Journal of KIIT. Vol. 19, No. 12, pp. 19-26, Dec. 31, 2021. pISSN 1598-8619, eISSN 2093-7571
http://dx.doi.org/10.14801/jkiit.2021.19.12.19

Unity ML-Agents 기반의 목표 추적 인공지능 성능 분석

김덕형*, 정현준**

Performance Analysis of Target Tracking AI based on Unity ML-Agents

Deokhyung Kim*, Hyunjun Jung**

이 논문은 2021학년도 군산대학교 신임교수 연구비 지원에 의하여 연구되었음

요 약

게임에서 NPC(Non-Player Character)들은 각자의 역할에 맞는 인공지능이 적용되어 플레이어와 상호작용하며 플레이어에게 재미를 느끼게 해주는 중요한 요소이다. 게임 인공지능 구현 방법은 다양하며 최근 강화학습이 각광받고 있다. 강화학습은 에이전트가 게임 환경에서 스스로 학습하여 환경에 맞는 인공지능을 구현하는 방법이다. 강화학습을 게임 인공지능 구현에 사용하면 NPC가 취해야 하는 행동을 코드로 구현해야 하는 번거로움도 없고, 인공지능 구현을 목적으로 만들어졌기 때문에 복잡한 인공지능도 구현할 수 있다. 이 논문에서는 강화학습으로 구현된 목표 추적 인공지능의 성능을 분석하고자 게임 제작 플랫폼인 Unity의 ML-Agents를 통해 강화학습으로 목표 추적 인공지능을 구현하고 추가로 기존에 목표 추적 인공지능 구현 방법인 NavMeshAgent로 구현된 목표 추적 인공지능과 성능을 비교한다.

Abstract

In the game, Non-Player Characters (NPCs) are an important component that allows players to feel fun by interacting with the player by applying Artificial Intelligence (AI) appropriate to their role. There are various ways to implement game AI, and reinforcement learning has recently been in the spotlight. Reinforcement learning is a method in which an agent learns by itself in a game environment and implements AI suitable for the environment. If reinforcement learning is used to implement game AI, there is no need to implement the actions that NPCs should write in code, and since it was created for the purpose of implementing AI, complex AI can also be implemented. In this paper, to analyze the performance of target tracking AI implemented by reinforcement learning, we implement target tracking AI with reinforcement learning through ML-Agents of Unity, a game creation platform, and compare performance with target tracking AI implemented with NavMeshAgent.

Keywords

AI, reinforcement learning, unity, ML-agents, navmeshagent, target tracking AI

* 군산대학교 소프트웨어학과 학부생
- ORCID: http://orcid.org/0000-0001-5172-3899
** 군산대학교 소프트웨어학과 교수 (교신저자)
- ORCID: https://orcid.org/0000-0002-6717-1395

· Received: Aug. 19, 2021, Revised: Sep. 13, 2021, Accepted: Sep. 16, 2021
· Corresponding Author: Hyunjun Jung
Dept. of Software at Gunsan University, 558, Daehak-ro, Gunsan-si, Jeollabuk-do, Republic of Korea
Tel.: +82-63-469-8917, Email: junghj85@kunsan.ac.kr

Ⅰ. 서 론

컴퓨터에게 사람과 같은 지능을 부여하여 문제를 해결하게 하는 인공지능(AI, Artificial Intelligence)은 적용되는 분야가 IT와 로봇뿐만 아니라 게임, 자동차, 농업, 의료 등 다양한 분야로 넓어지고 있다[1].

게임 분야는 인공지능이 활용되는 분야 중 하나로 게임 장르에 따라 쓰이는 인공지능이 달라지고, 같은 게임이라도 게임 내 NPC의 역할에 따라 다른 인공지능이 적용되기 때문에 다양한 인공지능이 요구된다. 게임 개발자들은 인공지능을 구현할 때 코드로 구현된 유한한 상태들이 그래프 형태로 연결되어 조건이 만족되면 상태가 전환되는 유한 상태 기계(FSM, Finite State Machine)[2][3]를 사용한다. 하지만 유한 상태 기계는 개발자가 행동(Action)을 코드로 작성해야하는 번거로움이 있고 복잡한 인공지능을 구현하기 어렵다[4].

인공지능 구현 방법 중 하나인 강화학습은 최근 머신러닝 분야의 딥러닝 기술이 발전하면서 사람과 비슷한 혹은 그 이상의 효율을 보이며 여러 분야에서 각광받고 있다[5][6]. 강화학습은 머신러닝의 한 종류로 학습을 통해 최적의 행동 패턴을 찾아내는 방법이다. 2015년에 구글의 딥마인드 사에서 'Deep Q-Network(DQN)'라는 강화학습 알고리즘을 개발하여 Atari사의 여러 게임 환경에서 에이전트(Agent)를 학습시켜 사람 수준의 플레이가 가능 하다는 것을 보여주었고, 2016년에는 '알파고(AlphaGo)'라는 바둑 인공지능을 개발하여 사람보다 더 뛰어난 실력을 보이며 강화학습의 성능을 증명했다. 그림 1은 강화학습의 아키텍처이다. 에이전트는 학습의 주체이며, 환경(Environment)과 상호작용을 통해 학습한다.

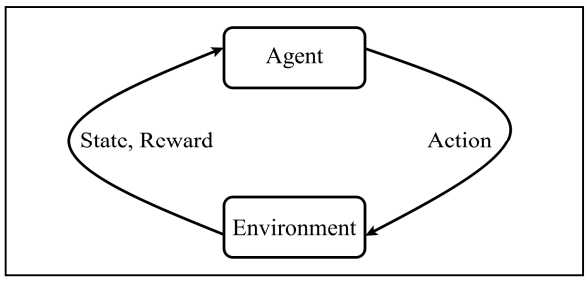

그림 1. 강화학습 아키텍처
Fig. 1. Architecture of reinforce learning

환경은 에이전트에게 상태(State) 정보를 주고 에이전트는 해당 상태에 맞는 행동 정보를 환경에게 전달한다. 환경은 에이전트가 취한 행동에 맞는 보상(Reward)을 에이전트에게 준다. 이런 학습 프로세스를 통해 에이전트는 학습을 진행하게 된다.

이 논문에서는 Unity 플랫폼의 CoinCollect 게임에 강화학습을 사용하여 목표 추적 인공지능을 구현한다. 목표 추적 인공지능이란 현재 자신의 위치로부터 목표로 갈 수 있는 여러 경로 중에서 최적의 경로를 추적하여 이동하는 인공지능을 뜻한다. 목표 추적 인공지능은 'A* 알고리즘'[7]이나 A* 알고리즘 기반의 '네비게이션 메쉬 알고리즘'[8][9]를 사용하여 구현된다. A* 알고리즘과 네비게이션 메쉬 알고리즘은 최적 경로를 찾기 위해 만들어진 알고리즘으로써 목표 추적 등의 최적 경로를 찾아 이동하는 인공지능 구현에 한정되어 사용 가능하다.

Unity는 직관적인 UI와 간단한 빌드로 사용하기 쉬운 플랫폼에 속하고 ML-Agents 툴을 통해 강화학습을 지원한다. 또한 NavMeshAgent를 통해 네비게이션 메쉬 알고리즘을 지원한다. 구현된 인공지능은 Unity로 개발된 게임이라면 어디든 적용시킬 수 있어 재사용도 가능하다는 이점이 있다. 다시 말하자면 이 실험에서 구현한 목표 추적 인공지능을 Unity로 만들어진 다른 게임의 NPC에 적용시킬 수 있다.

이 논문에서 ML-Agents로 구현되는 목표 추적 인공지능은 싱글 에이전트 환경과 멀티 에이전트 환경에서 CoinCollect 게임을 학습한다. 그리고 기존의 목표 추적 인공지능 구현 방법(네비게이션 메쉬 알고리즘)과 성능을 비교하고, 싱글 에이전트 환경과 멀티 에이전트 환경에 따라 수집된 동전의 개수와 학습에 걸린 시간을 비교한다.

이 논문의 2장에서는 FSM, 네비게이션 메쉬 알고리즘, 강화학습으로 구현된 인공지능을 주제로 한 관련 연구들을 소개한다. 3장에서는 실험에 쓰인 게임과 인공지능을 구현하는 과정을 담고 있다. Unity를 이용해 CoinCollect 게임을 구현하고 ML-Agents와 NavMeshAgent를 사용해 목표 추적 인공지능을 구현한다. 4장에서는 두 가지 실험의 과정과 결과에 대한 내용을 담고 있다. 첫 번째 실험에서는 ML-Agents와 NavMeshAgent로 구현된 인공지능이 적용된 게임에서 획득한 점수를 비교함으로써 인공지능

의 성능을 분석한다. 두 번째 실험에서는 ML-Agents 학습 환경에서 에이전트의 개수를 달리하여 인공지능을 구현하고 학습에 걸린 시간과 구현된 인공지능의 성능을 분석한다. 마지막으로 5장에서 결론과 향후 과제를 논한다.

II. 관련 연구

인공지능을 가진 NPC를 활용한 3D 게임 구현에서는 단순한 행동 결정이 아닌 조건부 알고리즘을 통해 플레이어의 행동을 토대로 다음 행동을 예측하여 행동을 결정하는 귀신 인공지능을 구현하였다[2]. 하지만 게임에 직접적인 영향을 주는 NPC의 경우에는 정해진 조건에 따라 행동이 전환되기 때문에 NPC가 취하는 행동 패턴이 단조롭다는 문제가 있다.

고스트들의 협력전술에 의한 팩맨게임에서는 팩맨 게임에서 플레이어를 쫓는 고스트들의 목표 추적 인공지능을 A* 알고리즘 기반의 협력 전술로 구현하였다[7]. 이 논문에서는 기존의 팩맨 게임에서 제안한 인공지능을 적용시켜 인공지능의 수준을 변경하였다. 이를 실험자들에게 적용하여 인공지능의 수준이 게임의 재미와 연관된다는 것을 보여준다.

레이싱 게임에서의 물리엔진과 네비게이션 메쉬에서는 네비게이션 메쉬를 사용해 레이싱 게임의 인공지능을 구현했다[8]. 우선순위기반 충돌 회피를 위하여 네비게이션 메쉬 알고리즘 기반의 충돌 회피 인공지능을 구현했다[9]. 네비게이션 메쉬의 기능을 사용하여 정해진 목적지로 이동하는 인공지능을 구현하여 네비게이션 메쉬 알고리즘이 작동하는 방식을 보여준다.

유한 상태 기계를 이용한 몬스터 인공지능은 게임에서 인공지능 구현에 많이 사용되는 방법인 FSM을 이용하여 몬스터 인공지능을 구현하였다[10]. 취할 수 있는 행동이 많은 몬스터의 인공지능을 FSM만을 사용하여 구현했는데 이로 인해 행동이 전이될 조건까지 인위적으로 정의하기 때문에 행동 사이의 관계를 명확히 정의하지 못한다면 부자연스러운 행동 패턴을 보인다.

O. P. Juhola(2019)는 Unity 플랫폼에서 ML-Agents의 싱글 에이전트와 멀티 에이전트 그리고 NavMeshAgent로 목표 추적 인공지능을 구현하고 성능을 분석하였다[11]. 하지만 이 논문에서는 충분히 학습되지 않은 싱글 에이전트 인공지능과 학습된 멀티 에이전트 인공지능을 비교하는 문제점을 가지고 있다. 즉, 이 논문의 분석환경은 멀티 에이전트 환경의 결과가 유리하도록 설계되어 있다. 이를 해결하기 위해서는 충분히 학습된 싱글 에이전트와의 비교가 필요하다.

이 논문에서는 CoinCollect 게임 환경에서 강화학습의 싱글 에이전트와 멀티 에이전트 그리고 네비게이션 메쉬로 구현된 인공지능의 성능을 비교 평가한다. 그리고 ML-Agents 환경에서 에이전트 개수에 따라 달라지는 학습 시간과 수집한 동전의 수를 비교 평가한다.

III. Unity에서 목표 추적 인공지능 구현

3.1 목표 추적 게임 구현

서론에서 서술한 것처럼 게임 장르, 또는 NPC의 역할에 따라 다른 인공지능이 쓰인다. 이 논문에서 다루는 인공지능은 목표 추적 인공지능이므로 우선 해당 인공지능을 사용할 수 있는 게임 환경이 필요하다. 이 논문에서는 NPC가 동전을 목표삼아 추적하여 수집하는 규칙을 가진 CoinCollect라는 간단한 게임을 구현하여 사용하였다. 게임 제작 플랫폼은 Unity를 사용하였다.

그림 2는 이 논문에서 구현한 목표 추적 게임(CoinCollect)의 기본 화면이자 게임을 구성하는 세 가지 오브젝트를 보여준다.

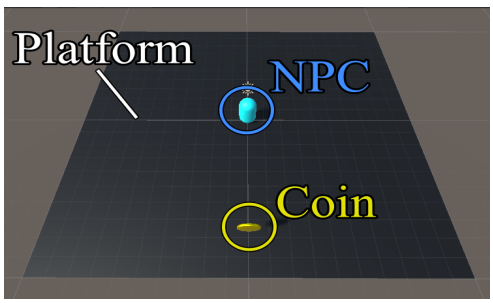

그림 2. CoinCollect 게임 속 세 가지 오브젝트
Fig. 2. Three objects in CoinCollect game

첫 번째로 게임 내에서 바닥 역할을 수행하는 플랫폼 오브젝트, 두 번째로 추적할 목표 역할을 하는 동전 오브젝트, 마지막으로 인공지능이 적용되어 게임의 주체 역할을 할 NPC 오브젝트이다.

우선 게임이 시작되면 동전은 바닥위의 랜덤한 위치에 생성된다. NPC는 동전의 위치를 파악하고 그 위치로 이동하여 동전을 수집한다. 동전을 수집하면 NPC는 1점을 얻고 다시 랜덤한 위치에 새로운 동전이 생성 된다. 만약 바닥 밖으로 떨어질 경우 NPC는 3점을 잃고 초기 게임 시작 위치로 이동된다. NPC는 제한시간 내에 바닥에서 떨어지지 않고 동전을 수집하여 최대한 높은 점수를 얻어야 한다[11]. 실험에서는 NPC 오브젝트에 목표 추적 인공지능을 적용하여 원래라면 사람이 해야 할 조작을 컴퓨터가 스스로 하게하였다. 우리는 목표 추적 인공지능이 적용된 NPC가 진행한 게임에서 획득한 점수가 높을수록 해당 인공지능의 성능이 좋다고 정의한다.

3.2 인공지능 구현

Unity에선 인공지능을 구현하는 방법으로 FSM, NavMeshAgent, ML-Agents를 모두 지원한다. 이 논문에서는 강화학습으로 인공지능을 구현하는 방법인 ML-Agents와 네비게이션 메쉬 알고리즘으로 인공지능을 구현하는 방법인 NavMeshAgent를 사용하여 인공지능을 구현하였다. ML-Agents는 Unity에서 강화학습을 사용하여 인공지능을 구현할 수 있도록 제공해주는 오픈소스 툴이다. 또한 강화학습을 보다 효율적으로 진행할 수 있도록 학습 가속화 및 안정성 향상, 학습에 사용되는 컴퓨터 자원 감소 등의 기능을 지원해준다[12][13].

ML-Agents를 사용하여 인공지능을 구현하려면 우선 강화학습을 진행 할 수 있는 학습 환경이 필요하다. 이 논문에서는 CoinCollect 게임을 이용하여 목표 추적 인공지능을 구현하기 위한 싱글 에이전트 학습 환경을 구축하였다. 그림 3은 CoinCollect 게임의 학습 아키텍처 이다. 에이전트는 NPC 오브젝트이며 ML-Agents를 사용하여 에이전트를 학습시켰다. 에이전트가 취할 수 있는 행동은 '방향 설정(Direction)'과 '이동(Move)'이다.

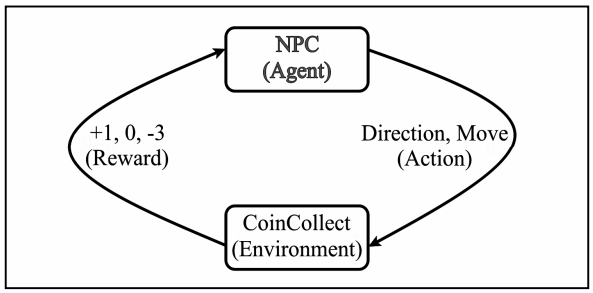

그림 3. CoinCollect 게임의 학습 아키텍처
Fig. 3. Learning architecture of CoinCollect game

학습이 시작하고 행동 패턴을 파악하기 전인 초기의 에이전트는 랜덤한 방향으로 이동한다. 아무런 사건(Event)이 일어나지 않으면 보상을 얻지 못해 0으로 유지된다. 우연히 동전을 획득하면 보상을 1점 얻고, 플랫폼 밖으로 떨어지면 보상을 3점 잃는다. 학습이 계속됨에 따라 에이전트는 보상을 잃거나 얻어가면서 점차 게임 환경에 맞는 최적의 행동 패턴을 찾아가며 학습하게 된다.

이 논문의 환경에서 사용된 파라미터는 표 1과 같으며 학습에 드는 비용은 학습에 걸리는 시간이다. 즉, 시간이 적게 걸릴수록 비용 측면에서 유리하다. 위에 서술한 학습 프로세스를 통해 싱글 에이전트 환경에서 에이전트의 학습에 소요된 시간은 약 15분 30초다. 그림 4는 싱글 에이전트 환경의 평균 보상 그래프다. 한 스텝 당 최저 보상은 플랫폼 밖으로 떨어진 경우인 -3이며, 최대 보상은 동전을 수집한 경우인 1이다. 우리는 평균 보상 그래프가 최대 평균 보상인 1에 도달하면 학습이 정상적으로 이루어졌고 인공지능으로써 작동할 수 있다고 정의한다.

표 1. ML-Agents 학습에 사용된 파라미터 값
Table 1. Parameter values used to ML-Agents learning

Parameter	Value
Agent speed	25
Reward for collecting coins	+1
Reward for falling off the floor	-3
Total learning steps	130,000
Max steps per episode	10000
Batch size	10
Buffer size	100
Algorithm	PPO[14]

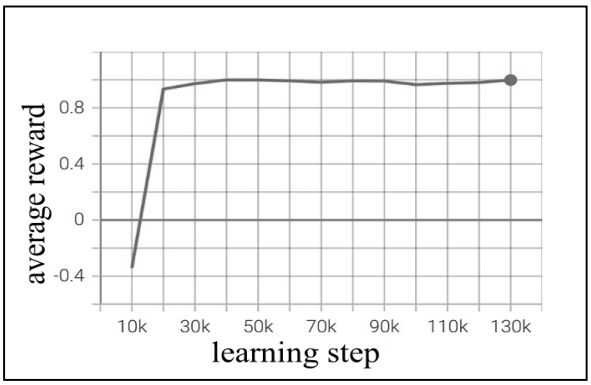

그림 4. 싱글 에이전트 학습의 평균 보상 그래프
Fig. 4. Average reward graph of single agent learning

싱글 에이전트 학습의 평균 보상 그래프는 학습이 시작되고 점점 1에 가까워지다가 약 400,000번째 학습 스텝에서 1에 도달한다. 400,000번째 이후의 학습 스텝에서도 1을 유지하였고 이로써 학습이 정상적으로 이루어졌음을 알 수 있다.

싱글 에이전트 환경과 멀티 에이전트 환경 사이의 학습 효율을 비교하기 위해 멀티 에이전트 환경에서도 인공지능을 구현했다. 그림 5는 15개의 독립된 학습 환경을 가진 멀티 에이전트 환경의 모습이다. 싱글 에이전트 환경을 프리팹(Prefab)으로 만들어 복사하는 방식으로 구현하였다. 멀티 에이전트 환경에서도 싱글 에이전트 환경과 동일하게 표 1의 파라미터 값을 사용하였다. 학습에 소요된 시간은 싱글 에이전트 환경에서 보다 3배가량 적어진 약 4분 20초이다.

그림 6은 멀티 에이전트 환경의 평균 보상 그래프다. 학습이 시작되고 1에 가까워졌고 싱글 에이전트 환경보다 빠른 약 300,000번째 학습 스텝에서 1에 도달하였다. 300,000번째 학습 스텝 이후에도 1을 유지했고 멀티 에이전트 환경에서의 학습도 정상적으로 이루어졌음을 알 수 있다.

ML-Agents와의 차이를 살펴보기 위해 Unity에서 제공하는 목표 추적 인공지능을 구현하는 다른 방법인 NavMeshAgent로 인공지능을 구현하였다.

NavMeshAgent는 목표와의 최적 경로를 연산해주는 알고리즘인 네비게이션 메쉬 알고리즘[8][9]를 사용하여 경로 탐색 또는 목표 추적 인공지능을 제작할 수 있게 해주는 Unity의 기본 컴포넌트이다. Unity 엔진에 기본 탑재되어있어 추가 설치가 필요 없다.

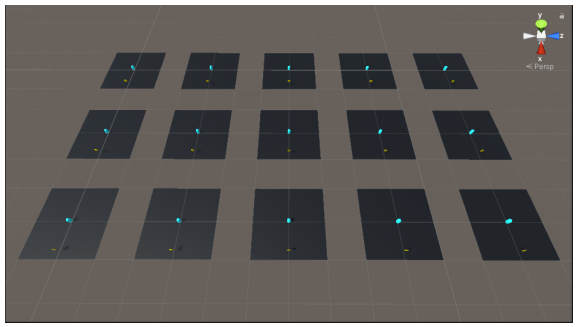

그림 5. 멀티 에이전트 환경
Fig. 5. Multi agent environment

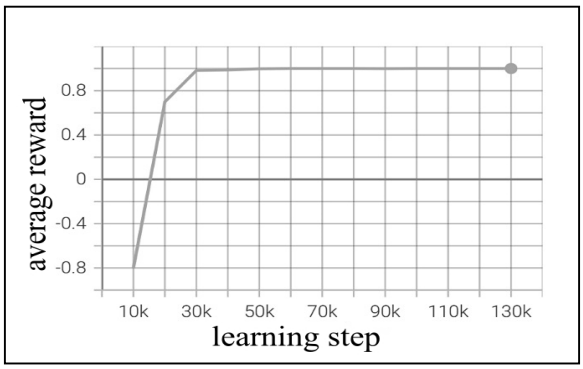

그림 6. 멀티 에이전트 학습의 평균 보상 그래프
Fig. 6. Average reward graph of multi agent learning

NavMeshAgent는 ML-Agents와는 달리 머신러닝 기법이 아니며 단순히 목표 오브젝트와 장애물 오브젝트 그리고 움직일 수 있는 범위를 설정해주면 최적 경로를 찾아 이동한다. NavMeshAgent는 오브젝트가 움직일 수 있는 범위를 지정하기 때문에 지정된 범위 밖으로 이동할 수 없다. 즉, 바닥 밖으로 떨어지지 않는다. 이 실험의 게임 환경에서는 장애물 오브젝트가 없기 때문에 움직일 수 있는 범위와 목표만 설정했다. 움직일 수 있는 범위는 플랫폼 오브젝트, 목표는 동전 오브젝트다. ML-Agents로 만들어진 인공지능과 성능을 비교하기 위해 ML-Agents와 이동 속도를 동일하게 설정했다. 이제 해당 인공지능이 적용된 NPC 오브젝트는 플랫폼 내에서 동전과의 최적 경로를 찾아 스스로 이동한다.

Ⅳ. 실험 및 평가

이 논문에서 사용된 실험 환경은 표 2와 같다. 실험에서 사용된 Unity 2020.3.12.f1 버전은 NavMeshAgent의 성능과 안정성이 향상되었다.

표 2. 실험 환경
Table 2. Experiment environment

Component	Specification
OS	Windows 10 Pro
CPU	Intel(R) Core(TM) i7-10700 CPU @ 2.90GHz
RAM	16GB
Unity version	2020.3.12f1(LTS)
ML-Agents version	2.1.0 exp.1 (June 09, 2021)

ML-Agents 2.1.0 exp.1 버전은 2021년 9월 기준 최신 버전이며 강화학습 알고리즘으로 PPO, SAC, POCA를 지원한다.

4.1 Unity ML-Agents를 이용하여 구현한 목표 추적 AI 효율 분석

동일한 환경에서 플레이어 오브젝트에 적용되는 인공지능만 달리하여 실험을 진행하였다. 한 게임당 2분씩 진행하였으며 한 번의 게임 플레이로는 보편적인 결과를 얻기 힘들기 때문에 각 10회씩 게임 플레이를 진행하였다. 표 3은 세 가지의 인공지능이 적용된 플레이어가 게임 플레이 동안 수집한 동전의 수다.

세 가지의 인공지능 중 가장 높은 성능을 보이는 것은 ML-Agents의 멀티 에이전트 환경에서 학습을 통해 구현된 인공지능이다.

표 3. 2분간의 게임 플레이로 수집한 동전의 수
Table 3. Number of collected coins during gameplay of two minutes

AI / Number	NavMeshAgent (traditional)	ML-Agents single agent	ML-Agents multi agent
1	52	143	155
2	50	147	158
3	48	146	166
4	49	153	165
5	44	146	156
6	39	154	160
7	51	145	163
8	54	148	160
9	46	149	179
10	53	140	174
Average	48.6	147.1	163.6

싱글 에이전트 환경에서 구현된 인공지능보다 학습에 걸린 시간도 약 10분 이상 적게 소요되었고, 약 12% 높은 성능을 보였다. NavMeshAgent로 만들어진 인공지능보다는 무려 약 336% 높은 효율을 보였다.

두 번째로 높은 성능을 보이는 것은 ML-Agents의 싱글 에이전트 환경에서 학습을 통해 구현된 인공지능이다. NavMeshAgent로 구현된 인공지능보다 약 300% 높은 성능을 보였다.

NavMeshAgent 인공지능은 학습 시간이 불필요하다는 이점이 있지만 ML-Agents 인공지능에 비해 성능은 3배 이상 낮았다.

이 실험으로 목표 추적 인공지능의 성능이 ML-Agents 멀티 에이전트, ML-Agents 싱글 에이전트, NavMeshAgent 순으로 높은 것을 확인했다.

NavMeshAgent는 개발자가 신경 써야 할 파라미터의 수가 적고, 학습하는 시간이 불필요하다는 점에서 ML-Agents 보다 빠르고 간단하게 인공지능을 구현할 수 있지만 성능은 ML-Agents보다 떨어진다. 따라서 Unity 엔진에서 목표 추적 인공지능을 구현할 땐 ML-Agents의 멀티 에이전트 환경에서 구현하는 것이 권장되고, NavMeshAgent는 높은 수준의 인공지능이 요구되지 않고 빠르게 인공지능을 구현해야 하는 상황에서 사용하는 것을 권장한다.

4.2 ML-Agents의 에이전트 수에 따른 효율 분석

이 논문의 4.1에서 진행한 실험의 결과로 ML-Agents의 멀티 에이전트 환경에서 만들어진 목표 추적 인공지능이 가장 높은 효율을 보이는 것을 확인할 수 있었다. 두 번째 실험에서는 ML-Agents 환경에서 학습에 사용되는 에이전트의 수를 늘려가면서 학습에 걸리는 시간과 구현된 인공지능의 성능 차이를 비교한다.

실험 환경과 파라미터 값은 각각 표 1, 표 2와 같으며 3.2에서 프리팹으로 만들어진 싱글 에이전트 환경을 복제하여 독립된 학습 환경을 가진 에이전트의 수만 늘려가며 진행하였다. 에이전트의 개수는 1개부터 시작하여 2배씩 늘려가며 512개까지 진행

하였다. 4.1의 실험과 마찬가지로 보편적인 결과를 얻기 위해 2분간의 게임 플레이를 각 10회씩 진행하여 에이전트 개수별 수집한 동전의 수의 평균과 학습에 소요된 시간을 결과로 얻었다. 그림 7은 에이전트 개수별 학습에 걸린 시간과 수집한 동전의 수를 나타낸 그래프이다.

실험 환경과 세부 파라미터 값에 따라 조금씩 다르겠지만, 이 실험에서는 32개의 에이전트 성능이 가장 좋게 나왔다. 학습에 소요된 시간의 감소율은 1개의 에이전트에서 2개의 에이전트 사이가 약 40%로 가장 크고, 이후 에이전트 개수가 2배씩 늘면서 꾸준히 2~30%의 감소율을 보인다. 에이전트의 개수가 늘어날수록 학습에 걸리는 시간은 줄어든다는 것을 확인했다. 학습에 걸리는 시간을 줄이고 싶다면 학습하는 에이전트의 수를 늘리면 된다.

에이전트 개수별 평균 동전 수집량, 즉 목표 추적 인공지능의 성능은 다른 경향을 보였다. 2~64개의 에이전트 환경에서는 싱글 에이전트 환경보다 3~12% 좋은 성능을 보였다. 하지만 64개의 에이전트부터는 에이전트의 수가 늘어날수록 오히려 싱글 에이전트보다 성능이 떨어져 512개의 에이전트를 가진 환경에서는 싱글 에이전트 보다 약 470% 감소된 성능을 보였다.

이 실험을 통해 싱글 에이전트 보다 멀티 에이전트 환경에서 만들어진 인공지능이 비교적 성능이 좋지만, 무분별하게 에이전트 수를 늘리는 것은 지양해야 하며 적절한 개수설정이 필요한 것을 알 수 있다.

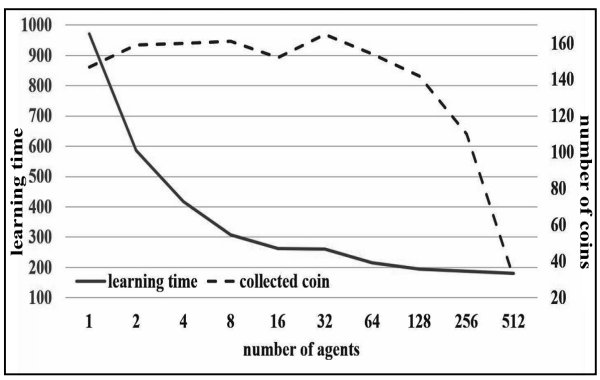

그림 7. 에이전트 개수별 학습 시간과 수집한 동전의 수
Fig. 7. Learning time and number of collected coins by number of agents

V. 결론 및 향후 과제

이 논문은 Unity의 ML-Agents로 구현된 목표 추적 인공지능과 기존의 방법(NavMeshAgent)으로 구현된 인공지능의 성능 차이를 알아보기 위해 첫 번째 실험을 진행하였고, 추가로 ML-Agents의 멀티 에이전트 환경에서 에이전트 개수에 따른 성능과 학습 시간을 알아보기 위해 두 번째 실험을 진행하였다.

첫 번째 실험에서는 Unity에서 구현된 목표 추적 인공지능은 기존의 NavMeshAgent로 구현된 것보다 ML-Agents로 구현된 것이 성능이 높음을 알 수 있었다. 두 번째 실험에서는 ML-Agents를 사용할 때 학습에 걸리는 시간을 줄이고 싶으면 에이전트 개수를 늘리면 되지만 지나치게 많은 에이전트는 오히려 성능에 악영향을 미치는 것을 알 수 있었다. 적용되는 게임이나 설정한 파라미터에 따라 다르므로 ML-Agents에서 에이전트의 개수는 환경에 맞는 유동적인 설정이 필요하다.

추후에는 실험 환경에 GPU를 사용하여 ML-Agents의 성능을 비교할 예정이다.

References

[1] Yerim Choi and Gwanho Kim, "Artificial intelligence overview and application examples", ie Magazine, Vol. 23, No. 2, pp. 23-29, Jun. 2016.

[2] Jo Myoung Kang, Young Hwa Cha, and Byung Joon Park, "Implementation of 3D game using NPC with articial intelligence", 2018 The Institute of Electronics and Information Engineers Conference, Incheon, Korea, pp. 892-893, Nov. 2018.

[3] Myounjae Lee, "An Artificial Intelligence Evaluation on FSM-Based Game NPC", Journal of Korea Game Society, Vol. 14, No. 5, pp. 127-135, Oct. 2014. https://doi.org/10.7583/JKGS.2014.14.5.127.

[4] Song Gyu Jin, "Study of artificial intelligence technology in computer games", KSCI Review,

Vol. 25, No. 1, pp. 11-19, Jun. 2017.

[5] B. H. Yoo, D. D. Ningombam, H. W. Kim, H. J. Song, G. M. Park, and S. Yi, "A Survey on Recent Advances in Multi-Agent Reinforcement Learning", Electronics and Telecommunications Trends, Vol. 35, No. 6, pp. 137-149, Dec. 2020. https://doi.org/10.22648/ETRI.2020.J.350614.

[6] S. Y. Jang, H. J. Yoon, N. S. Park, J. K. Yun, and Y. S. Son, "Research Trends on Deep Reinforcement Learning", Electronics and Telecommunications Trends, Vol. 34, No. 4, Aug. 2019. https://doi.org/10.22648/ETRI.2019.J.340401.

[7] Taeyeong Choi and Hyeon-Suk Na, "Making Levels More Challenging with a Cooperative Strategy of Ghosts in Pac-Man", Journal of Korea Game Society, Vol. 15, No. 5, pp. 89-98, Oct. 2015. http://dx.doi.org/10.7583/JKGS.2015.15.5.89.

[8] Seung-Ho Song and Hye-Young Kim, "Physics Engine with the Navigation Mesh from Racing Game", The 36th Conference of the KIPS, Seoul, Korea, Vol. 18, No. 2, pp. 1159-1161, Nov. 2011.

[9] Seongho Jeon, Hyongil Kim, Juntae Kim, Kyhyun Um, and Hyungje Cho, "Priority-Based Collision Avoidance on a Navigation Mesh", Proceedings of the Korean Information Science Society Conference, Vol. 31, No. 1, pp. 916-918, Apr. 2004.

[10] Jae-Won Jo and Jung-Won Bang, "A study on The Implementation of Monster AI using Finite-State Machine", Proceedings of the Korean Society of Computer Information Conference, Gumi, Korea, Vol. 27, No. 1, pp. 349-350, Jan. 2019.

[11] O. P. Juhola, "Creating Self-Learning AI using Unity Machine Learning", Bachelor's thesis, JAMK University of Applied Sciences, Sep. 2019.

[12] Haechan Park and Nakhoon Baek, "Reinforcement learning through deep learning supported game engine", KIISE Korea Software Congress 2020 (KSC2020), pp. 731-732, Dec. 2020.

[13] Juliani. A, Berges. V, Vckay. E, Gao. Y, Henry. H, Matta., M, and Lange. D, "Unity: A General Platform for Intelligent Agents", arXiv:1809.02627, May 2020.

[14] J. Schulman, F. Wolski P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithm", arXiv:1707.06347, Aug. 2017.

저자소개

김 덕 형 (Deokhyung Kim)

2017년 3월 ~ 현재 : 군산대학교 소프트웨어학과(학사)

관심분야 : 게임, 인공지능, 강화학습

정 현 준 (Hyunjun Jung)

2008년 : 삼육대학교 컴퓨터과학과(학사)

2010년 : 숭실대학교 컴퓨터학과(공학석사)

2017년 : 고려대학교 컴퓨터·전파통신공학과(공학박사)

2017년 8월 ~ 2020년 8월 : 광주과학기술원 블록체인인터넷경제연구센터

2021년 ~ 현재 : 군산대학교 소프트웨어학과 교수

관심분야 : 블록체인, 데이터 사이언스, 센서 네트워크, 사물인터넷, 머신러닝