Car Racing Game - Requirements Document

Valerie Holland

1. Introduction

The Car Racing Game is a simulation where multiple cars race along a predefined track, stopping at various locations and completing the race based on randomized speed factors. The game will determine the winner based on the fastest completion time.

2. Types of Requirements

The system follows the **FURPS+ model**, including:

- **Functionality**: Defines system actions and behavior.
- **Usability**: Determines user experience.
- **Reliability**: Ensures game stability.
- **Performance**: Sets speed and processing benchmarks.
- **Supportability**: Addresses future improvements.
- **Design Constraints & Implementation Needs**: Covers interface and hardware requirements.

---

3. Functional Requirements

3.1 Core Game Mechanics

- The system must allow multiple cars to race simultaneously.
- Each car should be assigned a randomized route consisting of multiple stops.
- Cars should have varying speeds and properties based on their engine and tires.
- The winner should be determined by the shortest total race time.

3.2 Car Behavior

- Cars must move from stop to stop based on their assigned route.
- Cars should calculate and update their total race time dynamically.

3.3 Race Management

- The RaceManager must initialize the race with multiple cars.
- The system must track all cars and determine when the race ends.
- The RaceManager should announce the winner at the end of the race.

3.4 Data Handling

- The system should store and display:
    - Each car's assigned route.
    - Car speed and total time taken.
    - The final ranking of cars.

---

4. Usability and Reliability

4.1 Usability

- The interface must be simple and easy to understand.
- The system should provide real-time updates on car positions and status.

4.2 Reliability

- The game should not crash under normal conditions.
- The race should be replayable without requiring a system restart.

---

5. Performance and Supportability

5.1 Performance

- The game must update car positions in real-time with minimal lag.
- The race simulation should complete in a reasonable time.
- The system should handle multiple cars without performance degradation.

5.2 Supportability

- Future versions should allow for:
    - Different track layouts.
    - New car attributes such as fuel consumption or acceleration.
    - Multiplayer support.

---

6. Design Constraints & Implementation

- The system should be developed in **Java**.
- The program should follow an **object-oriented approach** using the UML class diagram.
- A **graphical user interface (GUI)** should be implemented for visualization.
- The game should be compatible with standard desktop operating systems.

---

1. RaceManager

- Role: Manages the entire race, including cars and the racetrack.
- Relationships:
    - 1 to Many (1..*) with Car → RaceManager controls multiple cars.
    - 1 to 1 (1..1) with RaceTrack → The race occurs on a single track.
    - Associates with Winner (Car) → Determines the fastest car as the winner.
- Key Functionality:
    - Manages cars and their progress.
    - Tracks race conditions.
    - Declares the winner.

---

2. Car

- Role: Represents a racing car in the game.
- Relationships:
  - 1 to 1 (1..1) with Engine → Each car has an engine.
  - 1 to Many (1..*) with Tire → A car has multiple tires.
  - 0 to 1 (0..1) with RaceManager → A car is managed by a race.
  - 0 to 1 (0..1) with RaceTrack → A car moves along a defined track.
  - Uses a Route (List of Stops) → The car follows a specific path.
- Key Functionality:
  - Moves between stops.
  - Calculates total race time based on speed.

---

3. RaceTrack

- Role: Represents the racing environment where the cars compete.
- Relationships:
  - 1 to Many (1..*) with Stops → A racetrack consists of multiple stops.
  - 1 to 1 (1..1) with RaceManager → A single track is used per race.
- Key Functionality:
  - Defines track layout.
  - Generates random routes for cars.

---

4. Stop

- Role: Represents a checkpoint in the race.
- Relationships:
  - 1 to Many (1..*) with RaceTrack → A track has multiple stops.
  - Used in Route (List) by Car → Cars navigate through stops.

- Key Functionality:
    - Defines checkpoints for movement.

---

5. Engine

- Role: Represents the power source of a car.
- Relationships:
    - 1 to 1 (1..1) with Car → Each car has one engine.
- Key Functionality:
    - Determines horsepower and efficiency, affecting car speed.

---

6. Tire

- Role: Represents the tires of a car.
- Relationships:
    - 1 to Many (1..*) with Car → Each car has multiple tires.
- Key Functionality:
    - Defines grip and durability, affecting car performance.

---

7. Wheel

- Role: Represents the car's wheel structure.
- Relationships:
    - 1 to Many (1..*) with Car → Each car has multiple wheels.
- Key Functionality:
    - Specifies size and type of the wheel.

Summary of Responsibilities

Class	Main Role	Key Relationships
RaceManager	Controls the race and determines the winner.	Manages multiple Cars and a single RaceTrack.
Car	Represents a racing vehicle.	Has Engine, Tires, and follows a Route.
RaceTrack	Defines the racing circuit.	Has multiple Stops, managed by RaceManager.
Stop	Checkpoint in the race.	Part of RaceTrack, used in Car routes.
Engine	Powers the car.	A part of Car, determines speed.
Tire	Affects car performance (grip, durability).	Attached to Car (multiple tires per car).
Wheel	Defines wheel properties.	A part of Car.