Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

Design and Development of an Indian-Themed Open-World Urban Driving and Racing Simulation using Unreal Engine 5

Anshuman Behera, Harshit Verma, Nakka Vinayak Sai
Department of Computer Science and Engineering w/s in gaming technology
SRM Ramapuram, India

***Abstract*—This paper presents the design and implementation of a narrative-driven open-world driving and racing simulation system developed using Unreal Engine 5. Existing driving simulations often fail to represent the complexity of heterogeneous traffic environments and socio-economic dynamics found in regions such as India. To address this limitation, the proposed system models an Indian urban environment incorporating mixed traffic behavior, non-linear driving patterns, and progression-based gameplay.**

The system is built using a modular layered architecture that integrates real-time input handling, game logic, artificial intelligence, world management, and rendering subsystems. NPC behavior is implemented using a hybrid AI model combining finite state machines for efficient low-level traffic simulation and behavior trees for adaptive decision-making in complex agents such as police and enemy racers.

Additionally, the system introduces an economic progression model and a dynamic day–night cycle, enhancing realism and player engagement. Experimental evaluation demonstrates stable performance across varying environment densities and consistent NPC interaction behavior.

The proposed framework contributes to the development of scalable AI-driven game systems and highlights the potential of culturally contextualized simulation design in open-world environments.

***Index Terms*—Open-world simulation, NPC behavior, Game AI, Unreal Engine 5, Vehicle Simulation**

I. Introduction

Open-world simulation systems have emerged as a fundamental paradigm in modern game development and interactive applications due to their ability to model complex environments and provide non-linear, player-driven experiences [7]. These systems enable dynamic interaction between the player and multiple entities within a continuously evolving environment. However, most existing driving and racing simulations are designed around structured and predictable environments, often based on Western urban layouts and high-performance vehicle ecosystems.

Such systems fail to accurately represent the complexity of real-world urban traffic conditions, particularly in developing regions such as India. Indian urban environments are characterized by heterogeneous traffic composition, including rickshaws, motorcycles, hatchbacks, buses, and pedestrians coexisting within shared spaces. Additionally, these environments exhibit non-linear traffic flow, limited adherence to lane discipline, and strong socio-economic influences that affect transportation patterns.

Despite these complexities, current simulation systems lack the ability to model:

- Mixed-traffic ecosystems with diverse vehicle types
- Dynamic and unpredictable driving behavior
- Socio-economic progression within gameplay systems
- Integration of narrative elements with simulation mechanics

To address these limitations, this paper proposes the design and development of an Indian-themed open-world urban driving and racing simulation system. The proposed system integrates real-time simulation, artificial intelligence, and narrative-driven gameplay into a unified framework.

A key feature of the system is its progression-based gameplay model, where the player begins as a low-income rickshaw driver and gradually advances into competitive street racing. This progression is governed by an economic system that links player performance with resource accumulation and vehicle upgrades. The inclusion of a risk-based restart mechanism further enhances player engagement by introducing high-stakes decision-making.

From a technical perspective, the system is built using a modular layered architecture that separates input processing, game logic, artificial intelligence, world management, and rendering [4]. NPC behavior is implemented using a hybrid AI model combining Finite State Machines (FSM) for low-level traffic simulation [2] and Behavior Trees for high-level decision-making in complex agents such as police and enemy racers [6].

The primary contributions of this work are summarized as follows:

- Design of a culturally contextualized open-world driving simulation reflecting Indian urban traffic conditions
- Development of a hybrid AI framework for scalable and adaptive NPC behavior
- Integration of an economic progression model with gameplay mechanics
- Implementation of a modular system architecture supporting real-time simulation
- Evaluation of system performance and AI behavior in a dynamic open-world environment

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

The remainder of this paper is organized as follows. Section II presents the literature review of existing AI and simulation techniques. Section III describes the system architecture. Section IV explains the NPC behavior system. Section V details the gameplay design. Section VI discusses implementation aspects. Section VII presents results and analysis, followed by the conclusion in Section VIII.

II. Literature Review

This section reviews existing approaches in game artificial intelligence and open-world simulation systems, focusing on behavior modeling techniques and their applicability to real-time environments.

A. Finite State Machines in Game AI

Finite State Machines (FSM) are one of the earliest and most widely used techniques for modeling NPC behavior in games [2], [5]. An FSM represents behavior as a set of discrete states with transitions triggered by conditions.

The general FSM transition model can be defined as:

$$S_{next} = f(S_{current}, E) \tag{1}$$

where S represents the state and E represents environmental inputs.

FSMs are commonly used in traffic systems due to their simplicity and low computational cost [2]. They are particularly effective in scenarios where behavior is predictable and limited in complexity, such as vehicle movement along predefined paths.

1) Advantages of FSM:

- Low computational overhead, suitable for large-scale NPC populations
- Deterministic behavior, ensuring predictable outcomes
- Ease of implementation and debugging

2) Limitations of FSM: Despite their advantages, FSMs exhibit several limitations:

- State explosion problem as complexity increases
- Difficulty in handling dynamic and unpredictable environments
- Limited adaptability to player-driven interactions

These limitations make FSMs insufficient for complex NPC behaviors such as adaptive racing or police pursuit systems.

B. Behavior Trees for Decision Making

Behavior Trees (BT) are a hierarchical model used to represent decision-making processes in modern game AI [3], [6]. A behavior tree consists of nodes such as selectors, sequences, and action nodes, which define how decisions are evaluated.

The decision process in a behavior tree can be represented as:

$$Action = Selector(Sequence(Conditions, Actions)) \tag{2}$$

Behavior Trees are widely used in modern game engines due to their modularity and flexibility [6].

1) Advantages of Behavior Trees:

- Hierarchical structure enabling complex decision logic
- Reusability of behavior modules
- Improved adaptability to dynamic environments
- Easier debugging compared to deeply nested FSMs

2) Limitations of Behavior Trees:

- Higher computational cost compared to FSM
- Increased implementation complexity
- Requires careful design to avoid inefficient evaluation

C. Hybrid AI Models

Recent research and industry practices favor hybrid AI models that combine FSM and Behavior Trees [3]. In such systems:

- FSM is used for low-level, repetitive behaviors (e.g., traffic movement)
- Behavior Trees are used for high-level decision-making (e.g., police AI, enemy racers)

The hybrid model can be expressed as:

$$AI_{system} = FSM_{low-level} + BT_{high-level} \tag{3}$$

This approach balances computational efficiency with behavioral flexibility.

D. Pathfinding and Navigation Systems

In addition to decision-making models, NPC movement relies heavily on pathfinding algorithms such as A* and navigation mesh (NavMesh) systems [5].

Pathfinding can be defined as:

$$Path = \arg\min \sum Cost(Node_i) \tag{4}$$

where the objective is to find the optimal path between two points.

In modern game engines, NavMesh systems are used to allow dynamic navigation, enabling NPCs to adapt to changing environments [10].

E. Open-World Simulation Systems

Open-world games require efficient management of large environments and multiple interacting agents. Key challenges include:

- Real-time rendering of large maps
- Synchronization of multiple AI agents
- Dynamic loading and unloading of world segments

Technologies such as level streaming and world partitioning are commonly used to address these challenges [7].

F. Limitations of Existing Systems

Despite advancements in AI and simulation systems, existing approaches exhibit several limitations:

- Lack of cultural contextualization, particularly for Indian urban environments
- Limited integration between AI behavior and narrative systems

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

- Absence of economic progression models in simulation-based games
- Over-reliance on deterministic AI without adaptive learning

G. Research Gap and Motivation

Based on the analysis of existing literature, the following research gaps are identified:

- Need for simulation of heterogeneous traffic systems found in Indian cities
- Integration of AI behavior with narrative-driven gameplay
- Incorporation of economic progression as a gameplay mechanic
- Development of scalable AI systems suitable for open-world environments

H. Proposed Approach

To address these gaps, the proposed system adopts:

- A hybrid AI model combining FSM and Behavior Trees
- A layered system architecture for scalability
- A narrative-driven progression model
- An economic system integrated with gameplay

This approach enables the development of a realistic and engaging open-world simulation tailored to Indian urban environments.

III. System Architecture

The proposed system is designed using a modular layered architecture to ensure scalability, maintainability, and real-time performance. Layered architectures are commonly used in game system design to separate concerns and improve system organization [7].

The architecture is structured to handle interactions between player input, AI behavior, physics simulation, and rendering systems while maintaining synchronization across all components.

The overall system flow can be represented as:

$$Input \rightarrow GameLogic \rightarrow AI/Simulation \rightarrow World \rightarrow Rendering \quad (5)$$

A. Input Management Layer

The Input Management Layer acts as the interface between the player and the system. It captures real-time input signals from devices such as keyboard or controller and converts them into structured commands.

1) Input Processing: Player inputs include:

- Steering (left/right rotation)
- Acceleration and braking
- Interaction commands (mission triggers, UI navigation)

Input signals are processed using an event-driven model:

$$Command = f(Input, Time) \quad (6)$$

This approach is commonly used in modern game engines to ensure responsive interaction [4].

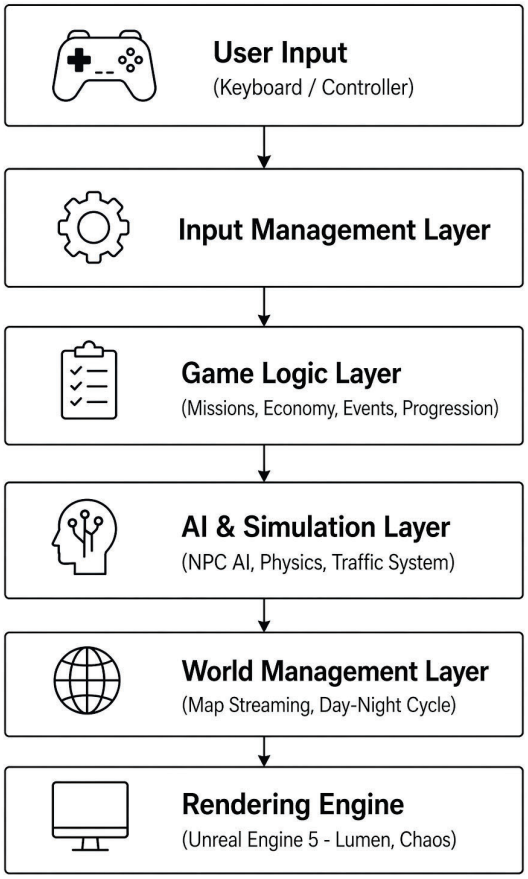

Fig. 1. Layered Architecture of the Proposed Open-World Simulation System

2) Input Synchronization: To ensure smooth gameplay, inputs are processed at fixed update intervals, preventing inconsistencies due to frame rate variation.

B. Game Logic Layer

The Game Logic Layer acts as the central control unit of the system. It manages gameplay rules, progression systems, and event coordination.

1) Mission Management System: Missions are handled using a state-based system:

$$MissionState = \{Inactive, Active, Completed\} \quad (7)$$

2) Economic System Integration: The economic system tracks player income and expenditure:

$$Balance_{t+1} = Balance_t + Income - Cost \quad (8)$$

3) Event Handling System: The system uses an event-driven architecture:

$$Event = f(PlayerAction, GameState) \quad (9)$$

Event-driven systems are widely used in interactive applications for managing asynchronous gameplay interactions [3].

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

C. AI and Simulation Layer

This layer is responsible for simulating dynamic entities and physical interactions.

1) NPC Behavior Integration: NPC behavior models are implemented using hybrid AI techniques combining FSM and Behavior Trees [2], [6].

$$Behavior = f(State, Environment, PlayerAction) \quad (10)$$

2) Vehicle Physics Simulation: Vehicle dynamics are modeled using physics engines such as Chaos in Unreal Engine [4]. Motion is governed by:

$$F = m \cdot a \quad (11)$$

$$Velocity_{t+1} = Velocity_t + a \cdot \Delta t \quad (12)$$

3) Collision Detection System: Collision detection is implemented using proximity checks:

$$Collision = (d < d_{threshold}) \quad (13)$$

D. World Management Layer

The World Management Layer controls large-scale environment handling.

1) World Partition and Streaming: The environment is divided into grid-based segments:

$$World = \sum_{i=1}^{n} Cell_i \quad (14)$$

Dynamic loading is performed based on player position:

$$LoadedCells = f(PlayerPosition) \quad (15)$$

Level streaming and world partitioning are standard techniques in open-world systems [7].

2) Day-Night System: The system dynamically updates environmental conditions:

$$EnvironmentState = f(Time) \quad (16)$$

This affects gameplay and AI behavior.

E. Rendering and Engine Layer

The Rendering Layer is responsible for visual output and frame generation.

1) Rendering Pipeline:

$$Frame = f(Geometry, Lighting, Camera) \quad (17)$$

2) Lighting and Physics Integration: The system utilizes Unreal Engine 5 technologies such as Lumen for global illumination and Chaos for physics simulation [4].

F. Inter-Layer Communication

Communication between layers is achieved using an event-driven model:

$$DataFlow = Input \rightarrow Logic \rightarrow AI \rightarrow World \rightarrow Render \quad (18)$$

G. System Scalability and Modularity

The modular design allows independent scaling of subsystems. Such architectures are widely adopted in game development for handling complex systems efficiently [7].

H. System Limitations

The current architecture has limitations:

- Increased computational cost in dense environments
- Dependency on rule-based AI systems
- Limited scalability for extremely large worlds

Future improvements can include distributed simulation and advanced AI optimization techniques.

IV. NPC Behavior System

The NPC behavior system is a core component of the proposed simulation, responsible for generating realistic interactions between the player and the environment. The system is designed using a hybrid artificial intelligence model that combines Finite State Machines (FSM), Behavior Trees, and rule-based decision systems.

NPCs in the system are categorized into multiple classes based on functionality:

- Traffic NPCs
- Police NPCs
- Enemy Racers
- Mission-Based NPCs
- Story and Character NPCs

Each category uses a different level of AI complexity depending on its role in gameplay.

A. Traffic AI System

Traffic NPCs are implemented using Finite State Machines (FSM) to ensure computational efficiency while simulating large numbers of agents.

1) State Definition: Each traffic vehicle operates within the following states:

- Idle: Vehicle is stationary
- Move: Vehicle follows predefined path
- Stop: Vehicle halts due to obstacle or signal
- Avoid Collision: Vehicle adjusts speed or direction

The state transition is defined as:

$$S_{next} = f(S_{current}, E) \quad (19)$$

where E represents environmental inputs such as:

- Distance to nearby vehicles
- Traffic signals
- Road constraints

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

2) Path Following Mechanism: Traffic vehicles follow spline-based paths. The movement update is defined as:

$$Position_{t+1} = Position_t + Velocity \cdot \Delta t \tag{20}$$

This ensures smooth and predictable motion.

3) Collision Avoidance Model: Collision avoidance is implemented using proximity detection:

$$d < d_{threshold} \Rightarrow Velocity = 0 \tag{21}$$

For dynamic adjustment:

$$Velocity = Velocity \cdot (1 - \alpha) \tag{22}$$

where α is a deceleration factor.

B. Police AI System

Police NPCs are implemented using Behavior Trees to allow dynamic and context-aware decision-making.

1) Behavior Tree Structure: The decision flow is structured as:

- Patrol State
- Detection Node
- Chase Sequence
- Interception Action

The decision selection can be represented as:

$$Action = Selector(Patrol, Detect, Chase, Intercept) \tag{23}$$

2) Player Detection Mechanism: Detection is based on multiple parameters:

- Player speed threshold
- Proximity to police NPC
- Illegal race participation

Detection condition:

$$Detection = (Speed > S_{limit}) \vee (Distance < d_{detect}) \tag{24}$$

3) Chase Behavior: During chase, the police AI adjusts speed and direction based on player position:

$$Direction = PlayerPosition - NPCPosition \tag{25}$$

C. Enemy Racer AI

Enemy racers are designed to simulate competitive driving behavior.

1) Behavior Characteristics:

- Aggressive overtaking
- Blocking player path
- Adaptive speed control

2) Racing Strategy Model: Enemy behavior is governed by:

$$Strategy = f(PlayerPosition, TrackCondition, Speed) \tag{26}$$

3) Difficulty Scaling: Enemy difficulty is adjusted dynamically:

$$Difficulty = Base + \beta(PlayerPerformance) \tag{27}$$

where β is a scaling factor.

D. Mission-Based NPC AI

Mission NPCs are used in story and side missions. These include passengers, informants, and event-trigger characters.

1) Interaction Model: NPC interaction is event-driven:

$$Event = f(PlayerAction, Location) \tag{28}$$

Examples include:

- Passenger pickup triggers mission start
- Drop-off triggers reward calculation
- Dialogue triggers story progression

2) Behavior Logic: Mission NPCs use simple state machines:

- Waiting
- Interacting
- Completed

E. Story and Villain AI

The main antagonist and story-driven characters use scripted AI combined with behavior logic.

1) Villain AI (Boss Race): The final race opponent uses advanced behavior:

- Aggressive acceleration patterns
- Strategic blocking
- High-speed path optimization

2) Boss Behavior Model:

$$BossAction = f(PlayerPosition, RaceProgress, AggressionLevel) \tag{29}$$

Aggression increases over time:

$$Aggression = Aggression_0 + \gamma t \tag{30}$$

F. Pedestrian AI System

Pedestrians are implemented using simple rule-based AI:

- Random movement patterns
- Road crossing behavior
- Collision avoidance with vehicles

Movement is modeled as:

$$Position_{t+1} = Position_t + RandomDirection \tag{31}$$

G. AI System Integration

All NPC systems are integrated into a unified AI framework:

$$AI_{system} = Traffic + Police + Enemy + Mission + Story \tag{32}$$

Each subsystem communicates through shared environmental data and event triggers, ensuring consistent interaction across the game world.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

H. Limitations of Current AI Model

The current AI system is rule-based and does not include learning mechanisms. Limitations include:

- Lack of adaptive learning
- Predictable behavior patterns over time
- Limited pathfinding intelligence in complex environments

Future improvements can include reinforcement learning and adaptive decision systems.

V. Gameplay Design

The gameplay design of the proposed system is centered around progression, risk, and player-driven decision-making. Unlike conventional racing games, the system integrates narrative elements with simulation mechanics to create a layered gameplay experience, which is a common design approach in modern open-world systems [7]. The player transitions through multiple phases of gameplay, each introducing new mechanics, challenges, and objectives.

A. Core Gameplay Loop

The core gameplay loop is designed as a progression cycle that reinforces player engagement through reward and challenge mechanisms:

$$Start \rightarrow Earn \rightarrow Upgrade \rightarrow Compete \rightarrow Risk \rightarrow Progress \tag{33}$$

Initially, the player begins as a rickshaw driver performing low-risk transportation tasks. As the player earns currency, they gain access to upgraded vehicles and higher-reward opportunities such as cab driving and street racing.

The loop is designed to ensure:

- Continuous progression through economic rewards
- Increasing difficulty and challenge levels
- Player motivation through vehicle upgrades and achievements

Such progression loops are widely used in game design to maintain player engagement [3].

B. Player Progression System

The progression system is divided into multiple stages:

- **Stage 1: Rickshaw Driving** — Basic transportation missions with low income and minimal risk
- **Stage 2: Cab Driving** — Moderate income with increased navigation complexity
- **Stage 3: Street Racing** — High-risk, high-reward competitive gameplay
- **Stage 4: Final Confrontation** — High-stakes race with complete progression reset on failure

Each stage introduces new gameplay mechanics and increases system complexity, aligning with structured difficulty progression models [3].

C. Mission System Design

The game incorporates a mission-based structure to guide player progression. Mission systems are widely used in open-world games to provide structured objectives within non-linear environments [7].

Missions are categorized into:

- **Transport Missions**: Picking up and dropping passengers
- **Time-Based Missions**: Completing tasks within a time limit
- **Race Events**: Competing against AI-controlled opponents
- **Story Missions**: Narrative-driven events that advance the plot

Mission completion is evaluated using:

$$Reward = f(Time, Distance, Performance) \tag{34}$$

where performance includes factors such as driving efficiency and collision avoidance.

D. Economic System and Resource Management

The economic system is a central component of gameplay progression. Player income is calculated as:

$$Income = \sum (PassengerFare + RaceReward - DamageCost) \tag{35}$$

Economic systems are commonly used in games to regulate progression and enforce strategic decision-making [3].

Key economic mechanics include:

- Vehicle purchase and upgrade system
- Repair costs based on vehicle damage
- Risk-reward trade-offs in racing

This system forces the player to make strategic decisions regarding spending and risk-taking.

E. Vehicle Upgrade and Progression Mechanics

Vehicles can be upgraded to improve performance characteristics such as:

- Acceleration
- Maximum speed
- Handling and stability
- Durability

Upgrade effectiveness can be modeled as:

$$Performance = BaseStats + UpgradeModifiers \tag{36}$$

This creates a tangible sense of progression and directly impacts gameplay outcomes.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

F. Risk-Based Gameplay Mechanism

The system introduces a high-risk gameplay element through a restart-based failure condition. If the player fails in critical events, particularly the final race, the system resets progression:

$$Failure \Rightarrow Reset(State) \tag{37}$$

Risk-reward balancing is a fundamental concept in game design, influencing player engagement and decision-making [3].

This mechanism introduces:

- Strategic decision-making
- Increased tension during gameplay
- Higher replayability

Players must balance risk and reward when choosing to participate in races.

G. Day-Night Gameplay Dynamics

The gameplay system dynamically alternates between day and night cycles, a common feature in open-world simulations used to introduce environmental variation [7].

1) Day Mode:

- High traffic density
- Availability of transport missions
- Lower risk gameplay

2) Night Mode:

- Reduced traffic density
- Activation of racing events
- Increased presence of adversarial NPCs
- Higher risk and reward

The transition between day and night introduces variation and prevents gameplay monotony.

H. Player Decision-Making and Strategy

The gameplay system encourages player-driven decisions based on:

- Economic condition
- Vehicle capability
- Risk tolerance

Decision-making can be represented as:

$$Decision = f(EconomicState, Risk, Reward) \tag{38}$$

Decision systems are essential in interactive design as they directly influence player engagement [3].

This ensures that player choices directly influence progression outcomes.

I. System Integration with Narrative

The gameplay design is closely integrated with the narrative structure. Key events such as races, enemy encounters, and progression resets are tied to story progression.

Narrative-driven gameplay is widely adopted in modern game design to enhance immersion and emotional engagement [7].

This integration ensures:

- Emotional engagement
- Contextual gameplay objectives
- Continuity between gameplay and narrative

VI. Implementation

This section provides a detailed description of the practical implementation of the proposed open-world driving simulation system. The implementation focuses on achieving real-time performance, realistic vehicle behavior, and scalable AI interaction while maintaining modularity across system components.

A. Development Tools and Engine Selection

The system is implemented using Unreal Engine 5 due to its advanced rendering capabilities, physics simulation framework, and support for large-scale open-world environments [4]. The engine provides built-in systems such as World Partition, Lumen global illumination, and Chaos Physics, which are essential for real-time simulation.

Unreal Engine follows a component-based architecture, where actors are composed of multiple components such as mesh, physics, and input handlers. This design allows flexible integration of gameplay systems [4].

Blueprint Visual Scripting is used for implementing gameplay logic, event handling, and rapid prototyping. The use of Blueprints enables faster iteration compared to low-level C++ implementation while maintaining sufficient control over system behavior.

Additionally, Blender is used for asset creation and modification, allowing customization of vehicles and environmental objects. Quixel Megascans are utilized for realistic environmental assets such as roads, buildings, and terrain.

B. Environment and Map Design

The game environment is designed as a semi-open world using Unreal Engine's World Partition system, which enables efficient handling of large-scale environments [4]. This system divides the world into grid-based cells, which are dynamically loaded and unloaded based on player position.

The map is structured into multiple functional zones to support different gameplay mechanics.

1) Slum Area (Low-Income Zone): This zone represents the starting environment of the player and is designed with high spatial density and limited road width. The purpose of this design is to:

- Restrict player speed and encourage controlled driving
- Introduce navigation challenges due to narrow paths
- Simulate realistic congestion found in low-income urban areas

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

2) City Center (Structured Urban Zone): The city center introduces structured traffic systems including signals and wider roads. This zone is designed to:

- Increase traffic complexity through signal-based movement
- Introduce police AI interaction
- Enable mid-level economic gameplay such as cab driving

Traffic nodes and intersections are implemented using spline-based path systems, a common approach in game AI navigation [5].

3) Industrial Zone (High-Risk Gameplay Area): The industrial zone is optimized for racing and action sequences. It features:

- Reduced obstacle density for high-speed movement
- Wide road segments for overtaking and drifting
- Low lighting conditions to enhance difficulty during night gameplay

4) Highway Network (High-Speed Simulation): The highway system is designed for maximum speed and minimal interruption. It includes:

- Long straight segments for acceleration testing
- Curved sections for drift and control mechanics
- Reduced NPC density to minimize collision probability

C. Vehicle System Implementation

Vehicle dynamics are implemented using the Chaos Physics Engine provided by Unreal Engine [4]. Each vehicle is represented as a rigid body with parameters defining its motion characteristics.

The vehicle motion model is governed by:

$$F = m \cdot a \tag{39}$$

where F represents the force applied by the engine, m is vehicle mass, and a is acceleration.

Key parameters configured for each vehicle include:

- Engine torque curve
- Tire friction coefficient
- Suspension stiffness
- Damping factor

Different vehicle categories are assigned distinct parameter sets:

- Rickshaw: Low speed, high instability, low torque
- Hatchback: Balanced performance and control
- Racing vehicles: High speed, high acceleration, improved grip

D. NPC Integration and Navigation System

NPC agents are integrated using Unreal Engine's navigation system, which relies on navigation meshes (NavMesh) for dynamic pathfinding [10]. Traffic vehicles follow predefined spline paths, while police and enemy NPCs use dynamic pathfinding based on player position.

The movement of an NPC is determined by:

$$Position_{t+1} = Position_t + Velocity \cdot \Delta t \tag{40}$$

Collision avoidance is implemented using distance-based checks:

$$d < d_{threshold} \Rightarrow ReduceSpeed \tag{41}$$

E. Event System and Gameplay Integration

The system uses an event-driven architecture where gameplay actions trigger events, a widely adopted approach in interactive systems [3].

Examples include:

- Mission start events triggered by location overlap
- Race initiation events triggered by player interaction
- AI behavior changes triggered by player actions

Events are handled using Blueprint event graphs, ensuring synchronization between subsystems.

F. Optimization Techniques

To maintain real-time performance, several optimization strategies are implemented.

1) Level Streaming: The world is divided into smaller segments, and only nearby segments are loaded into memory. Level streaming is a standard technique in open-world systems [7].

2) Level of Detail (LOD): Multiple versions of 3D models are used. The complexity of the model decreases with distance:

$$LOD = f(distance) \tag{42}$$

3) Occlusion Culling: Objects not visible to the camera are excluded from rendering calculations, reducing GPU workload.

4) Dynamic NPC Management: The number of active NPC agents is dynamically adjusted based on system performance to maintain stable frame rates.

G. System Integration and Data Flow

The entire system operates as an integrated pipeline:

$$Input \rightarrow GameLogic \rightarrow AI \rightarrow Physics \rightarrow Rendering \tag{43}$$

Each subsystem communicates through event triggers and shared data structures, ensuring synchronization and consistent system behavior.

VII. Results and Discussion

This section evaluates the performance, behavioral accuracy, and overall system effectiveness of the proposed open-world driving simulation. The evaluation is conducted based on runtime performance, NPC behavior validation, and gameplay system analysis.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

A. Experimental Setup

The system was tested on a mid-range hardware configuration to simulate realistic user conditions. The testing environment consisted of:

- Processor: Intel i5 / Ryzen 5 class CPU
- GPU: Mid-range GPU (GTX 1650 equivalent)
- RAM: 8–16 GB
- Engine: Unreal Engine 5 (Lumen and Chaos enabled)

The map size was constrained to a moderate open-world environment with multiple active NPC agents to evaluate system scalability.

B. Performance Metrics

The performance of the system was measured using frame rate (FPS), system stability, and responsiveness.

1) Frame Rate Analysis: The average frame rate observed across different zones is shown below:

- Slum Area (High Density): 42–50 FPS
- City Center (Moderate Density): 48–60 FPS
- Industrial Zone (Low Density): 55–65 FPS
- Highway (Sparse Environment): 60+ FPS

The decrease in FPS in high-density areas is primarily due to:

- Increased NPC count
- Collision calculations
- Rendering complexity

2) Latency and Input Response: Input latency was observed to be minimal due to real-time input mapping in the engine. The response time for player controls remained consistent across different environments, ensuring smooth gameplay interaction.

C. NPC Behavior Evaluation

NPC behavior was evaluated based on realism, responsiveness, and adaptability.

1) Traffic AI Performance: Traffic NPCs demonstrated consistent state transitions using FSM. Vehicles:

- Stopped at obstacles
- Maintained forward motion in free paths
- Avoided collisions in most scenarios

The FSM-based model ensured low computational overhead, allowing multiple NPCs to operate simultaneously.

2) Police AI Evaluation: Police NPCs implemented using Behavior Trees showed dynamic behavior patterns:

- Successful detection of player violations
- Initiation of chase sequences
- Adaptive pursuit behavior based on player movement

However, limitations were observed in:

- Pathfinding accuracy in dense environments
- Occasional delayed reaction in high-speed scenarios

3) Enemy AI Performance: Enemy racers exhibited aggressive behavior, including overtaking and blocking. The AI maintained competitive gameplay but lacked advanced prediction mechanisms for player movement.

D. Gameplay System Evaluation

The gameplay system was evaluated based on progression, engagement, and system integration.

1) Economic System Analysis: The economic model successfully created a progression loop:

- Players earned money through missions
- Upgrades provided noticeable gameplay benefits
- Progression felt balanced and rewarding

2) Risk and Restart Mechanism: The restart system introduced a high-risk factor, increasing player engagement. Players were required to:

- Strategically manage resources
- Improve driving performance
- Adapt to increasing difficulty levels

3) Day-Night Gameplay Variation: The day-night system significantly impacted gameplay:

- Day: Structured gameplay with traffic and missions
- Night: High-intensity racing and AI interaction

This variation enhanced replayability and gameplay diversity.

E. System Limitations

Despite successful implementation, the system has certain limitations:

- Limited scalability for large-scale maps
- Simplified AI decision-making models
- Absence of learning-based NPC adaptation

F. Discussion

The results indicate that the proposed system effectively integrates AI, narrative progression, and open-world simulation. The hybrid AI model provides a balance between performance and realism, making it suitable for real-time applications.

However, further improvements are required in advanced AI modeling and optimization for large-scale environments.

VIII. Conclusion and Future Work

This paper presented the design and development of an Indian-themed open-world urban driving and racing simulation system implemented using Unreal Engine 5. The primary objective of the work was to create a system that combines realistic urban traffic simulation, AI-driven NPC behavior, and narrative-based progression within a unified framework.

The proposed system successfully integrates a modular layered architecture consisting of input management, game logic, artificial intelligence, world simulation, and rendering subsystems. This separation of concerns enables scalability and maintainability while supporting real-time interaction.

A key contribution of this work is the implementation of a hybrid NPC behavior model that combines Finite State Machines (FSM) and Behavior Trees. FSMs were effectively used for lightweight traffic simulation, ensuring low computational overhead, while Behavior Trees enabled dynamic and context-aware decision-making for complex agents such as police

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

and enemy racers. This hybrid approach achieved a balance between performance efficiency and behavioral realism.

The system also introduces a progression-based gameplay model driven by an economic system and risk-based mechanics. The transition from rickshaw driving to competitive street racing provides a structured gameplay loop, while the restart mechanism introduces strategic decision-making and enhances player engagement. Additionally, the day–night cycle dynamically alters gameplay conditions, contributing to variation and replayability.

Experimental evaluation demonstrates that the system maintains stable performance across different environment densities, with acceptable frame rates even in high-NPC scenarios. NPC agents exhibit consistent behavior patterns, and the integration of AI with gameplay systems results in a cohesive and interactive experience.

Despite these achievements, the current system has several limitations. The NPC behavior models are primarily rule-based and do not incorporate adaptive learning mechanisms. Pathfinding accuracy and collision handling can be further improved, particularly in dense traffic scenarios. Additionally, the current implementation is limited in scale and does not fully explore large open-world environments.

Future work will focus on enhancing the system through the integration of advanced artificial intelligence techniques such as reinforcement learning for adaptive NPC behavior. Further improvements will include the expansion of the game world, implementation of multiplayer functionality, and refinement of physics and collision systems. The incorporation of data-driven AI models can also enable more realistic and personalized gameplay experiences.

In conclusion, this work demonstrates the feasibility of developing a culturally contextualized open-world driving simulation that integrates AI, narrative progression, and economic systems. The proposed framework can serve as a foundation for future research and development in game AI and simulation-based systems.

REFERENCES

[1] B. Bontchev, “Intelligent Adaptation of Difficulty and NPC Behavior in Serious Video Games,” IFAC-PapersOnLine, vol. 57, no. 1, pp. 1–6, 2024.
[2] I. Millington and J. Funge, *Artificial Intelligence for Games*, 2nd ed. Boca Raton, FL, USA: CRC Press, 2009.
[3] S. Rabin, *Game AI Pro: Collected Wisdom of Game AI Professionals*. Boca Raton, FL, USA: CRC Press, 2014.
[4] Epic Games, “Unreal Engine 5 Documentation,” [Online]. Available: https://docs.unrealengine.com/
[5] M. Buckland, *Programming Game AI by Example*. Plano, TX, USA: Wordware Publishing, 2005.
[6] A. Champandard, “Behavior Trees for Next-Gen Game AI,” in *Game Developers Conference (GDC)*, 2007.
[7] R. Smith, “Open-World Game Design and Simulation Techniques,” IEEE Computer Graphics and Applications, vol. 38, no. 2, pp. 34–43, 2018.
[8] J. Togelius et al., “Search-Based Procedural Content Generation: A Taxonomy and Survey,” IEEE Transactions on Computational Intelligence and AI in Games, vol. 3, no. 3, pp. 172–186, 2011.
[9] D. Silver et al., “Mastering the Game of Go with Deep Neural Networks and Tree Search,” Nature, vol. 529, pp. 484–489, 2016.
[10] Unity Technologies, “NavMesh and Pathfinding Systems in Game Development,” [Online].