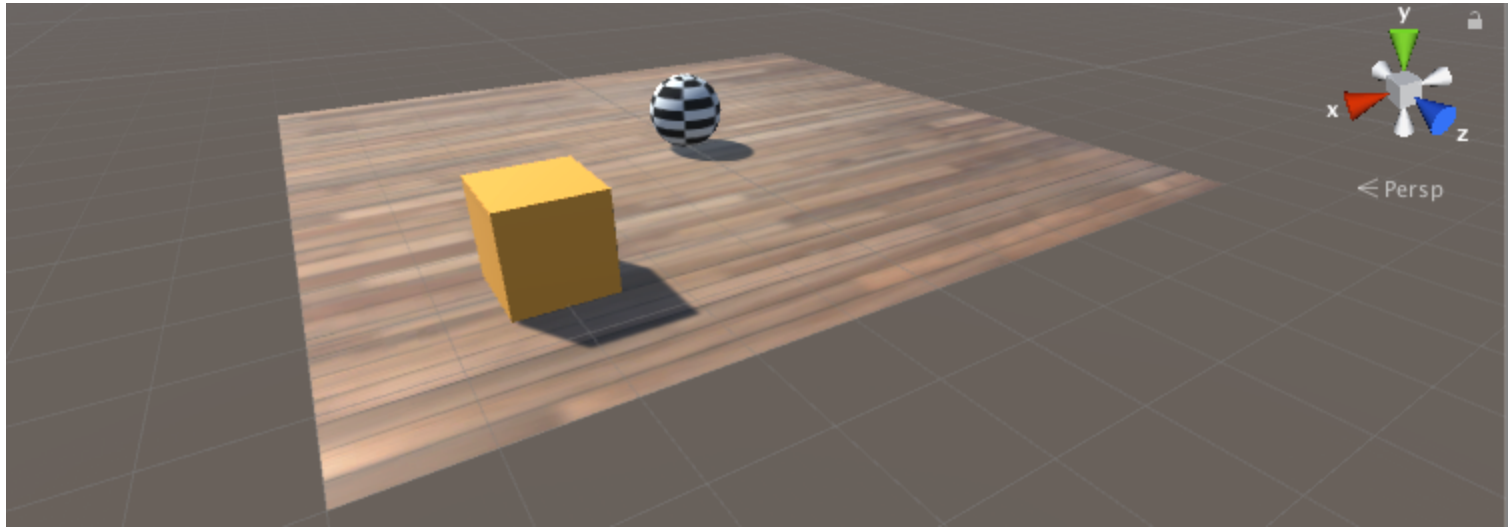
Handmade RL - third story

나만의 환경 만들기

wonseok Jung

PART1

새로운 환경 만들기



- Unity Environment를 만들기 위한 튜토리얼
- Unity Environment는 강화학습 에이전트를 훈련 시키기 위해 사용될 수 있다.

Overview

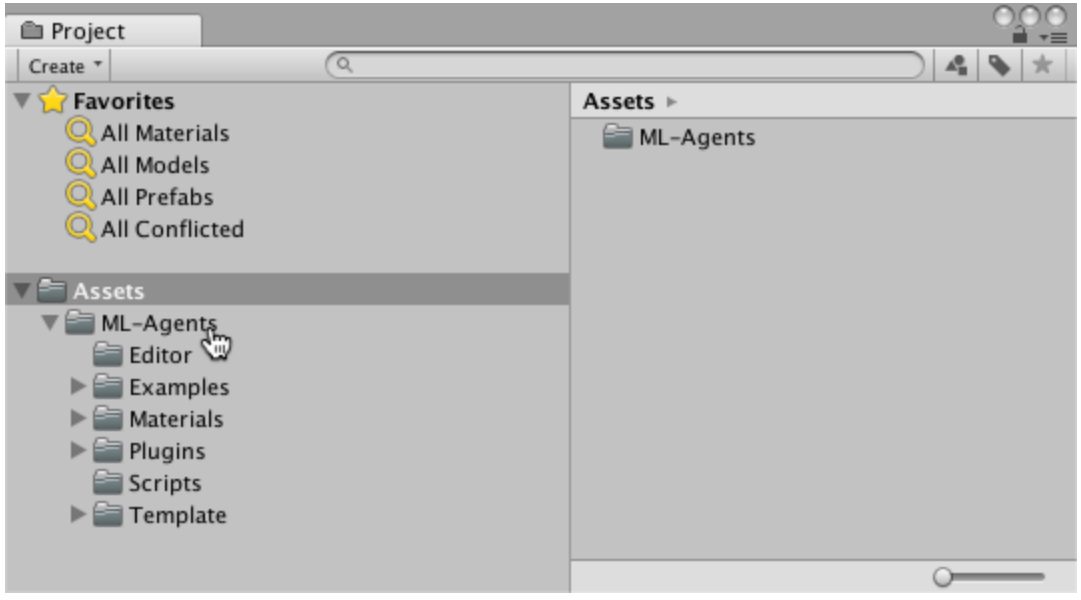
Basic steps to use ML-agent

1. 에이전트가 살수 있는 환경을 만든다.

좀 더 추가해야 할 부분

1. Set up the Unity Project

1. Unity Editor을 실행 한 뒤 RollerBall의 이름으로 새로운 프로젝트를 생성한다.
2. ML-Agents repository를 clone 하였던 폴더로 간다
3. unity-environments/Assets에 있는 ML-Agents폴더를 Unity Editor project window로 드래그 한다.



2. Create the Environment

ML-agent environment에서 simple scene를 만들어 보자

1. Hierarchy window에서 오른쪽 버튼을 클릭한다.
    - 3D object -> plane
2. Object의 이름을 Floor라고 지정한다.

2.1 Create the Envrionment-2

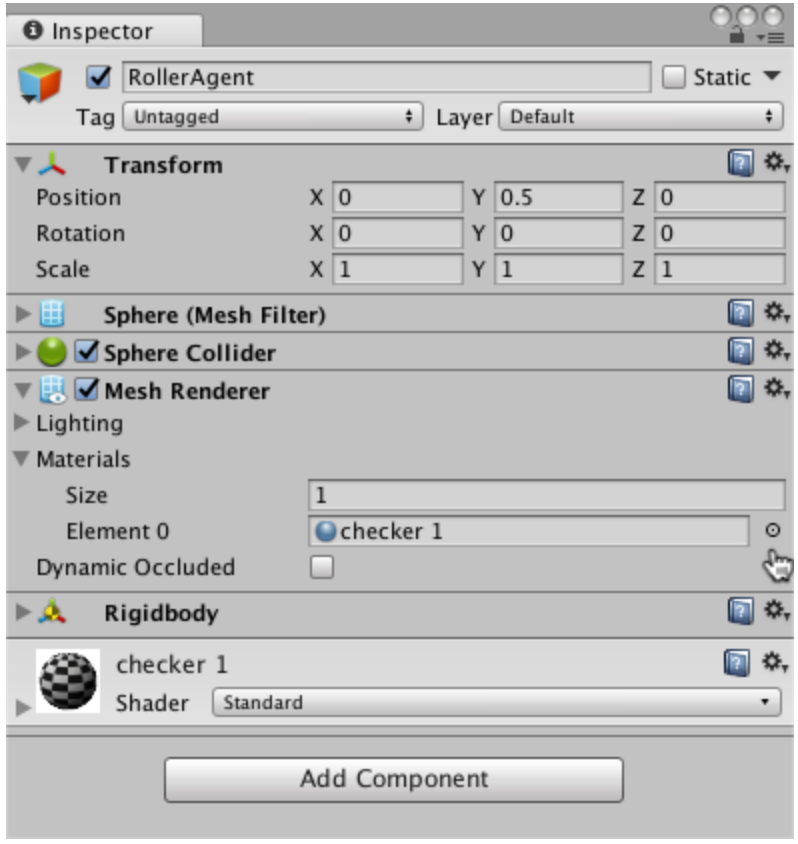
3. Transform을 수정한다.
    - Position =(0,0,0)
    - Rotation =(0,0,0)
    - scale = (1,1,1)
4. Mesh Render에서 Material property를 확장시키고 matrial을 floor로 바꾼다.

3. Add the Target Cube

1. Hierarchy window에서 오른쪽 마우스 클릭한 뒤 3D object -> spehere 를 선택한다.
2. RollerAgent로 이름을 바꾼다.
3. Target을 선택한뒤 Inspector window에서 properties를 본다.
4. Transform을 수정한다.
    - Position =(0,0.5,0)
    - Rotation =(0,0,0)
    - scale = (1,1,1)

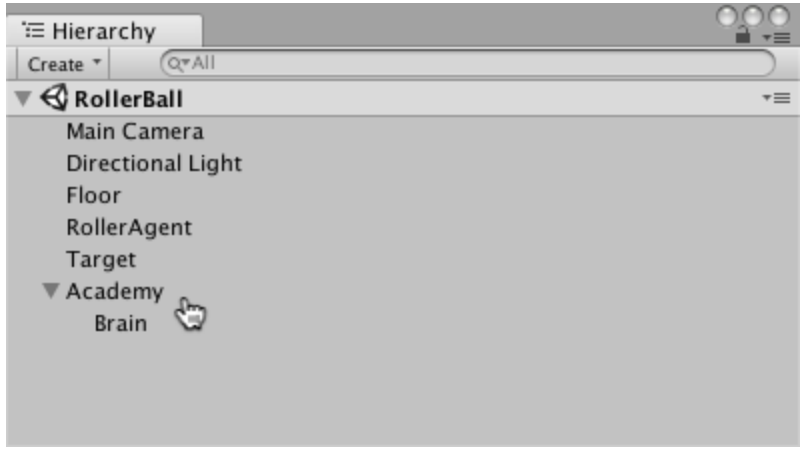
3.1 Add the Target Cube-2

5. Mesh Render에서 Material property를 확장시키고 matrial을 checker 1로 바꾼다.
6. Add Component를 클릭한다.
7. 이것을 Physics/Rigidbody에 추가한다.



4. Add Empty GameObjects to Hold the Academy and Brain

1. Hierarchy window에서 오른쪽 마우스 클릭한 뒤 Create Empty 를 선택한다.
2. Academy로 이름을 바꾼다.
3. Academy를 선택한 뒤 오른쪽 마우스 클릭한 뒤 Create Empty 를 선택한다.
4. Brain로 이름을 바꾼다.



5.Implement an Academy

1. Academy obeject를 선택하여 Inspector 창을 연다.
2. Add Component 선택
3. New SCript선택
4. RollerAcademy로 이름 변경
5. Creat and Add 선택

5.1 Edit the newollerAcademy sciprt

1.RollerAcademy를 더블클릭
2.Editor가 열리면, base class의 MonoBehaviour을 Academy로 변경한다.
3. `Start()` , `Update()` 를 지운다.

6. Add the Brain

1. 생성한 Brain을 선택하여 Inspector 창을 연다.
2. Add Component 선택
3. Brain type을 Player

Clear error 발생시

`Edit -> Project Settings -> Player Then in the inspector under "Other Settings" select the .NET 4.6 equivalent Scripting Runtime Version.`

7. Implement an Agent

1. RollerAgent 를 선택한뒤 inspector창을 연다
2. Add Component
3. New Script
4. RollerAgent로 이름을 바꾼다
5. Create and Add 선택

7.1 Edite new RolleraAgent

1.RollerAgent 를 더블클릭
2.Editor가 열리면, base class의 MonoBehaviour을 Agent로 변경한다.
3. `Start()` , `Update()` 를 지운다

So far

- 지금까지 Unity project에 ML-Agent를 add하는 작업을 했다.
- Agent가 Cube를 굴릴수 있도록 강화학습을 통해 학습시켜보자

Part 2

1. Initialization and Resetting the Agent

- Agent가 Target에 도착하면 상태를 초기화 시킨다. Agent가 platform 밖으로 떨어지면 다시 위로 올려놓는 작업을 자동으로 해주는 것이 필요하다.
- Target object를 옮기는 방법 :
    - Transform을 사용:
    - add a public field of type transform to Roller agent

- Agent 속도를 reset하기 위한 작업
    - Regibody는 Unity's physics simulation의 주요 구성이다.
- 이 reference를 사용하기 위해서해서
  Gameobject.GetComponent()를 사용하는 방법이 가장 좋다.
- Rigibody는 Start() script를 사용하여 부를수 있다.

Roller agent script는 다음과 같다.

```
public class RollerAgent : Agent {
    Rigidbody rBody;
    void Start ()
    {
        rBody = GetComponent<Rigidbody>();
    }
    public Transform Target;
    public override void AgentReset()
    {
        if (this.transform.position.y < -1.0)
        {
            //에이전트가 떨어지면
            this.transform.position = Vector3.zero;
            this.rBody.angularVelocity = Vector3.zero;
            this.rBody.velocity = Vector3.zero;
        }
        else
        {
        //타겟을 새로운 곳으로 옮긴다.
            Target.position = new Vector3(Random.value * 8 - 4,
                0.5f,
                Random.value * 8 - 4);
         }}}
```

Observing the Environment

- Agent는 information을 보내고 이 information을 사용하여 결정을 내린다.
- 에이전트 혹은 모델을 훈련시킬때 data가 neural network로 들어간다.
- 에이전트가 성공적으로 배우게 하기 위해서는 정확한 information이 필요하다

- 좋은 문제해결 방법은 correct data를 가지고 있냐에 따라 달렸다
- 여기서는 Agent가 collect하는 data는 다음과 같다.
    - Position of the target
    - Agent only collect z,x coordinates

```
Vector3 relativePosition = Traget.position -this.transform.posi

AddVectorObs(relativePosition.x / 5);
AddVectorObs(relativePosition.z / 5);
```

- edge of the plaform으로부터 distance

```
AddVectorObs((this.transform.position.x + 5) / 5);
AddVectorObs((this.transform.position.x - 5) / 5);
AddVectorObs((this.transform.position.z + 5) / 5);
AddVectorObs((this.transform.position.z - 5) / 5);
```

- Agent의 Velocity

```
AddVectorObs(rBody.velocity.x / 5);
AddVectorObs(rBody.velocity.z/5);
```

- 모든 value는 5로 나누어 normalize 해주었다.

Total of CollectObservations

```
public override void CollectObervations()
{
        # Relative position 계산
        Vector3 relativePosition = Traget.position -this.transf
        # Relative position
        AddVectorObs(relativePosition.x / 5);
        AddVectorObs(relativePosition.z / 5);

        # Distance to edge of the platform
        AddVectorObs((this.transform.position.x + 5) / 5);
        AddVectorObs((this.transform.position.x - 5) / 5);
        AddVectorObs((this.transform.position.z + 5) / 5);
        AddVectorObs((this.transform.position.z - 5) / 5);

        # Velocity
        AddVectorObs(rBody.velocity.x / 5);
        AddVectorObs(rBody.velocity.z/5);
```

Actions

- Agent Action()을 통해 Brain은 action array를 받는다.
- Roller agent의 action space는 continuous vector이고, 브레인으로부터 두개의 continuous control signal을 필요로 한다.
- action[0]은 x axis의 action[1]은 z axis의 action을 determine한다.(만약 Agent의 움직임을 세 방향으로 줄 수 있게 한다면, vector action size는 3이 될것이다.)

ACtions

```
Vector3 controlSignal = Vector3.zero;
controlSignal.x = Mathf.Clamp(action[0] , -1,1);
controlSignal.z = Mathf.Clamp(action[1], -1,1);

rBody.AddForce(controlSignal * speed);
```

- action value의 range를

Action values range : -1 ~ 1의 Range로 하였다.

Reward

- Reiforcement learning에서는 적절한 Reward이 설정으로 Agent가 목표를 달성할수 있게한다.
- AgentAction()에 reward를 assin한다.
- 이 reward를 사용하여 agent가 action을 선택할때 optimal한 action을 determine할수 있게 한다.
- Roller Agent는 Target에 도착했을때 reward를 받는다.

```
float distanceToTarget = Vector3.Distance(this.transform.positi
// 타겟에 도착하면

if (distanceToTarget < 1.42f)
{
        Done();
    AddReward(1.0f);
 }
```

Encourage the agent

- To Encourage the agent goes to target

```
// 가까워질때마다

if (distanceToTarget < previousDistance)

{

        AddReward(0.1f);
}
```

Time이 지날때마다 penalty

```
// Time penalty
AddReward(-0.05f);
```

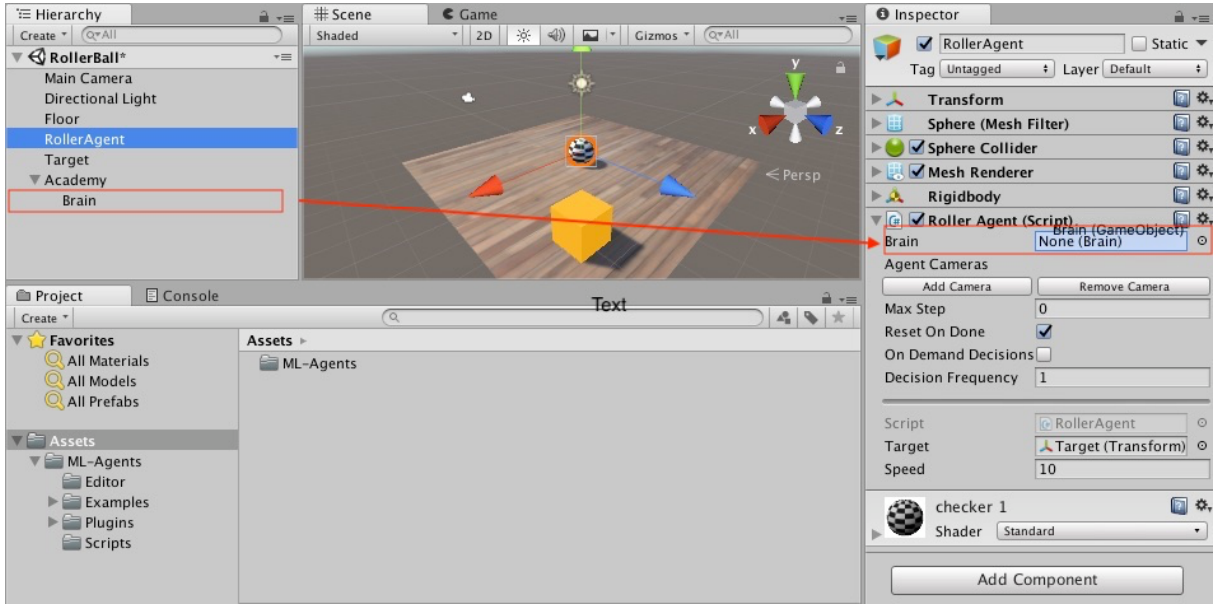
Agent가 밖으로 떨어질때

```
if (this.transform.position.y <- 1.0)
{
Done();
AddReward(-1.0f);
}
```

Action을 선택하는 로직은 다음과 같다.

```
public float speed = 10;
private float previousDistance = float.MaxValue;

public override void AgentAction(float[] vectorAction, string textAction)
{
    // Rewards
    float distanceToTarget = Vector3.Distance(this.transform.position,
                                              Target.position);

    // Reached target
    if (distanceToTarget < 1.42f)
    {
        Done();
        AddReward(1.0f);
    }

    // Getting closer
    if (distanceToTarget < previousDistance)
    {
        AddReward(0.1f);
    }

    // Time penalty
    AddReward(-0.05f);

    // Fell off platform
    if (this.transform.position.y < -1.0)
    {
        Done();
        AddReward(-1.0f);
    }
    previousDistance = distanceToTarget;

    // Actions, size = 2
    Vector3 controlSignal = Vector3.zero;
    controlSignal.x = Mathf.Clamp(vectorAction[0], -1, 1);
    controlSignal.z = Mathf.Clamp(vectorAction[1], -1, 1);
    rBody.AddForce(controlSignal * speed);
}
```

Final Editor step



- Hierarchy를 선택
- RollerAgent GameObject
- Brain을 drag하여 표시된 부분에 넣는다. (Target, Floor도 넣어야 한다)

Space Type 설정

- Hierarchy -> Academy -> Brain 선택
- Inspector에서 properties를 다음과 같이 변경

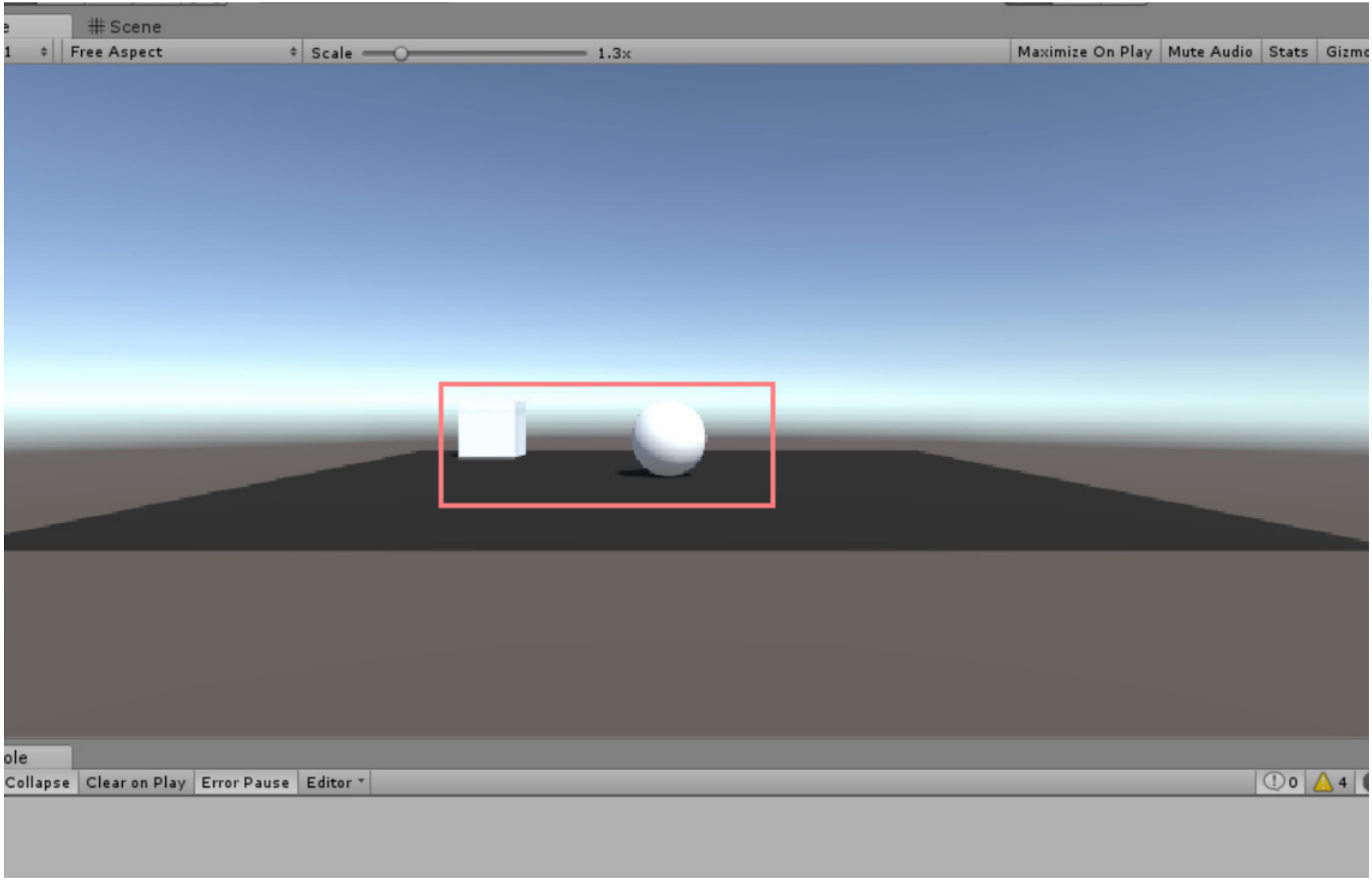
```
Vector Observation Space Type = Continuous
Vector Observation Space Size = 8
Vector Action Space Type = Continuous
Vector Action Space Size = 2
Brain Type = Player
```

환경 테스트해보기

- 환경이 다 만들어졌으니 타겟과 에이전트가 잘 움직이는지 방향키로 컨트롤
- Brain 선택
- Brain type을 Player로 변경
- Continuous layer Action을 Expand 한다.
- Size를 4로 설정
- 아래와 같이 키를 맵핑한다.

```
Element Key      Index   Value
Element 0        D       0       1
Element 1        A       0       -1
Element 2        W       1       1
Element 3        S       1       -1
```

실험



- w,s,a,d (키로 실험 해본다)

학습 시키기

- Brain Type을 Player에서 External로 바꾼다
- Training ML-Agents 에서 했던 방법과 동일하게 학습시킨다.
  링크 :
  https://github.com/wonseokjung/Handmade_RL/blob/master/Tutorial/2_secondstory/Handmade_RL-2.pdf

완료

공식홈페이지에 나와있는 tutorial은 미흡한 점이 많습니다

이 튜토리얼로 진행해주세요.

Scripts:
https://github.com/wonseokjung/Handmade_RL/tree/master/Tutorial/3_thridstory