Autonomous driving in Super Tux Kart using Reinforcement Learning
Mid Semester Report

Undergraduate Research Project for module LU3IN013 at Sorbonne University, France
January 2025 - May 2025

Authored by: Wilson Cedric GENEVIEVE
Safa GUNES
Mahmoud DARWISH
Badr BENHAMMOU

Undergraduate students in Computer Science, Sorbonne University, France
January 2025 - May 2025

Supervised by: Olivier SIGAUD
Computer Science Professor and Machine Learning Researcher
ISIR Robotics Laboratory, Sorbonne University

Abstract. This project aims to develop autonomous driving agents for *Super Tux Kart*, focusing on efficient track navigation, obstacle avoidance, and performance optimization. Special emphasis is placed on visualizing track layouts and refining navigation strategies to enhance agent behaviour before integrating reinforcement learning techniques.

1 Introduction

1.1 Brief Introduction to SuperTuxKart

SuperTuxKart is an open-source kart racing game developed by the SuperTuxKart development team. It offers a variety of dynamic tracks, items, and racing mechanics, making it an ideal platform for both casual gaming and artificial intelligence research. The game is actively maintained, with official resources available for both gameplay and development. The official SuperTuxKart website can be found at https://supertuxkart.net/Main_Page, and detailed documentation on the game's internal structure is available at https://doxygen.supertuxkart.net.

1.2 Illustration of Key Gameplay Elements

To better illustrate the core challenges faced by our autonomous agents, Figure 1 provides a visual example from SuperTuxKart. The image highlights several important gameplay elements that influence agent behaviour:

- **Track Obstacles:** Visible in the image are banana peels and item boxes. Banana peels act as hazardous obstacles that cause the kart to spin out upon contact, while item boxes provide power-ups that may affect the race positively or negatively.
- **Path Dynamics:** The track shown in the image curves gently while narrowing on the right side. This demonstrates the challenge of maintaining a stable trajectory while ensuring the agent avoids hazardous objects.
- **Rival Karts:** The presence of other racers on the track emphasizes the need for strategic positioning and evasive manoeuvrers to avoid collisions.

- **Speed Management:** The speed gauge in the bottom right corner highlights the importance of dynamic speed control, especially when approaching obstacles or navigating curves.

This visual reference reinforces the complexity of the environment in which our agents operate, emphasizing the necessity for adaptive behaviour that combines effective path-following, item avoidance, and performance optimization.

Figure 1: Illustration of key gameplay elements in SuperTuxKart - Screenshot taken from the BlackForest track.

1.3 Contextualization of the Research Field

The development of autonomous agents in video games like SuperTuxKart addresses a growing need for intelligent bots capable of competing with human players or enhancing gameplay dynamics. This demand is particularly relevant in racing games, where effective navigation and decision-making are crucial. While this challenge draws from fields such as artificial intelligence, computer vision, and control theory, it also presents unique constraints and opportunities. Moreover, insights gained from developing these agents can be applicable to real-world autonomous driving scenarios, providing a valuable testing ground in a controlled environment.

1.4 Definition of the Project's Scope

This project focuses on developing multiple racing agents with distinct strategies for achieving optimal performance in SuperTuxKart. The agents are evaluated based on their ability to:

- Maintain stability and precision during turns and curves.
- Avoid obstacles and items that may hinder progress.
- Balance speed control to improve lap times.

While developing these agents, we emphasize practical algorithms that can be adapted for reinforcement learning in future phases of the project.

1.5 Identification of Key Challenges and Motivations

Several challenges were identified during the initial stages of development:

- **Navigational Stability:** Ensuring the agent follows a stable trajectory even in complex track layouts.
- **Obstacle Avoidance:** Developing effective item detection mechanisms to prevent collisions.
- **Performance Optimization:** Achieving faster lap times while maintaining consistent performance.

These challenges guided the design and testing of our racing agents, motivating the development of improved algorithms and visualization tools to better understand track geometry and agent behaviour.

2 Problem Definition and Objectives

2.1 Summary of the Main Research Problem

The primary research problem in this project is to develop autonomous agents that can efficiently navigate dynamic racing environments. The challenge lies in designing agents that balance speed, precision, and adaptability while responding effectively to obstacles, sharp turns, and variable track conditions. Achieving this requires carefully designed algorithms that combine strategic path selection, item avoidance, and performance control.

2.2 Identification of the Project's Primary Objectives

Our project aims to achieve the following key objectives:

- Develop stable and precise navigation strategies to ensure agents remain on optimal racing paths.
- Design effective item avoidance mechanisms that minimize collisions and improve lap times.
- Implement visualization tools to analyze track geometry, identify racing line inefficiencies, and evaluate agent performance.
- We might establish a foundation for future reinforcement learning integration by refining control logic and observation processing.

2.3 Expected Outcomes and Measurable Success Criteria

The success of the project will be evaluated based on the following measurable criteria:

- **Navigation Stability:** Agents should maintain smooth and stable movement throughout the track with minimal erratic behaviour.
- **Item Avoidance Efficiency:** A reduction in the number of item collisions compared to baseline agents.
- **Performance Optimization:** Improved lap times compared to reference racing agents.
- **Visualization Effectiveness:** Clear and accurate visual representations of track layouts, item distributions, and agent trajectories to support development and debugging.

3 Exploration of Key Concepts

3.1 Acceleration Logic

Acceleration logic defines the decision-making process for controlling kart acceleration and braking. For instance, the `EulerAgent` (refer to EulerAgent technical datasheet) emphasizes speed maximization by refining acceleration logic while minimizing drift usage. Balancing acceleration ensures the agent maintains control while maximizing race performance.

3.2 Node Selection Strategies

Node selection strategies are used to identify the optimal path by targeting a track node several steps ahead of the kart's current position. The `MedianAgent` employs this strategy, focusing on an *n-th* node rather than the immediate next node to improve steering stability and reduce abrupt directional changes.

3.3 Peripheral Vision Mechanisms

Peripheral vision mechanisms expand the agent's awareness to detect items or obstacles located outside the kart's forward-facing direction. The `ItemsAgent` integrates peripheral vision to improve obstacle avoidance by identifying potential threats before they directly obstruct the kart's path.

4 Analysis of Provided Scripts

This section provides a condensed overview of the scripts provided in the `src` folder. The original codebase was forked from the repository available at https://github.com/bpiwowar/pystk2-gymnasium. A comprehensive and detailed analysis of all key algorithms, functions, and data structures has been conducted and will be included in the Final Report.

4.1 Summary of Key Files

- **`__init__.py`** — Handles environment registration and configuration, with support for various observation and action wrappers.
- **`definitions.py`** — Defines essential data structures and observation processing logic.
- **`envs.py`** — Establishes core racing logic, observation management, and action control.
- **`pystk_process.py`** — Manages SuperTuxKart's main process flow, track initialization, and environment stabilization.
- **`stk_wrappers.py`** — Implements observation and action wrappers for efficient data flow between agents and the environment.
- **`utils.py`** — Provides utility functions for data transformation, including quaternion-based rotation and coordinate conversions.
- **`wrappers.py`** — Introduces tools such as `SpaceFlattener` and `FlattenMultiDiscreteActions` for observation and action space management.

4.2 UML Diagrams

To summarize the relationships between the various components, two UML diagrams have been included in the Appendix for better readability:

- **Class Diagram:** Outlines key classes, their interactions, and critical data flow mechanisms to enhance understanding of the project's architecture (Refer to Appendix A).
- **Package Diagram:** Provides a high-level view of the project's module organization and their dependencies (Refer to Appendix A).

A comprehensive analysis of the code structure, including in-depth details of these diagrams, has been conducted and will be fully documented in the Final Report.

5 Technical Data Sheets and Analysis

In this section, we provide an overview of the technical data sheets created for each custom agent developed throughout this project. These data sheets are attached as supplementary material for detailed reference. Each sheet outlines the design process, hypotheses, experimental setup, results, and future improvements for the corresponding agent.

Important

Readers are encouraged to consult the detailed technical data sheets provided at the end of this document for comprehensive information, including algorithm designs, performance metrics, and graphical analysis.

5.1 MedianAgent

The MedianAgent was designed with the primary objective of maintaining the kart's position near the center of the track. The motivation for this agent stemmed from the observation that many traditional racing agents failed to adapt effectively to sharp turns and narrow pathways.

The development process involved several iterations, where we hypothesized that selecting an *n-th* node further along the track would provide better trajectory stability. Experiments demonstrated that this strategy improved the agent's steering control in wide curves but struggled in cases with abrupt elevation changes or split pathways.

Key observations include:

- Improved stability on moderately curved tracks.
- Poor adaptation to sharp turns when the n-th node was too distant.
- Effective performance when combined with controlled acceleration adjustments.

Future improvements identified in the data sheet focus on refining the node selection strategy, particularly by incorporating dynamic adjustments based on track curvature and speed.

5.2 EulerAgent - Experimental Phase

The EulerAgent was designed with the goal of maximizing speed while maintaining reasonable track stability. Unlike the MedianAgent, this agent prioritizes aggressive acceleration and minimal drift correction.

Our initial hypothesis was that by eliminating drifting mechanisms and focusing purely on acceleration control, we could improve overall lap times in straightforward tracks. Empirical results revealed that this approach indeed performed better in simple circuits but struggled in highly technical tracks with sharp turns or frequent item hazards.

Key observations include:

- Faster lap times in linear tracks with minimal sharp turns.

- Instability issues at high speeds, particularly during rapid elevation changes.
- Poor item avoidance due to limited peripheral awareness.

Recommended improvements include integrating adaptive speed reduction during complex segments and enhancing item detection strategies.

5.3 ItemsAgent - Under Development

The ItemsAgent is being designed specifically to test item avoidance logic. Inspired by the challenges faced with frequent item collisions during MedianAgent and EulerAgent testing, this agent leverages peripheral vision mechanisms to anticipate and avoid items placed near the kart's path.

Our hypothesis is that by actively monitoring objects in peripheral regions and incorporating dynamic avoidance manoeuvrers, the agent would demonstrate improved resistance to item-based penalties. The results expected are that the ItemsAgent successfully reduces collisions with isolated items, even when multiple hazards are clustered in narrow passages.

Key observations are expected to include:

- Effective avoidance of single items in clear track segments.
- Improved decision-making when balancing avoidance with optimal racing lines.

Future improvements of this agent include enhanced prediction of item trajectories and improved coordination between steering and acceleration adjustments.

6 Additional Implemented Utilities

6.1 TrackUtils and TrackPathWrapper

During the development of our custom agents, we encountered significant challenges in visualizing track data and agent behavior effectively. To address this, we developed two key utilities: `TrackUtils` and `TrackPathWrapper`. These tools were designed to simplify the visualization of track geometry, agent paths, and item locations in 3D space.

Motivation Our primary motivation for creating these utilities was to enable effective analysis of agent behavior and track complexity. Visualizing the track's structure, path nodes, and agent trajectories was crucial for understanding performance limitations and improving racing strategies.

In particular:

- Standard SuperTuxKart visualization options were insufficient for detailed trajectory analysis.
- Key visual insights were necessary to identify unexpected agent behaviors, such as incorrect node selection, poor item avoidance, or inefficient path-following strategies.
- A flexible plotting solution was required to support the development of different agents in varying track configurations.

6.1.1 TrackUtils

The `TrackUtils` module provides the foundation for visualizing the track layout and agent movement. It defines the `TrackVisualizer` class, which leverages Plotly to render 3D plots with the following features:

- Visualization of track centerlines, left and right boundaries.
- Support for plotting agent paths, ensuring that path gaps are managed to prevent inaccurate representations of track connectivity.
- Rendering of key path nodes as markers to highlight track structure.

The visualization capabilities were essential in debugging path-following issues, particularly when refining the MedianAgent's trajectory planning.

6.1.2 TrackPathWrapper

The `TrackPathWrapper` module builds on `TrackUtils` to provide additional functionality specific to plotting agent behaviour:

- Automated extraction of track data from the environment's observation space (`obs`).
- Conversion of local item coordinates to global coordinates for accurate positioning in the visual space.
- Storage of track data, node coordinates, and agent path data as CSV files to facilitate reproducible results and data inspection.

By integrating both modules, we achieved a consistent and reliable method for producing visual representations that highlight:

- The agent's precise movement across the track.
- The position of items relative to the agent's path.
- The alignment of nodes and center paths with the visualized data.

7 Future Works

In future work, our objective is to refine and strategically combine the strengths of our custom agents to achieve optimal performance across various racing scenarios. Each agent developed so far demonstrates unique advantages — the MedianAgent excels in stability, the EulerAgent emphasizes speed, and the ItemsAgent focuses on item avoidance.

Our goal is to implement a system that dynamically selects or combines agent behaviors based on track characteristics, kart conditions, and race objectives. By leveraging the strengths of each approach, we aim to create a more adaptive and versatile racing strategy.

Additionally, we look forward to integrating reinforcement learning techniques to enhance agent decision-making. Through reinforcement learning, agents could autonomously adapt to complex racing environments by learning optimal control strategies based on reward mechanisms. This would allow for improved trajectory planning, better acceleration management, and enhanced obstacle avoidance.

The implementation of this learning-based system will require careful tuning of reward functions and exploration strategies. Future experiments will focus on identifying the best balance between deterministic control algorithms and adaptive learning techniques to maximize performance across diverse track conditions.

8 Conclusion

This project aimed to develop autonomous racing agents capable of navigating complex track environments in SuperTuxKart. Through the design and testing of distinct agents — `MedianAgent`, `EulerAgent`, and `ItemsAgent` — we addressed key challenges such as navigation stability, item avoidance, and speed optimization.

Our visualization utilities provided crucial insights into track geometry, item positioning, and agent behaviour, enabling more informed development decisions.

Future work will focus on refining these agents while introducing reinforcement learning techniques to enhance adaptability and performance. By combining agent-specific strengths and incorporating learning-based strategies, we aim to create a versatile solution capable of thriving across varied racing conditions.

A Appendix: UML Diagrams

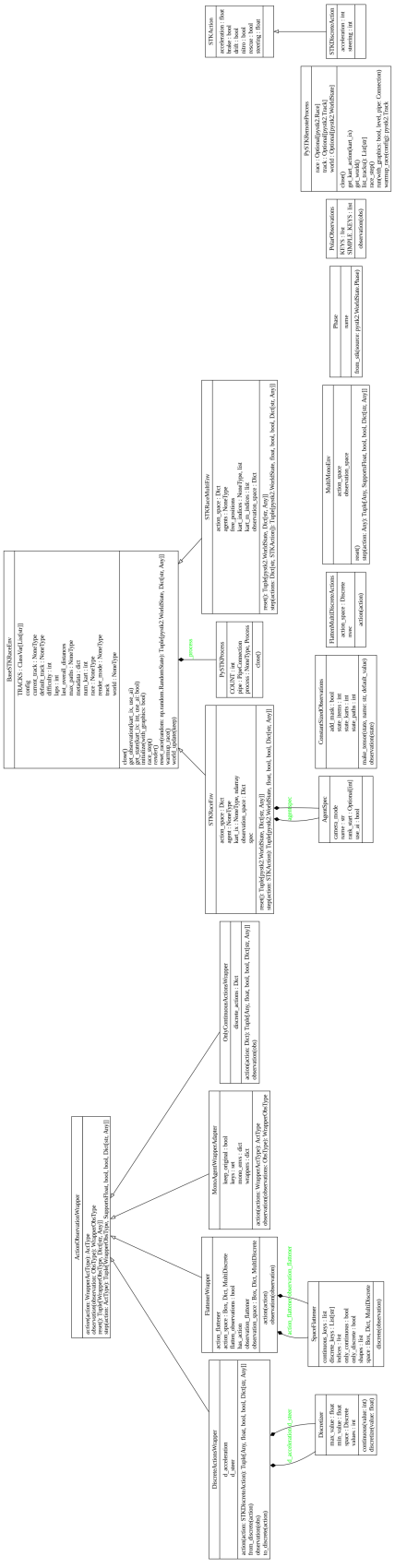

Figure 2: Class Diagram for PySTK2 and Gymnasium Integration (Full Page)

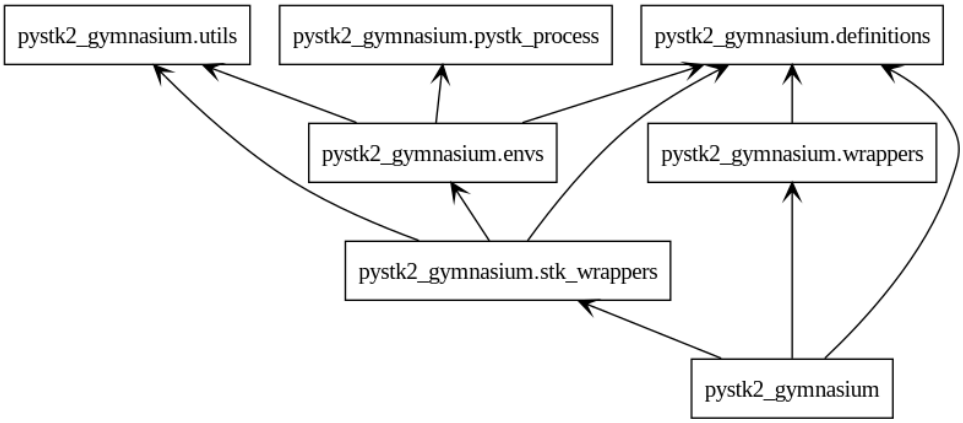

Figure 3: Package Diagram for PySTK2 and Gymnasium Integration