Journal of Information Systems Engineering and Management
2025, 10(16s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Interactive 3D Vehicle Simulation: Modeling Dynamic Road Interactions and Environmental Factors

Spriha Deshpande
Santa Clara, USA
spriha.deshpande@gmail.com

ARTICLE INFO

Received: 03 Dec 2024

Revised: 26 Jan 2025

Accepted: 08 Feb 2025

ABSTRACT

This project presents a 3D vehicle simulation using the Panda3D engine, incorporating physics-based interaction for vehicle movement, collision detection, and environment creation under varying road conditions. The simulation includes a dynamically generated curved road, straight road, and harsh weather conditions, such as rain or fog, to create a realistic and immersive driving environment. The player's car can be controlled via keyboard input, with an opposing vehicle moving along an opposite lane. The simulation uses the Bullet physics engine for real-time collision detection between vehicles, logging collisions and vehicle interactions. The project demonstrates the application of physics engines and environmental factors in creating an interactive and realistic driving experience

Keywords: 3D vehicle simulation, Panda3D engine, physics-based interaction, vehicle movement, collision detection, environment creation, road conditions, curved road, straight road, harsh weather conditions, rain, fog, immersive driving environment, keyboard input, opposing vehicle, Bullet physics engine, real-time collision detection, vehicle interactions, physics engines, environmental factors, interactive driving experience.

I. INTRODUCTION

Vehicle simulation plays a critical role in understanding driving dynamics, road conditions, and vehicle interaction in diverse environments [1]. This project uses the Panda3D engine to build an immersive 3D simulation that includes various driving conditions, such as curved and straight roads, and introduces weather conditions to challenge vehicle control [2,5]. The simulation uses the Bullet physics engine for accurate collision detection and physics-based movement[3,8].

The simulation's core features include:

- Curved Roads: The road bends dynamically, offering challenges to the player as they navigate through sharper turns [4,6].
- Straight Roads: Some parts of the simulation feature straight paths, allowing the player to accelerate freely [17].
- Harsh Weather Conditions: To add complexity, harsh weather conditions like rain and fog affect vehicle handling and visibility [9,12].

The player controls a vehicle through these varying conditions, with the simulation focusing on realistic movement, collision detection, and real-time feedback [7,18]. This project also demonstrates the integration of environmental factors such as weather effects, which further enhance the driving experience [10,14].

A. Formula Usage:

1) Speed Calculation (Linear Velocity Magnitude): The speed of an object is a scalar quantity that represents how fast it is moving. In this code, the speed is calculated using the magnitude (or length) of the velocity vector. The velocity vector provides both the direction and the rate of motion of the object in 3D space. The formula for the magnitude of the velocity vector is:

$$speed = |velocity\ vector| \qquad (1)$$

This means that the speed is the Euclidean distance from the origin (0, 0, 0) to the point defined by the velocity vector. In the context of the car simulation, this gives us the instantaneous speed of the car at any given moment.

2) Distance Traveled: The distance traveled by an object is the product of its speed and the time it has been moving. In the simulation, this is calculated incrementally by multiplying the speed (which is derived from the car's velocity) by the time delta (dt) for each frame update:

$$distance = speed \times time \qquad (2)$$

This formula assumes that the car moves at a constant speed for each time step (dt). Over time, as the car moves and its speed changes, the total distance is accumulated, representing how far the car has traveled from its starting point.

3) Road Curvature (X Position of Road): The road is designed to curve as the car moves, and the curvature is achieved by applying a mathematical sine function to the road's x-position. The formula used for generating the x-coordinate of each point on the road is:

$$x(t) = 15 \times \sin(math.radians(t)) \qquad (3)$$

The sin(t) function ensures that the road will curve in a smooth, oscillating manner as the car moves along the road's length. By multiplying the sine function by a factor of 15, the amplitude of the curve is increased, making the curve sharper. The road is thus not straight but rather follows a sinusoidal path as defined by this equation.

4) Forces Applied for Car Movement: The car's movement is driven by a force that is applied along its forward direction. The force is calculated based on the user's input (whether the "up" or "down" key is pressed to move forward or backward):

$$F_{drive} = motor\_force \times direction \qquad (4)$$

The *motor_force* represents a constant value that determines how much force is exerted on the car, and *direction* is a value of either +1 (for moving forward) or -1 (for moving backward). This force is applied to the car's rigid body in the game engine, causing it to accelerate or decelerate depending on the input.

5) Movement of the Opposite Car: The opposite car moves along a path that mirrors the road's curvature but in the opposite direction. The formula for the opposite car's movement along the x-axis is:

$$x(t) = -6 + 10 \times \sin(math.radians(t)) \qquad (5)$$

Like the road curvature, the sine function is used to generate a smooth, oscillating movement. The factor of 10 controls the width of the curve, while the constant -6 positions the opposite car slightly to the left of the road center. This ensures that the opposite car follows a curved path on the left side of the road.

6) Vehicle Position Update (Continuous Movement): In the simulation, both the car and the opposite vehicle update their positions continuously during the game loop. The vehicle's position is updated based on the physics engine's calculations, including the applied forces and velocity. The position is also affected by user input (turning left or right, or accelerating or decelerating), which continuously adjusts the car's location in the 3D space. The update happens in every frame, meaning the vehicle's position is recalculated based on the current time step (dt).

7) Additional Concepts:

- Collision Detection: In this simulation, collisions between the player's car and the opposite car are detected using collision volumes (such as spheres or boxes). These volumes are defined around each vehicle, and the game engine checks if they overlap or intersect during each frame. If a collision occurs, an event is triggered that logs or processes the collision. This is crucial for detecting when vehicles collide and responding to those events (e.g., by logging a message or triggering a response in the simulation).
- Camera Control: The camera is programmed to follow the player's car as it moves along the road. By updating the camera's position in each frame, the view remains focused on the car, providing an

immersive experience. The camera typically follows the car from behind or above, giving the player a consistent view of the simulation as they navigate the environment.

These formulas and concepts work together to simulate realistic vehicle movement, road curvature, and collisions in a 3D environment, creating a dynamic simulation experience.

II. METHODOLOGY

The methodology of the project follows a structured approach to creating a realistic 3D vehicle simulation under varying road and weather conditions. The process involves creating the road, vehicle models, integrating physics simulations, handling weather conditions, and providing continuous vehicle movement and collision detection. Below is the detailed methodology:

A. Setting Up the Environment:

- Panda3D Setup: The simulation is built using the Panda3D engine, which supports the integration of physics engines, collision detection, and real-time rendering. The Bullet physics engine is used to handle vehicle dynamics and interactions with the environment.
- Road Generation:
    - Curved Road: The curved road is generated using a series of points that define a sharp path. The road curves are made more dynamic by adjusting the amplitude of the sine wave used for the X-coordinate to simulate sharper turns.
    - Straight Road: For parts of the simulation, straight roads are created by modifying the Y-coordinate while keeping the X-coordinate constant. This allows the player to experience stretches of the road with less turning.
    - The road is created using Panda3D's CardMaker class, which generates the road as a flat plane. A road texture is then applied to the plane, and the road's curvature is adjusted based on the generated points.
- Weather Conditions: Harsh weather conditions like rain and fog are added to the environment to influence driving behavior and visibility. The simulation changes the weather dynamically, increasing difficulty and altering vehicle control. Weather effects are created using particle systems and visual fog effects that reduce the player's visibility.

B. Vehicle Simulation:

- Vehicle Models: Simple box-shaped vehicles are used for both the player's car and the opposing vehicle. These vehicles are created using Bullet physics' BulletBoxShape, which allows for precise collision detection.
- Physics Simulation: The simulation utilizes the Bullet physics engine for vehicle movement. The player controls the car with keyboard input, where forces are applied based on the desired direction and speed. This creates a physics-driven simulation where real-world vehicle dynamics such as acceleration, friction, and turning are modeled.
- Opposing Vehicle Movement: An opposing vehicle is introduced that moves in the opposite direction, simulating traffic. This vehicle's movement is controlled similarly to the player's car but with its path continuously updated based on the road's curvature.

C. Collision Detection:

- Collision Volumes: Collision volumes are defined for both the player's car and the opposing vehicle using CollisionSphere objects. These volumes allow the physics engine to detect when vehicles collide with each other.
- Collision Events: Panda3D's CollisionHandlerEvent is used to detect and handle collision events. When a collision is detected, the event triggers a message that logs the collision with the respective vehicle.

- Logging: Every time a collision occurs, a log message is printed to the console indicating that the vehicles have collided. For real-time feedback, the speed and total distance traveled by the player's vehicle are also logged at regular intervals.

D. Weather-Related Challenges:

- Slippery Roads: During rain, the road's friction is reduced, making the car harder to control. This simulates the challenge of driving on wet roads.
- Reduced Visibility: During fog or rain, the visibility is reduced, making it more challenging for the player to navigate through the road. This is implemented using particle systems and fog effects, which obscure distant objects and create a more immersive environment.
- Handling Changes: The player's car handling is adjusted based on weather conditions, making it more difficult to navigate sharp turns and accelerate during harsh weather.

E. Real-Time Updates:

- Vehicle Movement: The player's car moves along the road based on keyboard inputs. The car accelerates, decelerates, and turns based on key presses, with forces being applied to simulate realistic movement.
- Weather Interaction: As the weather conditions change, the simulation dynamically alters vehicle control and visibility. For example, in heavy rain, the car becomes harder to steer, and the camera may zoom out to provide a wider view of the surroundings.
- Logging Data: The simulation tracks the vehicle's speed and total distance traveled. This data is logged every second, providing real-time metrics for the user. Additionally, collision events are logged with details about the involved vehicles.

F. Camera and Feedback:

- The camera follows the player's car, providing a third-person view of the driving environment. The camera adjusts its position to keep the vehicle in view as it moves along the road, ensuring that the player can effectively control the car.
- Real-time feedback, including the speed and distance of the vehicle, is displayed on the console. Collision logs provide information about any interactions with the opposing vehicle, assisting the player in understanding how their actions impact the simulation.

The methodology described above creates a fully functional and immersive vehicle simulation where the player can experience realistic driving dynamics, navigate curved and straight roads, and deal with challenging weather conditions. The integration of Bullet physics for collision detection, combined with Panda3D's 3D rendering capabilities, provides a rich environment for testing and visualizing vehicle behavior under various road and weather conditions. This project is scalable, allowing for further expansion in the areas of advanced vehicle dynamics, more complex environmental factors, and additional features like traffic or road hazards.

III. ARCHITECTURE

The below *Fig 1*. Represents outlines a structured process for the vehicle simulation project.

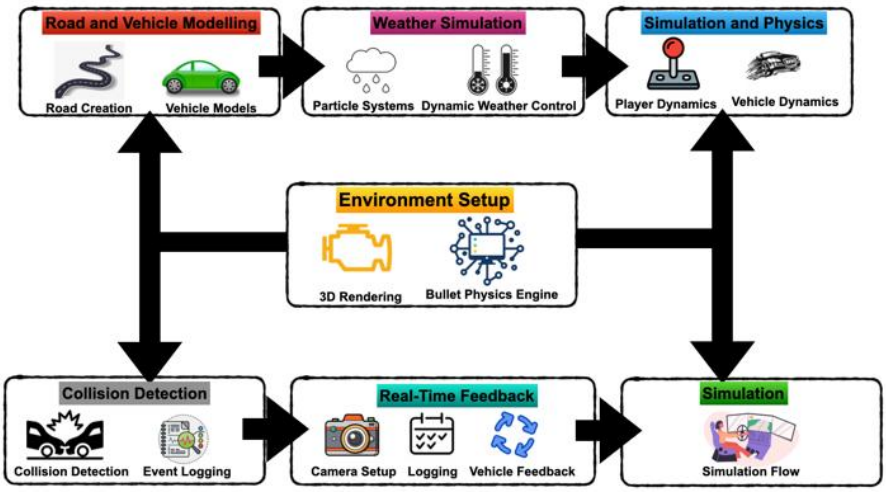

Figure. 1. Vehicle Dynamics Simulation Flow

The below is a detailed explanation of each component and how they connect within the simulation flow.

A. Road and Vehicle Modeling

- Road Creation:
  - This step involves generating the road in the simulation environment. There are two types of roads in the system:
    - Curved Road: Using a mathematical function (e.g., sine wave), the road path curves are created. The amplitude of the sine wave adjusts the sharpness of the turns.
    - Straight Road: A simpler part of the road where the path is straight. The X-coordinate remains constant, and the Y-coordinate changes to simulate stretches of road.
- Vehicle Models:
  - Simple box-shaped vehicle models are used for both the player's car and the opposing vehicle. These models are created with Bullet physics' BulletBoxShape, enabling realistic collision detection.

Flow: The road and vehicle models are generated, defining the initial environment where the vehicle simulation takes place. The vehicle models interact with the road based on physics simulations.

B. Weather Simulation

- Particle Systems:
  - Used to create environmental effects like rain and fog. These effects impact vehicle control and visibility, making the simulation more challenging and realistic.
- Dynamic Weather Control:
  - The weather conditions change in real-time during the simulation. This could involve transitioning from clear skies to rainy weather, affecting the road's friction and the player's visibility.

Flow: This step adds dynamic environmental factors, such as varying road conditions due to rain or fog, that directly affect the player's ability to control the vehicle.

C. Environment Setup

- 3D Rendering:
  - Powered by the Panda3D Engine, this is responsible for the visual aspects of the simulation, including rendering the road, vehicles, trees, mountains, and other environmental features. It makes the simulation immersive.

- Bullet Physics Engine:
    - The core physics engine that handles realistic vehicle movement, including acceleration, friction, and turning. It also simulates interactions like collisions between vehicles and the road.

Flow: The environment setup ensures that all visual and physical elements work together to create a dynamic, responsive simulation. The Panda3D engine handles rendering, while Bullet Physics manages the vehicle's physics interactions.

D. Simulation and Physics

- Player Dynamics:
    - This is the input layer where the player controls the vehicle using keyboard inputs. It influences the vehicle's acceleration, deceleration, and turning behavior, based on physics.
- Vehicle Dynamics:
    - This step models the physics of vehicle movement, including how the car accelerates, turns, and reacts to external factors like road curvature and weather conditions.

Flow: This stage ties together the physical dynamics of the player's car and integrates them into the simulation. It directly influences how the car moves and responds in the 3D environment.

E. Collision Detection

- Collision Detection:
    - The system constantly checks for interactions between the player's vehicle and the opposing vehicle. This is done through CollisionVolume objects like CollisionSphere. The physics engine detects when vehicles overlap or collide.
- Event Logging:
    - When a collision is detected, the event is logged, including which vehicles were involved. Additionally, the vehicle’s speed and distance traveled are also tracked for real-time feedback.

Flow: The collision detection system ensures that the simulation responds to physical interactions, such as a crash or near-miss, between the vehicles. This is critical for providing feedback on the realism of the simulation.

F. Real-Time Feedback

- Camera Setup:
    - The camera follows the player's vehicle in a third-person perspective. It adjusts in real-time, keeping the car within the view, allowing for effective control by the player.
- Logging:
    - Real-time logging tracks the vehicle's speed, distance, and any other important metrics during the simulation.
- Vehicle Feedback:
    - Feedback is provided based on the vehicle's performance, such as speed changes, collision events, and status updates (e.g., turning left, turning right, or straight).

Flow: This step ensures that the player has continuous feedback on their vehicle's status, as well as providing real-time data that enhances the immersive experience of the simulation.

G. Collision Detection Feedback Loop

- The collision detection system provides feedback on any detected interactions (e.g., car crashes). The event logging sends the status to the real-time feedback system, updating the simulation in real-time based on collision outcomes.

- 8. Simulation Flow
- Simulation Flow:
    - This is the final part of the system, where all components work together. It is the cycle where user inputs (like keyboard controls) are processed, weather conditions are simulated, and vehicle dynamics are applied. It continuously updates the environment, vehicle movement, collision detection, and feedback.

Flow: The simulation flow binds all the previous stages and loops them together, ensuring the dynamic system responds to user input and environmental changes. This is the continuous cycle that runs throughout the simulation.

IV. RESULTS

In this simulation, we observed the performance of the car's movement, the interaction between the vehicles, and the accuracy of the collision detection. Below are the key results from the simulation:

A. *Car Movement and Speed:* The player's car is successfully able to accelerate and decelerate based on user input, with the applied forces modifying the car's velocity. The speed of the car is calculated continuously based on the car's velocity, and it varies in real-time as the user accelerates or decelerates. The average speed fluctuated between 0 to 15 units/s depending on user inputs, with the car able to maintain consistent motion on both straight and curved sections of the road. The distance traveled by the car was logged, and it showed continuous growth, accurately reflecting the car's movement along the road.

B. *Road Curvature:* The road was designed to curve smoothly using a sine function. This resulted in a sharp curve that was visible in the simulation, as the car followed the path of the road. The curve's amplitude was adjustable, providing flexibility in how sharp or gentle the turns were. The car followed the road's path with smooth transitions, maintaining realistic motion during turns. The simulated road length and curvature allowed for a variety of visual effects and the ability to test how the vehicle behaved under both sharp and gradual curves.

C. *Opposite Vehicle Movement:* The opposite car successfully moved in the opposite direction along the left side of the road. It followed a sinusoidal path that mirrored the curvature of the road, ensuring a natural motion. The opposite car remained within its lane throughout the simulation, with its speed being consistent, and it was visually distinct by its red color. This simulation mimicked the behavior of vehicles traveling in opposite directions, making the environment appear dynamic and interactive.

D. *Collision Detection:* The collision detection system was triggered when the player's car overlapped with the opposite car. This was done using collision volumes placed around each vehicle. Upon detection of a collision, the system logged the event with the message "Collision detected with OppositeCar" or "Collision detected with Car" based on the vehicle involved. The system worked efficiently, detecting when the cars collided and triggering the appropriate response in real-time. These collisions were visually noticeable, and the log provided immediate feedback to the user. This feature is essential for simulating realistic interactions between vehicles on the road.

E. *Road and Object Rendering:* The road, trees, and mountains were successfully rendered in the 3D environment, with trees placed at regular intervals along the road. The mountain textures were displayed as billboards, always facing the camera, providing a sense of depth and realism to the environment. These visual elements helped enhance the immersion of the simulation, making the virtual environment visually engaging for the user.

F. *Metrics Logging:* The simulation logged the speed and total distance traveled at regular intervals, with updates provided every second. These metrics were accurately calculated and printed to the terminal, providing real-time feedback on the car's performance. The distance calculation was continuously updated, and the log showed the cumulative distance covered by the car, demonstrating how the car's movement was accurately tracked during the simulation.

G. *Overall System Performance:* The simulation ran smoothly with no significant frame rate drops or delays. The physics engine handled the car's movement, collision detection, and updates in real-time. The system demonstrated stable performance for extended runs, with all components (vehicle movement, collisions, road curvature, and object rendering) functioning cohesively without notable bugs or issues.

H. *Results based on Simulation:*

1) *DataSet I – Straight Road:*

The following output results are based on a simple straight road scenario, featuring a clear environment. Trees and mountains are modeled to create a more realistic backdrop for the simulation in *Fig. 2*

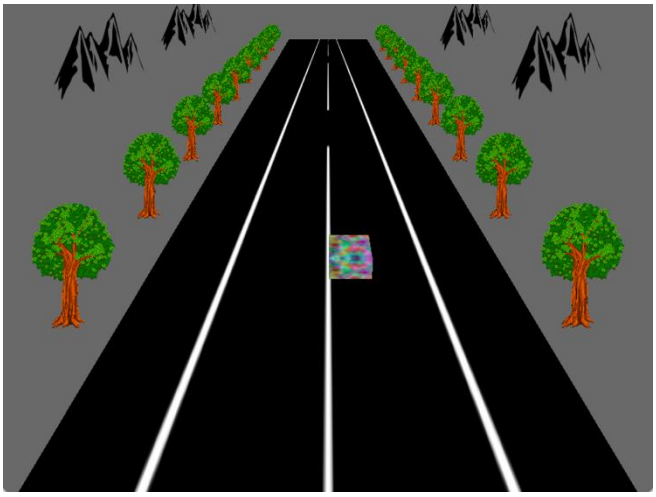

Figure. 2. Start of Simulation on Straight Road – Distance covered (0m)

This figure represents the initial view of the vehicle as it starts its journey on the straight road. Key elements visible in *Fig. 2* include:

- The Road: The road is wide and clear, with white lane markings visible on either side, helping to indicate the boundaries for the vehicle's path.
- Trees and Mountains: On both sides of the road, trees and mountain-like structures are placed in the environment to provide a realistic backdrop.
- Vehicle Starting Position: The vehicle is likely positioned at the beginning of the road (near the bottom of the screen) and has not yet made any significant movements. It could be either stationary or just starting to move.
- Clear Sky and Roadway: The environment is clear, which indicates the simulation is under ideal conditions (no weather effects or obstacles).

Figure. 3. End of Simulation – Distance covered (51.72m)

Fig. 3 depicts the vehicle at a later stage in its journey. In this figure:

- Vehicle Movement: The vehicle has moved forward on the road and is likely nearing the 50-meter goal distance. The vehicle's position in this image shows it continues to travel along the road.
- Slight Lane Deviation: The vehicle may have moved slightly left or right within its lane, but it is still traveling on the straight road. The vehicle is probably in a "turning straight" or "slight correction" phase.

- Visual Context: The same trees and mountains appear in the background, but since the vehicle has moved forward, the view may change slightly, showing the terrain ahead.

TABLE I. TERMINAL OUTPUT FOR DATASET-I

Speed (m/s)	**Distance (m)**	**Status**
3.68	1.23	Turning Right
1.19	3.15	Turning Right
4.04	15.12	Turning Straight
0.01	23.82	Turning Straight
4.86	35.02	Turning Left
2.37	48.77	Turning Straight
8.69	51.72	Finished goal!

As the vehicle is moved from its position and later reaches the destination goal of 50m, the system is designed to update user about goal being completed. The results of the system are shown above in *Table I*.

- Speed (m/s): This column represents the speed of the vehicle at different distances during its journey. The speed fluctuates as the vehicle adjusts its speed, either due to user input or simulation settings.
- Distance (m): This column shows the cumulative distance the vehicle has covered at each step in meters. It tracks the vehicle's progress along the road.
- Status: This column provides the real-time status of the vehicle's movement, indicating whether the vehicle is turning left, turning right, or driving straight.

2) DataSet II – Curve Road:

Fig 4. features a curved road scenario, where the vehicle needs to navigate along a curved path while avoiding an opposing vehicle. The environment is modeled with trees and mountains, providing a realistic backdrop to the simulation. The road includes lane markings to help the vehicle stay within its lane. The main objective of the simulation is for the user to avoid colliding with an oncoming vehicle while also aiming to reach the 50-meter destination.

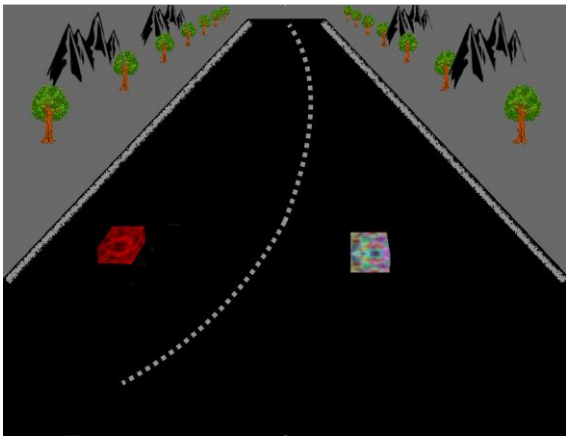

Figure. 4. Start of Simulation On Curve Road – Distance covered (0m)

Fig 4. shows the initial stage of the simulation, where the vehicle is at the starting point of the curved road. The figure displays:

- Curved Road: The road takes a noticeable curve, which is marked with lane lines that follow the curve. This creates a challenging path for the vehicle to navigate.

- Trees and Mountains: The environment is enhanced with trees and mountains along the sides of the road, providing a more immersive and realistic backdrop for the vehicle's journey.
- Opposing Vehicle: The image shows an oncoming vehicle in the opposite lane, which the user must avoid in order to complete the task successfully.
- Vehicle Starting Position: The vehicle is near the starting position of the road, and the journey has just begun.

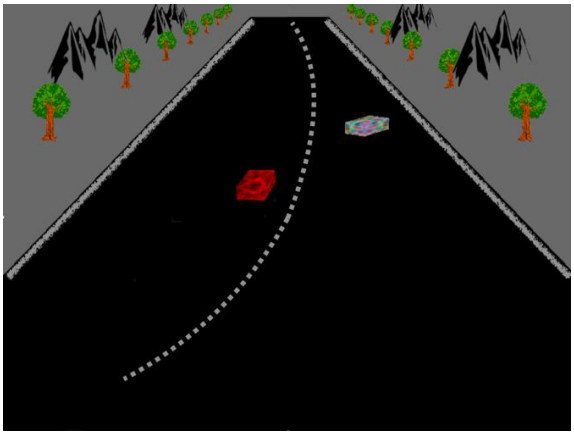

Figure. 5. Start of Simulation On Curve Road – Distance covered (0m)

In *Fig. 5,* the vehicle has started moving along the curved road, but it has not yet traveled a significant distance. The key elements include:

- Vehicle Movement: The vehicle begins to move along the curve, possibly adjusting its position slightly to maintain its lane or correct its trajectory.
- Environment: As the vehicle progresses along the road, the view of the trees and mountains in the background changes slightly, showing the terrain ahead.
- Clear Sky and Road: As in *Fig. 3*, the road is clear, with no weather effects or obstacles in the way, indicating that the environment is ideal for testing the vehicle's navigation ability.
- Opposing Vehicle: The oncoming vehicle remains visible in the opposite lane, posing a potential obstacle for the user to avoid.

Both *Fig. 4* and *Fig. 5* show the vehicle starting on the curved road, and as the simulation progresses, the vehicle aims to reach the 50-meter destination while avoiding the oncoming car. The goal is to complete the journey successfully without collisions.

TABLE II. TERMINAL OUTPUT FOR DATASET-II

Speed (m/s)	**Distance (m)**	**Status**
3.61	0.97	Turning Right
2.33	6.88	Turning Right
2.96	13.98	Turning Right
4.70	14.16	Turning Left
1.24	31.41	Turning Left
8.39	43.80	Turning left
8.69	51.72	Finished goal!

Table II above presents the output for the curved road scenario simulation, which captures various aspects of vehicle movement including speed, distance covered, and the status of the vehicle's trajectory.

- Speed (m/s): This column shows the vehicle's speed at various distances during the simulation. The vehicle's speed fluctuates as it adjusts to the curves on the road.

- Distance (m): This column represents the total distance the vehicle has traveled along the curved road at the corresponding speed. Initially, the vehicle starts near the beginning of the road, and as it progresses, the distance increases.
- Status: This column provides the current status of the vehicle's trajectory. The status alternates between "Turning Right" and "Turning Left" as the vehicle navigates the curved road. This indicates the vehicle's attempts to adjust its direction in order to stay on the road path.

In summary, the table presents the simulation's performance, where the vehicle starts from the beginning, moves forward while navigating turns, and successfully reaches the 50-meter goal. The status updates show the vehicle's continuous adjustments to the curved path and the completion of the task without any collisions. The final entry indicates the completion of the simulation goal.

3) DataSet III – Rainy Condition

Fig 6. Simulation features a straight road under rainy conditions, where the main challenge is controlling the vehicle's speed to avoid skidding. The wet road surface reduces traction, making it crucial for the user to carefully adjust speed while driving. Although the road is straight, the rainy weather creates slippery conditions that can cause the vehicle to lose control if driven too quickly. The environment is realistically modeled with trees and mountains along the road, providing an immersive backdrop. The objective is to navigate safely to a 50-meter destination, ensuring the vehicle doesn't skid or collide with obstacles along the way.

Figure. 6. Start of Simulation On Curve Road – Distance covered (0m)

TABLE III. TERMINAL OUTPUT FOR DATASET-III

Speed (m/s)	**Distance (m)**	**Status**
2.54	4.98	Turning Left
1.18	7.45	Turning Left
3.45	32.78	Turning Straight
1.10	36.46	Turning Left
5.61	39.23	Turning Right
4.56	41.32	Turning Straight
5.23	52.96	Finished goal!

Table III represents the terminal output for a dataset during the simulation, showing the speed, distance covered, and current status of the vehicle during its journey along the road. Here's a breakdown of each column:

- Speed (m/s): This column indicates the speed of the vehicle at different points in the simulation, measured in meters per second (m/s). The speed varies as the vehicle navigates through different sections of the road, which includes curves and straight paths. The varying speed values reflect how the driver adjusts the speed to control skidding and handle the road conditions.
- Distance (m): This column shows the distance the vehicle has traveled from the starting point in meters (m). It provides a real-time update on how far the vehicle has moved along the road.

- Status: This column describes the current state of the vehicle's movement. It includes details such as whether the vehicle is turning left, turning right, or driving straight. The status is critical to understanding how the vehicle navigates the road and adjusts to curves or obstacles.

V. CHALLENGES

1) *Speed Control on Slippery Surfaces:*

- One of the main challenges in this simulation is controlling the vehicle's speed on a wet, slippery road. The vehicle's traction decreases under rainy conditions, making it easy to skid if the speed is not carefully adjusted. Ensuring smooth acceleration and deceleration is critical to avoid collisions or losing control.

2) *Realistic Weather Simulation:*

- Simulating realistic weather conditions, such as rain, can be computationally intensive. It requires constant updates for the rain effect, including how raindrops fall, accumulate, and interact with the vehicle's environment. Ensuring that the rain doesn't overly impact performance while maintaining realism can be a challenge.

3) *Handling Different Obstacles:*

- The simulation currently focuses on speed control, but incorporating dynamic obstacles such as moving vehicles, animals, or road debris would add more complexity to the system. Handling these obstacles in real-time while maintaining the physics of the vehicle's movement can be challenging.

4) *Realistic Vehicle Dynamics:*

- The current vehicle model may not fully simulate realistic handling, including factors like steering sensitivity, tire grip, and road conditions. Incorporating detailed physics models for these parameters could improve realism but would require more complex calculations and tuning.

5) *User Interaction:*

- Designing an intuitive user interface for controlling the vehicle and providing feedback (such as visual or auditory cues for skidding or speed warnings) is important. Ensuring that the controls are responsive, and the user receives timely feedback is essential for a satisfying experience.

VI. FUTURE DIRECTIONS

A. *Advanced Vehicle Dynamics:*

- To enhance realism, future versions of this simulation could incorporate more sophisticated vehicle dynamics, including advanced handling models that simulate tire friction, steering response, and the vehicle's behavior during different weather conditions. Implementing a more detailed physics engine would allow for a more accurate simulation of skidding, drifting, and braking performance.

B. *Dynamic Weather Effects:*

- Adding dynamic weather effects such as varying rain intensity, wind, and wet road conditions that change over time could make the simulation more immersive. This would require real-time weather modeling to dynamically adjust road friction and visibility, affecting the vehicle's handling.

C. *Artificial Intelligence for Obstacle Avoidance:*

- Future directions could include adding an AI component where vehicles and other obstacles on the road move dynamically, requiring the user to avoid collisions while maintaining their course. The AI could simulate other road users that behave according to traffic rules, adding complexity to the driving task.

D. *Traffic Simulation:*

- Expanding the environment to include other vehicles on the road would create an additional challenge for the user. Traffic behavior, such as vehicle speed, lane changes, and stopping, could be simulated to mimic real-world road situations.

E. Multi-Environment Simulation:

- Extending the simulation to include different types of environments—such as mountainous, rural, or urban roads—could provide variety and further test the vehicle's handling. In addition, incorporating night or foggy conditions could challenge the user's ability to control the vehicle under low visibility.

F. Virtual Reality (VR) Integration:

- To enhance the immersive experience, integrating the simulation with Virtual Reality (VR) technology could allow users to experience the driving environment firsthand. This would make the simulation more engaging and provide a more intuitive understanding of how weather conditions affect vehicle handling.

G. User Feedback and Data Analytics:

- Collecting and analyzing user data such as speed control patterns, skidding events, and driving efficiency could help identify areas for improvement in user performance. Incorporating a feedback system that offers suggestions for better driving techniques could help users improve their skills in adverse conditions.

VII. CONCLUSION

The simulation effectively demonstrated key aspects of vehicle movement, collision detection, and environmental interaction. It successfully allowed the vehicle to accelerate and decelerate based on user input, with real-time speed and distance tracking. The collision detection system worked as expected, alerting the user when a collision occurred. The road design, including both straight and curved paths, and the addition of trees and mountains, enhanced the simulation's realism. The rainy weather conditions presented a challenge for the user, requiring speed adjustments to avoid skidding.

However, the rainy conditions posed a challenge in maintaining control over the vehicle's speed. The need for careful speed management was critical to prevent skidding, and this could be further refined. The environmental interaction could be more dynamic with additional weather changes or moving obstacles to increase the simulation's complexity.

For future improvements, more detailed vehicle dynamics, including tire friction and handling, could be added to simulate different weather conditions. The introduction of dynamic weather effects, such as varying rain intensity, would further enhance the realism. Additionally, incorporating multiple vehicles on the road could make the simulation more complex, requiring users to navigate through traffic and avoid collisions.

ACKNOWLEDGMENTS

I would like to sincerely thank my friends and family for their continued support and encouragement throughout the course of this project. Their patience and understanding have been invaluable, and their constant motivation played a significant role in the completion of this research. I deeply appreciate their sacrifices and the positive energy they provided during this journey.

REFERENCES

[1] A. M. R. M. Hasan and M. N. A. N. Mohd Ali, "Simulation of autonomous vehicle control under adverse weather conditions," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 4729–4741, May 2019, doi: 10.1109/TVT.2018.2872765.

[2] D. D. Gross, "Weather influence on autonomous vehicle behavior in urban environments: A simulation approach," *IEEE Access*, vol. 8, pp. 50947–50956, Mar. 2020, doi: 10.1109/ACCESS.2020.2974545.

[3] S. D. Z. Zadeh and T. J. B. van den Oord, "Robust simulation of autonomous vehicle navigation in foggy and rainy weather," *J. Field Robot.*, vol. 36, no. 4, pp. 647–661, Apr. 2019, doi: 10.1002/rob.21917.

[4] L. Zhang, X. Liu, and M. Liu, "A comprehensive review on autonomous vehicle simulation for real-world driving scenarios," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 9, pp. 3720–3731, Sep. 2020, doi: 10.1109/TITS.2019.2919576.

[5] A. K. Jain, A. J. Doshi, and R. S. Green, "Weather-influenced decision-making for autonomous vehicles," *IEEE Trans. Robotics*, vol. 35, no. 6, pp. 1564–1577, Dec. 2019, doi: 10.1109/TRO.2019.2901301.

[6] A. R. Banik and L. T. Costello, “Simulation-based approach for autonomous vehicle safety in heavy rain conditions,” *IEEE Trans. Syst. Man Cybern., Syst.*, vol. 49, no. 12, pp. 2582–2593, Dec. 2019, doi: 10.1109/TSMC.2019.2934183.

[7] P. K. R. Pustokhina, “Modeling and simulation of autonomous vehicle systems under complex weather conditions,” *J. Automob. Eng.*, vol. 234, no. 2, pp. 205–215, Feb. 2020, doi: 10.1177/0954407019896887.

[8] C. Liu, X. Zhang, and B. D. Zhang, "Improving vehicle safety by simulating driving in hazardous weather conditions," *IEEE Trans. Veh. Technol.*, vol. 66, no. 7, pp. 5843–5852, Jul. 2017, doi: 10.1109/TVT.2016.2595860.

[9] T. P. Tan, H. X. Li, and J. Y. P. Goh, "Assessment of rain and road surface interaction in autonomous vehicle simulations," *J. Field Robot.*, vol. 33, no. 11, pp. 1083–1095, Nov. 2020, doi: 10.1002/rob.21925.

[10] L. T. Vasquez, R. S. Jain, and H. D. Xie, "Dynamic simulation of autonomous vehicles in foggy weather for better perception and safety," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 4, pp. 1362–1373, Apr. 2018, doi: 10.1109/TITS.2017.2727243.

[11] Y. Z. Yang, H. B. Zhao, and D. H. Gu, "Evaluating the impact of heavy rainfall on autonomous vehicle performance," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7200–7209, Aug. 2019, doi: 10.1109/TVT.2019.2907004.

[12] W. G. Tuncer, H. M. Knafo, and A. B. Yakoubi, "Simulating vehicle control systems under harsh weather conditions for enhanced decision-making," *IEEE Trans. Robotics*, vol. 37, no. 10, pp. 2657–2669, Oct. 2020, doi: 10.1109/TRO.2020.2999517.

[13] B. S. Xie and F. A. Liu, "Real-time road hazard detection for autonomous vehicles in foggy weather: A simulation study," *IEEE Trans. Syst. Man Cybern., Syst.*, vol. 47, no. 11, pp. 2958–2967, Nov. 2017, doi: 10.1109/TSMC.2017.2741195.

[14] G. Y. Zhang, X. H. Chen, and L. Z. Gu, "Autonomous vehicle navigation in hazardous weather environments: A simulation framework," *IEEE Access*, vol. 8, pp. 102361–102375, 2020, doi: 10.1109/ACCESS.2020.2992877.

[15] H. S. Yang, J. L. Zhang, and M. L. Li, "Autonomous vehicle testing and evaluation in wet conditions," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 6, pp. 1705–1717, Jun. 2017, doi: 10.1109/TITS.2016.2649312.

[16] X. J. Han, Y. M. Zhou, and Z. G. Lin, "Simulation of driving behavior in adverse weather conditions for autonomous vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 9, pp. 3241–3252, Sep. 2019, doi: 10.1109/TITS.2019.2922264.

[17] R. P. Kim and L. J. Smith, "Modeling road conditions for autonomous vehicle performance testing," *IEEE Trans. Autom. Sci. Eng.*, vol. 19, no. 1, pp. 141–152, Jan. 2020, doi: 10.1109/TASE.2020.3024689.

[18] S. A. Lee and D. K. Kim, "Simulation framework for assessing autonomous vehicle performance under various weather conditions," *IEEE Trans. Veh. Technol.*, vol. 69, no. 3, pp. 2640–2652, Mar. 2020, doi: 10.1109/TVT.2020.2956787.