Group – 10

Syed Sakibul Haque -1620298042

Naima Siddika Toha – 2013506642

Ssaima Arafin Smrity – 1913061042

Md. Saidur Rahman Antu - 1911512042

Reinforced Racer: An AI-Based Racing Simulator

[[1]](#footnote-1)

***Abstract*— Reinforced Racer is a type of simulation based racing game which can explores Reinforcement Learning (RL) in any kind of gaming environments. The game change control from human players to AI agents. Also required them to learn optimal driving strategies to navigate 2D racetracks. Our project system is using Deep Q-Learning (DQN) in a custom Gymnasium- compatible environment featuring ray-based sensors. Dynamic reward mechanisms, and checkpoint- based progression also included. The work also outlines the architecture, pipeline, and performance metrics of our AI agents. We also aim to do 3D Unity-based environment in the future work.**

***Keywords*— Reinforcement Learning, Deep Q-Learning (DQN), Game Development, AI simulator, Gymnasium, Unity 3D.**

# I. INTRODUCTION

Artificial Intelligence (AI) has been making groundbreaking advancements across various industries, and the field of game development is no exception. In recent years, the use of AI for training autonomous agents in simulated environments has gained popularity, particularly through Reinforcement Learning (RL). Reinforcement Learning is a type of machine learning where an agent learns to make decisions by interacting with an environment, receiving feedback in the form of rewards or penalties.

The application of RL in racing games offers an exciting opportunity to simulate real-world challenges such as navigation, obstacle avoidance, decision-making under pressure, and efficiency optimization. These aspects closely resemble problems in robotics and autonomous vehicle navigation, making racing simulations an ideal testbed for AI research and experimentation.

**Reinforced Racer** is our attempt to explore this intersection of AI and gaming. Instead of players manually controlling the vehicles, our simulator requires users to design and train AI agents capable of completing racetracks autonomously. The project provides an interactive platform to visualize how RL algorithms learn and adapt in real-time.

Our initial prototype focuses on a 2D racetrack environment, where we designed a custom simulation using the Gymnasium framework. The environment includes ray-based sensors for perception, checkpoint systems for progress tracking, and dynamic reward functions to guide the learning process. Using Deep Q-Learning (DQN), a widely used algorithm in RL, we trained agents to complete laps while avoiding collisions and optimizing speed.

Beyond the 2D version, our long-term vision involves scaling the simulation to a 3D environment built with the Unity engine. This will not only enhance the visual realism of the game but also introduce more complex challenges for the RL agents, such as physics-based dynamics, 3D perception, and more nuanced decision-making. The integration of AI models into such realistic simulations brings us a step closer to real-world autonomous driving research.

Through Reinforced Racer, we aim to demonstrate how reinforcement learning can be applied creatively in interactive environments while making the learning process both visual and engaging. The project stands at the intersection of computer science, artificial intelligence, and digital entertainment, and showcases how educational tools and games can be merged to foster interest in AI technologies.

# II. Related Work

Reinforcement learning (RL) in simulated environments has been widely explored, and several studies offer valuable insights relevant to Reinforced Racer.

1. **Custom Environments in Game Engines**

A tutorial guide on creating a **custom RL environment in Unreal Engine 4** demonstrates how game engines can be adapted into Gym-compatible simulators. It outlines how to expose observations, actions, and reward signals from UE4 to RL agents, enabling training within visually rich environments. This is conceptually similar to our plan to integrate Unity and highlights technical considerations such as bridging engine-native API calls with Python-based RL frameworks.

1. **Physics Engine Comparison**

Kaup et al. conducted a comprehensive review of **nine physics engines** (Brax, Chrono, Gazebo, MuJoCo, ODE, PhysX, PyBullet, Webots, Unity), evaluating their performance, flexibility, and compatibility with RL research .They noted that while **MuJoCo** excels in speed and accuracy, **Unity** offers strong usability and accessibility—though it lags in fidelity and scalability. This supports our decision to use Unity down the line due to ease of integration and visual clarity, while acknowledging its limitations.

1. **Unity-Based Vehicle Control**

While not RL-specific, blog posts like **“Car Controllers for Unity 3D”** provide detailed overviews of implementing driving mechanics, steering, throttle, and braking via wheel colliders. These practical resources inform our prototype vehicle controllers in Unity before RL integration, reducing engineering effort and improving agent-environment fidelity.

1. **Autonomous Driving with Deep RL**

Literature reviews such as “Deep Reinforcement Learning for Autonomous Driving Systems” (IJFMR, 2024) and “Deep Reinforcement Learning in Autonomous Car Path Planning and Control: A Survey” summarize state-of-the-art DRL techniques—including DQN, PPO, SAC, and continuous control—applied to perception, path planning, and control tasks. These surveys consistently highlight the need for hybrid simulations combining perception, physics, and decision-making—all of which are reflected in Reinforced Racer’s architecture and future 3D extension.

# III. System Overview

The architecture of Reinforced Racer is structured into two key phases:

**Phase 1 – 2D Game Simulation with Deep Q-Learning**

In the first stage, we developed a 2D racing simulation using Python and Gymnasium. The goal was to build a simplified, yet realistic environment for training agents. The racetracks are procedurally generated or manually designed, featuring curves, narrow paths, and checkpoint systems.

**Ray-Based Sensors**: The agent uses raycasts in multiple directions to measure the distance to obstacles, similar to LiDAR in real cars.

**Checkpoint System**: The track contains a series of checkpoints that must be reached sequentially.

**Reward Function**: The agent receives positive rewards for reaching checkpoints, maintaining speed, and avoiding crashes. Negative rewards are assigned for collisions or going off track.

**Phase 2 – Planned 3D Game Using Unity Engine**

Our second phase involves building a 3D version of the game in Unity. The goal is to load the trained DQN model or retrain using Unity ML-Agents, adding realism to the simulation with physics-based motion, 3D sensors, and camera views.

Benefits of Unity include:

Visual realism for demo and education

Physics engine for better driving dynamics

Cross-platform compatibility

# IV. Implementation Details

**Training Environment**

**State Space**: Sensor readings (ray distances), current speed, angle to next checkpoint

**Action Space**: Discrete actions – turn left, turn right, accelerate, decelerate

**Frameworks Used**: Python, PyTorch, NumPy, Gymnasium

**Deep Q-Learning Model**

We used a classic DQN architecture:

**Input Layer**: Processes state vector

**Hidden Layers**: Three fully connected layers with ReLU activation

**Output Layer**: Q-values for each action

**Experience Replay**: Random batches from past experiences

**Target Network**: A secondary network updated periodically for stable training

**Epsilon-Greedy Strategy**: Balances exploration and exploitation

V. Conclusion

Reinforced Racer offers an innovative blend of reinforcement learning and game development. By creating a customizable 2D racing environment and training intelligent agents using DQN, we demonstrate how AI can be applied to real-time navigation and decision-making problems. Our roadmap includes expanding into 3D with Unity, enhancing the game’s realism and educational potential.

The project is not only a technical achievement but also a platform for learning, showcasing how simulation, machine learning, and interactive design can merge into a compelling AI-driven experience.

.

# Acknowledgment

We would like to express our sincere gratitude to our project supervisor, teammates, and lab instructors for their continuous support and encouragement. Special thanks to the online open-source communities behind Gymnasium, PyTorch, and Unity for providing valuable tools and documentation.

References

1. V. Mnih et al., “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
2. OpenAI Gym: <https://www.gymlibrary.dev/>
3. Unity ML-Agents Toolkit: <https://github.com/Unity-Technologies/ml-agents>
4. R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 2018.

1. [↑](#footnote-ref-1)