Behavior-Driven NPC Design in the Grim Reaper Game

Shreevatsan S S[1], Yasasree Lasya Annambhotla[1], C Madhuja [1],V Srujan[1], Pooja Gowda[2]
[1]Department of Computer Science and Engineering, Amrita School of Computing, Bengaluru,
Amrita Vishwa Vidyapeetham, India
[2]Department of Computer Science and Engineering, Amrita School of Computing, Bengaluru,
Amrita Vishwa Vidyapeetham, India

***Abstract*—Artificial Intelligence (AI) and Reinforcement Learning (RL) have been significantly transforming industries over the years. Gaming is a prominent utility for its ability to create adaptive and enhancing platforms to explore RL applications. This study explores the potential of AI and RL in developing dynamic, interactive, and tactical gameplay experiences. The research has been focused on designing an AI-driven gaming system where Non-Player Characters (NPCs) have been adaptively responding to player-induced challenges and environmental dynamics. The game has been designed with the open-source game engine Godot, integrating RL techniques and the A* pathfinding algorithm for NPC navigation. The system examines the adaptive survival responses by NPCs toward customizable traps and hazards while analyzing the tactics in gameplay under variant environmental conditions. Observations have revealed that RL techniques have significantly enhanced NPC behavior, enabling intelligent responses to dynamic conditions and efficient navigation around player-set obstacles. The results have reflected the potential for AI and RL to create rich, adaptive gaming experiences and provide a basis for the analysis of how agents behave and pathfinding efficiency. Future research may centre on improving AI techniques, incorporating more complex environmental variables, and systems of multiple agents for better interactions.**

***Index Terms*—Multi-agent system, Reinforcement learning, A-star algorithm, Games**

I. Introduction

Artificial intelligence (AI) has been changing the gaming industry in many ways. It has been helping create smarter game characters that can interact with players in more engaging ways. This improvement has been making games more enjoyable and challenging. This literature survey has been looking at research on reinforcement learning (RL), which is an important area in AI that focuses on how agents have been learning from their experiences.

From the existing studies, several key points have stood out. RL has been training intelligent agents to perform well in controlled environments in [1]. It has been observed that the agents have often struggled to adapt to new and unpredictable situations [2]. This drawback has been due to the RL model's inability to gauge the dynamic environment accurately [3]. Even in games with multiple characters, deep reinforcement learning (DRL) has faced difficulties in gauging when agents have perception ability (partial observability). This emphasizes the need to handle complex interactions between agents [4] [5]. Although there has been extensive research in multi-agent systems, problems with task allocation, planning, and questions regarding scalability have arisen [6] [7]. In some scenarios, Machine Learning has been implemented to collect, assess, and transmit information between the agents. But, with a limitation of inadequate procedural validation [8]. On the other hand, a multi-layer architecture for cooperative multi-agent systems has been introduced, which has been enhanced by using a layer-cooperation model. This enhancement hasn't been able to tackle coordination challenges between the multi-agent [9].

Another important area of research has been creating agents with limited memory, which have been learning to play different games. However, many of these agents haven't been performing well against human players, suggesting that current learning methods may not have been strong enough [10]. Some studies have also been looking at using fuzzy logic with neural networks to help agents manage uncertainty. While this approach has shown some promise, the learning methods still need improvement concerning the rationality within the agents [11].

In practical applications, RL has been used in simulations, like a coffee shop environment, where real-world testing has been necessary to see how well the models work outside of theory [12]. Combining AI with educational games has shown potential for improving students' problem-solving skills, but many of these games haven't been studied enough to prove their effectiveness [13]. Advances in pathfinding algorithms have been promising, but they still need further optimization for better navigation in games [14]. Some algorithms used for zero-sum games may have trouble with situations that are unpredictable or not straightforward, as they work best with clear, structured problems [15] [16]. An attempt has been made to use dynamic programming in [17], but this has proven to be highly computational and has not been a feasible solution.

Despite these advancements, there have been important gaps in research. Many studies have been focusing on theoretical ideas without enough practical testing, which has limited our understanding of how these systems work in real life. Questions about how well these systems can scale and work in real-world situations have remained, especially regarding the use of game theory in AI training [18]. A study has been conducted and has shown that AI and ML have been becoming more

important, but many teachers haven't fully understood or used these technologies [19]. There has been a gap in making AI easy to use and relevant for everyone, which has also affected areas like game design, where smart, responsive characters have been needed [20]. We aim to address these gaps by reviewing current literature and summarizing key findings to guide future research. Improving AI in the Grim Reaper game allows us to explore learning methods and better decision-making. This research helps create adaptive AI systems that can work in complex, changing environments.

Our project has been developing a game where players act as the Grim Reaper. The main task of the Grim Reaper has been to destroy the Non-Playable Character (NPC) by setting traps and obstacles in the game environment. Using RL, we have been creating NPCs that adapt and respond to player actions. The gameplay explores whether the Grim Reaper (the player) can win against an agent trained on RL. By examining how RL has been used in games, we have been understanding both the progress made and the ongoing challenges researchers have faced in scaling RL algorithms to complex problems.

II. Methodology

A. Design Architecture

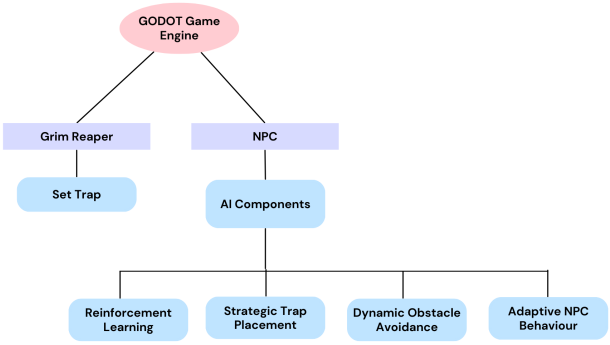

Fig. 1. Game Flow Overview

Figure 1 shows the structural hierarchy of a game created with the Godot game engine, in which the player controls a character named the Grim Reaper, whose goal has been to catch a non-playable character. The Grim Reaper can set traps deliberately positioned to capture NPCs as they go around the gaming world. The Godot game engine has been serving as the core structure for the characters, game mechanics, and player-NPC interactions.

The NPC has been equipped with several AI components that improve its responses and interactions in the game. These components include reinforcement learning, which enables the NPC to learn and adapt based on its interactions with the game environment. It can use intelligent trap installation to dodge or reduce the traps laid by the Grim Reaper. Furthermore, the NPC uses dynamic obstacle avoidance, allowing it to move around barriers in real time. To improve gameplay, the NPC has been demonstrating adaptive behavior, adapting its responses to the player's strategies over time. This layered AI approach has been making the NPC a more competitive and fascinating opponent because it learns and adapts throughout the game.

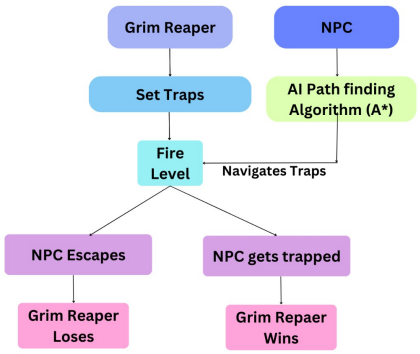

Fig. 2. Working of a level in the Grim Reaper game

Figure 2 shows the main chore of the Grim Reaper, which has been to create traps meant to either capture or confine the NPC. The NPC has thus been equipped with an artificial intelligence pathfinding algorithm—more especially, the A* (A-star) algorithm—which lets it navigate the level and try to evade the traps set by the Grim Reaper.

While the Grim Reaper sets traps in the NPC's path, the gameplay shows the NPC looking for an escape route. The A* method has been guiding the NPC into the optimum course of action free from possible traps and roadblocks. Based on the NPC's activities, the game offers two possible outcomes: the Grim Reaper loses the level if the NPC manages to evade the traps. On the other hand, the Grim Reaper wins if the NPC is caught. This presents a dynamic challenge because the player must predict the NPC's movements and modify trap placement in response, while the NPC constantly computes paths to avoid capture, therefore balancing strategy with AI-driven involvement.

B. Inputs

1) Player Inputs: The player's primary input has consisted of placing and activating traps and obstacles inside the game environment that have been intended to restrict or reroute NPCs. Each player action has modified the environment, causing recalculations in the NPCs' pathfinding algorithms. Traps have been strategically set using grid coordinates, and their impact radius and effect, such as slowing, damaging, or changing NPC behavior, have been predefined based on the trap type.

2) NPC Perceptual Inputs: NPCs have been autonomous agents that perceive their surroundings using a perception module. This module includes data inputs such as trap locations, obstacle distances, goal proximity, and dynamic hazard positioning. The perception model has been constantly updated as traps have been added or removed by the player, implying that NPCs have been judging changes in real-time. This approach has modeled perception in autonomous systems,

guaranteeing that NPCs respond dynamically to environmental changes and player strategy.

Here is your IEEE report revised to use the present-perfect continuous tense and simplified where appropriate, while maintaining the technical terms:

C. Outputs

1) Player Input Outputs:: The game has been providing the player with multi-modal input through visuals that reflect the NPCs' reactions to traps and hazards. Visual feedback, such as color-coded routes or marks indicating NPC avoidance behaviors, has been demonstrating how NPCs react to each trap. This feedback loop has been improving the player's capacity to monitor and adjust strategies in real-time.

2) NPC Behavioral Outputs: The NPCs' outputs have been represented by their movement routes, action selections, and avoidance patterns, all of which have been visible in the game environment. These results have been determined by real-time calculations of the best pathways as NPCs have been moving toward their goals while reducing danger.

D. State Space

The state space has been representing all possible gaming environment configurations at any given time, including every variable that drives NPC decision-making. The state space has included:

NPC Positional States:

Each NPC has been having a unique position inside the game's grid-based environment, and its status has been determined by a combination of geographical coordinates. By discretizing positions inside the grid, each NPC's spatial location has been linked to a unique state identifier, allowing for more exact and computationally efficient path calculations.

Trap and Hazard Configurations

Each trap in the game has been identified by its coordinates, kind, and range. These adjustments have been affecting NPCs by changing the cost function in pathfinding algorithms or imposing penalties for entering certain zones. The trap configuration state has been dynamic, changing when the player has altered trap positions, causing state recalculations in NPC navigation algorithms.

Environment and Terrain Variables:

The environment has been constructed as a multi-layered grid, with each cell representing a particular sort of terrain, for example, open paths, limited zones, or hazardous places. Each terrain type has had its own traversal cost, which has influenced the heuristic function within the A* algorithm and forced NPCs to take longer but safer courses when encountering barriers. This complicated environment has been presenting a variety of state transitions, broadening the state space when NPCs have encountered and reacted to different terrain types.

State Space Complexity:

As new traps and NPCs have been implemented, the state space has been expanding significantly, making the NPCs' pathfinding computationally intensive. The dimensionality of the state space has been capturing not just NPC placements, but also environmental configurations and player modifications, ensuring that NPC decision-making has been taking into account all environmental possibilities.

E. Action Space

The action space has been the collection of all possible actions for both NPCs and the player, outlining the range of decisions that any agent can make within the game.

Player Actions:

The player's action space has consisted of selecting, setting, and changing traps. Each action has been constrained by proximity restrictions, which have prevented traps from being too close together. Each trap type has had a unique effect on NPC behavior, such as being able to delay movement or redirect pathways, which have influenced the NPCs' calculated steps and preferred paths. The player's decisions have been instantaneously reflected in the state space, changing the routes of NPCs in reaction to each action.

NPC Actions:

NPCs have had a limited but adaptable action area that includes moving, avoiding, and interacting with objects in the environment. The A* pathfinding algorithm has been determining the NPCs' behaviors, with each advance toward the objective minimizing the estimated cost function.

NPC actions have also included hazard detection; when an NPC has detected a trap, it may have taken an avoidance action, which has caused it to recalculate paths with modified cost penalties for hazardous zones. These judgments have been updated dynamically in reaction to environmental changes, resulting in a reactive and adaptable gameplay experience.

Adaptive Action Selection:

NPCs have been using both deterministic action selection processes. While deterministic actions have followed planned courses, they have been impacting random decisions, such as choosing a different path when numerous safe routes have been available. This has allowed the NPCs to exhibit a wide range of behaviors, increasing the realism of their replies and making them less predictable to the user.

F. Actuators

Actuators have been the basic systems that allow NPCs and traps to interact in the game environment, transforming computed decisions into visible and mechanical actions.

NPC Movement Actuators:

These actuators have been controlling how NPCs move across the grid. After calculating the ideal path, the NPC actuator has been converting these calculations into step-by-step movements, providing precise spatial transitions in response to the estimated path. Movement actuators have been working perfectly with the NPCs' perception model, dynamically altering their route as new information has been received. This real-time translation of pathfinding decisions to movement has ensured that NPC behavior has remained flexible, allowing for seamless adaptability to player-driven changes in the environment.

Trap Actuators:

Each trap has had actuators that trigger specific responses when an NPC interacts with it. A trap actuator, for example, may have limited NPC mobility, inflicted time-based penalties, or activated visual effects that modify the NPC's estimated path. These trap actuators have been communicating directly with the NPC movement and decision layers, imposing extra limits on the NPC's action space. This integration has guaranteed that traps have served not just as physical barriers, but also influenced the NPC's perception of risk and safety in the environment.

Technical Implementation of Actuators:

Actuators have been built as modular code functions, each with a specific action type, allowing for a scalable system of NPC replies. For example, when a trap has been activated, the relevant actuator module has updated the NPC's state, causing pathfinding recalculations to avoid or minimize interaction with the trap. This actuator system has been critical for maintaining a responsive environment in which NPCs have responded to environmental manipulations in real-time, resulting in an engaging, AI-powered simulation that mimics autonomous agent behavior in limited and dynamic environments.

G. A* Pathfinding Algorithm

The A* (A-star) algorithm has been a pathfinding and graph traversal technique that seeks the shortest route between a start and a target node in a search area, such as a gaming environment. It has been both optimum and complete, which means it has always found the lowest-cost path if one has existed, making it perfect for real-time applications requiring computing efficiency and accuracy, such as NPC navigation in complicated game environments.

Overview of the A Algorithm:*

A* has been working by traversing nodes in a graph (or grid in a game setting), with each node representing a distinct position or state. The method has been evaluating pathways using a combination of two main components as [21]:

$$f(n) = g(n) + h(n) \tag{1}$$

where:

- **f(n)** has represented the anticipated total cost of the least expensive path across node n. It combines the costs incurred thus far with the expected remaining cost to achieve the goal.
- **g(n)** has been the exact cost from the starting node to the current node n. It has been the accumulated cost of going from the beginning point through each succeeding position till node n.
- **h(n)** has been a heuristic estimate of the cost of reaching the goal from node n. This heuristic function has been critical since it efficiently steers the algorithm's search phase to the desired outcome.

Heuristic Function:

The heuristic function $h(n)$ in A* has been crucial for evaluating the algorithm's efficiency and efficacy. A heuristic has been an estimate. In the context of A*, it should ideally have been:

1) **Admissible**: The heuristic should never overestimate the real cost of achieving the goal, so that A* has stayed optimal.
2) **Monotonic**: For any two nodes n and m, the heuristic has satisfied:

$$h(n) \leq d(n, m) + h(m) \tag{2}$$

where $d(n, m)$ has been the real cost of moving from n to m. This characteristic has promoted efficiency by assuring that once a node has been reached using the optimal path, it has not needed to be revisited.

Here have been some common heuristics:

- **Euclidean Distance**: Useful for continuous spaces when movement is unlimited, calculated as [22]:

$$h(n) = \sqrt{(x_{\text{goal}} - x_n)^2 + (y_{\text{goal}} - y_n)^2} \tag{3}$$

- **Manhattan Distance**: Common in grid-based environments with restricted movement, calculated as [23]:

$$h(n) = |x_{\text{goal}} - x_n| + |y_{\text{goal}} - y_n| \tag{4}$$

Relevance for NPC Pathfinding

In terms of NPC navigation, A* has been enabling non-player characters to efficiently navigate complicated landscapes. The program has been determining ideal paths to reduce hazards or costs based on changing game conditions.

- It has been adapting in real-time as the environment changes, such as when new traps have been added or impediments have been removed.
- It has been enabling smooth and intelligent NPC movement, making them appear more responsive and lifelike to the player.

By using A*, NPCs in the game have been balancing optimality with processing efficiency, pursuing the shortest possible path while constantly adjusting to new environmental inputs, improving gaming realism and NPC behavior.

III. Results and Discussions

Fig. 3. Difficulty levels of the Grim Reaper Game

As shown in 3, the game has been including a difficulty selection menu that offers three levels: Easy, Medium, and Hard. Each difficulty level has been varying in both the environment layout and the NPC's adaptability, which has been directly influencing the player's strategic planning. These levels have been providing a diverse range of challenges, from simpler, more predictable layouts to intricate environments where NPCs react dynamically, making the gameplay more engaging and complex.

A. Difficulty Levels and Gameplay Adjustments

1) Easy Mode: In Easy Mode, the layout has been designed to be straightforward, with only a few paths and many natural obstacles, such as stones and barriers, that limit the NPC's movement options. This setup, as illustrated in Figure 4, has been providing the player with clear opportunities for trap placement, as the confined paths have made it easy to predict NPC routes and block them effectively.

The NPC's learning rate, which has been determining its rate of adaptation to player traps, has been set to a low value in Easy Mode. This means that NPCs have had limited adaptability and have not rapidly adjusted their routes in response to obstacles or traps. Consequently, NPCs have tended to follow predictable paths with minimal avoidance behaviors, allowing the player greater control over their movement and placement of traps.

2) Medium Mode: Medium Mode has been presenting a moderately complex environment layout, with additional paths and fewer natural obstacles than Easy Mode. This increased openness, as also demonstrated in Figure 4, has been providing NPCs with more freedom to navigate, challenging the player to anticipate multiple potential routes and place traps accordingly.

The NPC's adaptation rate has been moderately increased in Medium Mode, allowing it to alter its path more effectively after encountering a trap. The NPC has been rerouting based on an enhanced awareness of hazards, making it more challenging for the player to confine the NPC to a specific area. In this mode, the player must consider a broader range of trap placements, as NPCs have started to adapt to repeated traps and develop avoidance behaviors.

3) Hard Mode: Hard Mode has been featuring the most challenging environment layout, with expansive open paths and minimal natural obstacles, giving NPCs significant freedom to move. As shown in 4, this complex layout has been allowing NPCs to explore multiple routes and avoid traps with ease, making it more difficult for the player to predict their movements.

In Hard Mode, the NPC's learning rate has been set to a high value, enabling it to quickly adapt to the player's strategies. NPCs in this mode have rapidly learned from prior encounters, changing their routes to avoid common trap placements. The player has been required to devise advanced strategies, frequently adjusting trap positions to stay ahead of the NPC's adaptive responses. Single-path blocking has become ineffective in this setting, and the player must employ a variety of traps to anticipate the NPC's learned behaviors.

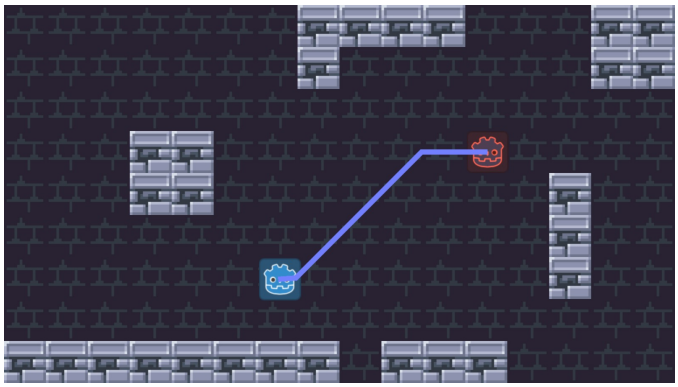

Fig. 4. View of Obstacles, Paths and NPC

B. Level Layout and Strategic Complexity

The layout complexity across difficulty levels has been significantly impacting the player's approach to trap placement. In Easy Mode, the simple layout with limited paths has allowed the player to control NPC movement with minimal traps. However, in Hard Mode, the expansive layout with multiple pathways and fewer obstacles has necessitated greater planning. With numerous potential routes, the player must anticipate and adapt to the NPC's changing paths, as shown in 4. This variation in layout complexity across levels has been adding depth to the game, challenging the player to move beyond basic strategies to more dynamic, multi-layered planning.

IV. Conclusion and Future Scope

This work uses the A* algorithm to create an adaptive simulation where players interact with NPCs. The NPCs' navigation and reactions are affected by adjustable learning rates and difficulty-specific layouts. The game has three difficulty levels which are easy, medium, and hard, that show how changes in spatial complexity and NPC behavior affect player strategy and involvement. The simulation models agent decision-making through adaptive pathfinding, with NPCs adjusting their paths based on changing costs and real-time updates. This work

shows how pathfinding and adaptive AI can create strong, interactive systems that improve strategy and complexity in controlled environments.

Future improvements could include advanced learning algorithms that allow NPCs to improve their reactions over time, making them better at responding to player actions. Adding moving obstacles would make the system more complex and realistic. These improvements could lead to further research into adaptive AI in simulations, with potential uses in self-driving technology, real-time decision-making systems, and training programs where agents need to respond to changing situations.

REFERENCES

[1] Szita, István. "Reinforcement learning in games." In Reinforcement Learning: State-of-the-art, pp. 539-577. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[2] A. C. H. Chen, "Training Strategy Based on Game Theory for Generative AI," 2024 International Conference on Smart Systems for applications in Electrical Sciences (ICSSES)

[3] Bendor, Jonathan, Dilip Mookherjee, and Debraj Ray. "Reinforcement learning in repeated interaction games." The BE Journal of Theoretical Economics 1, no. 1 (2001): 20011004.

[4] Lu, Yunlong, and Kai Yan. "Algorithms in multi-agent systems: a holistic perspective from reinforcement learning and game theory." arXiv preprint arXiv:2001.06487 (2020).

[5] Busoniu, Lucian, Robert Babuska, and Bart De Schutter. "A comprehensive survey of multiagent reinforcement learning." IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 38, no. 2 (2008): 156-172.

[6] J. Barambones, J. Cano-Benito, I. Sánchez-Rivero, R. Imbert and F. Richoux, "Multiagent Systems on Virtual Games: A Systematic Mapping Study,"

[7] Y. Zhao et al., "Winning Is Not Everything: Enhancing Game Development With Intelligent Agents," in IEEE Transactions on Games, vol. 12, no. 2, pp. 199-212, June 2020

[8] Kaswan, Kuldeep Singh, Jagjit Singh Dhatterwal, and Anupam Balyan. "Intelligent agents based integration of machine learning and case base reasoning system." In 2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), pp. 1477-1481. IEEE, 2022.

[9] K. -S. Hwang and J. -L. Lin, "A Multi-Layer Architecture for Cooperative Multi-Agent Systems," 2019 8th International Congress on Advanced Applied Informatics (IIAI-AAI)

[10] de Almeida, Lucas Antunes, and Marcelo Resende Thielo. "An Intelligent Agent Playing Generic Action Games based on Deep Reinforcement Learning with Memory Restrictions." In 2020 19th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames), pp. 29-37. IEEE, 2020.

[11] Elmanaseer, Shorouq, and Wael Alzyadat. "Intelligent Agents and Neural Fuzzy Logic: Enhancing Agent Intelligence in Complex Environments." In 2023 International Conference on Information Technology (ICIT), pp. 742-745. IEEE, 2023.

[12] Farhan, Mohammed, Brett Göhre, and Edward Junprung. "Reinforcement learning in anylogic simulation models: a guiding example using pathmind." In 2020 Winter Simulation Conference (WSC), pp. 3212-3223. IEEE, 2020.

[13] Alam, Ashraf. "A digital game based learning approach for effective curriculum transaction for teaching-learning of artificial intelligence and machine learning." In 2022 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS), pp. 69-74. IEEE, 2022.

[14] Sabri, Aimi Najwa, Nor Haizan Mohamed Radzi, and Azurah A. Samah. "A study on Bee algorithm and A* algorithm for pathfinding in games." In 2018 IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE), pp. 224-229. IEEE, 2018.

[15] L. Li and J. S. Shamma, "Efficient Strategy Computation in Zero-Sum Asymmetric Information Repeated Games", IEEE Transactions on Automatic Control, vol. 65, no. 7, pp. 2785-2800, July 2020.

[16] L. Li and J. Shamma, "Efficient computation of discounted asymmetric information zero-sum stochastic games", Proc. IEEE 54th Annu. Conf. Decis. Control, pp. 4531-4536, 2015.

[17] V. Kamble, "Games with vector payoffs: A dynamic programming approach", 2015.

[18] I. Oh, S. Rho, S. Moon, S. Son, H. Lee and J. Chung, "Creating Pro-Level AI for a Real-Time Fighting Game Using Deep Reinforcement Learning," in IEEE Transactions on Games, vol. 14, no. 2, pp. 212-220, June 2022

[19] K. Anyanwu, "Making Sense of Artificial Intelligence (AI) Machine Learning (ML) and Deep Learning (DL) in 21st Century K-12 Classrooms", Society for Information Technology Teacher Education International Conference, 2021.

[20] H. Guo and N. Li, Factors Impacting K-12 Teachers in Understanding Explanations of Machine Learning Model on Students Performance, 2020.

[21] Foead, Daniel, Alifio Ghifari, Marchel Budi Kusuma, Novita Hanafiah, and Eric Gunawan. "A systematic literature review of A* pathfinding." Procedia Computer Science 179 (2021): 507-514.

[22] Hocking, John G., and Gail S. Young. Topology. Courier Corporation, 2012.

[23] O'Regan, Gerard. Mathematics in computing. Springer International Publishing, 2020.