2D Car Simulation Using Genetic Algorithm

Project Duration: 3 months

# 1. Introduction This simulation project not only explores genetic algorithms but also mimics real-world vehicle design challenges by evolving solutions that optimize performance in varied terrains. By leveraging evolutionary principles, the simulator addresses optimization problems traditionally faced in engineering, providing a robust, automated method for testing and refining vehicle designs.

This report provides a comprehensive account of the Automated Car Evolution Simulation using Genetic Algorithms. The goal of this project was to develop a physics-based simulation that mimics natural selection, allowing cars to evolve over multiple generations to optimize their performance on a dynamic terrain. The simulation is governed by the laws of physics, and cars are evaluated based on their ability to travel the furthest distance over difficult terrain. A genetic algorithm is employed to evolve the car population, with new generations formed by crossover and mutation of the best-performing cars' genomes.

# 2. Objectives

The primary objectives of this project were:  
- Develop a 2D physics-based car simulation  
- Implement a dynamic terrain system  
- Use a genetic algorithm to evolve car populations over time  
- Optimize the simulation for speed and performance  
- Visualize the evolution and performance of cars across multiple generations

# 3. Project Timeline

## Month 1: Initialization and Physics Setup

The first month focused on the initialization of the simulation environment and setting up the physics engine:  
- Selected Planck.js as the physics engine for realistic simulation  
- Defined core simulation constants, including physics scale, terrain generation parameters, and car movement forces  
- Built the framework for rendering the terrain and cars using the createCanvas() function  
- Set up the main simulation loop with physics updates and rendering

## Month 2: Genetic Algorithm and Car Evolution

During the second month, the genetic algorithm was implemented to enable the evolution of car populations:  
- Developed random genome generation for cars, consisting of chassis vertices, wheel sizes, and wheel positions  
- Implemented the physics for creating cars and their wheels using Planck.js bodies  
- Wrote functions to apply forces to the cars and update wheel rotation  
- Introduced the crossover and mutation functions to evolve cars across generations  
- Added the terrain generation algorithm using Perlin noise to create challenging landscapes

## Month 3: Optimization and Visualization

In the final month, the focus was on optimizing the simulation and enhancing the visualization for better performance and user experience:  
- Optimized the camera system to follow the leading car in real-time  
- Improved the terrain generation algorithm to extend as the cars progress  
- Implemented user input fields to adjust parameters like the number of cars, mutation rate, and wheel sizes  
- Created visual feedback for displaying the leading car's performance, including chassis and wheel parameters  
- Added controls to start, pause, and resume the simulation

# 4. Simulation Workflow

## 4.1 Physics Engine Setup Planck.js is utilized for simulating the physical behavior of cars, including gravity, collisions, and joint dynamics. Each car’s chassis and wheels are defined as rigid bodies with properties like density, friction, and damping. The engine manages collisions between the car and terrain to ensure realistic movement. The vehicle's motion is governed by forces applied to the wheels, mimicking real-world mechanics.

The simulation leverages the Planck.js library, a JavaScript engine that simulates 2D rigid body physics. We created the physics world with gravity settings and scaled units for rendering. The world step rate was set to 60Hz for realistic movement.

## 4.2 Terrain Generation Perlin noise is used to procedurally generate the terrain, creating a varied and challenging landscape. The terrain adapts dynamically as cars progress, introducing hills, valleys, and obstacles that test the vehicle's capabilities. This method ensures that no two simulations are alike, making it an effective way to evaluate the genetic algorithm’s robustness.

The terrain is generated dynamically using a Perlin noise algorithm, which provides a natural, rolling landscape. As the simulation progresses, additional terrain is generated on-the-fly to ensure the cars are always driving on new terrain.

## 4.3 Car Generation and Physics

Each car consists of a chassis (body) and two wheels. The chassis is defined by a random set of vertices forming a polygon, and the wheels are defined as circular Planck.js bodies. Cars are subject to forces that propel them forward, and wheel rotation is calculated based on the chassis velocity.

## 4.4 Genetic Algorithm The genetic algorithm (GA) employed in this simulator operates through several stages: - Selection : Tournament selection is used, where the best vehicles from small groups are chosen as parents for the next generation. - Crossover : The genomes of two parent vehicles are combined, swapping segments to produce diverse offspring. This ensures a mix of features like wheel size and chassis shape. - Mutation : Random changes are introduced to offspring genomes to maintain diversity. This mutation affects attributes such as wheel position and chassis vertices, promoting exploration of new designs. - Fitness Evaluation : Fitness is determined by the distance traveled and the car's stability on different terrains. This multi-objective approach ensures that the fittest cars are selected not only for speed but also for balance and endurance.

The genetic algorithm is at the core of the simulation. Each car has a genome consisting of its chassis shape, wheel sizes, and wheel positions. The simulation runs for a set number of generations, with each generation evolving based on the fitness of the cars. The fitness is determined by the distance traveled before all cars stop moving. The best cars are selected for reproduction, and new cars are generated through crossover and mutation.

# 5. Conclusion 6. Future Scope This project opens avenues for further development, including: - Adaptive Genetic Algorithms : Incorporating adaptive mutation rates and crossover techniques to enhance diversity and convergence speed. - Neural Network Integration : Integrating simple neural networks to allow cars to learn driving behavior and adapt to terrain features. - Multiplayer Competitions : Developing a feature where multiple users can compete with their evolved vehicles, creating a community-driven optimization challenge. - Enhanced Terrain Features : Implementing diverse terrain types such as water bodies, mud, and slopes that require specialized vehicle designs.

The Automated Car Evolution Simulation successfully demonstrates how a genetic algorithm can evolve car designs to optimize their ability to traverse a dynamically generated terrain. Through multiple generations, cars adapt to overcome obstacles and maximize their distance traveled. The project offers valuable insights into the power of genetic algorithms in optimization tasks and opens possibilities for further improvements in simulation realism and efficiency.