CSE 202: Project Proposal

Group Members:
Akshay Gopalkrishnan
Shresth Grover
Ronit Chougule
Balaaditya Mukundan
Isheta Bansal

Introduction

Mario Kart is a popular video game that combines exciting racing mechanics with strategic decision-making on dynamic tracks. In this project, we model a Mario Kart race as a computational problem by representing the track as a weighted graph. Each section of the track is assigned a terrain type, influencing kart speed and control. Using graph algorithms to find the shortest path, we aim to determine the fastest route around the track while accounting for boosts, obstacles, and terrain variations. We hope our approach will provide a simplified yet engaging way to analyze the challenges and strategies of Mario Kart racing.

What is Mario Kart?

Mario Kart is a video game featuring races between up to 8 players on various tracks. Players can choose from a range of characters, each with unique attributes such as maximum speed and kart control. Races take place on tracks with diverse terrains, including roads, off-road sections, snow, underwater paths, sky tracks, and volcanic areas. These terrains affect the kart's maximum speed—karts travel faster on roads compared to off-road or volcanic surfaces—and influence control, with smoother terrains like roads and underwater areas allowing better handling, while icy surfaces reduce control due to their slippery nature.

Additionally, the tracks include boosts like mushrooms, which provide a short speed boost, and stars, which grant a greater speed boost along with improved control. To add to the challenge, obstacles are scattered throughout the course. These include static obstacles like Thwomps and Fire Bars, as well as dynamic ones, such as Shy Guys and Goombas, which move around the track.

Model

The Mario Kart track is imagined in this project as a 3D grid, with each cell denoting just a small section of the course. The terrain type in each cell, such as ordinary track, grass, out of bounds, or a boost zone, influences the kart's speed. For example, driving on regular track cells maintains a normal pace, whereas karts are slowed down by grass cells and temporarily accelerated by boost zones. If a player falls off the course and goes out of bounds, they must return to the nearest ordinary track zone and will be significantly slowed down. When determining the quickest way to finish a lap, these terrain variations are crucial.

Besides different terrain types, to make the model more representative of Mario Kart conditions, we include stationary obstacles in some of the grid cells. Usually, a player slows down as it hits certain obstacles, which increases the time costs for those cells. The additional time required to navigate around or past obstacles is captured by giving cells with obstacles larger weights. This allows us to replicate actual track conditions and the tactical decisions that players need to make in order to prevent delays.

This model's primary objective is to use weighted graph analysis with no negative edges to determine the fastest route around the track, where the start and end points are the same. We can determine the route that takes the least amount of time to complete a lap by adding up the travel costs across each edge we visit in a route.

Possibilities for Increased Model Complexity

To further increase the complexity of our problem, we can introduce some of the dynamic and probabilistic elements present in Mario Kart environments.

To start, we can consider the obstacles in a Mario Kart to be dynamic. Often, many characters in a Mario Kart map such as Goomba move in a periodical way throughout the course of a race. Consequently, in our algorithm, certain obstacles could move to different vertices in our graph representation of the map. This means that we will have a dynamic graph that changes at each timestep, so classic graph search algorithms that assume the graph stays consistent may not provide an optimal solution in this case. However, there is research related to methods to solving shortest path problems in dynamic graph environments that we can borrow from to address this added complexity.

Another key principle of Mario Kart we can introduce to our problem are the random item boxes present at certain static spots. When a player reaches a space with a random item box, then there is a certain probability this box provides them with a speed up item like the invincible star or mushroom speed ups. Typically, the choice of items a player has is dependent on the position in which they are. For example, if a player is in last place, they will have a much higher probability of getting a speed up item. However, since we are only considering single player races, we will assume that the random item boxes have a consistent probability distribution throughout the course of the race. We can see that optimal strategies to find the fastest path in a course will always try to hit these random item boxes, such that we can maximize our probability to get a speed up item to move faster within the course.

Model and Metrics for Collected Data

To create a Mario Kart course, the way in which terrain types, speed modifiers, and other resources are distributed in the 3D grid is vital. We can consider a game-inspired distribution for the above constraints.

<u>Terrain Distribution:</u>
In the game-inspired distribution, the course contains a mix of terrains based on typical Mario Kart track designs
For a single tile, there is a:

- 50% chance of being a road,
- 25% chance of being grass,
- 10% chance of being a boost,
- 10% chance of being an obstacle,
- 5% chance of being a special tile.

Speed Modifier Distribution:
Each tile type comes with a speed modifier representing the time required to traverse it:

- Road Tiles: Standard speed (base cost = 1 unit of time per tile).
- Grass Tiles: Slower speed (base cost = 2 units of time per tile).
- Boost Tiles: Faster speed (base cost = 0.5 units of time per tile).
- Obstacle Tiles: Impose a penalty if hit.
- Special Tiles: Variable speed modifiers depending on their functionality

Metrics for evaluation:
To evaluate the model's performance, we will focus on the time required to complete a single lap. The model is more successful if it achieves the lowest lap time. It's given by:

Lap Completion Time:
The total time required to traverse a lap from the starting point, calculated as the sum of traversal costs for all tiles in the chosen path. Lower lap times indicate better pathfinding and resource management.

Future Scope

After completing the foundational model for this project, an exciting task if time allows is the implementation of a computer player (AI opponent). This AI would adapt and compete against the user, enhancing the interactivity and complexity of the game. By incorporating machine learning and artificial intelligence techniques, the AI can analyze user strategies and improve its performance, creating a challenging and engaging gameplay experience. The following will be the key ideas for this enhancement:

AI Behavior Modeling:
The computer can use ML to navigate the track where the AI learns the best path by trial and error. The AI will be able to improve its racing strategy over time by associating rewards with lap times and avoiding obstacles.

Opponent Adaptation:
The AI can observe the user's driving patterns like preferred routes, obstacle avoidance etc. and become trained on that data to counter the user effectively.

Dynamic Difficulty Adjustment:
If the user consistently plays better than the AI then it can tweak its speed and strategies to maintain a challenging experience for the user. It can also slow down to make the game beginner friendly.