IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 20, 2020, accepted March 2, 2020, date of publication March 12, 2020, date of current version March 24, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2980380

Playtesting in Match 3 Game Using Strategic Plays via Reinforcement Learning

YUCHUL SHIN, JAEWON KIM, (Student Member, IEEE), KYOHOON JIN, (Student Member, IEEE), AND YOUNG BIN KIM, (Member, IEEE)

Department of Image Science and Art, Chung-Ang University, Dongjak 06974, South Korea

Corresponding author: Young Bin Kim (ybkim85@cau.ac.kr)

This work was supported in part by the National Research Foundation of Korea (NRF) funded by the Korea Government (MSIT) under Grant NRF-2018R1C1B5046461, and in part by the Seoul R&BD Program under Grant CY190039.

ABSTRACT Playtesting is a lifecycle phase in game development wherein the completeness and smooth progress of planned content are verified before release of a new game. Although studies on playtesting in Match 3 games have attempted to utilize Monte Carlo tree search (MCTS) and convolutional neural networks (CNNs), the applicability of these methods are limited because the associated training is time-consuming and data collection is difficult. To address this problem, game playtesting was performed via learning based on strategic play in Match 3 games. Five strategic plays were defined in the Match 3 game under consideration and game playtesting was performed for each situation via reinforcement learning. The proposed agent performed within a 5% margin of human performance on the most complex mission in the experiment. We demonstrate that it is possible for the level designer to measure the difficulty of the level via playtesting various missions. This study also provides level testing standards for several types of missions in Match 3 games.

INDEX TERMS Actor-critic, agent, artificial intelligence, game mission, game strategy, match 3, playtesting, reinforcement learning.

I. INTRODUCTION

Puzzle games constitute a genre of games that have been widely popular for a while owing to their simplicity. Among them, Match 3 games are still being released by developers worldwide and are frequently found to top the lists of free and paid games on Google Play Store or Apple App Store [1].

In Match 3 games, the game screen essentially comprises blocks and obstacles, and the players are required to solve missions assigned to each level before exhausting a given number of moves, by moving objects such as blocks and items. This goal differs from those of other games, such as the survival of a player or defeating opponents, as in Go or Starcraft[1]; in Match 3 games, a player is required to complete the given mission by simply matching three or more blocks constituting the level to make them disappear. The blocks to be moved in order to successfully complete the mission while satisfying the constraint are often chosen not on the basis of simple rules that meet the requirement but according to an optimal strategy befitting the situation.

The biggest concern for Match 3 game designer is to design 'levels' to keep users continually interested in the game. The task of building levels is typically assigned to level designers among the team of game designers. Measuring the difficulty of each level, error checking via playtesting, and establishing overall level balance are often extremely time-consuming, and therefore, successful level balancing via consistent playtesting is a very difficult task regardless of the designer's skill or physical condition.

Besides creating new game levels, the job of a level designer also involves estimating the intended difficulty of a level by personally playing that level and considering the placement of building blocks or tools. The development of a level is time-consuming because of these processes, and the subsequent modifications of level-related parameters based on heuristic experiences often produce unexpected results.

Numerous methods have been developed to address the issues related to level-designing via automation, including traditional ones such as heuristic methods [2], in which

The associate editor coordinating the review of this manuscript and approving it for publication was Utku Kose.

[1] www.starcraft.com

relevant decisions are made by mimicking the thought processes of human beings, finite-state machines (FSMs) [3], [4], Monte Carlo tree search (MCTS) [2], [5], and rule-based [6] methods as well as recent methods, such as those that constitute agents using a convolutional neural network (CNN) [7] or reinforcement learning [8], [9].

However, as these methods are developed according to specific sets of game rules, they need to be modified when the rules are changed, and the associated cost may be a hindrance to complete automation. A recent method using CNN utilizes actual game play data in this context. This approach is used only when a server is built from a real game service and the gameplay data are continually collected, which is inaccessible to most developers owing to staffing and financial constraints. The purpose of this study is, therefore, to present a method capable of checking the planned difficulty of each level and to confirm it via playtesting, even without collecting actual play data.

Therefore, this paper defines strategies that can be used in Match 3 games. Based on these strategies, a system is developed to perform automatic playtesting via reinforcement learning. In the learned automatic playtesting process, the completion of a mission is attempted by applying strategies that are appropriate to the current board states and selecting the appropriate blocks. This study is expected to contribute to the field of game development by proposing automatic playtesting, which aids determination of adequate level difficulties.

The rest of this paper is organized as follows. In Section 2, we present an overview of the related works. In Section 3, we propose our methods. In Section 4, we explain the experimental results. Finally, we conclude our work and discuss future avenues of research in Section 5.

II. RELATED WORKS

Playtesting is one of the most important steps in the game development process [10]. Game development is performed based on content determined by the game designer. However, it is necessary to check whether developer vision has been satisfied in practice during the playtesting process. Particularly in most Match 3 games, the enjoyment of a user is derived from completing each level, implying that appropriate judgment of whether the level configuration follows the level designer's intentions is very important.

Playtesting consists of two main phases: pretesting by a small number of people, including level designers and a small group of actual human testers. As automatic play is rarely used in the latter process, this study attempts to address the former process of direct testing by experts in the field.

The level designer team is composed of people with abundant experience in the genre. They quickly identify the strengths and weaknesses of each level that constitute a Match 3 game and achieve overall level balancing. This process is, however, time-consuming.

To overcome this problem, automatic playtesting process has been introduced and investigated for a long time. Algorithms such as FSM and MCTS, in particular, have been frequently used in various genres in this context. Recent automation in the process of game playtesting, however, has been attempted using deep learning or reinforcement learning, fueled by the development of artificial intelligence.

Researchers have endeavoured to create an agent capable of playing games in various genres. Artificial intelligence has demonstrated shown the ability to outplay humans by using deep learning and reinforcement learning, such as in Go, chess, shogi [11], [12], FPSs [13], [14], strategic simulation games [15], [16], racing games [17], card games [18], arcade games [19], [20], casual games [21]–[24] and sports games [25].

Throughout the history of games, reinforcement learning has been used to solve various associated problems, and in Match 3 games, algorithms have been developed with the aim of helping level designers with automatic playtesting of levels.

The previously proposed algorithms include methods such as MCTS [26], deep neural network (DNN) [27], CNN [28], [29], and reinforcement learning [30]. Although each method utilizes different algorithms and evaluation criteria, the most remarkable difference between them lies in the respective methodologies they employ to solve the problem.

Some researchers have attempted to reproduce the behaviors of users via user data collected during their gameplay in some levels of real games, while others have tried to identify the best method using random plays established via reinforcement learning.

Both of these approaches have their limitations. MCTS suffers from the weakness of being time-consuming because learning is required for each situation in environments where the positions and orientations of blocks are changed randomly and existing learned content is rendered unusable whenever new blocks or rules are added. The major weakness of methods using CNN is that they necessitate the collection of a large amount of playing data.

In a study using reinforcement learning [30], the missions given in the game could not be solved, and a method to obtain points by simply manipulating blocks according to the basic rules was adopted.

Therefore, we propose strategic plays based on reinforcement learning that focus on missions and special blocks, unlike existing methods, which require a large amount of data or simply breaking blocks.

III. METHODS

The method proposed in this paper involves playtesting using strategic plays, and the difficulty of a level is measured by the number of moves allowed for the level. The number of moves is determined by the level designer on the basis of the blocks and obstacles that constitute the level and is a criterion in determining the difficulty of the level. Regardless of how difficult a level is, it becomes easy if the number of available moves is increased without constraint.

(a) (b) (c) (d) (e)

FIGURE 1. Various strategic plays used in Match 3 games: (a) removing mission-related blocks, (b) eliminating blocks of a specific color, (c) using associated special blocks, (d) creating a special block, and (e) removing matched blocks.

First, this study defines strategic plays that are available to the user. Next, randomly generated boards are learned using advantage actor-critic (A2C) to determine the current state of the board. Playtesting is then performed by selecting blocks offering the biggest rewards from among the blocks that enable strategic action. The numbers of moves used in completing the missions by a player and the proposed Strategy Agent and are compared and the performance of the method is evaluated in Jewels Star Story,[2] a Match 3 game.

A. GAME ENVIRONMENT

Jewels Star Story is a Match 3 game in which a player solves given mission on each level by exchanging blocks in a 2D grid environment. In a 9×9 grid environment, blocks, obstacles, and special blocks are placed. Blocks are removed when the player exchanges adjacent blocks to create a connected area of three or more blocks (or, two or more blocks if a special block is used) of the same color.

The Match 3 game used in this study has more than 500 levels, and each level consists of up to four missions in which various blocks and obstacles are required cleared.

The available missions are summarized as follows.

- Collecting blocks
- Removing obstacles
- Sending item to specific position
- Creating special block

The missions vary depending on level, implying that the Strategy Agent needs to acquire the following pieces of information about the state of the current board:

- Position of blocks by colors
- Positions related to mission
- Positions of movable blocks.

[2]https://play.google.com/store/apps/details?id=com.soulfriends.jewelsstarstory

This study proposes a methodology, based on the aforementioned information, to solve the various constructed missions at each level.

B. DEFINITION OF STRATEGIC PLAY IN MATCH 3 GAME

A variety of strategic plays are required in Match 3 games to complete the missions. This study defines five strategic plays that are commonly used in missions, as presented in Fig. 1 and the reinforcement learning is performed based on the listed plays. The first is to remove blocks related to the mission. For example, if the current mission is to remove a red block, we select a move leading to the removal of a red block. This strategy of quickly removing blocks related to missions is the most basic strategy to complete the level. The second is to interact with the blocks that remove all the blocks of a specific color. This is the strategy to remove blocks of the same color as the matched blocks by using special blocks. This is often used when there are many blocks to destroy in difficult areas of a level. The third is to use associated special blocks that remove blocks by matching two special blocks. The effects include removing a vertical and horizontal line, a diagonal line, three vertical and horizontal lines, three diagonal lines, a vertical, horizontal, and diagonal line, a 3×3 range, and a 9×9 range. These blocks have the effect of removing more blocks. The fourth is to create the special blocks that have been mentioned above. Four or more blocks of the same color can be matched horizontally, vertically, or in T-shaped or L-shaped patterns. This strategy involves elimination of the blocks via the use of the third strategy. The fifth strategy is to remove matched blocks. When none of the aforementioned strategies are available, the basic rule of matching three vertical or horizontal blocks of the same color is used. The effect of using this strategy for a mission increases with the number of obstacles and level difficulty.

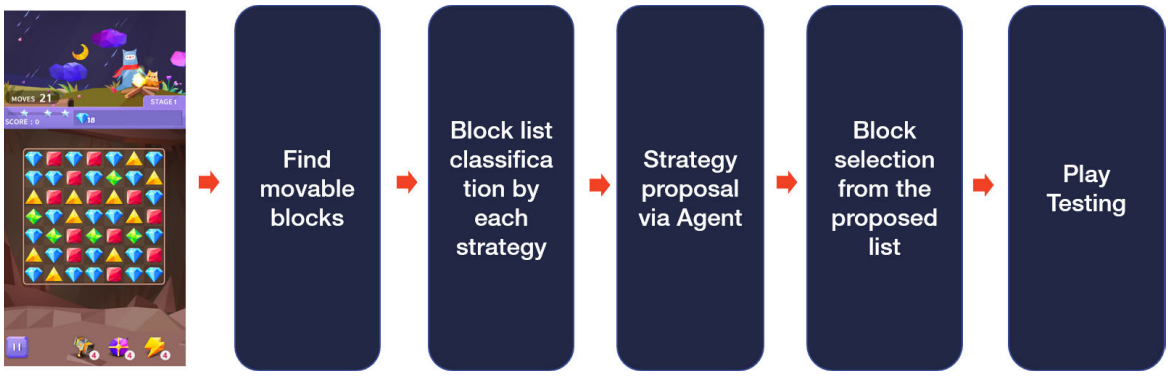


FIGURE 2. Overview of the playtesting process using the Strategy Agent.

C. ASSOCIATION BETWEEN MOVES AND MISSION AND DIFFICULTY ANALYSIS

The level designer assigns an appropriate number of moves to each level based on the corresponding perceived difficulty during the development process. The number of available moves is the maximum number of times players are allowed to act and the objective is to complete the given mission before exhausting the corresponding number of available moves.

Determining the appropriate number of moves for a given difficulty is very important in Match 3 games. The overall level balancing and enjoyment gained by players are optimal only when difficulty levels are set according to planned intentions. In general, the difficulty should increase with the progression of levels alternately increase and decrease beyond a certain difficulty level.

This implies that if level balancing fails at an early stage of the game, the difficulty might be too high for users to gain any enjoyment from the game, leading to immediate abandonment of the game. Particularly in the context of games built for mobile platforms, user retention is very important because it is difficult to induce users to return once they leave.

The appropriate number of moves is generally determined via heuristic methods based on the experience of the user. The factors considered in this process include the particular blocks and obstacles in the level and the placement of items. The initially determined number of moves is adjusted after testing.

To address this problem, this study proposes the appropriate number of moves as a standard for a given level. The proposed standard for the number of moves is calculated by averaging the number of moves required by the Strategy Agent to solve the level over 100 playing test, after excluding the highest and lowest values.

The difficulties are divided into 100 steps and expressed as a number between 0 and 1. The proposed number of moves is set as 0.5. The number of moves suggested by the level designer is compared with the proposed value to calculate the difficulty of the level. The standard of difficulty used in composing several hundreds of levels is proposed to the level designer.

D. A2C AGENT

A2C [31] agent, a policy-based reinforcement learning method, provides probabilistically strategic actions befitting randomly changing states via learning policies suitable to the state of the game board. The environment used in learning is linked to the game environment of Jewels Star Story. Next, the current state of the board is updated whenever the agent selects a block via a network using OpenAI Gym [32] interface and TCP/IP. The probabilistic values of the five strategic plays are determined at this step. The action with the highest probabilistic value is selected. If it does not correspond to the state of the board, the action with the second highest probability is selected. The agent is trained using levels designed separately for training and these levels have the same blocks, obstacles, mission types, and a 9×9 board as those used in the game. The agent is trained to learn the best strategies by checking the current state of board and mission using a policy-based method. This method aims to find and remove blocks that can complete the mission, rather than to simply match blocks.

E. AGENT INPUT AND OUTPUT

The state of board in a Match 3 game consists of blocks of various colors, obstacles, and special blocks on a 9×9 grid. During the training of the A2C Agent, four inputs are used and their contents are as follows:

1) Positions of movable blocks — "1" and "0" represent movable and non-movable blocks, respectively. The blocks or special blocks trapped by obstacles are not movable, and thus represented by "0".
2) Positions of grids related to mission — "1" and "0" represent grids related and unrelated to the mission, respectively. The grids are arranged in a 9×9 shape, and the positions of the grids related to the missions are determined based on the current state of the board.
3) Block types defined on board — "0" represents blocks undefined on the board, and numbers greater than "0" and less than "1" represent separate types of blocks.
4) Colors defined on board — "0" and "1" represent the colors of the blocks and special blocks undefined on the

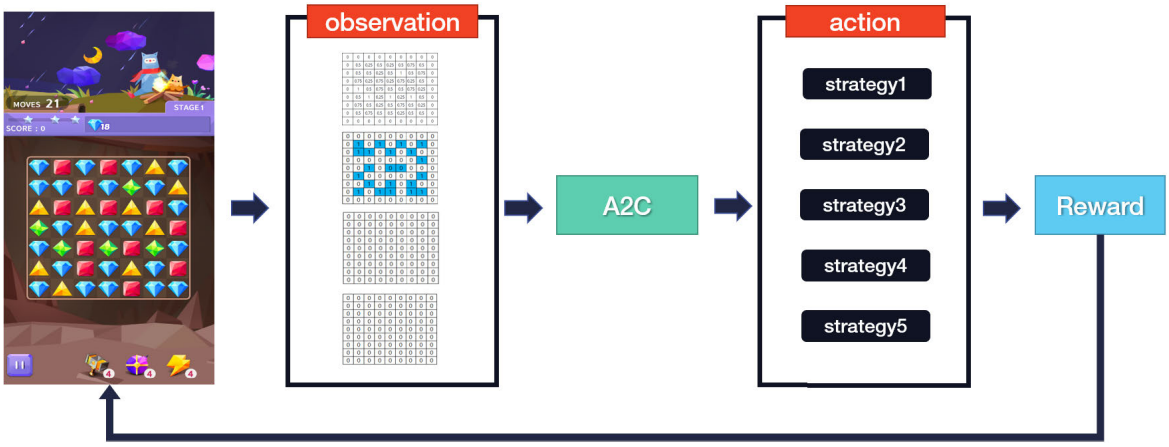


FIGURE 3. The Strategy Agent trains the Agent using the A2C algorithm, and the input returns the probability values corresponding to the five strategies that can be used with the four Feature Maps, and the training is performed via compensation.

board, respectively, and numbers greater than ''0'' and less than ''1'' represent the remaining colors of blocks and special blocks.

All five strategies are expressed as probabilistic values. After selecting a strategy, the rewards of block lists corresponding to the strategy are compared to determine the blocks that are to be exchanged. This allows game playtesting to be performed by evaluating the current state of board to find the best action, and not by simply exchanging blocks corresponding to the best rewards.

F. POLICY NETWORK

The Actor and Critic were composed using deep neural network architecture, and reinforcement learning was applied based on A2C of OpenAI Baseline. A brief outline of its structure has been presented in Fig. 3. The actions are defined as five strategic methods, and the observation of the current level was processed by using four feature maps as states. The rewards assign numbers between −1 denoting failure in completing the mission and 1 for successfully doing so.

IV. EXPERIMENTAL RESULTS

To investigate the performance of strategic play, this study measured the performances of the agents in a basic training level and an actual level in a game service. The levels serviced by Jewels Star Story were evaluated by comparing the average numbers of moves required by human players and the Strategy Agent to complete the level suggested by the level designer. Based on the evaluation, we confirmed how similar performance is to that of humans in playtesting.

We have mainly adopted three agents in our experiment: Strategy Agent, Random Agent, and Rule-Based Agent. The Strategy Agent is an agent from the proposed reinforcement learning algorithm. The Random Agent plays the given level in with purely random selection among the movable blocks. The Rule-Based Agent is an agent from the Jewels Star Story. It chooses the most suitable block under preset conditions (i.e., mission completion progress, and the number of each block type). In comparison with the Strategy Agent, the only similarity is a selection of a block among the pool of movable ones. But, the Rule-Based agent selects a block based on the designer provided conditions and the Strategy Agent is from the A2C policy network.

At the game service level, in addition, the evaluation is based on the average number of moves collected from 10,000 plays of users. Using the actual average number of moves used for completion of the level by users, this method determines the difference in performance between a human player and the Agents.

As the levels used during testing are basically divided into constant components and randomly appearing components upon initialization of the level, there were inherent differences in the environments of some tests. However, this did not affect the conclusions of the study significantly as game play was performed under similar conditions in commercial games.

A. COLLECT COLOR BLOCK & NO OBSTACLES

Removing a certain number of blocks of a specific color constituted the particular mission considered in this case. The mission used during testing was to remove 50 yellow blocks and the speed at which each Agent completed the mission was measured based on the corresponding number of moves required. As evidenced in level 1 of Table 1, the average number of moves required was 46.23 for the Random Agent and 37.59 for the Rule-Based Agent and 30.75 for the Strategy Agent, indicating that the proposed Strategy Agent performed 18.19% better than the Rule-Based Agent.

B. MULTIPLE COLLECT COLOR BLOCKS & NO OBSTACLES

The mission considered in this section comprised removing a certain number of blocks of multiple colors. The mission used during testing was to remove 50 yellow and 50 blue blocks. This mission was used to investigate whether the completion

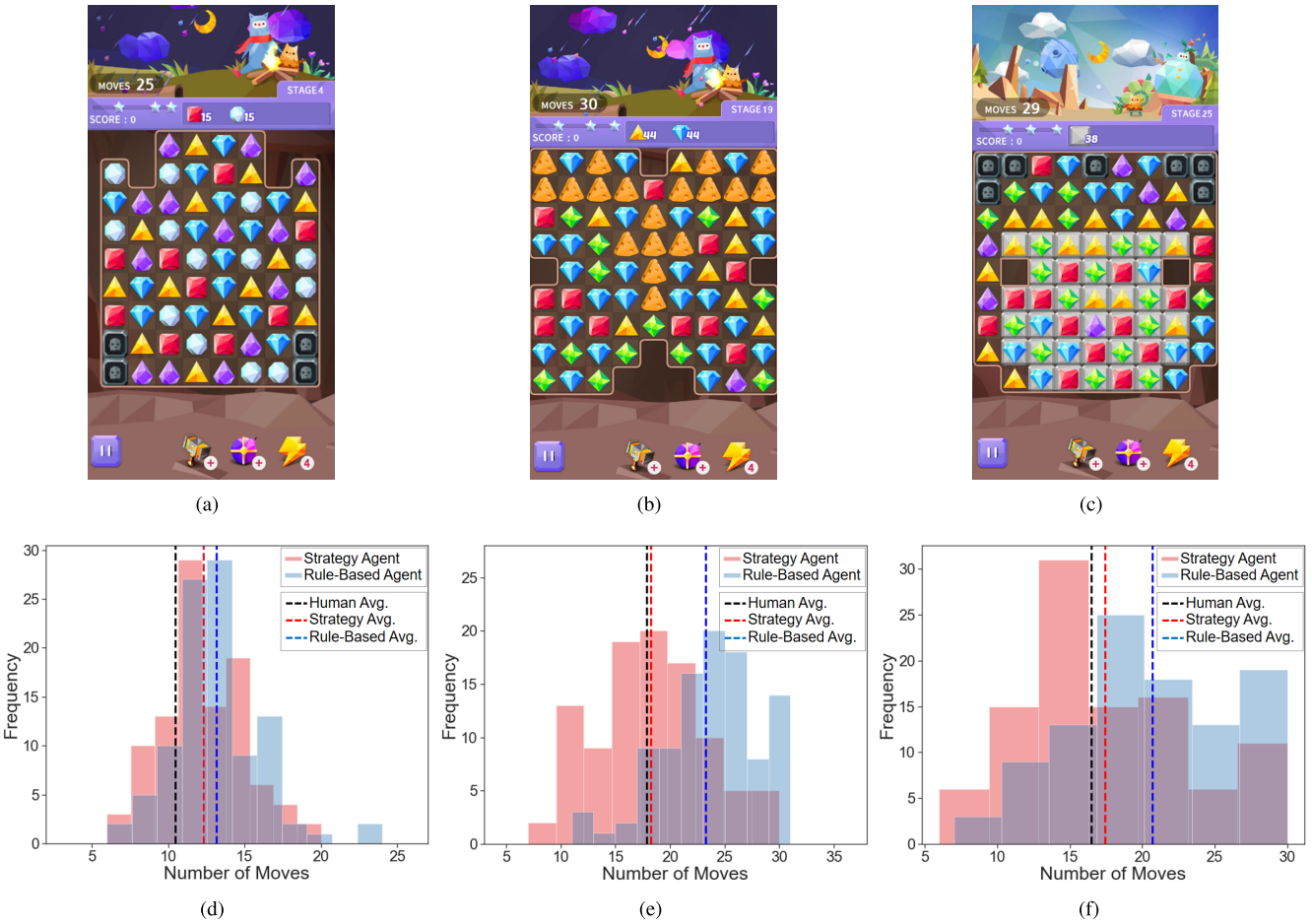


FIGURE 4. Evaluation using three levels (a), (b), and (c) of the test environment in Jewels Star Story and their results (d), (e), and (f), respectively. In (d), (e), and (f), the x-axis is the number of used moves to finish the level and the y-axis is the frequency of each move. The vertical dotted lines in (d), (e), and (f) depict the average moves used by each Agent or human player.

TABLE 1. Performance index in single & multi collection color block missions.

	Level/no. Moves	Mean	Median	Std.
Random Agent	1/100	46.23	46	10.61
Strategy Agent	1/100	30.75	30	6.56
Rule-Based Agent	1/100	37.59	37	8.34
Random Agent	2/100	47.94	49	7.12
Strategy Agent	2/100	38.15	35	7.74
Rule-Based Agent	2/100	42.33	42	7.27

of a mission involving the removal of two kinds of blocks is feasible. As evidenced in level 2 of Table 1, the average number of moves required was 47.94 for the Random Agent, 42.33 for the Rule-Based Agent, and 38.15 for the Strategy Agent, indicating that the proposed Strategy Agent performed 9.87% better than the Rule-Based Agent.

C. MULTIPLE COLLECT COLOR BLOCKS & NO OBSTACLES & COMMERCIAL GAME MISSION

To measure the difference between the performances of the Agents and that of a user, a mission to remove 15 red and 15 white blocks is used. As depicted in Fig. 4d, the average number of moves required was 10.46 for the Human Player, 12.29 for the Strategy Agent, and 13.16 for the Rule-Based Agent, indicating that proposed Strategy Agent performed 6.61% better than the Rule-Based and performed 17.5% worse than the Human Player in this mission.

D. MULTIPLE COLLECT COLOR BLOCKS & OBSTACLES & COMMERCIAL GAME MISSION

A variety of obstructive blocks as well as normal blocks were present in the game. The average number of moves required was 17.84 for the human player, 18.22 for the Strategy Agent, and 23.29 for the Rule-Based Agent, indicating that the Human Player and proposed Strategy Agent performed similarly in this mission, with a difference of under 5% in performance, as shown in Fig. 4e. Whereas, the proposed Agent performed 21.77% better than the Rule-Based Agent.

E. DESTROY SHADE MISSION LEVEL & COMMERCIAL GAME MISSION

A Shade Mission is a panel under normal blocks that is removed when the block above it is removed. The Destroy

Shade Mission is widely used in Match 3 games and may be used together with the Collect Single Mission and this mission is the most complex mission in the experiment. The average number of moves required for such a mission was observed to be 16.48 for the human player, 17.44 for the Strategy Agent, and 20.7 for the Rule-Based Agent, indicating, once again, that the human player and the proposed Strategy Agent performed similarly in this mission, with a difference of under 5% in performance, as depicted in Fig. 4f. And the Strategy Agent performed 15.75% better than the Rule-Based Agent in this mission.

V. CONCLUSION AND FUTURE WORK

In this study, we conducted a playtest by training a Strategy Agent using a Match 3 game comprising of various missions via reinforcement learning. The performance of the proposed Strategy Agent was evaluated based on automated playtests in various environments, and the experiments showed that the trained Strategy Agent outperforms the Rule-Based Agent that is widely used in the industry. Notably, the Strategy Agent performed 15.75% better than the Rule-Based Agent and performed within a 5% margin of human performance on the most complex mission in the experiment. This demonstrates that the agent could be used for playtesting without utilizing actual user play data. The time-consuming nature of training the Agent due to the diversity of game rules, we minimized the training time by using predefined strategic plays. Therefore, this method can serve as a baseline for difficulty evaluation in the absence of extensive testing of levels developed by the level designer.

However, the generalization of the results of this study to all levels with various missions, obstructive blocks, and random change is not recommended. The difficulty of a mission and the layouts of the obstruction blocks planned by the level designer differ from one to another, and they can be different from the form verified in this study.

We expect that this study will be of benefit to various game developers by suggesting methodologies to playtest the types and levels of missions via strategic plays in Match 3 games and by developing standardized agents for level testing for independent game companies.

REFERENCES

[1] M. Samorukov. (2018) *Game Market Analysis by Genre: Why Do This and How to Use It to Your Best Advantage*. [Online]. Available: https://appmagic.rocks/blog/genre-analysis/en

[2] C. Holmgård, M. C. Green, A. Liapis, and J. Togelius, "Automated playtesting with procedural personas through MCTS with evolved heuristics," *IEEE Trans. Games*, vol. 11, no. 4, pp. 352–362, Dec. 2019.

[3] R. Andrea, R. I. Akbar, and M. Fitroni, "Developing battle of Etam earth game agent with finite state machine (FSM) and Sugeno fuzzy," *ICCS Proc.*, vol. 1, no. 1, pp. 184–187, 2014.

[4] A. R. Hakim, R. Andrea, and D. Antoni, "Membangun Edugame 'Baby Zoo Puzzle' berbasis Android dengan game agent implementasi finite state machine," *Sebatik*, vol. 16, no. 1, pp. 9–15, 2016.

[5] O. Keehl and A. M. Smith, "Monster Carlo: An MCTS-based framework for machine playtesting unity games," in *Proc. IEEE Conf. Comput. Intell. Games (CIG)*, Aug. 2018, pp. 1–8.

[6] S. Bojarski and C. B. Congdon, "REALM: A rule-based evolutionary computation agent that learns to play mario," in *Proc. IEEE Conf. Comput. Intell. Games*, Aug. 2010, pp. 83–90.

[7] C. Clark and A. Storkey, "Training deep convolutional neural networks to play go," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1766–1774.

[8] S. Wender and I. Watson, "Using reinforcement learning for city site selection in the turn-based strategy game civilization IV," in *Proc. IEEE Symp. Comput. Intell. Games*, Dec. 2008, pp. 372–377.

[9] P.-A. Andersen, M. Goodwin, and O.-C. Granmo, "Deep RTS: A game environment for deep reinforcement learning in real-time strategy games," in *Proc. IEEE Conf. Comput. Intell. Games (CIG)*, Aug. 2018, pp. 1–8.

[10] R. Ramadan and Y. Widyani, "Game development life cycle guidelines," in *Proc. Int. Conf. Adv. Comput. Sci. Inf. Syst. (ICACSIS)*, Sep. 2013, pp. 95–100.

[11] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[12] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, Oct. 2017.

[13] G. Lample and D. S. Chaplot, "Playing FPS games with deep reinforcement learning," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 1–7.

[14] M. Jaderberg, W. M. Czarnecki, I. Dunning, L. Marris, G. Lever, A. G. Castaneda, C. Beattie, N. C. Rabinowitz, A. S. Morcos, A. Ruderman, N. Sonnerat, T. Green, L. Deason, J. Z. Leibo, D. Silver, D. Hassabis, K. Kavukcuoglu, and T. Graepel, "Human-level performance in first-person multiplayer games with population-based deep reinforcement learning," 2018, *arXiv:1807.01281*. [Online]. Available: http://arxiv.org/abs/1807.01281

[15] O. Vinyals *et al.*, "StarCraft II: A new challenge for reinforcement learning," 2017, *arXiv:1708.04782*. [Online]. Available: http://arxiv.org/abs/1708.04782

[16] T. Rashid, M. Samvelyan, C. Schroeder de Witt, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," 2018, *arXiv:1803.11485*. [Online]. Available: http://arxiv.org/abs/1803.11485

[17] N. Xu, B. Tan, and B. Kong, "Autonomous driving in reality with reinforcement learning and image translation," 2018, *arXiv:1801.05299*. [Online]. Available: http://arxiv.org/abs/1801.05299

[18] J. Heinrich and D. Silver, "Deep reinforcement learning from self-play in imperfect-information games," 2016, *arXiv:1603.01121*. [Online]. Available: http://arxiv.org/abs/1603.01121

[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," 2013, *arXiv:1312.5602*. [Online]. Available: http://arxiv.org/abs/1312.5602

[20] N. Tziortziotis, K. Tziortziotis, and K. Blekas, "Play ms. pac-man using an advanced reinforcement learning agent," in *Proc. Hellenic Conf. Artif. Intell.* Cham, Switzerland: Springer, 2014, pp. 71–83.

[21] P.-W. Wang, P. L. Donti, B. Wilder, and Z. Kolter, "SATNet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver," 2019, *arXiv:1905.12149*. [Online]. Available: http://arxiv.org/abs/1905.12149

[22] F. Agostinelli, S. McAleer, A. Shmakov, and P. Baldi, "Solving the Rubik's cube with deep reinforcement learning and search," *Nature Mach. Intell.*, vol. 1, no. 8, pp. 356–363, Aug. 2019.

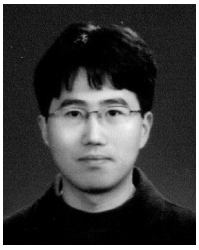
[23] N. Kondo and K. Matsuzaki, "Playing game 2048 with deep convolutional neural networks trained by supervised learning," *J. Inf. Process.*, vol. 27, pp. 340–347, 2019.

[24] K. Narasimhan, T. Kulkarni, and R. Barzilay, "Language understanding for text-based games using deep reinforcement learning," 2015, *arXiv:1506.08941*. [Online]. Available: http://arxiv.org/abs/1506.08941

[25] K. Kurach, A. Raichuk, P. Stańczyk, M. Zając, O. Bachem, L. Espeholt, C. Riquelme, D. Vincent, M. Michalski, O. Bousquet, and S. Gelly, "Google research football: A novel reinforcement learning environment," 2019, *arXiv:1907.11180*. [Online]. Available: http://arxiv.org/abs/1907.11180

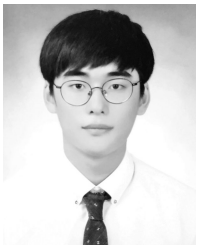
[26] E. R. Poromaa, "Crushing candy crush: Predicting human success rate in a mobile game using Monte-Carlo tree search," M.S. thesis, KTH Royal Inst. Technol., Stockholm, Swedn, 2017.

[27] S. Purmonen, "Predicting game level difculty using deep neural networks," M.S. thesis, M.S. thesis, KTH Royal Inst. Technol., Stockholm, Swedn, 2017.
[28] S. F. Gudmundsson, P. Eisen, E. Poromaa, A. Nodet, S. Purmonen, B. Kozakowski, R. Meurling, and L. Cao, "Human-like playtesting with deep learning," in *Proc. IEEE Conf. Comput. Intell. Games (CIG)*, Aug. 2018, pp. 1–8.
[29] P. Eisen, "Simulating human game play for level difficulty estimation with convolutional neural networks," M.S. thesis, KTH Royal Inst. Technol., Stockholm, Swedn, 2017.
[30] I. Kamaldinov and I. Makarov, "Deep reinforcement learning in Match-3 game," in *Proc. IEEE Conf. Games (CoG)*, Aug. 2019, pp. 1–4.
[31] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.
[32] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI gym," 2016, *arXiv:1606.01540*. [Online]. Available: http://arxiv.org/abs/1606.01540



YUCHUL SHIN received the B.S. degree in computer science from Kookmin University, Seoul, South Korea, in 2002. He is currently pursuing the M.S. degree in image engineering with the Graduate School of Advanced Imaging Science, Multimedia and Film, Chung-Ang University. His research interests include game development and deep learning.

JAEWON KIM (Student Member, IEEE) received the B.S. degree in management information system from Myongji University, Seoul, South Korea, in 2018. He is currently pursuing the M.S. degree in image engineering with the Graduate School of Advanced Imaging Science, Multimedia and Film, Chung-Ang University. His research interests include deep learning and image processing.



KYOHOON JIN (Student Member, IEEE) received the B.S. degree in applied statistics from Gachon University, Gyeonggi, South Korea, in 2019. He is currently pursuing the M.S. degree in image engineering with the Graduate School of Advanced Imaging Science, Multimedia and Film, Chung-Ang University. His research interests include deep learning and natural language processing.

YOUNG BIN KIM (Member, IEEE) received the B.S. and M.S. degrees in computer science and the Ph.D. degree in visual information processing from Korea University, in 2010, 2012, and 2017, respectively. From August 2017 to February 2018, he was a Principal Research Engineer with Linewalks. He is currently an Assistant Professor with the Graduate School of Advanced Imaging Science, Multimedia and Film, Chung-Ang University. His current research interests include data mining and deep learning.