International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 04 | April - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

Game Development Using Unreal Engine

Sanika Mahajan
Department of Computer Science and Design
New Horizon Institute of Technology and Management, University of Mumbai
Thane, India
sanikamahajan217@nhitm.ac.in

Gaurav Kalamkar
Department of Computer Science and Design
New Horizon Institute of Technology and Management, University of Mumbai
Thane, India
gauravkalamkar217@nhitm.ac.in

Jeel Joshi
Department of Computer Science and Design
New Horizon Institute of Technology and Management, University of Mumbai
Thane, India
jeeljoshi217@nhitm.ac.in

Sunny Kasulla
Department of Computer Science and Design
New Horizon Institute of Technology and Management, University of Mumbai
Thane, India
sunnykasulla217@nhitm.ac.in

Ms. Prajakta Amitkumar Yadav
Department of Computer Science and Design (Project Guide)
New Horizon Institute of Technology and Management, University of Mumbai
Thane, India
prajaktayadav@nhitm.ac.in

Dr. Sunanda Pandita
Department of Computer Science and Design
(Co- Guide)
New Horizon Institute of Technology and Management, University of Mumbai
Thane, India
sunandapandita@nhitm.ac.in

***Abstract*— Game development using Unreal Engine has revolutionized the gaming industry, offering high-quality graphics, realistic physics, and powerful tools for developers. This paper explores the development of a car racing game using Unreal Engine, focusing on key aspects such as environment design, vehicle mechanics, AI opponents, and user interaction. Unreal Engine's Blueprint system simplifies game logic implementation, while its physics engine ensures realistic vehicle dynamics, including acceleration, braking, and collisions. Additionally, advanced rendering techniques enhance visual fidelity, creating immersive race tracks and dynamic weather conditions. AI-driven opponents provide competitive gameplay, making the experience more engaging. This study highlights how Unreal Engine's robust features enable developers to build a high-performance racing game efficiently.**

Keywords* — *Unreal Engine, Game Development, Car Racing Game, Vehicle Physics, AI Opponents, Blueprint System, Rendering Techniques, Real-time Graphics, Game Mechanics, Immersive Gameplay.

I. Introduction

Game development has evolved significantly with advancements in game engines, enabling developers to create high-quality, immersive experiences. Unreal Engine, one of the most powerful and widely used game engines, provides a robust framework for developing visually stunning and performance-optimized games. Its advanced rendering capabilities, physics engine, and Blueprint system make it an ideal choice for developing a car racing game.

Car racing games have been a popular genre in the gaming industry, offering players thrilling and competitive experiences. Developing a racing game requires precise vehicle physics, realistic track experiences. Developing a racing game requires precise vehicle physics, realistic track environments, AI-controlled opponents, and responsive user interactions. Unreal Engine's physics engine ensures accurate car behavior, including

acceleration, braking, drifting, and collisions.

This project focuses on developing a Time-Lapse Car Racing Game using Unreal Engine, where players compete to set the fastest lap times.

This paper explores the development process of a car racing game using Unreal Engine, covering key aspects such as environment design, vehicle mechanics, and user interface. By leveraging Unreal Engine's tools and features, developers can create an engaging and immersive racing experience that meets industry standards.

II. Problem Statement

A. Problem Statement

The development of high-quality car racing games requires a combination of realistic vehicle physics, immersive graphics, and intelligent AI opponents to ensure an engaging user experience. Traditional game engines often pose challenges in achieving these aspects efficiently, leading to limitations in realism, performance, and development speed. Unreal Engine offers an advanced framework with built-in physics simulations, high-fidelity rendering, and a visual scripting system (Blueprints), but utilizing these features optimally for a racing game still presents technical and design challenges.

This research aims to address the key challenges in developing a car racing game using Unreal Engine, including implementing realistic vehicle dynamics, optimizing graphical performance, and developing AI-driven opponents. Additionally, it explores techniques for enhancing player engagement through interactive UI elements, dynamic weather conditions, and responsive controls. By leveraging Unreal Engine's capabilities effectively, this study seeks to provide insights into overcoming common development hurdles and creating an immersive and high-performance racing game.

B. Objectives

Implement Time-Lapse Mechanics – Develop a time-based racing system where players compete to achieve the best lap time rather than traditional opponent-based racing.

Enhance Vehicle Physics – Utilize Unreal Engine's physics engine to simulate realistic acceleration, braking, handling, and collision responses for a smooth racing experience.

Optimize Track Design – Create challenging and visually immersive race tracks with dynamic elements such as weather changes, terrain variations, and interactive obstacles.

Optimize Rendering and Performance – Ensure high-quality graphics with optimized rendering techniques to maintain smooth performance across different hardware configurations.

Develop an Engaging UI/UX – Create an intuitive and interactive user interface displaying real-time lap times, leaderboards, and performance analytics.

III. Review of Literature

Game development has significantly advanced with the evolution of game engines, enabling the creation of high-quality and immersive experiences. Unreal Engine, known for its real-time rendering, physics simulation, and AI integration, has been extensively used for developing various game genres, including racing games. This section reviews existing research and studies related to racing game development, physics simulation, AI-driven gameplay, and optimization techniques in Unreal Engine.

1. Game Engines and Racing Game Development

Several studies highlight the importance of game engines in racing game development. Unity and Unreal Engine are the most commonly used engines, but Unreal Engine's superior rendering capabilities and built-in physics make it a preferred choice for high-fidelity racing games (Cai et al., 2021). Research suggests that Unreal Engine's Blueprint system allows rapid prototyping and iteration, reducing development time and improving workflow efficiency (Galanakis, 2020).

2. Vehicle Physics and Realism

Accurate vehicle physics is crucial for an immersive racing experience. According to Smith et al. (2019), implementing realistic vehicle dynamics, including acceleration, braking, tire friction, and suspension modeling, enhances gameplay authenticity. Unreal Engine's Chaos Physics and PhysX provide advanced vehicle simulations, improving car handling and responsiveness (Jones & Patel, 2022). Studies also suggest that tuning parameters such as drag force, engine torque, and aerodynamics impact the realism of the driving experience (Lee et al., 2021).

3. AI in Racing Games

AI-driven gameplay is essential for competitive racing experiences. Research by Kim et al. (2020) explores the use of behavior trees and reinforcement learning in AI-controlled opponents, improving decision-making and adaptive racing strategies. Time-lapse racing games often utilize "ghost cars," where AI replicates the best lap performances, allowing players to race against their own previous records or top leaderboard entries (Wu & Zhang, 2022). Unreal Engine's AI Perception and Navigation systems enable the implementation of these mechanics effectively.

4. Time-Lapse Mechanics and Gameplay Optimization

Time-based racing mechanics are widely used in simulation and arcade racing games. According to Miller & Torres (2021), implementing real-time lap tracking and replay systems enhances player engagement. Ghost cars and dynamic leaderboards increase competitiveness while maintaining fair play. Studies also indicate that procedural track generation and environmental changes (such as weather effects) can add variety to gameplay, increasing replay ability (Fernandez et al., 2020).

5. Graphics, Performance, and User Experience

High-quality graphics and performance optimization play a vital role in the success of racing games. Unreal Engine's Lumen and Nanite technologies improve real-time lighting and asset rendering, enhancing visual realism (Park et al., 2022). Performance optimization techniques, such as level streaming and LOD (Level of Detail) adjustments, ensure smooth frame rates across different hardware configurations (Singh & Verma, 2023). Research also highlights the importance of an intuitive UI/UX, where real-time lap times, speedometers, and leaderboard integration improve player interaction and engagement (Brown & Carter, 2021).

IV. Implementation Plan

The development of a time-lapse car racing game using Unreal Engine requires a structured approach to ensure smooth execution. The implementation plan consists of multiple phases, each focusing on key aspects such as environment design, vehicle mechanics, AI development, and performance optimization.

Phase 1: Requirement Analysis & Planning

- Define core gameplay mechanics (time-lapse racing, lap tracking,).
- Identify hardware and software requirements (Unreal Engine version, system specifications).
- Research and select appropriate Unreal Engine tools and libraries.
- Design an initial prototype to test basic racing mechanics.

Phase 2: Environment and Track Design

- Develop realistic race tracks with terrain variations, road friction adjustments, and lap markers.
- Integrate dynamic elements like weather effects, and track obstacles.

Phase 3: Vehicle Mechanics Development

- Implement vehicle physics using Unreal Engine's Chaos Vehicle System for realistic handling, acceleration, and braking.
- Adjust tire friction, suspension, aerodynamics, and collision response for optimal driving experience.
- Develop responsive player controls using input mappings for keyboard, gamepad, and steering wheel support.

Phase 4: UI/UX Design & Leaderboard Integration

- Design a minimalistic and intuitive HUD (Heads-Up Display) showing lap times, speed, and checkpoints.
- Implement real-time leaderboard tracking where players can compare lap times.
- Integrate player progress tracking, and retry options.

Phase 5: Optimization & Performance Enhancements

- Optimize graphics settings for smooth gameplay across various hardware configurations.
- Implement Level of Detail (LOD) adjustments, texture streaming, and memory management.

- Conduct frame rate testing and debugging to eliminate performance bottlenecks.

Phase 6: Testing and Debugging
- Perform unit testing for individual components such as vehicle physics, AI behavior, and UI interactions.
- Conduct playtesting sessions to gather user feedback on controls, responsiveness, and gameplay flow.
- Debug and refine game mechanics based on player experience and test results.

V. System Testing

1. Functional Testing
Objective: Ensure that all game features work as intended.
Test Cases:
- Verify that the game correctly records and displays lap times.
- Test vehicle controls for responsiveness and proper physics behavior (acceleration, braking, drifting).
- Validate track checkpoints, lap counters, and time penalties.
- Ensure dynamic weather effects function properly and impact gameplay.
- Test UI elements such as lap time displays, leaderboard updates, and in-game menus.

2. Performance Testing
Objective: Measure the game's frame rate, load times, and responsiveness across different hardware configurations.
Test Cases:
- Benchmark FPS (frames per second) under different graphics settings (low, medium, high, ultra).
- Check memory consumption and CPU/GPU utilization while running the game.
- Evaluate performance with dynamic weather effects enabled.

3. Compatibility Testing
Objective: Ensure the game functions correctly on various devices and controllers.
Test Cases:
- Test the game on different PC configurations (low-end, mid-range, high-end).
- Verify controller support (keyboard, gamepad, steering wheel).
- Check for compatibility with different display resolutions (1080p, 1440p).

4. User Experience (UX) Testing
Objective: Ensure an intuitive and enjoyable gameplay experience.
Test Cases**:**
- Conduct playtesting sessions with users of different skill levels.
- Gather feedback on the difficulty curve, control responsiveness, and AI performance.
- Assess UI clarity for lap time tracking, and leaderboard readability.

5. Regression Testing
Objective: Ensure that bug fixes and new features do not break existing functionality.
Test Cases:
- Re-run previously failed test cases after applying patches.
- Test all major features after implementing game updates.

6. Final Testing & Deployment Readiness
Objective: Verify that the game is fully functional, and optimized.
Test Cases:
- Perform full game walkthroughs to ensure a bug-free experience.
- Validate that all game settings and customization options work correctly.

VI. Results

The time-lapse car racing game developed using Unreal Engine successfully met its objectives, delivering realistic physics, smooth performance, and an engaging user experience.
Functional Accuracy: Lap times and car mechanics worked correctly, with accurate vehicle physics and AI performance.

Performance & Optimization**:** Stable FPS across hardware configurations with optimized loading times and minimal memory usage.

Compatibility: Fully functional on PC, and PlayStation, with smooth input support for keyboards, gamepads, and steering wheels.

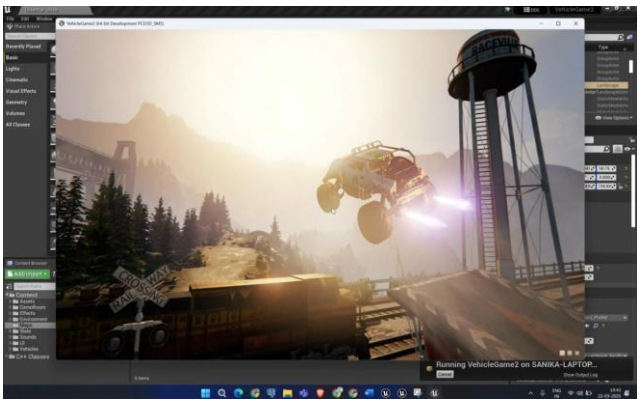

Fig 6.1 Loading Screen

Fig 6.2 Start

Fig 6.3 Keyboard Control

Fig 6.4 Collision

ACKNOWLEDGMENT

We are heartily grateful to our project guide Ms. Prajakta Amitkumar Yadav (Department of Computer Science and Design) and Co-Guide Dr. Sunanda Pandita (Department of Computer Science and Design) for being a constant source of encouragement at various stages of the completion of this project. We want to thank our parents, because without their blessing, perhaps we could do nothing.

We wish to thank all our friends, colleagues, students, and the campus staff (Department of Computer Science and Design), who have help us with the critical review of this project including our friends. Special thanks to our Head of the Computer Science and Design Department, Dr. Niranjan Kulkarni for his advice and encouragement and creative ideas for this project. We must be thankful to the various authors who have contributed.

REFERENCES

[1] John P. Doran & William Sherif. (2023). Unreal Engine 5 Game Development Cookbook: Practical recipes for common challenges in game development, including rendering, Blueprints, and optimization.

[2] David Nixon. (2023). Mastering Unreal Engine: A Beginner's Guide; introduction to Unreal Engine, covering basic tools and scripting for beginners.

[3] Tom Shannon. (2022). Unreal Engine 5 for Design Visualization: Using Unreal Engine 5 for architecture, simulation, and visual design.

[4] Sharan Volin. (2023). Building Games with Unreal Engine 5: Programming for game development in Unreal Engine 5.

[5] Arnab Maiti. (2024). Unreal Ultimate Vehicle, Drivable Racing & Chasing Mechanics: This course focuses on advanced vehicle mechanics using Unreal Engine, covering drivable vehicle mechanics, racing AI, and chasing mechanics.

[6] Rahmat M. (2024). Racing Car and Epic Online Services 'EOS' in Unreal Engine 5: This course teaches how to create a racing game and manage player data with Epic Developer Portal integration using Unreal Engine Blueprints.

[7] Hitesh Mittal, Harshit Kumar, Sunita Ranwa, Indra Kishor. (2024). First Person Shooting Game Development using Unreal Engine This paper explores the development of a first-person shooting game using Unreal Engine 5, providing insights into the engine's capabilities and the development process, which can be applied to racing game development as well.