2024 IEEE Smart Information Systems and Technologies (SIST)
15-17 May 2024, Astana, Kazakhstan

2024 IEEE 4th International Conference on Smart Information Systems and Technologies (SIST) | 979-8-3503-7486-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/SIST61555.2024.10629511

Enhancing Gameplay Experience Through Reinforcement Learning in Games.

Dinmukhammed Zhasulanov
Department of Computer Engineering
Astana IT University
Astana, Kazakhstan
211121@astanait.edu.kz
ORCID ID: 0009-0009-9843-4950

Bekarys Marat
Department of Computer Engineering
Astana IT University
Astana, Kazakhstan
211410@astanait.edu.kz
ORCID ID: 0009-0000-2655-8916

Kuanysh Erkin
Department of Computer Engineering
Astana IT University
Astana, Kazakhstan
211185@astanait.edu.kz
ORCID ID: 0009-0001-3073-1985

Ruslan Omirgaliyev, IEEE member
Department of Computer Engineering
Astana IT University
Astana, Kazakhstan
Ruslan.Omirgaliyev@astanait.edu.kz
ORCID ID: 0000-0003-3834-2359

Alman Kushekkaliyev
Department of Physics
West Kazakhstan University
Uralsk, Kazakhstan
Alman_k@mail.ru

Nurkhat Zhakiyev, IEEE member
Department of Science and Innovations
Astana IT University
Astana, Kazakhstan
Nurkhat.Zhakiyev@astanait.edu.kz
ORCID ID: 0000-0002-4904-2047

Abstract: **It is important to improve the game development field since games have great influence on economics and society. One of the ways to enhance game experience is to replace non-playable characters (NPC) by machine learning (ML) agents. Artificial Intelligence (AI) trained by Reinforcement learning algorithms can adapt to the game environment and act based on users' actions or changes in surroundings. This research project aims to test the idea of replacing the NPC by artificial intelligence agent trained by Reinforcement learning algorithms. The data for this project was collected by observation of experiments inside of the game created in Unity platform. Agent was trained in special toolkit for Unity platform called "Unity ML-Agents". The behavior of the agents was subjected to preparatory measures for replication, including preliminary data processing using behavior cloning methods. The selected algorithm of the Reinforcement learning was the Proximal Policy Optimization, which is based on balance between new actions and its rewards. The training progress of the agents was monitored utilizing TensorBoard, the default package within the Unity ML Agents toolkit. During the research initialization phase, there were set null hypothesis, which states that pre-scripted bots are better for games and alternative hypothesis, which states that AI can replace NPC and enhance the game experience for a player.**

Keywords: Game; Gameplay Experience; Reinforcement Learning; Unity; ML Agents; Non-playable character (NPC); Artificial Intelligence (AI);

I. INTRODUCTION

Today, the global media consumption market is changing rapidly, television is being replaced in every possible way by fascinating Internet technologies and computer games. Over the past few decades, the world of games has gone an amazing way from simple 8-bit pixel adventures to exciting ultra-realistic virtual worlds. But computer games have not always been so successful and popular. The gaming industry has changed with the arrival of a personal computer and the ability to do something more. Evolution was driven not only by the development of hardware, but also by breakthroughs in artificial intelligence and machine learning technologies. There are three problems with learning models of AI: Supervised, Unsupervised and Reinforcement learning. Supervised and Unsupervised learning models are based on datasets that are provided by developers, but only the Reinforcement learning gathers data and adapts to it. Because of this approach, among three learning methods, Reinforcement Learning stands out as a promising approach to improving the gameplay by creating intelligent and adaptive gaming environments.

Reinforcement learning (RL) is a powerful approach for training intelligent agents to perform a wide range of tasks, from playing games to navigating complex environments [1]. Reinforcement learning methods determine how agents change their policies because of accumulated action and experience. In our case the agent is AI, that is making the actions and learning. In general, the agent's goal is to maximize the total reward that he receives in the long term. In simple words, it is the same as teaching the dog, cat, or any other pet to do some commands, by rewarding it for correct actions.

Authorized licensed use limited to: Malta College of ArtsScience&Tech. Downloaded on November 24,2024 at 15:22:46 UTC from IEEE Xplore. Restrictions apply.

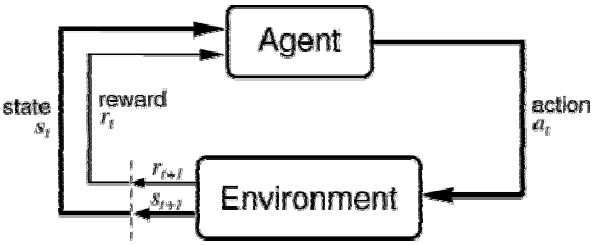

Figure 1. Reinforcement Learning steps [2]

While reinforcement learning and its applications have made great strides in various fields, specific applications for improving gameplay still require close attention. Existing research has demonstrated the feasibility and effectiveness of RL in tasks such as gaming artificial intelligence, NPC (non-player character) behavior, and procedural content generation. A comprehensive study of how to systematically use RL to optimize and personalize gameplay remains an area that has not been explored.

In addition, practical issues such as the integration of RL algorithms into existing game architectures, computational requirements, and accessibility to the broader community of game developers are topics that require in-depth research. Overcoming this knowledge gap will not only contribute to the development of artificial intelligence in games, but also pave the way for a more exciting and personalized gaming experience.

Specifically, the study will aim to answer the following question: How can reinforcement learning algorithms be used for non-playable characters (NPCs) in games and what impact would it have on player engagement?

Now computer games are more than just code fragments, they successfully accumulate expressiveness and ways of transmitting information. This is a set of visual components represented by the most sophisticated methods of computer graphics and visualization of animation effects accompanied by music. Artificial intelligence in games has become one of the main manifestations of computer games. Artificial intelligence can very plausibly imitate human behavior. Ultimately, games with artificial intelligence have unlimited potential and will always surprise us with their new achievements. Thus, based on the relevance of the study, we chose the topic of the study.

II. Literature Review

Our chosen articles, that are like our research paper are articles that based on a concept of "Automated Vehicle" and its simulation in games. For the automation of driving process, researchers used the Machine Learning algorithms that are based on reinforcement learning. Theories approve that this learning method is most effective, because of adaptation to environment. There are some articles, which were written during the experiments and tests on topic "reinforcement learning algorithms in automated vehicles" and in these articles research used almost the same technologies, such as TORCS (The Open Racing Car Simulator) or CARLA [3]. The study by [4] provided us with the information about the usage of deep learning algorithms in automated vehicle system. This article might help us to better understand the future problems of our research. During the analysis of existing research papers with the same topics we've found out, that algorithms such as: Simultaneous localization and mapping (SLAM)[4], asynchronous advantage actor-critic (A3C)[3], Soft Actor-Critic (SAC)[5], Rainbow Double Q Learning (Rainbow DQN)[5], Proximal Policy Optimization (PPO)[6,7], Parallel Online Continuous Arcing (POCA)[6], Behavioral Cloning (BC)[6,7] are already used and tested in the research papers that are based on Automated Vehicle technology.

Traditional game AI and AI with RL were tested in the research [8] and AI with RL is better for game experience, since it is more skilled and reliable than traditional AI. In this research paper we are planning to use behavior cloning method to prepare the agent for accomplishing the given task. In the research paper [9] it is stated that Behavior cloning might cause significant difference in performance of agent. There is a research paper [10], that compares Neuro-evolution and reinforcement learning for automated vehicle with the task of going through the circuit in less time. Even if Neuro-evolution performance was better in case of finding the optimal solution, agent with reinforcement learning model performed better during play mode.

It might be hard to balance the realistic behavior of AI based NPC, so it can lead to certain problems. Realistic AI may easily defeat the player and become too powerful to play against, or controversial it might become too simple and rob the enjoyment of the challenge. The idea of using AI in FPS games was tested in the research paper [11] and as its conclusion authors stated that it is hard to state if they bring better gameplay to the game. In the research [12], authors let AI play the strategy game as 2D football, it was proven that AI trained teams might underperform in positioning and hit each other, in other words, bunch up to each other. Research of Gomes [13] states that AI might be efficient in solving navigation problems, but there are problems, that need to be solved.

Training AI might be time and resource consuming. As students which have no access to private AI training sources, we can use only free available frameworks or libraries. The usage of free libraries might lead to lags or errors, which will transform to time that will be wasted to solve the problems.

The integration of reinforcement learning methods into games has attracted considerable attention from both academia and the gaming industry. However, some research gaps remain unresolved. For example, most of the research in this area focuses on specific game genres, such as platformers and strategy, which leaves a gap in understanding how RL can benefit a wider range of game types, such as immersive open-world games and virtual reality. Also, existing papers are mainly devoted to agents who prefer short-term rewards in games. But there is a noticeable gap in the development of agents who develop strategies and implement plans to achieve long-term goals in game scenarios. Issues related to long-term agents remain virtually unexplored.

III. Research Methodology

In the context of constantly changing game development, the integration of advanced technologies characterizes the evolution of the industry. One of these technologies, the Unity ML-Agents Toolkit, has become a

Authorized licensed use limited to: Malta College of ArtsScience&Tech. Downloaded on November 24,2024 at 15:22:46 UTC from IEEE Xplore. Restrictions apply.

transformative tool for creating intelligent, adaptive and immersive games. This study examines the synergetic integration of Unity, C# and Unity ML-Agents Toolkit to improve the gameplay of racing games through reinforcement learning. Unity Machine Learning Agents Toolkit (ML-Agents) is an open-source project that provides games and simulators as a learning environment for intelligent agents. The toolkit provides the implementation of modern algorithms based on PyTorch, which allows game developers and amateurs to easily train intelligent agents for 2D, 3D and VR/AR games. Trained agents can be used for various purposes, including controlling the behavior of NPCs.

A distinctive feature of this work is the experimental design of the study aimed at evaluating the possibilities of learning with reinforcement to improve the gameplay in racing games. The research project includes several stages, including the design of the racing environment, training and evaluation of agents trained on reinforcement. In this context, RL allows you to train artificial intelligence agents, specially created as characters of racing games, to navigate complex racetracks, make strategic decisions and arrange complex competitions for the player.

Our hypothesis-oriented research schemes include null (H0) and alternative (H1) hypotheses. The null hypothesis states that the application of reinforcement learning does not lead to a significant improvement in gameplay, and the alternative hypothesis suggests that artificial intelligence agents trained with reinforcement learning can significantly improve gameplay in a racing context.

In this study, a mixed methodology will be used, including both quantitative and qualitative elements.

1. Quantitative aspect: This study uses quantitative methods to evaluate the effectiveness of AI agents based on numerical rewards and points. These quantitative data provide a structured and measurable assessment of the agent's behavior, in particular, his ability to effectively accumulate points.
2. Qualitative aspect: Due to the research nature of this study, there is also a qualitative aspect to it. Its purpose is to study algorithms and behavior of artificial intelligence agents in the framework of simulation modeling. Qualitative data can be valuable for understanding the subtleties and nuances of the agent's interaction with the virtual environment.

In this study, data collection is carried out using a simulated or virtual environment in which artificial intelligence agents act as the main data collection tool. In this context, the agent, virtual environment (simulation or game) and related elements are considered as equivalents of traditional experimental devices.

For this approach used one of the well-known machine learning algorithms, PPO (Proximal Policy Optimization) is a reliable basis for collaborative learning using reinforcement from multiple agents, which often allows you to achieve competitive or superior results both in terms of final gain and sampling efficiency. Below shown configuration (Table 1) for our PPO trainer with hyperparameters and basic learning metrics.

TABLE I. CONFIGURATION FILE FOR TRAINING AGENT

Hyperparameters	**Value**
trainer type	ppo
batch size	1024
buffer size	10240
extrinsic strength	1.0
gail strength	0.8
behavior cloning	0.8
checkpoint interval	500000
max steps	5000000
time horizon	64

Agents are rewarded and punished depending on their behavior and interaction with the environment. The mechanisms of encouragement and punishment are briefly described below:

• Correct checkpoints: If an agent passes the correct checkpoint, he receives a reward of +1.0. This forces the agent to follow the specified path.

• Incorrect checkpoint: If an agent passes an incorrect checkpoint, he receives a penalty of -1.0. This prevents deviation from the specified path.

• Collisions with walls: Collisions with walls are penalized. Collisions impose a -0.5 fine on the agent and encourage safe navigation.

• Continuous contact with the wall: If the agent keeps contact with the wall, the penalty is only -0.1. This helps to avoid collisions and maintain a safe distance from obstacles.

As a result, we trained three different NN models of agent. In these three graphs (Figure 2, 3 and 4), which were generated by TensorBoard tool in ML-agents library, displayed the reward history of the agents. The X axis is the reward amount and Y axis is the number of the step it performed or checkpoints. In these graphs we can see that at the start of training the agents had more negative rewards, than at the end and this might serve as the result, that AI trains and performs better with each step/episode it takes.

Figure 2 represents the agent reward history with Behavior cloning, while Figure 3 is the reward history of agent without behavior cloning. In comparison AI without behavior cloning learns in more stable way and it is better to train an agent without behavior cloning method. Figure 4 shows the Cumulative reward history for the agent, that was trained on complex road. Agent had around 32 million steps, which is 1.8 days of training without any interruptions.

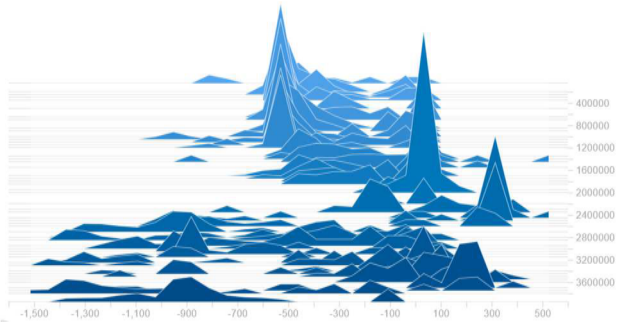

Figure 2. Cumulative reward history for agent with BC

Authorized licensed use limited to: Malta College of ArtsScience&Tech. Downloaded on November 24,2024 at 15:22:46 UTC from IEEE Xplore. Restrictions apply.

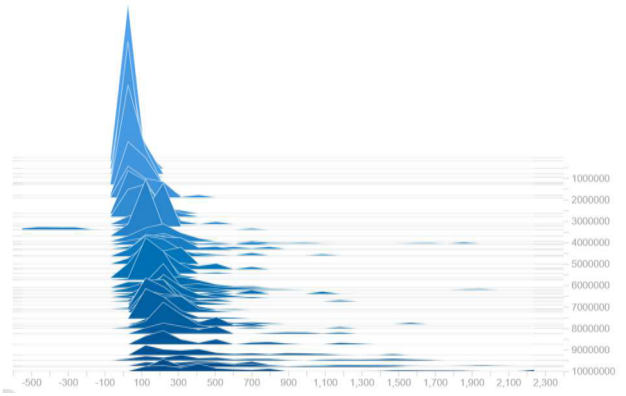

Figure 3. Cumulative reward history for agent without BC

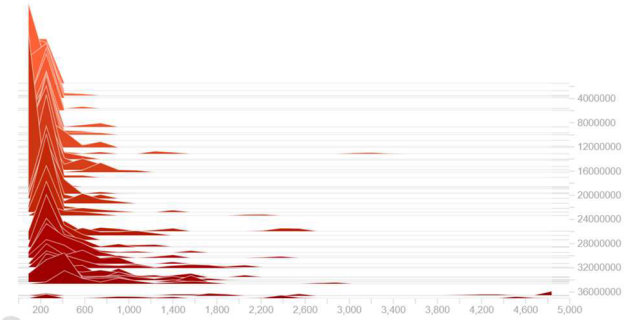

Figure 4. Reward history for agent on complex road

Figure 5 is a representation of Episode length comparison for AI with and without behavior cloning method. X axis is amount of steps, Y axis is duration of 1 episode. The figure shows that AI with behavior cloning has very small changes in duration if we compare its first and last steps, while AI without behavior cloning has evoultuion in duration of each episode. Since agent trains to try new actions and act based on its rewarding results, sometimes agent might seem inconsistent, but in the long term, these actions will help agent to adapt and fix its mistake, if it made one.

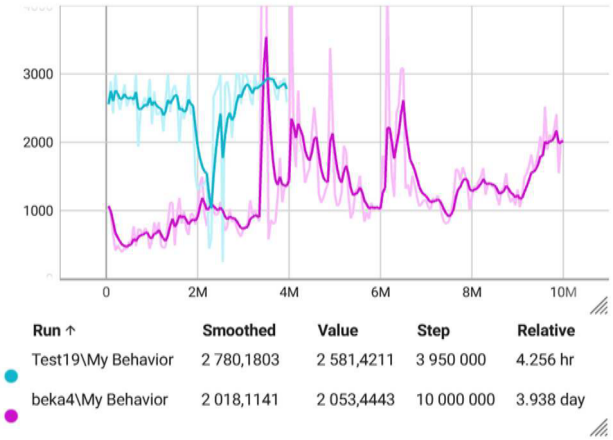

Figure 5. Episode length for training model

The Figure 6 shows the Reward history of agent with more complex training. From the comparison of amount of the rewards at the start, for example 5 millionth step and 35 millionth step, it is clear that agent is developing its driving skills. It is important to mention that the outliers at around 17 millionth step were created because of the experimental changes. These experimental changes were added since the agent seemed as if it was not improving in driving skills and hit the wall at the same place. After reverting the experimental changes and continuing the training process it was discovered that agent just needed more time for the training.

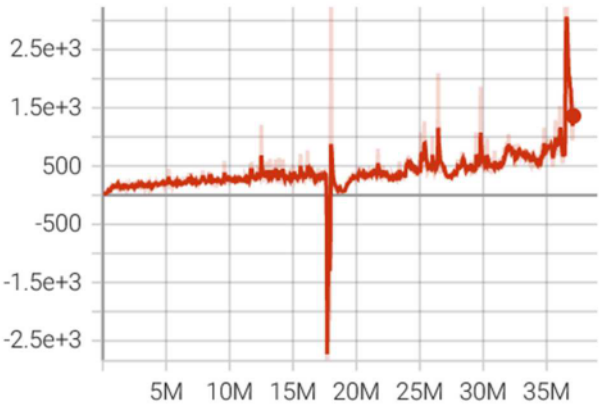

Figure 6. Cumulative reward history for agent with complex road

This study relies heavily on the Unity ML-Agents framework. Even though the framework is a powerful tool, there are limitations on the capabilities and functionality provided by it, which may limit the range of experiments and variations that can be implemented. Also, AI training by reinforcement learning is time-consuming action, since agent tries to input a lot of random numbers to move. It is difficult for standard RL algorithms to find a good policy for unformed reward functions, and the learning rate is very low. Things that are very easy to people, like pressing forward button in a certain time, then pressing turning button at a certain place might be very difficult for AI, because it doesn't understand the principles of surroundings, time, and movement, it only uses numbers. The movement is hard for our agent, because it has 2 action branches, numbers for forward acceleration, backward acceleration, and numbers for turning left, right, or not turning, since these actions are hard for agent to understand, the agent couldn't train properly, even when after 500000 steps. It could've trained faster, if it had only 1 branch of movement, like left, right, forward, backward, but these movements are not for realistic car movement.

IV. Results and Discussion

Environment that was provided for the agent is the square-shaped road with the invisible walls to avoid and checkpoints for the agent to pass. The houses, trees and other game objects are decorations, which are placed behind the walls.

In Figure 7, it is shown the environment of the game, that was created for the first version. Yellow car is the agent, gray car is the player to compete with. In Figure 8, it is shown the second environment with more hard curves, checkpoints and with both left and right sided turns.

Authorized licensed use limited to: Malta College of ArtsScience&Tech. Downloaded on November 24,2024 at 15:22:46 UTC from IEEE Xplore. Restrictions apply.

Figure 7. Final game environment (Map 1)

Figure 8. Final game environment (Map 2)

In the following diagram (Figure 9) we have five entities CheckpointSingle, TrackCheckpoints, CarDriver, CarDriverPlayer, CarDriverAgent and their relationship. The relationship between TrackCheckpoints and CheckpointSingle is “one to many” as a tracking can have many checkpoints. CarDriver is main controller of car, it has many properties and methods which helps to car move. Player and Agent have “one to one” relationship with CarDriver.

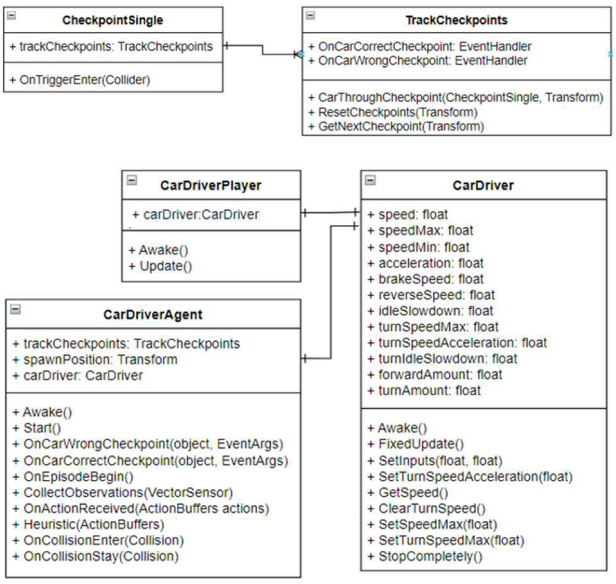

Figure 9. ER Diagram of Software part

During the training phase, agent was cloned and set in clear location. It appears, that agent learns to slow down or stop completely, if it had punishments in the given position during the previous episodes. This caused for the training agent to slow down on first turn (left upper corner in the Figure 10), since it remembers its first punishments, when it didn’t know how to turn.

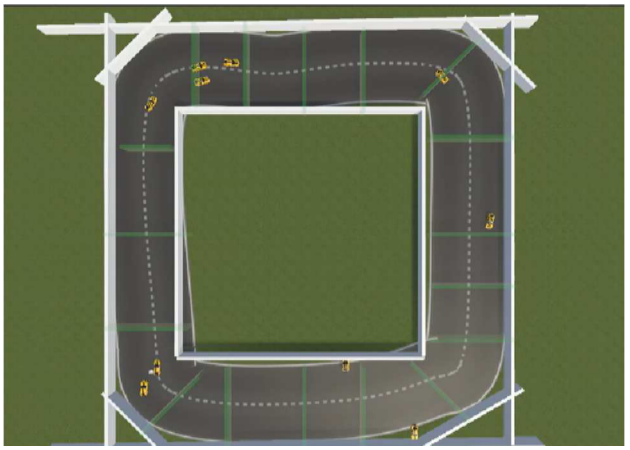

Figure 10. Training environment

The agent’s neural network model, in other words, brain, that was trained in more complex road (Figure 11) was able to easily ride in first version of the road (figure 10), which means, that agent learns how to ride and move according to its ray-perception sensors and not the road pattern, so if the agent will be trained in different kinds of road, it will be able to adapt to environment and challenge the players.

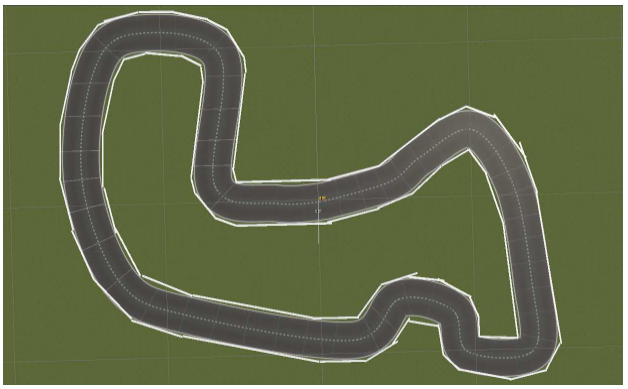

Figure 11. Complex training environment

In Figure 12 there is shown the gameplay of our game and competition with the agent. Agent learned to speed up on direct open road but slows down on turns, it caused for Player to be able to win the race, since players with experience in racing games are used to turn without slowing down.

Figure 12. Gameplay of racing with Hard Mode

For the future work of this research, we might test an agent with a more complex car controller, where it should be changing the speed limit at the certain speed like in manual transmission by pressing special button.

Authorized licensed use limited to: Malta College of ArtsScience&Tech. Downloaded on November 24,2024 at 15:22:46 UTC from IEEE Xplore. Restrictions apply.

V. CONCLUSION

In conclusion, after observing other experiments and games, we can state that AI is not worth to use in games if the game is considered hard for agent to learn. If the game is casual or hyper casual, where agent does not need a lot of actions to do, like in Go or Chess, usage of AI might be suitable, but in games like racing, strategy, action games or shooters, where there are a lot of mechanics and actions, it is better to use pre-scripted bots or NPCs. In terms of resources and outcomes of the agent training, it might not be the best solution to use AI agents to replace an NPC. So, for now we are rejecting our alternative hypothesis, and accepting null hypothesis by saying that the application of reinforcement learning does not lead to a significant improvement in gameplay.

VI. ACKNOWLEDGEMENT

This research has been funded by Science Committee of the Ministry of Science and Higher Education of the Republic of Kazakhstan (AP19579354)

VII. REFERENCES

[1] Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep Reinforcement Learning: A Brief Survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, Nov. 2017, doi: https://doi.org/10.1109/msp.2017.2743240.

[2] "3.1 The Agent-Environment Interface," Incomplete Ideas, http://www.incompleteideas.net/book/ebook/node28.html

[3] A. Dosovitskiy, G. Ros, F. Codevilla, A. López, and V. Koltun, "CARLA: An Open Urban Driving Simulator," 2017. Available: http://proceedings.mlr.press/v78/dosovitskiy17a/dosovitskiy17a.pdf

[4] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," Journal of Field Robotics, vol. 37, no. 3, Nov. 2019, doi: https://doi.org/10.1002/rob.21918.

[5] K. Güçkıran and B. Bolat, "Autonomous Car Racing in Simulation Environment Using Deep Reinforcement Learning," 2019 Innovations in Intelligent Systems and Applications Conference (ASYU), Izmir, Turkey, 2019, pp. 1-6, doi: 10.1109/ASYU48272.2019.8946332.

[6] Y. Savid, R. Mahmoudi, R. Maskeliūnas, and R. Damaševičius, "Simulated Autonomous Driving Using Reinforcement Learning: A Comparative Study on Unity's ML-Agents Framework," Information, vol. 14, no. 5, p. 290, May 2023, doi: 10.3390/info14050290.

[7] C. Urrea, F. Garrido, and J. Kern, "Design and Implementation of Intelligent Agent Training Systems for Virtual Vehicles," Sensors, vol. 21, no. 2, p. 492, Jan. 2021, doi: 10.3390/s21020492.

[8] X. Hou, "Exploring the Role of Reinforcement Learning in Video Game Environments," Advances in computer science research, pp. 193–201, Jan. 2023, doi: https://doi.org/10.2991/978-94-6463-300-9_20.

[9] J. Ortega, N. Shaker, J. Togelius, and G. N. Yannakakis, "Imitating human playing styles in Super Mario Bros," Entertainment Computing, vol. 4, no. 2, pp. 93–104, Apr. 2013, doi: https://doi.org/10.1016/j.entcom.2012.10.001

[10] Kristián Kovalský and G. Palamas, "Neuroevolution vs Reinforcement Learning for Training Non Player Characters in Games: The Case of a Self Driving Car," Lecture Notes in Computer Science, pp. 191–206, Dec. 2020, doi: https://doi.org/10.1007/978-3-030-76426-5_13.

[11] P. Afonso, CarvalhoV., and A. Simões, "Reinforcement Learning Applied to AI Bots in First-Person Shooters: A Systematic Review," Algorithms, vol. 16, no. 7, pp. 323–323, Jun. 2023, doi: https://doi.org/10.3390/a16070323.

[12] A. Smit, H. A. Engelbrecht, W. Brink, and A. Pretorius, "Scaling multi-agent reinforcement learning to full 11 versus 11 simulated robotic football," Autonomous Agents and Multi-Agent Systems, vol. 37, no. 1, Mar. 2023, doi: https://doi.org/10.1007/s10458-023-09603-y.

[13] G. Gomes, Creto Augusto Vidal, Joaquim Bento Cavalcante-Neto, and Y. Lenon, "Two Level Control of Non-Player Characters for Navigation in 3D Games Scenes: A Deep Reinforcement Learning Approach," Oct. 2021, doi: https://doi.org/10.1109/sbgames54170.2021.00030.

Authorized licensed use limited to: Malta College of ArtsScience&Tech. Downloaded on November 24,2024 at 15:22:46 UTC from IEEE Xplore. Restrictions apply.