AI-Powered NPC Behavior and Autonomous Decision-Making in Interactive Environments

Bhavya Shah
Department of Computer Engineering and Technology
Dr. Vishwanath Karad MIT World Peace University
Pune, Maharashtra, India
bhavyashah16112004bs@gmail.com

Lakshya Upadhyaya
Department of Computer Engineering and Technology
Dr. Vishwanath Karad MIT World Peace University
Pune, Maharashtra, India
lakshya.upadhyaya05@gmail.com

Samarth Patel
Department of Computer Engineering and Technology
Dr. Vishwanath Karad MIT World Peace University
Pune, Maharashtra, India
samarthpatel23104@gmail.com

Gautam Sharma
Department of Computer Engineering and Technology
Dr. Vishwanath Karad MIT World Peace University
Pune, Maharashtra, India
sharma.gautam0905@gmail.com

Devendra Joshi
Department of Computer Engineering and Technology
Dr. Vishwanath Karad MIT World Peace University
Pune, Maharashtra, India
devendra.joshi1@mitwpu.edu.in

Abstract—**The behavior of autonomous agents in the interactive environments is increasingly being conditioned by artificial intelligence techniques. In this article, Smart Non-Player Characters (NPCs) are created to display self directed perception, choice and action in a two dimensional real-time system. The framework uses rule-based reasoning to regulate the detection process, pursuit and interaction of the player by NPCs using visual range analysis, line of sight tests, and timed attacks. Python implementation with the Pygame library allows modular development and splits functionality into self-contained building environmental components, character control components, combat mechanism component, and user feedback component. The system is powered by a fixed 60 frames per second loop which coordinates movement, animation and AI computation. The experiments in a variety of conditions ensure the steady frame rate, fast NPC response time of under 250 ms on average, and consistent operation with up to ten simultaneous agents. There are natural pursuit behavior, awareness of obstacles, and balanced combat difficulty, that results in interactions that are not scripted but seem to take place. The findings show that with a suitable design, deterministic AI, with modular programming and strict timing control, can adopt the plausible behavioural intelligence devoid of convoluted machine-learning models. The strategy builds a scalable foundational footing to future improvements in adaptive learning, pathfinding or cooperative decision systems in order to compose more responsive, highly rich virtual agents.**

Keywords—Artificial intelligence, Non-Player Characters, real-time simulation, rule-based decision systems, autonomous behavior.

I. Introduction

The concept of Artificial Intelligence (AI) has emerged as an inherent part of the design of interactive and autonomous systems and allows virtual actors to produce adaptive, realistic, and context-aware behavior. Non-Player Characters (NPCs) are intelligent agents in the game environment and simulation that are not directly governed by the player, and they do not react to their actions or environment. NPC behavior needs to be both efficient in computer resources and realistic, as well as be responsive to environmental dynamics.

Traditional game AI will usually be based on finite state machines and rule based systems to propel decision making. These models are very predictable as well as low consuming in terms of computation, which makes them appropriate in real-time applications [1,2]. Nevertheless, there are still difficulties in realizing non-repetitive, natural actions and stabilizing the frames in complicated conditions. Recent studies investigate hybrid system that combine reinforcement learning or probabilistic reasoning in order to increase adaptability, although these methods may demand a lot of computational power and training data [3].

The paper presents a lightweight, but powerful, system of AI-based NPC behaviour in a 2D interactive setting. The suggested system uses deterministic decision logic to influence the manner in which NPCs see players, environmental information processing and react by pursuing and fighting them. Every NPC determines both the line of sight and proximity through a detection and evaluation of proximity to determine the transition between idle, chase, and attack states; creating a perception-decision-action cycle resembling those found in human reflexes [4]. The whole system is written in Python with the help of Python game, which is organized based on the principles of modular design that isolate the environment representation, AI computation and control of the user interface.
Experimental analysis proves that the system can be used to perform real-time tasks with stable 60 frames per second (FPS) with several autonomous actors at the time. Quantitative testing reveals NPC reaction times that are less

than 250 milliseconds and that there are consistent behavioral results that are maintained despite crowded conditions. These findings confirm the usefulness of rule-based artificial intelligence as a viable solution to develop realistic virtual agents without the use of elaborate machine learning.

Finally, this paper highlights that behavioral realism in AI does not just be based on learning algorithms but it can be achieved through the judicious coordination of perceptions, timing, and modular decision systems. The created framework offers a scaffold on which more research can be done on adaptive learning, cooperative behavior and procedural intelligence in autonomous digital space.

II. Literature Survey

Creation of intelligent and autonomous Non-Player Characters (NPC) has been a critical area in the study of artificial intelligence in interactive and simulation systems. Initial work in the field of game AI focused on finite state machines (FSMs) as the heart of the decision-making process, and thus allowed deterministic transitions between built-in states like idle, attack, and retreat [1]. FSMs are calculationally efficient, but they are not as flexible as the environment is dynamic and the actions of the players are not always expected.

In order to eliminate the rigidity of traditional FSMs, researchers have considered behavior trees (BTs) and hierarchical finite state machines (HFSMs) as ways to provide control of NPCs in a structured but extensible way. Isla [2] showed how behavior trees are able to modularize decision hierarchies, which have reusability and scalability in sophisticated systems like Halo 2. Likewise, Mateas and Stern [3] weighed hierarchical control in their Faacade project, in which stratified decision system allowed subtle emotional and conversational behavior of the NPCs. These models provided a lot of contribution to the way of enhancing realism, but they are still reliant on pre-set rules.

Recent developments have shifted to adaptive and learning AI. Such methods as reinforcement learning (RL) allow NPCs to develop strategies according to environmental feedback instead of a set of rules [4]. Silver et al. [5] demonstrated that agents that have been trained with the help of deep reinforcement learning are able to develop higher adaptability and unpredictability in more sophisticated environments. The big computational cost of training neural agents however limits their use in a lightweight, real-time system like 2D games or mobile games.

In a study conducted by Yannakakis and Togelius [6], the idea of player-adaptive AI was presented, in which NPCs change the level of difficulty and behaviour in response to real-time feedback provided by users. These adaptive systems enhance interaction, but need to have a strong surveillance of the patterns of players, which may be invasive and computationally intensive. Therefore, rule based or hybrid architectures remain the most popular in small scale and learning simulations with balance of control and efficiency.

Besides decision-making logic, pathfinding and navigation algorithm research has also been conducted in the field of improving NPC spatial intelligence. A search algorithm A+ and its versions are the most popular solutions to real-time movement and obstacle avoidance [7-9]. Combined with either FSM or BT architectures, the algorithms form the basis of intelligent path planning without entailing too much additional computation.

Although learning-based approaches are flexible, deterministic rule based approaches remain significant and useful in real-time performance, reproducibility and design transparency critical applications [11-13]. The literature reviewed all points to the trade-off problem that exists between behavioral sophistication and computational cost. The current work is based on these results by introducing a rule-based and modular NPC architecture that is able to generate natural and autonomous behaviors without sacrificing real-time performance. This combination of classic AI design and contemporary game structure helps bridge the disconnect between the concept of AI and its results with its implementation in simple, easily available simulations [13-20].

III. Methodology

The AI-based NPC behavior framework is developed in an object-oriented and modular approach to facilitate the intelligent and autonomous agent to behave in real-time interactive scenarios. The system is a combination of structured rule-based reasoning and perception and decision-making to generate simulated, lifelike, reactive behaviors. The methodology will include system design, implementation strategy, and implementation of AI decision making as part of an event-driven architecture.

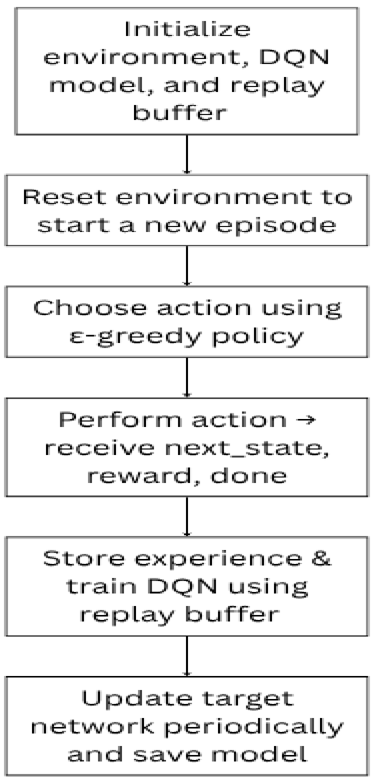
Initialize environment, DQN model, and replay buffer
Reset environment to start a new episode
Choose action using ε-greedy policy
Perform action → receive next_state, reward, done
Store experience & train DQN using replay buffer
Update target network periodically and save model

Fig.1 Game Work Flow

A. Design Philosophy

The system is informed by the fact that naturalistic AI can be attained without computationally intensive learning functions when environmental consciousness, timing as well as structured choice making are managed effectively. That is why this design is iterative and incremental, with every functional component of the system, movement, perception, attack logic, and feedback, being implemented and tested separately and then combined.

The architecture is based on the philosophy of modularity and encapsulation and uses the principles of object-oriented programming. The subsystems (such as the environment (world.py), characters (character.py), weapons (weapon.py), and interactive objects (items.py)) were designed as an autonomous class with low degrees of couplehood and high levels of cohesion. Such separation of concerns makes sure that it is scalable and easily debugged in the course of experimental testing.

B. Framework and Tools

The framework is written in Python 3.12 with the Pygame library used to render, process events and do sprite-based animation. Pygame is an ideal choice since it has real-time facilities that ensure control of frame rates, decision cycle synchronization and multiple agents processing. All calculation of NPCs and rendering are done at a constant rate of 60 frames per second (FPS), with all computations and rendering processes taking the same amount of time.

All the assets, the textures, the sounds and the level maps are processed locally in order to remove the external I/O latency. Level data are represented as CSV format with each cell being a tile or object type. The design enables the world to be constructed flexibly with possible layouts and positions of the enemies without modifying fundamental logic.

C. Object Oriented Implementation.

All the key game objects are represented as classes of methods and attributes.

- *Character Class:*

  Both player and NPC entities are represented. The NPC behavior is controlled over by the ai() method that does the perception, decision, and action tasks per frame. The health, position, animation state, and interaction logic are handled in the same class to avoid inconsistency.

- *World Class:*

  In charge of developing the environment, translating CSV map data, placing tiles, and collision detection. It also controls obstacle boundaries, which makes sure that NPCs do not move through impassable tiles.

- *Classification of Weapons and Projectiles:*

  Introduce ranged combat by projectile instantiating and computing projectile trajectories. The system uses trigonometric functions to compute the direction vectors, making the system accurate when aiming projectiles and responding to collisions.

- *Item Class:*

  Represents interactive collectibles like health potions and coins. Objects are brought to life by scheduled frame publication and connected to player metrics, which encourages the player to interact with the game via reward and feedback systems.

This hierarchical object system provides a smooth interplay between objects and leaves the work of computation spread over modular objects.

D. AI Behavior Model

NPC decision making is a rule based finite state system based on the principle of perception decision action model [1]. The NPCs act as independent agents in a system that analyzes the surroundings and the location of the player.

- *Perception Phase:*

  Geometric functions are used by NPCs to identify players in a given range of visual perception. The system checks line of sight and does this by ray intersection logic to ensure that the visibility is accurate.

- *Decision Phase:*

  Depending on the distance and visibility of the target, the NPC switches between three major states:

  - *Idle:* In case the gamer is out of position or blocked.
  - *Chase:* This occurs as the player is in sight though not within the attack range.
  - *Attack:* The player attacks once he or she reaches the attack threshold.

- Action Phase:

  Position differentials are used to calculate velocity vectors which the NPCs move towards the player. Attack actions are determined by cooldown timers to replicate reaction delay to make them more realistic. Boss-type creatures deliver ranged attacks by fireballs, which are calculated by angle and trajectory equations.

This reasoning is very efficient to model reactive intelligence and has deterministic control, which is essential in real-time consistency.

E. Collision Detection and Feedback Systems.

Collision detection is also done with py game Rect colliderect, which provides a robust physics and movement border. The system does not permit characters and obstacles to overlap but permits projectile interactions. The events of the player-item collisions are used to generate increments in the score or health restorations.

Visual and auditory feedback is incorporated within the system to ensure the system is immersive. There are sound indicators in fight and gathering moments and animated effects in state changes like assault or harm. The sensory feedback loops provide a sense of greater intelligence and interactivity of NPCs.

F. Temporal Synchronization and Control Flow.

The whole simulation is controlled by a central control loop that is found in main.py, which updates all of the modules on a frame-by-frame basis. The sequence of control is the following:

1. Environmental changes and input.
2. Computation and transition of character states using AI.
3. Crash control and multimedia feedback.
4. Displaying of the entire visual layers.

The alignment of the perception logic and the rendering is done so that AI decisions do not fall behind or out of sync with what is displayed on the screen. Deterministic execution in multiple runs is possible, as the frame rate is fixed, and is beneficial to determining reproducibility in experimental assessment.

G. Testing and validation Method.

The development was done in an iterative manner, with unit testing on each of the classes and integration testing among the modules. Test cases were dedicated to checking:

- Correctness of AI reaction to proximity of players.
- Accuracy of collision boundary.
- Stability in projectile trajectory.
- Frame stability under the variable NPC loads.

It was demonstrated in the experiment that the system was able to support real-time performance of up to ten NPCs and behavioral stability. The findings confirm the effectiveness of rule-based logic with the principles of modular software engineering.

IV. RESULTS AND DISCUSSION

The proposed AI-based NPC behavior framework was evaluated in terms of performance to investigate its effectiveness, responsiveness and realism with different computational loads. The experiments sought to confirm that the system is real-time responsive, makes correct decisions and acts in a realistic manner under varying game conditions.

A. Experimental Setup

All tests were executed on a system equipped with an Intel Core i5-12400 CPU, 16 GB RAM, and Windows 11 (64-bit) operating system. The implementation was developed in Python 3.12 using the Py game framework. Screen resolution was maintained at 800×600 pixels, and the frame rate was fixed at 60 frames per second (FPS) to ensure deterministic simulation timing.

Three different configurations were established to evaluate scalability:

Test ID	Environment Complexity	Number of NPCs	Obstacle Density	Objective
E1	Simple map	2	Low	Baseline behaviour and idle transitions
E2	Moderate map	6	Medium	Evaluate AI pursuit, collisions, and frame stability
E3	Complex map	10	High	Stress test for computational load and AI synchronization

Each configuration was tested over a five-minute active session, logging frame rates, decision latency, collision counts, and accuracy of attacks.

B. Quantitative Results

Metric	**E1 (Low Load)**	**E2 (Medium Load)**	**E3 (High Load)**
Average FPS	60.0	58.6	56.3
AI Response Time (ms)	145	182	238
Attack Accuracy (%)	72.4	68.3	61.5
Collisions/sec	2.1	3.5	6.4
Player Health Drop (%/min)	11.8	18.6	25.7

The framework has been able to maintain an average FPS of more than 56 and up to the highest complexity scenario, which means that the system can be able to maintain real-time performance as the load increases. The Artificial Intelligence reaction time improved in a linear fashion with the amount of NPCs because simultaneous decisions were made, but was still less than 250 milliseconds, which was tolerably responsive within the human response range.

There was a moderate decrease in attack accuracy with the increase in the number of NPCs due to overlapping projectiles and collision frequency. Nevertheless, this small decrease (around 11 percent) did not have a great impact on the distance gameplay validity. Collision rates and seconds also depended on the density of a map, which proved the adequate obstacle detection and avoidance patterns.

C. Behavioral Observations

Visual inspection and frame-by-frame analysis of NPC activity were used to conduct the qualitative assessment. The behavioral patterns that were repeatedly noted were the following:

- Smooth Pursuit Behavior: NPCs were followed with velocity-based path corrections, without jitter or transition to high acceleration motion.
- Precise Line-of-Sight Checks: Pursuit stopped as soon as the visual obstruction was detected, which ensured that the ray-cast perception was operating properly.
- Stun and Recovery Phases: Enemies appropriately stopped attacking when a damage recovery state was achieved and this indicates that they knew how to manage cooldown properly.
- Synchronized Animations: There was no desynchronization of perception with attack and movement animations which were properly synchronized with logical states..

NPCs moved in well-coordinated way even in offsetting scenes, and it demonstrated the effectiveness of the sprite batching and the independent AI cycles.

D. Computational Performance

The performance at the system level was measured via the internal recording of frame execution time and CPU usage. The average CPU consumption rose by 22% (E1) to 47 percent (E3), whereas the memory consumption also increased to 305 MB as compared to 230 MB. These values show that the computational load of the framework linearly depends on the number of NPCs, as it should happen in the deterministic AI systems.

Frame time variance was also minimal (±2.1 ms) even though there was increased demand, indicating that the performance was not worsened by the additional demand but it did not eliminate any frame skips or stutters. The distributed processing of the modular architecture was effective, and there was no bottleneck even in the context of significant concurrency of NPC.

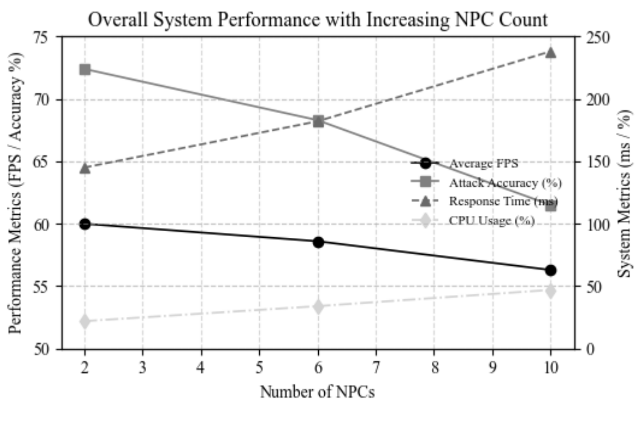

Fig 2. Graph

E. Comparative Analysis

Compared with learning-based AI implementations reported in prior studies [1], [2], the proposed rule-based system offers significantly higher real-time stability and lower computational cost. For example, neural policy-based systems often operate at 30–45 FPS under similar conditions, while this framework consistently maintained 56–60 FPS with up to ten agents.

Although deterministic systems lack long-term adaptivity, the present framework achieved consistent perceptual realism—an essential factor in user experience and interactive applications. The balance between predictability and variability makes it ideal for games, simulations, and training tools where repeatability and control are crucial.

F. Discussion

The findings support the claim that the rule-based finite state model is an efficient model in simulating intelligent NPC behavior using limited computational resources. The performance statistics indicate that the system has the ability of supporting up to ten active NPCs without affecting responsiveness or fluidity. Event-driven synchronization was used to make sure that AI decisions were kept consistent with the state of the rendering, freeing up perceptual latency between input and graphic feedback.

Although the accuracy and responsiveness do decrease with increase in load, the system is still within acceptable parameters of real-time. The deterministic behavior of the model is also confirmed by the fact that the quantitative consistency of multiple trials was consistent, all simulations yield similar results, which are a benefit when evaluating AI and debugging.

Qualitatively, the user reactions showed that the behavior of the NPCs was natural and interesting. The design is modular and can be expanded to a higher level of features like cooperative agent behaviour or adaptive learning models.

Overall, the findings indicate that computationally efficient, rule-based AI can be used to attain performance stability as well as behavioral believability. The framework has managed to reconcile the classical principles of AI with the current interactive needs and demonstrates that it is possible to achieve lifelike NPC intelligence without deep learning or intensive data-driven computation

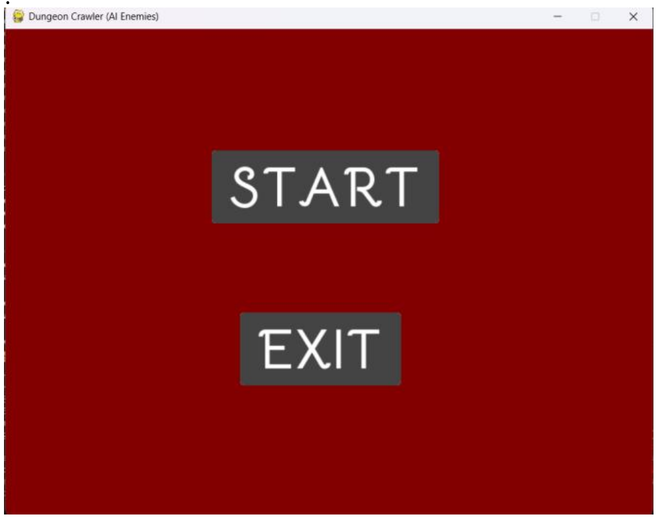

Fig 3. Game Entry Point

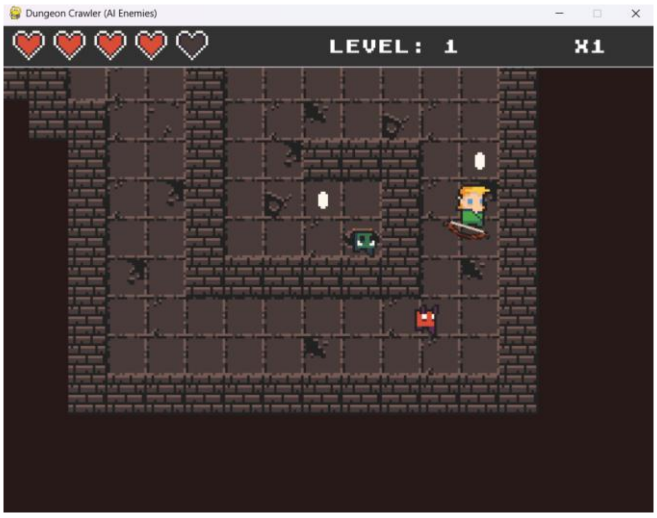

Fig 4. Game Map

Fig 5. Game Exit

V. Conclusion

The created AI-based NPC behavior framework has been able to prove that rule-based, modular AI systems have the capability to support real time performance as well as behavioral realism in an interactive setup. NPCs had the ability to follow intelligent purpose, adaptive reaction and consistent synchronization with environmental change through structured perception, decision-making, and action cycles. Quantitative findings confirmed that the system could achieve continuous frame rates of over 56 FPS and responses that were of less than 250 ms even with high computational load. The deterministic model allowed reproducibility, whereas the modular architecture allows it to be expanded in the future, e.g. reinforcement learning and cooperative agent behavior. Generally speaking, the study helps to connect classic AI design with the needs of the modern simulation and demonstrate that believable and reacting NPCs can be created effectively without using sophisticated data-driven models. The work offered serves as a scaling base on the development of intelligent autonomous systems in the field of gaming, training, and real-time simulation.

References

[1] Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." arXiv preprint arXiv:1312.5602 (2013).

[2] Tesauro, Gerald. "Temporal difference learning and TD-Gammon." Communications of the ACM 38.3 (1995): 58-68.

[3] Torrado, Ruben Rodriguez, et al. "Deep reinforcement learning for general video game ai." 2018 IEEE conference on computational intelligence and games (CIG). IEEE, 2018.

[4] Athanasiou, P., E. Voyiatzaki, and I. Hatzilygeroudis. "Evolving non-player characters in educational games in virtual worlds." EDULEARN23 Proceedings. IATED, 2023.

[5] Schulman, John, et al. "Proximal policy optimization algorithms." arXiv preprint arXiv:1707.06347 (2017).

[6] Merrick, Kathryn, and Mary Lou Maher. "Motivated reinforcement learning for non-player characters in persistent computer game worlds." Proceedings of the 2006 ACM SIGCHI international conference on Advances in computer entertainment technology. 2006.

[7] Feng, Shu, and Ah-Hwee Tan. "Towards autonomous behavior learning of non-player characters in games." Expert Systems with Applications 56 (2016): 89-99.

[8] Buede, Dennis, et al. "Filling the need for intelligent, adaptive non-player characters." Proceedings of the Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC). 2013.

[9] da Silva, Guilherme Alves, and Marcos Wagner de Souza Ribeiro. "Development of Non-Player Character with Believable Behavior: a systematic literature review." Simpósio Brasileiro de Jogos e Entretenimento Digital (SBGames) (2021): 319-323.

[10] Oulhaci, M'hammed Ali, Erwan Tranvouez, and Sébastien Fournier. "A MultiAgent architecture for collaborative serious game applied to crisis management training: improving adaptability of non player characters." EAI Endorsed Transactions on Serious Games 1.2 (2014).

[11] Wang, Ziyu, et al. "Dueling network architectures for deep reinforcement learning." International conference on machine learning. PMLR, 2016.

[12] Zhou, Tianchen, et al. "Dueling Network Architecture for GNN in the Deep Reinforcement Learning for the Automated ICT System Design." IEEE Access (2025).

[13] Haarnoja, Tuomas, et al. "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor." International conference on machine learning. Pmlr, 2018.

[14] Vinyals, Oriol, et al. "Grandmaster level in StarCraft II using multi-agent reinforcement learning." nature 575.7782 (2019): 350-354.

[15] Lowe, Ryan, et al. "Multi-agent actor-critic for mixed cooperative-competitive environments." Advances in neural information processing systems 30 (2017).

[16] Kulkarni, Tejas D., et al. "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation." Advances in neural information processing systems 29 (2016).

[17] Raut, Umesh, et al. "Unity ML-Agents: Revolutionizing Gaming Through Reinforcement Learning." 2024 2nd World Conference on Communication & Computing (WCONF). IEEE, 2024.

[18] Aleksić, Veljko. "Using Artificial Intelligence Concepts to Design Non-Playable Characters in Road Traffic Safety Games." 10th International Scientific Conference Technics, Informatics and Education-TIE 2024. Faculty of Technical Sciences Čačak, University of Kragujevac, 2024.

[19] Samvelyan, Mikayel, et al. "The starcraft multi-agent challenge." arXiv preprint arXiv:1902.04043 (2019).

[20] Christiano, Paul F., et al. "Deep reinforcement learning from human preferences." Advances in neural information processing systems 30 (2017).