Journal of Convergence for Information Technology
Vol. 11. No. 3, pp. 1-6, 2021
e-ISSN 2586-4440
DOI : https://doi.org/10.22156/CS4SMB.2021.11.03.001

매치 3 게임 플레이를 위한 PPO 알고리즘을 이용한 강화학습 에이전트의 설계 및 구현

박대근[1], 이완복[2*]
[1]공주대학교 게임디자인학과 석사과정, [2]공주대학교 게임디자인학과 교수

Design and Implementation of Reinforcement Learning Agent Using PPO Algorithim for Match 3 Gameplay

Dae-Geun Park[1], Wan-Bok Lee[2*]
[1]Student, Department of Game Design, Kongju National University
[2]Professor, Department of Game Design, Kongju National University

요 약 매치 3 퍼즐 게임들은 주로 MCTS(Monte Carlo Tree Search) 알고리즘을 사용하여 자동 플레이를 구현하였지만 MCTS의 느린 탐색 속도로 인해 MCTS와 DNN(Deep Neural Network)을 함께 적용하거나 강화학습으로 인공지능을 구현하는 것이 일반적인 경향이다. 본 연구에서는 매치 3 게임 개발에 주로 사용되는 유니티3D 엔진과 유니티 개발사에서 제공해주는 머신러닝 SDK를 이용하여 PPO(Proximal Policy Optimization) 알고리즘을 적용한 강화학습 에이전트를 설계 및 구현하여, 그 성능을 확인해본 결과, 44% 정도 성능이 향상되었음을 확인하였다. 실험 결과 에이전트가 게임 규칙을 배우고 실험이 진행됨에 따라 더 나은 전략적 결정을 도출 해 낼 수 있는 것을 확인할 수 있었으며 보통 사람들보다 퍼즐 게임을 더 잘 수행하는 결과를 확인하였다. 본 연구에서 설계 및 구현한 에이전트가 일반 사람들보다 더 잘 플레이하는 만큼, 기계와 인간 플레이 수준 사이의 간극을 조절하여 게임의 레벨 디지인에 적용된다면 향후 빠른 스테이지 개발에 도움이 될 것으로 기대된다.

주제어 : 게임 인공지능, 자동 플레이, 강화학습, 매치 3 퍼즐, 유니티 ML

Abstract Most of the match-3 puzzle games supports automatic play using the MCTS algorithm. However, implementing reinforcement learning agents is not an easy job because it requires both the knowledge of machine learning and the way of complex interactions within the development environment. This study proposes a method in which we can easily design reinforcement learning agents and implement game play agents by applying PPO(Proximal Policy Optimization) algorithms. And we could identify the performance was increased about 44% than the conventional method. The tools we used are the Unity 3D game engine and Unity ML SDK. The experimental result shows that agents became to learn game rules and make better strategic decisions as experiments go on. On average, the puzzle gameplay agents implemented in this study played puzzle games better than normal people. It is expected that the designed agent could be used to speed up the game level design process.

Key Words : Game AI, Auto Play, Reinforcement Learning, Match 3 Puzzle, Unity ML

*Corresponding Author : Wan-Bok Lee (wblee@kongju.ac.kr)
Received December 29, 2020
Revised February 23, 2021
Accepted March 20, 2021
Published March 28, 2021

1. 서론

애니팡과 같은 매치 3 게임 장르에서 유저들의 인기를 꾸준히 유지하기 위해서는 정교하게 레벨 디자인된 스테이지 구성을 연속적으로 업데이트 하는 것이 중요하다. 스테이지의 빠른 개발이 필요한 만큼 레벨 디자인을 도와줄 자동 플레이 AI 개발이 필요하며,[1] 이때 주로 사용되는 알고리즘이 MCTS(Monte Carlo Tree Search)이다[2].

MCTS는 게임의 상태를 트리로 구성하는 만큼 게임에 새로운 요소가 추가되어도 자동으로 처리할 수 있다는 장점이 있지만 경우의 수가 많을 경우 트리의 크기가 기하급수적으로 커져 탐색 시간이 길어진다는 단점이 있다[3].

이러한 단점 때문에 상대적으로 짧은 학습 시간과 높은 정확도를 가진 강화학습을 통해 자동 플레이 AI를 개발하는 사례가 많아지고 있지만 머신 러닝에 대한 기술이나 지원이 용이하지 않기 때문에 게임 분야에서 적용된 사례가 많이 보고되고 있지는 않은 편이다[4].

본 연구에서는 게임 분야 개발자들이 보편적으로 많이 사용하는 유니티 Machine Learning(ML) SDK를 이용하여 강화학습 에이전트를 손쉽게 개발할 수 있는 방법을 제안하며 게임 플레이 에이전트를 설계 및 구현하고 그 효용성에 대해 알아보려 한다.

개발한 게임 플레이 AI는 “캔디 크러쉬 사가의 99 스테이지(Candy crush saga 99 stage)” 게임에 적용하여 게임플레이를 학습시키고 에이전트의 플레이 성능을 측정하기 위해 데이터를 수집한 후 결과를 분석하였다.

본 논문의 구성은 다음과 같다. 2장에서는 강화학습과 유니티 ML에 대한 기존 연구에 대해 살표보고, 3장에서는 게임 플레이 에이전트의 설계 및 구현에 대하여 제시한다. 4장에서는 게임 플레이 실험 결과를 도출하고, 분석하였으며 5장에서 결론을 맺는다.

2. 배경 연구

2.1 몬테카를로 트리 탐색

몬테카를로 트리 탐색(Monte Carlo Tree Search, MCTS)은 시뮬레이션을 진행하며 트리를 구성하고 게임의 결과를 기반으로 트리의 노드(행동)를 평가하여 가장 가치가 높은 행동만을 선택해나가는 것을 반복하는 알고리즘이다[5].

MCTS는 Fig. 1에서 보이는 바와 같이 매 회 선택, 확장, 시뮬레이션, 역전파 네 단계로 진행된다[6].

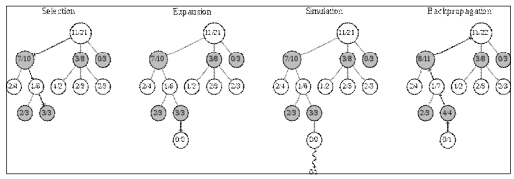


Fig. 1. Four Stages of MCTS

선택 단계에서 어떤 자식 노드를 선택해야할지 결정하기 위해 각 노드의 Upper Confidence Bounds Apply To Tree(UCT) 알고리즘으로 점수를 계산하여 자식 노드를 선택한다[7] .

MCTS 알고리즘은 단순하여 게임에 적용하기 쉽고 트리의 검색이 유망한 하위 트리로 집중되기 때문에 분기 요소가 높은 게임에서 고전 알고리즘보다 우수하며 언제든 탐색을 중단하고 게임에 적용할 수 있다는 장점이 있다 . 하지만 경우의 수가 많아질수록 탐색 과정이 기하급수적으로 길어지고 계산량이 폭증한다는 단점이 있다[8] .

2.2 강화학습 (Reinforcement Learning)

강화학습은 에이전트가 환경을 인식하여 선택 가능한 행동들 중 보상(Reward)을 최대화하는 행동을 선택하는 방법으로 환경과 끝임없이 상호작용하며 정보를 학습하고 보상을 최대로 만드는 행동(Optimal Policy)을 찾는 기법이다[9].

강화학습이 이루어지는 구조는 Fig. 2와 같이 에이전트와 환경 두 부분으로 구성되며, 에이전트는 환경을 관찰하여 정책에 따라 행동을 선택하여 실행한다. 그 후 환경에서 다음 상태와 보상 결과를 받고 보상을 최대화하는 방향으로 정책을 수정한다[10]. 이 과정을 충분한 횟수만큼 반복하여 최적의 정책을 찾아가게 된다.

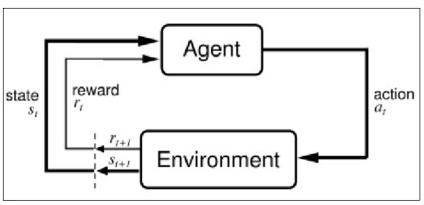


Fig. 2. Structure of Reinforcement Learning

기본적으로 강화 학습에서 환경 모델은 MDP(Markov Decision Process)로 표현된다[11]. MDP는 의사 결정 과정을 확률과 그래프로 모델링한 것으로 다음 상태(t+1)는 이전 상태(t)에만 영향을 받는다는 조건에서 구성된다[12].

Q-Learning은 MDP의 최적의 정책을 찾기 위해 사용할 수 있는 모델 없이 학습하는 강화 학습 기법 가운데 하나이다. 주어진 상태(s)와 행동(a)의 기댓값을 예측하는 Q 함수($Q(s,a)$)를 학습함으로써 최적의 정책을 찾아낸다. 에이전트는 각 상태에서 가장 큰 보상을 얻을 수 있는 행동을 취하도록 학습해야하며 이를 위해 Q 함수 학습 과정을 반복해서 수행하는데 시간 t에서 상태(s), 행동(a)를 취하고 새로운 상태 s_{t+1}로 전이하며, 이 때 보상 r_t가 얻어지고 Q 함수가 갱신된다[11].

$$Q(s_t,a_t) \leftarrow (1-\alpha) \cdot Q(s_t,a_t) + \alpha \cdot (r_t + \gamma \cdot \max Q(s_{t+1},a)) \quad (1)$$

행동을 취할 때 낮은 확률로 랜덤한 선택을 하는데 탐험을 통해 더 좋은 행동을 찾기 위해서이다. Epsilon Greedy 알고리즘을 통해 수행되며 0~1의 랜덤한 실수를 추출하여 입력한 Threshold보다 낮으면 랜덤하게 액션을 취한다. 이 Threshold는 학습이 진행되면서 점점 낮아지게 되기 때문에 학습이 진행될수록 학습 과정이 안정된다.[12]

3. 설계 및 구현

본 연구에서는 먼저 유니티 엔진을 이용하여 캔디 크러시 사가를 모방하여 매치 3 게임을 구현하였으며 구현한 매치 3 게임에 유니티 ML SDK를 적용하여 매치 3 게임 자체를 에이전트로 구성하였다.

유니티 ML은 유니티사에서 공개한 유니티 엔진 통합형 Machine Learning(ML) SDK이다. ML SDK를 사용하면 유니티 엔진으로 제작한 게임이나 시뮬레이션을 심층 강화 학습(Deep Reinforcement Learning), 진화 전략(Evloutionary Strategies) 등 머신 러닝 방법을 사용하기 쉬운 파이썬 API를 통해 지능형 에이전트로 훈련시킬 수 있는 환경으로 전환할 수 있다.

3.1 Match 3 Game 구현

게임을 진행하는 MatchThreeGame Class에는 Fig. 3에서 보이는 바와 같이 게임 진행을 위한 기본적인 기능인 X*Y Field 생성, Block 선택, Block 연결 확인, 연결된 Block 삭제, Field 정렬, Field 채우기, 특수 Block 생성 기능이 구현되어 있다. 정렬과 채우기는 캔디 크러시 사가를 모방하여 위에서 아래로 정렬 및 채우되 빈 곳이 있으면 빈 곳으로 블록이 흐르도록 만들어졌으며 Match 방식과 특수 Block 생성의 조건도 캔디 크러시 사가의 방식과 동일하도록 구현하였다.

```
class MatchThreeGame:
    void GenerateField()
    void SelectBlock(Position)
    void MatchCheck()
    void SortField()
    void FillField()
    void DestoryMatchBlocks()
    void CreateSpecialBlock(Type)
```

Fig. 3. MatchThreeGame Class

게임의 진행은 Field 생성 후 스왑(Swap) 할 Block 두 개를 선택하여 선택된 두 Block이 이웃한 위치에 있을 경우에 스왑한다. 스왑 후 선택된 두 Block이 특수 Block이라면 조합 효과가 발동될 수 있다. 두 Block이 특수 Block이 아니라면 3개 이상 연결 됐는지 확인하며, 그렇지 않다면 두 Block을 원래대로 다시 스왑하고 Block 선택 부분으로 돌아간다.

3.2 ML 에이전트 구성

학습을 위한 에이전트는 유니티 ML SDK에 있는 Agent Class를 상속받음으로써 구성하였으며 본 논문에서는 Fig. 4에 보이는 바와 같이 게임을 진행하는 MatchThreeGame Class에 상속하여 단일 에이전트로 구성하였으며, 관측 데이터 입력과 행동 정의는 Sami Purmonen이 제안한 방식으로 진행하였다[13].

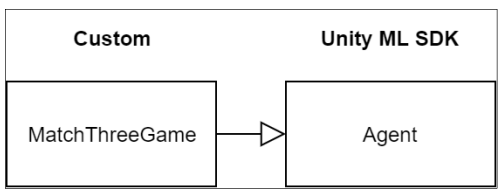


Fig. 4. Agent Structure

행동 선택을 위한 관측 정보로는 [Fig. 5]에서 보이는 바와 같이 Vector Observation 데이터 형태로 현재 Field를 구성하고 있는 Block들의 ID를 1부터 시작하여 ID별로 층을 나누어 넘겨준다. 이때 해당 ID의 블록이 없는 위치에는 0을, Field의 변형을 통해 가로막혀 있는 부분에는 -1을 넣어준다. 폭탄 Block의 Count처럼 관측에 도움이될 정보들도 하나의 층으로 만들어서 넣어준다.

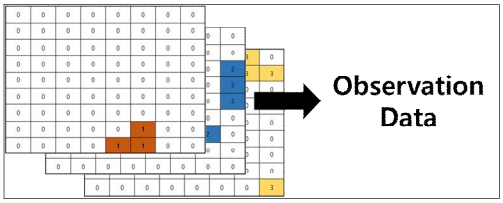


Fig. 5. Observation Data

4. 실험 및 결과

실험은 캔디 크러쉬 사가의 99 스테이지를 모방하여 만든 스테이지를 이용하여 에이전트를 학습시키고 그 결과를 분석하였다. 실험은 Window 10 Home 1909 OS에서 Unity 2019.3.0b4과 ML-Agents Beta 0.11.0을 이용하여 16 게임을 동시에 시뮬레이션하는 과정을 통하여 에이전트를 학습시켰다.

4.1 학습 환경

4.1.1 스테이지 구성

학습에 이용되는 스테이지는 [Fig. 6]에서 보이는 캔디 크러시 사가의 99 스테이지를 모방하여 구성하였다.

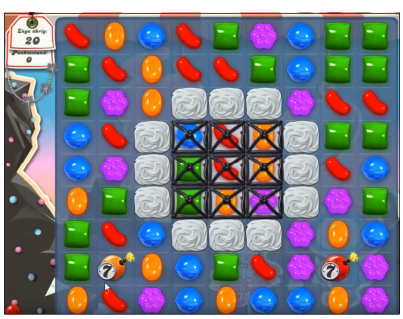


Fig. 6. Candy Crush Saga 99 Stage

99 스테이지는 20턴의 이동 제한을 가진 스테이지로써 가운데 (3*3) 사이즈의 감옥에 갇힌 젤리 블록이 존재하며 7턴의 시간 제한을 가진 폭탄 블록 2개가 꾸준히 유지되기 때문에 폭탄 블록의 관리와 젤리 제거 두 가지 모두를 달성해야하는 난이도가 높은 스테이지에 해당한다. 또한 임의로 채워지는 블록들로 인해 아이템이 없으면 클리어가 불가능할 수도 있다.

4.1.2 Hyperparameter

학습을 위한 Hyperparameter 들은 [Table 1]과 같이 구성하였다. 보상이 자주 이루어지고 블록의 랜덤성이 크기 때문에 빠른 업데이트를 위해 수치들을 낮게 지정하도록 구성하였다.

4.2 비교 데이터 설정

에이전트의 성능 측정을 위한 비교 데이터는 유튜브를 통해 아이템을 사용하지 않고 캔디 크러시 사가 99 스테이지를 클리어한 영상 49개를 수집하여 클리어를 위해 행동한 턴 수를 분석하였으며 그 결과는 [Table 2]의 Expert Group의 수치와 같다.

게이머들은 스테이지 클리어를 위해 최소 10턴이 필요 했고 최대 게임이 제공하는 20턴을 모두 필요로 했으며 평균적으로 15.9턴 플레이 이후에 미션을 완수하였다.

Table 1. Hyperparameter Setting

Hyperparameter	
Name	Value
lambd	0.95
buffer_size	256
batch_size	32
num_epoch	2
learning_rate	1e-3
time_horizon	5
max_steps	1e5
beta	1e-3
Epsilon	0.2
normalize	false
num_layers	1
hidden_layer	128
extrinsic/gamma	0.9

4.3 결과 분석

Tensorboard로 살펴 본 학습의 과정은 [Fig. 7]과 같다. 학습 과정이 불안정한 면이 있는데 이것은 블록

이 랜덤하게 배치되기 때문에 게임의 클리어가 운에 따라 많은 영향을 받을 수 있기 때문이다.

하지만 보상 그래프가 급격히 상승하는 부분의 꼭지점을 이어보면 에이전트가 불안정한 학습 과정 속에서도 더 높은 보상을 향해 활발히 움직이고 있다는 것을 살펴 볼 수 있었다.

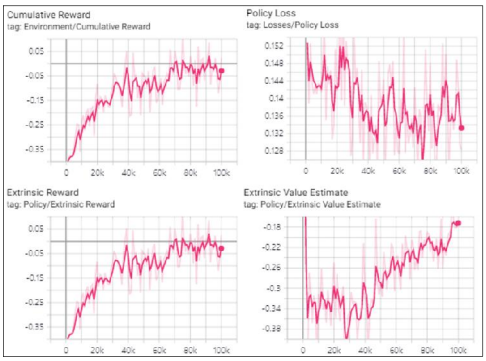


Fig. 7. Tensorboard Graph

게임의 학습 횟수가 누적됨에 따라 클리어되는 횟수는 [Fig. 8]과 같이 나타났다. 학습 과정과 마찬가지로 게임의 학습 횟수가 100단위로 올라갈 때마다 클리어 횟수가 많아지고 있는 것을 확인할 수 있었다.

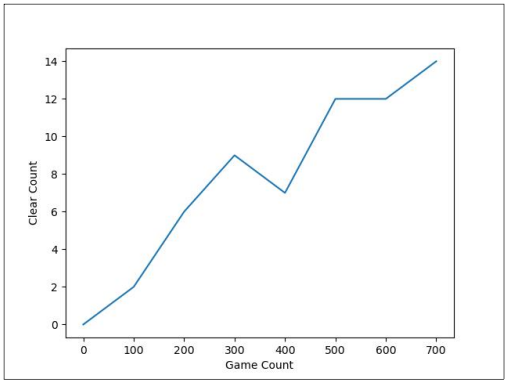


Fig. 8. Number of Clear Games Per Game

스테이지의 클리어 조건인 남은 젤리 수는 게임 학습이 진행됨에 따라 [Fig. 9]와 같이 나타났다. 스테이지에 존재하는 9개의 젤리를 2번씩 지워야 게임이 종료되기 때문에 스테이지 클리어를 위해서는 18개의 젤리를 지워야하는데, 게임의 학습 횟수가 100단위로 올라갈 때마다 게임당 남은 젤리의 수가 적어지는 것을 확인할 수 있었다.

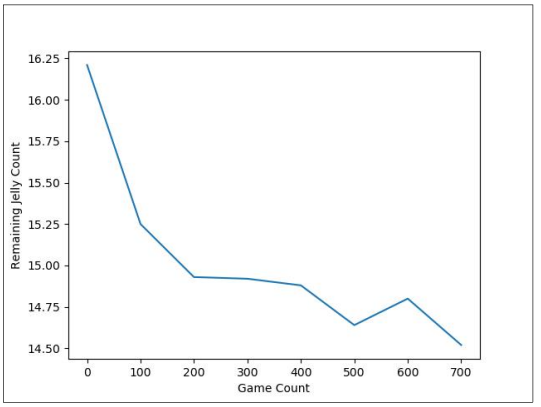


Fig. 9. Number Of Remaining Jelly Per Game

게임 횟수별 클리어 횟수 증가와 남은 젤리 수의 감소를 통해 에이전트가 게임의 클리어 방법을 올바르게 학습하고 있다고 판단할 수 있었다.

학습된 에이전트의 게임 플레이 성능을 평가하기 위해, 비교 데이터와 같은 조건이 되도록 49번의 게임 클리어를 진행하면서 소요된 턴 수를 비교 분석하면 Table 2와 같다.

에이전트는 스테이지 클리어를 위해 비교 데이터보다 15% 적은 최소 7턴이 필요하였으며, 최대 필요 턴은 비교 데이터보다 10% 적은 18턴이 필요 했었다. 평균적으로 11턴 이후에 게임을 종료하기 때문에 비교 데이터보다 평균 44%적은 턴 수를 필요로 하였다.

실험 데이터들을 통해 에이전트가 게임의 규칙을 학습한 이후, 매우 전략적 판단을 하고 있다는 것을 알 수 있었으며 본 연구에서 구현한 퍼즐 게임 플레이 에이전트가 평균적으로 사람보다 퍼즐 게임을 더 잘 플레이한다는 것을 알 수 있었다.

Table 2. Required Turn Comparison

	Expert group	Agent
total number of clear games	49	49
minimum required turn	10	7
maximum required turn	20	18
average required turn	15.9	11.0

5. 결론

이에 본 연구는 매치 3 게임 개발에 주로 사용되는 유니티3D 엔진과 유니티 개발사에서 제공해주는 머신러닝 SDK를 이용하여 강화학습 에이전트를 손쉽게 개

발하는 방법을 제안하며 게임 플레이 에이전트를 설계 및 구현하였다. 실험을 위해 유명한 매치 3 게임 중 하나인 캔디 크러시 사가의 99 스테이지를 모방하여 스테이지를 만들었으며 유튜브를 통해 아이템을 사용하지 않고 99 스테이지를 클리어한 영상 49개를 수집하여 에이전트의 게임 플레이 성능을 비교 분석하였다.

실험 결과 학습된 에이전트가 게임의 규칙을 학습한 이후 매우 전략적 판단을 하고 있다는 것을 알 수 있었으며 본 연구에서 구현한 퍼즐 게임 플레이 에이전트가 평균적으로 사람보다 44% 정도 짧은 시간에 게임을 마칠 정도로 게임을 더 잘 플레이한다는 것을 알 수 있었다.

본 연구에서 설계 및 구현한 에이전트는 사람보다 높은 수준의 플레이 수준을 보여주기 때문에, 기존의 MCTS알고리즘을 적용한 플레이에 비해 머신러닝 SDK를 이용한 PPO알고리즘을 적용할 경우 성능이 상당히 향상되기 때문에, 향후 게임 레벨 디자인에 효과적으로 적용될 수 있을 것으로 기대된다.

REFERENCES

[1] E. Poromaa . (2017 . *Crushing Candy Crush : Predicting Human Success Rate in a Mobile Game using Monte-Carlo Tree Search* . Student thesis . KTH .

[2] R. Coulom. (2006). *Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search* . 5th International Conference on Computer and Games . May 29-31 .

[3] D. Silver. (2016). *Mastering the game of Go with deep neural networks and tree search*. Nature. 529(7587). 484-489.

[4] A. Andelkovic. (2018). *Using Artificial Intelligence to Test the Candy Crush Saga Game. Alexander Andelkovic*. comaqa. (Online). https://www.youtube.com/watch?v=4xECMpgeOxE/

[5] D. Silver. (2017). Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. *Science. 362,* 1140-1144.

[6] L. Kocsis & C. Szepesvári. (2006). Bandit based monte-carlo planning. *In European conference on machine learning* (pp. 282-293). Springer, Berlin, Heidelberg.

[7] S. Gelly, Y. Wang, R. Munos & O. Teytaud. (2006). *Modification of UCT with Patterns in Monte-Carlo Go*. Computer Science.

[8] Aiandgames. (2018). *Monte-Carlo Tree Search in TOTAL WAR: ROME II's Campaign AI*. aiandgames. (Online). https://aiandgames.com/revolutionary-warfare-the-ai-of-total-war-part-3/

[9] M. V. Otterlo & M. A. Wiering. (2012). Reinforcement learning and markov decision processes. *In Reinforcement learning* (pp. 3-42). Springer, Berlin, Heidelberg. DOI : 10.1007/978-3-642-27645-3_1

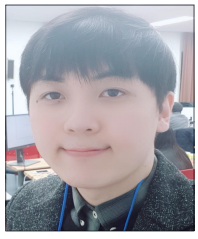
[10] F. S. Melo, (2007). *Convergence of Q-learning: a simple proof*. Proceedings of the European Control Conference 2007. 2-5.

[11] R. Bellman. (1957). A Markovian Decision Process. *Journal of Mathematics and Mechanics. 6(5).* 679-684.

[12] M. Tokic & G. Palm. (2011). Value-Difference Based Exploration: Adaptive Control Between Epsilon-Greedy and Softmax. Advances in Artificial Intelligence, *Lecture Notes in Computer Science. 7006.* 335-346 DOI : 10.1007/978-3-642-24455-1_33

[13] S. Purmonen, (2017). *Predicting Game Level Difficulty Using Deep Neural Networks*. Student thesis of KTH.

박 대 근(Dae Geun Park) [학생회원]



· 2018년 2월 : 호서대학교 게임학과(공학사)
· 2020년 2월 : 공주대학교 게임디자인 석사
· 관심분야 : VR, AR
· E-Mail : uemonwe@gmail.com

이 완 복(Wan Bok Lee) [정회원]

· 2004년 2월 : KAIST 전자전산학과 (전기 및 전자공학 전공 공학박사)
· 현재 : 공주대 게임디자인학과 교수
· 관심분야 : 게임엔진, 시뮬레이션, 이산사건시스템
· E-Mail : wblee@kongju.ac.kr