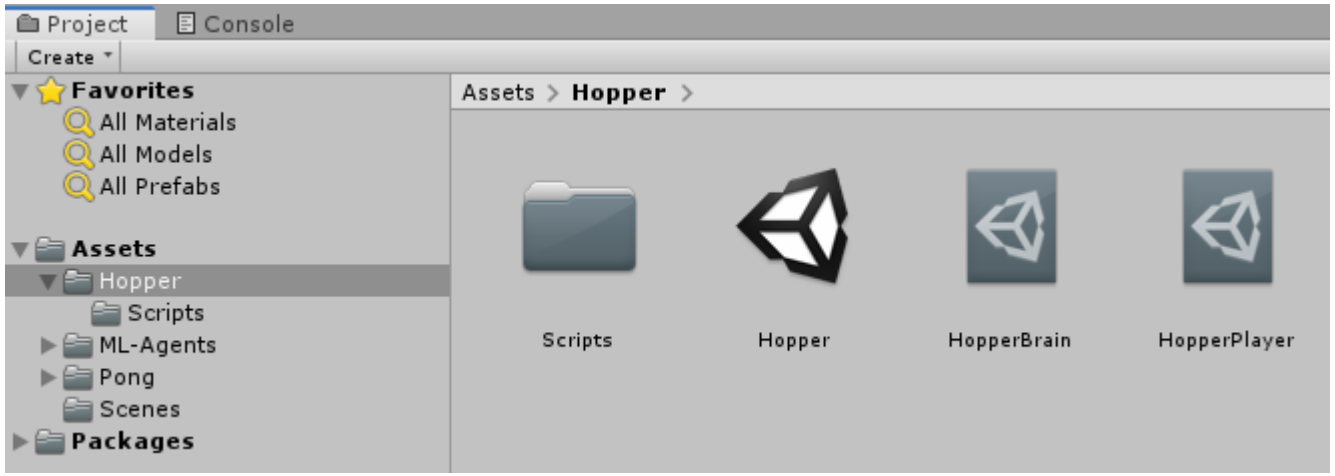
```
public override void AgentReset()
{
    foot.transform.position = foot_init_pos;
    foot.transform.rotation = foot_init_rot;
    leg1.transform.position = leg1_init_pos;
    leg1.transform.rotation = leg1_init_rot;
    leg2.transform.position = leg2_init_pos;
    leg2.transform.rotation = leg2_init_rot;
}
```

Unity ML-agents 예제

Learning (External)

- Learning Brain과 Player Brain 추가

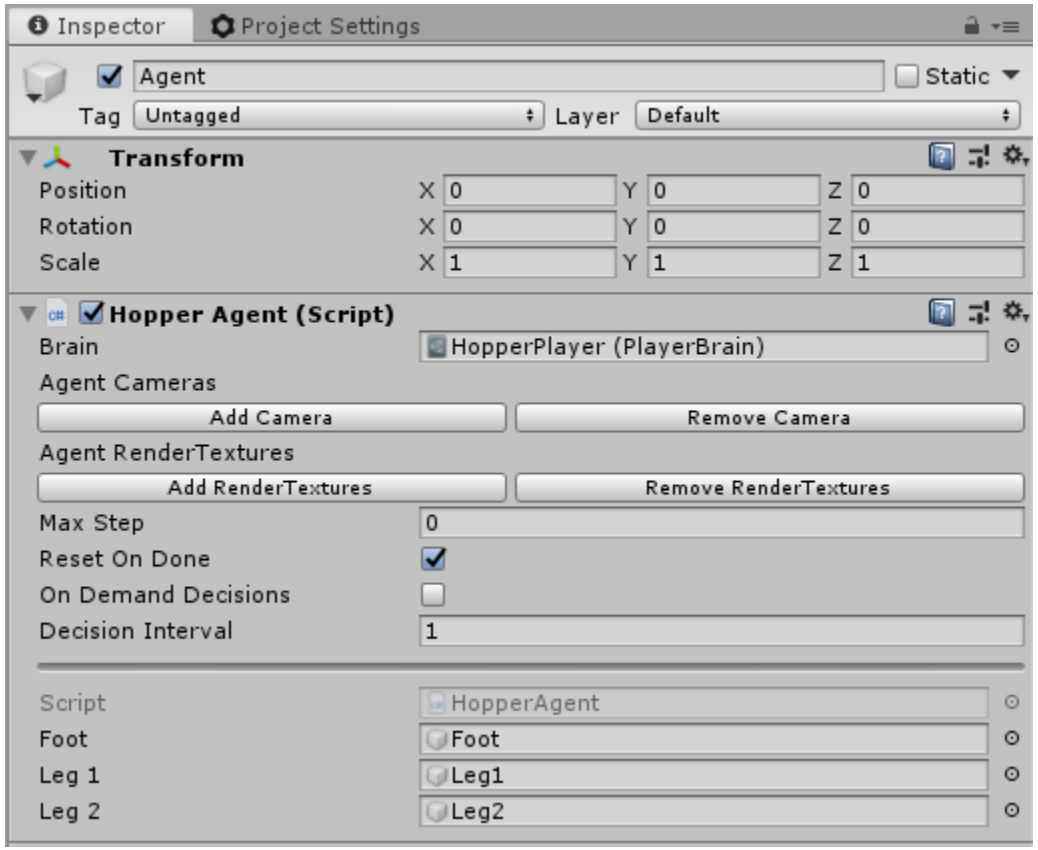




Unity ML-agents 예제

Learning (External)

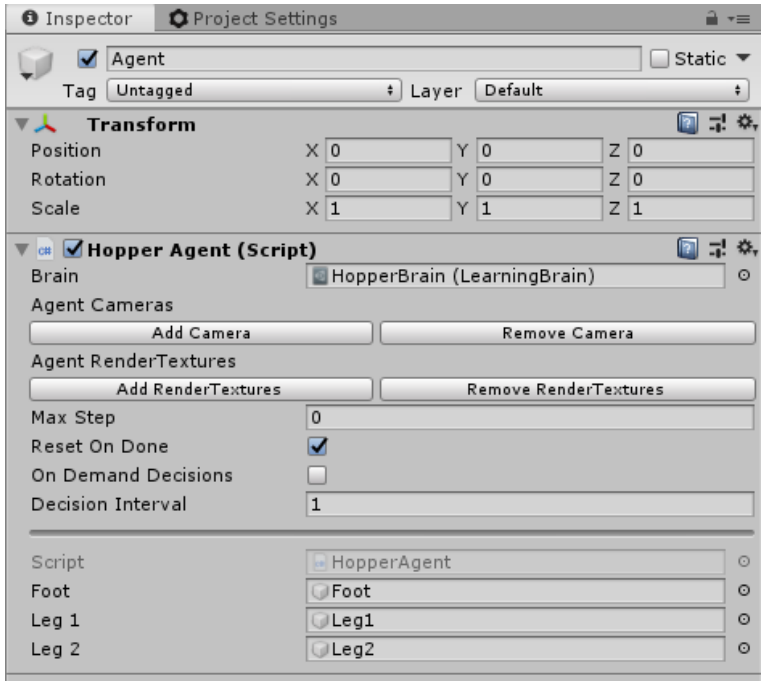
- Agent 스크립트의 Foot, Leg1, Leg2에 오브젝트 연결
- 처음에는 PlayerBrain으로 실행해서 오류 없이 작동하는지 확인

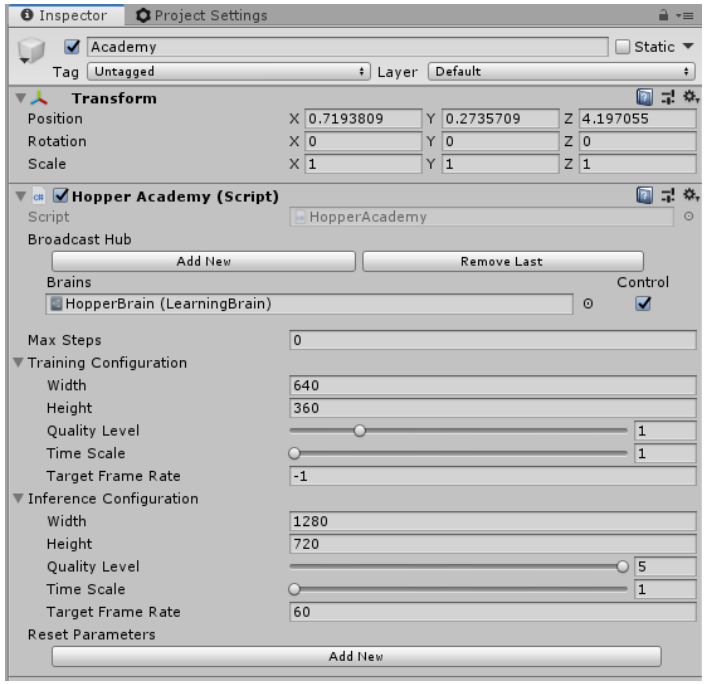


Unity ML-agents 예제

Learning (External)

- 오류가 없는 경우 Hopper Agent에 Learning Brain 설정
- Academy 설정

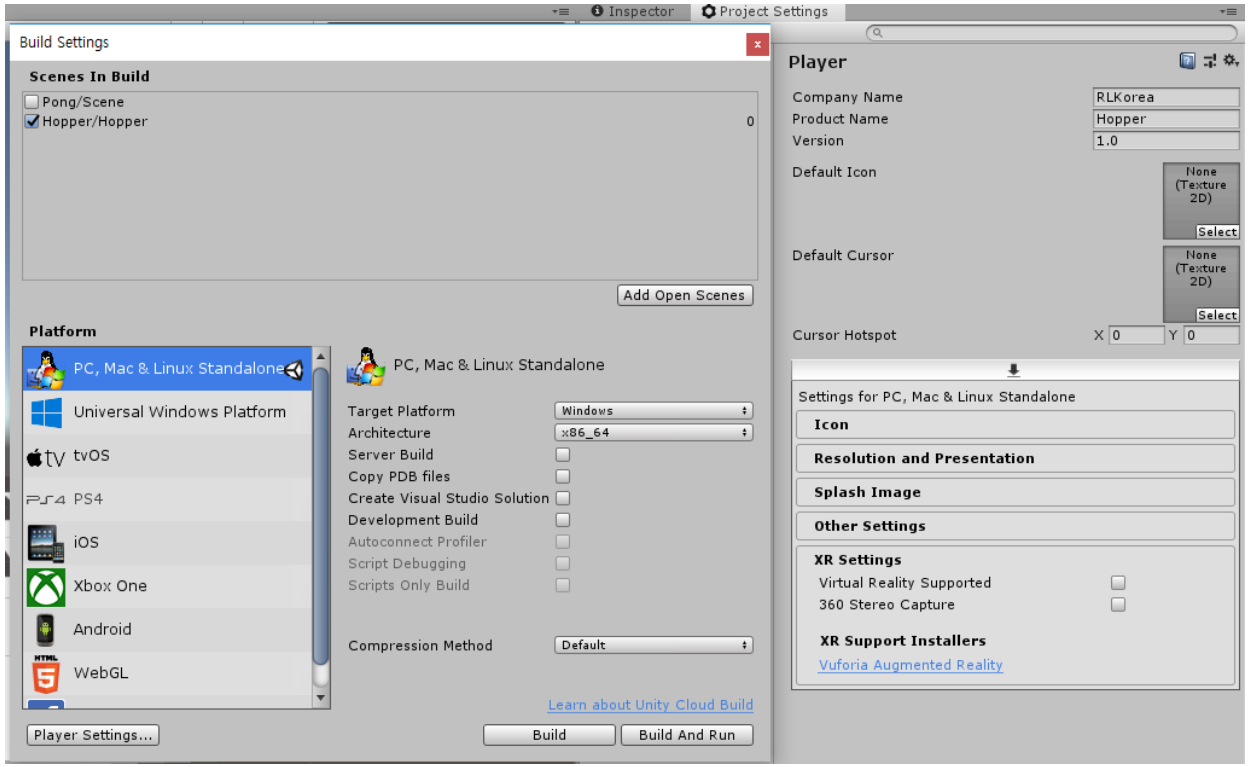




Unity ML-agents 체제

Learning (External)

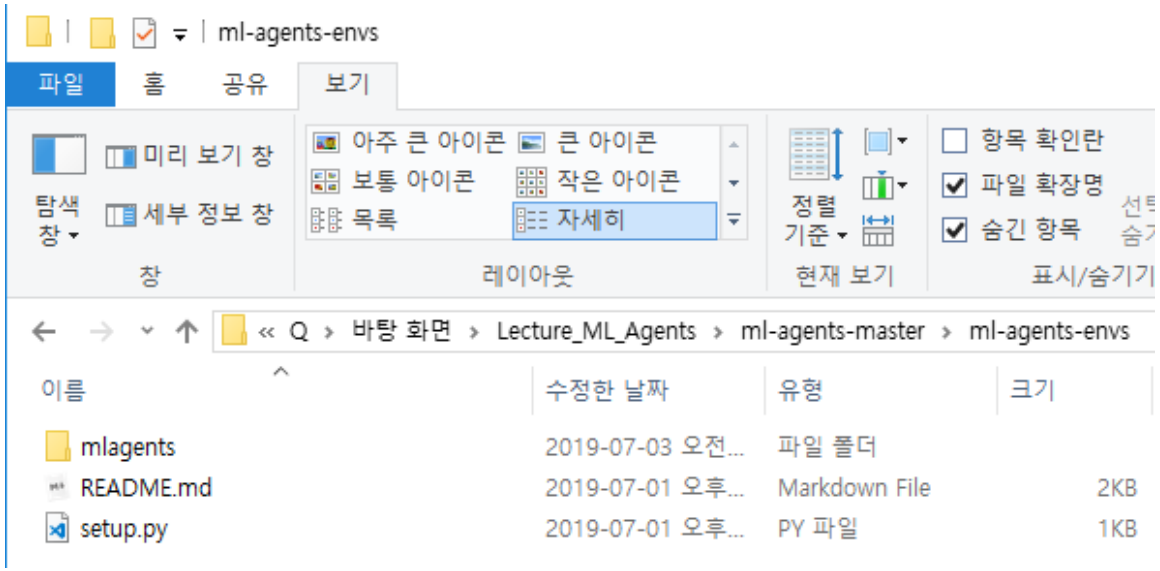
- 환경 빌드

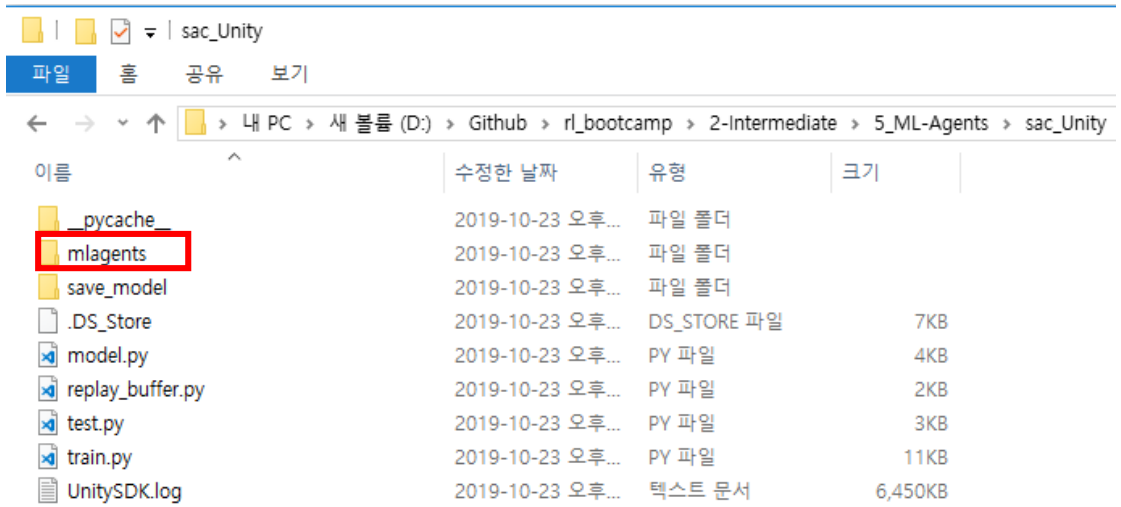


Unity ML-agents 예제

Learning (External)

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





Unity ML-agents 예제

SAC 코드 변경 (train.py)

```
1  import os
2  import gym
3  import time
4  import argparse
```

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

Unity ML-agents 예제

SAC 코드 변경 (train.py)

```python
parser = argparse.ArgumentParser()
parser.add_argument('--training_eps', type=int, default=500)
parser.add_argument('--eval_per_train', type=int, default=50)
parser.add_argument('--evaluation_eps', type=int, default=100)
parser.add_argument('--threshold_return', type=int, default=-230)
parser.add_argument('--gamma', type=float, default=0.99)
parser.add_argument('--alpha', type=float, default=0.05)
parser.add_argument('--automatic_entropy_tuning', type=bool, default=True)
parser.add_argument('--buffer_size', type=int, default=10000)
parser.add_argument('--batch_size', type=int, default=64)
parser.add_argument('--actor_lr', type=float, default=1e-4)
parser.add_argument('--qf_lr', type=float, default=3e-3)
parser.add_argument('--alpha_lr', type=float, default=1e-4)
args = parser.parse_args()
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
```

```python
parser = argparse.ArgumentParser()
parser.add_argument('--training_eps', type=int, default=50000)
parser.add_argument('--eval_per_train', type=int, default=20)
parser.add_argument('--evaluation_eps', type=int, default=5)
parser.add_argument('--threshold_return', type=int, default=25)
parser.add_argument('--gamma', type=float, default=0.99)
parser.add_argument('--alpha', type=float, default=0.05)
parser.add_argument('--automatic_entropy_tuning', type=bool, default=True)
parser.add_argument('--buffer_size', type=int, default=50000)
parser.add_argument('--batch_size', type=int, default=64)
parser.add_argument('--actor_lr', type=float, default=1e-4)
parser.add_argument('--qf_lr', type=float, default=3e-3)
parser.add_argument('--alpha_lr', type=float, default=1e-4)
args = parser.parse_args()
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
```

Unity ML-agents 예제

SAC 코드 변경 (train.py)

```python
def main():
    # Initialize environment
    env = gym.make('Pendulum-v0')
    obs_dim = env.observation_space.shape[0]
    act_dim = env.action_space.shape[0]
    act_limit = env.action_space.high[0]
    print('State dimension:', obs_dim)
    print('Action dimension:', act_dim)

    # Set a random seed
    env.seed(0)
    np.random.seed(0)
    torch.manual_seed(0)
```

```python
def main():
    # Initialize environment
    env = UnityEnvironment(file_name='../env/Hopper/Hopper')

    default_brain = env.brain_names[0]
    brain = env.brains[default_brain]

    env_info = env.reset(train_mode=True)[default_brain]

    obs_dim = env_info.vector_observations[0].shape[0]
    act_dim = brain.vector_action_space_size[0]
    print('State dimension:', obs_dim)
    print('Action dimension:', act_dim)

    # Set a random seed
    np.random.seed(0)
    torch.manual_seed(0)
```

Unity ML-agents 예제

SAC 코드 변경 (train.py)

```python
def run_one_episode(steps, eval_mode):
    total_reward = 0.

    obs = env.reset()
    done = False

    # Keep interacting until agent reaches a terminal state.
    while not done:
        steps += 1

        if eval_mode:
            action, _, _ = actor(torch.Tensor(obs).to(device))
            action = action.detach().cpu().numpy()
            next_obs, reward, done, _ = env.step(action)
        else:
            # Collect experience (s, a, r, s') using some policy
            _, action, _ = actor(torch.Tensor(obs).to(device))
            action = action.detach().cpu().numpy()
            next_obs, reward, done, _ = env.step(action)
```

```python
def run_one_episode(steps, eval_mode):
    total_reward = 0.

    env_info = env.reset(train_mode=True)[default_brain]
    obs = env_info.vector_observations[0]
    done = False

    # Keep interacting until agent reaches a terminal state.
    while not done:
        steps += 1

        if eval_mode:
            action, _, _ = actor(torch.Tensor(obs).to(device))
            action = action.detach().cpu().numpy()
            env_info = env.step(action)[default_brain]

            next_obs = env_info.vector_observations[0]
            reward = env_info.rewards[0]
            done = env_info.local_done[0]
        else:
            # Collect experience (s, a, r, s') using some policy
            _, action, _ = actor(torch.Tensor(obs).to(device))
            action = action.detach().cpu().numpy()
            env_info = env.step(action)[default_brain]

            next_obs = env_info.vector_observations[0]
            reward = env_info.rewards[0]
            done = env_info.local_done[0]
```

Unity ML-agents 예제

SAC 코드 변경 (train.py)

```
250
251 if __name__ == '__main__':
252     main()
```

```
264     env.close()
265
266 if __name__ == '__main__':
267     main()
```

Unity ML-agents 예제

SAC 코드 변경 (test.py)

```python
import os
import gym
import argparse
import numpy as np
import torch
from model import *
```

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

```python
import os
import argparse
import numpy as np
import torch
from model import *
from mlagents.envs import UnityEnvironment
```

경로 지정!

```python
# Configurations
parser = argparse.ArgumentParser()
parser.add_argument('--load', type=str, default="Hopper_ep_480_rt_26.04_t_10116.pt",
                    help='load the saved model')
args = parser.parse_args()
```

Unity ML-agents 예제

SAC 코드 변경 (test.py)

```python
def main():
    env = gym.make('Pendulum-v0')
    obs_dim = env.observation_space.shape[0]
    act_dim = env.action_space.shape[0]
```

```python
def main():
    env = UnityEnvironment(file_name='../env/Hopper/Hopper')

    default_brain = env.brain_names[0]
    brain = env.brains[default_brain]

    env_info = env.reset(train_mode=False)[default_brain]

    obs_dim = env_info.vector_observations[0].shape[0]
    act_dim = brain.vector_action_space_size[0]
```

Unity ML-agents 예제

SAC 코드 변경 (test.py)

```python
    for episode in range(1, 10001):
        total_reward = 0.

        obs = env.reset()
        done = False

        while not done:
            if args.render:
                env.render()

            action, _, _ = mlp(torch.Tensor(obs).to(device))
            action = action.detach().cpu().numpy()
            next_obs, reward, done, _ = env.step(action)

            total_reward += reward
            obs = next_obs

        sum_returns += total_reward
        num_episodes += 1

        average_return = sum_returns / num_episodes if num_episodes > 0 else 0.0

        if episode % 10 == 0:
            print('----------------------------------------')
            print('Episodes:', num_episodes)
            print('AverageReturn:', average_return)
            print('----------------------------------------')

if __name__ == "__main__":
    main()
```

```python
    for episode in range(1, 10001):
        total_reward = 0.

        obs = env_info.vector_observations[0]
        done = False

        while not done:
            action, _, _ = mlp(torch.Tensor(obs).to(device))
            action = action.detach().cpu().numpy()
            env_info = env.step(action)[default_brain]

            next_obs = env_info.vector_observations[0]
            reward = env_info.rewards[0]
            done = env_info.local_done[0]

            total_reward += reward
            obs = next_obs

        sum_returns += total_reward
        num_episodes += 1

        average_return = sum_returns / num_episodes if num_episodes > 0 else 0.0

        if episode % 10 == 0:
            print('----------------------------------------')
            print('Episodes:', num_episodes)
            print('AverageReturn:', average_return)
            print('----------------------------------------')

    env.close()

if __name__ == "__main__":
    main()
```

학습 결과