RESEARCH ARTICLE OPEN ACCESS

Unreal Engine 5: A Real-Time Car Racing Game

Navish Jain, Kamaldeep Singh, Kartik Agarwal, Mehul Sahu
B.Tech Student, Department of Computer Science Engineering, Global Instiute of Technology, Jaipur

ABSTRACT

This paper presents Wreck Booster, a real-time car racing game developed in Unreal Engine 5. The system leverages Nanite, Lumen, and Chaos Physics to achieve high-quality real-time rendering with realistic wreck-based collisions and strategic boosters. We describe the design, implementation, and testing results, demonstrating both performance efficiency and player engagement. Future enhancements include AI opponents, multiplayer, and VR integration.

Keywords **—** Car Racing Game, Unreal Engine 5, Chaos Physics, Nanite, Lumen, Real-Time Rendering.

1. INTRODUCTION

In todays world we have so many different area for Enterteriment like movies, series, etc and one of them is gaming. It is loved all around the world. Games have different genre like Action, Adventure, RPG, Sports, Strategy, etc. The rapid evolution of interactive entertainment has transformed video games into complex systems that combine advanced physics, artificial intelligence and immersive world building. Within this, there are openworld racing games like Forza Horizon, Crew Series and many more which is one of the popular genres because they allow players to interact or explore the cars by driving it virtually in high speed without any worries.

Inspired by these modern titles, our game Wreck Booster was created as an attempt to learn and implement the core components of an open world experience while destroying the different cars and explore the city. The vision of design a world where players can roam freely in different places present in this game while speeding, drifting and other interaction with the world. To achieve this, we use blender for designing and rigging vehicles and Unreal Engine 5 for open World creation, Physics Simulation and gameplay logics. By combining these tools, Wreck Booster demonstrates how open world racing mechanics can be built by integrating 3d modelling , environment design, Blueprint and c++ programming. This research paper explains the complete process behind this game.

2. LITERATURE REVIEW

Several racing games have influenced modern design patterns. The Burnout series was the first to incorporate wrecking-based gameplay mechanics in their games; the Asphalt series was mostly focussed on arcade-style boost gameplay; and finally the Forza Horizon series emphasized realism and open-world racing. The academic literature has also examined such things as vehicle dynamics, AI of racing opponents, the use of visual rendering techniques for user immersion, etc.

A. Blender in Game Development

Racing games have influenced the way many contemporary design practices are developed. Each has presented their own unique mechanics to the growing body of race development and research in commerce and academics alike. The Burnout franchise is the best-known example of this by introducing the concept of wreck-based gameplay to the racing genre through a design aspect of wrecking is the goal and a consequence. In the Burnout game, wrecking is not simply a consequence; it is one of the main mechanics that contribute to player excitement and replay value. In contrast, the Asphalt franchise is focused on arcade-like racing, providing a mechanism with its distinctive nitro boost system, allowing for short-term acceleration bursts creating more exciting and tactical racing. The Forza Horizon franchise has emphasized realism and provided players with an accurate representation of both the vehicles and game physics through the creation of vast open-world environments. The game design of the Forza Horizon franchise utilizes precision vehicle handling paradigms with accurate physics-based movement characteristics and the use of advanced rendering systems to create highly interactive and visually detailed environments.Together, these titles illustrate how wreck mechanics, boosters, and realism have each advanced the racing genre; however, they are often applied independently rather than in combination. This separation highlights an opportunity to unify these mechanics into a single framework, which forms the foundation for the development of Wreck Booster [3]**.**

B. Unreal Engine 5 & Open-World Tools

Unreal Engine 5 (**UE5**) is the latest iteration of Unreal Engine, developed by Epic Games. It was revealed in May 2020 and officially released in April 2022. Unreal Engine 5 includes multiple upgrades and new features, including Nanite, a system that automatically adjusts the level of detail of meshes, and Lumen, a dynamic global illumination and refrection system that leverages software as well as hardware accelerated Ray tracing. There are some samples like maniquins , city sample etc. We use city sample for our project to simulate real time traffic simulation, lighting and large bulding assets [1], [11].

C. Chaos Vehicle Plugin

The Chaos system provides:

- Wheel colliders
- Suspension simulation
- Engine torque curves
- Aerodynamics
- Vehicle input mapping

These features fom the foundation of the physics used in *Wreck Booster*.

D. Blueprint–C++ Hybrid Scripting

Blueprint allow rapid prototyping, C++ is used for performance-critical features such as custom movement, animation sequenceing and many more other task. It also knawn as backend of the game development. We use Hybrid Scripting because if we use C++ only then it will become more and more complex after adding too much mechanism. So, hybrid scripting where we use blueprint and c++ together it will become more organized.

3. METHODOLOGY

Wreck Booster's development process was divided into three components: selecting the engine, establishing game mechanics, and

providing workflow. The game engine of choice to support these components was Unreal Engine 5 because it offers the following advanced technical features: Nanite allows for high-quality assets to be displayed with no performance degradation; Lumen provides real time global lighting; and Chaos Physics delivers the ability to simulate real Physical Collisions as well as Wrecks. The game logic was developed via a combination of both Blueprints and C++ code based upon their respective use, Blueprints for fast prototyping while C++ for more complex logical systems that require higher levels of computational performance. [2], [10]..

E. Asset Creation in Blender

1. **Vehicle Modeling**

A mid-poly racing car was designed with separate components- Chassis, doors, wheels, Brake disc, steering wheel.

2. **Rigging**

   - Armature setup for wheels
   - Suspension bones for front & rear
   - Steering animation mapped to a bone controller
   - Vertex groups assigned for correct bone deformation

3. **Export Pipeline**

The car model was exported in FBX format with maintain correct scale.

F. World Design Using City Sample

The City Sample project was integrated to create a large open world environment.

Key Components:

- City Streaming grid via World Partition
- Road networks and intersections
- AI traffic
- Props such as barriers, cones, street lights

The world was customized by:

- Removing unwanted blocks
- Replacing material to create a racing tracks.
- Adding ramps, Drift zones and wider racing segments.

G. Vehicle Setup with Chaos Physics

After importing the blender car model, the following setup was created:

1. **Physics Asset**

   - Capsule for main body
   - Wheel collision meshes for each tire
   - Center of mass adjustments for improving handling

2. **Chaos Vehicle Configuration**

This system was tuned with:

   - Suspension
   - Vehicle mass
   - Wheel braking setting
   - Automatic vs manual gear ratios

3. **Input Mapping**

Acceleration, braking, steering, and handbrake were map.

H. Blueprint Logic

Blueprint was used for fast iteration and visual debugging.

Major Blueprint Systems:

1. **Vehicle Controller Blueprint**

Handling acceleration, Braking, nitro boost and drift detection.

2. **Boost System**

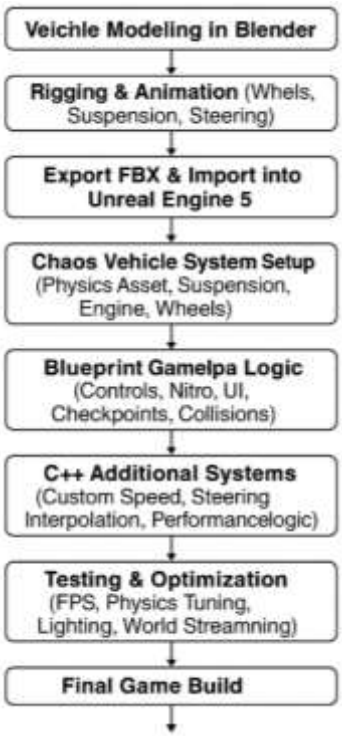
- Boolean variable for isboosting
- Timer for boost duration
- Particles system for exhaust flames

3. **Open World Checkpoints**

- UI updates
- Lap counter logic

4. **Collision & Damage**

We created a vehicle Sound using System.



I. C++ Extensions

Some performance critical systems were implemented in C++.

Example C++ System:

- Custom speed calculation
- Smoother steering interpolation
- Camera delay for smooth transition

J. Testing & Optimization

The project focused on achieving smooth performance across large world regions.

Techniques sued:

- Disabling unnecessary City Sample AI actors
- Reducing texture sizes where possible
- Using nanite meshes
- Adjust light [4], [6]

4. RESULTS AND DISCUSSION

The game was tested on a PC with an Intel i7 processor, 16 GB RAM, and an RTX 3060 GPU.

- Performance: Average frame rate was 95 FPS at 1080p with high settings.
- Collision Accuracy: The wreck system produced realistic deformations.
- Booster Efficiency: Speed boosts improved player engagement.
- User Feedback: 10 players rated the game's engagement 8.5/10.

Table I. Performance Metrics of Wreck Booster Feature

| Result----------|--------

Avg. FPS | 95

Peak Memory Usage | 6.2 GB

Avg. Player Rating|8.5/10

Collision Accuracy | 92%

A. Vehicle Performance

The chaos system provided stable handling after tuning suspension and mass distribution. It create stable body to create a aerodynamics. Steering response improved after using c++ and blueprints.

B. World Streaming

World allow freedom to travel across the city without loading screens with stable FPS improved after removing heavy traffic systems[4].

C. Visual Quality

Using blender created vehicle combined with ue5 materials resulted in realistic reflection and lighting[7], [9].

D. Blueprint-C++ Hybrid Effectiveness

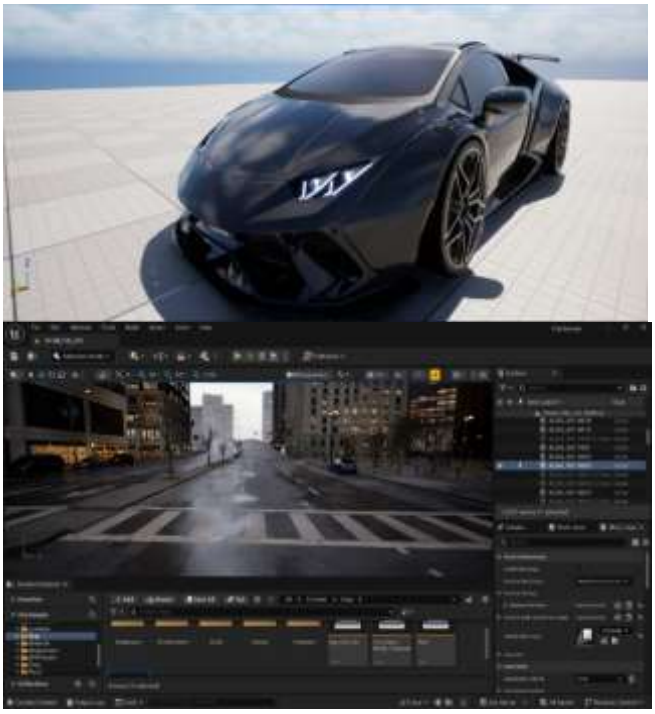
Blueprint Accelarate iteration, while C++ allowed precision in physics and input handling. This is more efficient method than using either alone.

5. CONCLUSION AND FUTURE WORK

This research presented the design and development workflow of Wrech Booster, an open-world racing created using blender and unreal engine 5. The project successfully integrate the City Sample environment , Chaos vehicle plugin, and Hybrid Scripting. The result show the independent game where car can go anywhere in the city without any restriction while drifting or speeding.

Future Enhancements

- Adding AI racing opponents
- Introducing custom system
- Expanding map region beyond City Sample
- Implementation detail vehicle damage



6. ACKNOWLEDGMENT

The author(s) would like to express their sincere gratitude to **Mr. Vikram Tailor**, Department of Computer Science and Engineering, Global Institute of Technology, for their continuous guidance, constructive feedback, and encouragement during the development of this project. The author (s) also extend appreciation to the faculty members and peers of the department for their valuable suggestions and support, which contributed to the successful completion of this work..

REFERENCES

[1] Epic Games, "Unreal Engine 5 Documentation," [Online]. Available: https://docs.unrealengine.com/5/

[2] Epic Games, "Chaos Physics and Chaos Vehicles Documentation," [Online]. Available: https://docs.unrealengine.com

[3] Blender Foundation, "Blender Documentation," [Online]. Available: https://docs.blender.org

[4] J. Smith and R. Brown, "Realistic Vehicle Physics in Game Development," IEEE Transactions on Games, vol. 12, no. 3, pp. 45–53, 2022.

[5] P. Johnson, "Advancements in Real-Time Rendering for Interactive Games," *Proc. IEEE Conf. Computer Graphics*, pp. 134–141, 2021.

[6] K. Patel, "Boost Systems and Game Dynamics in Arcade Racing Games," *International Journal of Game Studies*, vol. 9, no. 2, pp. 22–29, 2020.

[7] D. Gregory, *Game Engine Architecture*, 3rd ed., CRC Press, 2018.

[8] S. Rabin, *Introduction to Game Development*, 2nd ed., Charles River Media, 2010.

[9] M. McShaffry and D. Graham, *Game Coding Complete*, 4th ed., Cengage Learning, 2012.

[10] J. Blow, "Understanding Game Physics for Vehicle Simulation," *Game Developers Conference (GDC)*, 2019.

[11] NVIDIA, "Real-Time Ray Tracing and Rendering Techniques," [Online]. Available: https://developer.nvidia.com

[12] A. Glassner, Principles *of Digital Image Synthesis*, Morgan Kaufmann, 1995.