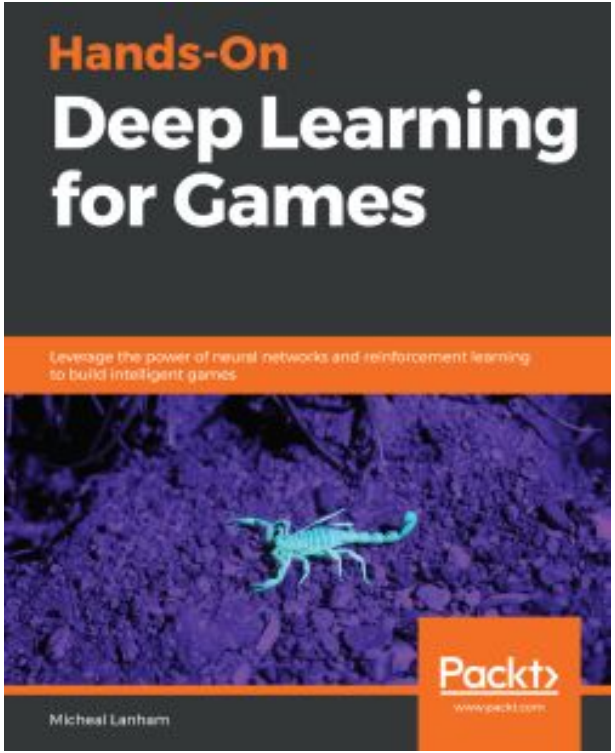
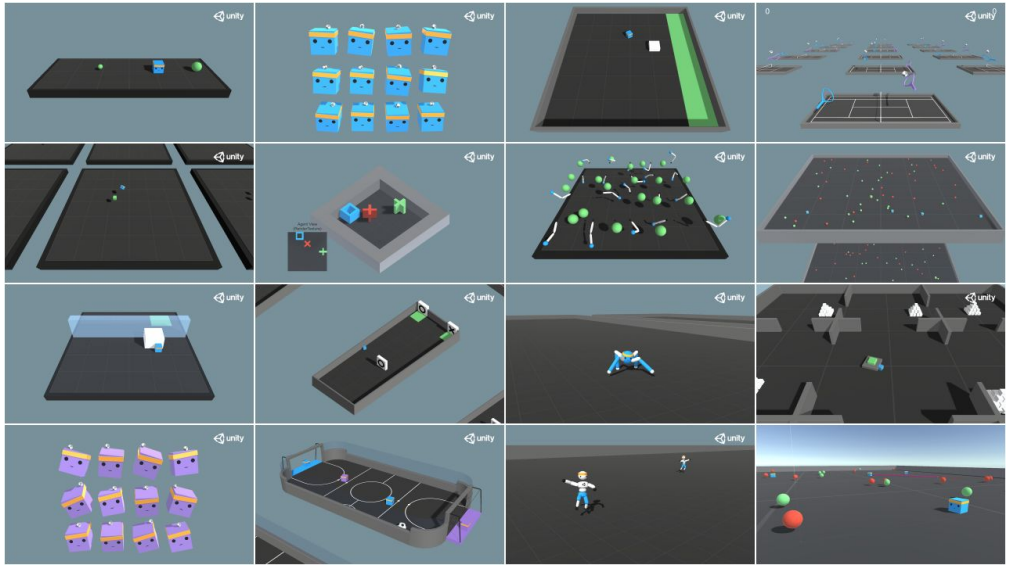
Unity ML-Agent Quick Guide

2019.10 이경만

본 강좌는 Hands-on Deep learning for game를 참고하고 있고 Unity ML agent 0.10을 기준으로 하고 있습니다.





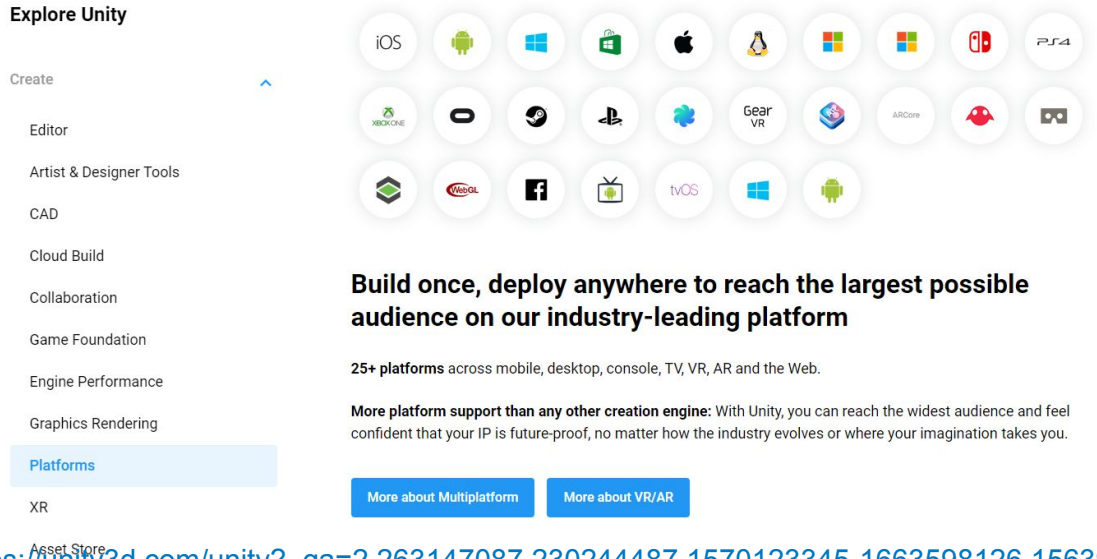
https://github.com/Unity-Technologies/ml-agents

Index

1. Why Unity ML Agent?
2. Install
3. ML Agent Basic
4. 기본 학습 : Grid world
5. vector observation 과 ray perception : Hallway
6. 커리큘럼 러닝 : wall jump
7. reward와 호기심 기반 학습 : pyramid
8. on/off line BC : tennis
9. GAIL : hallway
10. 결론

Why Unity 3D?

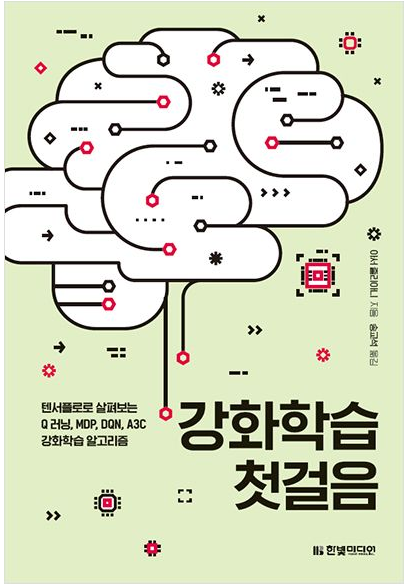
- 전세계에 650만 개발자가 사용.
- 게임 분야 뿐 아니라 시뮬레이션,애니매이션,영화등의 다양한 분야의 사용처
- 자체 애셋스토어/외주 플래폼등을 지원
- 27개이상의 플래폼을 지원,개인 사용자에게는 모두 무료



https://unity3d.com/unity?_ga=2.263147087.230244487.1570123345-1663598126.1563930107

Why ML Agent

- 2017년 부터 개발 , 아서 줄리아니가 주도
- 현제 0.10 (아직 베타버전)
- 시뮬레이션과 학습 환경의 유기적인 결합
- 다양한 학습 방법 지원(PPO,SAC , BC , GAIL , 커리큘럼 러닝)
- 잘 정리된 소스코드
- 활발하게 개발/사용 되고 있는 open source project



http://www.hanbit.co.kr/media/communi
review_view.html?hbr_idx=4915&page=
7

Unity Machine Learning Agents Toolkit https://unity3d.ai

reinforcement-learning unity3d deep-learning unity deep-reinforcement-learning neural-networks

1,433 commits 54 branches 0 packages 31 releases 84 contributors Apache-2.0

Branch: master New pull request Find File Clone or download

chriselion Only run bot on issues (#2671) Latest commit b5f05ab in 31 minutes

September 3, 2019 – October 3, 2019

Period: 1 month

Overview

104 Active Pull Requests

144 Active Issues

92 Merged Pull Requests

12 Proposed Pull Requests

Sep 3, 2017 – Oct 4, 2019

Contributions: Commits

Contributions to master, excluding merge commits

Excluding merges, **15 authors** have pushed **65 commits** to master and **296 commits** to all branches. On master, **793 files** have changed and there have been **52,985 additions** and **54,527 deletions**.

https://github.com/Unity-Technologies/ml-agents

Install

- Unity3d 설치 와 러닝 환경 설치로 나뉨
- Unity3d 설치 : 개발자 등록 먼저 Unity Hub 설치 권장
- Unity Hub는 다양한 버전의 유니티를 관리할 수 있는 프로그램

https://unity3d.com/kr/get-unity/download

- 학습환경 (ml agent)
- https://github.com/Unity-Technologies/ml-agents 에서 다운로드
- https://github.com/Unity-Technologies/ml-agents/blob/master/docs/Installation.md 를 참고해 설치
- https://github.com/Unity-Technologies/ml-agents/blob/master/docs/Installation-Windows.md 윈도우 버전 설치 문서

Install (python anaconda)

ML Agent Basic

https://github.com/Unity-Technologies/ml-agents/blob/master/docs/ML-Agents-Overview.md

- **Agents** - which is **attached to a Unity GameObject** (any character within a scene) and handles **generating its observations**, **performing the actions** it **receives and assigning a reward** (positive / negative) when appropriate. Each Agent is linked to exactly one Brain.
- **Brains** - which **encapsulates** the logic for **making decisions for the Agent.** In essence, the Brain is what holds on to the policy for each Agent and determines which actions the Agent should take at each instance. More specifically, it is the component that receives the observations and rewards from the Agent and **returns an action.**
- **Academy** - which **orchestrates the observation and decision making process**. Within the Academy, several environment-wide parameters such as the rendering quality and the speed at which the environment is run can be specified. The **External Communicator** lives within the Academy.

ML Agent Basic

https://github.com/Unity-Technologies/ml-agents/blob/master/docs/ML-Agents-Overview.md

- **Observations** - what the medic perceives about the environment. **Observations can be numeric and/or visual**. Numeric observations measure attributes of the environment from the point of view of the agent. For our medic this would be attributes of the battlefield that are visible to it. For most interesting environments, an agent will require several continuous numeric observations. Visual observations, on the other hand, are images generated from the cameras attached to the agent and represent what the agent is seeing at that point in time. **It is common to confuse an agent's observation with the environment (or game) state.** The environment state represents information about the entire scene containing all the game characters. The agents observation, however, only contains information that the agent is aware of and is typically a subset of the environment state. **For example, the medic observation cannot include information about an enemy in hiding that the medic is unaware of.**

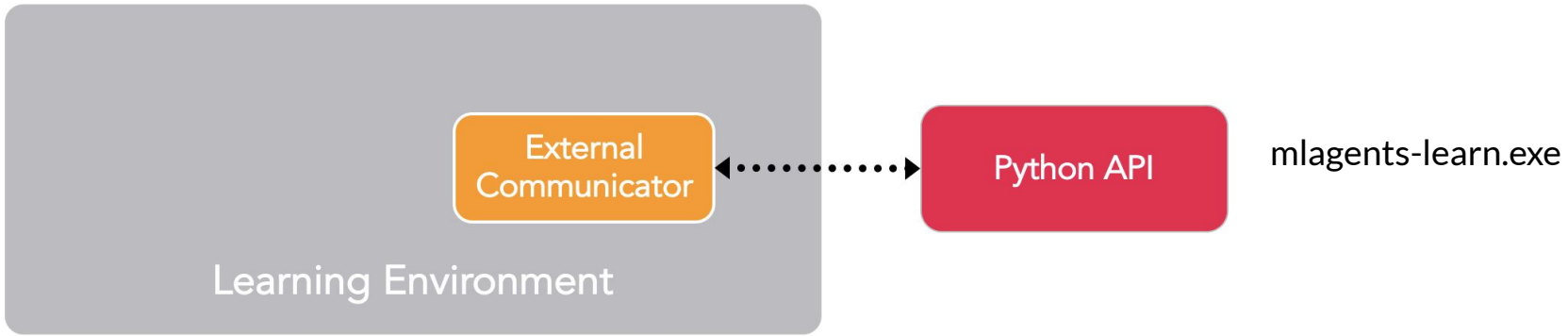
ML Agent Basic

https://github.com/Unity-Technologies/ml-agents/blob/master/docs/ML-Agents-Overview.md

- **Actions** - what actions the medic can take. Similar to observations, **actions can either be continuous or discrete depending on the complexity of the environment and agent.** In the case of the medic, if the environment is a simple grid world where only their location matters, then a discrete action taking on one of **four values (north, south, east, west)** suffices. However, if the environment is more complex and the medic can move freely then using two continuous actions (one for direction and another for speed) is more appropriate.
- **Reward signals** - a scalar value indicating how well the medic is doing. Note that the reward signal need not be provided at every moment, but only when the medic performs an action that is good or bad. For example, it can receive a large negative reward if it dies, a modest positive reward whenever it revives a wounded team member, and a modest negative reward when a wounded team member dies due to lack of assistance. Note that the reward signal is how the objectives of the task are communicated to the agent, so they need to be set up in a manner where maximizing reward generates the desired optimal behavior.

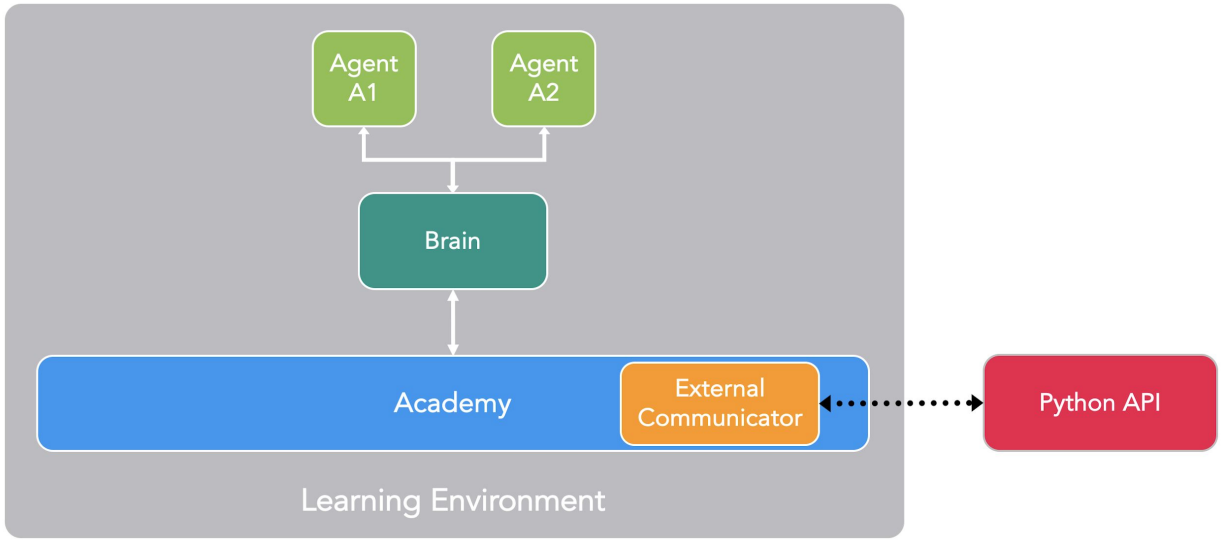
ML Agent Basic

https://github.com/Unity-Technologies/ml-agents/blob/master/docs/ML-Agents-Overview.md



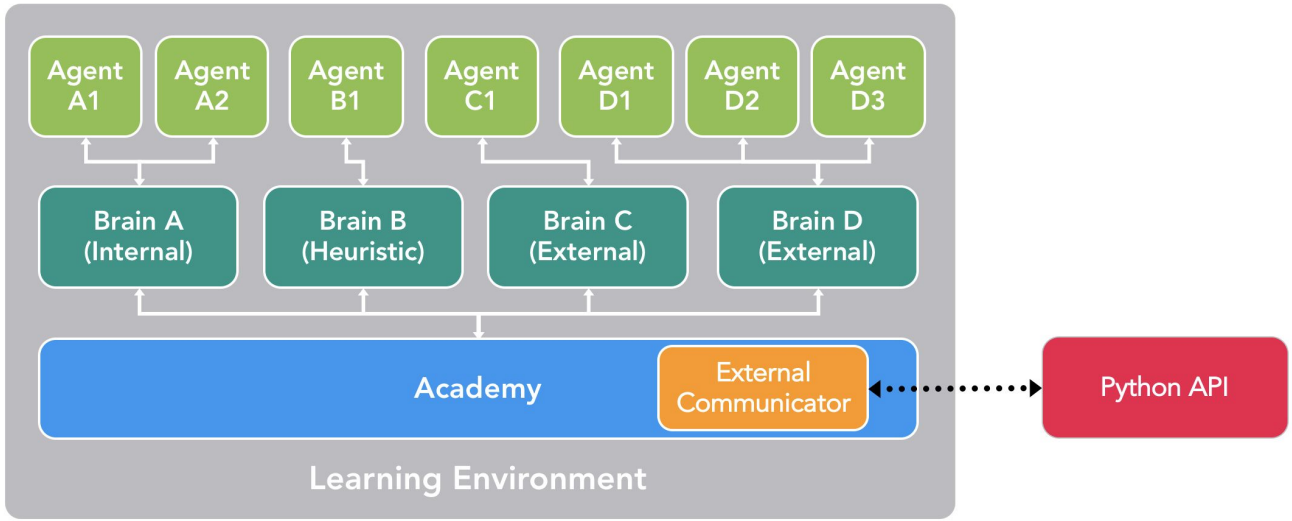
ML Agent Basic

https://github.com/Unity-Technologies/ml-agents/blob/master/docs/ML-Agents-Overview.md



ML Agent Basic

https://github.com/Unity-Technologies/ml-agents/blob/master/docs/ML-Agents-Overview.md



ML Agent Basic

유니티 강좌

https://unity.com/learn

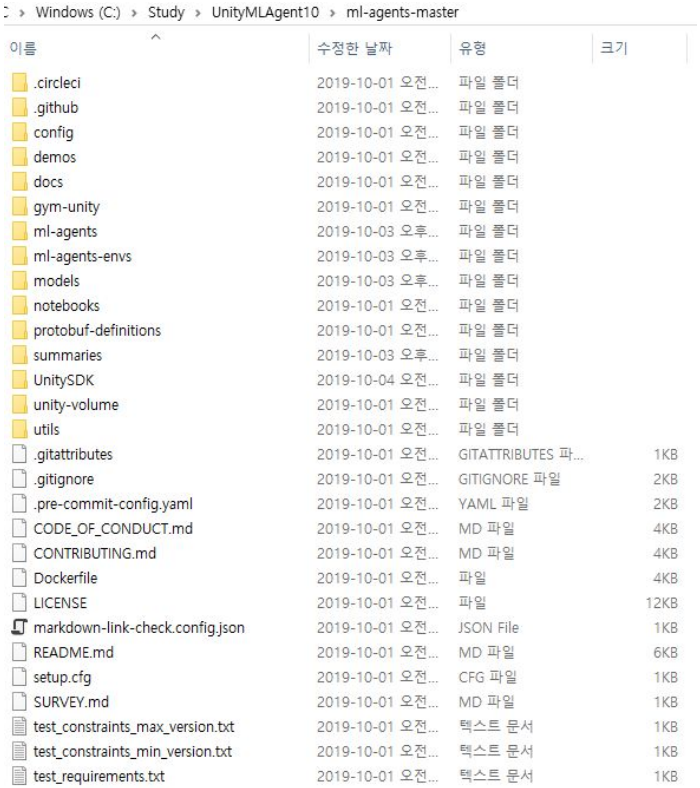
https://online.fastcampus.co.kr/p/dev_online_unitygame (유료)

https://www.edwith.org/unity5_2015_001 (무료, 5.0 버전기준)

https://www.udemy.com/courses/search/?src=ukw&q=%EC%9C%A0%EB%8B%88%ED%8B%B0
(유니티 유료 강좌)

ML Agent Basic

ML Agent directory



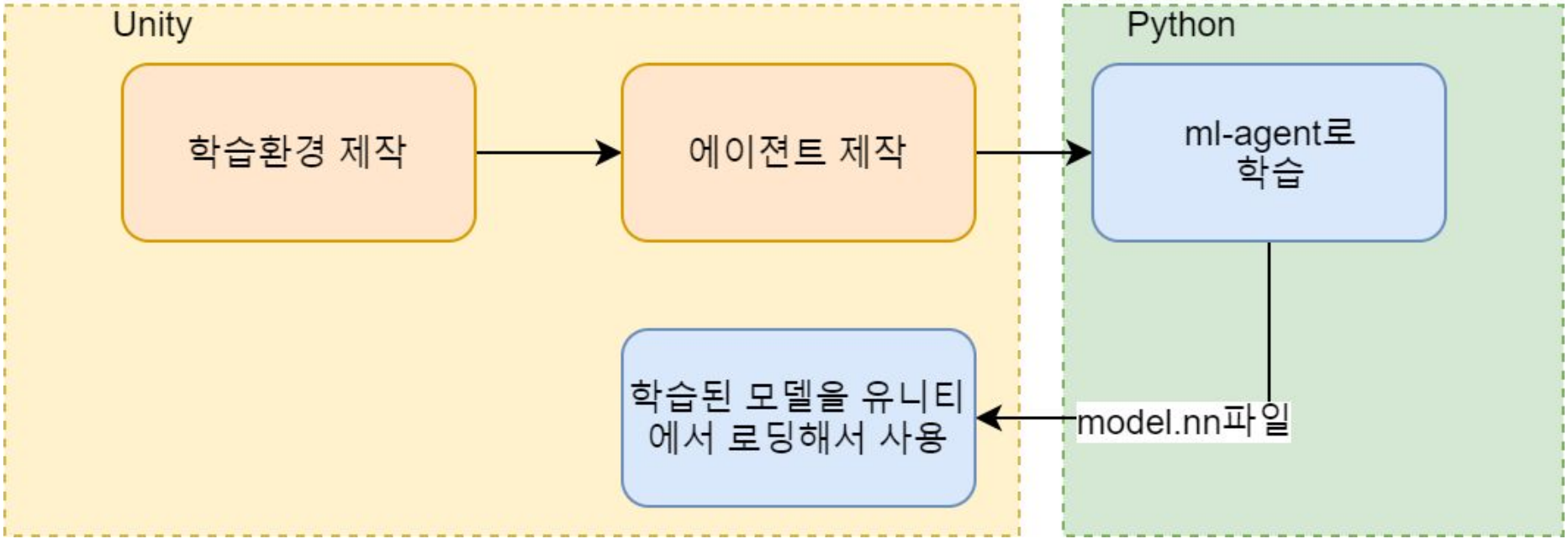
ML Agent directory

- **config : 학습용 config 파일이 저장 (mlagents-learn에서 사용)**
- demos : expert의 demo파일이 저장 , BC와 GAIL용
- docs:문서 파일 (github에서 보는게..)
- ml-agents : 트래이닝 파이썬 코드
- **models : 학습 결과물 + 중간 결과물이 저장되는 디렉토리**
- summaries : tensorboard 파일이 저장
- **UnitySDK : 유니티용 SDK 파일이 저장(샘플 예제도 여기에 있음)**

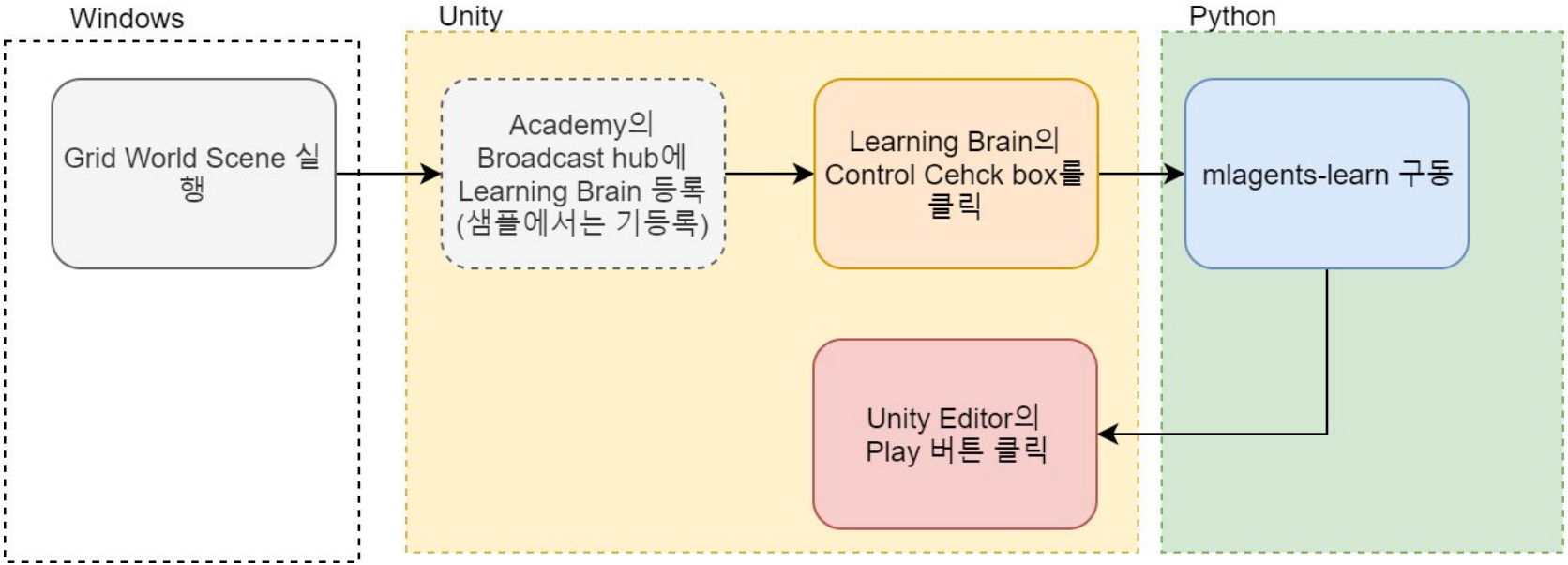
mlagents-learn

- 유니티와 통신해 강화학습을 수행하는 python 학습코드를 수행하는 프로그램
- 샘플 실행을 위해서는 ml agent의 설치 디렉토리에서 실행
- 실행 방법 : mlagents-learn config_path --run-id='실험이름' --train
  EX) mlagents-learn config/trainer_config.yaml --run-id=test --train
- 실행 후 유니티의 Play 버튼을 눌러 유니티 환경을 실행
- 학습의 경과가 화면에 출력됨
- 유니티에서 학습을 종료하면 실험이름.nn파일이 models 폴더에 생성
- 해당 파일을 유니티로 옮겨서 학습 내용 확인 가능

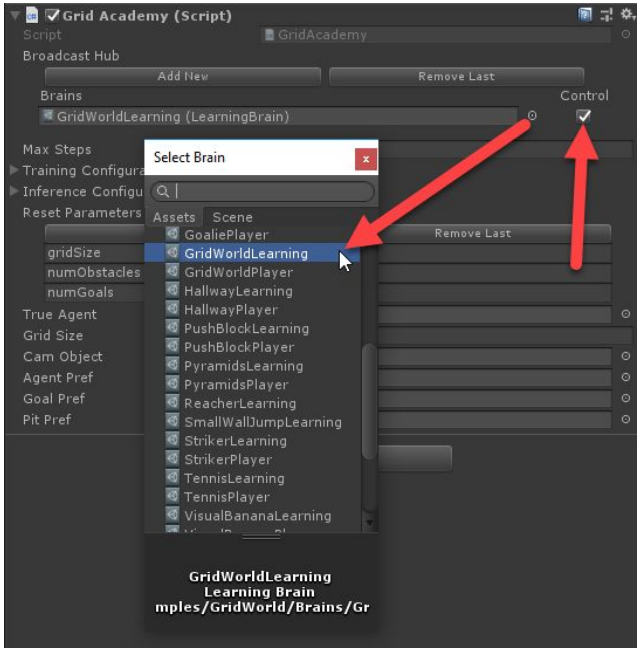
전체 학습순서


Unity
학습환경 제작
에이전트 제작
학습된 모델을 유니티에서 로딩해서 사용
Python
ml-agent로 학습
model.nn파일

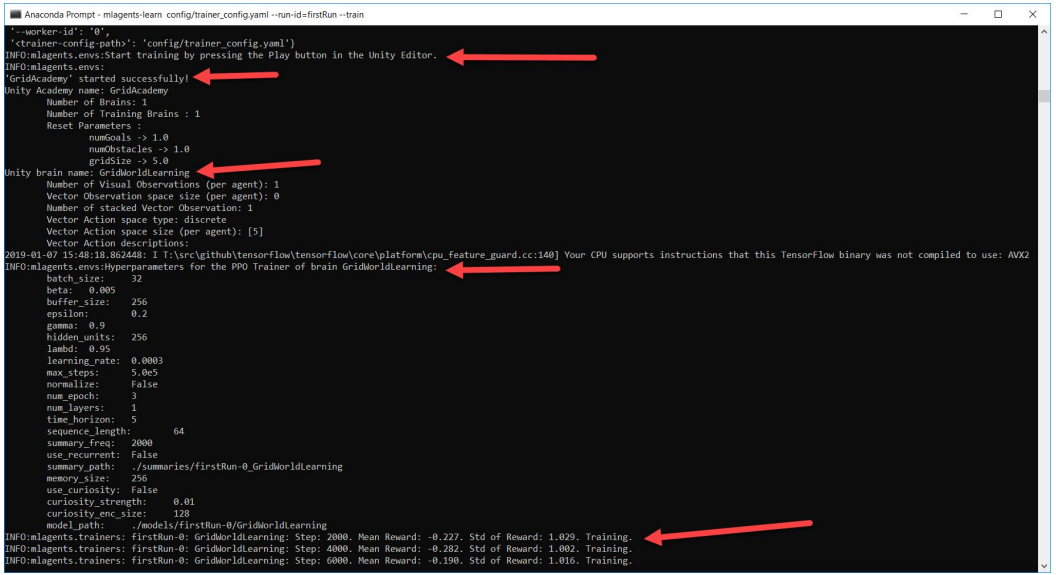
Grid world - 학습 시작



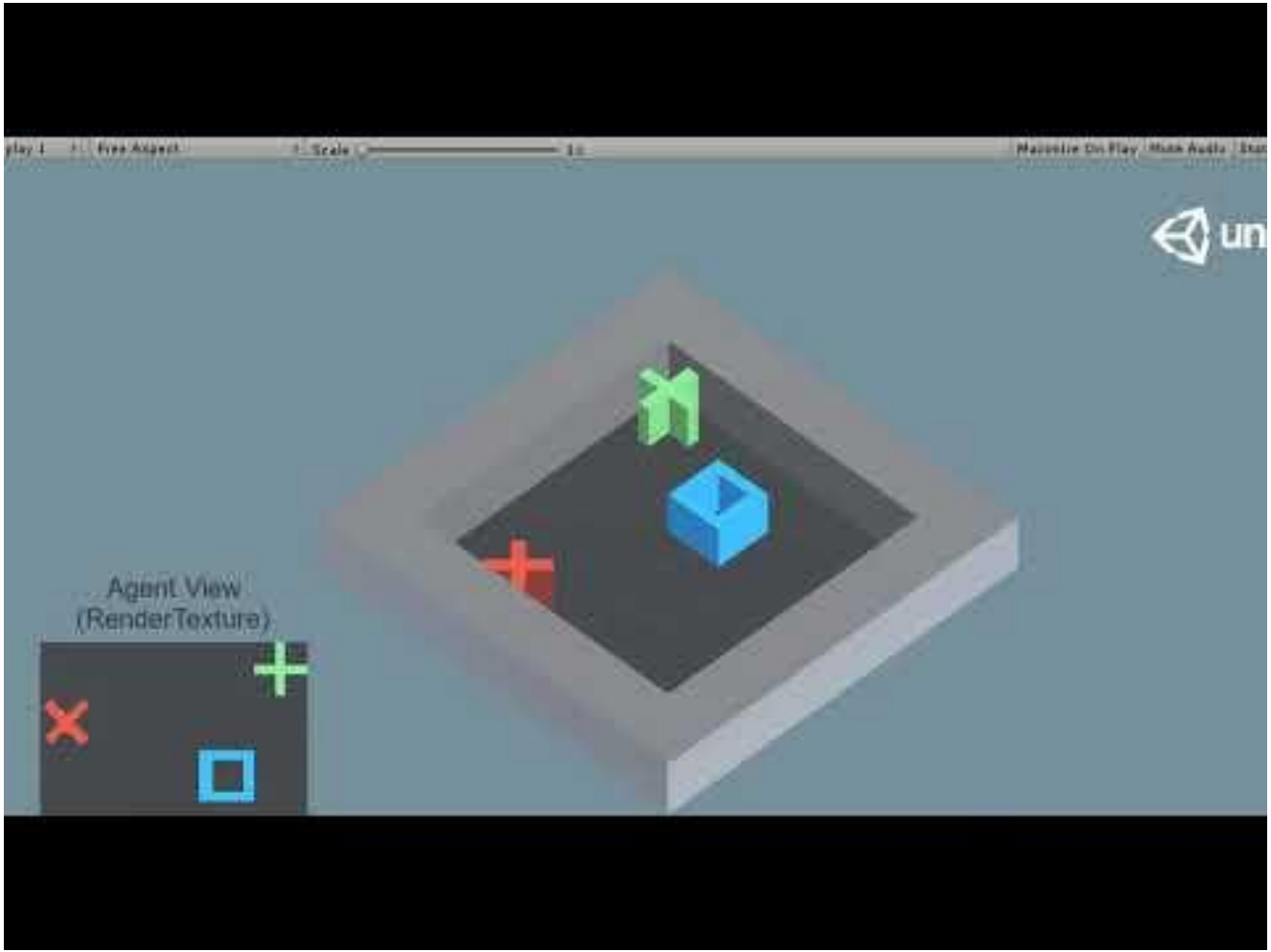
Grid world - 학습 시작



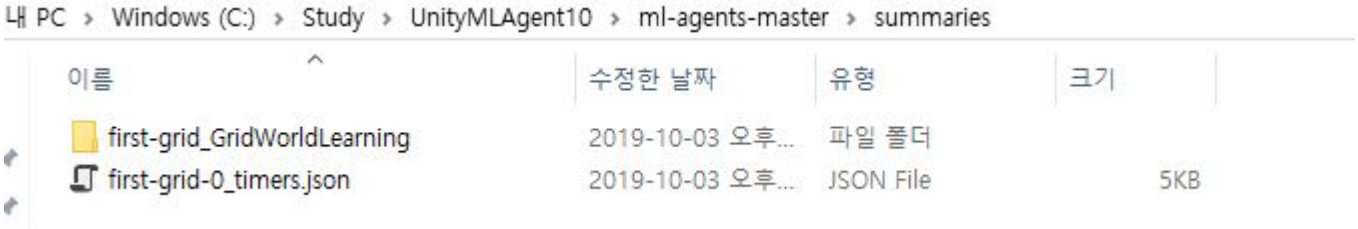

```
mlagents-learn config/trainer_config.yaml --run-id=firstRun --train
```

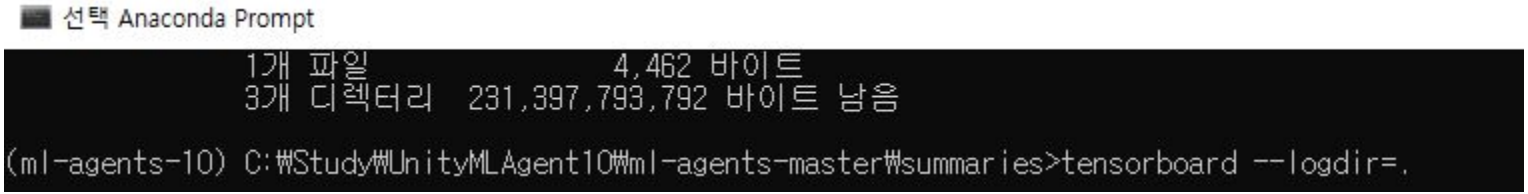


https://www.youtube.com/watch?v=ZMV4o0Tu530



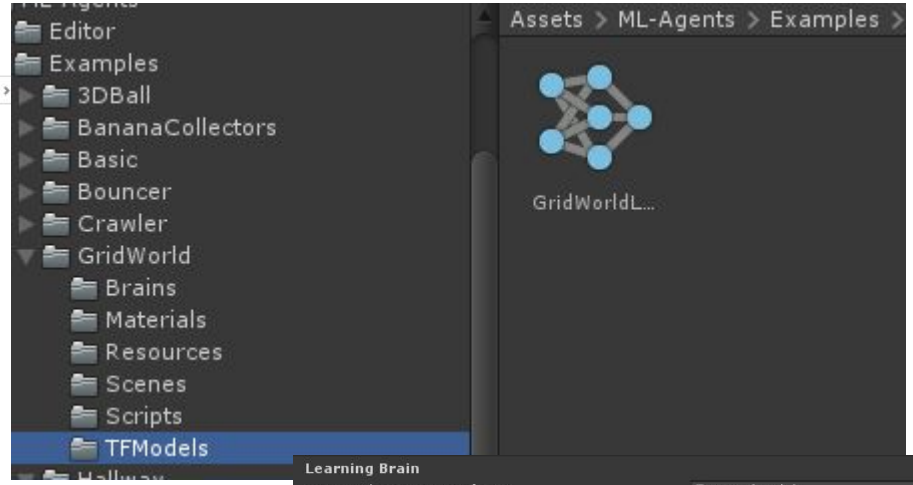
Grid world - 학습 확인

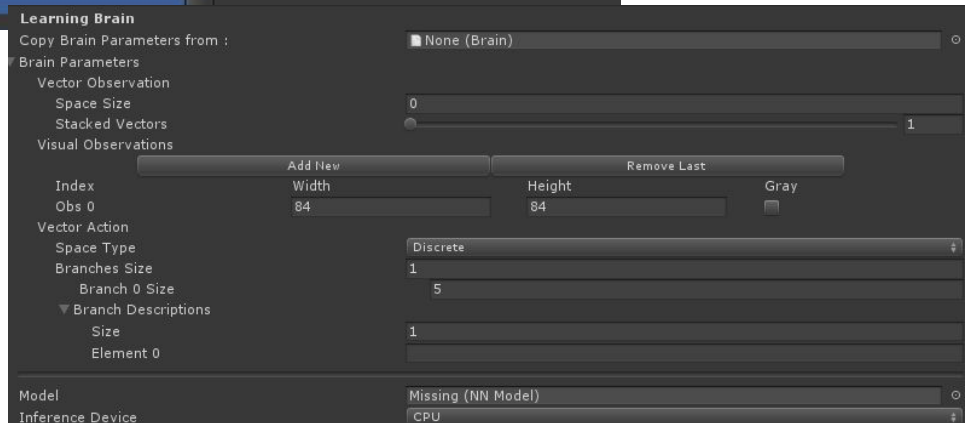




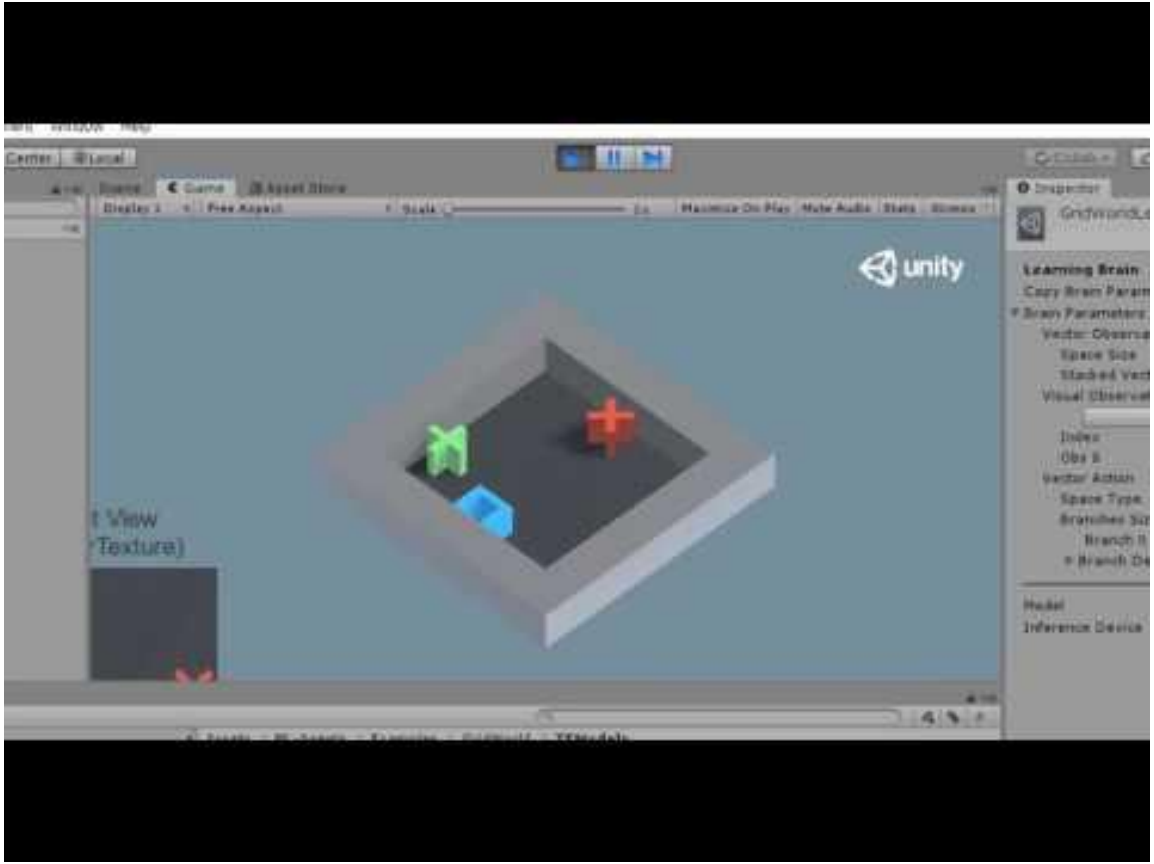
Grid world - 학습 확인

Grid world - 학습 적용

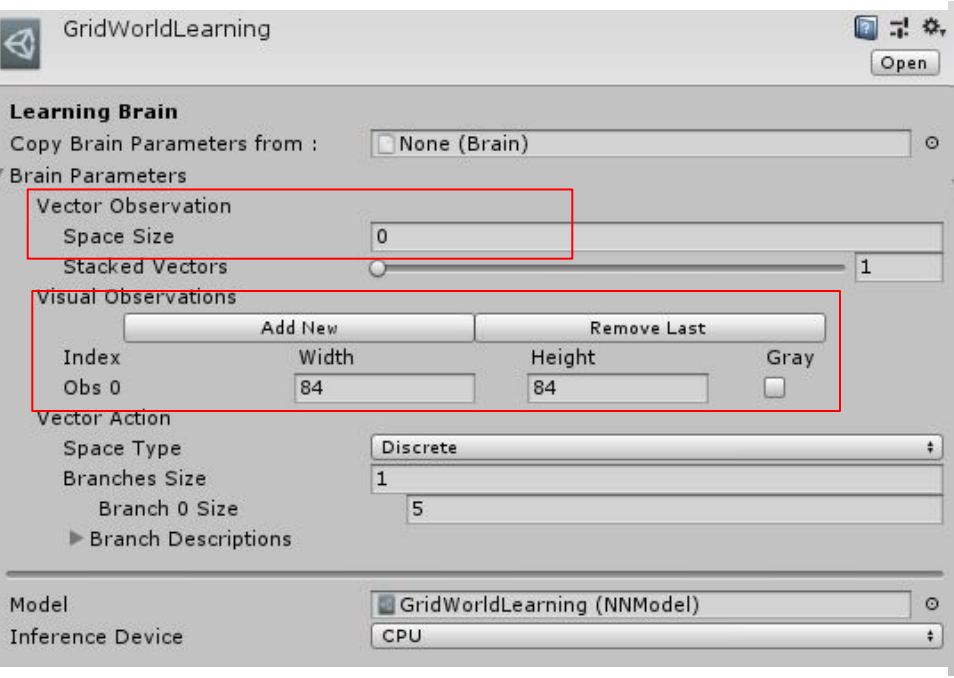


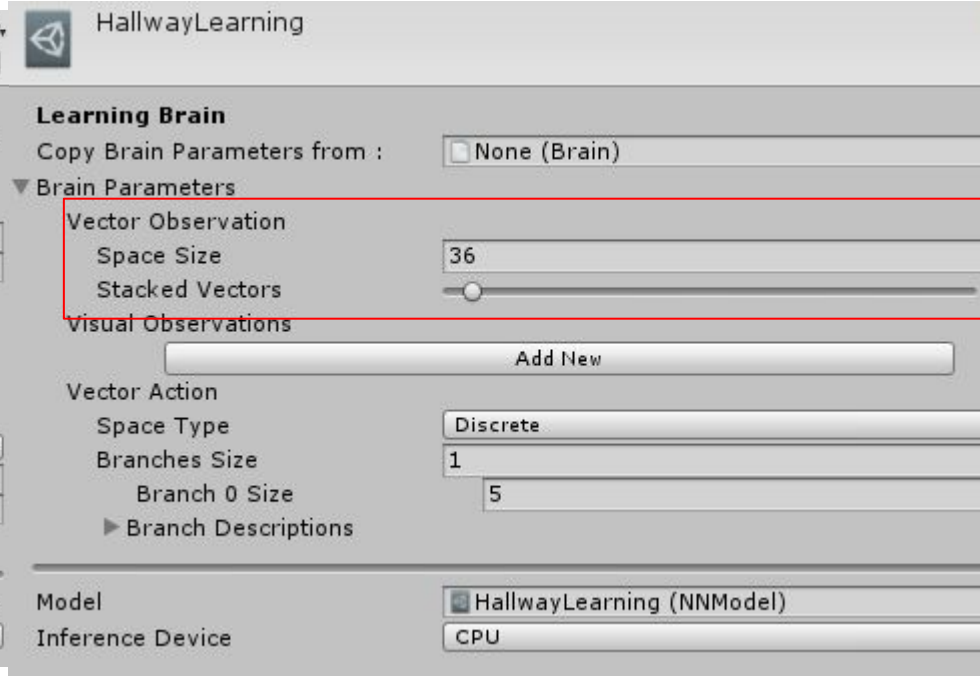


https://youtu.be/rd508meIJZo



Understanding Brain





Brain parameters

`Brain Parameters` - Define vector observations, visual observation, and vector actions for the Brain.

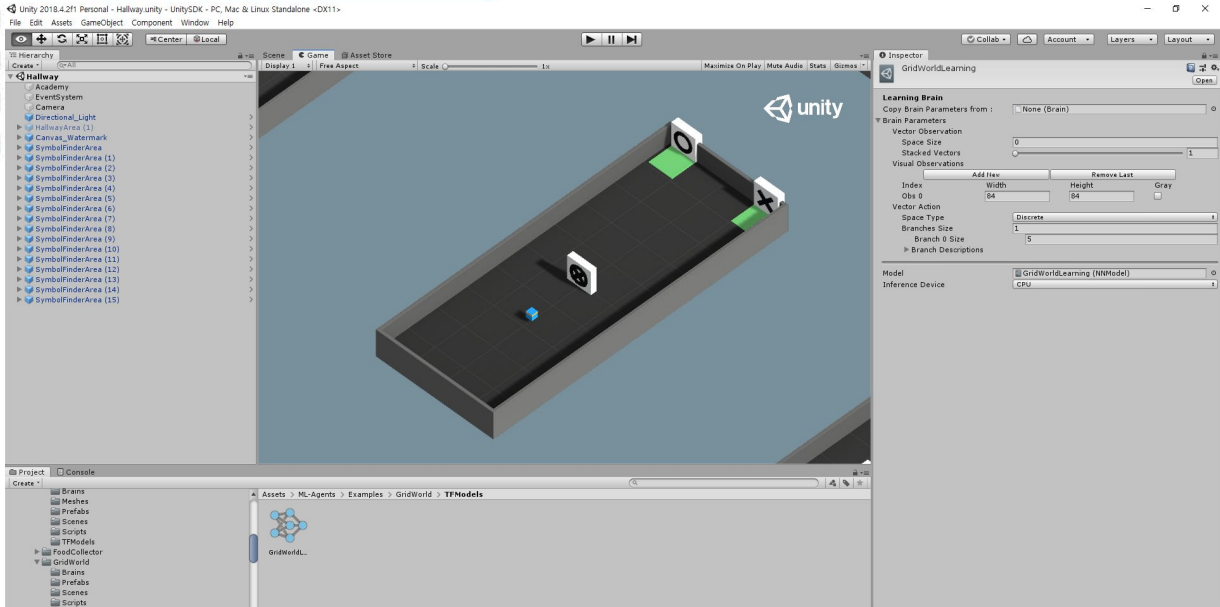
- `Vector Observation`
  - `Space Size` - Length of vector observation for Brain.
  - `Stacked Vectors` - The number of previous vector observations that will be stacked and used collectively for decision making. This results in the effective size of the vector observation being passed to the Brain being: *Space Size* x *Stacked Vectors*.
- `Visual Observations` - Describes height, width, and whether to grayscale visual observations for the Brain.
- `Vector Action`
  - `Space Type` - Corresponds to whether action vector contains a single integer (Discrete) or a series of real-valued floats (Continuous).
  - `Space Size` (Continuous) - Length of action vector for Brain.
  - `Branches` (Discrete) - An array of integers, defines multiple concurrent discrete actions. The values in the `Branches` array correspond to the number of possible discrete values for each action branch.
  - `Action Descriptions` - A list of strings used to name the available actions for the Brain.

https://github.com/Unity-Technologies/ml-agents/blob/master/docs/Learning-Environment-Design-Brains.md

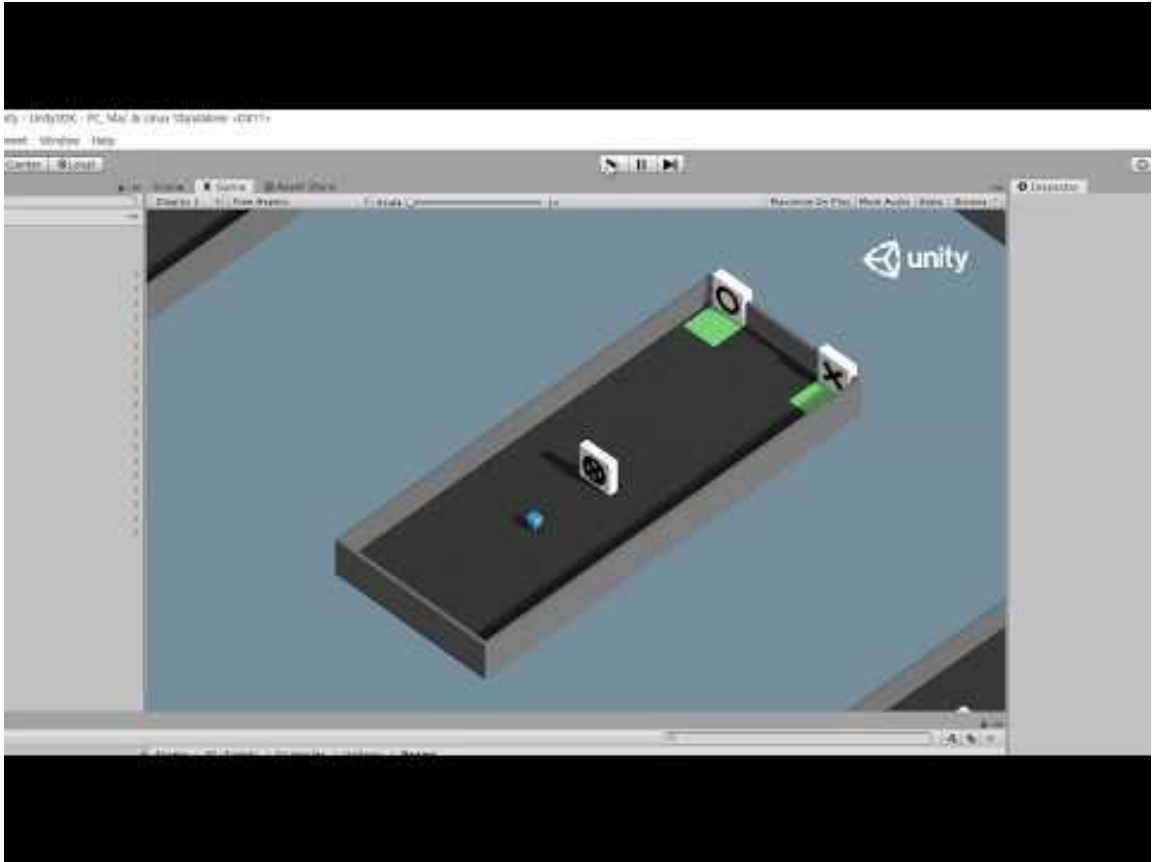
Hallway 환경 열기

C › Windows (C:) › Study › UnityMLAgent10 › ml-agents-master › UnitySDK › Assets › ML-Agents › Examples › Hallway › Scenes

이름	수정한 날짜	유형	크기
Hallway.unity	2019-10-01 오전...	Unity scene file	49KB
Hallway.unity.meta	2019-10-01 오전...	META 파일	1KB
HallwayIL.unity	2019-10-01		
HallwayIL.unity.meta	2019-10-01		
VisualHallway.unity	2019-10-01		
VisualHallway.unity.meta	2019-10-01		

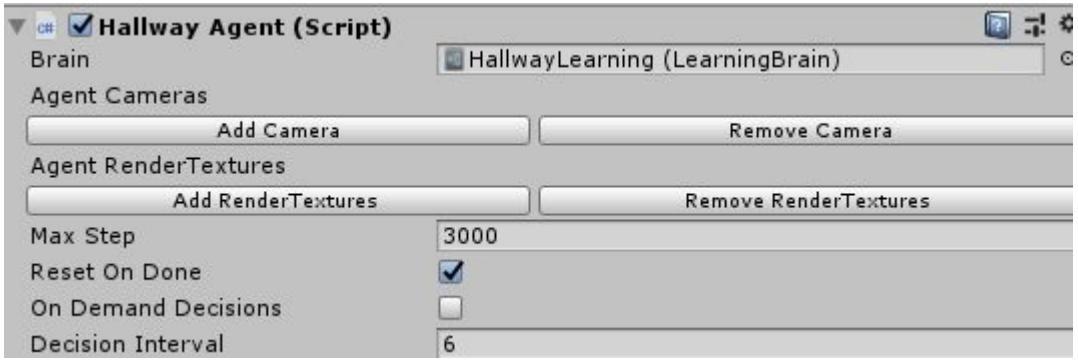


Hallway overview



Agent

https://github.com/Unity-Technologies/ml-agents/blob/master/docs/Learning-Environment-Design-Agents.md



Max step : 에피소드 최대 스탭 , 3000 스탭이후에도 끝나지 않으면 강제 종료
Reset on Done : Done일때 에피소드를 끝낼것인가의 여부
Decision Interval : 의사 결정 빈도 , 1/60 초 단위 6은 1/10초마다 의사 결정을 하겠다는것
On Demand Decision : 의사 결정 빈도를 정하지 않고 환경에서 Agent.RequestDecision() 함수를 호출할때만 의사 결정 하겠다는 의미

Collecte Observation

- Visual Observation인 경우는 Camera만 지정하면 됨
- Vector Observation의 경우는 Agent class의 CollectObservations 함수에 AddVectorObs함수를 이용해 넣어주면 됨

HallwayAgent.cs RayPerception.cs

HallwayAgent GoalScoredSwapGroundMaterial(Material mat, float time)

```
public override void CollectObservations()
{
    if (useVectorObs)
    {
        var rayDistance = 12f;
        float[] rayAngles = { 20f, 60f, 90f, 120f, 160f };
        string[] detectableObjects = { "symbol_O_Goal", "symbol_X_Goal", "symbol_O", "symbol_X", "wall" };
        AddVectorObs(GetStepCount() / (float)agentParameters.maxStep);
        AddVectorObs(m_RayPer.Perceive(rayDistance, rayAngles, detectableObjects, 0f, 0f));
    }
}
```

Collecte Observation

총 36개의 Observation을 정의

AddVectorObs(GetStepCount() / (float)agentParameters.maxStep);
현재 Step/ Max Step 값 -> float값 한개

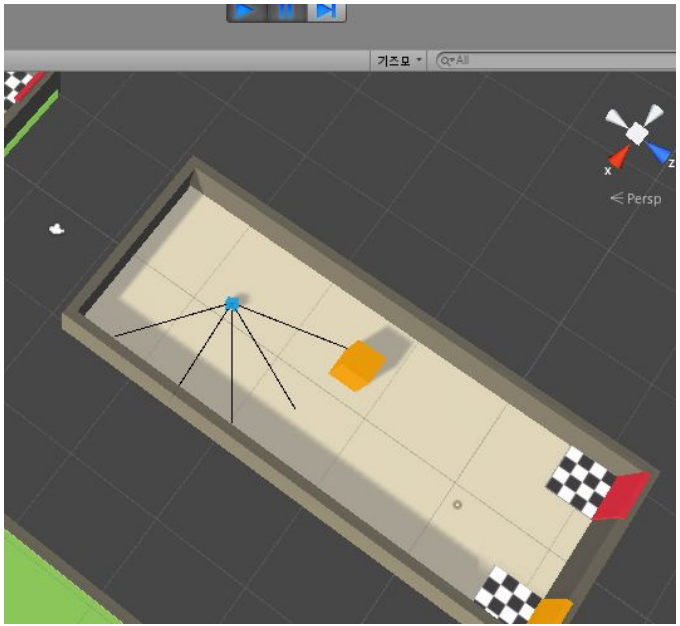
AddVectorObs(rayPer.Perceive(rayDistance, rayAngles, detectableObjects, 0f, 0f));
-> 35개의 float list

rayPer.Perceive(rayDistance, rayAngles, detectableObjects, 0f, 0f) //float[35]

Ray perception

유니티의 raycast함수를 사용.

일종의 직사광선을 보내 처음 검출되는 GameObject의 tag를 반환.



https://docs.unity3d.com/ScriptReference/Physics.Raycast.html

Ray perception

ray perception class의 Perceive함수를 재정의 해줘야 함.

```
/// <param name="rayDistance">Radius of rays</param>
/// <param name="rayAngles">Angles of rays (starting from (1,0) on unit circle).</param>
/// <param name="detectableObjects">List of tags which correspond to object types agent can see</param>
/// <param name="startOffset">Starting height offset of ray from center of agent.</param>
/// <param name="endOffset">Ending height offset of ray from center of agent.</param>
public override List<float> Perceive(float rayDistance,
    float[] rayAngles, string[] detectableObjects,
    float startOffset, float endOffset)
{
    perceptionBuffer.Clear();
    // For each ray sublist stores categorical information on detected object
    // along with object distance.
```

Ray perception

rayAngles만큼 루프를 돌고 에디터인 경우 디버깅을 위해 직선을 그려줌.

```
foreach (float angle in rayAngles)
{
    endPosition = transform.TransformDirection(
        PolarToCartesian(rayDistance, angle));
    endPosition.y = endOffset;
    if (Application.isEditor)
    {
        Debug.DrawRay(transform.position + new Vector3(0f, startOffset, 0f),
            endPosition, Color.black, 0.01f, true);
    }
```

Ray perception

- 디텍트 가능한 Object의 tag를 인자로 주고 sphereCast함수를 호출해 ray cast를 한다.

- subList를 디텍트가능한 오브젝트 크기 +2 만큼 만듬

- 디텍트된 오브젝트에 1을 주고 아무것도 디텍트 되지 않으면 마지막 벡터에 1을 줌.

- 디텍트된 거리를 정규화해서 list의 끝에 넣어줌.

```
float[] subList = new float[detectableObjects.Length + 2];
if (Physics.SphereCast(transform.position +
                       new Vector3(0f, startOffset, 0f), 0.5f,
    endPosition, out hit, rayDistance))
{
    for (int i = 0; i < detectableObjects.Length; i++)
    {
        if (hit.collider.gameObject.CompareTag(detectableObjects[i]))
        {
            subList[i] = 1;
            subList[detectableObjects.Length + 1] = hit.distance / rayDistance;
            break;
        }
    }
}
else
{
    subList[detectableObjects.Length] = 1f;
}

perceptionBuffer.AddRange(subList);
```

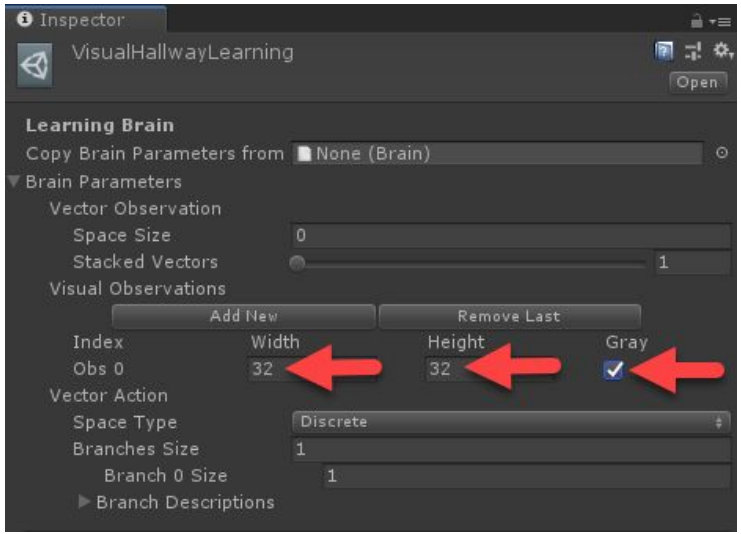
ray perception vector

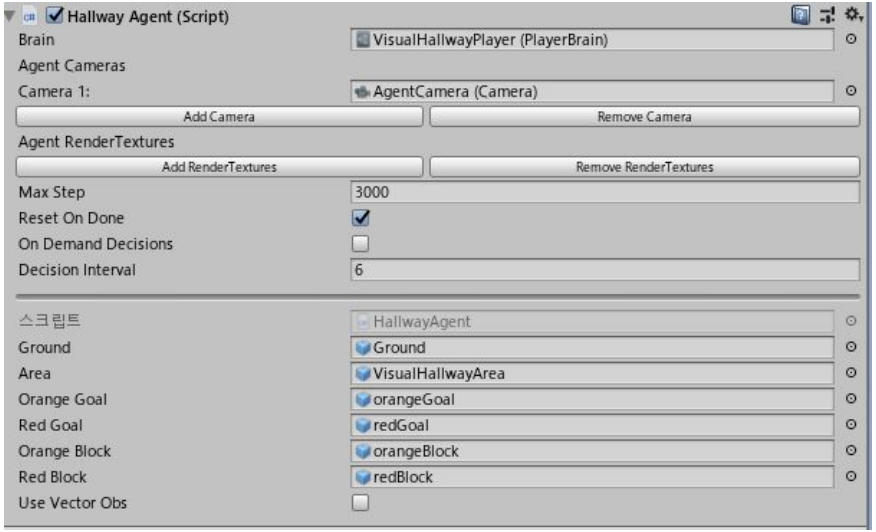
Collecte Observation

detectableObjects가 5개이므로 5+2 = 7 * 5개의 angle = 35.

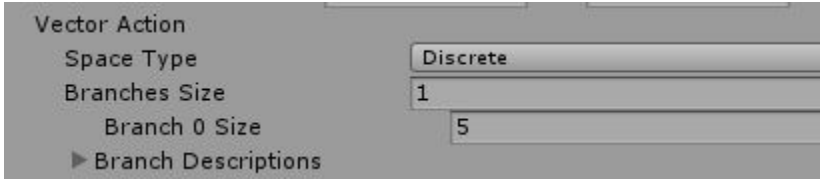
```
public override void CollectObservations()
{
    if (useVectorObs)
    {
        float rayDistance = 12f;
        float[] rayAngles = { 20f, 60f, 90f, 120f, 160f };
        string[] detectableObjects = { "orangeGoal", "redGoal", "orangeBlock", "redBlock", "wall" };
        AddVectorObs(GetStepCount() / (float)agentParameters.maxStep);
        AddVectorObs(rayPer.Perceive(rayDistance, rayAngles, detectableObjects, 0f, 0f));
    }
}
```

Visual Observation





Action




```
public override void AgentAction(float[] vectorAction, string textAction)
{
    AddReward(-1f / agentParameters.maxStep);
    MoveAgent(vectorAction);
}
```

HallwayAgent.cs

Action

```
public void MoveAgent(float[] act)
{
    var dirToGo = Vector3.zero;
    var rotateDir = Vector3.zero;
{
    var action = Mathf.FloorToInt(act[0]);
    switch (action)
    {
        case 1:
            dirToGo = transform.forward * 1f;
            break;
        case 2:
            dirToGo = transform.forward * -1f;
            break;
        case 3:
            rotateDir = transform.up * 1f;
            break;
        case 4:
            rotateDir = transform.up * -1f;
            break;
    }
}
transform.Rotate(rotateDir, Time.deltaTime * 150f);
m_AgentRb.AddForce(dirToGo * m_Academy.agentRunSpeed, ForceMode.VelocityChange);
```

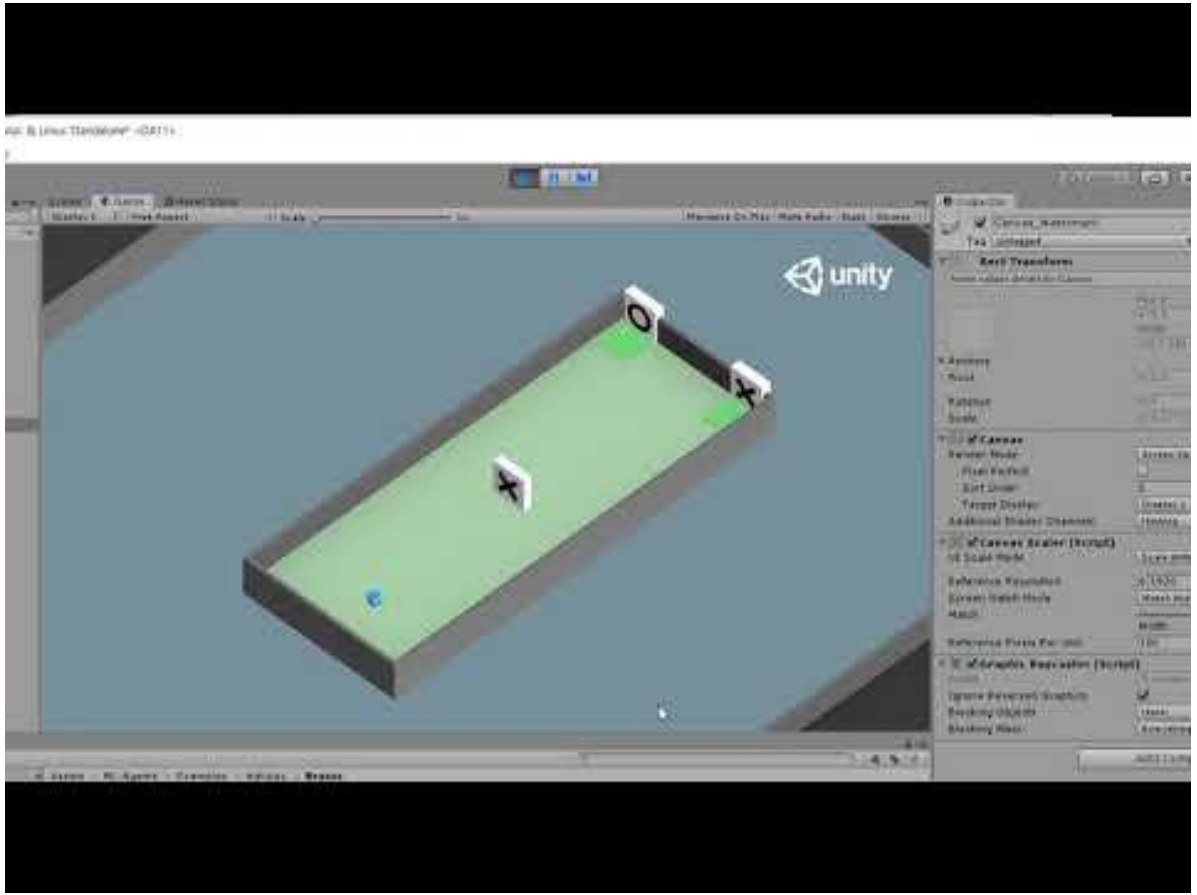
학습 시키기

```
mlagents-learn config/trainer_config.yaml --run-id=hallway --train
```

중간에 끊었다가 이어서 학습하고 싶을때

```
mlagents-learn config/trainer_config.yaml --run-id=hallway --train --load
```

https://youtu.be/Nn_ONN1TzAc



config/trainer_config.yaml

trainer_config.yaml × gail_config.yaml ×

```yaml
default:
    trainer: ppo
    batch_size: 1024
    beta: 5.0e-3
    buffer_size: 10240
    epsilon: 0.2
    hidden_units: 128
    lambd: 0.95
    learning_rate: 3.0e-4
    learning_rate_schedule: linear
    max_steps: 5.0e4
    memory_size: 256
    normalize: false
    num_epoch: 3
    num_layers: 2
    time_horizon: 64
    sequence_length: 64
    summary_freq: 1000
    use_recurrent: false
    vis_encode_type: simple
    reward_signals:
        extrinsic:
            strength: 1.0
            gamma: 0.99
```

```yaml
HallwayLearning:
    use_recurrent: true
    sequence_length: 64
    num_layers: 2
    hidden_units: 128
    memory_size: 256
    beta: 1.0e-2
    num_epoch: 3
    buffer_size: 1024
    batch_size: 128
    max_steps: 5.0e5
    summary_freq: 1000
    time_horizon: 64
```

config/trainer_config.yaml

https://github.com/Unity-Technologies/ml-agents/blob/master/docs/localized/KR/docs/Training-PPO.md

Number of Layers

`num_layers` 는 관측 입력 후 혹은 시각적 관측 (Visual Observation)의 CNN 인코딩 이후 몇개의 은닉층 (Hidden Layer)을 사용할지 결정합니다. 간단한 문제에서는 적은 수의 층을 사용하여 빠르고 효율적으로 학습해야합니다. 복잡한 제어 문제에서는 많은 층을 사용할 필요가 있습니다.

일반적인 범위: `1` - `3`

Hidden Units

`hidden_units` 은 인공신경망의 각 완전연결층 (Fully Connected Layer)에 몇개의 유닛을 사용할지 결정합니다. 최적의 행동이 관측 입력의 간단한 조합으로 결정되는 단순한 문제에 대해서는 이 값을 작게 설정합니다. 최적의 행동이 관측 입력의 복잡한 관계에 의해 결정되는 어려운 문제에 대해서는 이 값을 크게 설정합니다.

일반적인 범위: `32` - `512`

Reward

```
void OnCollisionEnter(Collision col)
{
    if (col.gameObject.CompareTag("symbol_O_Goal") || col.gameObject.CompareTag("symbol_X_Goal"))
    {
        if ((m_Selection == 0 && col.gameObject.CompareTag("symbol_O_Goal")) ||
            (m_Selection == 1 && col.gameObject.CompareTag("symbol_X_Goal")))
        {
            SetReward(1f);
            StartCoroutine(GoalScoredSwapGroundMaterial(m_Academy.goalScoredMaterial, 0.5f));
        }
        else
        {
            SetReward(-0.1f);
            StartCoroutine(GoalScoredSwapGroundMaterial(m_Academy.failMaterial, 0.5f));
        }
        Done();
    }
}
```

Reward function

```
public override void AgentAction(float[] vectorAction, string textAction)
{
    AddReward(-1f / agentParameters.maxStep);
    MoveAgent(vectorAction);
}
```

AddReward : 에피소드 중간에 반복적으로 줄 수 있음 , AgentAction 함수는 1/60 초단위로 불려짐에 유의 , 예제에서 maxStep값은 3000

SetReward : 에피소드가 끝나는 시점에 (이기고 지고) 줌, 이를 호출하고 Done()함수를 호출하면 에피소드 종료

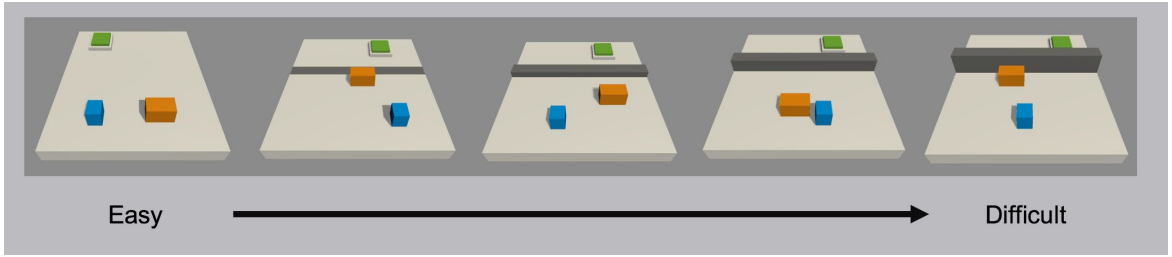
Sparsity of rewards

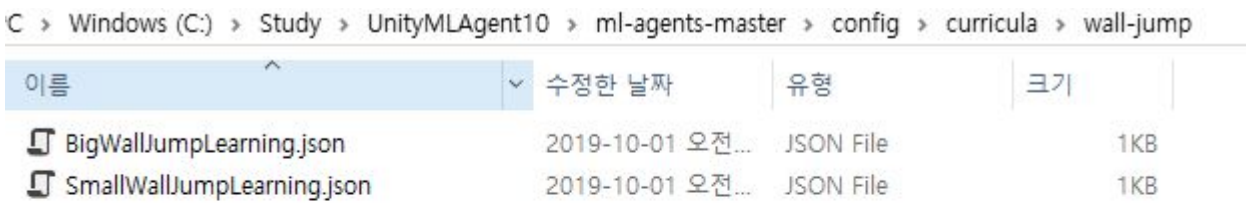
- 보상은 일종의 신호로 -, + 상관없이 더 자주 받게 되면 학습에 도움이 된다.
- 보상이 드물게 주어지는 경우는 학습이 안되거나 더디게 되어 다른 방법을 써야 한다. -> 커리큘럼 러닝

커리큘럼 러닝

점진적으로 문제를 어렵거나 복잡하게 만들어 쉬운것 부터 어려운것까지 학습하는 방법.

- 리워드를 더 자주 얻게 만드는 환경 -> 리워드를 적게 얻는 환경으로 변경






```
mlagents-learn config/trainer_config.yaml--curriculum=config/curricula/wall-jump/ --run-id=wall-jump-curriculum
--train
```

BigWallJumpLearning.json

```
{
    "measure" : "progress",
    "thresholds" : [0.1, 0.3, 0.5],
    "min_lesson_length": 100,
    "signal_smoothing" : true,
    "parameters" :
    {
        "big_wall_min_height" : [0.0, 4.0, 6.0, 8.0],
        "big_wall_max_height" : [4.0, 7.0, 8.0, 8.0]
    }
}
```

브레인 이름.json 파일

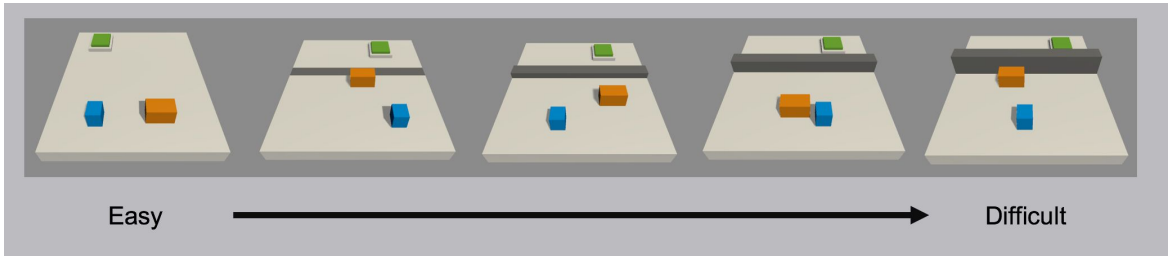
커리큘럼 러닝

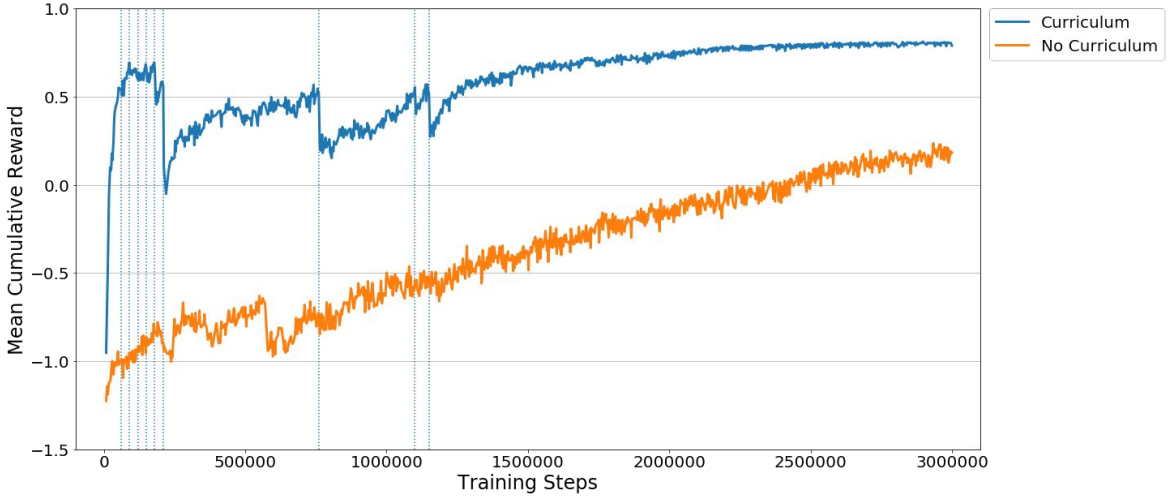
https://github.com/Unity-Technologies/ml-agents/blob/master/docs/Training-Curriculum-Learning.md

- `measure` - What to measure learning progress, and advancement in lessons by.
  - `reward` - Uses a measure received reward.
  - `progress` - Uses ratio of steps/max_steps.
- `thresholds` (float array) - Points in value of `measure` where lesson should be increased.
- `min_lesson_length` (int) - The minimum number of episodes that should be completed before the lesson can change. If `measure` is set to `reward`, the average cumulative reward of the last `min_lesson_length` episodes will be used to determine if the lesson should change. Must be nonnegative.

  Important: the average reward that is compared to the thresholds is different than the mean reward that is logged to the console. For example, if `min_lesson_length` is `100`, the lesson will increment after the average cumulative reward of the last `100` episodes exceeds the current threshold. The mean reward logged to the console is dictated by the `summary_freq` parameter in the trainer configuration file.
- `signal_smoothing` (true/false) - Whether to weight the current progress measure by previous values.
  - If `true`, weighting will be 0.75 (new) 0.25 (old).
- `parameters` (dictionary of key:string, value:float array) - Corresponds to Academy reset parameters to control. Length of each array should be one greater than number of thresholds.

커리큘럼 러닝


Easy
Difficult


Curriculum
No Curriculum
Mean Cumulative Reward
1.0
0.5
0.0
−0.5
−1.0
−1.5
0
500000
1000000
1500000
2000000
2500000
3000000
Training Steps

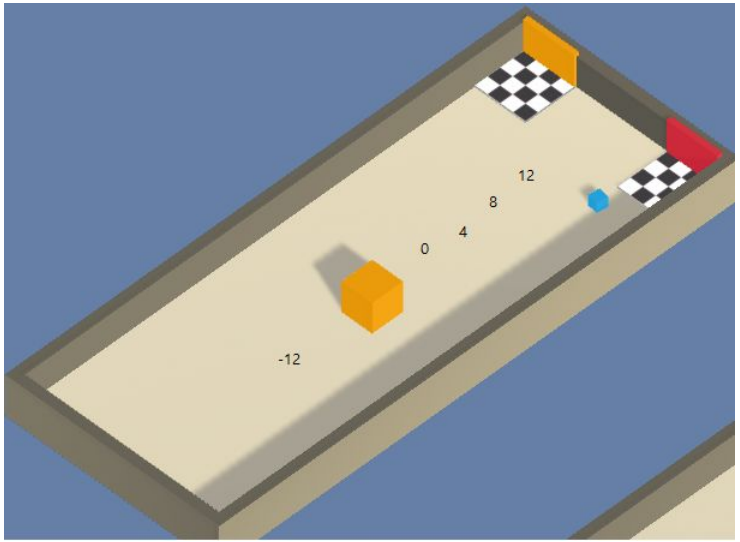
Back play

게임의 후반부 부터 플레이 하는 커리큘럼 러닝 방법의 일종

특정 지점이나 시점부터 플레이 가능한 환경에서 사용가능

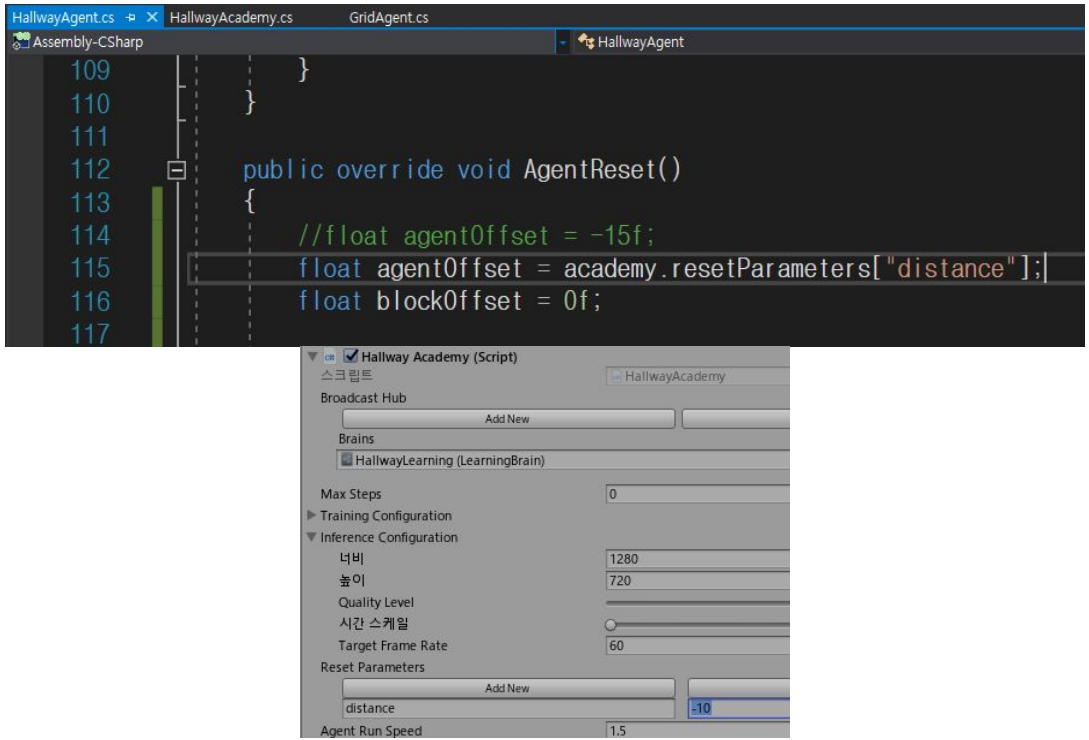
유니티에서 커리큘럼 러닝을 이용해서 사용 가능

Back play - hallway




```
{
    "measure" : "rewards",
    "thresholds" : [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7],
    "min lesson length": 100,
    "signal smoothing" : true,
    "parameters" :
    {
        "distance" : [12, 8, 4, 2, -2, -4, -8, -12]
    }
}
```

Back play - hallway



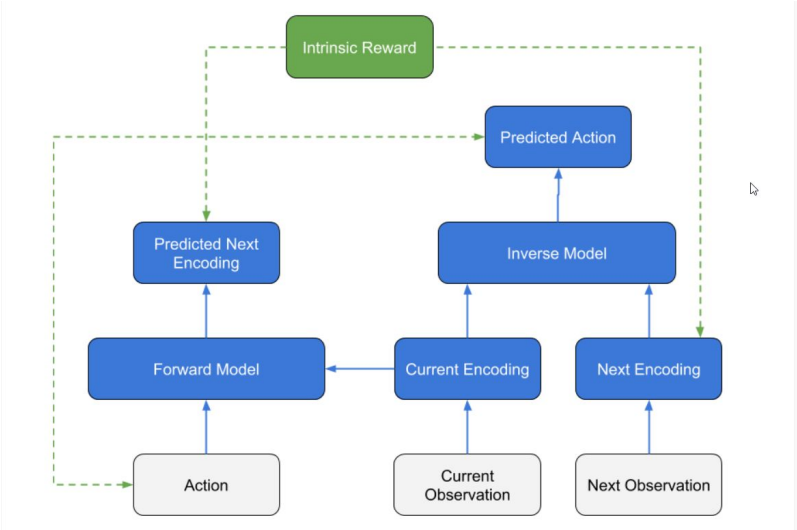
Curiosity Driven Exploration by Self-Supervised Prediction

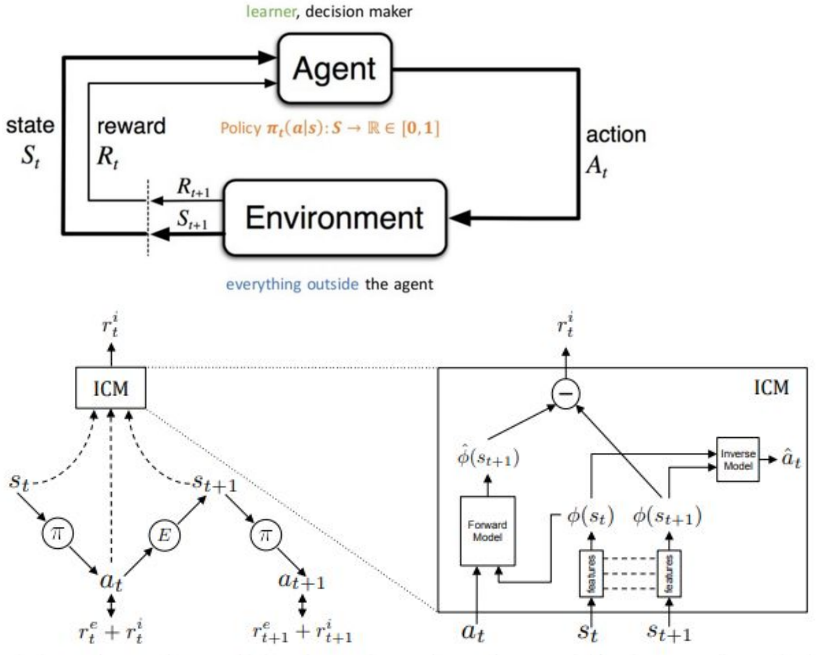
ICML 2017

Deepak Pathak, Pulkit Agrawal, Alexei Efros, Trevor Darrell
UC Berkeley

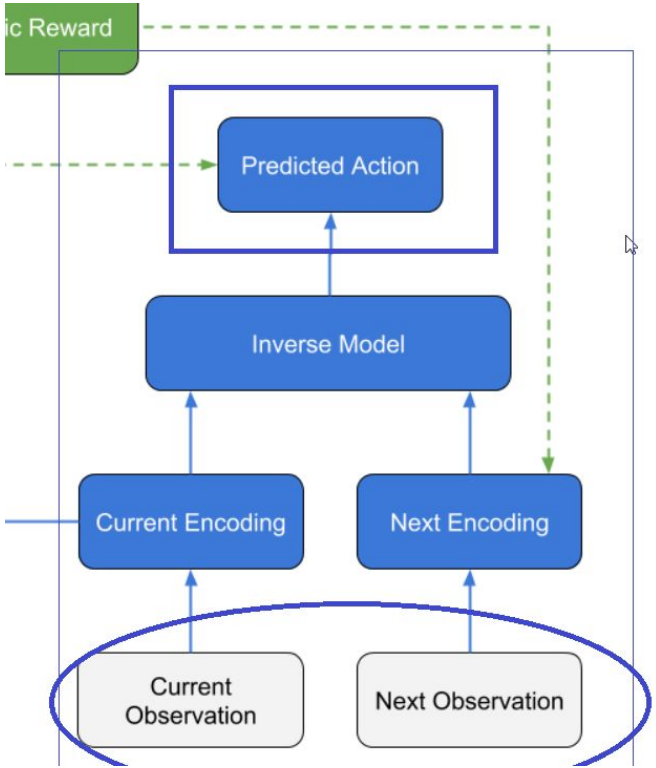
Curiosity learning

Curiosity-driven Exploration by **Self-supervised** Prediction



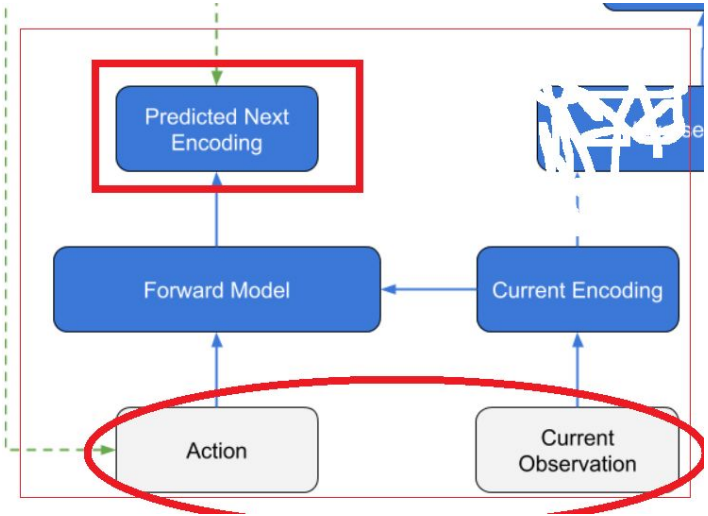


Curiosity learning - 모듈 A



- 모듈 A의 인코더가 학습하는 것인 액션을 추출하기 위해서 환경에서 주목해야 하는 부분의 정보를 학습하게 됨.
- 학습하면 학습할수록 Observation(t , t+1) 로부터 Action을 추측을 잘하게 만드는 Encoder를 얻게 됨.

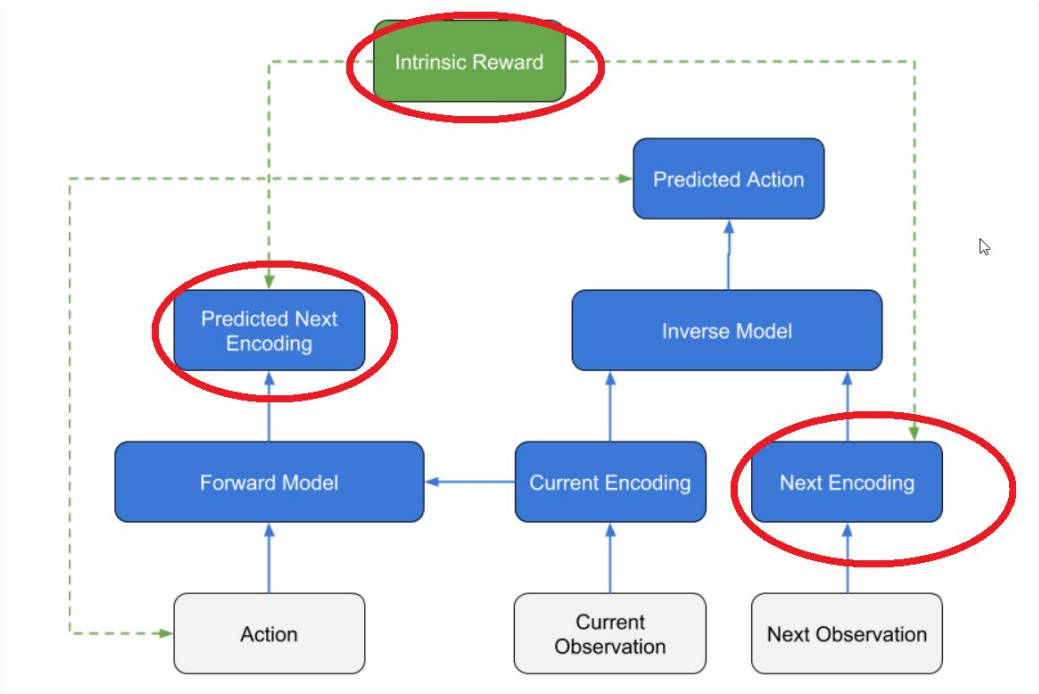
Curiosity learning - 모듈 B



모듈B의 Forward model은 얼마나 현재 학습상태에서 환경에 대해 예측 가능한지에 해당하는 모델. 즉 환경에 대한 이해를 나타낸다고 할 수 있음.

예측한 인코딩된 다음 상태 - 실제 인코딩된 다음 상태가 해당 상태가 얼마나 흥미로운지에 대한 텀이 됨 = 호기심

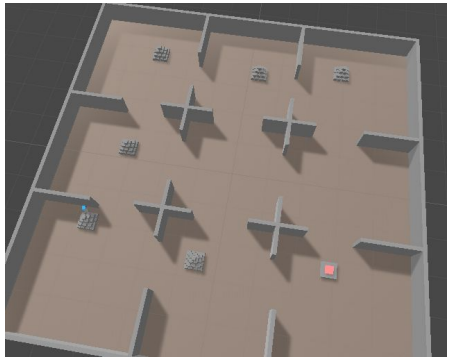
Curiosity learning

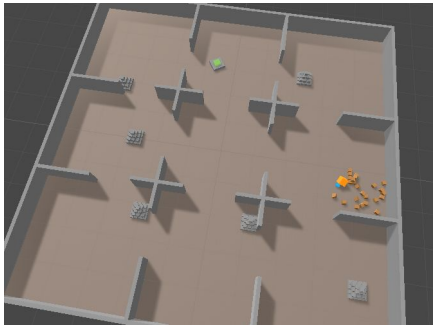


모듈A의 Next Encoding 과 모듈B의 Predicted Next Encoding 의 값의 차이가 호기심 리워드.

예측한 인코딩된 다음 상태 - 실제 인코딩된 다음 상태

Curiosity learning- pyramid



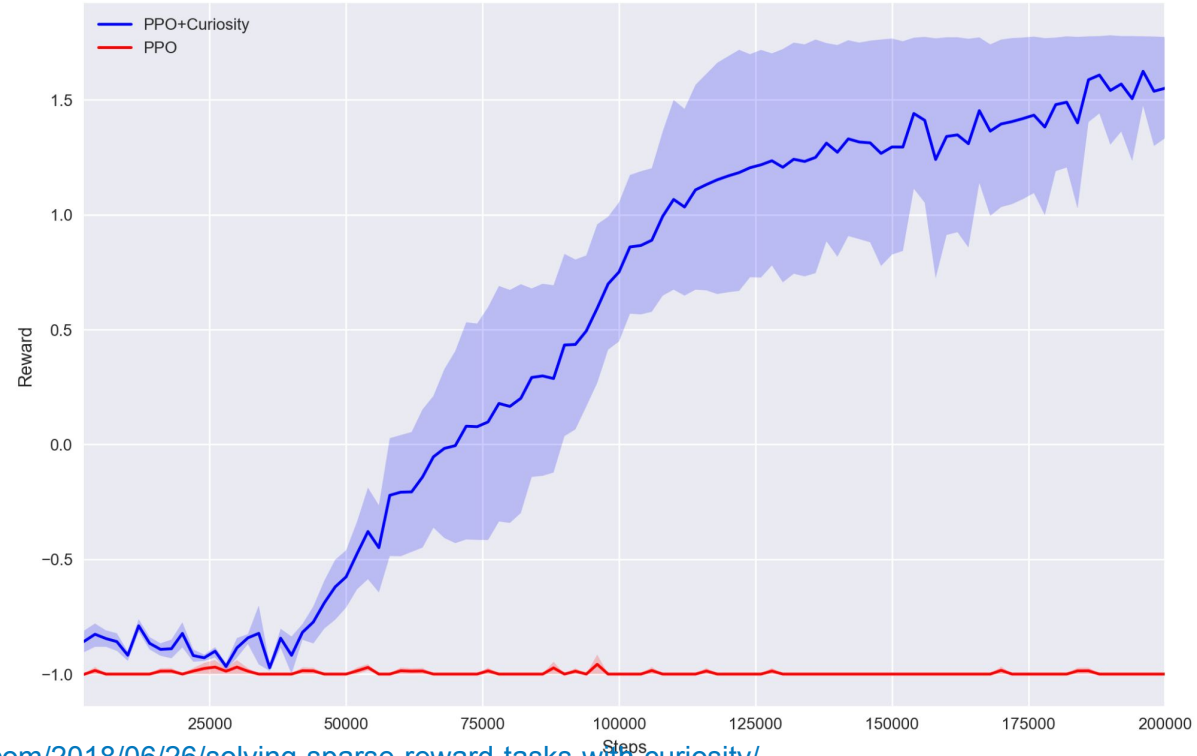


피라미드 환경은 복잡하고 단계적인 목표를 가지고 있고 일반적인 PPO로는 거의 학습이 되지 않는다.

- 9개의 방의 랜덤 위치에 스위치 생성
- 스위치을 건드림 -> 랜덤 위치에 피라미드가 생긴다.
- 랜덤 위치의 피라미드를 부신다
- 상층부의 노란색 블럭을 먹는다. (+2 리워드)

https://blogs.unity3d.com/2018/06/26/solving-sparse-reward-tasks-with-curiosity/

Curiosity learning- pyramid



https://blogs.unity3d.com/2018/06/26/solving-sparse-reward-tasks-with-curiosity/

Curiosity learning . trainer_config.yaml

```yaml
PyramidsLearning:
    use_curiosity: true
    summary_freq: 2000
    curiosity_strength: 0.01
    curiosity_enc_size: 256
    time_horizon: 128
    batch_size: 128
    buffer_size: 2048
    hidden_units: 512
    num_layers: 2
    beta: 1.0e-2
    max_steps: 5.0e5
    num_epoch: 3
```

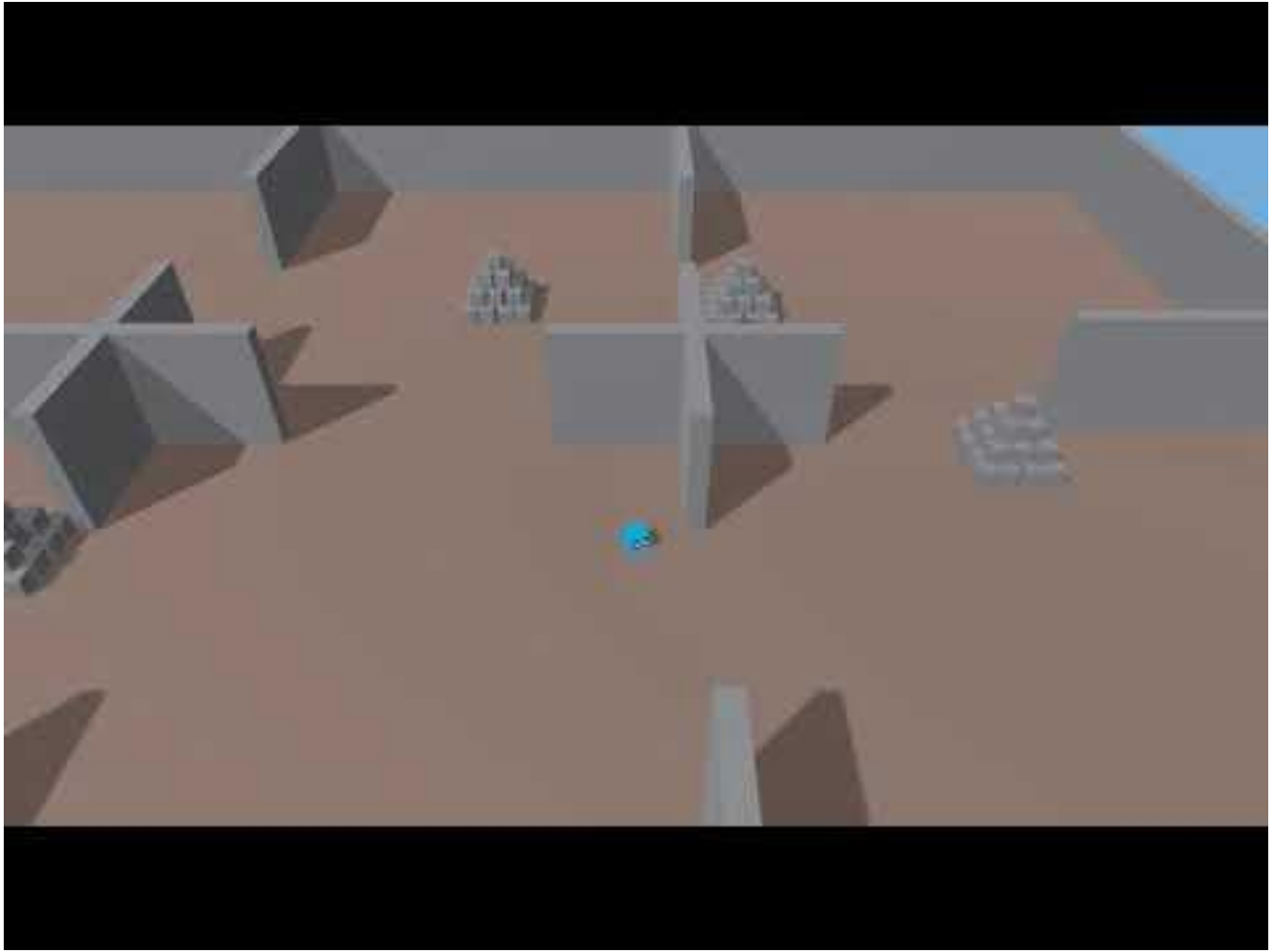
하이퍼 파라메터

use_curiosity: true // 호기심 리워드 사용 여부

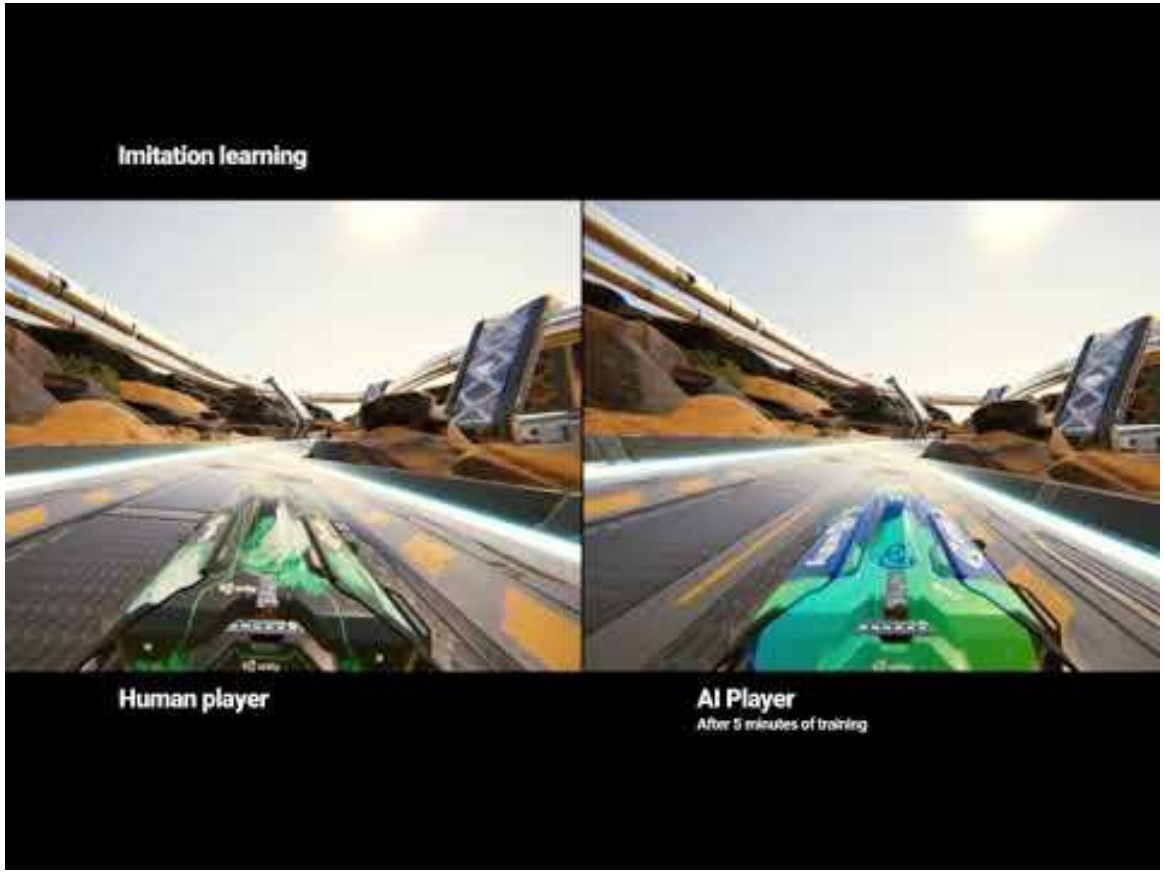
curiosity_strength: 0.01 // 호기심 리워드의 강도

curiosity_enc_size: 256 // 호기심 인코더의 벡터 사이즈

환경을 인코딩 한다고 생각했을때 환경에 따라 충분히 큰값이어야 한다.



행동 복사 (behavior cloning)



Behavior Cloning

- Self-driving cars as example

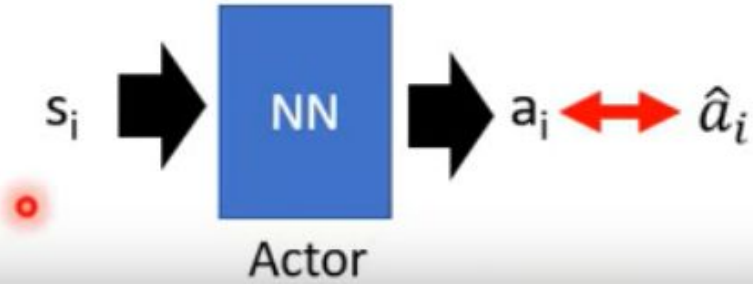
observation

Expert (Human driver): 向前

Machine: 向前

Training data:

$(s_1, \hat{a}_1)$
$(s_2, \hat{a}_2)$
$(s_3, \hat{a}_3)$
.....

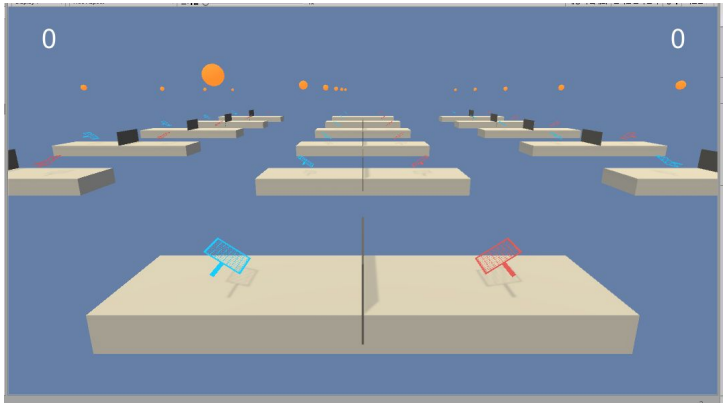


https://youtu.be/rl_ozvqQUU8

Tennis환경

```
public override void CollectObservations()
{
    AddVectorObs(invertMult * (transform.position.x - myArea.transform.position.x)); // X위치
    AddVectorObs(transform.position.y - myArea.transform.position.y); // Y위치
    AddVectorObs(invertMult * agentRb.velocity.x); // X속도
    AddVectorObs(agentRb.velocity.y); // Y속도

    AddVectorObs(invertMult * (ball.transform.position.x - myArea.transform.position.x)); //공 X위치
    AddVectorObs(ball.transform.position.y - myArea.transform.position.y); // 공 Y위치
    AddVectorObs(invertMult * ballRb.velocity.x); // 공 X속도
    AddVectorObs(ballRb.velocity.y); // 공 Y속도
}
```



<float,8> vector 의 옵져베이션

2개의 액션 (x, y 방향으로 가해지는 힘, continus)

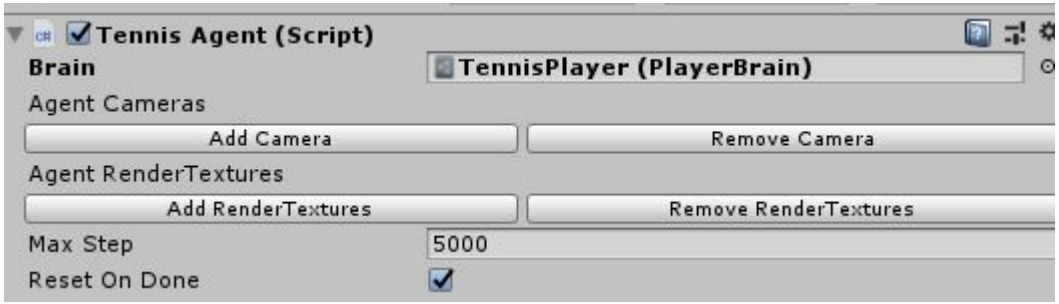
y 방향으로 중력이 작용 (횡이동 , 점프)

```
public override void AgentAction(float[] vectorAction, string textAction)
{
    var moveX = Mathf.Clamp(vectorAction[0], -1f, 1f) * invertMult; // X 축 속도
    var moveY = Mathf.Clamp(vectorAction[1], -1f, 1f); // Y축 속도
```

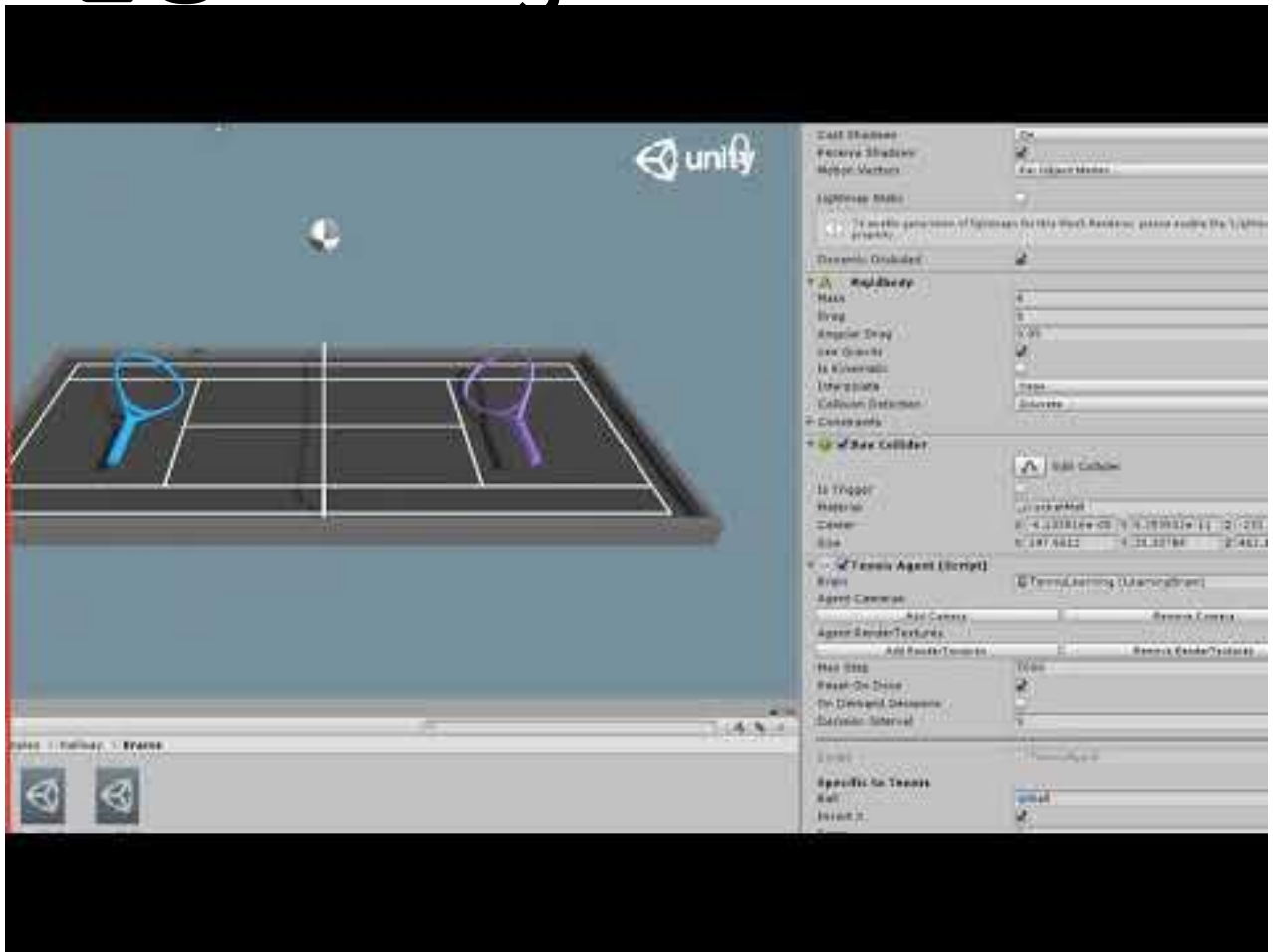
Tennis 환경


PC › Windows (C:) › Study › UnityMLAgent10 › ml-agents-master › UnitySDK › Assets › ML-Agents › Examples › Tennis › Scenes
이름
수정한 날짜
유형
크기
Tennis.unity
2019-10-01 오전...
Unity scene file
57KB
Tennis.unity.meta
2019-10-01 오전...
META 파일
1KB
TennisIL.unity
2019-10-01 오전...
Unity scene file
23KB
TennisIL.unity.meta
2019-10-01 오전...
META 파일
1KB


TennisIL*
Main Camera
Canvas
EventSystem
Academy
TennisArea
Ball
TeacherAgent
Invisible Walls
StudentAgent
Scenery
Directional_Light
Canvas_Watermark


Tennis Agent (Script)
Brain
TennisPlayer (PlayerBrain)
Agent Cameras
Add Camera
Remove Camera
Agent RenderTextures
Add RenderTextures
Remove RenderTextures
Max Step
5000
Reset On Done

Tennis IL 환경 - setting



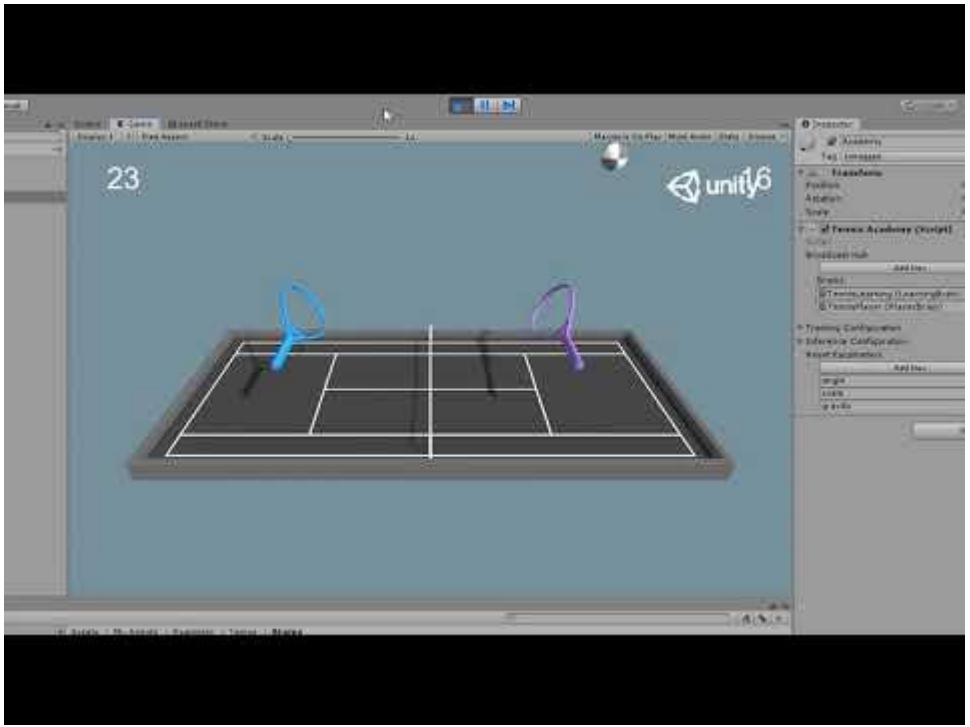
online 행동 복사

```
mlagents-learn config/online_bc_config.yaml --run-id=tennis_il --train --slow
```

online_bc_config.yaml :

```
 TennisLearning:
trainer: online_bc
    max_steps: 10000
    summary_freq: 1000
brain_to_imitate: TennisPlayer
    batch_size: 16
    batches_per_epoch: 5
    num_layers: 4
    hidden_units: 64
    use_recurrent: false
    sequence_length: 16
```

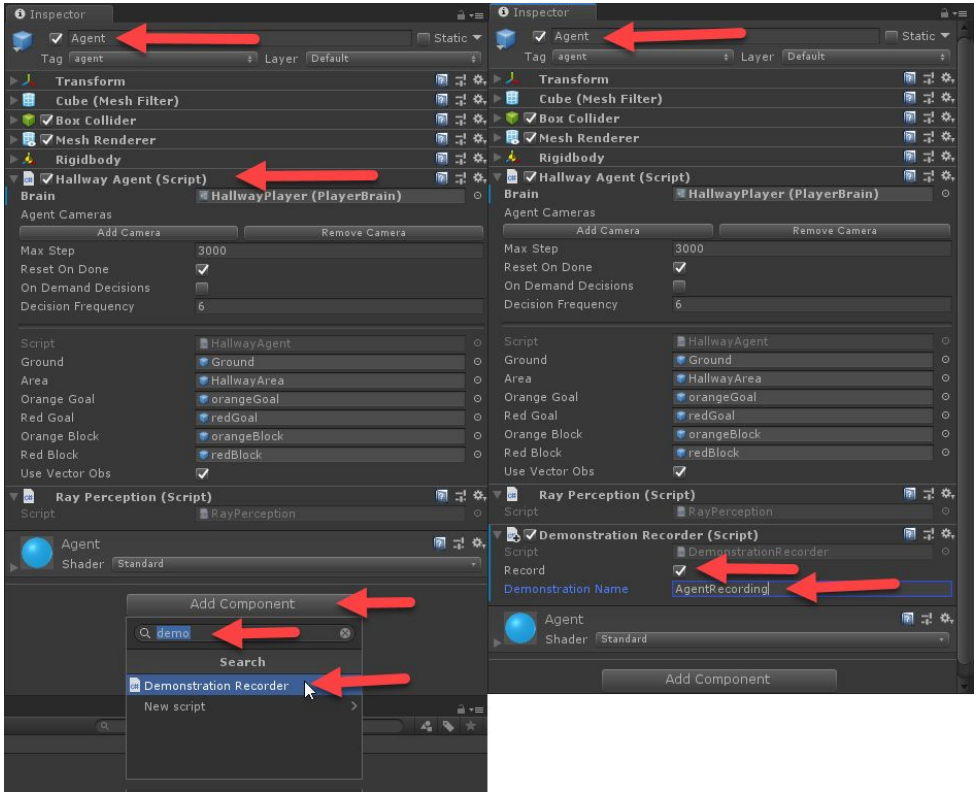
Tennis 환경 트레이닝 online BC

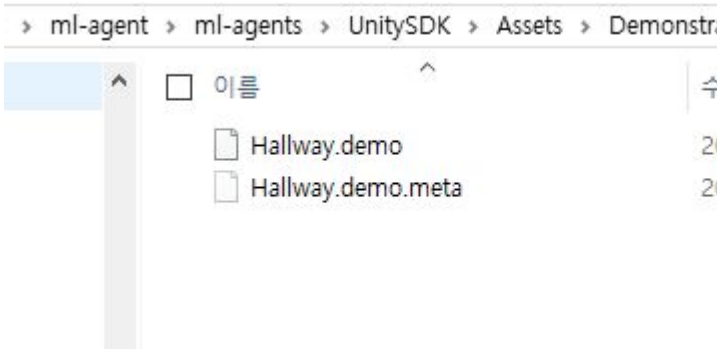


offline 행동 복사

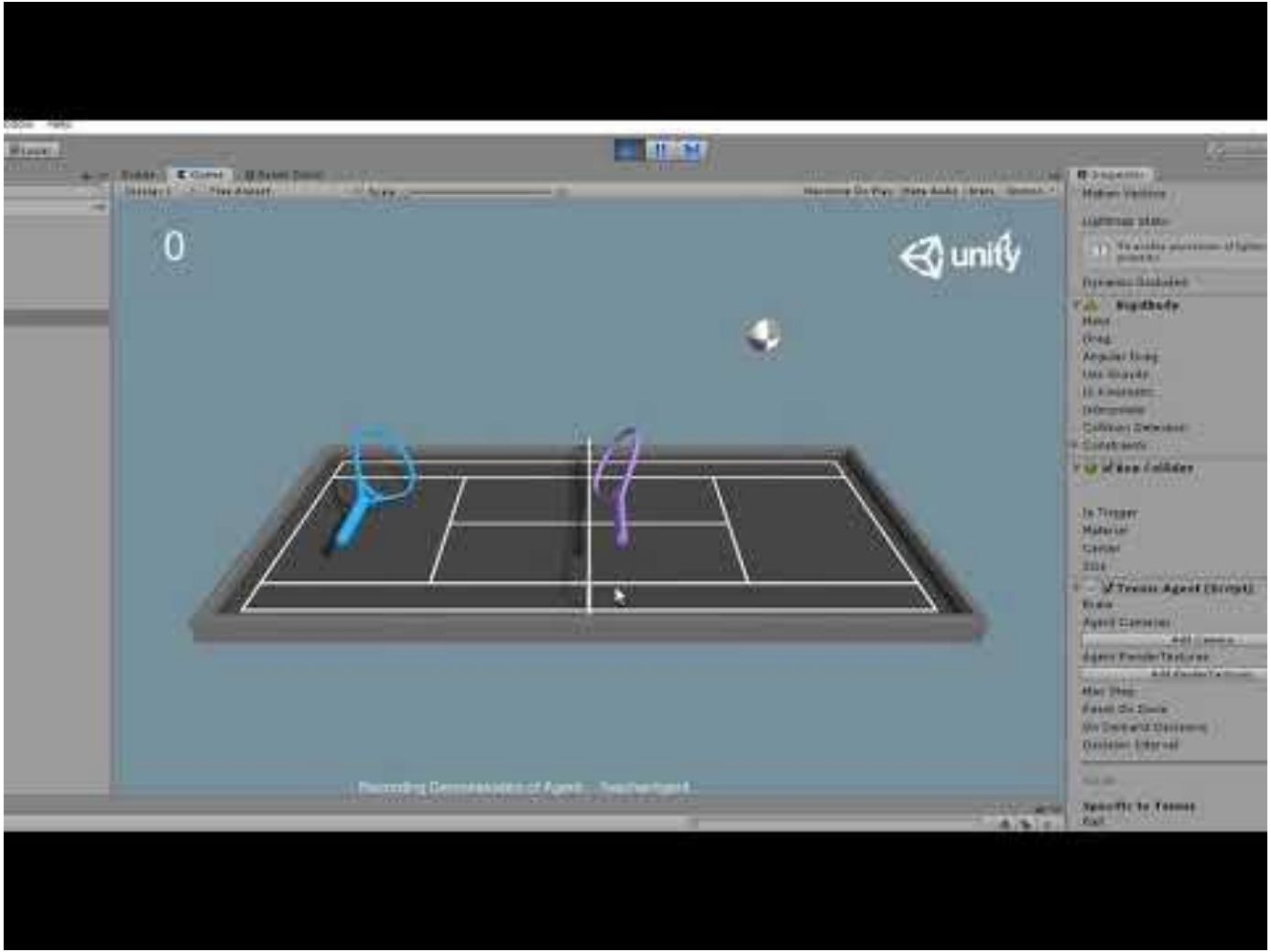
- 플레이어의 행동 데이터를 레코딩해서 행동복사에 이용하는 방법
- Unity ml-agent에서는 demo recorder 를 제공
- demo recorder 를 통해 얻어진 demo파일로 에이젼트를 학습시키는 형태
  a. demo 저장
  b. yaml 파일 수정
  c. 학습

offline 행동 복사 - demo 저장





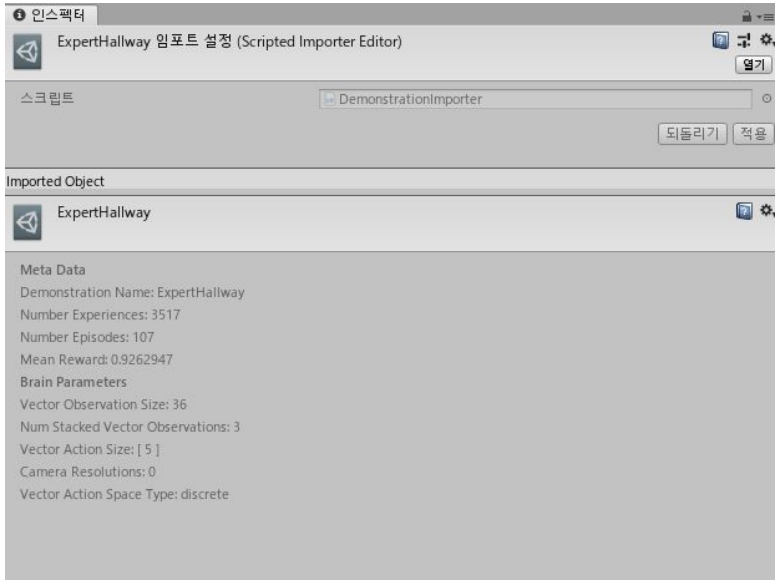
파일명을 Hallway로 지정해서 Hallway.demo파일이 생성


0
unity

offline 행동 복사 - 제공되는 demo파일

C › Windows (C:) › Study › UnityMLAgent10 › ml-agents-master › demos

이름	수정한 날짜	유형	크기
Expert3DBall.demo	2019-10-01 오전...	DEMO 파일	33KB
Expert3DBallHard.demo	2019-10-01 오전...	DEMO 파일	57KB
ExpertBasic.demo	2019-10-01 오전...	DEMO 파일	8KB
ExpertBouncer.demo	2019-10-01 오전...	DEMO 파일	10KB
ExpertCrawlerDyn.demo	2019-10-01 오전...	DEMO 파일	883KB
ExpertCrawlerSta.demo	2019-10-01 오전...	DEMO 파일	3,815KB
ExpertFood.demo	2019-10-01 오전...	DEMO 파일	552KB
ExpertGrid.demo	2019-10-01 오전...	DEMO 파일	786KB
ExpertHallway.demo	2019-10-01 오전...	DEMO 파일	3,336KB
ExpertPush.demo	2019-10-01 오전...	DEMO 파일	2,840KB
ExpertPyramid.demo	2019-10-01 오전...	DEMO 파일	1,386KB
ExpertReacher.demo	2019-10-01 오전...	DEMO 파일	584KB
ExpertTennis.demo	2019-10-01 오전...	DEMO 파일	142KB
ExpertWalker.demo	2019-10-01 오전...	DEMO 파일	4,212KB

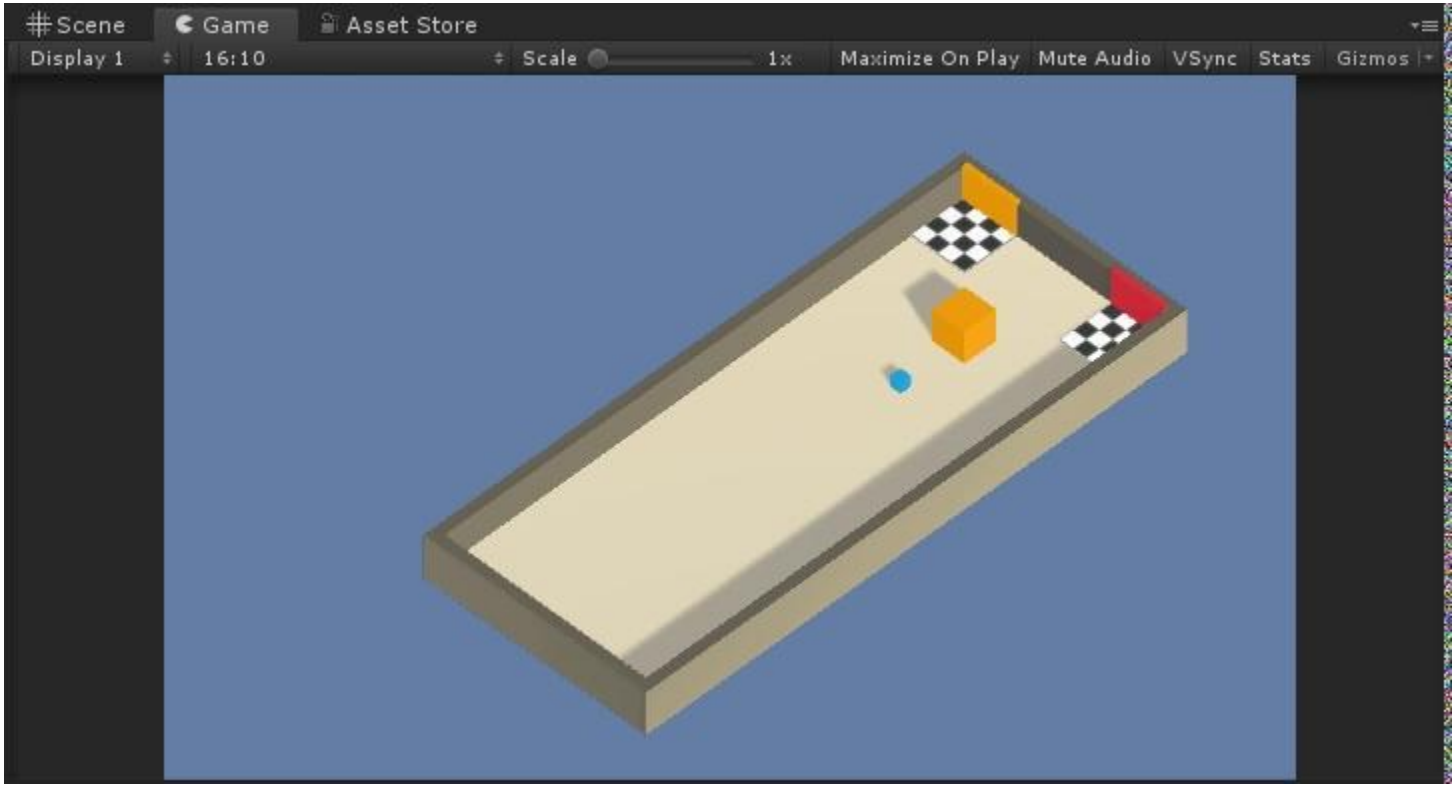


offline-bc yaml

HallwayLearning:
  trainer: offline_bc
  max_steps: 5.0e5
  num_epoch: 5
  batch_size: 64
  batches_per_epoch: 5
  num_layers: 2
  hidden_units: 128
  sequence_length: 16
  use_recurrent: true
  memory_size: 256
  sequence_length: 32
  demo_path: ./UnitySDK/Assets/Demonstrations/Hallway.demo

```
mlagents-learn config/offline_bc_config.yaml --run-id=hallway_il
--train
```

offline BC hallway



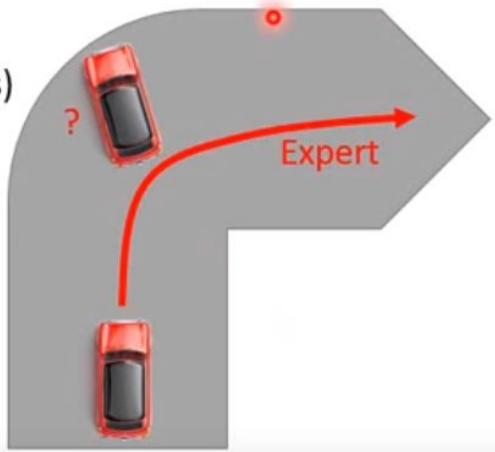
학습이 잘되지 않는 이유

1. sample 이 전체 액션스페이스를 대변하지 않음
2. Action space 가 랜덤한 환경에서는 행동복사는 환경에 대한 패턴인식이 부족할 수 있다.
3. 초반 에러가 중후반에 큰 에러가 될 수 있다.
4. 데모 데이터가 있다고 해도 모든 액션 스페이스를 커버할 수 있는 모델의 용량이 부족할 수 있다.

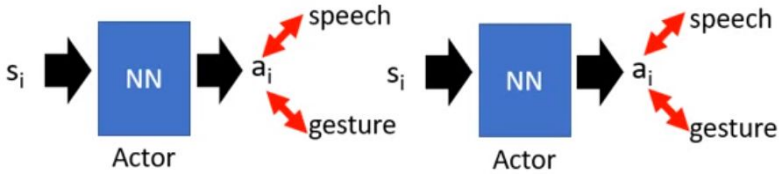
- Problem

Expert only samples limited observation (states)

Let the expert in the states seem by machine



- Major problem: if machine has limited capacity, it may choose the wrong behavior to copy.



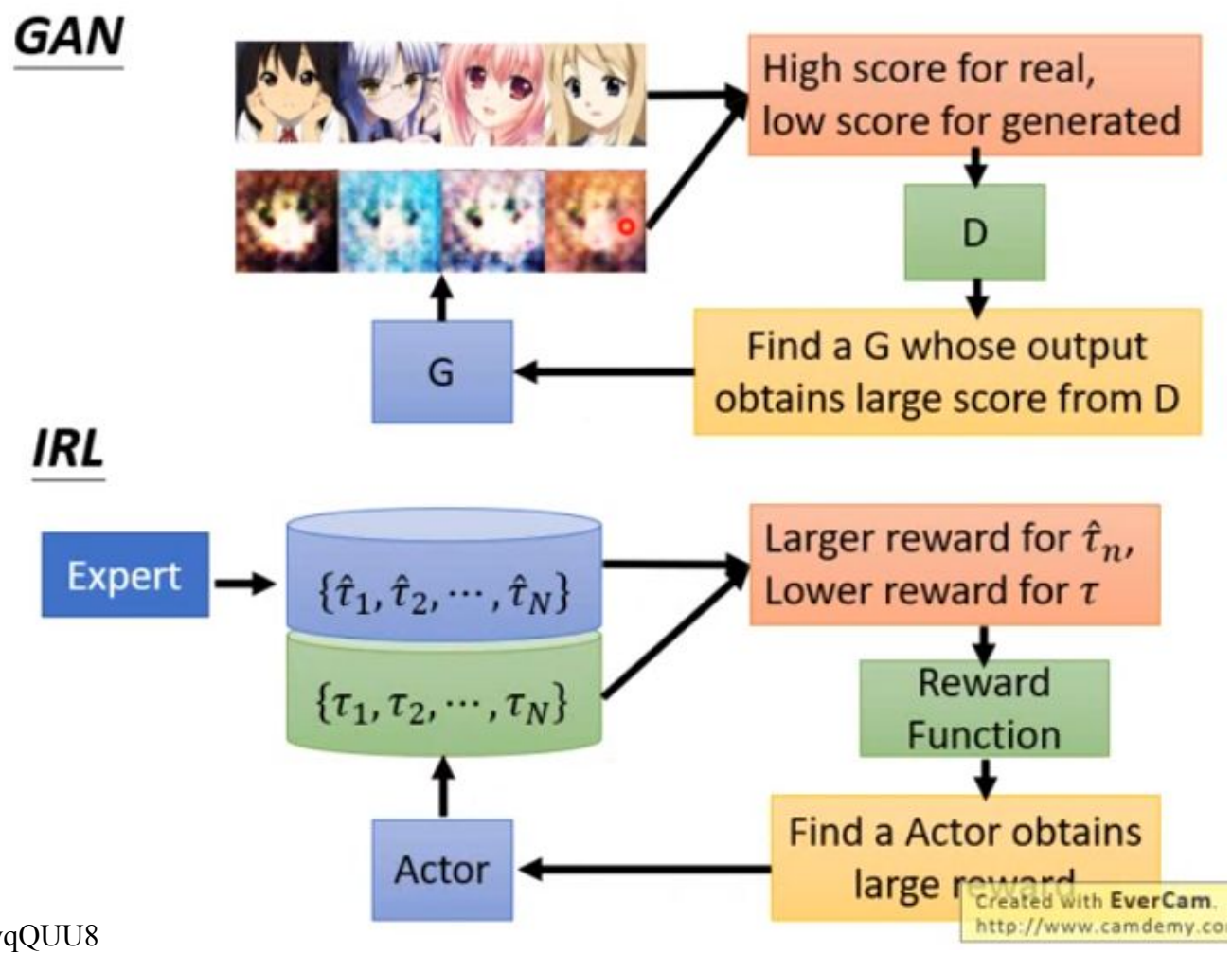
https://youtu.be/rl_ozvqQUU8

ML-Agents Beta 0.9.0

ervteng released this on 2 Aug · 55 commits to master since this release

New Features

- Added Generative Adversarial Imitation Learning (GAIL), a new way to do imitation learning. (#2118)
  - Unlike Behavioral Cloning, which requires demonstrations that exhaustively cover all the scenarios that an agent could encounter, GAIL enables imitation learning from as few as 5-10 demonstrations. This makes GAIL more applicable to more problems than Behavioral Cloning, and less work (in recording demonstrations) to set up.
  - GAIL can also be used with reinforcement learning to guide the behavior of the agent to be similar to the demonstrations. In environments where the reward is sparse, providing demonstrations can speed up training by several times. See imitation learning for more information on how to use GAIL, and a comparison of training times on one of our example environments.
- Enabled pre-training for the PPO trainer. (#2118)
  - Pre-training can be used to bootstrap an agent's behavior using human-provided demonstrations, and helps the agent explore in the right direction during training. It can be used in conjunction with GAIL for further training speedup, especially in environments where the agent rarely sees a reward, or gets "stuck" in certain parts. See imitation learning for more information on how to use pre-training.



https://youtu.be/rl_ozvqQUU8

Algorithm

- Input: expert trajectories $\{\hat{\tau}_1, \hat{\tau}_2, \cdots, \hat{\tau}_N\}$
- Initialize discriminator D and actor π
- In each iteration:
  - Using actor to obtain trajectories $\{\tau_1, \tau_2, \cdots, \tau_N\}$
  - Update discriminator parameters: Increase $D(\hat{\tau}_i)$, decrease $D(\tau_i)$

$$D(\tau) = \frac{1}{T}\sum_{t=1}^{T} \underbrace{d(s_t, a_t)}_{\text{reward}}$$

Find the reward function that expert has larger reward.

  - Update actor parameters: Increase $D(\tau_i)$

$$\theta^{\pi} \leftarrow \theta^{\pi} + \eta \sum_{i=1}^{N} D(\tau_i) \nabla_{\theta^{\pi}} log P(\tau_i|\pi)$$

Find the actor maximizing reward by reinforcement learning

https://youtu.be/rl_ozvqQUU8

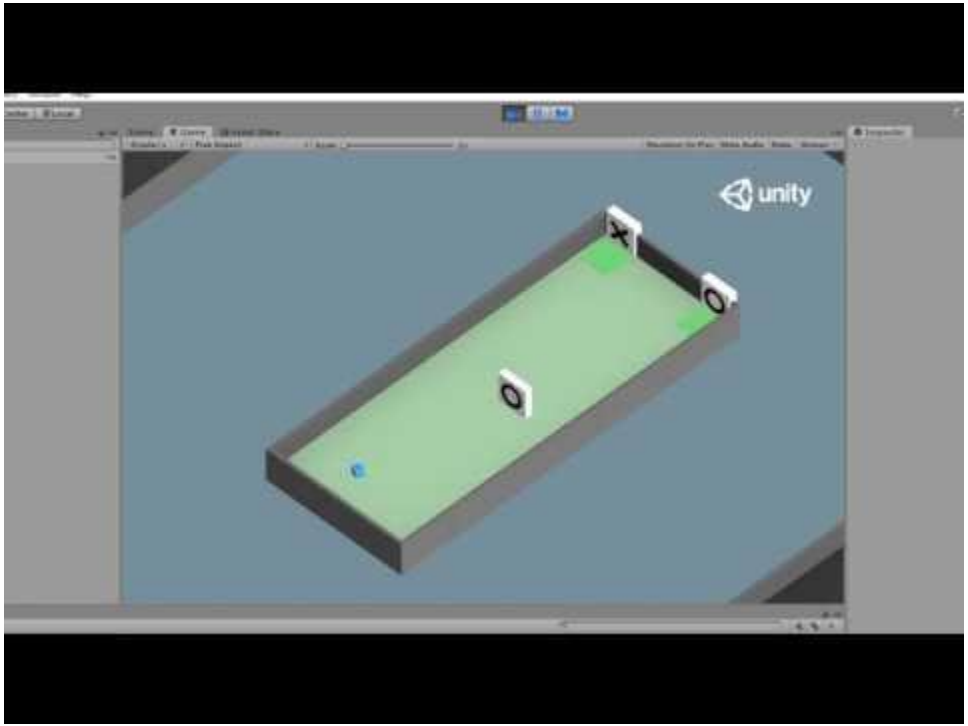
Gail Training

ppo와 동일하지만 reward signal에 gail 항목을 추가.

gail_config.yaml
HallwayLearning:
……..
  reward_signals:
    extrinsic:
      strength: 1.0
      gamma: 0.99
    gail:
      strength: 0.1
      gamma: 0.99
      encoding_size: 128
       demo_path: demos/ExpertHallway.demo

mlagents-learn .\config\gail_config.yaml --run-id=hallway_gail --train

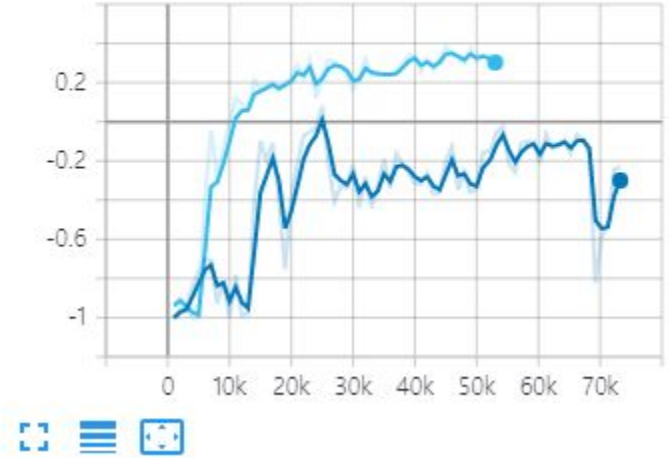
Hallway Gail training



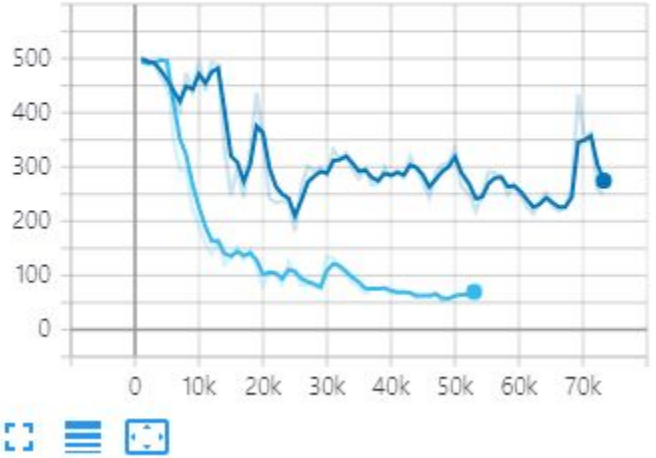
비교

Environment

Cumulative Reward
tag: Environment/Cumulative Reward



Episode Length
tag: Environment/Episode Length



결론

- Unity ML Agent로 Unity로 만든 환경에 쉽게 강화학습을 할 수 있다.
- ppo, 커리큘럼 러닝 , 호기심 기반 학습, BC , GAIL 등을 바로 적용 해 볼 수 있었다.
- 강화학습에서 학습 환경은 매우 중요하며 물리엔진등을 가지고 있는 유니티는 좋은 강화학습 도구이다.