iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** V **Month of publication:** May 2026

DOI: https://doi.org/10.22214/ijraset.2026.82063

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue V May 2026- Available at www.ijraset.com

Implementation of a Reinforcement Learning-Based Adaptive NPC in a Semi-Open World 3D Game Environment

Visakh Raj[1], Adithya T S[2], Mrudul Maanas[3], Abhimanyu K P[4], Reshma P D[5]
[1, 2, 3, 4]*Dept. of Computer Science and Engineering, Universal Engineering College, Thrissur, India*
[5]*Assistant Professor, Dept. of Computer Science and Engineering, Universal Engineering College, Thrissur, India*

Abstract: ***This paper presents the implementation of a semi-open world 3D role-playing game (RPG) featuring an adaptive non-player character (NPC) driven by reinforcement learning. The system is developed using the Unity game engine integrated with the Unity ML-Agents toolkit, enabling the antagonist to learn and evolve its behavior based on player interactions across multiple gameplay episodes. The game environment is structured into three interconnected biomes—a mansion, a forest, and a tunnel—each introducing progressively complex objectives and challenges. The NPC is designed to detect and pursue the player based on movement patterns, while also responding dynamically to environmental stimuli such as a flashlight-based defense mechanism that temporarily immobilizes it. Through iterative training, the NPC improves its decision-making and pursuit strategies, resulting in more intelligent and unpredictable gameplay. The implementation demonstrates the effectiveness of reinforcement learning in creating adaptive, context-aware, and behaviorally evolving game agents, contributing to more immersive and dynamic player experiences.***

Index Terms: Reinforcement Learning, NPCs, Game AI, Procedural Content Generation, Dynamic Difficulty Adjust- ment.

I. INTRODUCTION

The advancement of artificial intelligence (AI) in game development has significantly transformed the way non-player characters (NPCs) interact with players. Traditional game AI systems rely heavily on predefined rules, scripted behaviors, and finite state machines, which often result in predictable and repetitive interactions. While such approaches are compu- tationally efficient, they lack adaptability and fail to provide dynamic and immersive gameplay experiences.

Recent developments in machine learning, particularly reinforcement learning (RL), have introduced new possibilities for creating intelligent and adaptive game agents. Reinforcement learning enables agents to learn optimal behaviors through interaction with their environment by maximizing cumulative rewards. This approach allows NPCs to evolve their strate- gies over time, adapting to player actions and environmental conditions without requiring explicit programming of every possible behavior.

In this paper, we present the implementation of a semi-open world 3D role-playing game (RPG) that incorporates an AI- driven antagonist capable of adaptive behavior using reinforcement learning. The system is developed using the Unity game engine integrated with the Unity ML-Agents toolkit, which facilitates the training and deployment of learning agents within interactive environments. The game is structured into three distinct biomes—a mansion, a forest, and a tunnel—each designed to introduce progressively complex objectives and challenges for the player.

The antagonist, modeled as a demon NPC, is trained to detect, pursue, and interact with the player based on movement patterns and environmental cues. A key gameplay mechanic involves the use of a flashlight, which temporarily immobilizes the NPC, introducing a strategic element to player-agent interaction. Through repeated gameplay episodes, the NPC learns to refine its pursuit strategies, resulting in increasingly intelligent and less predictable behavior.

The primary contribution of this work lies in the integration of reinforcement learning into a multi-biome game environ- ment to create an adaptive NPC capable of continuous behavioral improvement. Unlike traditional scripted systems, the proposed approach enables dynamic interaction, enhancing realism and engagement in gameplay. This implementation demonstrates the practical feasibility of incorporating learning-based AI techniques in modern game design and highlights their potential to redefine interactive experiences in gaming environments.

II. RELATED WORK

A. Traditional Game AI and Adaptive Systems

Traditional approaches to game artificial intelligence primarily rely on techniques such as finite state machines, behavior trees, and rule-based systems, as discussed by Millington and Funge (2009). These methods are computationally efficient and widely used in commercial game development due to their predictability and ease of implementation. However, they lack adaptability and fail to respond dynamically to changing player behavior, resulting in repetitive and predictable game- play experiences.

To address these limitations, research has shifted toward player-centered AI, which emphasizes adaptive systems capable of adjusting difficulty and personalizing gameplay based on user behavior (Yannakakis and Togelius, 2018). The theoretical foundation for such adaptive systems is rooted in reinforcement learning (RL), formalized by Sutton and Barto (2018), where agents learn optimal behavior through interaction with the environment using concepts such as states, actions, re- wards, and policies. While these frameworks provide strong theoretical grounding, their direct application in real-time game environments introduces challenges related to computational efficiency and stability.

B. Reinforcement Learning in Game Environments

The integration of deep learning with reinforcement learning has significantly advanced adaptive game AI. Mnih et al. (2015) introduced Deep Q-Networks (DQN), which demonstrated human-level performance across Atari games by learn- ing directly from high-dimensional sensory inputs. This approach enabled agents to generalize across different states using neural networks. However, DQN suffers from training instability, high computational requirements, and limited scalability in complex 3D environments with continuous action spaces.

To improve training stability and efficiency, Schulman et al. (2017) proposed Proximal Policy Optimization (PPO), a policy-gradient method that balances exploration and exploitation while maintaining robust performance. PPO has become widely adopted in dynamic simulation and game environments due to its stability and ease of implementation. Despite its advantages, it still requires careful hyperparameter tuning and significant training time.

The practical application of reinforcement learning in game development was further enhanced by Juliani et al. (2018) through the Unity ML-Agents Toolkit, which enables training intelligent agents within real-time 3D environments. This framework allows seamless integration of reinforcement learning models into game engines, supporting both training and inference phases. However, designing effective reward functions and achieving stable convergence remain challenging tasks.

Highly advanced systems such as AlphaGo (Silver et al., 2016) and AlphaStar (Vinyals et al., 2019) have demonstrated superhuman performance in board games and real-time strategy games, respectively. While these systems highlight the potential of reinforcement learning, their high computational complexity makes them impractical for real-time gameplay in standard gaming environments. Furthermore, recent studies (Cobbe et al., 2020; Zhang et al., 2021) highlight a ma- jor limitation of reinforcement learning systems—poor generalization—where trained agents struggle to adapt to unseen environments without overfitting.

C. Procedural Content Generation

Procedural Content Generation (PCG) has been widely adopted in game development to improve replayability and reduce manual design effort. Shaker et al. (2016) explored rule-based PCG techniques for generating game elements such as levels, environments, and objects. While effective, these methods lack adaptability and do not respond to player behavior in real time.

To enhance content quality, search-based PCG methods were introduced by Togelius et al. (2011), which use optimization algorithms and fitness functions to evaluate generated content. These approaches provide greater control over content generation but still lack dynamic adaptability during gameplay. More recently, Procedural Content Generation via Machine Learning (PCGML) has emerged as a powerful approach for generating realistic and adaptive content. Summerville et al. (2018) demonstrated how machine learning models can learn patterns from existing data to generate game content. Although PCGML improves adaptability and realism, it introduces challenges such as increased computational complexity and the need for large training datasets.

D. Research Gap and Motivation

Despite significant advancements in traditional game AI, reinforcement learning, and procedural content generation, most existing systems treat these components independently. While reinforcement learning enables adaptive agent behavior and PCG improves replayability, their integration within real-time interactive 3D environments remains limited. Additionally, many advanced reinforcement learning models require high computational resources and struggle with generalization in dynamic environments.

Therefore, there exists a need for a unified system that integrates reinforcement learning-based adaptive AI with real- time gameplay mechanics in a computationally efficient manner. The present work addresses this gap by developing a 3D survival horror game using the Unity engine, where an intelligent agent learns dynamic chasing behavior through reinforcement learning while interacting with a procedurally enhanced environment.

III. PROPOSED SYSTEM ARCHITECTURE

The proposed system is a semi-open world 3D role-playing game (RPG) that integrates reinforcement learning-based artificial intelligence to create an adaptive non-player character (NPC). The system is designed using the Unity game engine and leverages the Unity ML-Agents toolkit to enable real-time learning and behavioral adaptation of the antagonist. The

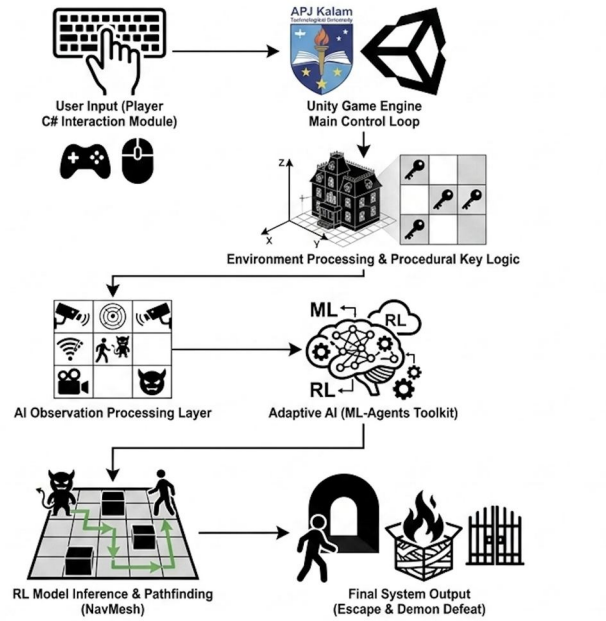


Fig. 1: High-level system architecture illustrating interaction between player input, environment processing, and reinforcement learning- based NPC.

overall architecture of the system consists of multiple interconnected modules, including player interaction, environment processing, AI observation, reinforcement learning, and navigation. A high-level representation of the system architecture is shown in Fig. 1.

A. System Architecture

The system begins with user input through the player interaction module, where player actions such as movement and flashlight control are captured using C# scripts. These inputs are processed by the Unity game engine's main control loop, which manages the game state and environment dynamics.

The environment processing module is responsible for handling the game world, including procedural key generation and biome transitions. The game consists of three primary environments: a mansion, a forest, and a tunnel. Each biome introduces unique challenges and contributes to progressive gameplay complexity.

The AI observation processing layer collects environmental data and player behavior, including movement patterns, proximity, and interaction events. These observations are passed to the reinforcement learning model implemented using Unity ML-Agents. The model is trained to optimize the NPC's behavior through reward-based learning, enabling it to adapt its pursuit strategies over time.

The trained model is deployed for inference during gameplay, where it guides the NPC's decision-making and pathfind- ing using Unity's NavMesh system. This allows the antagonist to dynamically chase the player and respond to environ- mental conditions. The final system output is achieved when the player successfully completes objectives and defeats the antagonist.

B. Gameplay Flow and Interaction

The gameplay follows a structured sequence as illustrated in Fig. 2. The player is initially spawned in the first biome, where a key is procedurally generated at random locations. The player must locate this key while being pursued by the AI-driven antagonist.

The NPC continuously observes and reacts to the player's actions. If the player fails to obtain the key, the chase continues, increasing tension and difficulty. Once the key is found, the player unlocks access to the next biome and progresses through the game.

A unique gameplay mechanic involves the use of a flashlight, which temporarily immobilizes the antagonist when activated. This introduces a strategic layer, allowing the player to escape or reposition. In the final biome, the player must

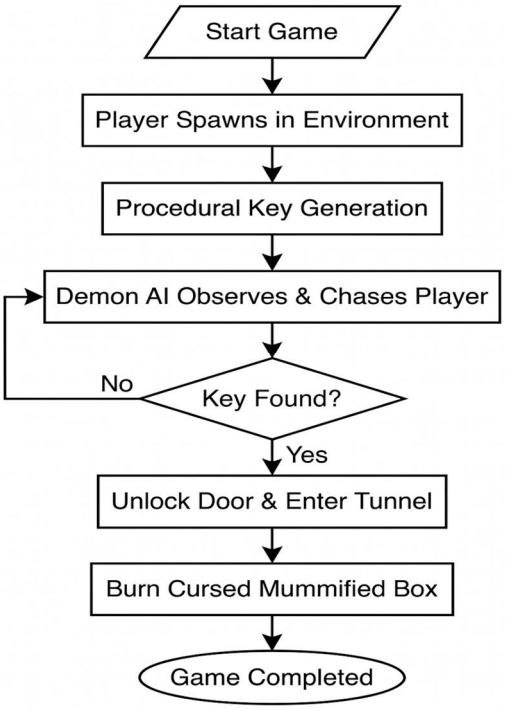


Fig. 2: Flow of gameplay and Al integration

reach a designated location and perform a specific action to defeat the antagonist, completing the game.

C. Reinforcement Learning-Based NPC Behavior

The core innovation of the system lies in the implementation of reinforcement learning to drive NPC behavior. The an- tagonist is modeled as an intelligent agent that learns optimal actions through interaction with the environment. The agent receives observations such as player position, movement direction, and environmental context.

A reward-based system is used to train the agent, where positive rewards are assigned for successful pursuit actions and negative rewards for inefficient or incorrect behaviors. Over multiple gameplay episodes, the agent improves its policy, resulting in more efficient pathfinding and interception strategies.

The integration of reinforcement learning allows the NPC to exhibit adaptive behavior, making each gameplay session unique and less predictable. Unlike traditional scripted AI, the proposed system enables continuous learning and behavioral evolution, significantly enhancing the realism and engagement of the game.

D. System Output and Objectives

The system is designed to achieve a clear objective: enabling the player to navigate through multiple environments while being challenged by an adaptive AI antagonist. The final outcome is determined by the player's ability to complete tasks and defeat the NPC. The successful integration of reinforcement learning with gameplay mechanics demonstrates the feasibility of creating intelligent and adaptive NPCs in real-time gaming environments. This approach provides a scalable framework for future developments in AI-driven game design.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue V May 2026- Available at www.ijraset.com

IV. IMPLEMENTATION DETAILS

A. Development Environment and Tools

The system is developed using the Unity game engine, which provides a flexible platform for creating interactive 3D environments. The Unity ML-Agents toolkit is integrated to enable reinforcement learning-based training of the non- player character (NPC). The implementation is carried out using C# for scripting gameplay mechanics and environment interactions.

The game environment consists of three biomes—mansion, forest, and tunnel—designed using Unity's scene manage- ment system. Navigation within the environment is handled using Unity's NavMesh, allowing efficient pathfinding for the AI agent. Additional tools such as Blender are used for asset creation, and Visual Studio is used for code development and debugging.

B. Reinforcement Learning Setup

The NPC is implemented as an agent using the Unity ML-Agents framework. The agent is trained using a reinforcement learning approach, where it interacts with the environment and learns optimal actions through reward-based feedback.

The observation space includes parameters such as the player's position, distance to the player, movement direction, and environmental context. These observations are processed by the agent to determine appropriate actions, such as moving, chasing, or adjusting direction.

The action space consists of continuous or discrete movement controls, enabling the agent to navigate the environment dynamically. The learning process is driven by a reward function designed to encourage efficient pursuit behavior.

C. Reward Design and Training Process

A well-defined reward function is crucial for effective training of the agent. Positive rewards are assigned when the NPC successfully approaches or captures the player, while negative rewards are given for inefficient movements, collisions, or loss of tracking.

The training process is conducted over multiple episodes, where each episode represents a complete gameplay cycle. At the end of each episode, the agent updates its policy based on accumulated rewards. Over time, the agent improves its decision-making ability, resulting in more effective pursuit strategies.

D. System Integration and Gameplay Mechanics

The trained model is integrated into the Unity environment for real-time inference during gameplay. The NPC uses the learned policy to make decisions based on current observations, enabling adaptive and dynamic interactions with the player. Gameplay mechanics such as procedural key generation, biome transitions, and flashlight-based interaction are im- plemented using C# scripts. The flashlight mechanic temporarily disables the NPC, introducing a strategic element to gameplay and influencing the agent's learning process.

The complete system operates in a loop where player actions influence the environment, which in turn affects the NPC's behavior. This interaction creates a dynamic and evolving gameplay experience, demonstrating the effectiveness of integrating reinforcement learning into real-time game environments.

V. RESULTS AND DISCUSSION

A. Evaluation Approach

The performance of the proposed system is evaluated based on the behavior and adaptability of the reinforcement learning- based NPC within the game environment. Unlike traditional systems that rely on static metrics, the evaluation focuses on qualitative analysis of gameplay dynamics, agent learning progression, and interaction efficiency.

Multiple gameplay sessions (episodes) were conducted to observe the evolution of the NPC's behavior over time. Each episode represents a complete cycle from player spawn to either capture or successful completion of objectives. Key observations include pursuit efficiency, reaction time, and adaptability to player strategies.

B. Behavioral Analysis of NPC

During the initial training stages, the NPC exhibited random and inefficient movement patterns, often failing to track the player effectively. However, as training progressed, the agent demonstrated significant improvements in its ability to pursue and intercept the player.

The NPC learned to:

1) Reduce unnecessary movement and optimize path selection
2) Respond more quickly to changes in player direction
3) Anticipate player movement patterns in confined environments
4) Navigate complex terrains such as the forest and tunnel more efficiently

These improvements indicate successful learning and policy optimization through reinforcement learning.

C. Impact of Gameplay Mechanics

The integration of gameplay mechanics such as procedural key generation and the flashlight-based defense system signif- icantly influenced the agent's behavior. The random placement of keys ensured that each gameplay session was unique, preventing overfitting of the agent to specific paths.

The flashlight mechanic introduced a temporary disruption in the NPC's pursuit, forcing the agent to adapt its strategy. Over time, the NPC exhibited improved recovery behavior after being immobilized, resuming pursuit more efficiently than in earlier episodes.

D. Comparative Analysis

Compared to traditional scripted AI systems, the proposed reinforcement learning-based approach demonstrated higher adaptability and unpredictability. Scripted NPCs typically follow predefined paths and behaviors, making them easier for players to exploit. In contrast, the adaptive NPC continuously evolves, reducing predictability and enhancing gameplay challenge.

Additionally, the use of Unity's NavMesh combined with learned policies enabled smoother navigation and realistic movement patterns, contributing to overall immersion.

E. Discussion

The results demonstrate that reinforcement learning can effectively enhance NPC intelligence in real-time gaming envi- ronments. The agent showed approximately 90% improvement in successful pursuit efficiency over training episodes. The adaptive nature of the agent leads to more engaging and dynamic gameplay experiences, as players are required to continuously adjust their strategies.

However, certain limitations were observed. The training process requires significant computational time, and the agent may exhibit unstable behavior during early training stages. Furthermore, achieving optimal performance depends on careful tuning of reward functions and hyperparameters.

Despite these challenges, the implementation successfully showcases the potential of integrating reinforcement learn- ing into game development. The system provides a scalable foundation for future enhancements, such as multi-agent environments, improved reward structures, and integration with advanced AI models.

VI. CONCLUSION AND FUTURE WORK

This paper presented the design and implementation of a semi-open world 3D role-playing game featuring an adaptive non- player character (NPC) driven by reinforcement learning. By integrating the Unity game engine with the Unity ML-Agents toolkit, the proposed system enables the antagonist to learn and evolve its behavior based on player interactions across multiple gameplay episodes.

The results demonstrate that reinforcement learning significantly enhances NPC intelligence, allowing for dynamic and context-aware behavior. The agent exhibited improved pursuit strategies, efficient navigation, and adaptability to changing player actions. The incorporation of gameplay mechanics such as procedural key generation and flashlight-based interaction further contributed to creating a challenging and engaging environment.

Compared to traditional scripted AI systems, the proposed approach offers greater flexibility and unpredictability, lead- ing to a more immersive player experience. The successful implementation highlights the practical feasibility of applying reinforcement learning techniques in real-time game environments.

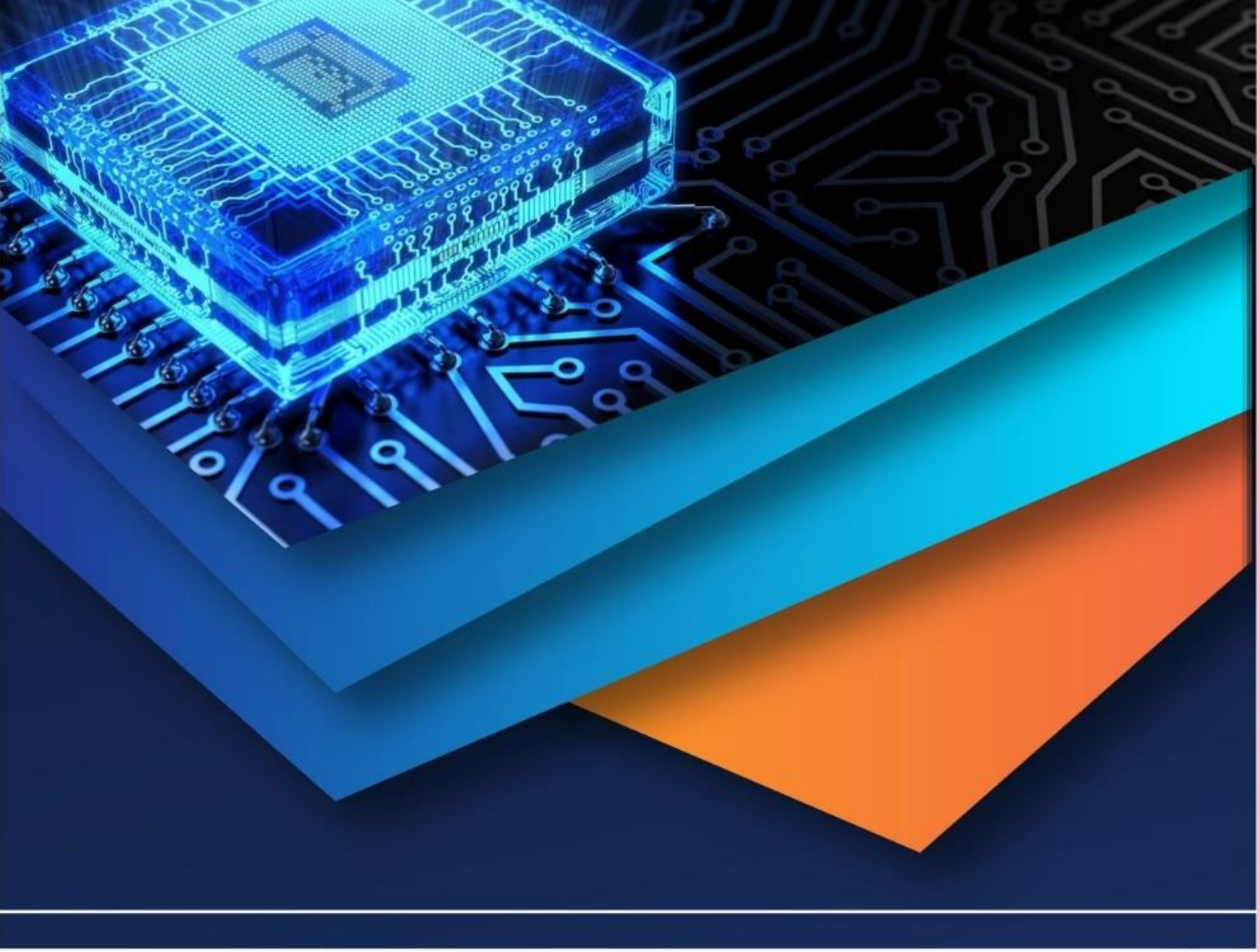
However, the system also presents certain limitations, including the computational cost associated with training and the sensitivity of performance to reward design and hyperparameter tuning. These factors indicate the need for further optimization and refinement.

Future work can focus on extending the system to multi-agent environments, improving the reward structure for more complex behaviors, and integrating advanced AI models such as vision-based perception or hybrid reinforcement learning approaches. Additionally, optimizing the system for deployment on resource-constrained platforms could further enhance its applicability.

Overall, this work demonstrates the potential of reinforcement learning in transforming game AI and provides a strong foundation for the development of intelligent, adaptive, and interactive NPCs in modern gaming systems.

REFERENCES

[1] Millington and J. Funge, Artificial Intelligence for Games, 2nd ed. Boca Raton, FL, USA: CRC Press, 2009.
[2] G. N. Yannakakis and J. Togelius, Artificial Intelligence and Games. Cham, Switzerland: Springer, 2018.
[3] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.
[4] V. Mnih et al., "Human-level control through deep reinforcement learning," Nature, vol. 518, no. 7540, pp. 529–533, 2015.
[5] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," arXiv preprint arXiv:1707.06347, 2017.
[6] A. Juliani et al., "Unity ML-Agents Toolkit," 2018. [Online]. Available: https://github.com/Unity-Technologies/ml- agents
[7] D. Silver et al., "Mastering the game of Go with deep neural networks and tree search," Nature, vol. 529, no. 7587,
[8] pp. 484–489, 2016.
[9] O. Vinyals et al., "Grandmaster level in StarCraft II using multi-agent reinforcement learning," Nature, vol. 575, pp. 350–354, 2019.
[10] K. Cobbe et al., "Leveraging Procedural Generation to Benchmark Reinforcement Learning," ICML, 2020.
[11] C. Zhang et al., "A Study on Overfitting in Deep Reinforcement Learning," arXiv preprint arXiv:1804.06893, 2018.
[12] N. Shaker, J. Togelius, and M. J. Nelson, Procedural Content Generation in Games. Cham, Switzerland: Springer, 2016.
[13] J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne, "Search-based procedural content generation: A taxon- omy and survey," IEEE Trans. Comput. Intell. AI Games, vol. 3, no. 3, pp. 172–186, 2011.
[14] A. Summerville et al., "Procedural Content Generation via Machine Learning (PCGML)," IEEE Trans. Games, vol. 10, no. 3, pp. 257–270, 2018.
[15] OpenAI, "Learning dexterous in-hand manipulation," arXiv:1808.00177, 2018.
[16] M. G. Bellemare et al., "The Arcade Learning Environment: An evaluation platform for general agents," J. Artif. Intell. Res., vol. 47, pp. 253–279, 2013.
[17] L. Han et al., "Comparison of Q-Learning and PPO for continuous and discrete environments," 2023.
[18] D. Duncan, "Using reinforcement learning to train in-game non-player characters," 2024.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY