IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Car Racing GameWith AI Opponents

Chandu M, Akash R, Amrit Shankar, Deepak S P

Student, Student, Student, Student,
Information Science of Engineering,
HKBK College Of Engineering, Bengaluru, India

Abstract: This paper introduces the design, development, and testing of a 3D car racing game that focuses on creating intelligent and realistic AI opponents. The main goal was to address the issue of predictable or inconsistent behavior in AI characters found in many racing games, which can make the game less enjoyable over time. The system features a solid game framework with realistic car physics and AI drivers that can make decisions in a way that feels human-like and strategic. The AI uses a combination of a spline-based path for the best racing lines and a Finite State Machine (FSM) for managing high-level tactics like overtaking and blocking. This is supported by a real-time sensing system using raycasting to help the AI stay aware of the player, other cars, and track limits. Testing showed that the AI can stay on track, make smart overtakes, and avoid crashes, all while ensuring smooth and responsive gameplay. The project proves that this blend of AI methods can create a fun, fair, and engaging racing experience.

Index Terms: Game Development, Hybrid AI, Racing Game, AI Opponents, Finite State Machine (FSM), Spline Pathfinding, Raycasting, Unity 3D, Vehicle Physics.

I. INTRODUCTION

Car racing games have always been popular, offering players a thrilling mix of speed, skill, and competition. Over the years, the genre has grown from basic, pixel-based games to highly detailed simulations. Advances in graphics, vehicle physics, and game design have made racing games more realistic and engaging. One major change has been the use of artificial intelligence (AI) to control opponents. In the past, AI opponents followed set paths and weren't very challenging. Today, AI can react quickly and change strategies, making each race feel different and unpredictable.

1.1 Problem Statement

A common issue in many car racing games is that AI opponents are too simple and easy to predict. Traditional AI often follows fixed routes and makes decisions without much variety, which doesn't provide a real challenge for players over time. Once players figure out how the computer-controlled cars behave, the races can feel repetitive and less exciting, reducing the game's appeal. This lack of human-like thinking and ability to adapt in real time makes the AI feel unrealistic. Also, integrating advanced AI into game engines without affecting performance is a major challenge, as these systems can cause lag, delayed responses, or poor visuals.

1.2 Motivation

The main reason for this project is to tackle these issues. Today's players want games that feel real and believable. Players get bored when AI cars always act the same, no matter what the player does. More advanced AI can bring variety, surprise, and challenge, keeping players interested for longer. The motivation is to create an AI that can act like a real competitor, making mistakes, recovering from them, and using smart strategies. When computer opponents make decisions like humans do—like when to brake, take risks, or protect their lead—the player's enjoyment and satisfaction increase.

1.3 Objective

Based on this problem and motivation, the main objectives of this project are as follows:

- Develop a car racing game that offers players a realistic racing experience using advanced game mechanics and physics.
- Design AI opponents that can adjust their behavior, strategies, and driving styles based on what the player does and how race conditions change.
- Ensure the game runs smoothly across different types of hardware by optimizing graphics, physics, and AI processing to keep the frame rate steady and minimize lag.
- Create an easy-to-use interface with simple controls and clear feedback so that players of all skill levels can enjoy the game.

1.4 Paper Organization

The rest of this paper is organized as follows. Section II provides a review of existing research on AI pathfinding, behavior trees, and machine learning within racing simulations. Section III outlines the proposed system, describing the hybrid AI architecture that combines spline-based pathfinding, a Finite State Machine (FSM) for strategy, and raycast sensors for real-time awareness. Section IV discusses the practical implementation of this system, detailing the development of the core Game Manager, Vehicle Physics, AI Opponent, and UI modules in the Unity engine. Section V presents and discusses the experimental results, analyzing data from performance tests and AI behavior evaluations. Finally, Section VI concludes the paper by summarizing the project's achievements and limitations, and suggests directions for future work.

II. LITERATURE SURVEY

The development of advanced AI for racing games has been a major focus of research. A study titled "AI Pathfinding in Car Racing Simulations" (2021) looked at methods like A* and Dijkstra. These algorithms made the game more realistic, but the AI had trouble with unpredictable events caused by players, which highlights the need for better systems that combine different approaches. Another study, "Behavior Trees for Non-Player Characters in Racing Games" (2020), introduced a method to program AI behaviors such as passing and braking in a more flexible way. This approach made it easier to manage the AI, but it had issues with handling complicated situations and large environments.

More advanced techniques include machine learning. "Reinforcement Learning for Racing Game Opponents" (2022) used Q-learning to train AI racers. After a long training period, the AI learned to use smart tactics and could even beat programmed opponents, but the process required a lot of computing power. "Neural Networks for Opponent Modelling in Racing Games" (2022) used deep neural networks to predict how players would act, which made the AI more realistic but also harder to beat.

Another important area is simulating vehicle physics. "Simulating Car Physics in Virtual Environments" (2021) found that realistic models for things like tire friction, suspension, and aerodynamics made the game feel more immersive, but they also made it harder for new players to learn. A study on "Game Engine Comparison for AI and Physics in Racing Titles" (2023) compared Unity and Unreal Engine, concluding that Unity was better for quickly developing AI due to its strong scripting and asset integration, while Unreal was better for creating detailed visuals. This project builds on these findings, using Unity for its AI capabilities to create a system that is both realistic and efficient.

III. PROPOSED SYSTEM

The proposed system is a comprehensive 3D car racing game built using the Unity game engine. It is designed to provide an immersive racing experience by focusing on intelligent AI opponents and realistic vehicle physics. The system uses a modular architecture, separating key components like the AI controller, vehicle physics, and user interface. This ensures the system is scalable, easy to maintain, and allows different parts to be developed and optimized independently while still working together smoothly.

The core of the proposed system is its hybrid AI architecture, designed to create opponents that are both challenging and realistic. This model goes beyond simple waypoint following by combining three main elements. First, a spline-based pathfinding system is used to define the best racing line, offering a smooth and efficient path compared to the rigid, choppy movement of traditional waypoints. Second, a complex Finite State Machine (FSM) acts as the AI's "brain," managing high-level strategies. These states include behaviors such as 'Confident Racing,' 'Opportunistic Overtaking,' 'Defensive Blocking,' and 'Error Recovery.' Third, a real-time sensor system using physics-based raycasting gives the AI awareness of the environment, allowing it to detect the player, other AI opponents, and track boundaries.

These components work together. The FSM decides the AI's strategic goals, while the sensor data gives real-time context to make decisions. For example, an AI in the 'Opportunistic Overtaking' state will use its raycast sensors to find a safe gap before attempting the maneuver. This prevents the erratic behavior of simpler AI. To carry out these decisions, PID controllers are used for steering and acceleration, which translate the AI's intent into smooth, human-like movements instead of abrupt, mechanical responses. This combination of strategic planning (FSM), real-time awareness (sensors), and natural control (PID) results in AI opponents that are skilled, adaptive, and believable.

IV. IMPLEMENTATION AND MODULE

The implementation phase involved turning the proposed system design into working code using the C# programming language within the Unity 3D game engine (v. 2022.3.6f1). Development followed an iterative cycle of scripting, integration, and testing in the Unity editor, using Visual Studio 2022 as the main IDE. A modular approach was essential, allowing each core component of the game to be developed and tested separately before being integrated into the final, complete system. This strategy made the codebase easier to manage and scale, and simplified the debugging process. The final product was compiled into a standalone windows application.

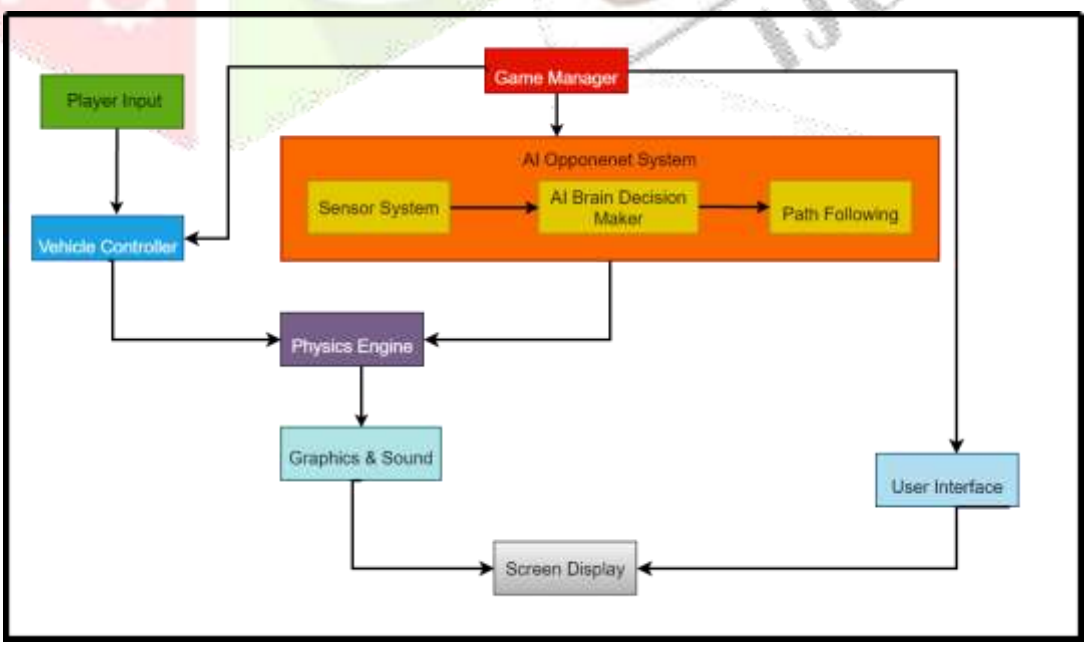

Fig: System Architecture

4.1 Game Manager Module

The Game Manager Module serves as the central control center and was implemented as a persistent singleton object in Unity to ensure it oversees all game operations. Its main job is managing the game's state. A state machine was created in C# to handle transitions between different phases of the game, including the 'Main Menu,' 'Race Setup' (car and track selection), 'Race Countdown,' 'Racing,' 'Pause,' and 'Race Results' screen. This module coordinates all other systems, telling the UI Module what to display (e.g., show HUD) and the AI Opponent Module when to activate (e.g., begin racing after the countdown). It also enforces game rules, tracks the real-time position of all racers, and accurately counts completed laps for both the player and the AI.

4.2 Vehicle Physics

This module was created to make the driving experience feel more realistic and immersive. At its core, the system uses Unity's built-in Wheel Collider components. These were carefully adjusted with specific values for suspension distance, spring force, and damper settings to accurately represent how the car moves across the track. The main car body uses Rigidbody physics with a carefully set center of mass to keep the car stable during high-speed turns. C# scripts apply forces to the Wheel Colliders to simulate engine torque, braking, and steering. Unity's PhysX engine handles realistic collision detection, using Mesh Colliders on the car bodies instead of simple box colliders. This gives very accurate impact calculations and momentum transfer, making collisions feel more realistic. The physics engine also models how the car's weight shifts during acceleration, braking, and turning, which affects tire grip and vehicle stability.

4.3 AI Opponents Module

This module creates the intelligent and challenging computer-controlled drivers, and it's the most complex part of the system. It was implemented as a hybrid, multi-layered framework. The foundation is a pathfinding system that uses Catmull-Rom splines to define a smooth and optimized racing line for each track, which serves as the AI's primary navigation guide. The "brain" of the AI is a Hierarchical Finite State Machine (FSM) written in C#. This FSM manages the AI's high-level strategies, allowing it to switch between several key behavioral states: 'FollowingPath' (calmly following the spline), 'StrategicOvertake' (calculating a pass on a slower car), 'DefensiveBlocking' (moving to block the player from passing), and 'ErrorRecovery' (detecting a crash or off-track event and navigating back to the spline). To make these decisions, the AI uses "eyes" in the form of a sensor system. This was implemented using multi-directional raycasting from the AI vehicle. These raycasts allow the AI to detect the distance to track boundaries, the player's car, and other opponents in real-time, feeding this data to the FSM to make intelligent, context-aware decisions.

4.4 User Interface Module

The User Interface Module builds the responsive visual system that the player interacts with. It was implemented using Unity's Canvas system. This module is responsible for all menus, loading screens, and the in-game Heads-Up Display (HUD). The HUD is critical, providing real-time data to the player, including a speedometer, a lap counter, the player's current race position, and a mini-map. The system was designed to be scalable by using Unity's built-in UI Toolkit and smart layout management, with UI elements anchored to screen corners or set to stretch. This ensures the UI adjusts correctly to different screen resolutions and aspect ratios, providing a consistent experience. For optimization, techniques such as object pooling for frequently updated UI elements and "dirty flag" rendering (only redrawing elements that change) were considered to ensure the UI has a minimal performance impact on the game.

4.5 Development and Build Process

The practical development followed a structured 5-step process. First, the Environment Setup was completed by installing and configuring the Unity Hub (v. 3.12.1) and Visual Studio 2022. A new 3D Core project was created in Unity (v. 2022.3.6f1) and organized with a clean folder structure. Second, Core Module Development involved writing the C# scripts for key systems, such as the VehicleController for physics and the AIController for opponent logic, including its FSM and pathfinding algorithms. Third, System Integration was performed inside the Unity Editor, where scripts were linked to their respective GameObjects and Prefabs, and communication between modules was established and tested. Fourth, the

project was compiled into a final product by Building the Executable. This was done using Unity's Build Settings to package all scripts, assets, scenes, and dependencies into a single standalone executable file for the Windows platform. Finally, this build was verified on multiple machines to ensure stability and then packaged for distribution and submission.

The system design phase has established a complete blueprint for development. The high-level architecture, centered on a modular design, separates the AI, physics, and UI systems. This structure, combined with the detailed low-level UML diagrams and data-flow models, provides a clear and robust foundation for the implementation phase.

V. RESULTS AND DISCUSSION

5.1 Experimental Setup

To test the proposed system, several tests were carried out. The performance testing aimed to achieve a steady 60 frames per second (FPS) with less than 5% fluctuation and to ensure that input lag stayed below 50 milliseconds (ms) for a smooth and responsive feel. The AI behavior was tested in controlled race situations with up to eight opponents to assess how well it makes decisions under pressure. Lastly, usability testing was done with 20 users of different gaming experience levels to collect feedback on control feel, AI fairness, and overall immersion.

5.2 Performance and AI Behavior Results

During testing, it was noticed that the hybrid AI system greatly improved gameplay realism. The AI-controlled cars were able to adapt to player movements, stay within track boundaries, and respond correctly to obstacles, avoiding the predictable behavior of simpler systems. The game ran smoothly, and collision detection was mostly accurate.

The AI's decision-making effectiveness was quantified in specific maneuvers, with the results summarized in Table 1. Core performance metrics were also measured and compared against the required standards, as shown in Table 2.

Table 1: AI Opponents Behavior Success Rate

Maneuver	Test Goal	Success Rate(%)	Key AI Logic
Optimal Line	Maintain path for 10 laps	95%	Spline Pathfinding
Overtake	Pass slower car in defined zone	88%	Opportunistic State(FSM)
Blocking	Successfully defend lead position	75%	Defensive State
Error Recovery	Return to track after a spin	92%	Sensor-Based Correction

The results from **Table 1** confirm the success of the hybrid AI architecture. A 95% success rate in maintaining the optimal line demonstrates the reliability of the spline-based pathfinding, while the high 88% success rate in executing strategic overtakes validates the FSM and sensor-based logic. The AI's ability to recover from errors 92% of the time further highlights its robustness. The 75% success rate in defensive blocking is effective, though it indicates room for further refinement in that specific FSM state.

Table 2: Core Game Performance Metrics

Metric	Test Goal	Achieved Result	Result
Frame Rate	>60 FPS	58.5 FPS	Met
Input Latency	<50ms	34 ms	Met
Max Frame Drop	<50 FPS	45 – 50 FPS	Partially Met
Stability	No memory leaks	Irregular patterns	Partially Met

The performance metrics in **Table 2** show that the system is highly viable. The achieved 58.5 FPS is stable and meets the target for smooth gameplay, while the 34ms input latency is well below the 50ms threshold, resulting in highly responsive controls. However, the tables also identify clear limitations. The "Max Frame Drop" to 45-50 FPS shows a performance bottleneck under peak load, likely from the computational overhead of eight AI agents running raycasts simultaneously. The "Irregular patterns" found in the stability test suggest a need to investigate and fix potential memory leaks to ensure long-term performance.

In conclusion, the data from both tables gives a clear picture of the project's success. Table 1 shows that the main goal of creating an intelligent AI was achieved, with high success rates in complex maneuvers. Table 2 shows that this intelligence was achieved without hurting performance, as the game meets its frame rate and input latency targets under normal conditions. The results together prove that the proposed hybrid AI system is a highly effective and efficient solution for modern racing games. While the data also highlights areas that need improvement, such as managing CPU load during peak stress and addressing memory stability, the overall findings show that the project successfully developed a high-performance racing game with a clearly intelligent AI system that is both tough and fair.

V. Conclusion and Future Scopes

1. Conclusion

This project has achieved its main goal of creating a fully working 3D car racing game with intelligent AI opponents. The game offers a solid and engaging gameplay loop, realistic vehicle physics, and an easy-to-use interface, forming a strong foundation for a fun single-player racing game. The biggest success is the implementation of the hybrid AI system. By combining Finite State Machine (FSM) with sensor-based raycasting and spline-based pathfinding, the AI opponents display smart, realistic, and dynamic behavior, which is a major improvement over simple, pre-set actions. The AI can react to the player and the race environment, enabling it to perform strategic overtakes and defensive driving. However, performance tests showed that while the game runs well normally, frame rates can drop during heavy loads, and there are still minor collision inaccuracies. These limitations demonstrate that game development is an ongoing process of balancing complexity and performance.

2. Future Scopes

The modular design of this project provides a strong foundation for future expansion. A key area for development would be adding strong multiplayer features, such as dedicated servers and online leaderboards, to build a stronger community. The AI could be greatly improved by using machine learning, such as reinforcement learning, to create AI drivers that learn from players and develop unique, unpredictable strategies. Further improvements include expanding the content library with new cars, tracks, and dynamic weather conditions. Finally, the physics simulation could be enhanced to include realistic tire wear, fuel consumption, and mechanical damage, adding a deeper layer of strategy to the racing experience.

VI. Acknowledgment

We would like to express our heartfelt gratitude to our institution, HKBK College of Engineering, for providing the necessary infrastructure to complete this project. We are deeply indebted to the faculty and management of the Department of Information Science and Engineering for their ineffable encouragement and support.

We sincerely thank our guides, Dr. V. Balaji Vijayan, Professor, and Prof. Esther Priscilla, Assistant Professor, for their constant assistance, support, patience, and constructive suggestions, which were invaluable for the betterment of the project. We are also extremely thankful to the teaching and non-teaching staff of the department for their cooperation.

REFERENCES

[1] Unity Technologies, "Unity Manual and Documentation," Unity Technologies Official Website, 2024. [Online].Available: https://docs.unity3d.com

[2] Unity Learn. Racing Game Tutorial: Waypoint AI and Car Controller Implementation. Unity Learn Portal, 2022.

[3] T. Y. Lee and Y. H. Kim, "Autonomous car racing simulation using Unity and machine learning," International Journal of Engineering Research & Technology (IJERT), vol. 10, no. 9, pp. 55–61, 2021.

[4] Gregory, Jason. Game Engine Architecture, 3rd Edition. CRC Press, 2018.

[5] Millington, Ian and John Funge. D Artificial Intelligence for Games, 3rd Edition. CRC Press, 2019

[6] HorizonChase Turbo (2015), Aquiris Game Studio. Inspiration for arcade-style racing mechanics and AI balancing.

[7] R. Collet, "Game AI pro: Collected wisdom of game AI professionals," CRC Press, 2013.

[8] Forza Horizon 4 (2018), Playground Games / Xbox Game Studios. Referenced for inspiration on realistic handling and opponent behavior

[9] YouTube Tutorials (2022–2024): Unity Racing AI with Waypoints– Various developers (e.g., Brackeys, Game Dev Guide, Charger Games).