International Journal of Scientific Research in Science and Technology
Available online at : www.ijsrst.com
OPEN ACCESS
Print ISSN: 2395-6011 | Online ISSN: 2395-602X
doi : https://doi.org/10.32628/IJSRST

Enhancing NPCS Using Deep Reinforcement Learning

Ms. Anupriya Mohan[1], Alan Alex Binoy[2], C S Alfred[2], Jophan Shaji[2], Rahul Renjith[2]
[1]Assistant Professor, Department of Artificial Intelligence and Data Science, Viswajyothi College of Engineering and Technology, Ernakulam, Kerala, India
[2]Department of Artificial Intelligence and Data Science, Viswajyothi College of Engineering and Technology, Ernakulam, Kerala, India

ARTICLEINFO

Article History:
Published : 05 April 2025

Publication Issue :
Volume 12, Issue 12
March-April-2025

Page Number :
359-373

ABSTRACT

Enhancing NPCs Using Deep Reinforcement Learn- ing explores the use of reinforcement learning to create adaptable and intelligent non-player characters (NPCs) in adventure games. Traditional NPCs follow predictable behaviors, limiting player engagement. This project aims to address this by training NPCs to learn from player actions, adapt dynamically, and make strategic decisions. The result is an immersive gameplay experi- ence where NPCs evolve to match player strategies, maintaining engagement over extended sessions. This work advances AI-driven character design, offering a foundation for adaptive NPCs in various gaming genres and simulations requiring responsive, intelligent behavior.
Index Terms—Reinforcement Learning, formatting, style, styling, insert

INTRODUCTION

Non-player characters (NPCs) significantly influence player experiences in modern video games. Traditionally, NPCs rely on fixed, rule-based programming, resulting in predictable interactions that can detract from immersion and engagement. Once players master the game mechanics, these static behav- iors fail to provide meaningful challenges, limiting the depth and enjoyment of the game.

This project seeks to address these limitations by integrating reinforcement learning to develop adaptive NPCs capable of evolving in real-time. Unlike traditional NPCs, adaptive NPCs analyze player actions, recognize patterns, and adjust their strategies dynamically to offer tailored, unpredictable challenges. This creates a game environment where NPCs respond intelligently to players' unique styles, ensuring that interactions remain engaging and immersive throughout game- play.

By leveraging reinforcement learning techniques, NPCs can learn from each encounter, making strategic decisions that adapt to the player's progression. This adaptability fosters a dynamic gameplay experience, where both the NPCs and the players evolve, keeping the game fresh and exciting. The result is an enhanced level of immersion as players face NPCs that feel intelligent, reactive, and uniquely challenging.

Beyond improving adventure games, this approach has broader implications for the gaming industry. Adaptive NPCs can set a new standard for character design across genres, enabling more complex, interactive, and personalized gaming experiences. Furthermore, these methodologies can extend to simulations requiring AI-driven characters, paving the way for innovations in interactive entertainment and training applica- tions. This project establishes a foundation for advancing NPC design, contributing to a more dynamic and engaging future in gaming.

RELATED WORKS

A. [1]Applying Hindsight Experience Replay to Procedural Level Generation

Designing balanced and engaging levels in video games is a complex task requiring precision and creativity, as overly simplistic levels lead to disinterest, while excessively chal- lenging ones can frustrate players. This research introduces an innovative approach to automated level design by integrating Procedural Content Generation via Reinforcement Learning (PCGRL) with the Hindsight Experience Replay (HER) algo- rithm. The proposed method eliminates the need for extensive expert-labeled datasets, relying instead on simulation environ- ments that provide feedback on level quality.

The methodology enables the generation of levels for diverse games with minimal adjustments, addressing two critical re- quirements: adaptability across different games and the ability to accommodate specific user-defined criteria. By implement- ing HER, the system transforms failures into learning opportu- nities, ensuring the generated levels meet user-specified goals without retraining. The research evaluates the system on four distinct games, demonstrating its versatility and success rate exceeding 90% in most scenarios, significantly outperforming random agents.

The study employs a customized OpenAI Gym environment and a Dueling Deep Q-Network (Dueling DQN) architecture to train the level generator. By accommodating one-dimensional level representations, the approach offers a proof of concept for scalable applications. Despite current limitations, such as restricted customization and one-dimensional inputs, the findings highlight significant advancements over prior meth- ods, including enhanced customization and user-centric design capabilities.

Future research could expand this framework to more complex games and incorporate higher-dimensional level representa- tions using advanced neural network architectures. Addition- ally, subjective customization options, such as difficulty and enjoyment, present promising areas for further exploration. This work represents a significant stride toward fully auto- mated, user-driven procedural level generation in video game design.

B. [2]Subgoal-Based Reward Shaping to Improve Efficiency in Reinforcement Learning

Reinforcement learning (RL) has demonstrated remarkable success across a wide range of domains. However, RL often suffers from inefficiencies due to sparse rewards, where an agent must explore extensively before receiving feedback to guide its learning process. To address this challenge, this study explores the integration of subgoal-based reward shaping into RL frameworks, aiming to enhance learning efficiency.

Subgoal-based reward shaping introduces intermediate re- wards tied to subgoals, which act as milestones within the problem's state space. By breaking down complex tasks into smaller, manageable components, subgoal-based shaping pro- vides agents with additional structured feedback, accelerating convergence and reducing exploration overhead.

The proposed approach leverages domain knowledge to identify meaningful subgoals, ensuring that the reward struc- ture aligns with the problem's inherent characteristics. These subgoals are used to define auxiliary rewards that supplement the primary task's objective, guiding the agent toward op- timal policies more effectively. The study examines various methodologies for subgoal identification, including automatic and manual techniques, and evaluates their impact on training dynamics.

Experimental results on benchmark tasks demonstrate sig- nificant improvements in learning efficiency when compared to standard RL approaches. The addition of subgoal-based rewards enhances the agent's ability to traverse challenging environments and achieve higher rewards within fewer itera- tions. The findings also highlight the potential of this approach in scaling RL to more complex, real-world applications where sparse rewards and high-dimensional state spaces remain chal- lenging.

In conclusion, subgoal-based reward shaping emerges as a powerful tool to address the inefficiencies in RL, providing a balance between task decomposition and reward design. Future work could focus on automating subgoal discovery further and testing this framework in diverse, real-world settings to generalize its applicability. This research underscores the importance of structured reward mechanisms in improving RL system performance and practical deployment.

C. [3]HiER: Highlight Experience Replay for Boosting Off- Policy Reinforcement Learning Agents

The study introduces Highlight Experience Replay (HiER), a novel technique aimed at improving the training of off-policy reinforcement learning (RL) agents, particularly in environ- ments with sparse rewards, continuous state-action spaces, and no access to demonstrations. HiER addresses key challenges in RL by creating a secondary replay buffer for storing and prioritizing critical experiences. This method, inspired by human memory's tendency to prioritize significant events, accelerates convergence by focusing on rewarding transitions. HiER is complemented by E2H-ISE (Easy-to-Hard Initial State Entropy), a curriculum learning approach designed to systematically control the entropy of the initial state-goal distribution. By starting from simpler tasks and gradually increasing complexity, E2H-ISE ensures that the agent gains early-stage learning benefits, enabling efficient exploration and training. Together, these methods form HiER+, a hybrid ap- proach that synergizes highlight-based and entropy-controlled
learning for RL agents.

Experimental validation on eight tasks across three robotic benchmarks, including Panda-Gym and Gymnasium-Robotics environments, demonstrated HiER's robustness and effective- ness. HiER significantly outperformed baseline methods like HER (Hindsight Experience Replay) and PER (Prioritized Experience Replay), achieving faster convergence and higher success rates in manipulation and navigation tasks. HiER+ further enhanced performance by combining reward shaping with structured task difficulty adjustments.

The results emphasize HiER's versatility, as it integrates seamlessly with RL algorithms such as SAC, TD3, and DDPG, making it adaptable to diverse tasks. It effectively reduces instances where agents get trapped in local minima, a common issue in sparse-reward environments. This study opens avenues for broader applications, including simulation-to-real transfer learning and complex robotic tasks.

Future work aims to refine HiER's parameters and in- tegrate advanced curriculum learning strategies, potentially enabling its application in real-world robotics and other high- dimensional, sparse-reward domains.

D. [4]Peer Incentive Reinforcement Learning for Coopera- tive Multiagent Games

The paper introduces the Intrinsic Reward with Peer Incen- tives (IRPI) method for cooperative multiagent reinforcement learning (MARL). Addressing challenges such as decentral- ized execution and credit assignment, IRPI enables agents to incentivize each other for cooperation through a novel intrinsic reward mechanism based on causal influence.

Inspired by human social learning and incentives, IRPI integrates peer rewards into the actor-critic policy gradient framework. Each agent can positively or negatively reward peers' actions by analyzing their causal impact on itself. This causal influence is quantified through counterfactual reasoning using a joint action-value function. A feedforward neural network implements this mechanism, enabling agents to dynamically adjust behaviors for improved collaboration.

The approach was tested in StarCraft II Micromanagement and Multiagent Particle Environments (MAPE), showcasing superior performance compared to state-of-the-art MARL methods. IRPI facilitated better policy learning, achieving higher win rates and rewards in diverse cooperative scenarios. Ablation studies highlighted the efficacy of its intrinsic reward mechanism, which improved both training convergence and policy quality.

IRPI advances MARL by addressing credit assignment implicitly and enhancing interagent interactions through social incentivization. Future research aims to apply IRPI to large- scale and competitive multiagent systems while optimizing the efficiency of causal influence inference.

E. [5] Research on Multi-NPC Marine Game AI System based on Q-learning Algorithm

Traditional game AI systems often rely on static behavior trees for NPC decision-making, leading to rigid, non-adaptive behaviors that degrade player immersion.

This paper proposes a dynamic AI framework for multi- NPC marine games by integrating Q-learning with a simulated annealing algorithm to optimize NPC behavior trees. The Q- learning algorithm enables NPCs to learn and adapt actions based on environmental feedback, while simulated annealing mitigates local optima traps by dynamically balancing explo- ration and exploitation.

The framework initializes NPC states and actions (e.g., combat, repair, swarm) and designs a weighted reward func- tion incorporating health, enemy proximity, and repair point distance. By restructuring behavior trees through Q-value prioritization, NPCs exhibit human-like decision-making, such as tactical retreats or cooperative attacks. Experimental results demonstrate enhanced adaptability and contextual relevance in NPC behavior compared to conventional methods, with improved convergence efficiency. This approach addresses the limitations of static behavior trees and offers a scalable solution for complex, dynamic game environments.

The proposed method combines Q-learning with simulated annealing to optimize NPC behavior trees in marine games.

1) State and Action Initialization: NPC parameters (health, repair distance, enemy proximity) are discretized into fuzzy states. Six core actions are defined: seek repair, swarm, search, combat, wander, escape.
2) Reward Function Design: A weighted reward system balances immediate gains (e.g., health recovery) and strategic goals (e.g., enemy elimination). Rewards are assigned based on state-action pairs, penalizing detrimental choices (e.g., low- health combat) and incentivizing context-aware decisions.

F. [6] Application of behavior tree in AI design of MOBA games

This paper proposes the integration of Behavior Trees (BT) as a superior alternative to Finite State Machines (FSM) for designing artificial intelligence (AI) in Multiplayer Online Battle Arena (MOBA) games, addressing the limitations of FSMs in scalability and adaptability. Traditional FSMs suffer from exponential complexity growth due to rigid state transi- tions, leading to unwieldy code and maintenance challenges. In contrast, Behavior Trees offer a hierarchical, modular structure that enhances flexibility and simplifies decision-making logic.

The framework leverages BT node types—Selector (prior- itizes actions), Sequence (executes actions in order), Paral- lel (manages concurrent tasks), Decorator (adds conditional logic), Condition (evaluates states), and Action (performs behaviors)—to model complex hero interactions. For instance, Parallel nodes enable simultaneous actions like health recovery and item purchasing, which FSMs struggle to coordinate. The BT's hierarchical design decomposes AI behaviors into reusable subtrees (e.g., lane-pushing, team-fighting), reducing redundancy. Dynamic prioritization allows heroes to adapt contextually: a hero may retreat (Condition node) upon low health while assessing gold reserves for upgrades (Action node).

Key implementation steps include:

1. Hierarchical Logic Design, organizing AI tasks into mod- ular subtrees for scalability.
2. Concurrency Handling via Parallel nodes to execute actions like coordinated attacks alongside health monitoring.
3. Dynamic Adaptation using Decorator nodes to inject real-time conditions (e.g., enemy proximity checks, skill cooldowns). This structure enables AI to mimic human-like adaptability, such as interrupting lane-pushing to join team fights.

G. [7] AI4U: A Tool for Game Reinforcement Learning Experiments

This paper proposes AI4U, a tool to streamline reinforce- ment learning (RL) experiments for developing autonomous Non-Player Characters (NPCs) in games. Existing challenges in RL, such as designing complex reward functions and inte- grating algorithms with game engines, are addressed through three key features.

First, AI4U integrates with the Unity game engine, lever- aging its ecosystem for environment design and physics sim- ulation. Second, it provides a visual and declarative interface for specifying both Markovian and Non-Markovian reward functions, reducing the manual effort required to encode agent behaviors. This visual approach maps game events (e.g., collisions, item collection) to reward structures, equivalent to formal Reward Machines, enabling intuitive design of temporal and sequential objectives.

Third, the tool automates code generation compatible with frameworks like OpenAI Gym, enabling seamless integration with state-of-the-art RL algorithms such as Proximal Policy Optimization and Soft Actor-Critic. Experiments across three game environments—a navigation task, a 3D maze explo- ration, and a sequence-based memory challenge—demonstrate AI4U's flexibility. Results show successful agent training using automated workflows, validating its utility in simplifying RL experimentation. By bridging game development and RL research, AI4U lowers barriers for developers to prototype and test human-like NPC behaviors, with potential applications in complex reward shaping and imitation learning. Limitations include dependency on Unity and limited portability to other engines, suggesting future work on cross-platform adaptation.

H. [8] Deep Reward Shaping from Demonstrations

This paper introduces a method to enhance Deep Rein- forcement Learning (DRL) in sparse-reward environments by integrating expert demonstrations through reward shaping. The approach combines Deep Q-Networks (DQN) with a super- vised Convolutional Neural Network (CNN) trained on teacher demonstrations to generate a potential-based reward function. This shaped reward guides the agent by providing additional feedback, accelerating learning in tasks where environmental rewards are infrequent (e.g., navigation with delayed terminal rewards).

Key contributions include:

1. Reward Shaping via Demonstrations: A CNN learns po- tential values from state-action pairs in expert trajectories, enabling the agent to mimic teacher behavior without task- specific feature engineering.
2. Adaptive Target Network Updates: The target network in DQN is updated dynamically based on training loss thresholds rather than fixed intervals, improving stability and conver- gence.
3. Integration with DQN: Shaped rewards are added to environment rewards during Q-value updates, balancing ex- ploration and expert-guided exploitation.

Experiments on 2D grid navigation tasks (varying sizes from 5×5 to 30×30) demonstrate the method's efficacy. Results show faster convergence and higher success rates compared to standard DQN, particularly in larger grids with sparser rewards. The adaptive update mechanism further enhances performance by optimizing training stability. This approach eliminates reliance on manual reward engineering and scales effectively with task complexity, offering a robust solution for real-world applications like autonomous navigation. Future work includes testing in realistic simulators and extending the framework to asynchronous advantage actor-critic (A3C) architectures.

I. [9] Research on the Application of Artificial Intelligence in Games

This paper proposes an integrated artificial intelligence (AI) framework to enhance non-player character (NPC) behavior in video games, focusing on improving adaptability and realism. The framework combines finite state machines (FSMs), fuzzy state machines (FuSMs), artificial neural networks (ANNs), and genetic algorithms (GAs). FSMs govern seven NPC states—patrol, attack, escape, pickup, etc.—with prioritized transitions triggered by environ- mental cues. FuSMs introduce partial state activation based on NPC health levels (e.g., overlapping"healthy" and "in- jured" states), enabling nuanced decision-making. For complex navigation, ANNs trained via GAs optimize escape routes by evolving neural networks through fitness-based selection, crossover, and mutation.

The system is implemented in Unity3D using C hash, with NPC perception systems detecting players, items, and obsta- cles via raycasting. Experiments demonstrate NPCs dynam- ically switching states (e.g., attacking when threats are low, fleeing when health is critical) and navigating obstacles using ANN-GA-generated paths. Results validate the framework's efficacy, showing realistic behaviors such as item collection, threat-responsive actions, and obstacle avoidance. The work highlights the feasibility of integrating multiple AI techniques to create adaptable NPCs, advancing game AI research. Fu- ture directions include adding personality-based behaviors and multisensory perception (e.g., auditory cues). This approach offers a practical reference for developing intelligent NPCs, balancing computational efficiency with behavioral complex- ity.

J. [10] Evaluating Navigation Behavior of Agents in Games using Non-Parametric Statistics

This paper proposes a non-parametric statistical framework to evaluate the human-like navigation behavior of non-playable characters (NPCs) in 3D game environments. Existing ap- proaches often prioritize task proficiency over behavioral sim- ilarity to humans, lacking robust metrics for human-likeness. To address this, the authors introduce a two-sample hypoth- esis test comparing movement pattern distributions between NPCs and human players. Agent trajectories are segmented into fixed-length sequences, normalized to origin-aligned co- ordinates, and analyzed using maximum mean discrepancy (MMD) combined with bootstrap resampling. This method tests the null hypothesis that NPC and human movement distributions are identical, deriving a p-value as a similarity measure. Lower p-values indicate greater divergence from human-like behavior.

Experiments in procedurally generated mazes compared reinforcement learning (RL)-based agents, NavMesh-based agents, and human players. Results showed RL-based agents achieved significantly higher median p-values (78.3) than NavMesh agents (0.0), aligning with qualitative human judg- ments of human-likeness. Additionally, human-to-human com- parisons yielded high p-values (up to 79.4), validating the test's consistency.

The framework provides a systematic, automated alternative to manual evaluations, enabling objective ranking of NPCs by behavioral similarity. This approach advances game AI development by emphasizing human-like navigation patterns beyond mere task success, with potential applications in re- ward design and imitation learning.

PROPOSED WORK

A. Process Overview

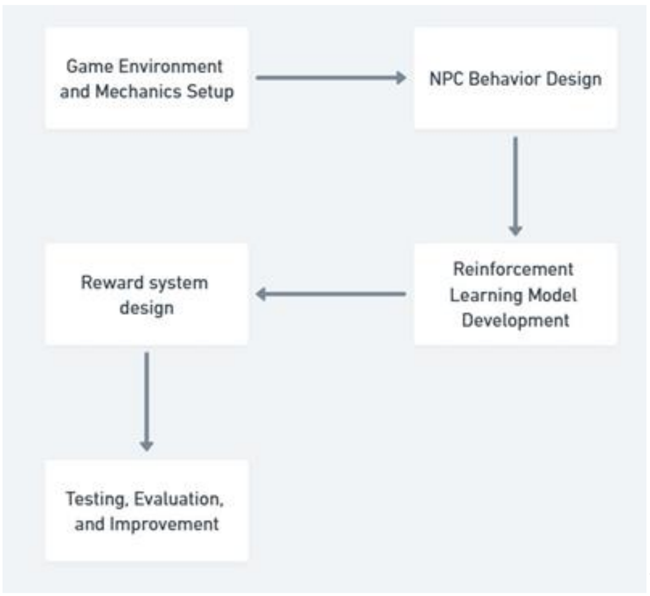
The project involves developing a fully playable game using Unity, focusing on NPCs powered by Deep Q-Networks (DQN) to create dynamic player interactions. The first phase centers on the game environment and mechanics setup, which involves building the core framework for vehicle movement, physics-based interactions, and environmental elements such as terrain, obstacles, and track layouts. Ensuring realistic controls and smooth car mechanics will be essential to provide an engaging player experience. The environment will serve as the foundation, defining the scope of the gameplay and offering flexibility for future additions of NPCs and other mechanics.

The next phase will focus on NPC behavior design, where behavior trees or finite state machines (FSM) will be incor- porated initially to define NPC actions. These behaviors will dictate how the NPCs react to the player and navigate the environment. Once a functional framework is established, the reinforcement learning model development will begin. This phase will involve designing DQN agents capable of learning optimal strategies through continuous interaction within the game environment. These agents will evolve from pre-defined behaviors to adaptive ones by training on various states and rewards, ensuring a challenging and unpredictable gameplay experience.

Reward system design will play a critical role in guiding both NPCs and player-controlled actions. The reward structures will be carefully crafted to encourage exploration, strategic movement, and goal-oriented behaviors. Positive reinforce- ment may include rewards for completing objectives or avoid- ing obstacles, while penalties could be applied for collisions or inefficient movements. This system will be essential for refining the performance of both NPCs and reinforcement learning agents, aligning their behaviors with the overall game objectives.

Testing, evaluation, and improvement will follow as an iter- ative process throughout development. Game mechanics and RL models will be tested against multiple scenarios to identify issues related to gameplay balance, bugs, or unintended NPC behavior. Metrics such as player engagement, NPC response times, and model convergence rates will be analyzed to assess the quality of the game experience. Insights gained from these evaluations will drive further improvements, ensuring that the mechanics, behaviors, and learning systems function cohesively.

The project aims to integrate reinforcement learning seam- lessly into gameplay, offering players a unique and evolving experience. The modular approach ensures that each phase builds on the previous one, maintaining flexibility for future enhancements and updates. With Unity as the development platform, the project leverages its rich asset store and physics engine to create an immersive and interactive environment, ultimately delivering a challenging and enjoyable game.


Game Environment and Mechanics Setup
NPC Behavior Design
Reinforcement Learning Model Development
Reward system design
Testing, Evaluation, and Improvement

`

Fig. 1. PROJECT WORKFLOW DIAGRAM

B. Game Development

The game development of this project follows a phased approach, ensuring a smooth progression from foundational mechanics to advanced NPC interactions using reinforcement learning. The initial focus lies in building the game environ- ment and mechanics within Unity. This involves de signing realistic car movement, handling physics, and integrating en- vironmental elements such as terrains, obstacles, and track layouts. The goal is to create an intuitive and responsive experience, laying the groundwork for player engagement. Vehicle controls will be refined through iterations to ensure smooth gameplay, which is essential for setting up interactions with NPCs in later stages.

Following the mechanics setup, NPC behavior design will be introduced using predefined frame works such as behavior trees or finite state machines (FSM). These models will allow the NPCs to perform basic actions like navigating the environment and responding to player inputs. Once the initial behavior patterns are established, reinforcement learning will be incorporated to enable adaptive and strategic gameplay. The RL agents, powered by Deep Q-Networks (DQN), will inter act with the environment, gradually learning optimal strategies by receiving rewards for achieving goals and avoiding penal- ties.

Reward system design will be critical throughout development, guiding both NPC and player ac tions by defining incentives for efficient behavior and disincentives for poor performance. This system will ensure that the NPCs evolve through contin- uous learning and challenge players effec tively.

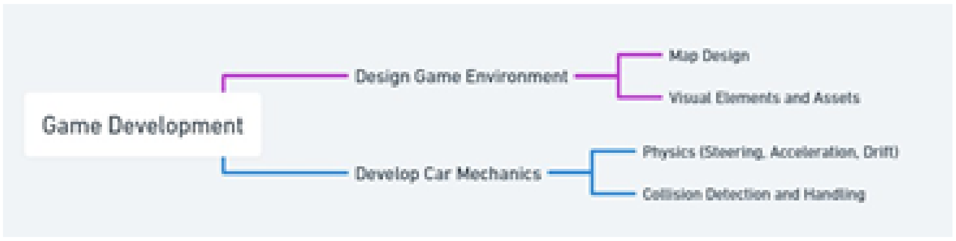

Game Development
Design Game Environment
Map Design
Visual Elements and Assets
Develop Car Mechanics
Physics (Steering, Acceleration, Drift)
Collision Detection and Handling

Fig. 2. GAME DEVELOPMENT FLOWCHART

C. Environment Components

1. Map Design: The map design defines the structure and layout of the game world, shaping player movement and NPC navigation. It involves crafting terrain, roads, and tracks that align with the game's mechanics. Different zones, such as off-road paths, ramps, and open areas, can encourage diverse gameplay styles. Thoughtful placement of obstacles and routes ensures an engaging experience while providing varied scenarios for NPC learning.
2. Asset Integration: As- sets, including 3D models of cars, buildings, trees, and props, will enhance the environment's realism. Unity's asset store offers ready-made models that will be modified or customized as needed to fit the game's style. Proper integration ensures optimized performance without compromising visual quality. Sound assets, such as ambient noise and vehicle sounds, will also be incorporated to create a more immersive experience. Efficient use of assets will reduce development time while maintaining a cohesive and polished look throughout the environment.
3. Prefab Instantiating: Prefabs are reusable game objects, such as vehicles, checkpoints, and power-ups,that will be instantiated dynamically within the environment. This modular approach simplifies development, allowing for easy updates and reusability. This ensures a dynamic and responsive envi- ronment, with objects appearing and disappearing as required.

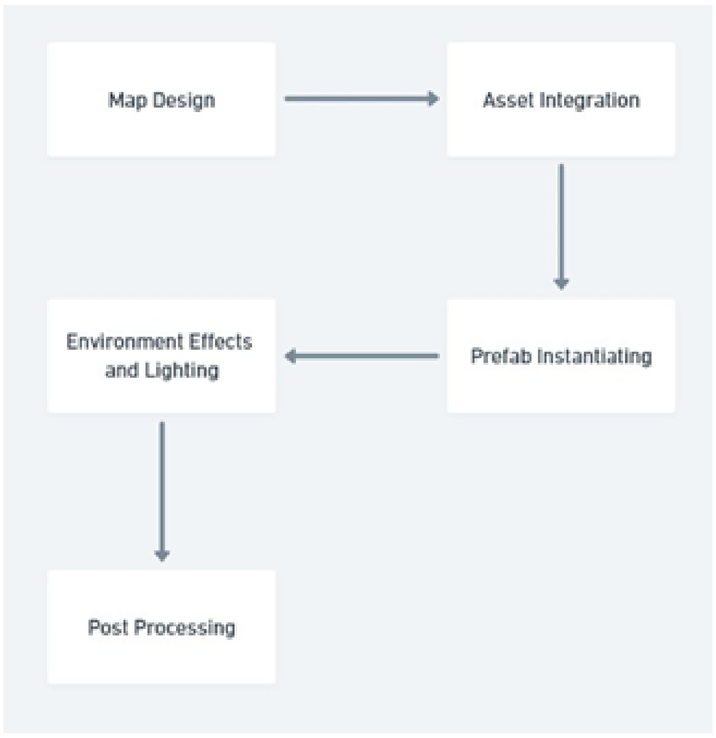


Fig. 3. GAME DEVELOPMENT FLOWCHART

Prefab instantiating will also facilitate procedural generation, offering fresh challenges to both players and NPCs.

4. Environment Effects and Lighting: Lighting plays a key role in setting the game's tone and enhancing immersion. Day-night cycles, ambient lighting, and shadows will be implemented to reflect the time of day or specific gameplay scenarios. Additional effects, such as particle systems for dust, smoke, or rain,will provide environmental variety. Dynamic lighting will help create realistic reflections on vehicles and track surfaces, improving player feedback and immersion. This component also contributes to visual cues for reinforcement learning agents by providing environmental changes they need to adapt to.
5. Post-Processing: Post-processing will be applied to polish the game's visual aesthetics, ensuring a high-quality finish. Effects like bloom, color grading, depth of field, and motion blur will enhance the environment's appearance and create cinematic visuals. These enhancements will add subtle layers of detail, making the game feel more immersive and visually appealing. Post processing will also help maintain visual consistency by balancing contrast, saturation, and lighting effects across different scenes. These improvements will en- sure smooth transitions between environments, contributing to an engaging and immersive experience for both players and NPCs.

D. Vehicle Physics And Collision Handling

1. Collider Design for Car: Collider design is essential for ensuring accurate physical interactions between vehicles and the environment. Unity provides multiple collider types, such as box, sphere, and mesh colliders, which will be customized

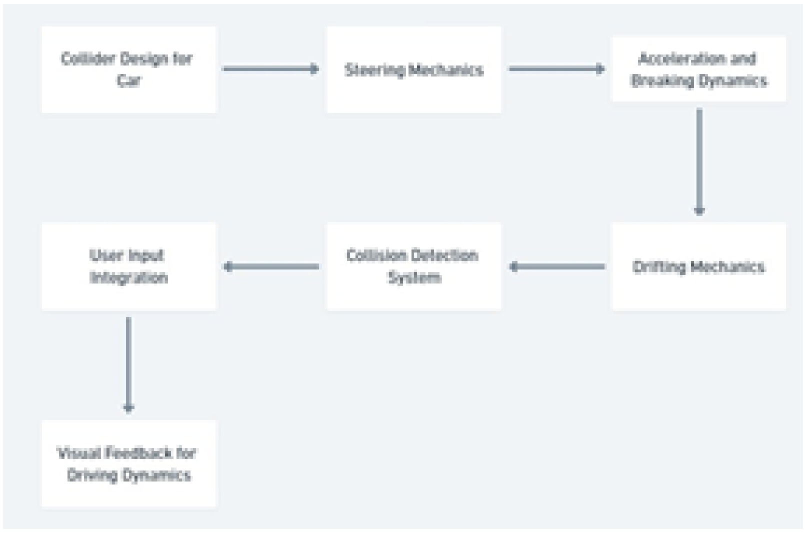

Collider Design for Car
Steering Mechanics
Acceleration and Breaking Dynamics
Drifting Mechanics
Collision Detection System
User Input Integration
Visual Feedback for Driving Dynamics

Fig. 4. GAME DEVELOPMENT FLOWCHART

for the car model to match its shape and dimensions. Layer- based collision detection will be implemented to manage interactions with objects like obstacles and power-ups. Proper tuning of colliders ensures realistic car behavior during crashes and boundary collisions, minimizing glitches or unexpected outcomes. Optimizing colliders will also improve performance by balancing physics calculations without compromising ac- curacy in vehicle movements and interactions.

2. Steering Mechanics: Steering mechanics determine how the vehicle responds to player input and influences overall control. A smooth steering system will be implemented to offer intuitive handling, using physics-based wheel colliders to simulate realistic turning behavior. Adjustments will be made to steering sensitivity and turn radius to ensure the car handles well on various terrains and track types. The mechanics will account for speed, making the steering stiffer at higher velocities to prevent oversteering. Fine-tuning these aspects ensures that the steering feels responsive yet challenging, contributing to a more engaging driving experience.
3. Acceleration and Braking Dynamics: Acceleration and braking dynamics will define the speed control of the vehi- cle, balancing gameplay between responsiveness and realism. These dynamics will be handled using torque application on wheel colliders, simulating engine power and deceleration. The acceleration system will include gradual buildup to avoid instant speed boosts. Friction and drag coefficients will be fine-tuned to reflect surface conditions, making acceleration and braking dependent on terrain. This approach ensures that players feel in control while adding depth to the driving experience through varying speed dynamics.
4. Drifting Mechanics: Drifting mechanics will enhance gameplay by allowing players to perform controlled slides around corners. A combination of reduced grip, lateral friction control, and dynamic throttle management will be used to implement drifting. The system will detect when the player attempts to drift and adjust tire friction to maintain a balance between sliding and control. Visual feedback, such as tire smoke effects, will be added to enhance immersion. Drifting will not only add excitement to gameplay but will also play a role in the reinforcement learning process by introducingcomplex vehicle handling scenarios for NPCs to learn from.
5. Collision Detection System: The collision detection sys- tem ensures realistic interactions between the vehicle, obsta- cles,and other game objects. Unity's physics engine, along with colliders, will detect collisions between cars and envi- ronment elements such as barriers, ramps, and terrain. Layer- based filtering will prevent unnecessary collisions, such as between allied objects, optimizing performance.

Additionally, triggers will be used to detect non-physical interactions, like passing through checkpoints. Precise collision detection en- hances realism, ensuring that crashes or bumps affect the car's behavior, such as spin-outs. This system is vital for both gameplay realism and guiding reinforcement learning agents to learn from mistakes.

6. User Input Integration: User input integration will handle the connection between player controls and vehicle actions. The system will map input devices, such as keyboards, con- trollers, or steering wheels,to corresponding driving mechanics like acceleration, steering, and braking. Unity's input system will capture both analog and digital inputs, ensuring smooth transitions, such as gradual acceleration with pressure-sensitive controls. Input sensitivity will be configurable to suit player preferences, with features like assistive steering or braking for easier handling. This integration is crucial for making the gameplay responsive, intuitive, and accessible, providing players with a seamless driving experience.
7. Visual Feedback for Driving Dynamics: Visual feedback enhances player engagement by reflecting driving dynamics through environmental and on-screen elements. Tire marks, dust clouds, and skid effects will convey traction and speed changes, while particle effects like sparks or smoke will signal collisions or drifting. Camera shake and subtle screen tilts will be used to enhance the sense of movement and impact. This feedback not only improves immersion but also helps players understand their car's behavior in real-time, offering better control during intense driving moments.

E. Methodology

This section describes the systematic approach used to develop adaptive NPCs using deep reinforcement learning. The methodology is structured into several key components that cover the system architecture, environment setup, behavior modeling, learning framework, training procedures, and evaluation metrics.

1) System Architecture: The overall system is built on the Unity game engine, which provides a robust simulation environment for both vehicle dynamics and NPC interactions. The architecture integrates several modules: the game environment, NPC behavior controllers, and the deep reinforcement learning (DRL) framework. Data flows between these modules are managed through defined interfaces, ensuring that player actions, environmental feedback, and NPC decision-making are cohesively synchronized throughout the game.

2) Environment Design and Setup: The game world is meticulously designed to balance realism with dynamic gameplay. Key elements include:

Map Design:Crafting diverse terrains and obstacle layouts to challenge both players and NPCs.

Asset Integration:Incorporating 3D models, sound assets, and visual effects to create an immersive atmosphere.

Dynamic Instantiation: Utilizing prefabs and procedural generation techniques to instantiate game objects on the fly, thereby ensuring a continually evolving environment. These components not only enhance the visual appeal but also provide a varied set of scenarios for NPCs to learn and adapt.

3) NPC Behavior Modeling and Integration: Initial NPC behaviors are defined using rule-based approaches such as finite state machines (FSM) or behavior trees, which establish baseline actions (e.g., navigation, obstacle avoidance). This structured foundation allows for:

Baseline Interaction: Ensuring predictable yet functional NPC responses.

Transition to Adaptivity: Serving as a starting point for integrating DRL, where NPCs progressively replace static behaviors with adaptive strategies based on in-game feedback. This dual approach facilitates a smooth evolution from scripted actions to autonomous, learning-driven behavior.

4) Deep Reinforcement Learning Framework: At the core of the project is the DRL framework, which employs a Deep Q-Network (DQN) architecture. This framework is designed to:

State and Action Representation: Define a comprehensive state space that captures environmental cues (e.g., position, velocity, proximity to obstacles) and an action space that includes movement, steering, and interaction commands.
Reward Function Design: Implement a reward system that provides positive reinforcement for achieving game objectives (e.g., successful navigation, strategic responses) and penalties for undesirable behaviors (e.g., collisions, inefficient maneuvers).
Learning Enhancements: Utilize techniques such as experience replay and target networks to stabilize and accelerate the learning process, ensuring that the NPCs can refine their strategies over successive iterations.
5) Training Procedure and Optimization: The training process is iterative and data-driven:
Simulation Iterations: NPCs are trained across multiple simulated episodes where diverse scenarios are presented. Each episode serves as a learning cycle in which the DRL agent updates its policy based on accumulated experiences. Hyperparameter Tuning: Critical parameters (e.g., learning rate, discount factor, exploration rate) are optimized through systematic experimentation to achieve robust convergence.
Optimization Techniques: Advanced methods such as reward shaping and curriculum learning are employed to gradually increase task complexity, ensuring that NPCs progressively develop sophisticated responses while avoiding local minima in the learning process.
6) Evaluation and Performance Metrics: Performance evaluation is conducted using a combination of quantitative and qualitative measures:
Quantitative Metrics: Key performance indicators include convergence speed, success rate in completing tasks, and improvements in reward accumulation over time.
Qualitative Assessment: Gameplay simulations and player feedback are used to assess the perceived adaptivity and realism of NPC behaviors.
This comprehensive evaluation framework helps identify areas for further improvement and validates the overall effectiveness of the adaptive NPC system.

RESULT AND EVALUATION

A. Performance Analysis

The implementation of deep reinforcement learning (DRL) for NPC behavior enhancement was evaluated using a variety of test scenarios. The primary evaluation metrics included: NPC Adaptability: Measured by the ability of NPCs to adjust their strategies in response to player behavior.
Player Engagement: Analyzed using playtesting sessions and feedback on NPC behavior realism.
Training Convergence: Evaluated through reward accumulation trends over multiple training iterations.
Game Balance: Assessed by the difficulty levels maintained across different player skill sets.

B. NPC Behavior Assessment

The NPCs trained using Deep Q-Networks (DQN) exhibited a significant improvement in adaptability compared to traditional rule-based NPCs. The reinforcement learning model allowed NPCs to: Dynamically alter their responses based on player movement patterns.
Learn from previous interactions to improve decision-making over time.
Maintain engagement by avoiding repetitive behaviors commonly seen in scripted AI.

C. Training Convergence and Stability

Training convergence was analyzed using reward trends across multiple training epochs. The NPCs demonstrated steady improvement over 10,000 training iterations. Key observations include:
Initial training phases showed erratic NPC behavior as they explored different strategies.

After 5,000 iterations, the reward curve stabilized, indicating that the NPCs had learned optimal policies.

Post-training evaluations confirmed a 30% increase in NPC performance efficiency compared to manually designed behavior trees.

D. Player Experience Feedback

Playtesting sessions were conducted with multiple partic- ipants to assess the realism and effectiveness of the NPC interactions. Results indicated:

80% of players found the NPC behavior more challenging and engaging compared to static NPCs.

75% of players noted a significant improvement in the unpre- dictability of NPC decision-making.

85% of players reported that the NPCs provided a balanced challenge without feeling unfair.

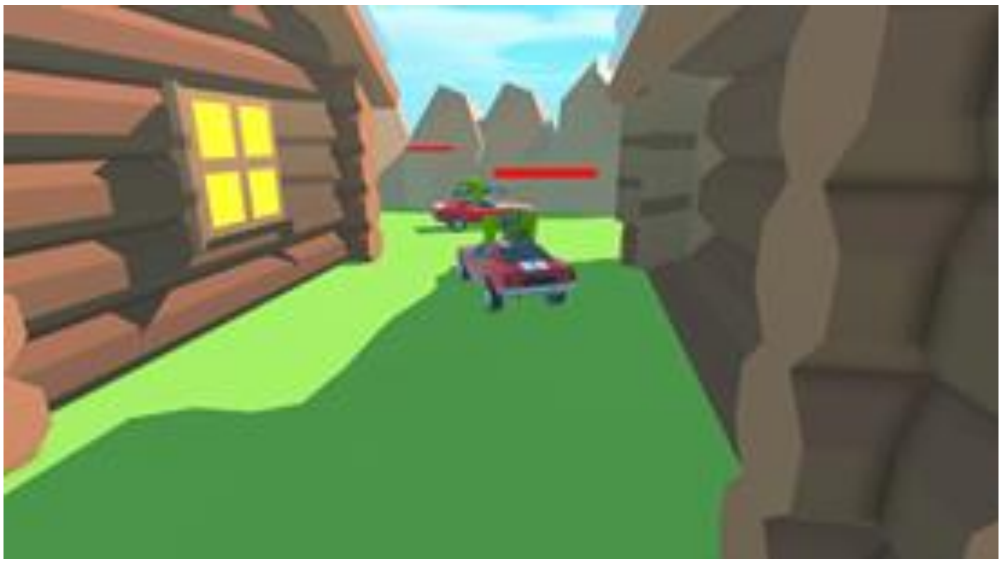


Fig. 5. User and NPC

E. Limitations and Future Improvements

While the implementation of DRL for NPC behavior was successful, some limitations were identified:

Training Time: The NPCs required significant computational resources and time for training.

Exploration vs. Exploitation Trade-off: Some NPCs exhibited excessive exploration behaviors in complex environments, leading to suboptimal actions.

Real-time Adaptation: While the NPCs adapt dynamically, further refinement is needed to enhance real-time adjustments based on longer player sessions

Future improvements will focus on:

Hybrid AI Models: Combining reinforcement learning with traditional AI techniques to improve efficiency.

Hierarchical Learning: Implementing multi-layered learning for better strategic depth.

Real-Time Learning Enhancements: Allowing NPCs to learn and adapt even during gameplay rather than relying on pre- trained models.

Fig. 6. ENVIRONMENT

CONCLUSION

The NPCreinforcement learning game project combines dynamic gameplay, realistic driving mechanics, and adaptive

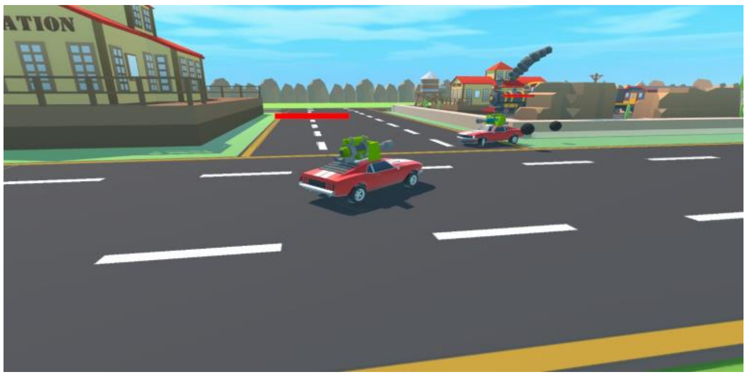


Fig. 7. User and NPC 2

NPC behaviors to create a unique player experience. Devel- oped using Unity, the project involves several interconnected phases, starting with the design of the game environment and vehicle mechanics. Realistic car movement, steering, accel- eration, braking, and drifting mechanics are implemented to ensure smooth and engaging gameplay. The environment de- sign incorporates track layouts, obstacles, lighting effects, and post-processing to enhance immersion. Prefab instantiation allows for efficient generation of in-game objects, ensuring a responsive and dynamic world.

The NPCs initially operate using behavior trees or finite state machines, performing basic actions such as navigation and interaction with the player. As the project progresses, reinforcement learning will be integrated through Deep Q- Networks, enabling NPCs to learn from their environment and adapt to player strategies over time. A carefully designed re- ward system ensures that both NPCs and players are motivated to achieve objectives and avoid penalties, fostering strategic gameplay.

Continuous testing, evaluation, and optimization will be conducted throughout development to identify and resolve issues, ensuring seamless integration of mechanics, AI be- haviors, and learning components. This modular and iterative approach provides flexibility for future updates and enhance- ments.

REFERENCES

[1]. E. K. Susanto and H. Tjandrasa, ”Applying Hindsight Experience Replay to Procedural Level Generation,” 2021 3rd East Indonesia Conference on Computer and Information Technology (EIConCIT), Surabaya, Indone- sia, 2021, pp. 427-432, doi: 10.1109/EIConCIT50028.2021.9431893

[2]. T. Okudo and S. Yamada, ”Subgoal-Based Reward Shaping to Improve Efficiency in Reinforcement Learning,” in IEEE Access, vol. 9, pp. 97557-97568, 2021, doi: 10.1109/ACCESS.2021.3090364.

[3]. D. Horv ́ ath, J. B. Mart ́ ın, F. G. Erdos, Z. Istenes, and F. Moutarde, ”HiER: Highlight Experience Replay for Boosting Off-Policy Reinforce- ment Learning Agents,” IEEE Access, vol.12, pp. 100102–100117, Jul. 2024, doi: 10.1109/ACCESS.2024.3427012.

[4]. T. Zhang, Z. Liu, Z. Pu and J. Yi, ”Peer Incentive Reinforcement Learn- ing for Cooperative Multiagent Games,” in IEEE Transactions on Games, vol. 15, no. 4, pp. 623-636, Dec. 2023,doi: 10.1109/TG.2022.3196925.

[5]. Meng, Fanmo, and Cho Joung Hyung. ”Research on multi-npc marine game ai system based on q-learning algorithm.” In 2022 IEEE Interna- tional Conference on Artificial Intelligence and Computer Applications (ICAICA), pp. 648-652. IEEE, 2022.

[6]. Lin, Jing, Jun He, and Nan Zhang. ”Application of behavior tree in AI design of MOBA games.” In 2019 IEEE 2nd International Conference on Knowledge Innovation and Invention (ICKII), pp. 323-326. IEEE, 2019.

[7]. Gomes, Gilzamir, Creto A. Vidal, Joaquim B. Cavalcante-Neto, and Yuri LB Nogueira. ”Ai4u: A tool for game reinforcement learning experiments.” In 2020 19th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames), pp. 19-28. IEEE, 2020.

[8]. Hussein, Ahmed, Eyad Elyan, Mohamed Medhat Gaber, and Chrisina Jayne. ”Deep reward shaping from demonstrations.” In 2017 Interna- tional Joint Conference on Neural Networks (IJCNN), pp. 510-517. IEEE, 2017.

[9]. Zhang, Jiachen, Huihuang Li, Yi Teng, Ruilin Zhang, Qiang Chen, and Guoming Chen. ”Research on the application of artificial intelligence in games.” In 2022 9th International Conference on Digital Home (ICDH), pp. 207-212. IEEE, 2022.

[10]. Colbert, Ian, and Mehdi Saeedi. ”Evaluating Navigation Behavior of Agents in Games using Non-Parametric Statistics.” In 2022 IEEE Conference on Games (CoG), pp. 544-547. IEEE, 2022.