International Journal of Internet, Broadcasting and Communication Vol.16 No.2 267-277 (2024)
http://dx.doi.org/10.7236/IJIBC.2024.16.2.267

IJIBC 24-2-31

Goal-oriented Movement Reality-based Skeleton Animation Using Machine Learning

Yu-Won JEONG

Visiting Professor, College of IT Engineering, Dept. of Media Software, Sungkyul University, Korea
new1thexx@sungkyul.ac.kr

Abstract

This paper explores the use of machine learning in game production to create goal-oriented, realistic animations for skeleton monsters. The purpose of this research is to enhance realism by implementing intelligent movements in monsters within game development. To achieve this, we designed and implemented a learning model for skeleton monsters using reinforcement learning algorithms. During the machine learning process, various reward conditions were established, including the monster's speed, direction, leg movements, and goal contact. The use of configurable joints introduced physical constraints. The experimental method validated performance through seven statistical graphs generated using machine learning methods. The results demonstrated that the developed model allows skeleton monsters to move to their target points efficiently and with natural animation. This paper has implemented a method for creating game monster animations using machine learning, which can be applied in various gaming environments in the future.

The year 2024 is expected to bring expanded innovation in the gaming industry. Currently, advancements in technology such as virtual reality, AI, and cloud computing are redefining the sector, providing new experiences and various opportunities. Innovative content optimized for this period is needed to offer new gaming experiences. A high level of interaction and realism, along with the immersion and fun it induces, must be established as the foundation for the environment in which these can be implemented. Recent advancements in AI technology are significantly impacting the gaming industry. By applying many elements necessary for game development, AI can efficiently optimize the game production environment. Through this research, We demonstrate that the application of machine learning to Unity and game engines in game development can contribute to creating more dynamic and realistic game environments. To ensure that VR gaming does not end as a mere craze, we propose new methods in this study to enhance realism and immersion, thereby increasing enjoyment for continuous user engagement.

Keywords: *Game, Machine-Learning, Reinforcement Learning, Animation, Immersion, Fun, Skeleton*

Manuscript Received: April. 18, 2024 / Revised: April. 27, 2024 / Accepted: May. 1, 2024
Corresponding Author: new1thexx@sungkyul.ac.kr (Yu-won JEONG)
Tel: +82-31-467-8438
Visiting Professor, College of IT Engineering, Dept. of Media Software, Sungkyul University, Korea

1. Introduction

The year 2024 is expected to be another year of innovation in the gaming industry. Apart from the growth and profitability of the industry, the current trend in gaming innovation is poised to redefine the industry through technological advancements such as virtual reality (VR) and artificial intelligence (AI), which provide new experiences and diverse opportunities. According to Statista, the VR software market began to grow exponentially from 2021 due to the impact of COVID-19. In 2021, it nearly doubled from the previous year, reaching $1.5 billion (approximately 2.025 trillion KRW), and is estimated to reach $4.18 billion (approximately 5.643 trillion KRW) by 2024. The VR software market is divided into the VR video and VR gaming markets, with the proportion of the VR gaming market gradually increasing [1]. One of the growth factors is the release of next-generation VR devices. Meta's Meta Quest has been leading the popularization of VR devices with a focus on cost-effectiveness, and the third-generation Oculus HMD, released last October, has gained popularity among consumers due to its enhanced performance and reasonable price. Meta is also striving to build a VR market ecosystem through its own app store. Apple's Vision Pro is also expected to have a significant impact on the VR market. Additionally, PlayStation VR 2, released in early 2023, has solidified its position as a console gaming device. Companies such as Samsung, Google, and Qualcomm have also announced new HMD releases for 2024, offering consumers more VR device choices than ever before.

To provide new gaming experiences optimized for these next-generation VR devices, innovative content is essential. For the sustained growth of the VR gaming market, high levels of interaction, realism in games provided through VR devices like HMDs, and consequently, immersion and fun are necessary. The environment to implement this must be established. In line with this trend, advancements and innovations in AI technology are beginning to significantly impact the gaming industry. AI can optimize the game development environment by applying many necessary elements to game production efficiently.

2. Related Works

2.1. Reinforcement Learning

Reinforcement Learning (RL) is a concept devised by Marvin Lee Minsky, which aims to solve specific problems efficiently based on accumulated experiences through pattern recognition and trial-and-error, especially when the method to solve the problem is unknown [2-3]

Figure 1 explains the structure of reinforcement learning. the agent observes the state of the environment After executing the action, the agent receives a reward and updates its knowledge to improve future decision-making.

Reinforcement Learning is influenced by behavioral psychology, and its structure involves an agent observing the environment and its state to perform an action. When the agent performs the correct action among the possible actions, it receives a reward. This method involves selecting actions or sequences of actions. However, RL has the limitation that it is not always clear what action should be taken next [4-5].

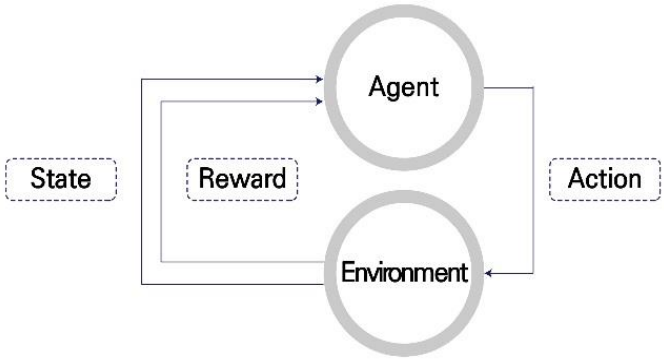

Figure 1. Reinforcement learning structure

2.2. ML-Agent Machine Learing

ML-Agents is an officially supported machine learning toolkit by Unity, which serves as an open-source platform for training game and simulation agents [6]. It provides PyTorch-based algorithms and enables the training of agents through reinforcement learning, imitation learning, and neuroevolution. As an open-source project, it allows the use of games and simulations as environments for training intelligent agents. It offers algorithms that enable game developers to easily train intelligent agents for 2D, 3D, and VR/AR games. The ML-Agents toolkit benefits both game developers and AI researchers by providing a central platform to evaluate AI advancements in Unity's rich environments and then access a broader research and game development community [7]

2.3. PPO(Proximal Policy Optimization)

The PPO (Proximal Policy Optimization) algorithm is a type of reinforcement learning algorithm that is an advanced form of policy gradient [8]. It is advantageous because it simplifies the calculation to a first-order format, making it easy to implement and fast in performance [9]. Additionally, it accelerates learning by reusing the same experience multiple times and imposes limits to prevent excessively rapid changes [10]. It enhances the stability of the learning process by applying clipping to limit the size of policy updates.

Figure 2 illustrates the PPO algorithm. The PPO algorithm is used to optimize the policy (the strategy for generating responses) of a Large Language Model (LLM). When the LLM performs various tasks, its performance can be improved through reinforcement learning. The Reward Model evaluates the quality of the LLM's outputs (responses) and is used by the PPO algorithm to update the policy.

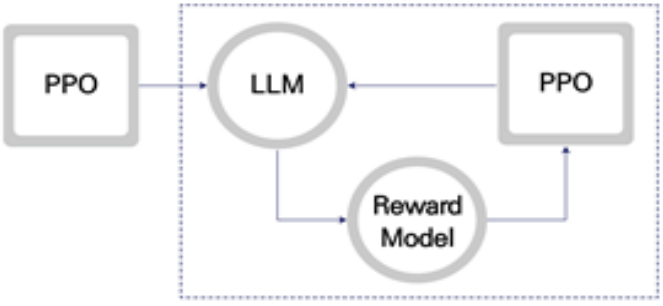

Figure 2. PPO Algorithm structure

Unity's Configurable Joint component allows for high customization as it integrates all the functionalities of different joint types. This means anything from modified versions of existing joints to specially designed unique joints can be created. In detail, like other joints, the Configurable Joint restricts the movement of objects and uses forces to move them to a target speed or position. However, with many available configuration options, it can be very subtle when combined. Experimenting with different options is necessary to achieve a joint that moves exactly as desired [11].

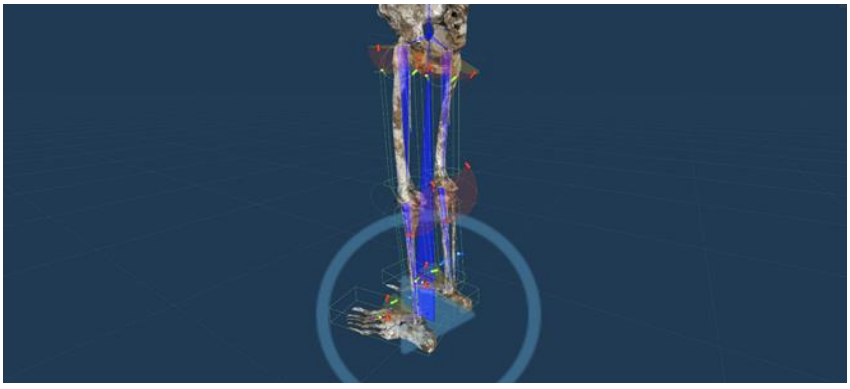

Figure 3. Configurable joint of skeleton

Using the Configurable Joint, as shown in Figure 3, specific parts of the skeleton can be designed to move only to certain positions, restricting the rotation values to simulate a humanoid skeleton.

3. System

3.1. Skeleton Monster Design

Figure 4 shows the basic skeleton object for the skeleton monster machine learning currently in development. It is based on the Walker example from Unity's ML-Agents and uses the Configurable Joint to adjust the rotation values of each bone, allowing it to move like a human.

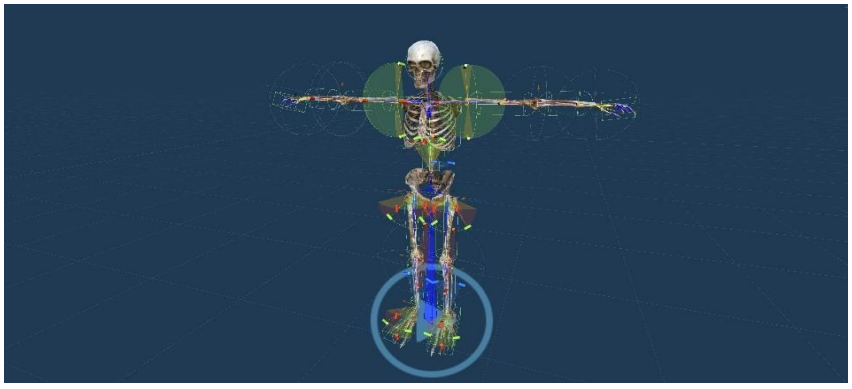

Figure 4. Object of the developed skeleton

The conditions for the skeleton monster to receive rewards include speed matching reward, directional reward, leg movement reward, and target contact reward. The speed matching reward incentivizes reaching the target quickly. The directional reward is adjusted based on how well the agent's head direction aligns with the target direction, with the angle set between -10˚ to 10˚. The leg movement reward was implemented to encourage more leg movement, as the legs were not used much in the initial development stages. The target contact reward is given when the target is touched. Conditions for not receiving rewards include mismatched speed and direction, and inefficient leg movement. If the speed decreases below a certain level, a negative reward is given. Similarly, if the direction falls outside the -10˚ to 10˚ range, a negative reward is applied. Inefficient leg movement also results in a negative reward to encourage more active leg use.

Table 1 represents the content as follows. the hyperparameter settings used in machine learning for the skeleton model. These values are provided by default in Unity machine learning and can be easily accessed through configuration files with the yaml extension. Batch size refers to the number of experiences processed in each iteration. Buffer size indicates the number of experiences collected before updating the policy. Learning rate sets the initial learning rate used in gradient descent. Beta increases the entropy of the policy to introduce more randomness. Epsilon is used to control the rate of policy improvement. Lambda determines whether to rely more on the estimated reward or the actual received reward. Num epoch sets the number of times the data in the experience buffer is processed. The learning_rate_schedule can be set to linear or constant; if set to linear, the learning rate gradually decreases as learning progresses, promoting stable learning in environments with longer episodes.

Table 1. Hyper parameter setting values

Hyper parameter			
Name	Value	Name	Value
Batch_size	2048	Hidden_units	256
Buffer_size	20480	Num_layers	3
Learning_rate	0.0003	Gamma	0.995
Beta	0.005	Strength	1
Bpsilon	0.2	Keep_checkpoints	5
Lambda	0.95	Max_steps	50000000
Num_epoch	3	Time_horizon	10000
Learning_rate_schedule	linear	Summary_freq	30000

The right side of Table 1 is as follows Hidden units determine the number of units in the neural network, which can be set to simple or complex depending on the network's complexity. Num layers sets the number of layers in the network. Gamma sets the discount rate for the reward signal. Strength adjusts the factor for the set reward. Keep checkpoints determines how often the training model is saved. Max steps indicates the maximum number of steps for the training. Time horizon sets the number of experiences steps the agent collects before storing them in the experience buffer; a larger value increases variance, while a smaller value decreases variance. Summary frequency sets the frequency at which the training cycle is output to the console and can be adjusted depending on the complexity of the environment.

4. Experiments

In this paper, we verified the performance of seven different implementations to enhance the behavior of a skeleton monster using machine learning-based reinforcement learning. Table 2 shows the list of experiments The changes in cumulative reward indicate the successful progression of learning, while the episode length quantity reflects the improvement in the agent's mobility. Changes in loss signify the situation understanding through accurate prediction. Policy loss highlights the stability of policy loss levels. Changes in the learning rate are associated with entropy changes based on the action policy and the speed of learning over time.

Table 3 details the hardware and software environments used in the experiments.

Table 2. Performance verification targets

Method	Significance
Changes in cumulative reward	Successful progression of learning
Episode length quantity	Improvement in agent's mobility
Changes in loss.	Situation understanding through accurate prediction
Policy loss	Stability of policy loss levels
Changes in learning rate.	Entropy changes based on action policy
Changes in learning rate.	Speed of learning over time

Table 3. H/W and S/W Spec for experiments

Method	**Significance**
Platform	Unity (Ver.2023.2)
Machine Learning	ML-Agents (Ver.21)
Language	Python (Ver.3.11)
CPU	Intel Core i7-12700K
GPU	GeForce RTX 4060
RAM	32GB

Figure 5 is a TensorBoard graph visualizing the changes in cumulative rewards during the training process of the skeleton monster agent using reinforcement learning. The data depicted was obtained from a training session conducted over a total of 50,000,000 steps.

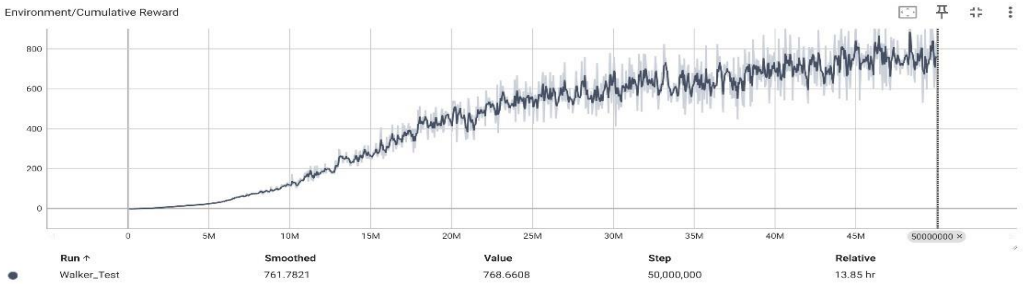

Figure 5. Accumulated compensation graph

The horizontal axis of the graph represents the number of training steps, while the vertical axis represents the cumulative rewards obtained by the agent up to that point in the environment. In the initial stages, the reward levels are low, but as the agent progresses in its learning, the rewards gradually increase. The volatility observed in the mid-stage reflects the agent exploring various strategies during the learning process. This volatility signifies the agent's attempts to try new strategies and converge towards an optimal direction. Ultimately, the graph shows a stable rise towards relatively high cumulative reward values, indicating successful progress in the learning process.

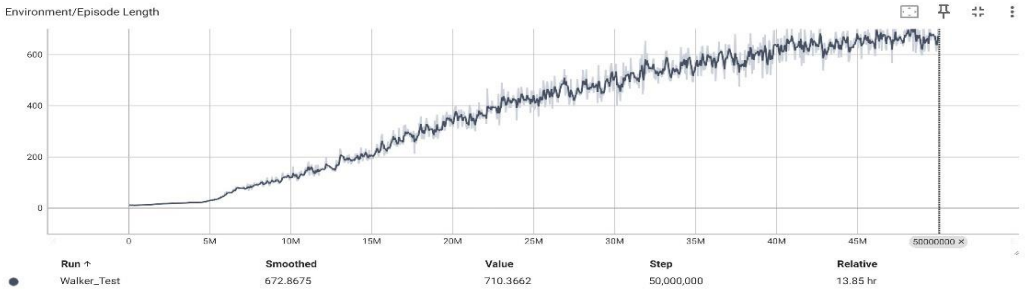

Figure 6. Episode length graph

Figure 6 illustrates the phenomenon of the episode length gradually increasing as the skeleton monster agent undergoes the reinforcement learning process. In the early stages of training, the agent has not yet learned appropriate behaviors, so episodes reset as soon as any part of the body other than the feet touches the ground. This reflects the agent's unstable movement patterns and is clearly evident through the short episode lengths. However, over time, the graph shows a gradual upward trend, indicating an improvement in the agent's locomotion abilities through learning. This signifies the agent's enhanced ability to walk stably without falling, which is a crucial indicator of progress. Consequently, the observed increase in episode length during the training process indicates that the skeleton monster's machine learning is developing in the right direction.

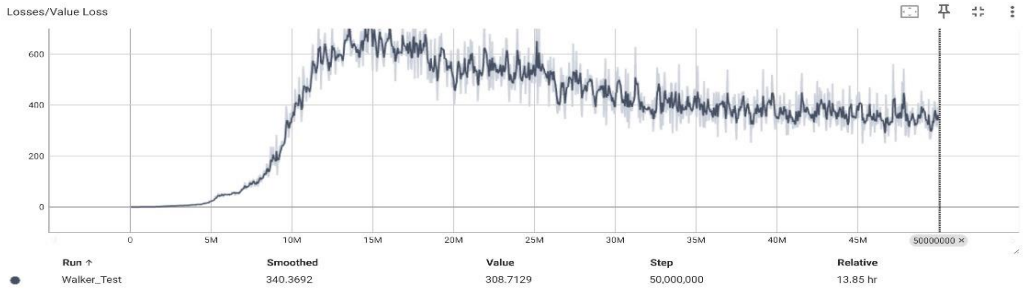

Figure 7. Loss prediction graph

Figure 7 visualizes the changes in the loss rate for predicting the value of each state during the reinforcement learning process of the skeleton monster agent. The graph shows the number of training iterations on the X-axis and the loss values on the Y-axis. The loss value is an indicator of how accurately the agent predicts the value of a specific state, with a lower loss value indicating more accurate predictions. Initially, the loss value is high, but as training progresses, the loss rate gradually decreases, forming a downward curve. This indicates that the agent is learning to predict the value of each state more accurately through experience. From the middle

stage onwards, the loss rate stabilizes and reaches a consistent level, indicating that the learning process is successfully progressing. Successful reinforcement learning is characterized by a continuously decreasing loss rate that eventually stabilizes. These changes demonstrate that the agent is learning a more accurate policy, enabling it to take optimal actions in the given environment.

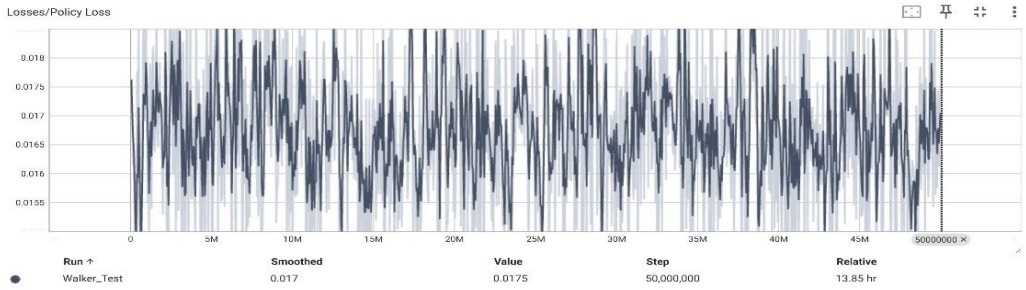

Figure 8. Policy loss graph

Figure 8 presents the policy loss graph, which quantitatively illustrates the changes in the decision-making process of the skeleton monster agent during the reinforcement learning process. As shown in the graph, the agent tends to maintain a certain level of loss value while improving its policy through training. This characteristic suggests that the agent maintains consistent performance even during the learning process. The policy loss values remaining between approximately 0.018 and 0.0155 on average indicate that the agent is undergoing a relatively stable learning process. This shows that the agent is efficiently learning about the given environment and continuously optimizing its behavior to maximize the overall reward. The stable level of policy loss also indicates that the agent is maintaining a good balance between exploration and exploitation. A loss value that is too high would imply excessive exploration, while a loss value that is too low would imply insufficient exploration. This graph demonstrates that the agent is finding an appropriate balance between these two aspects.

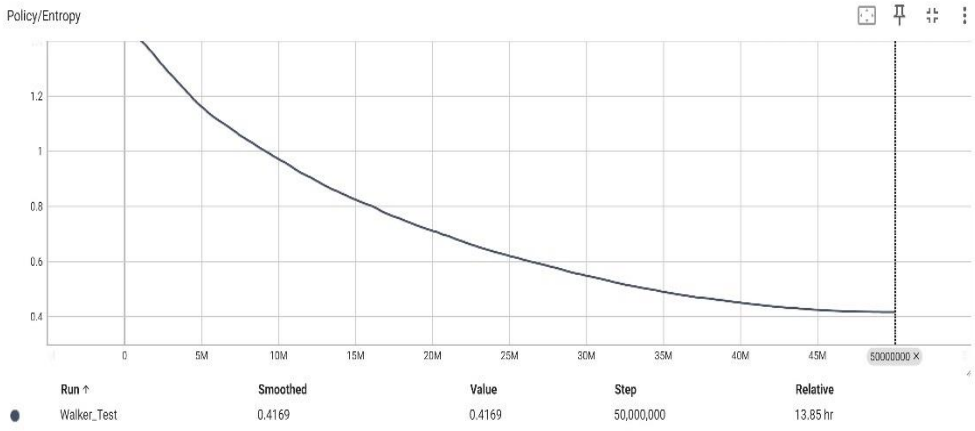

Figure 9. Entropy graph

Figure 9 illustrates the changes in the entropy of the agent's behavior policy as it improves through the reinforcement learning algorithm. Entropy measures the degree of randomness in the agent's decision-making process and typically decreases as the agent's policy becomes more decisive. The graph in Figure 9 shows how policy entropy changes over the course of 50,000,000 time steps of training. Starting with a high entropy value

reflects the agent's exploration of various actions. As training progresses, the continuous decrease in entropy indicates that the agent is learning more accurate and predictable behavior patterns, converging towards decisive action choices. The stable downward curve demonstrates that the agent is learning about the environment, acting accordingly, and developing in the right direction.

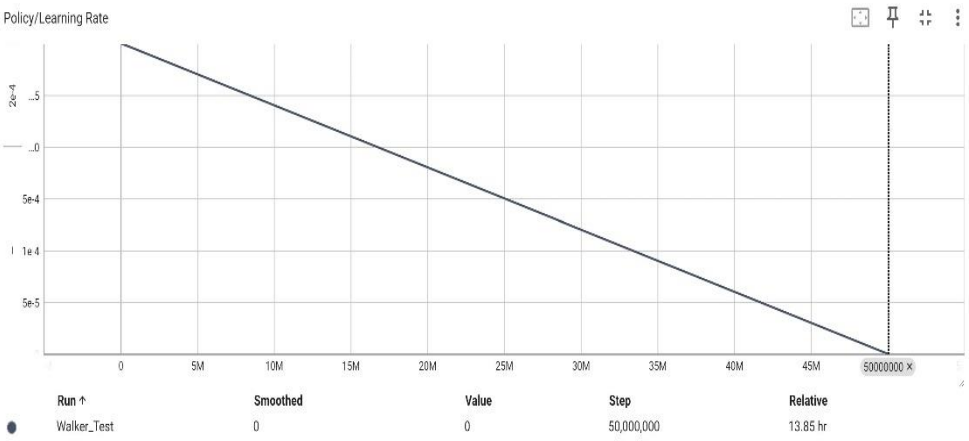

Figure 10. Learning rate graph

Figure 10 is a graph illustrating the decrease in learning speed of the reinforcement learning algorithm. This graph shows the trend of the time required for the algorithm to search for the optimal policy decreasing throughout the learning process. The learning speed is based on the learning rate applied to update the agent's behavior policy, and it can be seen that it gradually decreases as learning progresses compared to the initial stage. This is a common technique used in reinforcement learning: a high learning rate is initially applied to allow the agent to quickly learn about the environment and explore various strategies. However, as the agent starts to form an optimized policy based on accumulated experience, the learning rate is reduced to decrease volatility and promote stable learning. This approach helps the agent minimize unnecessary exploration and effectively utilize the knowledge gained through learning to make policy decisions more quickly, thereby reducing the overall training time.

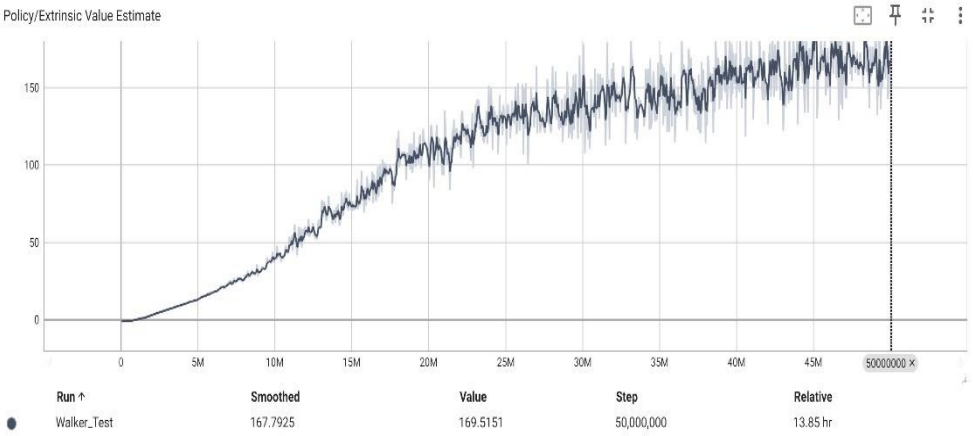

Figure 11. Learning rate graph

Figure 11 is a graph depicting the changes in policy value estimates resulting from the reinforcement learning algorithm. This graph represents the average value estimates of the policy across the entire state space, which is a crucial indicator for understanding the algorithm's learning performance. The graph shows the sum of the predicted value functions for various states experienced by the agent over 50,000,000 steps of learning. Starting with low value estimates initially, the graph gradually rises as learning progresses, indicating that the agent is learning a policy that yields higher rewards. In the early stages of learning, the agent lacks information about the environment, resulting in low value estimates. However, over time, as the agent accumulates more experience and forms more effective behavior policies, the value estimates show an upward trend. The graph's volatility reflects the natural process of the agent exploring and learning new policies. This volatility tends to decrease over time, suggesting that the agent is converging towards a consistent policy.

5. Conclusion

In this paper, we developed a mechanism for generating goal-oriented movement and realistic animations for skeleton monsters, commonly used in game environments. We employed the PPO algorithm and the ML-Agents toolkit to adjust the monster's movements in real-time and researched natural animation effects through the use of Configurable Joints that incorporate physical constraints. This approach enables the monster to move in a way that closely resembles real-life behavior. Experimental results demonstrate that this technical approach can effectively control the monster's movements, adding substantial realism to the game environment. Specifically, the design of the learning environment and various reward mechanisms played a crucial role in optimizing the monster's movement patterns, contributing to advancements in future game development and simulation technologies. Our research contributes to the field of game development using machine learning and plans to explore applications for various characters and complex game scenarios to implement diverse game environments in future research. Additionally, the technology for enhancing the intelligence of agents using machine learning is expected to exhibit potential across various platforms, including virtual reality and augmented reality. Moving forward, we plan to apply machine learning to the animation production of other types of monsters, creating realistic animations and expanding development by applying machine learning to various monsters.

References

[1] Global Game Industry Trends, July/August 2023, Issue No. 60, 2023.

[2] S. D. Yoo, "Governance research for Artificial intelligence service", The Journal of The Institute of Internet, Broadcasting and Communication, Vol. 24, No. 2, pp.15-21, Apr 30, 2024. DOI: https://doi.org/10.7236/JIIBC.2024.24.2.15

[3] H. W. Jo and H. W. Jung, "Design and Implementation of RPG Auto Play System using Reinforcement Learning", Journal of the Korea Entertainment Industry Association, 17(4), pp. 351-359, 2023. DOI: 10.21184/jkeia.2023.6.17.4.351

[4] Y.H. Lee, "Implementation of Intelligent Agent Based on Reinforcement Learning Using Unity ML-Agents", The Journal of JIIBC, Volume 24 Issue 2, pp. 205-211, 2024. DOI: https://doi.org/10.7236/JIIBC.2024.24.2.205

[5] D. S. Lim, Y. A. Min and D. K. Lim, "Recommendation System of University Major Subject based on Deep Reinforcement Learning", The Journal of The Institute of Internet, Broadcasting and Communication, Vol. 23, No. 4, pp.9-15, Aug 31, 2023. DOI: https://doi.org/10.7236/JIIBC.2023.23.4.9

[6] C. S. Kim, N. G. Kim and K. K. Young, "Research Trends Analysis of Machine Learning and Deep Learning:

Focused on the Topic Modeling", Journal of the Korea Society of Digital Industry and Information Management, 15(2), pp.19-28, 2019. DOI: https://doi.org/10.17662/ksdim.2019.15.2.019

[7] S. C Park, D. Y. Kim and W. J. Lee, "UnityPGTA: A Unity Plat former Game Testing Automation Tool Using Reinforcement Learning", Journal of KIISE, 51(2), pp.149-156, 2024. DOI: 10.5626/JOK.2024.51.2.149

[8] Unity-Technologies, ml-agents, https://github.com/Unity-Technologies/ml-agents

[9] D. E. Park, "Design and Implementation of Puzzle Game Play Agent Using PPO Algorithm", Journal of JCIT, vol.11, no.3, pp. 1-6, 2021. DOI : 10.22156/CS4SMB.2021.11.03.001

[10] S. G. Park and D. H. Kim, "Autonomous Flying of Drone Based on PPO Reinforcement Learning Algorithm", Journal of Institute of Control, Robotics and Systems, 26(11), pp. 955-963, 2020. DOI: 10.5302/J.ICROS.2020.20.0125

[11] S. G. Lee, K. S. Byun and H. J. Yoon, "Adaptive Fast Calibration Method for Active Phased Array Antennas using PPO Algorithm", Journal of IKEEE, 27(4), pp. 269-276, 2023. DOI: https://doi.org/10.7471/ikeee.2023.27.4.636