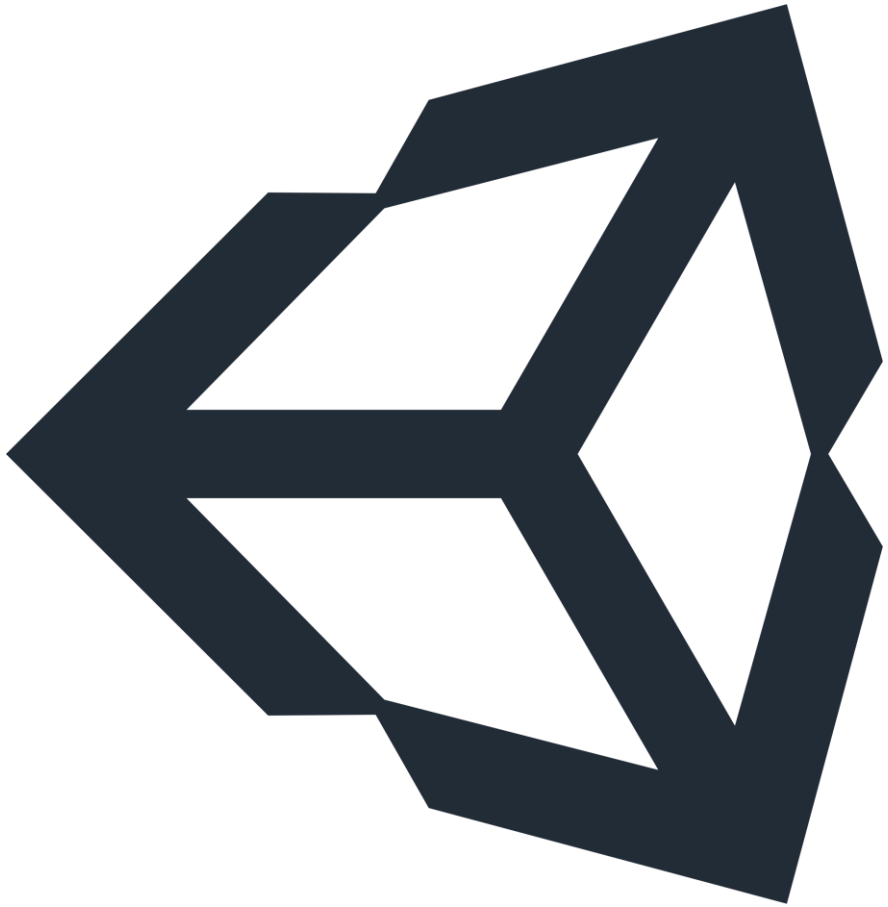
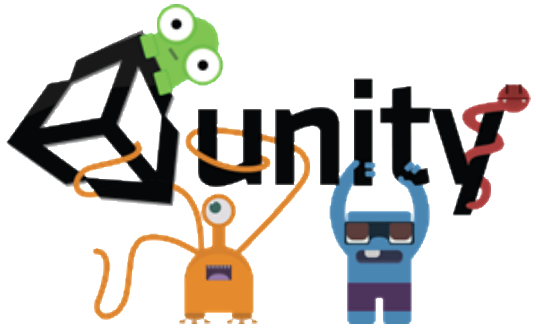
Unity ML-agents

2nd DLCAT 2019.07.04

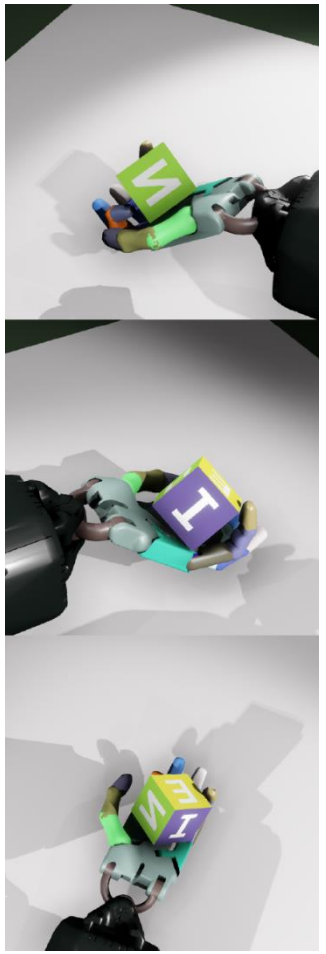


Contents

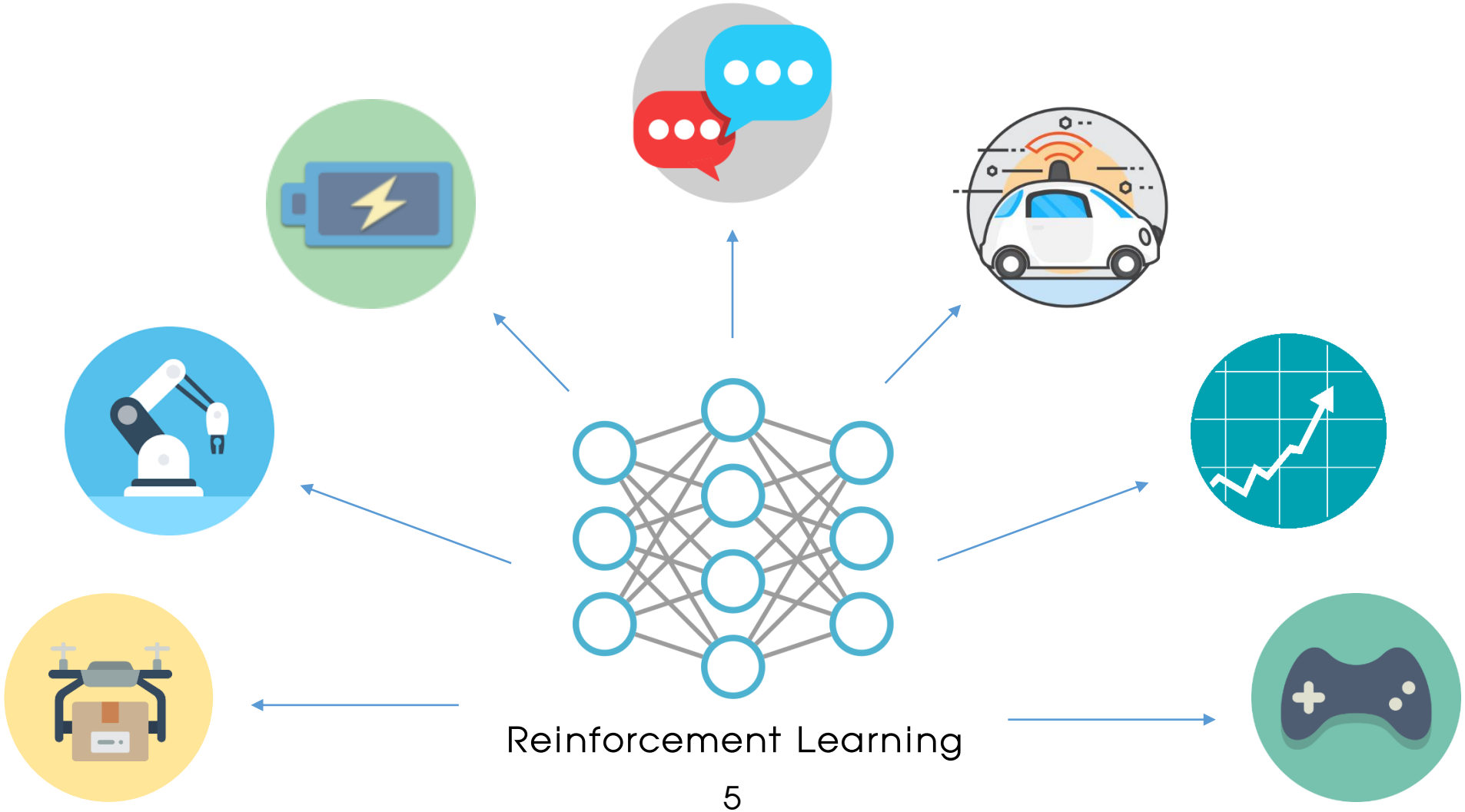


Machine Learning Agents

Unity ML-agents



Unity ML-agents



Reinforcement Learning

Unity ML-agents

Agent

Action (a)

Jump, forward, backward, run, ⋯

State (s)

Position of agent, enemy, coins

Reward (r)

Good

Bad

Environment

Unity ML-agents

Deep Q Network

Rainbow DQN

Deep Deterministic Policy Gradient

Trust Region Policy Optimization

Proximal Policy Optimization

Go-Explore

OpenAI GYM

Atari

GTA 5

Super Mario

Mujoco

Malmo

Unity ML-agents

환경에 대한 수정이 어려움!

필요한 환경이 없을 수도 있음!

Unity ML-agents

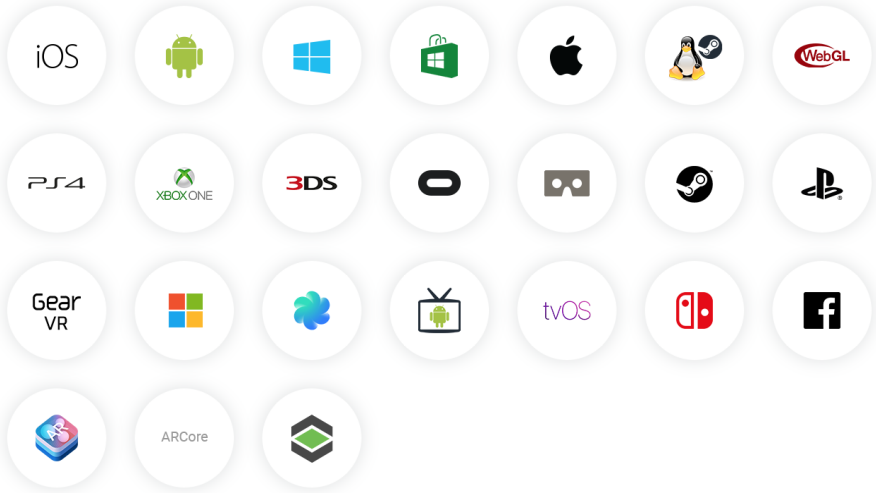
강화학습을 하는 사람들의 고민

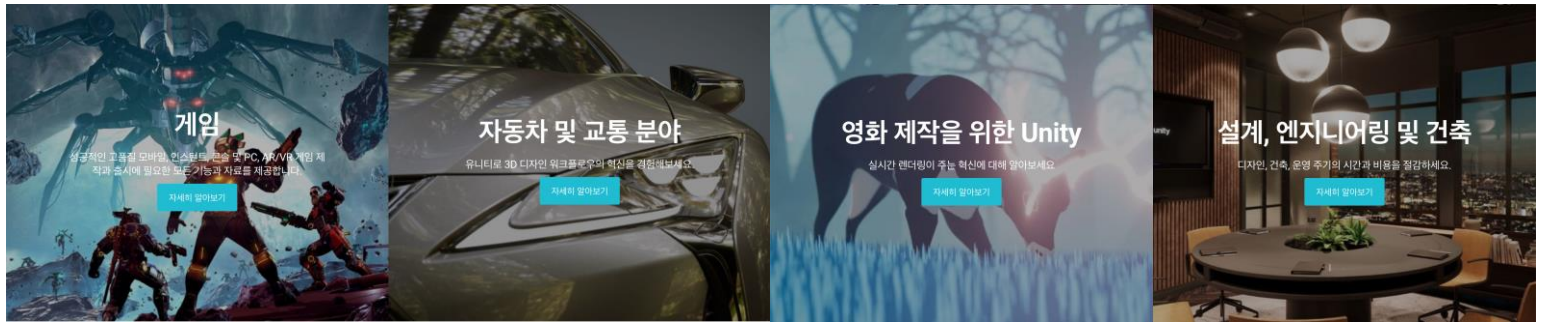
강화학습을 테스트할 환경 제작

Unity ML-agents

Unity

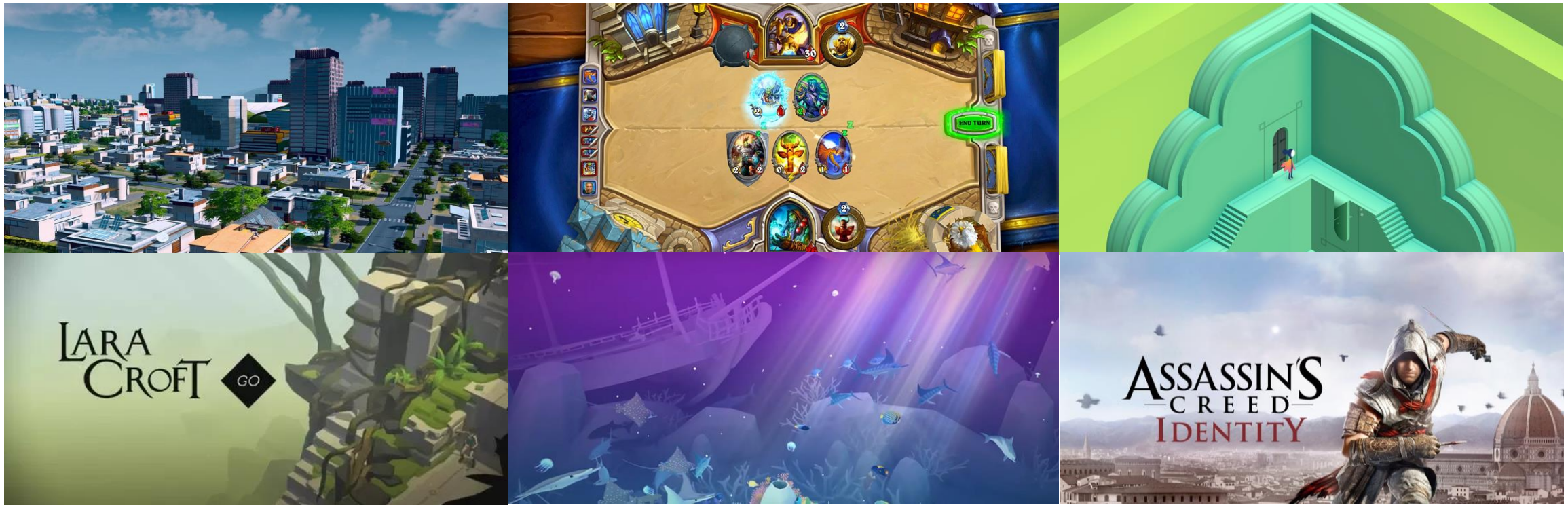
- 3D 및 2D 비디오 게임의 개발 환경을 제공하는 게임 엔진
- 3D 애니메이션과 건축 시각화, 가상현실(VR) 등 인터랙티브 콘텐츠 제작
- 게임 엔진 시장의 45% 이상 차지, 등록 개발자 수 500만 명 이상



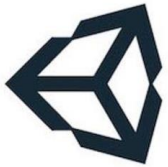


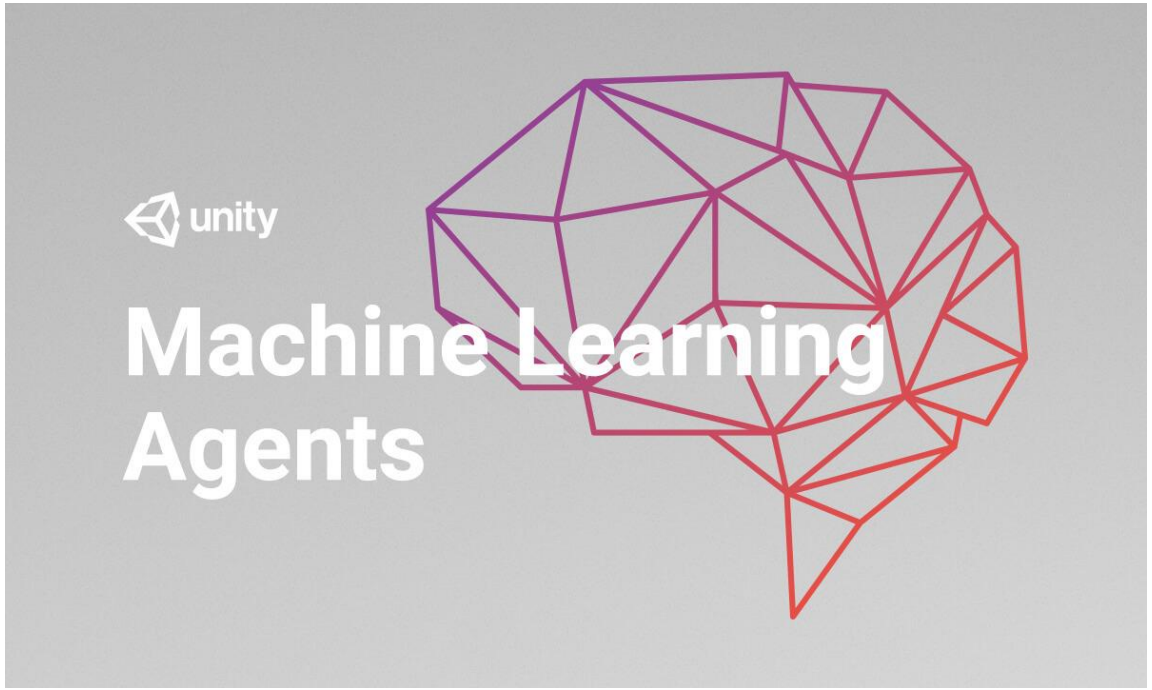
Unity ML-agents

Unity



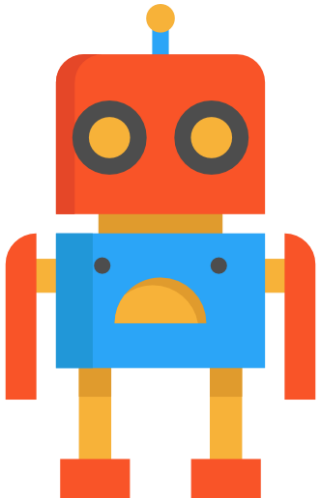
Unity ML-agents

Unity 



2017.09.19 출시 -> 현재 버전 0.8

Unity ML-agents



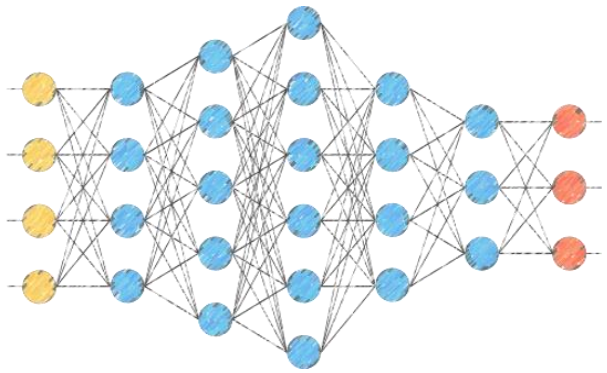
Agent

Action

State, Reward

Environment

Unity ML-agents



Agent

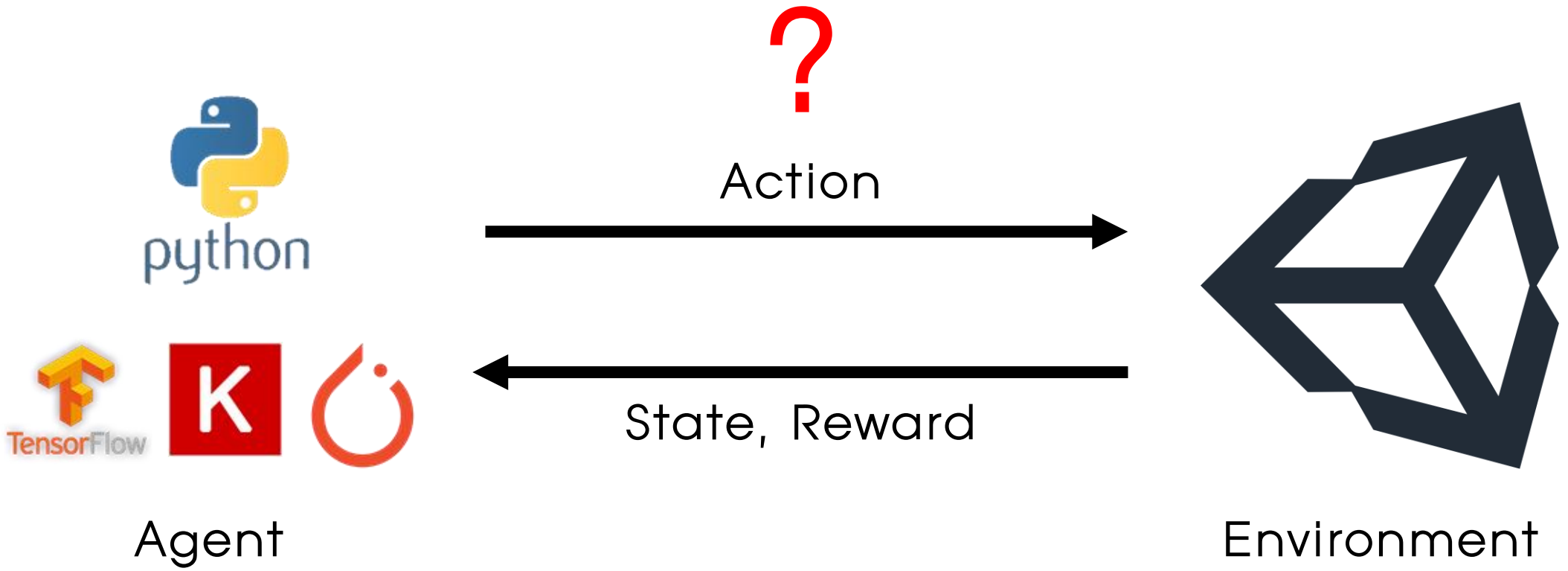
?

Action

State, Reward

Environment

Unity ML-agents



Unity ML-agents

Action

State, Reward

Agent

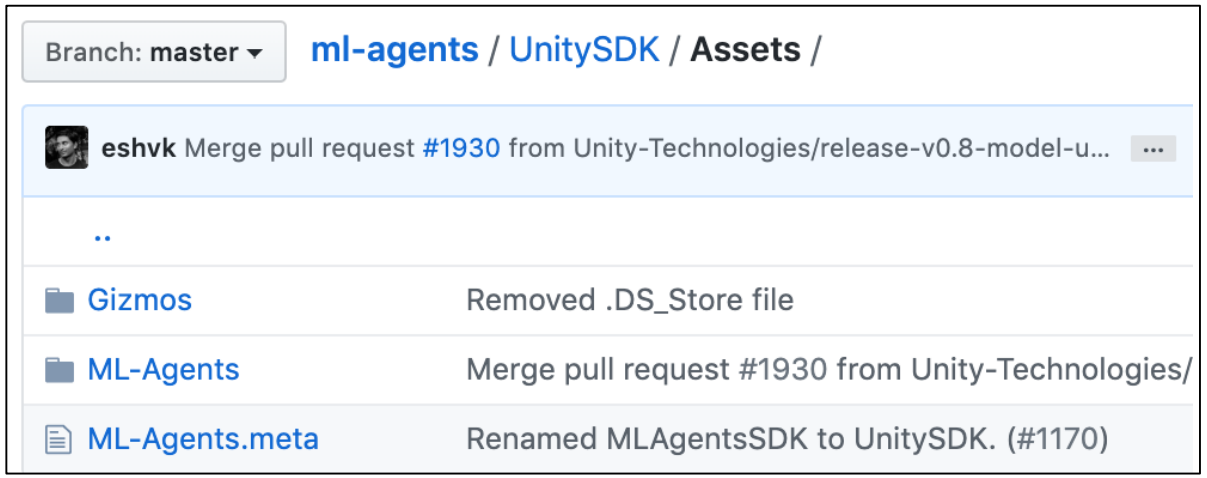
Environment

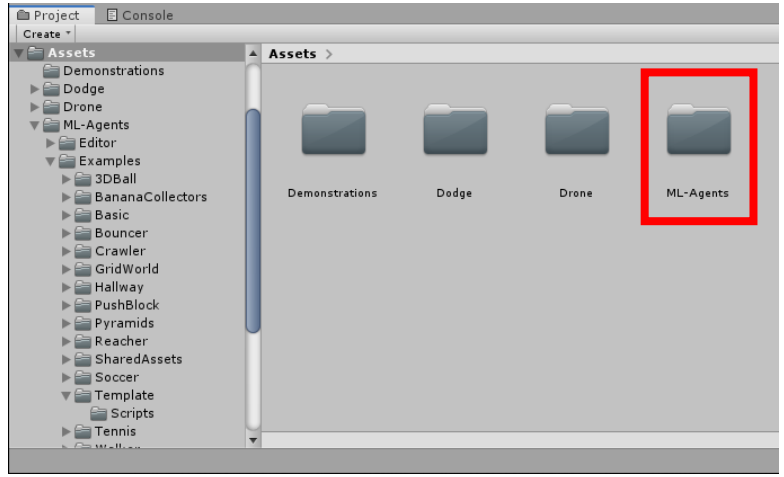
Unity ML-agents

- 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

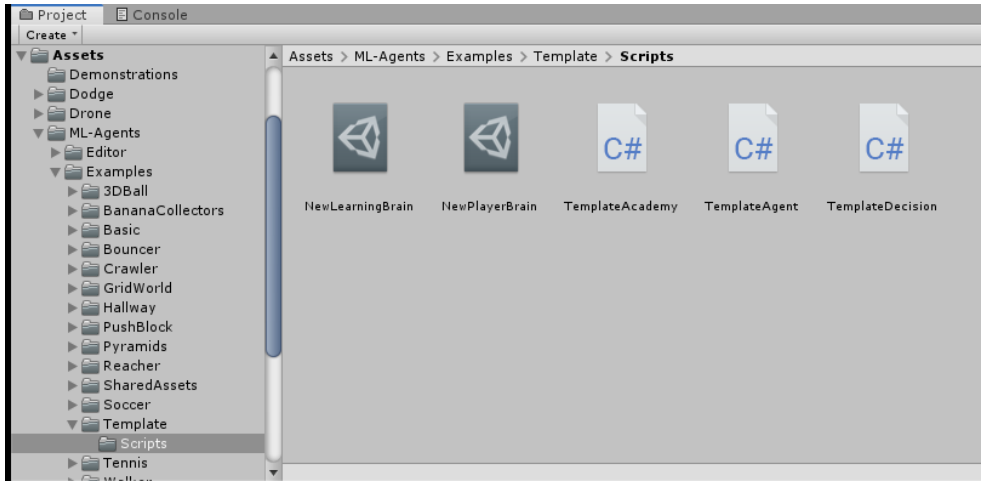
- ML-agents 깃허브에서 ML-agents 프로젝트 받기
  - https://github.com/Unity-Technologies/ml-agents
- ML-agents 프로젝트의 [UnitySDK 폴더 -> Assets 폴더] 내부의 폴더를 유니티의 Assets 내부에 복사

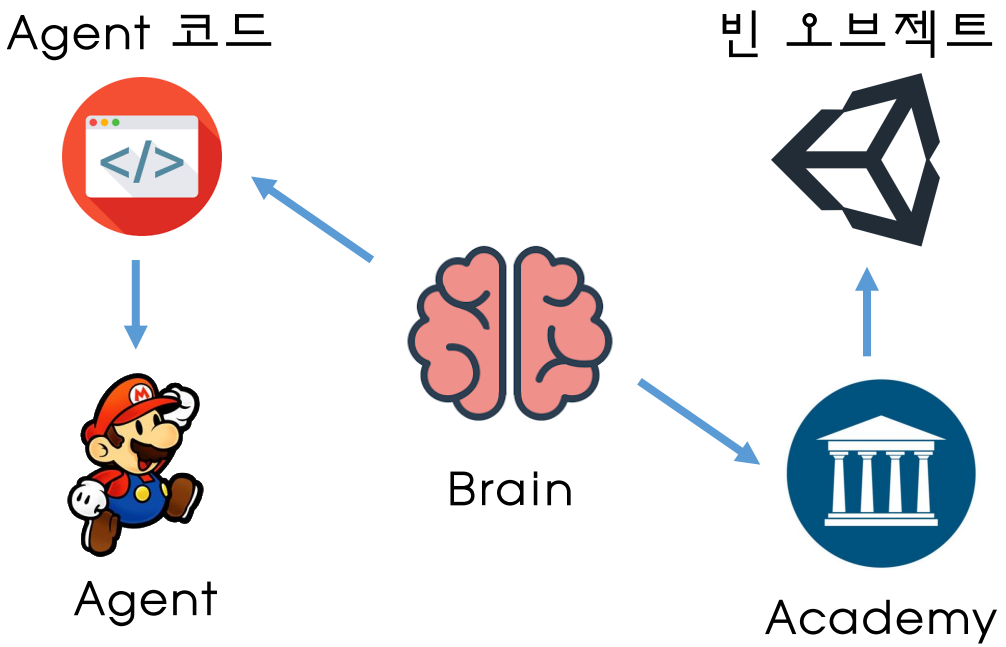




Unity ML-agents

- 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

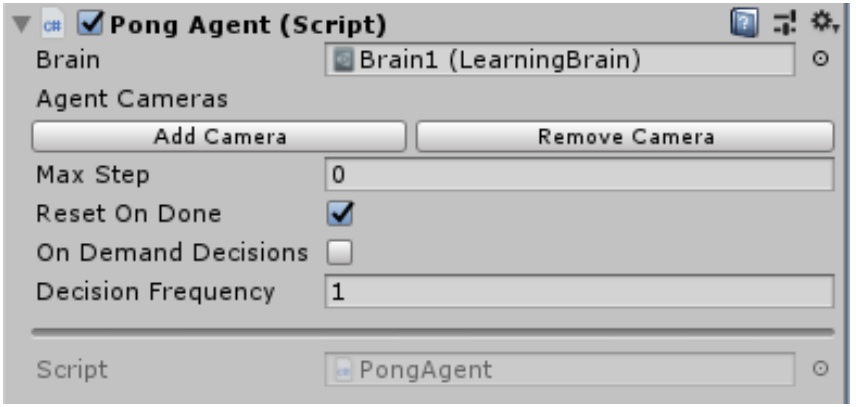




Agent

- 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 위하도록 설정
  - Decision Frequency: 해당 스텝마다 한번씩 액션을 취하도록 설정

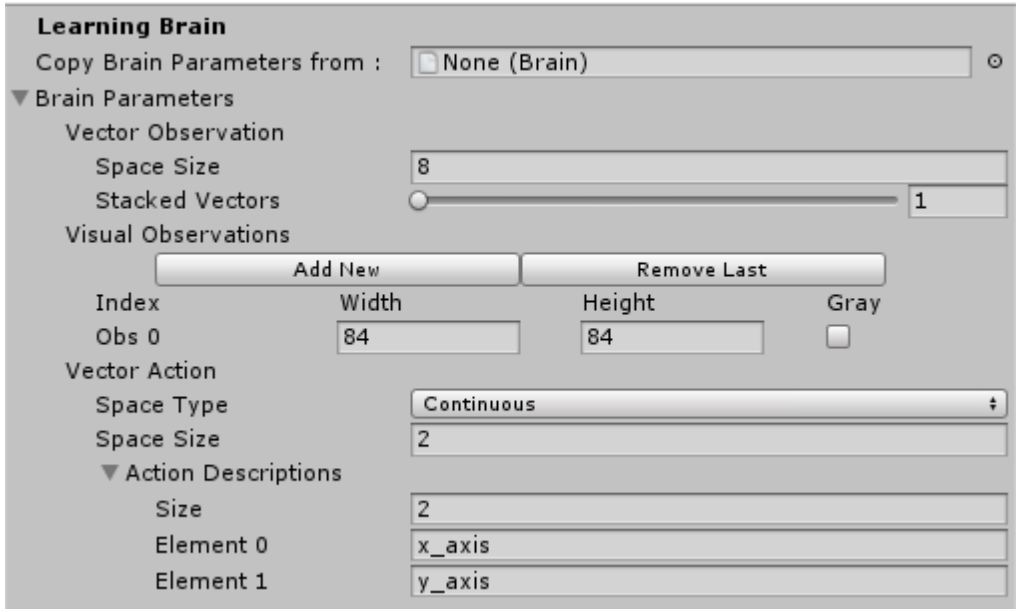


Brain 

- 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

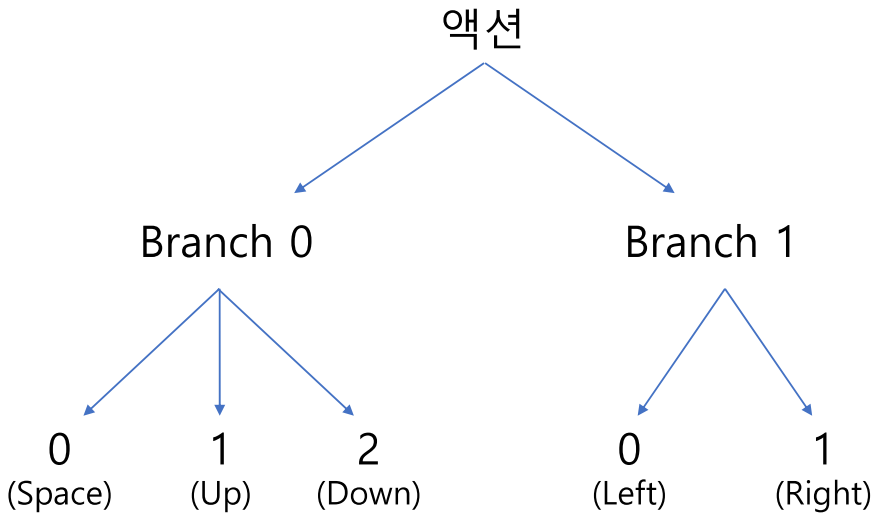
Brain

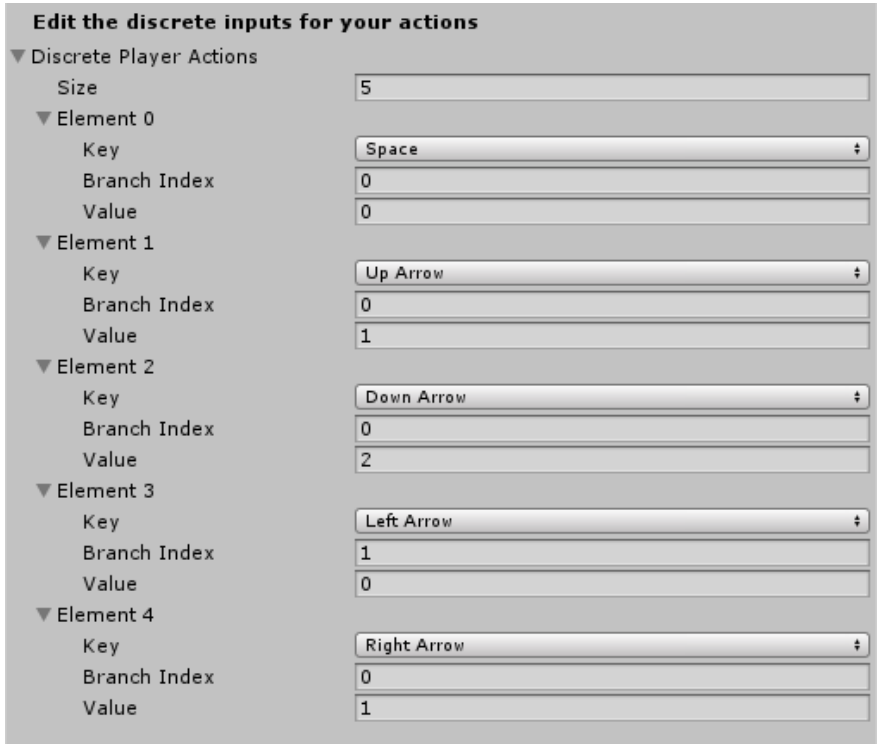
- 브레인 설정 (모든 브레인에 공통적으로 적용)
- Inspector View
  - Copy Brain Parameters from: 브레인의 파라미터 복사
  - Vector Observation
    - Space Size: Vector Observation의 크기 결정
    - Stacked Vectors: 설정한 스텝만큼 vector 정보를 stack
  - Visual Observation
    - Visual observation의 크기와 grayscale 여부 결정
  - Vector Action
    - 액션의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

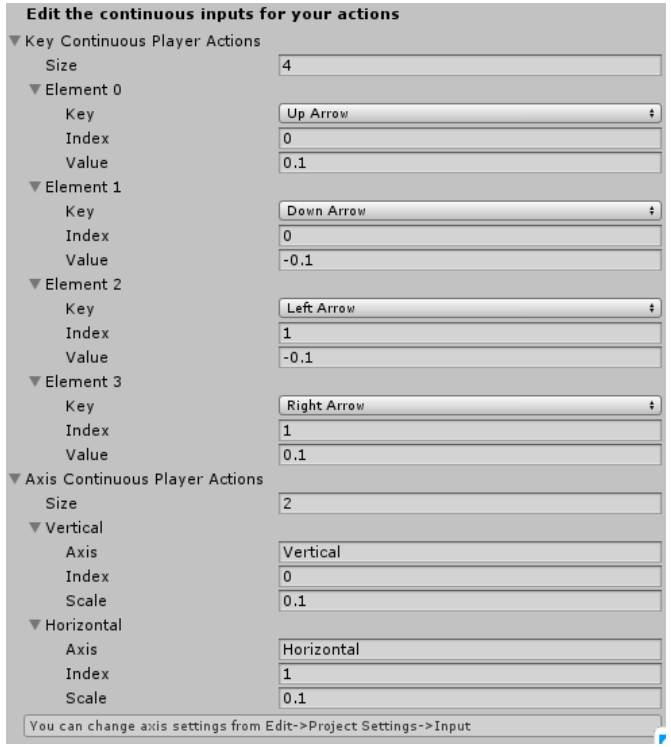


Brain

- Player 브레인 설정

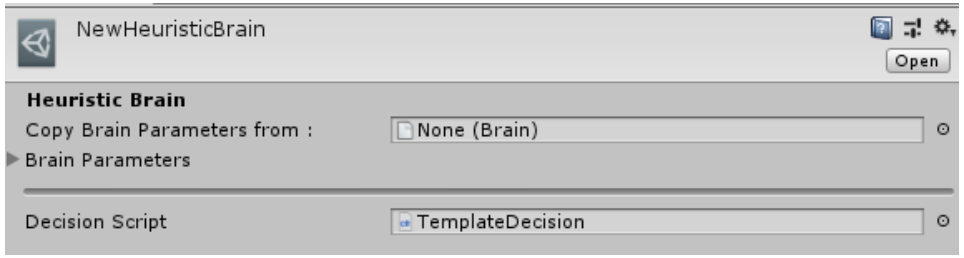




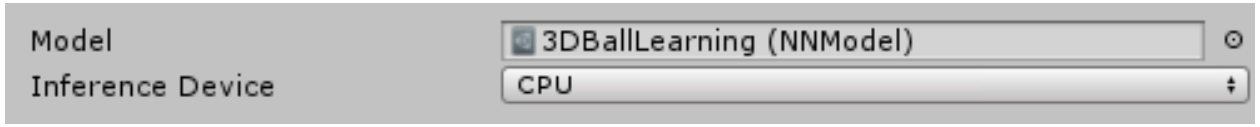


Brain

- Heuristic 브레인 설정
  - Decision Script에 코딩된 Decision 스크립트 파일 연결

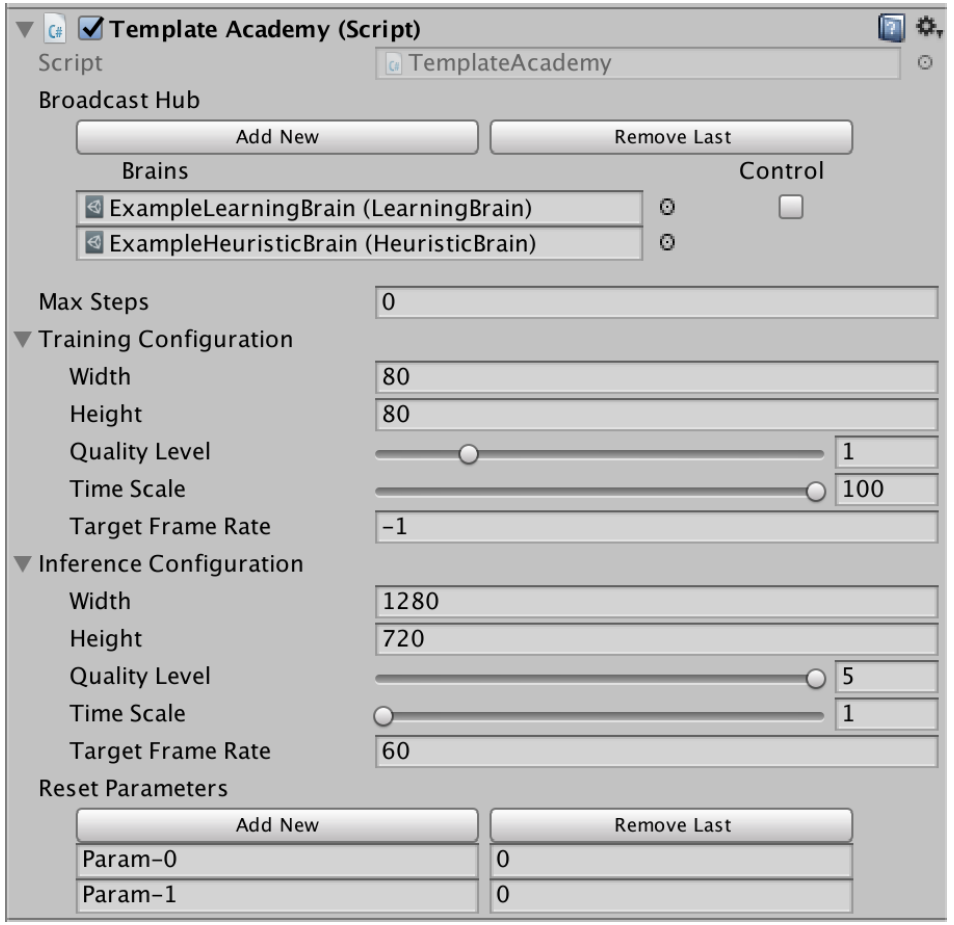


- Training 브레인 설정
  - 유니티 내부에서 사용할 nn 모델 연결 및 연산 장치 결정

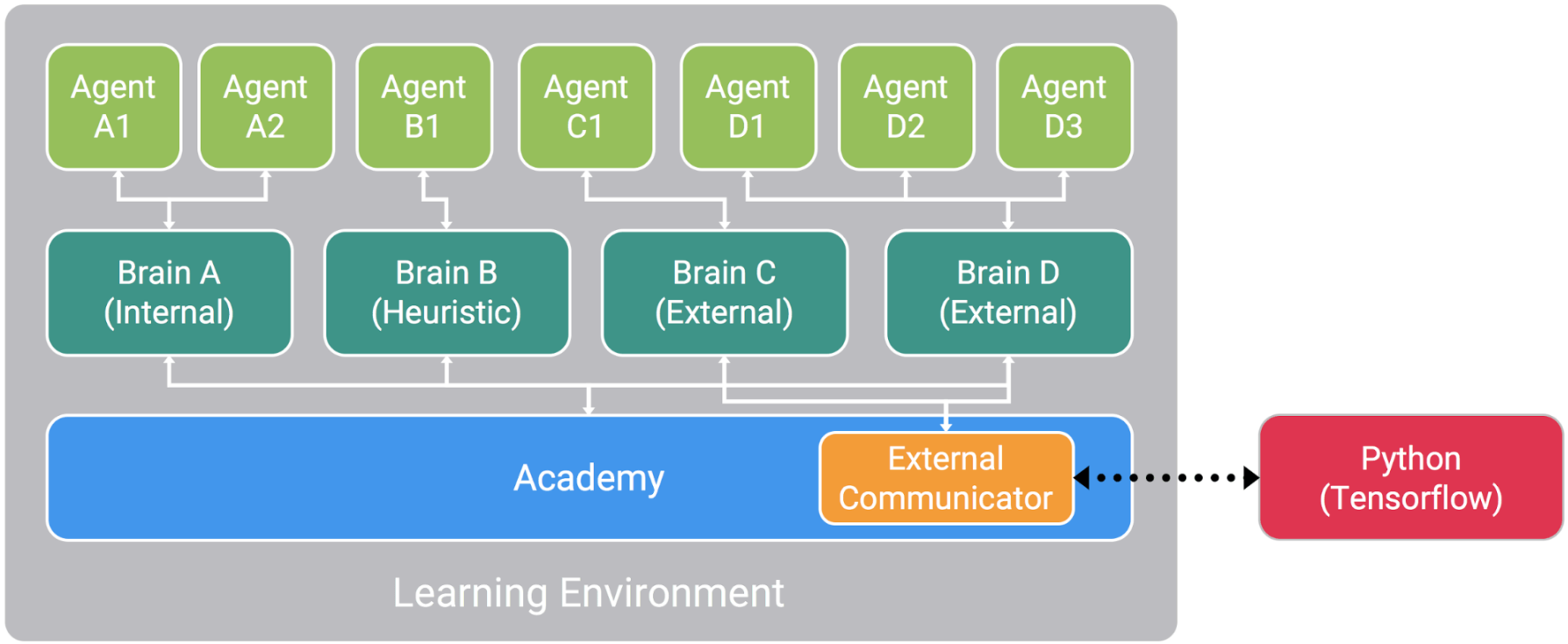


Academy

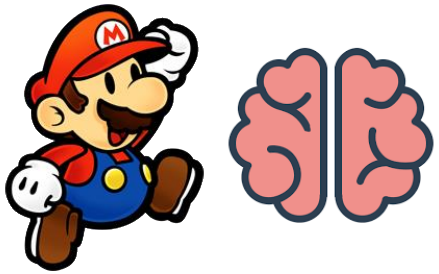
- 스크립트 내부는 3개의 함수로 구성
  - InitializeAcademy 함수: 환경이 처음 실행될 때 딱 한번 호출되는 함수
  - AcademyReset 함수: 매 에피소드가 끝날 때마다 한번씩 호출되는 함수
  - AcademyStep 함수: 매 스텝마다 호출되는 함수
- Inspector View
  - Broadcast Hub: 해당 환경에서 사용할 브레인을 추가
    - Learning brain의 control을 체크하면 external, 체크하지 않으면 internal
  - Max Steps: Global done을 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

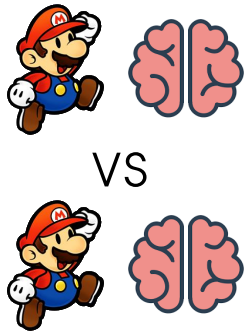


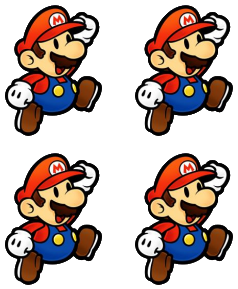
Unity ML-agents

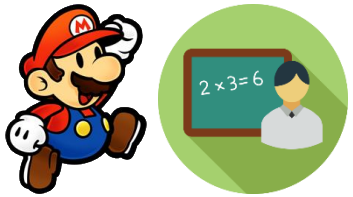


Unity ML-agents


: Single Agent

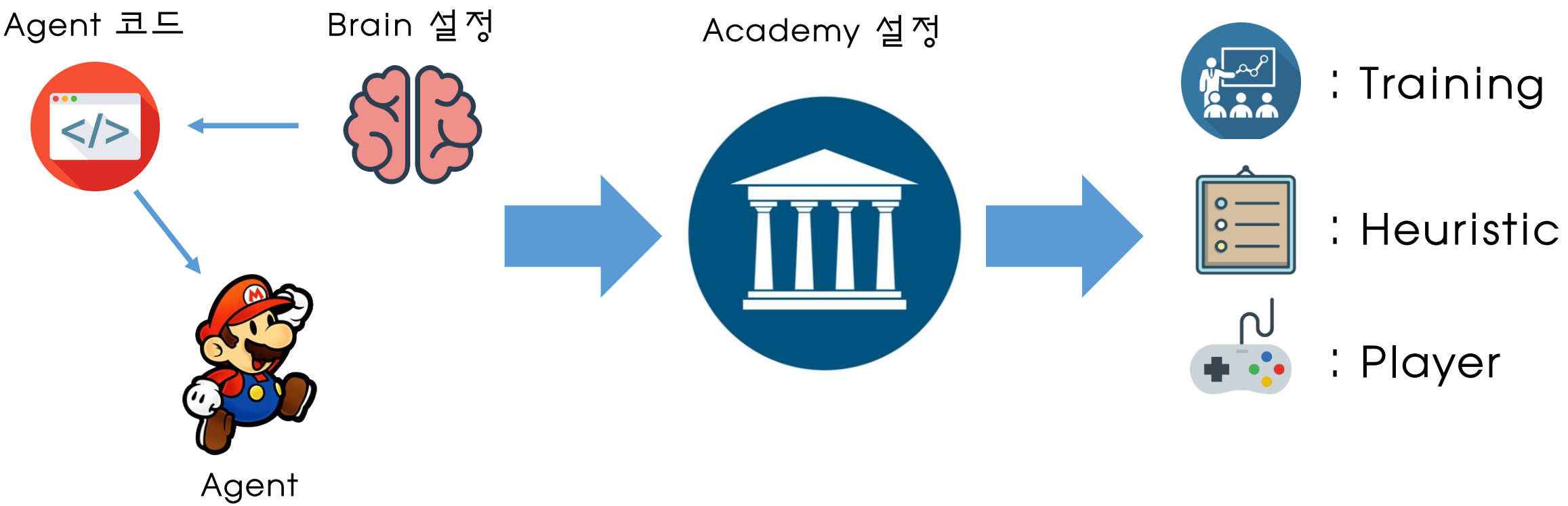

VS
: Adversarial Agents


: Multi-agents

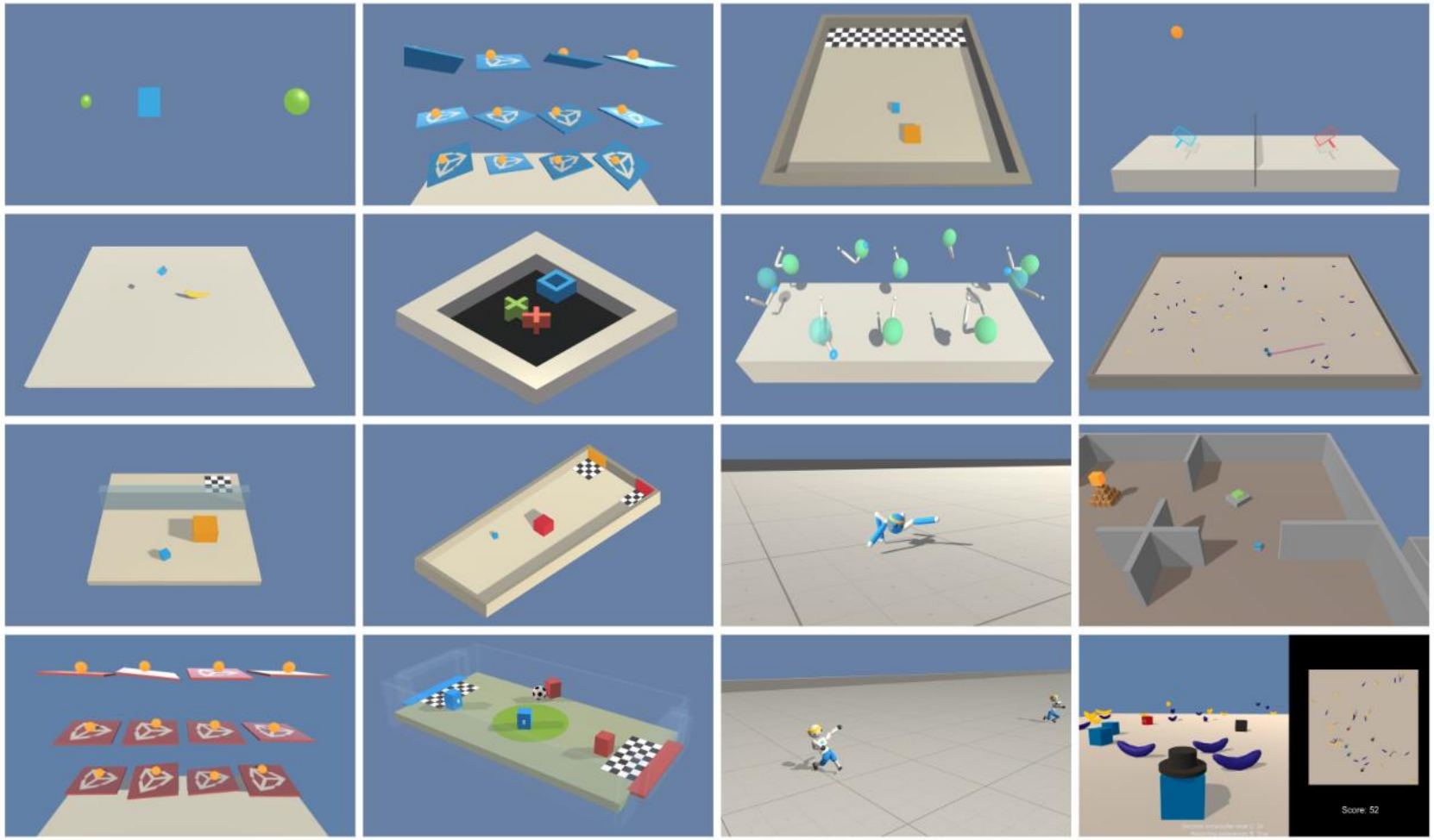


: Imitation Learning

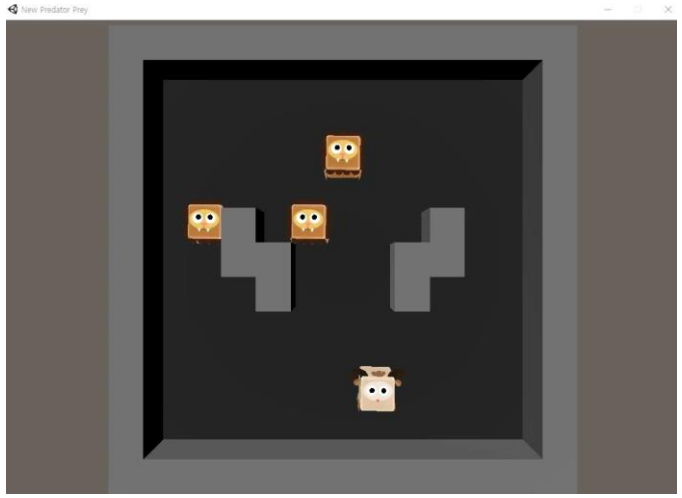
Unity ML-agents

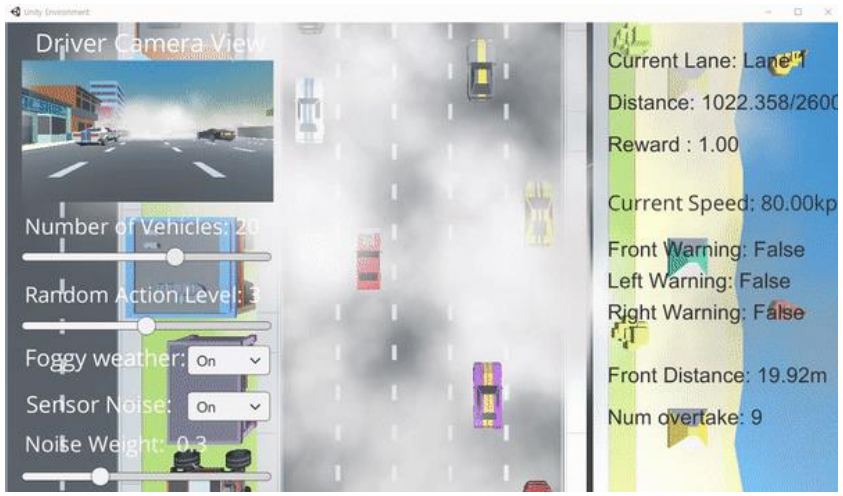


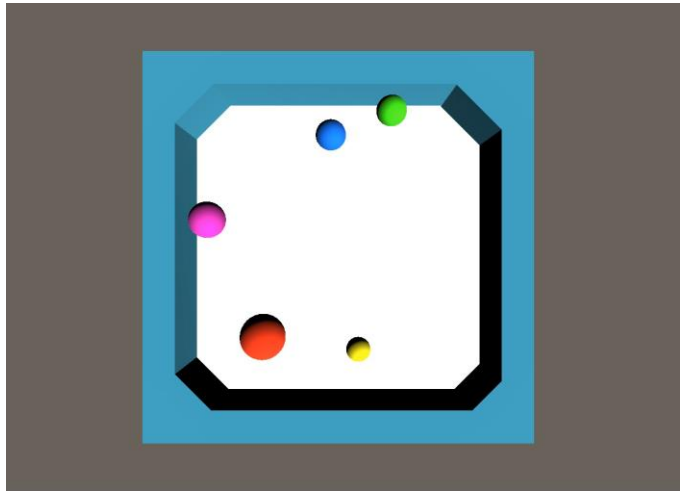
Unity ML-agents

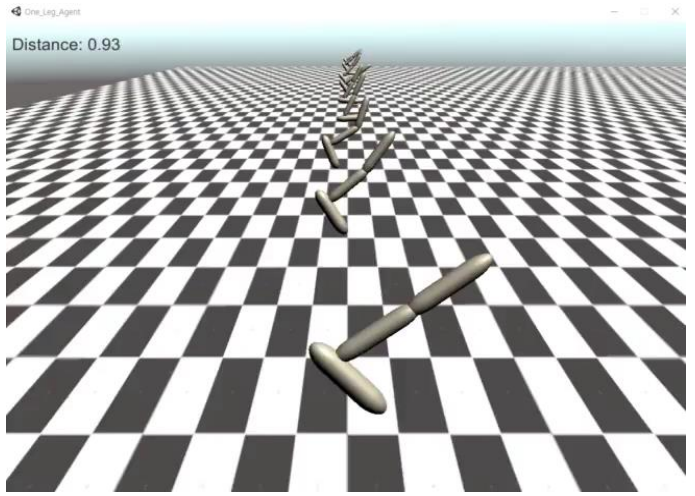


Unity ML-agents



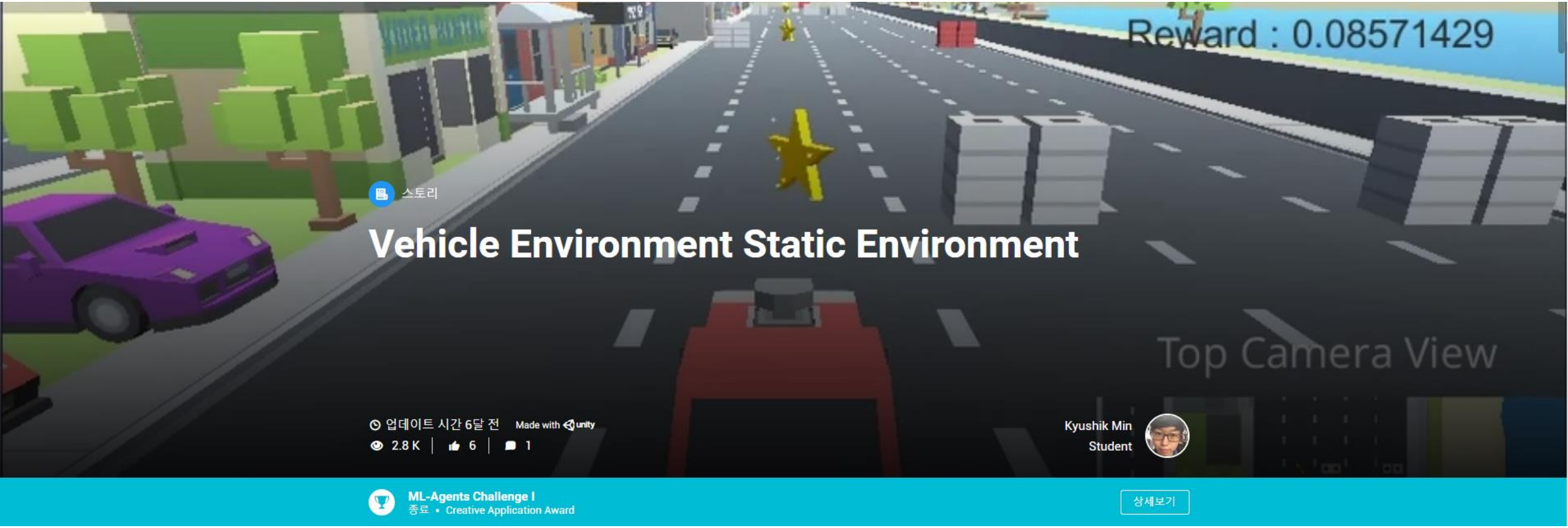




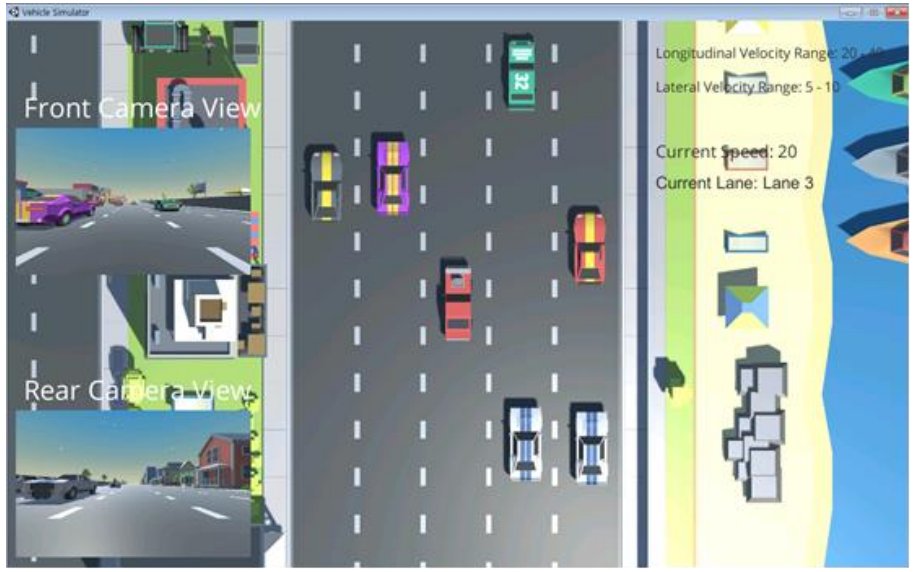


Unity ML-agents

Unity ML-agents



Unity ML-agents



unity Vehicle Simulator

Unity ML-agents

Unity ML-agents

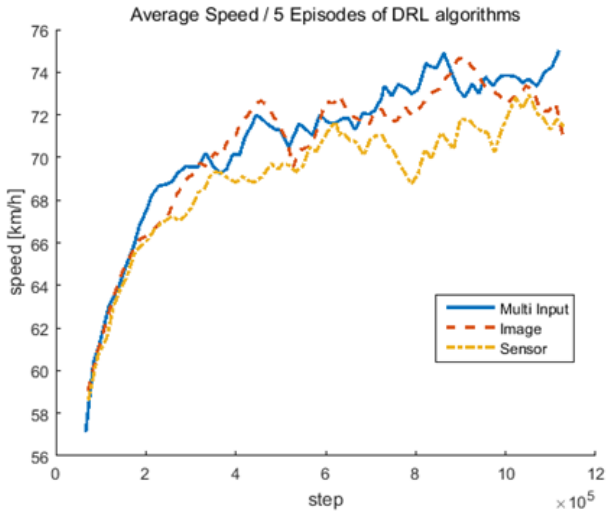
2018 IEEE Intelligent Vehicles Symposium (IV)
Changshu, Suzhou, China, June 26-30, 2018

Deep Q Learning Based High Level Driving Policy Determination

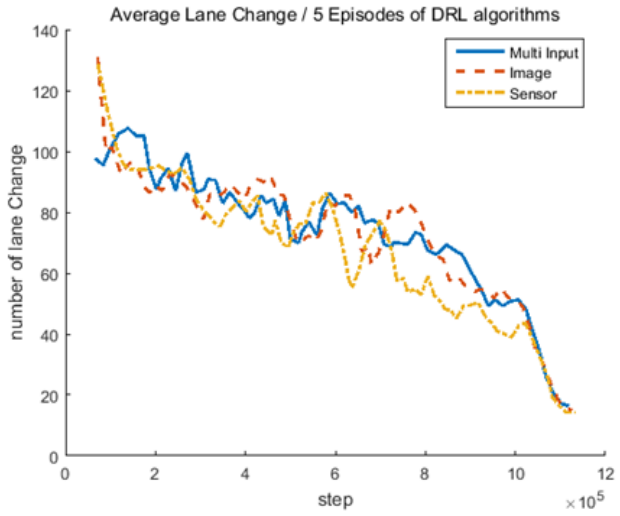
Kyushik Min, Hayoung Kim and Kunsoo Huh, *Member, IEEE*

Unity ML-agents

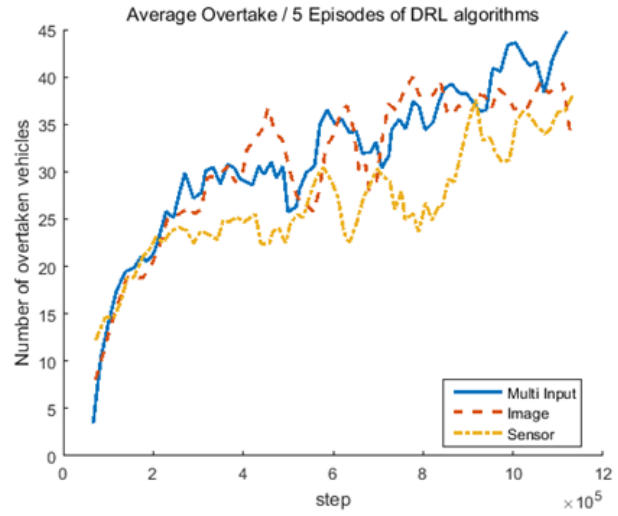
Speed



Lane Change



Overtake



Input Configuration	Average Speed (km/h)	# of Average Lane Change	# of Average Overtaking
Camera Only	71.0776	15	35.2667
LIDAR Only	71.3758	**14.2667**	38.0667
Multi-Input	**75.0212**	19.4	**44.8**

IEEE TRANSACTIONS ON INTELLIGENT VEHICLES, VOL. -, NO. -, AUGUST 2019 1

Deep Distributional Reinforcement Learning Based High Level Driving Policy Determination

Kyushik Min, Hayoung Kim, Kunsoo Huh, *Member, IEEE*



IEEE Transactions on Intelligent Vehicles (T-IV)

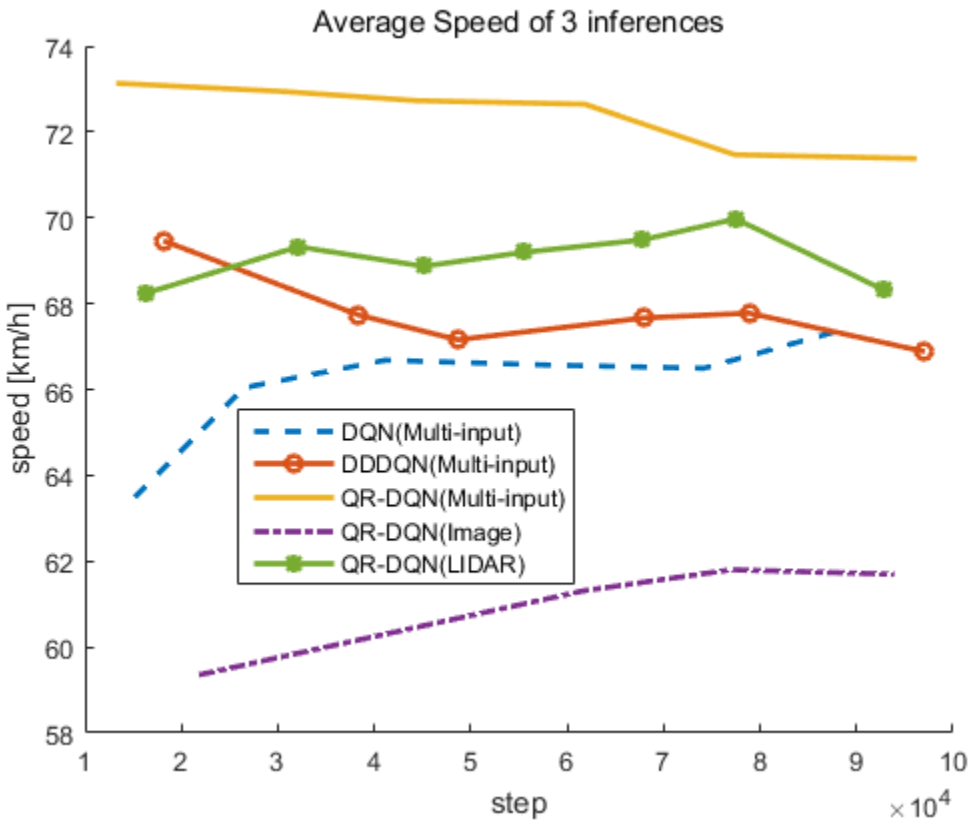
- T-IV
- Editorial Board
- Author Information

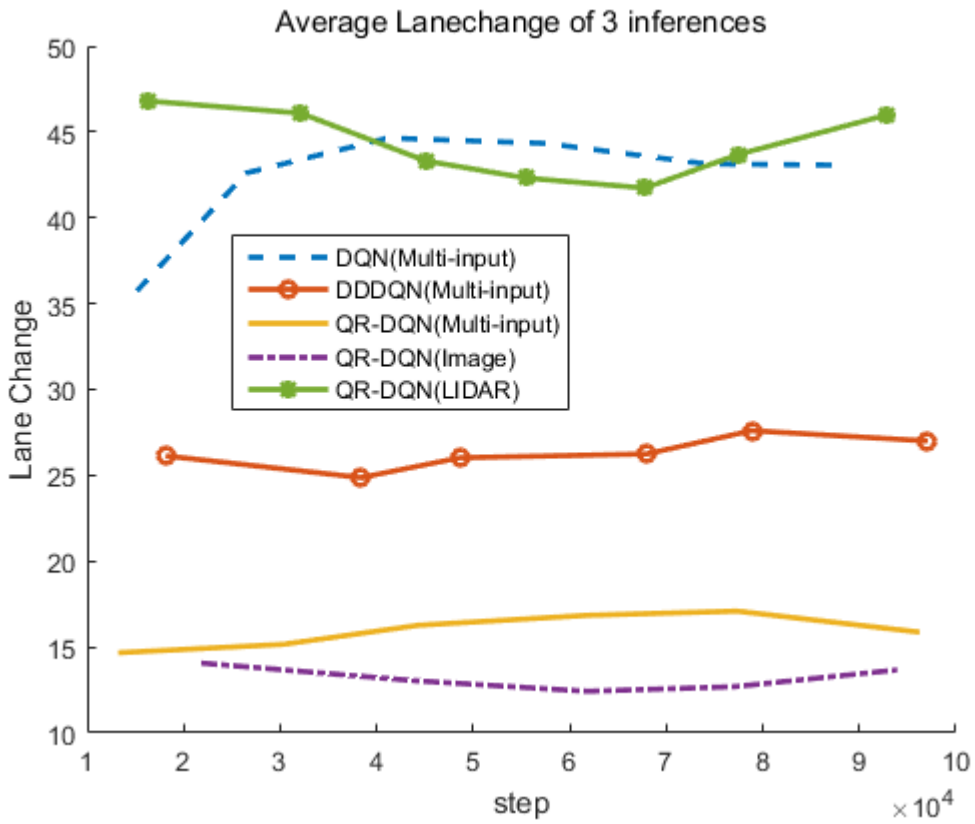
IEEE Transactions on Intelligent Vehicles is a new journal for the field of ITS.

Transactions on Intelligent Vehicles will be published four times per year.

Scope
The IEEE Transactions on Intelligent Vehicles (T-IV) publishes peer-reviewed articles that provide innovative research concepts and application results, report significant theoretical findings and application case studies, and raises awareness of pressing research and application challenges in the areas of intelligent vehicles, and in particular in automated vehicles.

Unity ML-agents

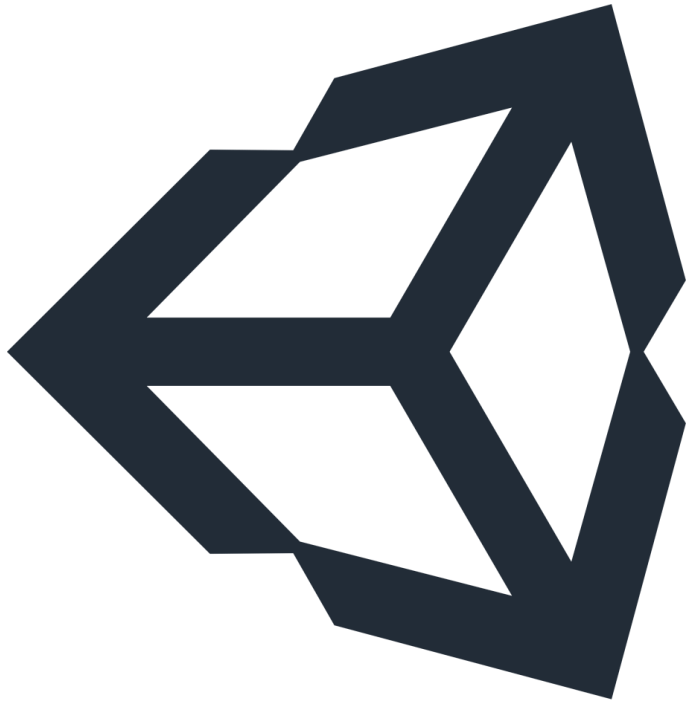




Unity ML-agents

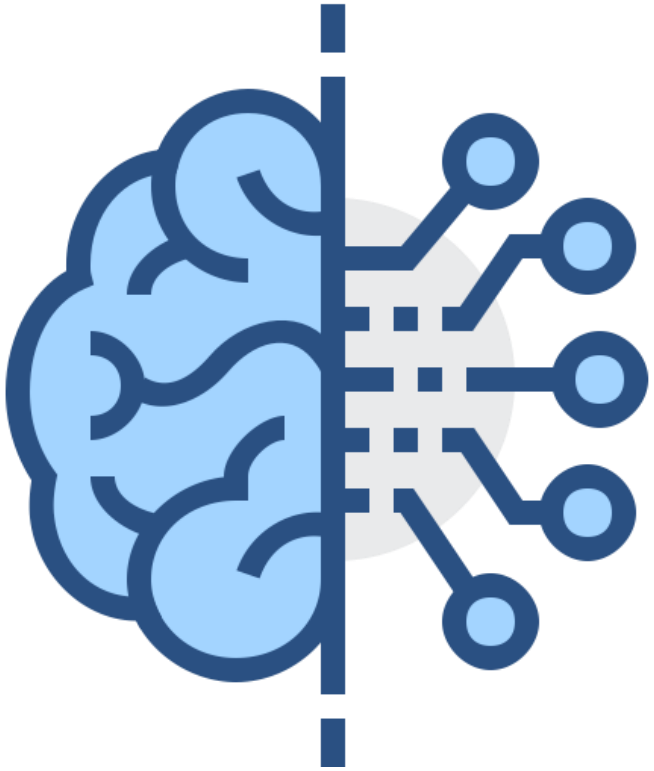
GitHub

https://github.com/MLJejuCamp2017/DRL_based_SelfDrivingCarControl



Tutorial

Unity ML-agents



인공지능(RL)

게임 개발

ML-agents Tutorial Team

~ Reinforcement Learning Korea ~

RL KOREA


민규식


정규석

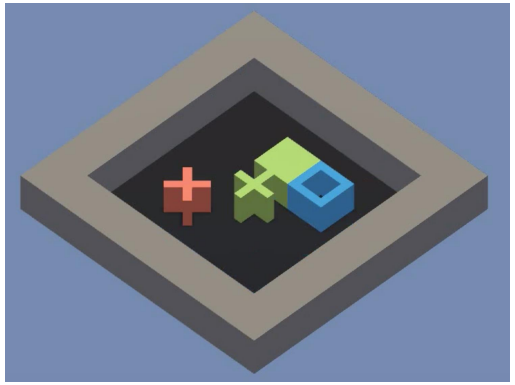

신명재


이현호


윤성훈


조동헌

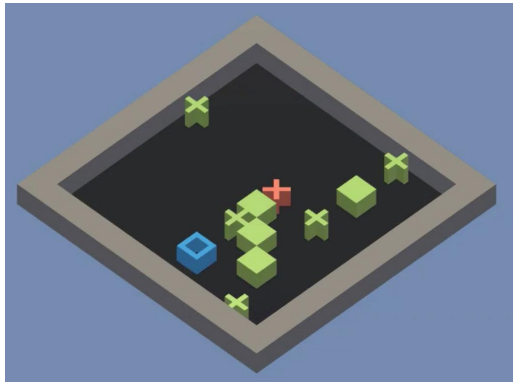
Tutorial



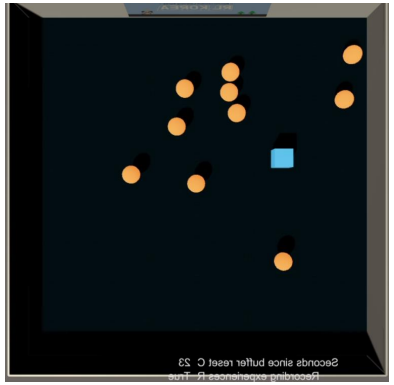
Discrete Action

Continuous Action

Adversarial Agents



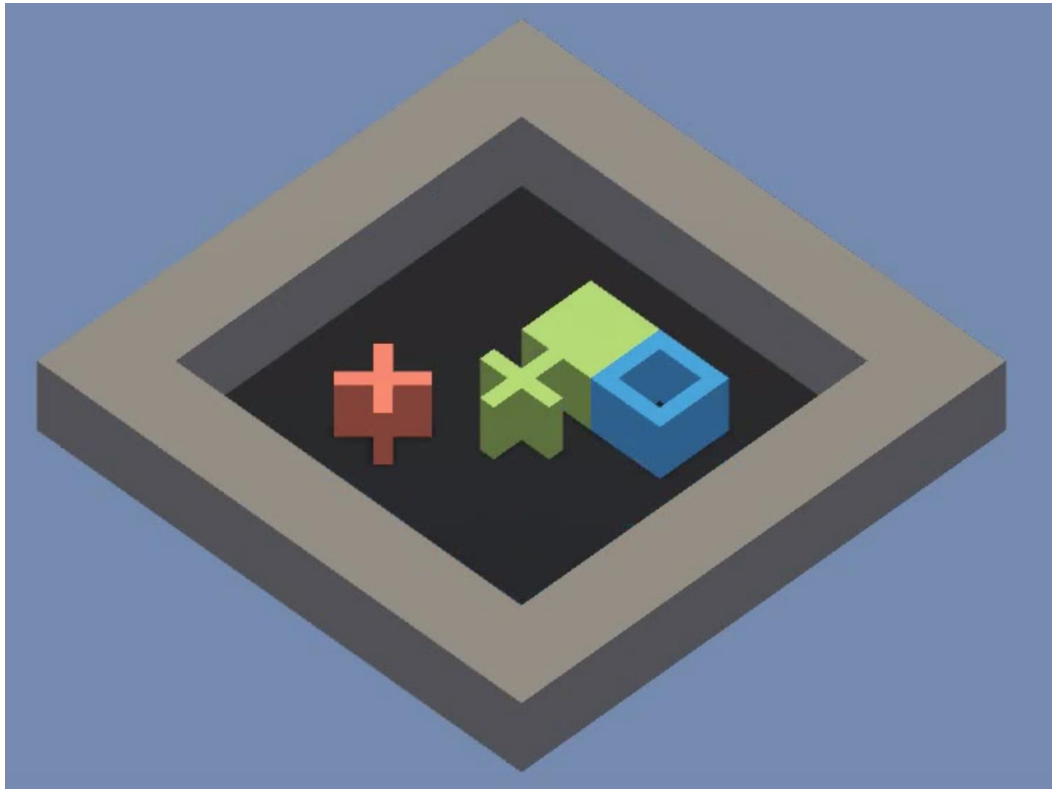
Curriculum Learning



Imitation Learning

Multi-agents

Sokoban

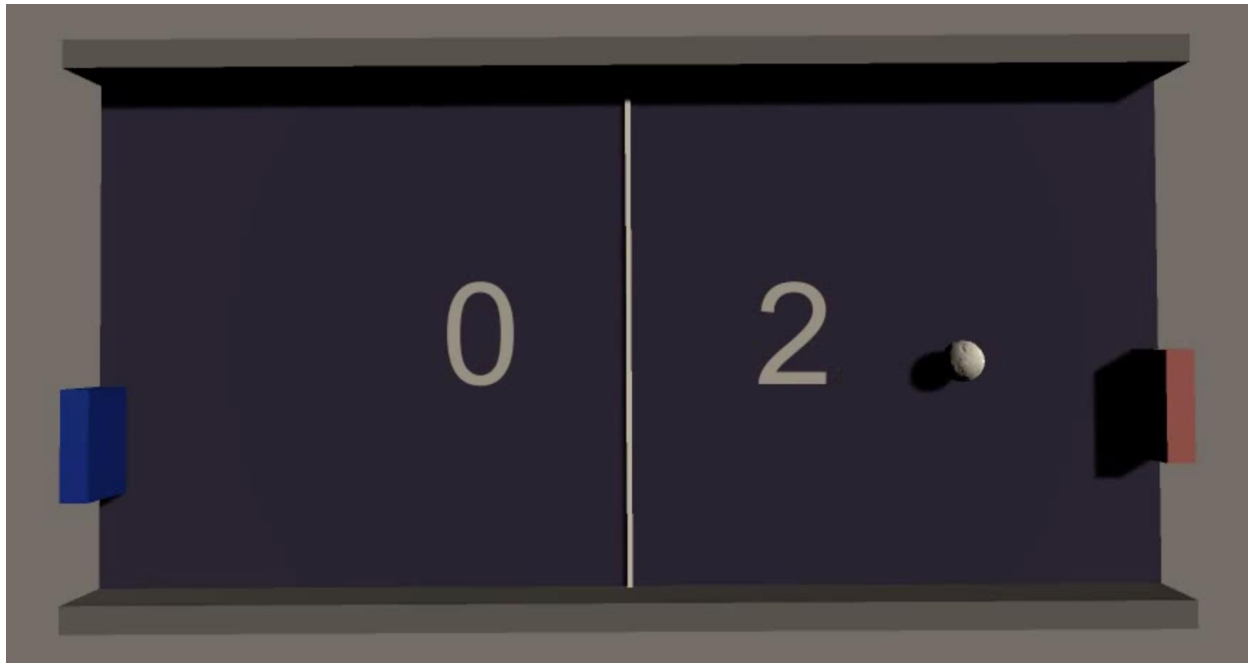


- ML-agents의 example 환경인 GridWorld 환경을 수정하여 Sokoban으로 변경
- Discrete action 환경
- 파란 네모가 초록색 박스를 밀어서 +에 넣는 것이 목표
- 박스가 x에 들어가거나 agent가 +나 x에 들어가면 감점
- resetParameter를 통해 게임판 사이즈, +의 개수, 박스의 개수, x의 개수 조절 가능
- DQN(Deep Q Network)을 이용하여 학습 수행

Drone

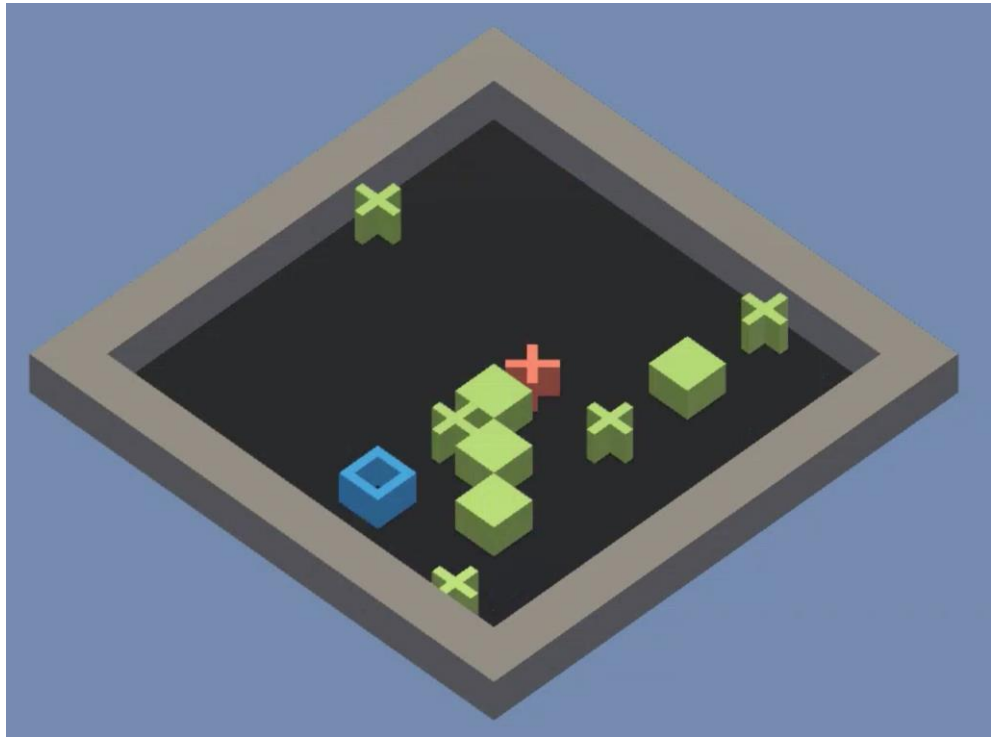
- Unity Asset store에서 받은 드론 모델을 이용하여 제작
- 드론이 하얀 구체까지 날아가는 것이 목표
- 상하좌우, 전후 방향으로 이동
- Deep Deterministic Policy Gradient (DDPG) 알고리즘을 통해 학습
- 네트워크에 노이즈를 섞는 방식으로 exploration 수행

Pong



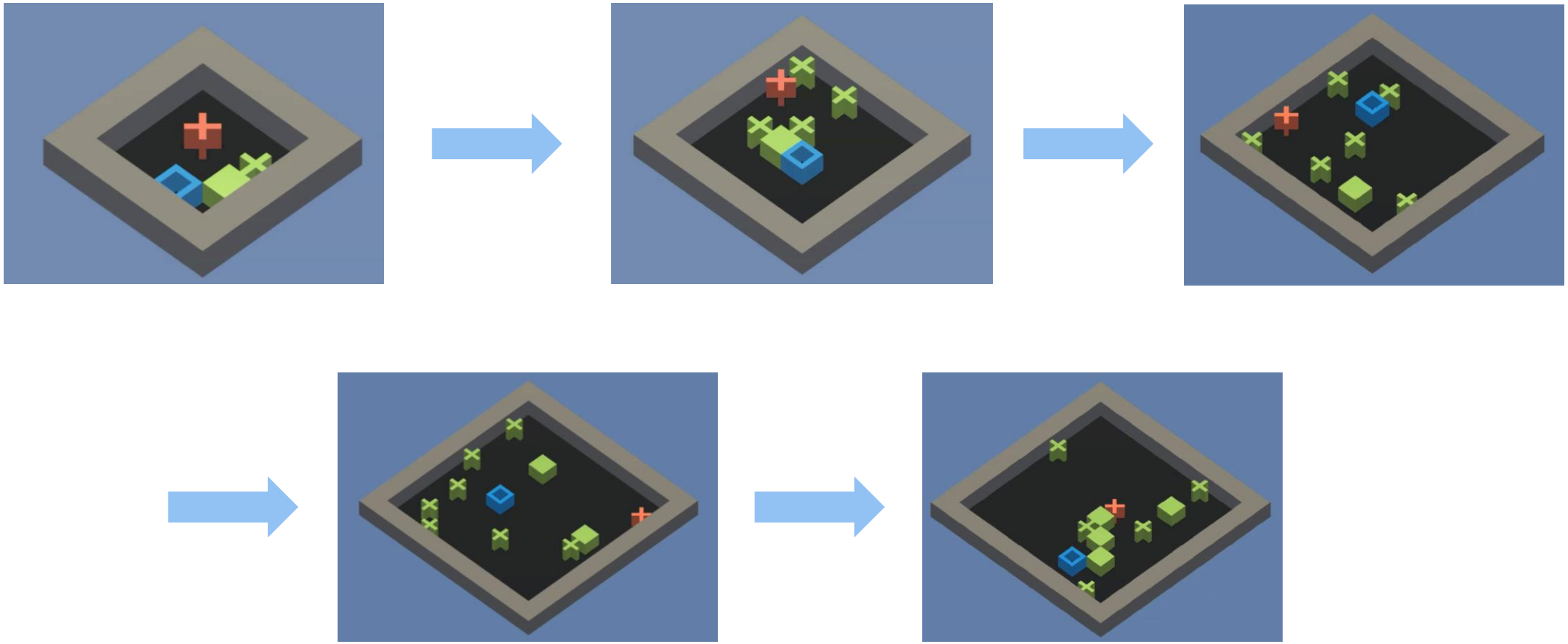
- 두개의 agents가 서로 대결하는 환경
- 각각의 agent에 brain을 설정하여 학습 수행
- 각 막대가 공을 받아치는 것이 목표
- 공을 받아치면 +0.5, 공을 놓치면 -1, 상대방이 공을 놓피면 +1의 reward를 받음
- 각각의 알고리즘을 DQN으로 학습 수행

Sokoban Curriculum

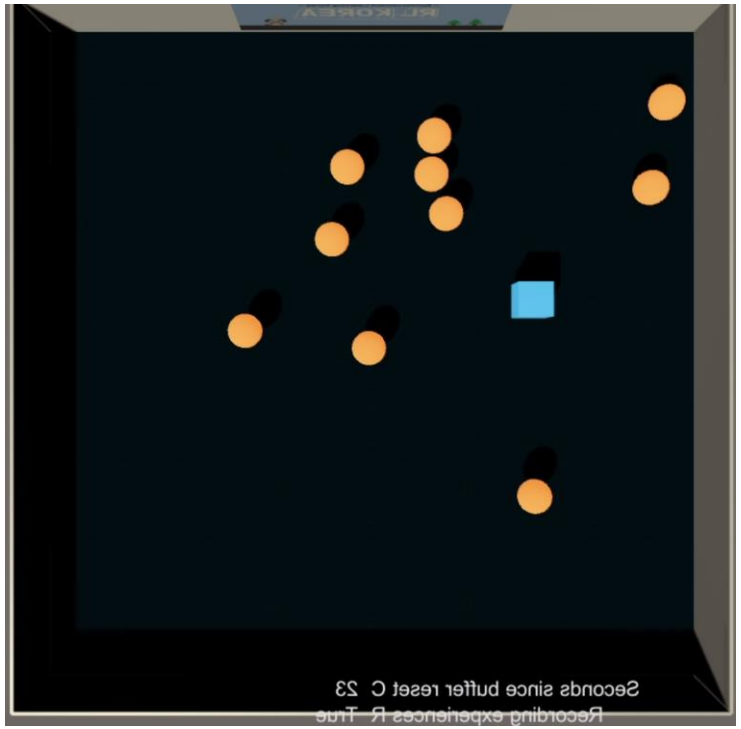


- 소코반 환경을 변형하여 curriculum 환경으로 제작
- 학습을 진행할수록 게임판의 크기를 늘리고 +의 수와 박스의 수도 함께 늘리는 방식으로 난이도 향상
  - level-0 : "gridSize": 4, "numGoals": 1, "numBoxes": 1, "numObstacles": 1
  - level-1 : "gridSize": 6, "numGoals": 4, "numBoxes": 1, "numObstacles": 1
  - level-2 : "gridSize": 8, "numGoals": 6, "numBoxes": 1, "numObstacles": 1
  - level-3 : "gridSize": 10, "numGoals": 7, "numBoxes": 2, "numObstacles": 1
  - level-4 : "gridSize": 10, "numGoals": 6, "numBoxes": 3, "numObstacles": 1
  - level-5 : "gridSize": 10, "numGoals": 5, "numBoxes": 4, "numObstacles": 1
- Dueling Double DQN 알고리즘을 통해 학습

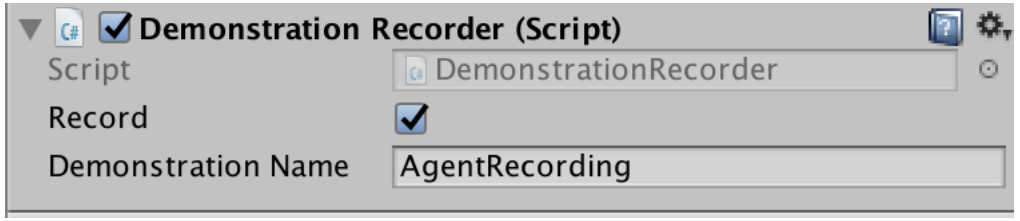
Tutorial



Dogde 환경

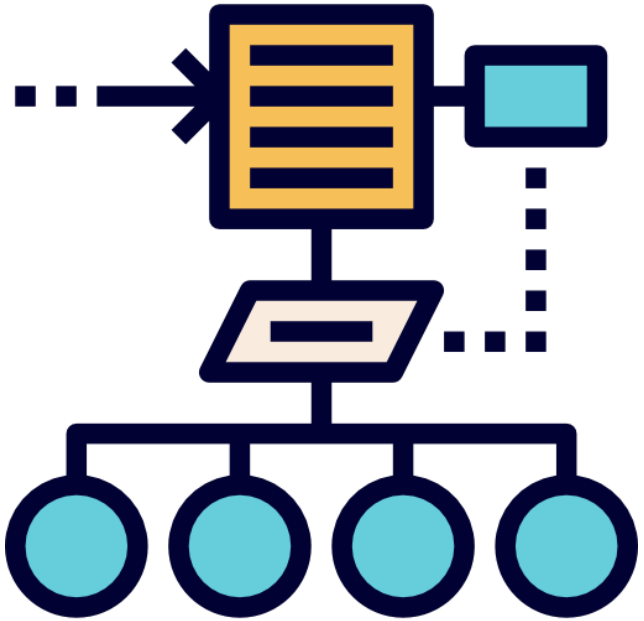


- 파란 네모가 상하좌우로 움직이면서 오렌지색 공을 피하는 환경
- 사람의 플레이를 recording demonstration 코드를 통해 기록
- State, action, reward, done 정보를 기록하여 .demo 파일로 저장
- Python 상에서 해당 정보를 로드 후 Supervised learning 기반 학습 수행
- Action 결과를 softmax를 통해 확률로 변환 후 확률적으로 action 선택



Tutorial

제작된 환경



알고리즘

Tutorial

https://github.com/reinforcement-learning-kr/Unity_ML_Agents

Thank You!

https://www.facebook.com/groups/ReinforcementLearningKR/