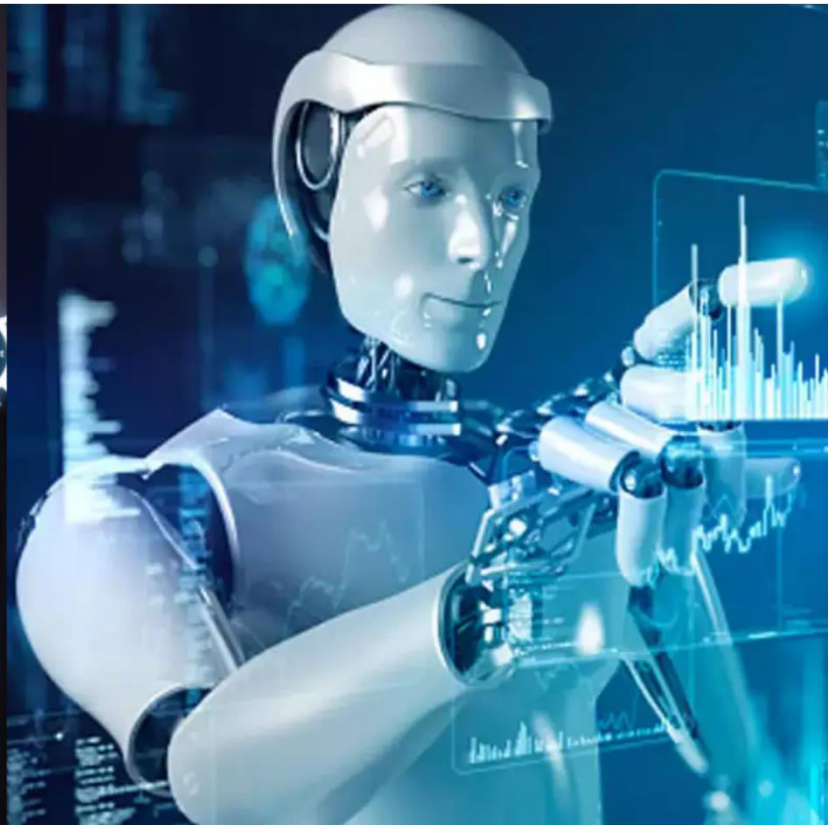
IJIRSET
International Journal of Innovative Research in Science | Engineering | Technology

ISSN(O): 2319-8753
ISSN(P): 2347-6710

International Journal of Innovative Research in Science Engineering and Technology (IJIRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 8.699

Volume 14, Issue 4, April 2025

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 14, Issue 4, April 2025

|DOI: 10.15680/IJIRSET.2025.1404173|

AI–Driven Space Shooter

T.S.Chilesh Naidu, Akula Ajith, Alla Dishwanth, Sagam Mohan, L.Shalini

Department of CSE, Bharath Institute of Higher Education and Research University, Chennai, Tamil Nadu, India

Guide, Department of CSE, Bharath Institute of Higher Education and Research University, Chennai, Tamil Nadu, India

ABSTRACT: Gaming is one of the favorite past times that is enjoyed by the people of all ages. The space themed games have always been in its vogue since its inception. This project focuses on the development of an engaging 2.5D - Space Shooter game using the Unity engine and C#. Further, the project uses Finite State Machine (FSM) algorithm for the dynamic movements of enemy spaceships. The primary objective is to develop gaming experience by integrating Artificial Intelligence (AI-driven) enemy behaviors, advanced visual effects, and immersive audio. The updated game features multiple assets, including a hero spaceship, enemy spaceships, and asteroids, providing an interactive and challenging environment. One of the key goals of the project is to address common issues found in older space shooter games, such as predictable enemy behavior and non-interesting visual effects, by introducing AI-based enemy patterns that evolve during gameplay. The game aims to provide smooth and enjoyable performance, ensuring players have an immersive experience. Improvements in AI logic, visual design, and overall game performance will be implemented to increase player engagement and overall enjoyment of the game

I. INTRODUCTION

The Space Shooter Game is an action-packed arcade-style game developed using Unity and C#. It immerses players in an exciting outer-space battle where they must control a spacecraft, eliminate enemy forces, and survive for as long as possible. The game is designed to be both engaging and skill-based, with increasing difficulty levels that challenge players' reflexes and strategic thinking. The development of this game follows a well-structured process, making it an excellent example of game design, artificial intelligence (AI) implementation, and real-time physics. This project also serves as a valuable learning experience for aspiring game developers by showcasing the practical applications of Unity's features and C# scripting.

1.1 Objective

The primary objective of this project is to develop an interactive 2D space shooter game using Unity, integrating smooth player controls and responsive shooting mechanics. The game is designed to provide an engaging experience by incorporating enemy AI behavior, ensuring that enemy movements and attack patterns increase in difficulty as the player progresses. A crucial aspect of the game is the scoring system, which encourages competitive gameplay by rewarding players based on their performance. Additionally, accurate collision detection mechanisms are implemented to ensure realistic interactions between the player's spacecraft, enemy ships, and projectiles. This project also aims to serve as an educational demonstration of game development, helping developers understand how industry-standard tools and techniques can be used to create interactive experiences.

1.2 Domain of Project

The Space Shooter Game falls within the gaming and entertainment domain, specifically in the action and arcade gaming category. It is focused on game development, artificial intelligence implementation, and interactive UI design. This project explores fundamental computer science and software engineering concepts, including game physics, which ensures smooth and realistic motion handling of objects; artificial intelligence (AI) 2 which dictates enemy behavior and increases difficulty; user interface (UI) and user experience (UX) design, which enhances the player's interaction with the game; and performance optimization techniques, ensuring smooth gameplay even when multiple objects interact on the screen simultaneously.

1.3 Aim of the Project

The primary aim of the Space Shooter Game is to develop an engaging and fast-paced arcade game that tests players' reflexes and strategic thinking. The game introduces essential game development concepts, making it an

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 14, Issue 4, April 2025

|DOI: 10.15680/IJIRSET.2025.1404173|

accessible learning platform for developers and enthusiasts interested in Unity and C# programming. Another aim is to explore Unity's capabilities in handling complex game mechanics, real-time UI updates, and AI-driven gameplay elements. The project also seeks to provide an immersive gaming experience, where players encounter increasing levels of difficulty, enemy variations, and rewards for high scores. Additionally, the development process encourages problem-solving and debugging skills, as optimizing performance, refining AI behavior, and ensuring fluid controls are integral challenges in game development.

1.4 Importance of space shooter

Game development is a rapidly growing field in the software industry, with an increasing demand for skilled developers. The Space Shooter Game serves an important role in demonstrating the core principles of game design and development. Its educational value lies in providing a step-by-step learning experience for developers who want to understand the fundamentals of C# scripting, game physics, and AI behavior. The project also offers hands-on programming experience, enabling developers to work with real-time game engines like Unity and apply industry-relevant techniques. By implementing AI-driven enemy behavior and physics-based mechanics, the game demonstrates how artificial intelligence can be used to enhance gameplay and create dynamic challenges for players. Additionally, the game is not only an educational tool but also an entertainment product, offering players an engaging experience that improves reflexes, decision-making skills, and strategic thinking.

1.5 Challenges of space shooter

The development of the Space Shooter Game came with multiple challenges that required careful problem-solving and optimization techniques. One major challenge was performance optimization, as managing multiple enemy entities simultaneously required efficient resource management to prevent lag and slowdowns. Another challenge was AI development, where creating intelligent enemy behavior that adapts to the player's actions added complexity to the game mechanics. Ensuring precise hit detection and collision accuracy was another hurdle, as slight miscalculations could lead to unfair gameplay mechanics. Smooth player controls also required balancing movement sensitivity, ensuring that the spacecraft responded naturally to user input. Lastly, game balance and difficulty scaling needed to be carefully adjusted to make the game both challenging and enjoyable, preventing it from being too easy or frustratingly difficult for players.

1.6 Problem Identification

Several core problems were identified during the initial stages of game development. One of the primary concerns was how to ensure a seamless and lag-free gaming experience, especially when multiple objects were present on the screen at once. Another key issue was designing enemy AI behavior that felt both natural and progressively challenging, rather than predictable or overly simplistic. Creating an engaging gameplay experience that retained player interest over multiple sessions was another area that required careful consideration, as repetitive mechanics can lead to boredom. Furthermore, implementing an intuitive control system was essential for ensuring that players could navigate their spacecraft smoothly and react quickly to enemy attacks without feeling frustrated by clunky controls.

1.7 Methodology

The development of the Space Shooter Game followed a structured methodology to ensure an efficient and well-organized creation process. The first step involved requirements gathering, where the core mechanics and features of the game were identified. During this phase, Unity was selected as the game engine due to its versatility and robust game development features, while C# was chosen as the programming language for scripting game logic. The next step was design and planning, where a game design document (GDD) was created to 4 outline the mechanics, UI elements, and AI behavior. The game flow, level progression, and difficulty scaling were also planned during this stage to ensure a smooth player experience. In the implementation phase, the player controller system was developed first, allowing for smooth movement and shooting mechanics. The enemy AI behavior and attack patterns were then programmed, ensuring that enemies provided an engaging challenge for players.

II. IMPLEMENTATION

2.1 Implementation

The AI Space Shooter game is developed in Unity using C#, following a structured process to ensure modularity and smooth gameplay. The project is set up in a 2.5D environment with layered sprites for all visual elements. The

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 14, Issue 4, April 2025

|DOI: 10.15680/IJIRSET.2025.1404173|

player controller uses Rigidbody2D for movement and Unity's Input System for handling controls and shooting with cooldown mechanics. Enemy AI is driven by a Finite State Machine (FSM) with states like Idle, Patrol, Chase, Attack, and Dead, reacting dynamically to the player's actions. A GameManager handles enemy waves, spawning, difficulty scaling, and game state resets. Power-ups spawn randomly and apply temporary effects like shields or enhanced firepower upon collection. The UI, built with Unity UI and TextMesh Pro, includes health bars, score, and prompts. Collision detection is managed using Unity's Physics2D, triggering damage, destruction, or pickups. Audio and particle systems enhance shooting, explosions, and effects. Dynamic camera shake and zoom add intensity to combat. This FSM-driven approach creates intelligent enemy behavior and an immersive gameplay experience.

2.2 Preparation Phase

The preparation phase lays the groundwork for AI Space Shooter development, focusing on planning, tool selection, and resource setup. It begins with requirement analysis to define gameplay mechanics and AI behavior. Unity is chosen as the game engine with C# for scripting, alongside tools like Photoshop or asset sites for visuals. The environment is set up with Unity installation, version control (e.g., Git), and project structuring. Research includes studying FSMs and Unity best practices. Storyboards and design drafts are created for game elements and UI, while a project timeline outlines milestones for each development stage.

2.3 Scalability and Maintainability

The AI Space Shooter game is designed with scalability and maintainability in mind to support future enhancements and long-term project evolution. The use of modular scripts, such as separate classes for player controls, enemy behavior, power-ups, and UI systems, ensures that new features can be added with minimal impact on existing code. The implementation of a Finite State Machine (FSM) for enemy AI allows easy modification or addition of new states and behaviors without rewriting core logic. Moreover, prefabs and scriptable objects are used for easy duplication and customization of game elements like enemies, bullets, and power-ups. Code readability and structured hierarchy in the Unity editor make maintenance efficient, allowing different developers to understand, debug, and extend the project with ease. This architecture ensures the game can grow in complexity adding new levels, AI types, weapons, or multiplayer features while keeping the core stable and manageable.

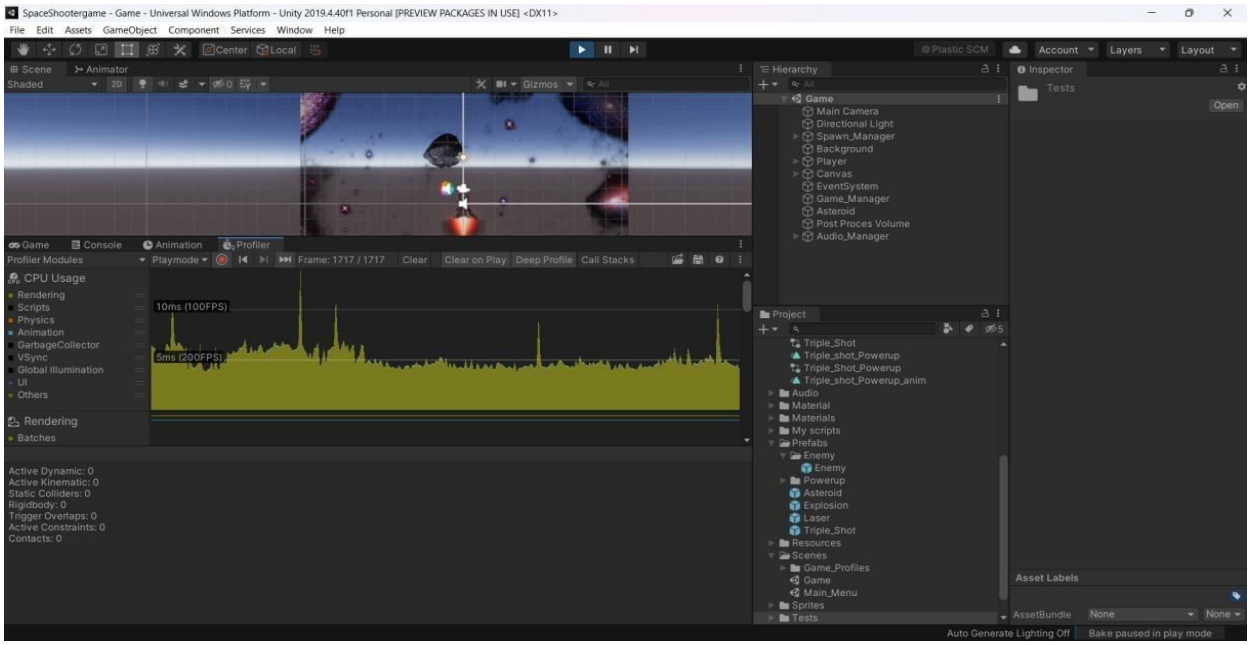
International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

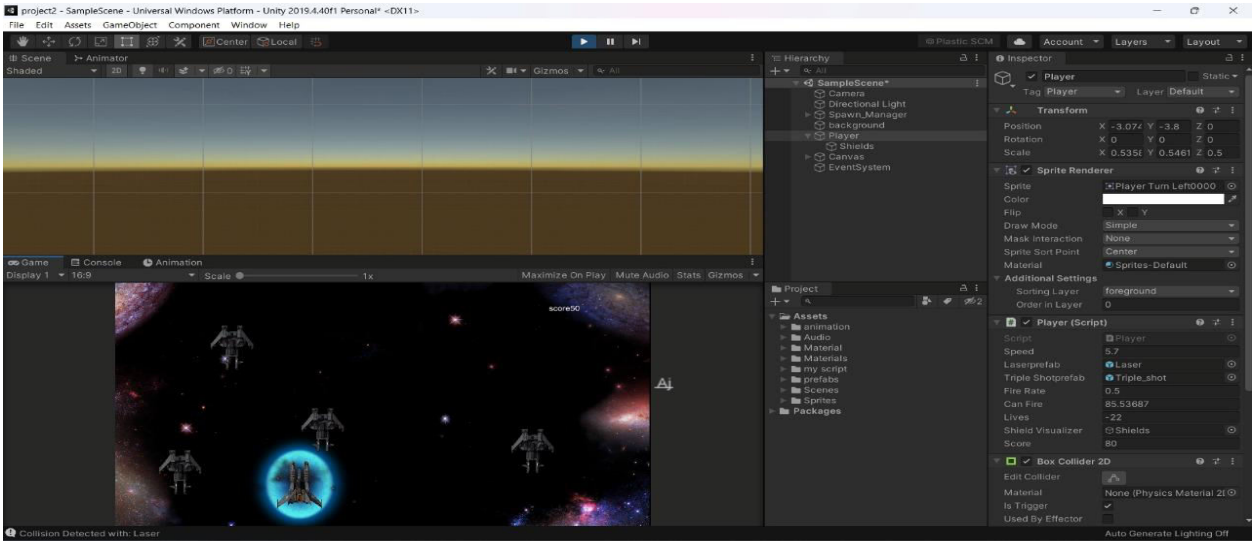
Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 14, Issue 4, April 2025

|DOI: 10.15680/IJIRSET.2025.1404173|





III. RESULTS AND DISCUSSION

The AI Space Shooter game delivers a well-structured, fast-paced space combat experience using a 2.5D environment in Unity. The implementation of a Finite State Machine (FSM) effectively controls enemy behavior, enabling intelligent state transitions such as idle, patrol, chase, and attack modes, depending on player actions and positioning. The player can smoothly navigate the game world using intuitive controls and engage with enemies through responsive shooting mechanics. Power-ups like Triple Shot and Shield work successfully, giving players temporary advantages and adding strategic depth to the gameplay. The game also includes a real-time scoring system, wave-based progression, and interactive UI elements like health bars, score counters, and restart buttons. Key features such as explosion effects, background music, and sound effects for shooting and pickups contribute to an immersive user experience. The collision system functions effectively, ensuring that bullets hit targets, and power-ups activate correctly upon pickup. The game runs smoothly with optimized performance, minimal lag, and no major bugs during testing. All modules player movement, shooting, enemy AI, power-ups, and UI were

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 14, Issue 4, April 2025

|DOI: 10.15680/IJIRSET.2025.1404173|

successfully integrated and tested. The final build supports a clean game loop, including start, gameplay, pause, and game over states.

OUTPUT

FIG: RESULT-1

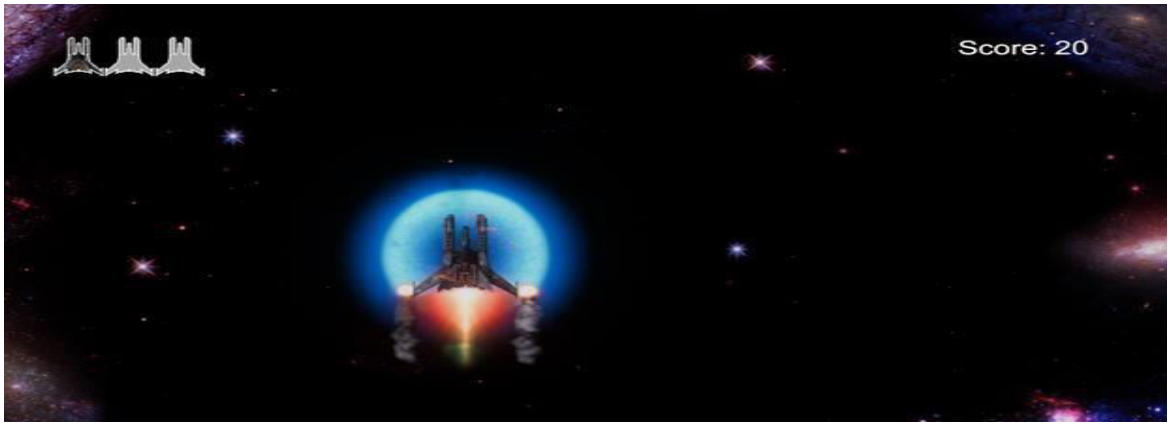


FIG: RESULT-2

FIG: RESULT-3

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 14, Issue 4, April 2025

|DOI: 10.15680/IJIRSET.2025.1404173|

Discussion :

The AI Space Shooter game showcased practical game development using Unity, with a strong focus on Finite State Machines (FSM) for AI behavior. FSM enabled enemies to react dynamically by transitioning between states like idle, patrol, chase, and attack, enhancing gameplay depth. The modular design simplified debugging and future expansion. Power-ups like Triple Shot and Shield added strategic elements and excitement. Unity's tools and physics streamlined development, though syncing AI transitions required fine-tuning. Despite early challenges, the project achieved responsive controls, engaging AI, and scalable design, proving how well AI and mechanics can merge to deliver a fun gaming experience.

IV. CONCLUSION & FUTURE WORK

CONCLUSION:

The Space Shooter game project offered valuable experience in game design, programming, and real-time physics using Unity and C#. It achieved its goals with smooth controls, dynamic AI, interactive UI, and optimized performance. Challenges like collision issues and AI balancing were overcome through techniques like object pooling and adaptive AI. Positive player feedback highlighted the game's engaging mechanics and intuitive controls. Overall, the project provided a solid foundation in game development and demonstrated the versatility of Unity and C#, making it a great learning tool for future enhancements and aspiring developers.

Future Work

The Space Shooter Game offers strong potential for future enhancements to boost engagement and replayability. Key improvements include adding multiplayer modes, advanced adaptive AI, and more diverse power-ups and weapons. Boss battles, structured levels, and procedural generation can introduce variety and challenge. Enhancing visuals, sound design, and cross-platform support would broaden its appeal. AI-based difficulty scaling can ensure balanced gameplay for all skill levels. These upgrades could transform the game into a competitive, immersive experience and lay the foundation for advanced game development and research in AI and procedural content.

REFERENCES

1. [1].Backe, H. J. (2020). Spaces of Allegory. Non-Euclidean Spatiality as a Ludo-Poetic bdevice. In Proceedings of DiGRA 2020: Play Everywhere. Digital Games Research Association (DiGRA)
2. Dewey, J. (2021). Deneyim olarak sanat. (N. Küçük, Trans.). İstanbul: Vakıf Bank Kültür Yayınları.
3. Rezapour, M.M.; Fatemi, A.; Nematbakhsh, M.A. A methodology for using players' chat content for dynamic difficulty adjustment in metaverse multiplayer games. Appl. Soft Comput. **2024**, 156, 111497. [CrossRef]
4. Amprimo, G.; Rechichi, I.; Ferraris, C.; Olmo, G. Measuring Brain Activation Patterns from Raw Single-Channel EEG during Exergaming: A Pilot Study. Electronics **2023**, 12, 623. [CrossRef]
5. Campo-Prieto, P.; Cancela-Carral, J.M.; Rodríguez-Fuentes, G. Feasibility and Effects of an Immersive Virtual Reality Exergame Program on Physical Functions in Institutionalized Older Adults: A Randomized Clinical Trial. Sensors **2022**, 22, 6742. [CrossRef] [PubMed]
6. Yang, Z.; Sun, B. Hyper-Casual Endless Game Based Dynamic Difficulty Adjustment System For Players Replay Ability. In Proceedings of the 2020 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing,
7. Fisher, N.; Kulshreshth, A.K. Exploring Dynamic Difficulty Adjustment Methods for Video Games.Virtual Worlds **2024**, 3, 230–255.
8. Foreword, J.H.; Schell, J. Unity in Action, Third Edition Multiplatform Game Development IN C#; Manning Publications Co.: New York, NY, USA, 2022.
9. Bal, H., Epema, D., de Laat, C., van Nieuwpoort, R., Romein, J., Seinstra, F., . . . Wijshoff, H. (2016). A medium-scale distributed system for computer science research: Infrastructure for the long term. Computer , 49 (5), 54– 63.
10. Soviany, P.; Ionescu, R.; Rota, P.; Sebe, N. Curriculum Learning: A Survey. arXiv **2022**, arXiv:2101.10382. [CrossRef]
11. Juliani, A. Introducing ML-Agents Toolkit v0.2: Curriculum Learning, New Environments, and More—Unity blog. 8 December 2018.Available online: https://blog.unity.com/community/introducing- ml-agents-v0-2-curriculum-learning-new environments-and-more (accessed on 20 June 2023)

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 14, Issue 4, April 2025

|DOI: 10.15680/IJIRSET.2025.1404173|

12. Unity Team. The ML-Agent's Github Page. Available online: https://github.com/Unity- Technologies/ml-agents (accessed on 20 June 2023).
13. Soviany, P.; Ionescu, R.; Rota, P.; Sebe, N. Curriculum Learning: A Survey. arXiv **2022**, arXiv:2101.10382. [CrossRef]
14. Ketkar, N.; Moolayil, J. Deep Learning with Python: Learn Best Practices of Deep Learning Models with PyTorch, 2nd ed.; Apress: Berkeley, CA, USA, 2021. [CrossRef]
15. Attanas, D.B. and Monica, H.G. (2012). Effects of green house gases, In Proc. IOOC-ECOC, pp. 557-998.Gurudeep, P.R. and Mahin, P. (2009). Risk sensitive estimation model II.
16. IEEE Transactions on Automatic Control, 43 (15): 355 - 363.
17. Prakas, K. (2011). Feedback and optimal sensitivity: Model reference transformations, multiplicative seminorms, and approximate inverses. IEEE Transactions on Automatic Control, 26(2): 301–320
18. Ram, R., Krishna, S. and Peter, K. (2005a). Risk sensitive estimation and a differential game.IEEE Transactions on Automatic Control, 39(9): 1914–1918.
19. Ram, R., Krishna, S and Peter, K. (2005b). Differential rectification using control points.IEEE Transactions on Geoscience and Remote sensing, 55: 914– 918.
20. Singh, K. and Robin, R. (2008). A linear- quadratic game approach to estimation and smoothing. In American Control Conference, New York. June 20 – 25, 2008, pp. 2818–2822

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi cross ref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details