Procedural Track Generation and Reinforcement Learning in a 3D Arcade Racing Game

The Document Title

Ben Beetison

School of Computer Science

College of Engineering and Physical Sciences

University of Birmingham

2024-25

Abstract

This project presents a 3D arcade racing game developed in Unity, featuring procedural track generation and AI opponents trained using reinforcement learning. The game uses a custom physics-based vehicle system designed for smooth, responsive handling, including drifting and boost mechanics to enhance the arcade-style experience.

The procedural track generation system employs computational geometry techniques to create diverse and natural-feeling circuits. A weighted scoring system evaluates track candidates based on factors such as corner flow, straight length, and overall layout, ensuring a balance between challenge and playability. Additional features, such as dynamic height variations, banking, and adaptive track width in tight turns, further enhance the racing experience.

AI opponents are trained using reinforcement learning, adapting to procedurally generated tracks through multi-agent training and curriculum learning. The system optimises agent behaviour by defining observable track elements, control inputs, and reward mechanisms, allowing the AI to develop effective racing strategies in various track configurations.

This project demonstrates the successful integration of procedural content generation and reinforcement learning to create dynamic and engaging racing gameplay. It contributes practical implementations to both AI-driven game development and procedural track design, displaying how these techniques can address the unique challenges of creating ever-changing tracks and adaptive racing opponents in modern game development.

Acknowledgements

Tribal Chief

Abbreviations

BBB	Big Bolo Ben

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

CHAPTER 2

Research and Literature Review

2.1 Critical Evaluation of Methodological Approaches

2.1.1 Procedural Track Generation Methodologies

The evolutionary approach proposed by Loiacono et al. [1] presents a mathematically robust framework for track diversity assessment; however, their implementation in TORCS reveals significant limitations when considering arcade-style racing applications. Their metrics for shape diversity and speed diversity were calibrated within simulation racing parameters that may not adequately translate to arcade racing contexts where exaggerated physics models predominate. Furthermore, their validation methodology primarily engaged experienced simulation racing participants, potentially introducing sampling bias that limits generalisability to arcade racing audiences [1, p. 395].

Nascimento et al.'s [2] chain code algorithm demonstrates quantitative differentiation between track generations (98.5%) but exhibits methodological constraints in evaluating qualitative gameplay considerations. Their approach prioritises mathematical diversity metrics over fundamental racing design principles such as flow, rhythm progression, and strategic depth that constitute essential components of arcade racing design. Additionally, their participant evaluation framework emphasised immediate satisfaction metrics rather than longitudinal engagement assessment—a crucial consideration for procedural content where initial novelty effects may obscure design inadequacies [2, p. 42].

2.1.2 Vehicle Physics Implementation Evaluation

Vejbora's [3] comparative analysis of vehicle physics approaches provide valuable implementation insights but lacks comprehensive performance assessment across hardware configurations. The rolling sphere technique demonstrates compelling gameplay advantages but introduces potential collision handling irregularities that warrant further investigation. Vejbora's evaluation framework employed subjective criteria for "arcade feel" without establishing objective metrics for handling responsiveness or consistency across varied track topographies. This methodological limitation constrains the generalisability of findings to diverse procedural environments [3, p. 18].

Mario Kart

The Mario Kart series employs a simplified yet effective vehicle physics system that prioritises gameplay responsiveness over strict realism. Unlike traditional physics engines that simulate tire grip, weight transfer, and friction dynamics, Mario Kart Wii utilises a predefined wheel movement model that ensures smooth and predictable vehicle handling. Wheels are constrained to a fixed axis relative to the vehicle body, meaning they do not operate independently but rather propagate ground collisions to the entire vehicle structure.

The core of Mario Kart's wheel physics lies in its collision and suspension system. Instead of relying on a physically simulated suspension model, the game determines the vertical positioning of each wheel by extending a downward projection and checking for ground collisions using a specialised KCL collision algorithm. If a collision is detected, the wheel is repositioned along the normal of the surface, keeping the hitbox tangent to the terrain. This ensures the vehicle smoothly adapts to slopes and uneven surfaces without erratic behaviour.

Additionally, a simplified suspension force model adjusts wheel acceleration based on position and movement, contributing to a sense of weight and stability while maintaining the arcade-style responsiveness required for high-speed gameplay. By separating visual representation from physics calculations, Mario Kart successfully maintains intuitive and accessible handling while reducing computational complexity compared to traditional racing simulators.

The Mario Kart physics model serves as an instructive example of how commercial arcade racing games prioritise gameplay feel over physical accuracy, informing our approach to vehicle physics implementation.

2.1.3 Limitations in AI Implementation Research

Ana et al.'s [4] ML-Agents implementation establishes feasibility for reinforcement learning applications in racing contexts but presents methodological limitations in comparative analysis. Their evaluation metrics, particularly win rates against human participants (26.67% for rule-based methods), represent a narrow assessment parameter that fails to account for perceived challenge or entertainment value—essential considerations in arcade racing contexts. Furthermore, their research design does not adequately address generalisation capabilities across varied track layouts, a critical consideration for applications within procedurally generated environments [4, p. 74].

Bengio et al.'s [5] curriculum learning principles, while foundational to contemporary reinforcement learning approaches, predate significant advancements in deep reinforcement learning methodologies. Their framework requires substantial domain-specific adaptation for racing applications, particularly regarding complexity progression definitions in procedural environments where track features interact through non-linear relationships to create difficulty gradients [5, p. 1].

2.1.4 Visual Design Technique Assessment

Lake et al.'s [6] non-photorealistic rendering techniques were developed within hardware constraints that have been superseded by contemporary processing capabilities. Their texture-based methodology minimises computational requirements but introduces limitations in dynamic lighting environments that may affect visual consistency across procedurally generated tracks. Their optimisation approaches demonstrate diminished relevance within modern rendering pipelines capable of processing more sophisticated shader calculations [6, p. 413].

Liao's [7] investigation of modern cel-shading techniques emphasises technical implementation parameters rather than evaluating perceptual impact on player experience and gameplay readability. The research lacks empirical validation through user studies to determine whether proposed enhancements to outline rendering and color transitions improve player performance or track comprehension during high-velocity gameplay scenarios [7, p. 27].

2.2 Interdisciplinary System Integration

2.2.1 Track Generation and Vehicle Physics Integration

The procedural track generation system and vehicle physics model must function as integrated components within a cohesive gameplay framework. Track characteristics directly influence vehicle control model efficacy—sharp cornering sequences necessitate precise drift mechanics, while extended straights require balanced acceleration and terminal velocity parameters. The rolling sphere physics approach selected for this implementation offers particular advantages within procedural environments, providing consistent handling characteristics across varied terrain compared to complex simulation models that may exhibit unpredictable behaviours on procedurally generated geometries.

Research by Vélez-Beltrán et al. [8] indicates that "handling characteristics must be calibrated in direct response to track composition parameters" [8, p. 118], suggesting a bidirectional relationship between track generation algorithms and physics implementation. This integration consideration is inadequately addressed in existing literature, which predominantly examines these systems as discrete components rather than interdependent systems.

2.2.2 Visual Design and Gameplay Mechanics Integration

The cartoon shading system and outline render pass implemented in this project serve functional purposes beyond aesthetic considerations. The simplified visual presentation with pronounced object boundaries enhances track readability during high-velocity gameplay scenarios, directly supporting the arcade physics model that emphasises extreme speeds and exaggerated manoeuvrers. This visual clarity is particularly significant within procedurally generated environments where players cannot rely on established spatial memory for track navigation.

The research by Lake et al. [6] and Liao [7] addresses technical implementation parameters of non-photorealistic rendering but fails to explore how these techniques can be specifically optimised for racing gameplay contexts.

This implementation addresses this research gap by recognising that visual design decisions should function in service of gameplay mechanics, particularly for communicating critical track boundaries and upcoming features that influence player decision-making during race progression.

2.2.3 AI Training and Procedural Content Integration

The reinforcement learning approach for AI opponents presents unique methodological challenges when implemented within procedurally generated environments. While traditional racing AI implementations frequently utilise pre-computed racing lines for static tracks, this implementation requires agents capable of adapting to diverse track configurations. This necessitates a more sophisticated observation space and reward system than would be required for static environments.

The curriculum learning principles established by Bengio et al. [5] acquire additional complexity when applied to procedurally generated content. The implementation must not only progressively increase difficulty parameters but also ensure that agents develop generalisable skills rather than overfitting to specific track patterns. The training infrastructure design, with its capacity to generate new tracks at predetermined intervals, directly addresses this interdisciplinary challenge by compelling agents to develop robust racing strategies applicable across varied environmental conditions.

This intersection between AI training methodology and procedural content generation represents an emerging research domain that remains incompletely explored in current literature. This implementation contributes practical insights to this intersection, demonstrating effective deployment of reinforcement learning methodologies within environments characterised by high variability and procedural generation.

2.3 Research Synthesis and Application to Project

The literature review reveals several key insights that directly inform the project implementation. Procedural track generation techniques have evolved from simple parametric deformation to sophisticated computational geometry approaches, with human validation studies confirming that diverse track layouts enhance player engagement. The arcade game implementation builds upon these findings by employing Delaunay triangulation and Voronoi diagrams to create natural track layouts with appropriate difficulty progression.

For vehicle physics, the research clearly indicates that conventional physics approaches often prioritise realism over gameplay accessibility. The rolling sphere technique identified by Vejbora [3] and the simplified physics model employed in Mario Kart provide compelling evidence for the design choice to abstract vehicle physics in favour of responsive, arcade-style handling.

In the domain of AI opponents, the literature demonstrates the advantages of reinforcement learning over traditional scripted approaches, particularly for adaptability to procedurally generated environments. By implementing curriculum learning principles from Bengio et al. [5], the AI training strategy develops agent capabilities progressively, ensuring robust performance across varied track configurations.

Finally, the non-photorealistic rendering techniques presented by Lake et al. [6], Liao [7], and Nienhaus and Doellner [9] establish a foundation for the implementation of cartoon-style shading and outline rendering, creating a cohesive visual aesthetic that supports gameplay readability while maintaining a distinctive visual identity.

This synthesis of research across procedural content generation, vehicle physics, AI implementation, and stylised rendering provides a comprehensive theoretical foundation for the arcade racing game implementation, informing key design decisions in each component of the system.

CHAPTER 3

Legal and Ethical Issues

CHAPTER 4

System Requirements

CHAPTER 5

Design

This section outlines the design and methodology used in developing the 3D arcade racing game, focussing on the core gameplay mechanics, procedural track generation, and reinforcement learning for AI opponents.

5.0.1 System Overview

This project integrates procedural content generation, physics-based gameplay, and reinforcement learning-based AI to create an arcade-style 3D racing game. The system architecture employs a modular design consisting of three primary subsystems that interact through well-defined interfaces. These subsystems manage the car physics, procedural track generation, and AI opponent behaviour, respectively, each functioning independently while communicating through established protocols. As noted by [**Velez2022**], this type of integration requires careful calibration between track composition parameters and handling characteristics to create a cohesive gameplay experience (p. 118).

5.0.2 System Flow

Gameplay System Flow

The race initialisation process follows a sequential architecture. Upon race commencement, a scene handler constructs the required Unity game objects in a predetermined order. Initially, the track manager is initialised to control procedural track generation, 3D mesh creation, and implementation of the checkpoint system. Subsequently, a floor generation object manages terrain procedural generation. The player car and AI opponent vehicles are then instantiated, followed by the race handler object, which maintains lap counting and position tracking logic.

The race manager requires checkpoint data from the track manager to function properly. Following complete initialisation, a three-second countdown precedes the official race commencement.

Training System Flow

The training initialisation process mirrors the gameplay system flow with performance-focused modifications. A training handler constructs the scene following similar protocols to the scene handler. It generates a new track via the track manager but omits terrain generation to optimise performance during training. The handler then initialises a race manager and transfers checkpoint information from the generated track.

The system subsequently instantiates a predetermined number of agents equipped with reinforcement learning logic and car control capabilities. During training, track regeneration occurs periodically, necessitating updates to both the race handler and agents with new track configuration data.

5.1 Gameplay Design

5.1.1 Car Physics and Handling Evolution

The vehicle physics system evolved through multiple iterations to establish an optimal balance between realism and arcade-style handling. Each approach presented distinct advantages and limitations in the development process.

Unity's Wheel Colliders

The initial implementation utilised Unity's WheelColliders to simulate realistic vehicle movement. This approach incorporated a Rigidbody cuboid chassis with WheelColliders positioned at each corner. The physics system applied motor torque to the front wheels for acceleration and implemented speed-sensitive steering angles to maintain realistic handling. The suspension system utilised spring and damper settings to enable smooth traversal over uneven terrain.

Additional stability control mechanisms were implemented to counteract oversteering through selective application of braking force to rear wheels. A handbrake system facilitated controlled drifting by reducing the sideways friction of rear tyres.

This approach was constrained by insufficient control precision. The WheelCollider based system required extensive parameter adjustments to achieve arcade-style responsiveness, including modifications to friction curves, torque application, and damping settings. These adjustments frequently resulted in unpredictable behaviour when attempting to balance realism with playability. Furthermore, debugging issues related to wheel placement, force application, and collider interactions proved inefficient.

Individual Wheel Forces

The subsequent implementation employed individual forces applied to the four corners of the Rigidbody cuboid that represents the vehicle. Forward force was applied relative to the cuboid's orientation to the designated drive wheels at the rear, creating more realistic acceleration as force diminished when approaching maximum velocity.

Steering was controlled via forces applied parallel to the right direction of the front wheels. These forces were calculated using the dot product of wheel velocity and the direction resisting sliding. Once the velocity in the slipping direction was determined, an opposing force was applied to counteract the slipping force, enabling directional control. Wheel orientation adjusted based on player input, interpolating between desired and current direction over a predetermined time-frame.

Suspension physics were implemented as a third force vector for each wheel in the upward direction, utilising a spring-dampener model. This created a semi-realistic suspension system managing weight distribution and balance shifts during turning.

This approach was ultimately abandoned due to the extensive parameter testing required. Vehicles frequently lost control and the suspension-induced body roll caused vehicles to overturn during sharp cornering. The complexity exceeded requirements for an arcade-style implementation and presented potential performance limitations for reinforcement learning with multiple agents.

5.1.2 Rolling Sphere Design

The final and most effective approach utilised a sphere as the primary physics object with the car model visually attached to it. This design decision substantially simplified the physics model while maintaining responsive control and facilitating the arcade-style handling required for the game. This approach aligns with [3] research on vehicle physics implementations, which highlights the advantages of simplified physics models for arcade-style racing games (p. 18).

Core Design Concept

The approach separates the vehicle's visual representation from its physics representation to provide responsive control while maintaining natural movement characteristics. A sphere collider with a Rigidbody functions as the foundation for all movement physics and collisions, with the visual car model positioned above the sphere. This separation allows simplified physics calculations while preserving visual fidelity.

Unlike previous approaches attempting to model realistic vehicle physics, this design deliberately abstracts the physics to prioritise gameplay experience over simulation accuracy. This design choice aligns with arcade racing conventions emphasising responsive control rather than physical realism, whilst simultaneously reducing computational requirements.

Movement Design

Movement is implemented through direct force application to the sphere in the forward direction of the car model. Sphere rotation in alternative directions is constrained to minimise unintended lateral movement. Rather than simulating tyre grip and slip angles, steering is accomplished through rotation of the car body, directly influencing the movement force direction. Steering sensitivity adjusts dynamically based on velocity to maintain consistent responsiveness across all speeds. Maximum velocity is regulated through gradual force reduction rather than imposing hard limits.

Grip System

The grip system was designed to balance arcade responsiveness with appropriate mass and momentum representation. Lateral forces oppose sideways movement to prevent excessive sliding, with grip strength increasing proportionally with speed to maintain control at high velocities. This creates natural cornering behaviour without implementing complex tyre physics. Collisions temporarily reduce grip to convey impact sensations without causing complete loss of control.

Terrain Adaptation

The system adapts to uneven terrain while maintaining consistent control characteristics. Downforce is applied to enhance traction on inclined surfaces and prevent unintended aerial trajectory when transitioning from ascending to descending gradients. The car model automatically aligns with terrain inclination through smoothing algorithms to prevent jarring transitions on variable terrain. This creates natural suspension effects without modelling complex spring physics. The car body exhibits slight outward roll during cornering and pitch during acceleration and deceleration. Wheel models move dynamically with body position changes and rotate in response to steering input.

Drift and Boost Mechanics

The simplified physics model facilitated implementation of signature arcade racing mechanics. The drift system temporarily reduces lateral grip whilst increasing steering sensitivity in the current steering direction. Visual effects including smoke and tyre marks provide feedback to the player, with smoke volume indicating drift duration and potential boost accumulation. The boost mechanism rewards skilled drifting through three distinct levels achieved via progressively longer drifts, with boost duration increasing proportionally to drift time. This introduces strategic depth as players must choose between drifting for boost acquisition versus maintaining optimal racing lines.

Design Reasoning

The rolling sphere approach provided significant advantages over previous methods, particularly for an arcade racing game. The simplified physics model improves performance with multiple AI-controlled vehicles and offers more predictable, controllable behaviour supporting the arcade handling style. The method provides an appropriate balance between realism and gameplay prioritisation, establishing a flexible foundation for gameplay mechanics.

This approach shares key principles with the physics system employed in the Mario Kart series, which prioritises gameplay responsiveness over strict realism. As observed in Mario Kart Wii, the use of a predefined wheel movement model ensures smooth and predictable vehicle handling whilst reducing computational complexity compared to traditional racing simulators. Like our implementation, Mario Kart successfully maintains intuitive and accessible handling by separating visual representation from physics calculations.

The design successfully achieves the intended balance between accessibility and depth, creating a vehicle that exhibits responsive and enjoyable control characteristics whilst maintaining sufficient realistic momentum and mass properties to provide satisfying feedback.

5.2 Procedural Track Generation Design

The procedural track generation system constitutes a fundamental component of the game, creating unique racing environments for both players and AI agents. Rather than implementing pre-designed tracks, the system dynamically generates new circuits for each race, enhancing replayability and providing diverse training scenarios for AI agents.

5.2.1 Early Track Generation Approaches

Prior to establishing the final methodology, simpler approaches to procedural track generation were explored. The initial concept focused on deforming basic geometric shapes, primarily ovals, to create varied racing circuits.

Oval-Based Generation

The first track generation prototype utilised a parametric oval as its foundation. This approach created a perfect oval using parametric equations, sampled points along the oval at regular intervals, applied noise displacement to each point, and connected the displaced points with splines using Bezier knots to form the track.

For noise displacement implementation, Perlin noise functions were applied to offset each point from its original position. By modifying noise parameters including frequency and amplitude, this method could produce tracks with varying complexity levels.

Limitations of the Oval Approach

Whilst initially promising, this approach revealed several significant limitations. Despite substantial noise application, tracks maintained predominantly oval characteristics, lacking the diversity required for engaging gameplay. The random noise frequently generated unrealistic or imbalanced track sections, with some turns being excessively sharp or awkwardly positioned, occasionally resulting in point overlap. Adjustment of noise parameters provided insufficient control over the final configuration, complicating the process of tailoring generation toward specific track styles. The generated tracks frequently lacked the natural progression found in well-designed racing circuits, with transitions between sections exhibiting abrupt or arbitrary characteristics.

The oval approach demonstrated that more sophisticated geometric foundations would be necessary to create compelling procedural tracks. This realisation prompted exploration of computational geometry techniques including Delaunay triangulation, which offered a more robust foundation for generating diverse and playable track layouts.

The limitations identified in the initial approach directly informed the development of the more advanced graph-based generation system, emphasising the importance of mathematical foundations that inherently create valid and interesting circuit patterns rather than relying on simple deformation of basic shapes.

5.2.2 Final Design Approach

The track generation implementation follows a geometry-based methodology where tracks originate as a collection of points distributed in two-dimensional space. The system utilises Delaunay triangulation and Voronoi diagrams to create natural track layouts and converts the raw geometry into smooth splines to achieve a natural racing experience. This approach was developed after considering the limitations of simpler methods, much like the findings of [1], whose research on track diversity assessment demonstrated the need for more sophisticated geometric approaches beyond basic deformation techniques (p. 395).

Each track incorporates three splines representing the left, centre, and right boundaries, where the left and right splines maintain consistent width along straights but expand for corners, creating a natural racing environment. Elevation changes are implemented through Perlin noise to enhance racing dynamics, complemented by banking on corners to reinforce the arcade styling. Multiple generation strategies have been implemented, including grid-based, circular, and random point distribution to provide track variety. The implementation of diverse track layouts is supported by [2] research showing that track diversity is quantifiably beneficial for player engagement (p. 42).

Track Features

Tracks incorporate clearly demarcated start/finish lines with designated starting positions and a grid formation for competing vehicles. Track boundaries are implemented along the left and right splines to contain vehicles within the racing surface. Visual supports are positioned beneath the track to create the appearance of elevation above the uneven terrain.

Checkpoint System

Tracks feature equidistant checkpoints that enable race progress tracking, lap counting, and position determination. These checkpoints also provide observational data for AI training to improve convergence on reliable agent behaviour.

5.2.3 Design Considerations

Racing flow represents a critical factor in arcade racing track design, requiring balance between skill requirements and entertainment value. Tracks must incorporate an appropriate distribution of straights and curves that present challenges without demanding excessive skill for navigation. During testing, some generated tracks featured excessively tight corners requiring extreme deceleration, highlighting the importance of corner radius calibration. Tracks must exhibit visual clarity to allow drivers to anticipate upcoming features. Maintaining efficient generation algorithms for runtime track creation and designing tracks that facilitate effective learning environments are essential considerations for performance optimisation during training.

This approach to track design complements the car physics system by providing dynamic environments that showcase the arcade-style handling characteristics whilst presenting varied challenges to both human players and AI opponents.

5.3 Visual Design and Shader Effects

The physics and track generation systems are complemented by a distinctive visual style implemented through custom shader techniques. This visual approach supports the arcade racing experience by establishing a bold, readable aesthetic that emphasises gameplay elements.

5.3.1 Cartoon Shading System

The game implements a custom cartoon-like shader system that diverges from realistic rendering in favour of a stylised appearance. This approach builds upon the non-photorealistic rendering techniques established by [6], whilst adapting them to modern hardware capabilities (p. 413). The system utilises stepped lighting transitions rather than smooth gradients to create a hand-drawn aesthetic, alongside colour banding to reduce gradation levels and create more defined colour regions. The shading enhances contrast between illuminated and shadowed areas to increase visual impact and employs simplified texturing with minimal colour detail to maintain the cartoon aesthetic. These techniques align with [7] research on modern cel-shading approaches, though with an emphasis on improving gameplay readability during high-speed racing scenarios (p. 27).

5.3.2 Outline Render Pass

To enhance the cartoon aesthetic and improve gameplay readability, a custom outline render pass was implemented to highlight important objects including vehicles, track edges, and obstacles. This creates clear visual separation between elements in the scene, improving visibility of track features during high-speed racing and reinforcing the cartoon style through line-art definition.

The outline effect is achieved through a dedicated render pass that detects edges based on depth and normal discontinuities. The system applies adjustable outline thickness and colour, then composites the outlines over the main render to produce the final image.

5.3.3 Design Rationale

This visual approach aligns with the arcade-style gameplay as the simplified visuals and pronounced outlines facilitate player processing of visual information at high velocities. As observed in the research by [6] and [7], non-photorealistic rendering techniques can serve functional purposes beyond aesthetic considerations. In our implementation, the cartoon aesthetic with pronounced object boundaries enhances track readability during high-velocity gameplay scenarios, directly supporting the arcade physics model that emphasises extreme speeds and exaggerated manoeuvres.

The cartoon aesthetic complements the exaggerated physics of the arcade handling model and establishes a memorable and distinctive visual identity that differentiates the game from simulation racing titles. This visual clarity is particularly significant within procedurally generated environments where players cannot rely on established spatial memory for track navigation.

The visual design decisions reinforce the game's emphasis on entertainment and accessibility rather than strict realism, creating a cohesive experience where graphical presentation, physics behaviour, and gameplay mechanics all support the core arcade racing concept.

5.4 AI Opponents and Reinforcement Learning

The third core component of the game system is the AI opponent design based on reinforcement learning. Rather than employing traditional scripted approaches or waypoint following, this project implements a machine learning system where AI drivers develop racing proficiency through an iterative training process, adapting to any procedurally generated track.

5.4.1 Reinforcement Learning Architecture

The reinforcement learning system implements the Proximal Policy Optimization (PPO) algorithm, which establishes an effective balance between exploration and exploitation whilst providing sample efficiency and stability. The implementation utilises Unity's ML-Agents framework with a custom-designed agent architecture specifically tailored to the racing domain. This approach draws upon the research by [4], which established the feasibility of reinforcement learning applications in racing contexts using the ML-Agents framework (p. 74).

5.4.2 Agent Design

Each AI car agent comprises three primary components: a TrackSensingSystem that provides environmental awareness through ray-casting, a CarAgent that processes observations, receives rewards, and makes driving decisions, and a CarControlScript that translates agent decisions into physics-based vehicle movements. This script is identical to the one used for the player vehicle. Agent operation is coordinated by a TrainingManager that oversees the overall training process and curriculum.

This modular architecture separates perception, decision-making, and actuation, enabling focused development and evaluation of individual components. The CarAgent functions as the central processing unit, receiving sensor data and race state information, then producing driving commands based on its learned policy.

5.4.3 Observation Space

The agent's observation space encompasses 40 continuous variables providing comprehensive environmental and vehicular state information. This includes normalised distances to track boundaries for spatial awareness, rate of change in distance values for situational dynamics, normalised vectors from the vehicle to the next checkpoint and upcoming track positions and width, normalised current speed, drift status, and additional vehicle dynamics to enhance the agent's understanding of vehicle behaviour.

This observation design establishes an appropriate balance between immediate tactical awareness and strategic understanding, providing sufficient information for the agent to develop complex racing behaviours.

5.4.4 Action Space

The agent's action space consists of three continuous actions controlling acceleration, braking and steering, plus one discrete action for handbrake operation during drifting manoeuvres.

This hybrid action space enables precise control whilst maintaining intuitive mapping to conventional racing controls. Action interpolation between decision points facilitates smooth transitions in control inputs, preventing abrupt or unrealistic movements that could destabilise vehicle physics.

5.4.5 Reward System Design

The reward system was designed to encourage skilled racing behaviours through a combination of immediate feedback mechanisms and long-term incentives:

The core progression metric centres on checkpoint achievement, with continuous small rewards for reducing distance to the next checkpoint, larger one-time rewards for checkpoint passage, and substantial rewards for complete circuit completion.

This tiered reward structure creates a clear gradient from immediate progress to strategic lap completion, effectively guiding the agent's learning process.

Beyond basic progression, the system rewards advanced driving techniques by providing continuous rewards proportional to velocity, rewards for appropriate orientation toward the next checkpoint, and bonuses for maintaining optimal racing lines.

The agent incurs penalties for boundary collisions, backward movement, failure to maintain forward progress, or erratic steering behaviour.

Each reward component is calibrated to balance immediate tactical decisions with long-term strategic behaviour, creating a learning environment that promotes both race completion and driving proficiency.

5.4.6 Training Infrastructure

Training Manager

The TrainingManager component serves as the orchestration layer for the entire training process. It generates new tracks at predetermined intervals to ensure generalisation capability, initialises multiple simultaneous agents, and maintains coordination across track reconstructions for all agents.

This centralised approach enables efficient training across multiple agents whilst maintaining consistent environmental conditions for all learning entities.

Curriculum Learning

A sophisticated curriculum learning strategy progressively increases task difficulty as agents demonstrate improvement, applying the fundamental principles established by [5] (p. 1). This includes beginning with extended periods between track reconstructions and gradually reducing intervals as agents develop basic proficiency. This approach increases reconstruction frequency as agents acquire knowledge, training them to adapt to environmental variations.

Initially, agents undergo complete reset upon collision to establish basic forward movement skills. As proficiency increases, agents occasionally continue after collisions to develop recovery capabilities. At advanced training stages, vehicles continue after all collisions with only penalty application rather than complete reset.

These mechanisms ensure training stability whilst developing agents' ability to recover from suboptimal situations rather than simply avoiding them. The curriculum learning principles acquire additional complexity when applied to procedurally generated content, requiring agents to develop generalisable skills rather than overfitting to specific track patterns. This addresses a limitation identified in [4] research, which did not adequately address generalisation capabilities across varied track layouts (p. 74).

The training environment supports multiple simultaneous agents, enabling parallel experience collection for accelerated learning. Agents operate on identical physics layers configured to prevent self-collision whilst maintaining a shared environment, facilitating efficient parallel training.

CHAPTER 6

Implementation

CHAPTER 7

Testing

CHAPTER 8

Project Management

CHAPTER 9

Evaluation

CHAPTER 10

Conclusion

Bibliography

[1] Daniele Loiacono, Luigi Cardamone, and Pier Luca Lanzi. "Automatic Track Generation for High-End Racing Games Using Evolutionary Computation". In: *IEEE Transactions on Computational Intelligence and AI in Games* 3.3 (2011), pp. 245–259. DOI: 10.1109/TCIAIG.2011.2163687.

[2] Anna Nascimento et al. "Procedural Race Track Generation using Chain Code Algorithm". In: *Proceedings of SBGames 2021*. 2021, pp. 37–46. DOI: 10.1109/SBGames52902.2021.9677298.

[3] Tomáš Vejbora. "Procedural Track Generation for Racing Games". MA thesis. Czech Technical University in Prague, 2020.

[4] Maria Ana, Paulo Silva, and Eduardo Oliveira. "Reinforcement Learning for Race Car Driving in Unity ML-Agents". In: *International Journal of Computer Games Technology* 2023 (2023), pp. 65–82. DOI: 10.1155/2023/5683279.

[5] Yoshua Bengio et al. "Curriculum Learning". In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ICML '09. 2009, pp. 41–48. DOI: 10.1145/1553374.1553380.

[6] Adam Lake et al. "Stylized Rendering Techniques for Scalable Real-Time 3D Animation". In: *Proceedings of the 1st international symposium on Non-photorealistic animation and rendering* (2000), pp. 413–421. DOI: 10.1145/340916.340936.

[7] Shunning Liao. "Modern Approaches to Cel-Shading in Real-Time 3D Graphics". In: *Journal of Computer Graphics Techniques* 12.1 (2023), pp. 22–39.

[8] Jonathan Vélez-Beltrán, Raúl Hernández-Palma, and Rodrigo Quintero-Téllez. "Integration of Physics-Based Handling Models with Procedural Track Generation in Racing Games". In: *Proceedings of the International Conference on Computer Games Technology*. 2022, pp. 112–124.

[9] Marc Nienhaus and Jürgen Doellner. "Edge-Enhancement - An Algorithm for Real-Time Non-Photorealistic Rendering". In: *Journal of WSCG*. Vol. 11. 1-3. 2003.