The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 24, No. 2, pp.205-211, Apr. 30, 2024. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2024.24.2.205
JIIBC 2024-2-30

유니티 ML-Agents를 이용한 강화 학습 기반의 지능형 에이전트 구현

Implementation of Intelligent Agent Based on Reinforcement Learning Using Unity ML-Agents

이영호*

Young-Ho Lee*

요 약 본 연구는 유니티 게임 엔진과 유니티 ML-Agents를 이용하여 강화 학습을 통해 목표 추적 및 이동을 지능적으로 수행하는 에이전트를 구현하는 데 목적이 있다. 본 연구에서는 에이전트의 효과적인 강화 학습 훈련 방식을 모색하기 위해 단일 학습 시뮬레이션 환경에서 하나의 에이전트를 트레이닝하는 방식과 다중 학습 시뮬레이션 환경에서 여러 에이전트들을 동시에 병렬 트레이닝하는 방식 간의 학습 성능을 비교하기 위한 실험을 수행하였다. 실험 결과를 통해 병렬 트레이닝 방식이 싱글 트레이닝 방식보다 학습 속도 측면에서 약 4.9배 빠르고, 학습 안정성 측면에서도 더 안정적으로 효과적인 학습이 일어남을 확인할 수 있었다.

Abstract The purpose of this study is to implement an agent that intelligently performs tracking and movement through reinforcement learning using the Unity and ML-Agents. In this study, we conducted an experiment to compare the learning performance between training one agent in a single learning simulation environment and parallel training of several agents simultaneously in a multi-learning simulation environment. From the experimental results, we could be confirmed that the parallel training method is about 4.9 times faster than the single training method in terms of learning speed, and more stable and effective learning occurs in terms of learning stability.

Key Words : AI, Intelligent Agent, ML-Agents, Reinforcement Learning, Unity

Ⅰ. 서 론

현재 인공지능은 다양한 산업 및 서비스의 지능화를 목표로 널리 활용되고 있으며, 이에 따라 인공지능 기술을 적용한 지능형 모델에 관한 연구도 활발히 진행되고 있다. 게임 분야에서도 인공지능 적용 사례는 흔히 찾아볼 수 있을 만큼 과거에서부터 현재까지 게임과 인공지능은 지속적인 발전을 거듭하며 서로 오랜 역사를 공유해왔다.

인공지능 기술이 점차 고도화됨에 따라 게임의 지능화를 실현할 수 있게 되었고, 이는 사용자에게 현실에 가까울 법한 피드백을 선사하며 게임에 대한 몰입감을 극대화하게 만든다[1]. 그럼에도 중요한 게임 콘텐츠 요소 중 하나인 NPC(Non-Player Character)는 이동, 추적 등

*정회원, 배재대학교 게임공학과
접수일자 2024년 3월 20일, 수정완료 2024년 3월 29일
게재확정일자 2024년 4월 5일

Received: 20 March, 2024 / Revised: 29 March, 2024 /
Accepted: 5 April, 2024
*Corresponding Author: brain@pcu.ac.kr
Dept. of Game Engineering, PaiChai University, Korea

의 행동에 있어 대부분 프로그램된 패턴을 가지고 있다. 이러한 패턴을 플레이어가 파악하는 경우, 게임 플레이가 단조로워지거나 NPC를 상대로 하는 전략성이 떨어질 수 있다. 또한, NPC가 비효율적인 이동이나 상황에 맞지 않는 부적절한 행동을 하게 될 수 있다. 이러한 요인들은 결국 게임의 재미나 난이도를 저하시키는 문제를 야기한다. 따라서, 이러한 문제를 해소하기 위해서는 NPC의 인공지능을 고도화하여 더욱 지능적인 행동을 수행할 수 있도록 개선해야 할 필요가 있다.

게임에서 NPC의 행동을 지능화하는 전통적인 방식으로 FSM(Finite State Machine)이 있다. FSM은 구현이 용이하고 단순하다는 장점이 있어 NPC 인공지능을 구현하는 데 가장 보편적으로 사용되었다. 그러나, FSM은 상태의 무한한 증가, 유지관리의 어려움, NPC 행동에 대한 예측 가능성 등 많은 단점이 존재한다[2].

머신 러닝 알고리즘은 과거 플레이로부터 학습하고 게임을 통과할 때마다 더 향상되므로, 신속하게 최상의 결과를 얻을 수 있다는 장점을 가지고 있다[3]. 또한, 머신 러닝 기술은 행동의 자동 생성 및 선택을 용이하게 하여 NPC의 행동 역학을 개선시키는 방법을 제공함으로써 게임 인공지능의 기능을 향상시키고 더 매력적이고 재미있는 게임 플레이를 경험할 수 있는 기회를 제공한다[4]. 최근에는 머신 러닝의 한 부류인 강화 학습을 이용하여 향상된 인공지능을 구현하고 있다. 강화 학습은 비교적 단시간에 효과적으로 에이전트(agent)를 학습시킬 수 있다는 장점이 있다.

본 연구에서는 유니티(Unity) 게임 엔진과 유니티 ML-Agents(Machine Learning Agents)를 이용하여 학습 환경과 학습 알고리즘을 구현하고, 강화 학습 훈련을 통해 탐색 및 이동을 지능적으로 수행하는 에이전트를 구현하고자 한다.

II. 관련 연구

1. 강화 학습

강화 학습은 지도 학습, 비지도 학습과 함께 머신 러닝 기법 중 하나이며, 에이전트가 환경과 상호작용하며 시행착오와 보상을 통해 최대한 보상을 많이 받는 행동을 학습하게 된다. 강화 학습에서 에이전트는 어떻게 행동하는지 배우지 않고 행동하는 방법에 있어 자유로운 선택을 한다. 대신 에이전트가 선택한 행동이 좋은지 나쁜지를 보상을 통해 알려준다[5].

강화 학습에 필요한 구성 요소로는 '에이전트'와 '환경'이 있다. 에이전트는 강화 학습의 주체로서 현재 환경의 상태를 인식하고 행동을 결정하고, 환경은 에이전트에게 현재 상태와 에이전트가 선택한 행동에 대한 보상을 주는 역할을 한다. 에이전트와 환경 간의 상호작용은 상태(state), 행동(action), 보상(reward)으로 이루어진다.

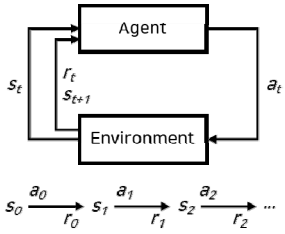

그림 1. 강화 학습 과정
Fig. 1. Reinforcement Learning Process

그림 1은 강화 학습 수행 과정을 나타낸 것으로, t는 시간, s는 상태, a는 행동, r은 보상을 의미한다. 먼저, 환경은 현재 환경의 상태 정보 s_t를 에이전트에 전달한다. 에이전트는 이러한 상태 정보를 바탕으로 에이전트가 취할 수 있는 행동들 중 최적의 행동 a_t를 선택하여 결정한다. 에이전트가 행동을 수행하면, 이에 따라 환경의 상태는 변하게 된다. 환경은 에이전트가 결정한 행동에 대한 보상 r_t와 새로 바뀐 환경의 상태 정보 s_{t+1}을 에이전트에게 반환한다. 이러한 과정을 반복해서 수행함에 따라 보상을 최대화하는 방향으로 에이전트가 점차 학습하게 된다[6][7].

특정 상태에서 에이전트가 어떤 행동을 선택할지는 정책(policy)에 의해 결정된다. 강화 학습의 목적은 에이전트가 앞으로 누적될 보상을 최대화하는 일련의 행동을 결정하는 최적의 정책을 획득하는 것이다[8].

2. 유니티 ML-Agents

유니티 ML-Agents는 심층 강화 학습 및 모방 학습을 사용하여 지능형 에이전트를 훈련하기 위한 게임 및 시뮬레이션 환경을 제공하는 플러그인 툴킷이다[9]. ML-Agents를 사용하여 지능형 에이전트를 구현하려면 우선 강화 학습 훈련을 진행할 수 있는 시뮬레이션 환경이 필요하다. 이러한 시뮬레이션 환경은 유니티 에디터를 이용하여 제작할 수 있으며, 파이썬(Python) API를 통해 상호작용할 수 있다.

ML-Agents의 세 가지 핵심 요소는 '센서', '에이전트', '아카데미(academy)'로 분류된다. 에이전트는 학습

대상이 되는 오브젝트로서 관측된 정보를 수집하고, 행동을 수행하며, 보상을 받는 역할을 담당한다. 에이전트는 렌더링(rendering)된 이미지, 레이캐스트(raycast) 결과 또는 임의의 길이 벡터(vector)와 같은 다양한 형태의 정보에 반응하는 다양한 센서를 사용하여 관측 정보를 수집할 수 있다. 아카데미는 시뮬레이션 단계를 추적하고, 에이전트를 관리하며, 환경 시뮬레이션을 전체적으로 조율하는 역할을 담당한다. 또한, 아카데미는 런타임(runtime) 시 환경 설정을 변경하는 데 사용되는 환경 파라미터(parameters)를 정의하는 기능을 포함하고 있다[10].

ML-Agents의 학습 환경은 '에이전트', '브레인', '아카데미'로 구성된다. 에이전트는 상태 정보와 관측에 의한 데이터를 가지고 있고, 학습 환경 내에서 고유의 행동을 수행하며, 이에 대한 적절한 보상을 받게 된다. 브레인은 특정 상태와 행동 공간을 정의하고, 연결된 에이전트가 어떤 행동을 취할지에 대한 정책을 결정한다. 아카데미는 학습 환경에 포함된 모든 브레인을 하위 요소(자식 오브젝트)로 포함하며, 브레인과 에이전트, 그리고 학습 환경의 에피소드(episode)를 관리한다. 브레인이 'external'로 설정된 에이전트들의 상태와 관측 정보는 아카데미에 존재하는 외부 커뮤니케이터(external communicator)에 의해 수집된다. 또한, 머신 러닝 라이브러리를 사용하여 처리할 때에도 외부 커뮤니케이터를 통해 파이썬 API와 통신하게 된다. 학습 환경 구조 예시는 그림 2와 같다[11].

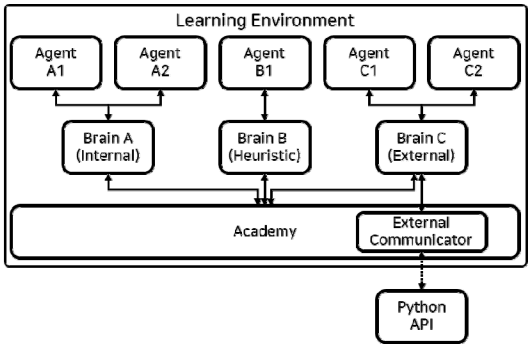

그림 2. 학습 환경 구조 예시
Fig. 2. Example of a Learning Environment Framework

3. Agent 클래스의 주요 메서드

ML-Agents를 이용하여 유니티 환경 내에서 에이전트가 강화 학습 훈련을 수행하도록 하기 위해서는 학습 알고리즘이 정의된 스크립트를 구현하여 에이전트에 적용해야 한다. Agent 클래스는 표 1과 같이 에이전트 및 학습 환경 초기화, 관측을 통한 주변 환경 정보 수집, 브레인에 의해 결정된 행동 수행 등 에이전트 학습을 위한 다양한 메서드들을 제공하고 있으며, 이러한 메서드를 게임 및 시뮬레이션 환경에 맞게 오버라이딩(overriding)하여 사용할 수 있다.

표 1. Agent 클래스의 기본 메서드
Table 1. Basic Methods of Agent Class

메서드	기능
Initialize()	환경 정보 관측을 위한 변수 초기화
OnEpisodeBegin()	에이전트 및 학습 환경 초기화
CollectObservation()	환경 정보 관측 및 수집
OnActionReceived()	에이전트가 수행할 동작 정의
Heuristic()	테스트용 수동 컨트롤 로직 정의

Ⅲ. 학습 시뮬레이션 환경 구현

1. 학습 환경 플랫폼 제작

Unity 2021.3.0f1을 사용하여 3D 가상 공간에 학습 환경 플랫폼을 제작하였다. 바닥에 해당하는 'Floor', 목표물에 해당하는 'Target', 훈련용 에이전트에 해당하는 'Agent'를 표 2와 같이 설정, 배치하였고, 이를 'Area'라는 이름으로 그룹화하여 학습 환경 플랫폼을 구현하였다. 에이전트의 사실적인 물리적 움직임을 구현하기 위해 Agent에 Rigidbody 컴포넌트를 추가하였다. 이에 따라 에이전트가 이동 중 바닥 영역 밖으로 벗어났을 경우, 중력에 의해 아래로 추락하게 된다.

본 연구에서 구현한 학습 환경 플랫폼은 장애물 등 다양한 오브젝트를 추가하여 학습 환경을 재구성하거나 확장하기 용이하다는 장점을 가지고 있다.

표 2. 학습 환경 플랫폼 구성
Table 2. Configuration for the learning environment platform

오브젝트 이름	오브젝트 유형	초기 설정
Floor	Plane	Position: 0, 0, 0 Scale: 1, 1, 1
Target	Cube	Position: 3, 0.5, 3 Scale: 1, 1, 1
Agent	Sphere	Position: 0, 0.5, 0 Scale: 1, 1, 1

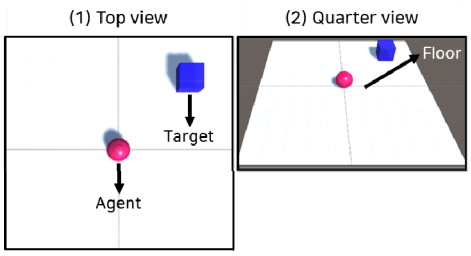

그림 3. 학습 환경 플랫폼 구조
Fig. 3. Learning environment platform structure

2. 스크립트 구현 및 컴포넌트 설정

유니티 에디터에서 C# 스크립트 파일을 생성한 후, Microsoft Visual Studio Community 2019를 이용하여 에이전트 초기화, 관측 정보 수집, 결정된 행동 수행 및 보상 처리 등을 수행하는 코드를 다음과 같은 절차대로 작성하였다.

첫째, Agent 클래스의 메서드를 오버라이딩하여 사용하기 위해 Agent 클래스를 부모 클래스로 선언하였다. 둘째, 오브젝트의 컴포넌트를 참조하기 위한 변수와 에이전트의 이동력을 설정하기 위한 변수를 선언하였다. 셋째, 에이전트 및 목표물의 초기화를 위해 OnEpisodeBegin() 메서드를 정의하였다. 에피소드 시작 시, 에이전트의 위치는 항상 바닥 중앙에 위치하도록 설정하였고 이동 속도와 회전 속도는 0이 되도록 설정하였다. 목표물이 생성되는 위치는 바닥 영역 내에서 y축을 제외하고 랜덤하게 변경되도록 설정하였다. 넷째, 에이전트의 위치와 이동 속도(velocity), 목표물의 위치 정보를 수집하여 브레인에 전달하기 위해 CollectObservations() 메서드를 정의하였다. 이 메서드 수행을 통해 에이전트 x, y, z축 좌표 3개, 에이전트 x, z축 이동 속도 2개, 목표물 x, y, z축 좌표 3개 총 8개의 정보를 수집하게 된다. 다섯째, 브레인이 결정한 정책을 전달받아 에이전트의 이동을 처리하고, 이에 따른 적절한 보상을 주기 위해 OnActionReceived() 메서드를 정의하였다. 에이전트와 목표물 사이의 거리를 계산하여 거리 값이 1.5 미만인 경우에는 보상으로 1을 가산하도록 설정하였고, 에이전트가 바닥 아래로 추락했을 경우에는 현재 에피소드를 종료하도록 설정하였다. 한편, 에이전트가 목표물에 도달하는 데 소요되는 시간에 따른 벌점은 설정하지 않았다. 여섯째, 실험자가 직접 에이전트를 조종하여 학습 환경을 테스트해 볼 수 있도록 Heuristic() 메서드를 정의하였다. 상하좌우 방향키 입력을 통해 에이전트를 네 방향으로 이동할 수 있도록 설정하였다.

작성된 스크립트를 Agent 오브젝트에 적용하면, 그림 4와 같이 스크립트와 함께 'Behavior Parameters'라는 이름의 스크립트 컴포넌트가 자동으로 추가된다.

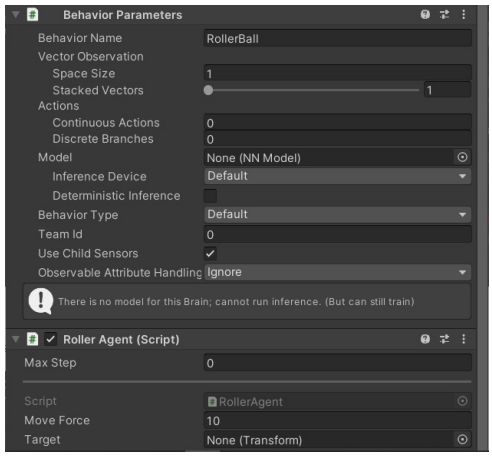

그림 4. 에이전트에 추가된 2개의 컴포넌트
Fig. 4. Two components added to the agent

에이전트의 강화 학습 환경 세팅을 위해 Behavior Parameters 스크립트 컴포넌트를 표 3과 같이 설정하였다.

표 3. Behavior Parameters 스크립트 컴포넌트 설정
Table 3. Set-up for the Behavior Parameters component

속성	설정 값
Space Size	8
Continuous Actions	2

수집하는 관측 정보의 개수는 총 8개이므로, Space Size를 8로 설정하였다. 에이전트가 수행할 행동은 x축 및 z축 방향으로의 이동이며, 좌표 값은 비이산적인(non-discrete) 데이터, 즉 연속적인(continuous) 데이터이므로, Continuous Actions를 2로 설정하였다.

Agent에 적용한 Roller Agent 스크립트 컴포넌트에 대하여 표 4와 같이 설정하였다.

표 4. Roller Agent 스크립트 컴포넌트 설정
Table 4. Set-up for the Roller Agent component

속성	설정 값
Max Step	5000
Target	Target(Transform)

에피소드당 최대 행동 시도 횟수를 지정하기 위해 Max Step을 5000으로 설정하였다. 에이전트가 바닥을 벗어나 추락했을 경우나 목표물에 도달했을 경우, Max Step 설정 값과 무관하게 해당 에피소드를 종료하고 새로운 에피소드를 시작하게 된다. 에이전트가 추적할 목표물의 위치를 지정하기 위해 Target 속성에 Target 오브젝트를 할당하여 Transform 정보를 참조할 수 있게 하였다. 본 연구의 에이전트는 수집된 관측 정보를 기반으로 어떻게 행동해야 할지 결정하기 위해 브레인에게 요청해야 하므로, Agent 오브젝트에 그림 5와 같이 Decision Requester 스크립트 컴포넌트를 설정하였다.

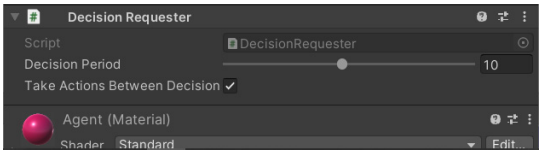

그림 5. Decision Requester 스크립트 컴포넌트 설정
Fig. 5. Set up for the Decision Requester component

3. 강화 학습 트레이너 설정

본 연구에서는 에이전트의 강화 학습 훈련을 위해 PPO(Proximal Policy Optimization) 알고리즘을 적용하였다. 이에 따라 그림 6과 같이 trainer_type을 'ppo'로 설정하고, 효과적인 훈련이 가능할 수 있도록 관련 파라미터들을 적절한 값으로 설정하여 트레이너 설정 파일(.yaml)을 편집하였다. 트레이너 설정은 깃허브(GitHub) 리포지토리에 등록된 유니티 테크놀로지(Unity-Technologies) 공식 문서를 참조하였다[12].

```
behaviors:
  RollerBall:
    trainer_type: ppo
    hyperparameters:
      batch_size: 10
      buffer_size: 100
      learning_rate: 3.0e-4
      beta: 5.0e-4
      epsilon: 0.2
      lambd: 0.99
      num_epoch: 3
      learning_rate_schedule: linear
      beta_schedule: constant
      epsilon_schedule: linear
    network_settings:
      normalize: false
      hidden_units: 128
      num_layers: 2
    reward_signals:
      extrinsic:
        gamma: 0.99
        strength: 1.0
    max_steps: 500000
    time_horizon: 64
    summary_freq: 10000
```

그림 6. PPO 기반 강화 학습 트레이너 설정
Fig. 6. Trainer configurations based on PPO

Ⅳ. 실험 및 결과

본 연구의 강화 학습 실험은 ML-Agents 2.3.0-exp.2 패키지가 설치된 Unity 2021.3.0f1과 Python 3.9, PyTorch 1.8.2가 설치된 시스템 환경에서 진행하였다. 실험 방법은 싱글 트레이닝과 병렬 트레이닝 환경으로 나누어 에이전트를 훈련시킨 후, 실험 결과를 비교하여 학습 성능을 평가하였다. 강화 학습 훈련 실행을 위해 그림 7과 같이 Windows 명령 프롬프트 환경에서 트레이너 설정 파일을 지정하여 학습 명령을 실행하였다.

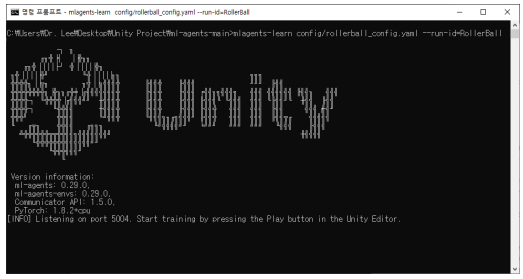

그림 7. 강화 학습 훈련 명령 실행
Fig. 7. Command execution for reinforcement learning

1. 싱글 트레이닝 강화 학습 실험

본 연구에서 학습 시뮬레이션 환경 구현을 위해 제작한 단일 학습 환경 플랫폼에서 싱글 트레이닝을 실행하였다. 총 훈련 횟수는 50만회이고, 1만 스텝(step) 단위로 훈련 실행 결과를 출력하도록 하였다. 싱글 트레이닝 과정 및 결과는 그림 8과 같으며, 강화 학습에 총 소요된 시간은 11518.702초로 산출되었다. 실험 결과를 분석하기 위해 텐서보드(TensorBoard)를 이용하여 그래프로 시각화하여 나타내었다.

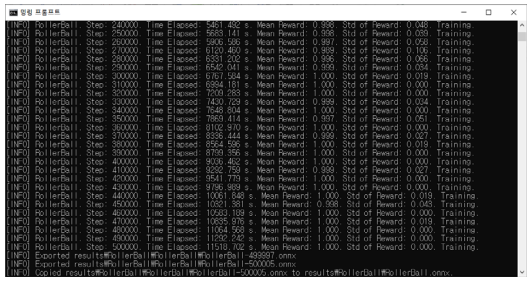

그림 8. 싱글 트레이닝 과정 및 결과
Fig. 8. Single training process and resul

그림 9는 싱글 트레이닝에 의한 강화 학습에 대한 스텝별 누적 보상치를 나타내는 그래프이다. 싱글 트레이

닝에 의한 강화 학습에서는 약 5만 번째 스텝에서 누적 보상치가 처음으로 1에 수렴하는 것을 알 수 있었다. 이후 누적 보상치가 등락을 거듭하다가 약 18만 번째 스텝에서 급격히 하락하였고, 약 39만 번째 스텝부터는 보상 누적치의 변동 폭이 작아지면서 1에 수렴하는 결과를 보였다.

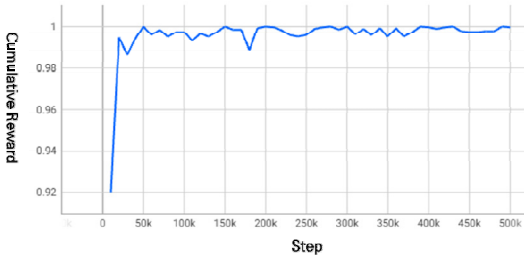

그림 9. 싱글 트레이닝 수행에 따른 누적 보상 그래프
Fig. 9. Cumulative reward graph when performing single training

2. 병렬 트레이닝 강화 학습 실험

유니티 에디터에서 기존에 구현한 단일 학습 환경 플랫폼을 복제하여 총 9개의 학습 환경 플랫폼을 구축한 후, 싱글 트레이닝과 동일한 트레이너 설정 파일을 적용하여 동일한 조건으로 병렬 트레이닝을 실행하였다. 병렬 트레이닝 과정 및 결과는 그림 10과 같으며, 강화 학습에 총 소요된 시간은 2349.574초로 산출되었다. 실험 결과를 분석하기 위해 텐서보드를 이용하여 그래프로 나타내었다.

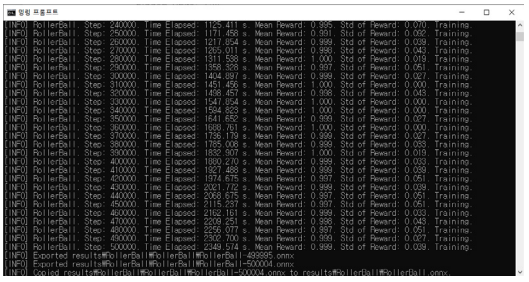

그림 10. 병렬 트레이닝 과정 및 결과
Fig. 10. Parallel training process and results

그림 11은 병렬 트레이닝에 의한 강화 학습에 대한 스텝별 누적 보상치를 나타내는 그래프이다. 병렬 트레이닝에 의한 강화 학습에서는 약 7.5만 번째 스텝에서 누적 보상치가 처음으로 1에 수렴함을 보였다. 10만 번째 스텝 이후부터 누적 보상치가 등락을 거듭하는 추세를 보이면서 약 18만 번째 스텝에서 급격히 하락하였고, 약 26만 번째 스텝부터는 보상 누적치의 변동 폭이 작아지면서 1에 수렴하는 결과를 보였다.

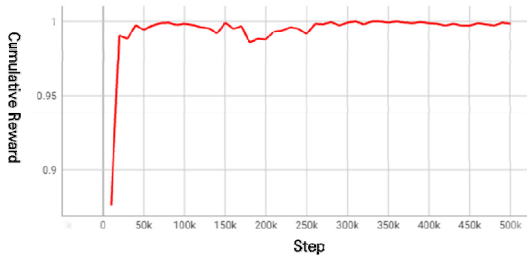

그림 11. 병렬 트레이닝 수행에 따른 누적 보상 그래프
Fig. 11. Cumulative reward graph when performing parallel training

3. 학습 성능 비교 및 평가

병렬 트레이닝 방식과 싱글 트레이닝 방식의 학습 소요 시간을 비교했을 때, 각각 2349.574초와 11518.702초로, 학습 속도 측면에서 병렬 트레이닝 방식이 싱글 트레이닝 방식보다 약 4.9배 빠름을 알 수 있었다. 한편, 싱글 트레이닝 방식이 병렬 트레이닝 방식보다 누적 보상치가 1에 도달하는 시점이 빨랐으나, 이는 일시적인 결과이기 때문에 싱글 트레이닝 방식의 강화 학습이 더 효과적이라고 판단하기는 어렵다. 이러한 이유로 학습 안정성을 검토하게 되었으며, 그래프 중반 이후부터 누적 보상치가 다소 안정적으로 1에 수렴하는 병렬 트레이닝 방식이 더 효과적으로 학습이 이루어진다는 결론을 얻었다. 따라서, 학습 속도 및 안정성 분석 결과를 종합해 보았을 때, 병렬 트레이닝에 의한 강화 학습 방식이 싱글 트레이닝에 의한 강화 학습 방식보다 학습 성능이 우수하다는 결론을 도출할 수 있었다.

V. 결 론

본 연구에서는 게임 NPC의 지능화를 위해 유니티 에디터를 이용하여 학습 시뮬레이션 환경을 만들고, 유니티 ML-Agents를 이용하여 강화 학습을 통해 지능형 에이전트를 구현하였다. 또한, 강화 학습 실행 시 단일 학습 시뮬레이션 환경에서 하나의 에이전트를 훈련하는 싱글 트레이닝 방식을 적용했을 경우와 다중 학습 시뮬레이션 환경에서 여러 에이전트들을 동시에 훈련하는 병렬 트레이닝 방식을 적용했을 경우의 강화 학습 성능을 비

교하기 위한 실험을 수행하였다. 본 연구의 실험 결과, 병렬 트레이닝 방식이 싱글 트레이닝 방식보다 학습 속도 및 안정성 측면에서 더 성능이 우수함을 알 수 있었다. 본 연구를 통해 게임의 전략적 재미와 난이도 향상을 위한 지능형 NPC 구현에 관한 연구와 개발이 게임 산업계 전반에서 더욱 촉진되길 기대한다.

향후 연구 방향으로는 트레이너 파일의 파라미터 세트 설정이 강화 학습 성능에 미치는 영향에 관한 연구와 SAC(Soft Actor-Critic) 알고리즘을 적용한 강화 학습 실험을 통해 PPO 알고리즘과 학습 성능을 비교하여 분석해 보는 연구가 될 것이다.

References

[1] Sung-Won Ahn, "Users who are Passionate about Artificial Intelligence - AI that Changed the Game Industry", Technology and Innovation, Vol. 447, pp. 20-23, 2021.

[2] Concepcion, A. I., Munoz, E., Hawkins, M., and Balane, D., "Using an Inference Engine for AI in the Office Tactics Video Game", Proceedings on the International Conference on Artificial Intelligence, pp. 1-7, 2014.

[3] Jagdale, D. "Finite State Machine in Game Development", International Journal of Advanced Research in Science, Communication and Technology, Vol. 10, No. 1, pp. 384-390, 2021.
DOI: https://doi.org/10.48175/ijarsct-2062

[4] Galway, L., Charles, D., and Black, M., "Machine Learning in Digital Games: a Survey", Artificial Intelligence Review, Vol. 29, No. 2, pp. 123-161, 2008.
DOI: https://doi.org/10.1007/s10462-009-9112-y

[5] Ghory, I., "Reinforcement Learning in Board Games", Department of Computer Science, University of Bristol, pp. 1-57, 2004.

[6] Tom M. Mitchell, "Machine Learning", McGraw-Hill Education, pp. 1-414, 1997.

[7] Deokhyung Kim, and Hyunjun Jung, "Performance Analysis of Target Tracking AI based on Unity ML-Agents", The Journal of Korean Institute of Information Technology, Vol. 19, No. 12, pp. 19-26, 2021.
DOI: https://doi.org/10.14801/jkiit.2021.19.12.19

[8] Otterlo, M. V., and Wiering, M., "Reinforcement Learning and Markov Decision Processes", Springer, pp. 3-42, 2012.
DOI: https://doi.org/10.1007/978-3-642-27645-3_1

[9] Unity-Technologies, "Unity ML-Agents Toolkit", https://github.com/Unity-Technologies/ml-agents

[10] Juliani, A., et al., "Unity: A General Platform for Intelligent Agents", Unity Technologies, pp. 1-28, 2020.

[11] Juliani, A., "Introducing: Unity Machine Learning Agents Toolkit", https://blog.unity.com/technology/introducing-unity-machine-learning-agents

[12] Unity-Technologies, "Training Configuration File", https://github.com/Unity-Technologies/ml-agents/blob/release_19_branch/docs/Training-Configuration-File.md

저 자 소 개

이 영 호(정회원)

- 2013년 : 수원대학교 컴퓨터학과 (이학박사)
- 2021년 ~ 현재 : 배재대학교 게임공학과 조교수
- 관심분야 : 인공지능, 게임 디자인, 게임 프로그래밍, VR/AR

※ 이 논문은 2023학년도 배재대학교 교내학술연구비 지원에 의하여 수행됨