Self-driving Car Simulation using Genetic Algorithm

J-Component Document

Mihir S P (19BCE2680)

Submitted to

Prof. Gladys Gnana Kiruba B, SCOPE

School of Computer Science and Engineering

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Table of Contents

Topic

Abstract:

According to the WHO, around 1.35 million people die every year in road traffic crashes. Year after year the advancements in technology have paved a way for the inclusion of **artificial intelligence** even into automobiles. In this project we will simulate an environment using **Unity** where we will train a car to go around a track on its own by using a simple **Neural Network** and training it with a **genetic algorithm**.

Keywords: Artificial Intelligence, AI, Neural Network, Genetic algorithm, Unity.

1. Introduction:

A self-driving car or driverless car is a vehicle that is capable of sensing its environment and moving safely without the need for any human input. The future of this technology will have a huge impact on multiple industries and other circumstances. Self-driving cars combine various sensors to perceive their surroundings, such as sonar, radar, lidar, GPS, inertial measurement units and odometry. Advanced control systems interpret sensory information to identify appropriate navigation paths, as well as obstacles and relevant signage.

In this project, we will simulate an environment using **Unity** where we will train a car to go around a track on its own by using a simple **Neural Network** and training it with a **genetic algorithm**.

1.1. What is a Neural Network?

Artificial neural networks (ANNs), also called neural networks (NNs), are computing systems inspired by the biological neural networks in animal brains.

Neural networks are a set of algorithms, modelled loosely after the neural networks in the human brain, that are designed to recognize patterns. They interpret sensory data through labelling, clustering raw input or a kind of machine perception. All real-world data, be it images, sound, text or time series, must be translated into a numerical form as the patterns they recognize by a Neural Network are numerical, contained in vectors.

1.2. Genetic algorithm

A genetic algorithm is inspired by Charles Darwin's theory of natural evolution. The algorithm is a search heuristic that reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation.

The process of natural selection starts with the selection of the fittest individuals from the starting population. These selected individuals will produce offspring which inherit the characteristics of the parents and will be added to the next generation. If parents have good fitness, their offspring will have a better chance of having fitness better than parents and have

a better chance at surviving. This process keeps on iterating until a generation with the fittest individuals will be found. This notion can be applied to a search problem. We consider a set of solutions for a problem and select the set of best ones out of them.

Phases considered in a genetic algorithm are:

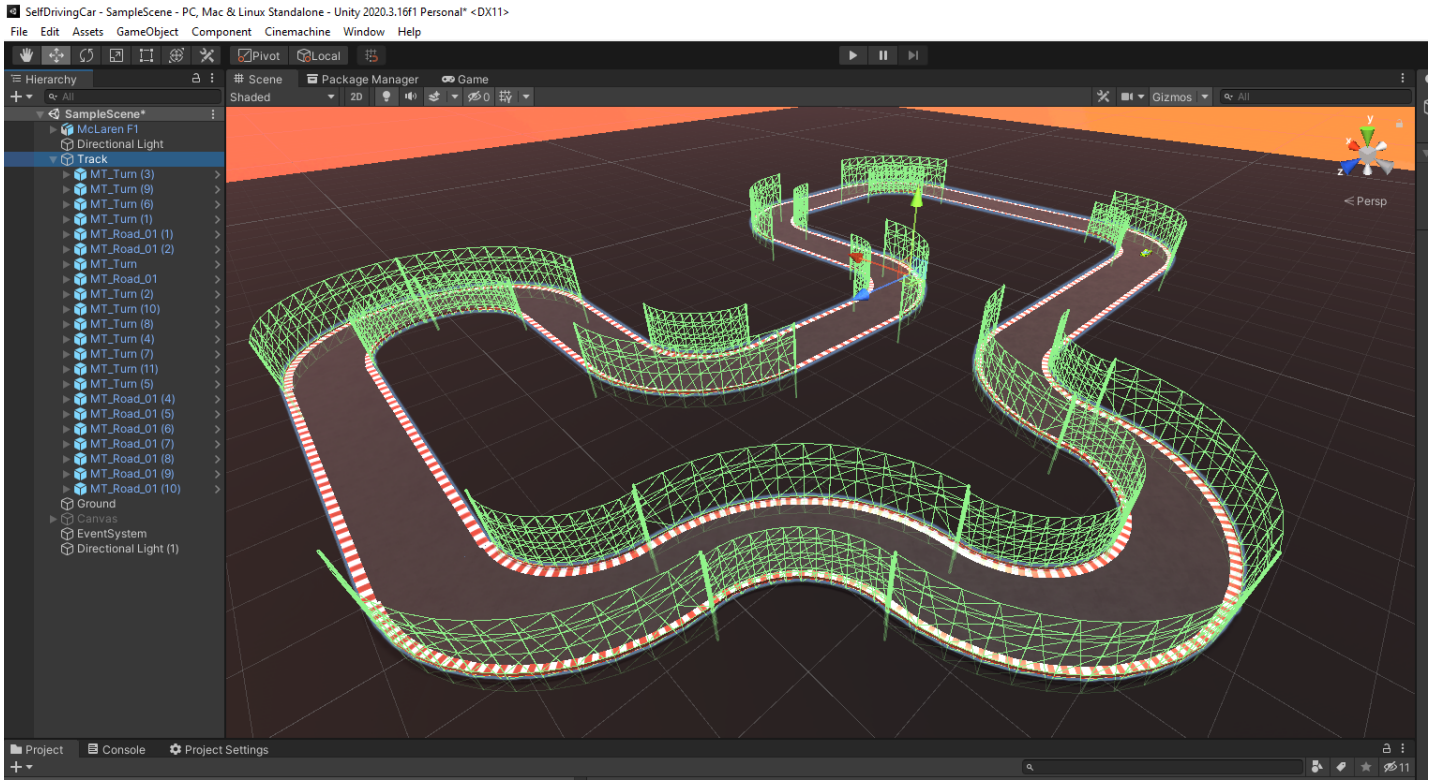
1. Initial population
2. Fitness function
3. Selection
4. Crossover
5. Mutation

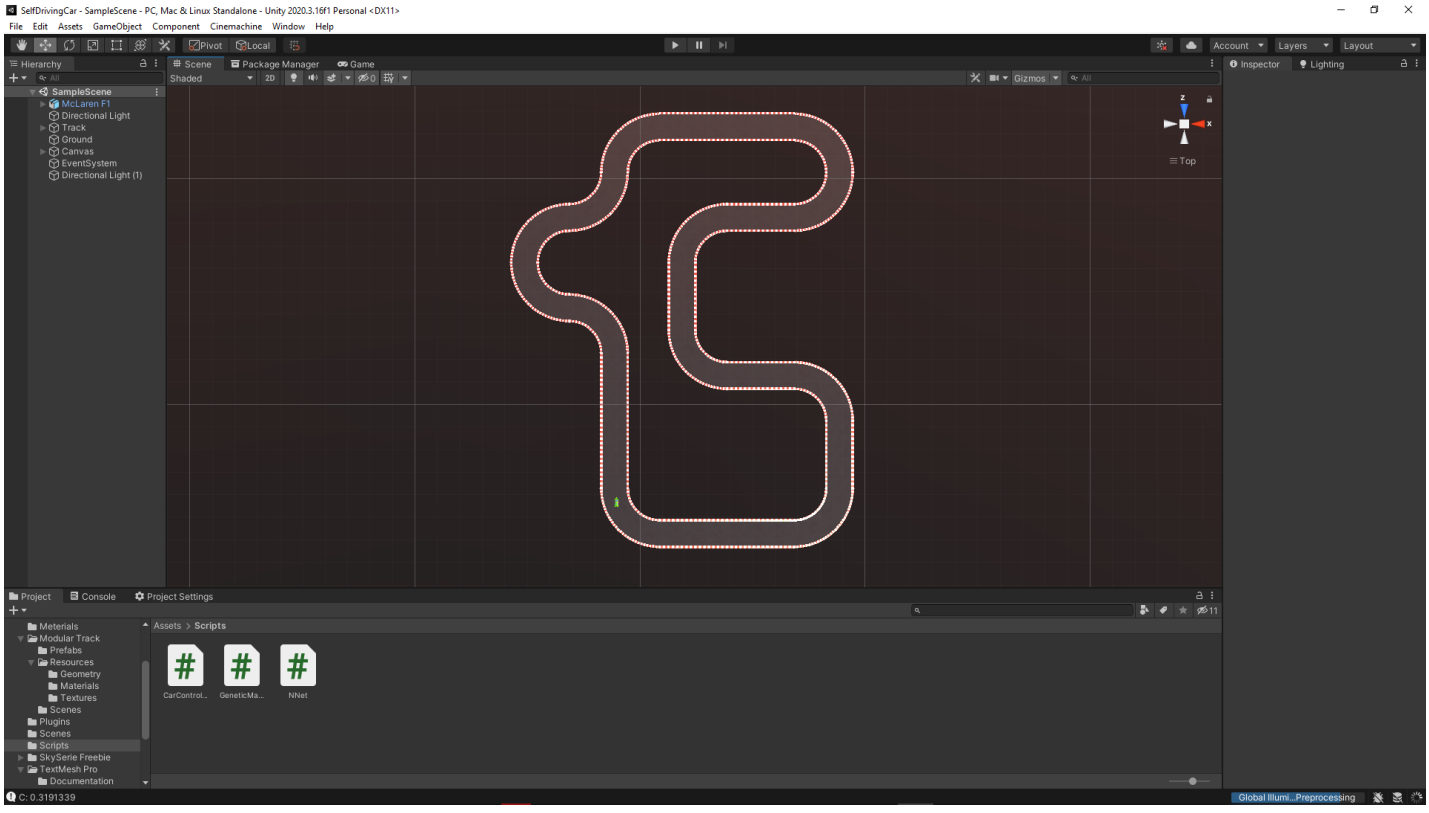
Using the above information a Genetic algorithm can be used to train a neural network with 3 input nodes (3 of the distance sensors) and 2 output nodes (acceleration and steering) to control a car around a track.

3. Proposed work

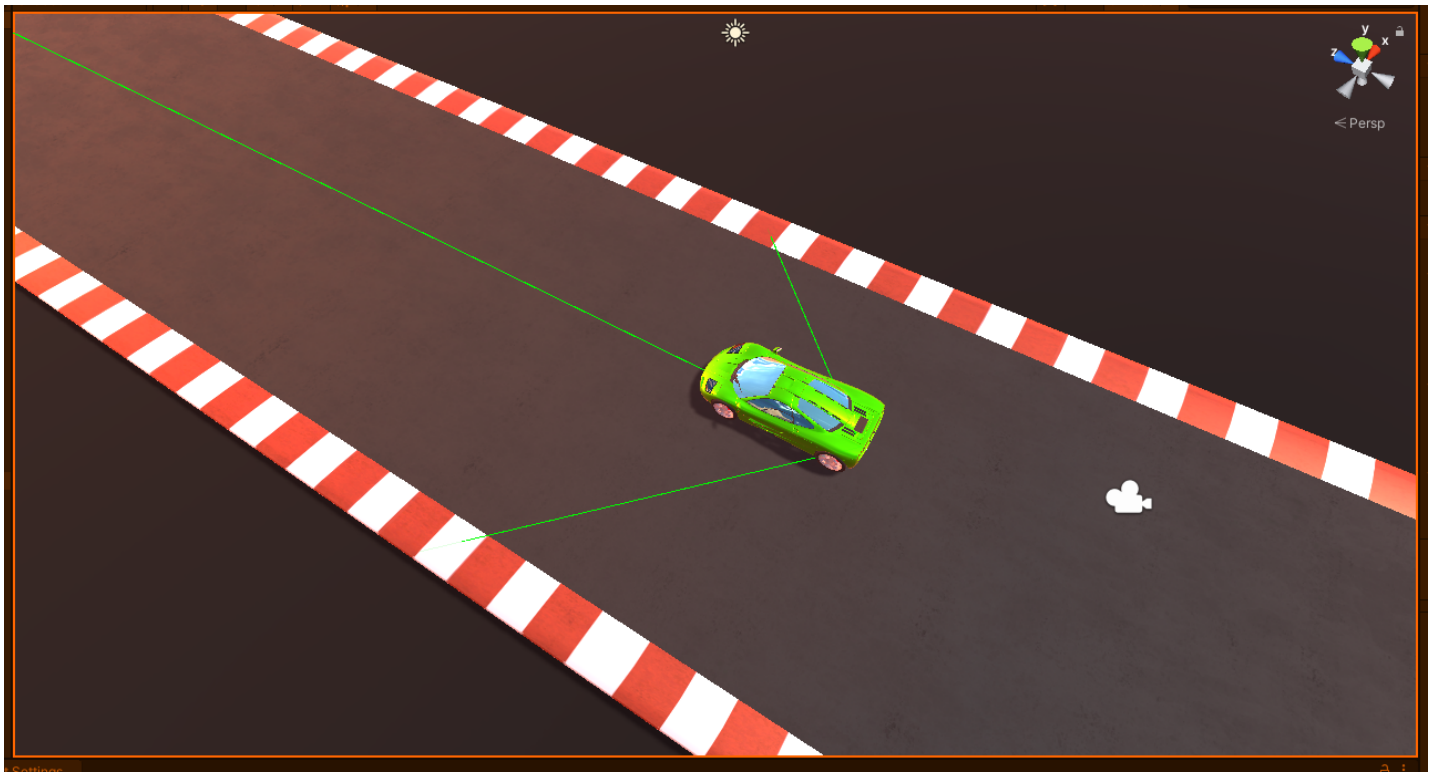
3.1. Architecture

First, the Simulation environment is created in Unity with 3D assets.






SelfDrivingCar - SampleScene - PC, Mac & Linux Standalone - Unity 2020.3.16f1 Personal <DX11>
File Edit Assets GameObject Component Cinemachine Window Help
Hierarchy
SampleScene
McLaren F1
Directional Light
Track
Ground
Canvas
EventSystem
Directional Light (1)
Scene
Package Manager
Game
Inspector
Lighting
McLaren F1
Transform
Rigidbody
Box Collider
Car Controller (Script)
Fitness
Network Options
Button Ref
N Net (Script)
Genetic Manager (Scrip
References
Controls
Project
Console
Project Settings
Assets > Scripts
Materials
Modular Track
Prefabs
Resources
Geometry
Materials
Textures
Scenes
Plugins
Scenes
Scripts
SkySerie Freebie
TextMesh Pro
Documentation


Persp

The code for controlling the Car is written in **C#.**

There are three main scripts When it comes to implementing the “Self-driving” nature of the project:

- CarController.cs
- GeneticManager.cs
- NNet.cs

CarController.cs:

- This is the main class which is responsible for controlling the car.
- It is responsible for getting the input from the sensors and using the inputs to run the Neural Network from The NNet class
- It is responsible for getting the output from the Neural Network and controlling the car accordingly.
- It is also responsible for calculating the fitness values for the car.

GeneticManager.cs:

- This is the class where the Genetic algorithm is used.
- It is responsible for all the functions of a genetic algorithm namely:
    - Initial population
    - Fitness function
    - Selection
    - Crossover
    - Mutation

NNet.cs:

- This is the class of the Neural network.
- It is responsible for handling the Initialization of the Neural network, assigning weights and biases to the nodes of the network, controlling the number of nodes and hidden lavers in the Neural network and also running the Neural network

3.2. Hardware and Software requirements:

This project is done using **Unity** so the hardware and software requirements to run the unity editor are:

Minimum requirements	Windows	macOS	Linux (Support in Preview)
Operating system version	Windows 7 (SP1+), Windows 10 and Windows 11, 64-bit versions only.	High Sierra 10.13+	Ubuntu 20.04, Ubuntu 18.04, and CentOS 7
CPU	X64 architecture with SSE2 instruction set support	X64 architecture with SSE2 instruction set support	X64 architecture with SSE2 instruction set support
Graphics API	DX10, DX11, and DX12-capable GPUs	Metal-capable Intel and AMD GPUs	OpenGL 3.2+ or Vulkan-capable, Nvidia and AMD GPUs.
Additional requirements	Hardware vendor officially supported drivers	Apple officially supported drivers	Gnome desktop environment running on top of X11 windowing system, Nvidia official proprietary graphics driver or AMD Mesa graphics driver. Other configuration and user environment as provided stock with the supported distribution (Kernel, Compositor, etc.)

3.3. Methodology adapted:

- Initially, a population of 60 cars is generated at random ie. the weights and biases if the genome is generated at random.
- A fitness function is to calculate the overall fitness of the genome. The function takes into to account the following:
    - Distance travelled by car.
    - The average speed of the car.
    - The average reading of the three sensors ie. the average distance the car maintained from obstacles and the sides of the track.
- The weightage given to the three factors can be tweaked to give different results.
    - Example: If more weightage is given to the average reading of the three sensors, then the resulting car will be very conscious and only drive in the centre of the road. If more weightage is given to Average speed then the resulting car will be faster.
- The genome is killed when it collides with an obstacle or goes off-road.
- 6 genomes with the most fitness and 2 genomes with the worst performance were added to the genome pool of the next generation.
- There are 26 Crossovers performed to generate the rest of the 52 genomes.
- The probability of Mutation is set to 0.1.
- This process keeps on iterating until a generation with the fittest genome will be found.

4. Tools used:

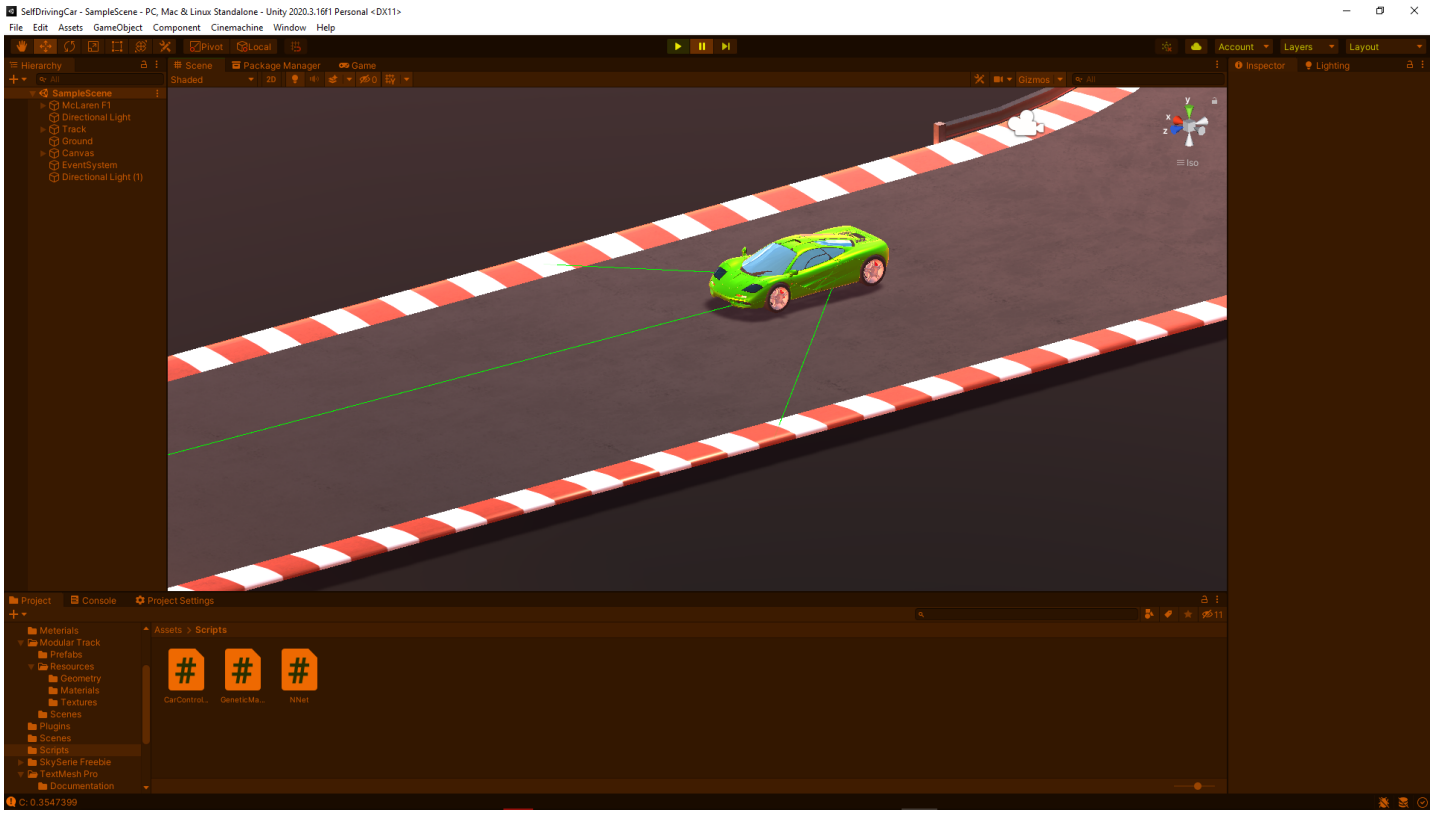
- Unity is the game engine for simulating the project in a 3D environment.
- Visual studio code as the text editor.
- Blunder for creating 3D models.
- 3D model and assets used:
    - Modular Track in the unity Asset store.
    - Mclaren F1 1994 3D Model form free3d.com

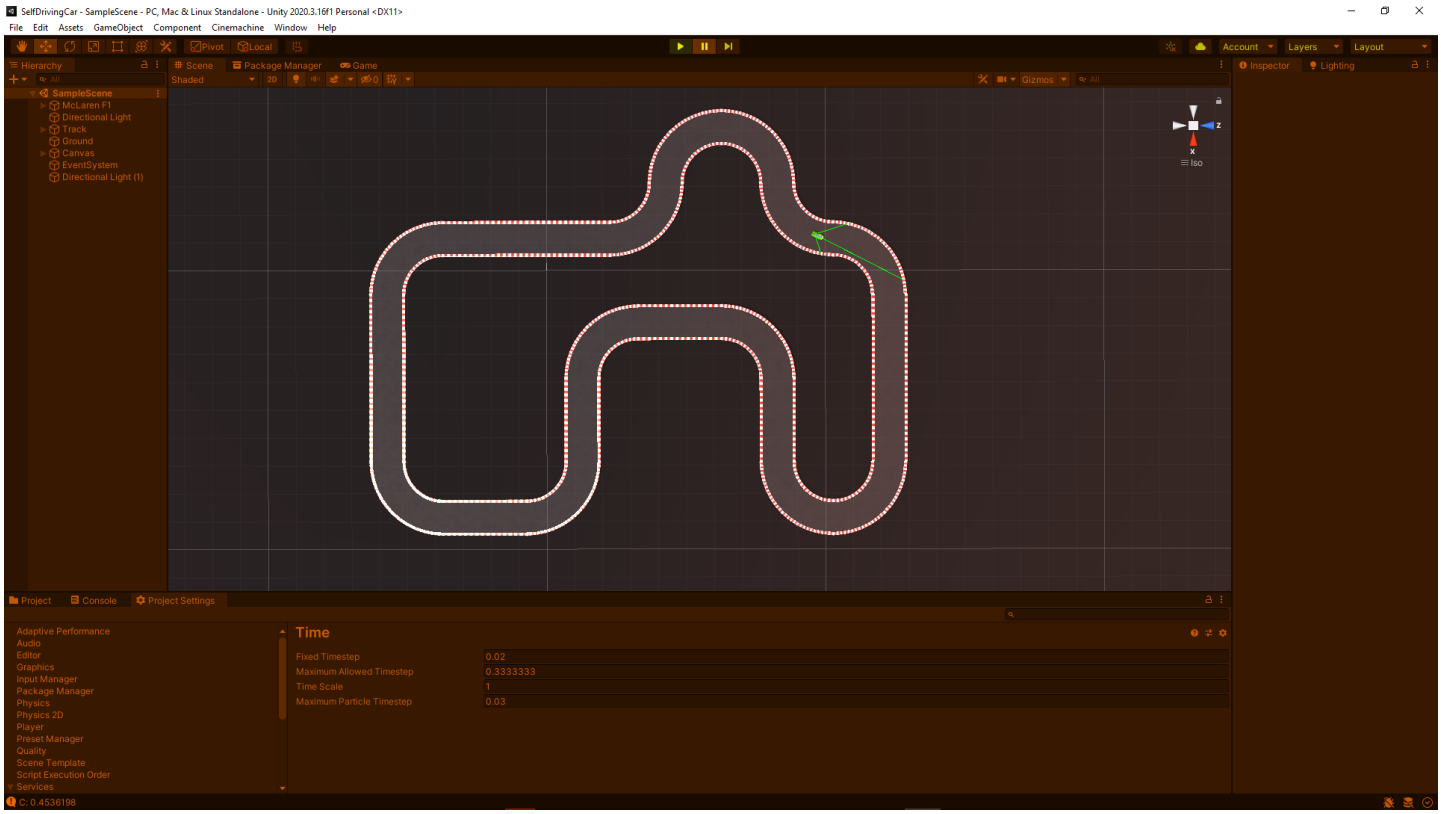
5. Implementation and Results

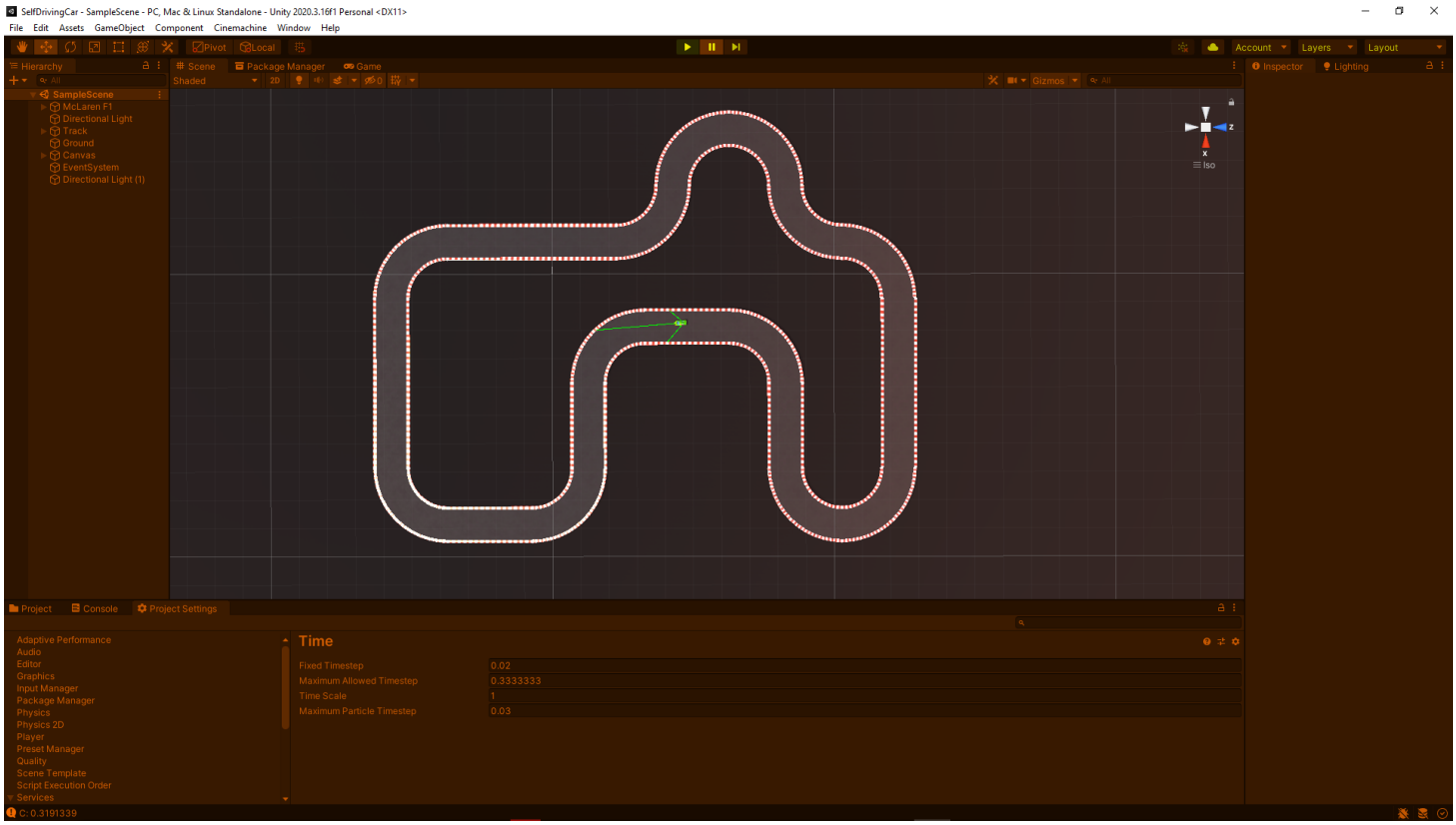
5.1. The Compleat project GitHub Link:

- **https://github.com/mihirsp18/Self-Driving-Car**
- The scripts can be found in **Assets/Scripts**

5.2. Result


SelfDrivingCar - SampleScene - PC, Mac & Linux Standalone - Unity 2020.3.16f1 Personal <DX11>
File Edit Assets GameObject Component Cinemachine Window Help
Hierarchy
SampleScene
McLaren F1
Directional Light
Track
Ground
Canvas
EventSystem
Directional Light (1)
Scene
Package Manager
Game
Inspector
Lighting
Project
Console
Project Settings
Assets > Scripts
CarControl...
GeneticMa...
NNet
C: 0.3547399


SelfDrivingCar - SampleScene - PC, Mac & Linux Standalone - Unity 2020.3.16f1 Personal <DX11>
File Edit Assets GameObject Component Cinemachine Window Help
Hierarchy
SampleScene
McLaren F1
Directional Light
Track
Ground
Canvas
EventSystem
Directional Light (1)
Scene
Package Manager
Game
Inspector
Lighting
Project
Console
Project Settings
Time
Fixed Timestep 0.02
Maximum Allowed Timestep 0.3333333
Time Scale 1
Maximum Particle Timestep 0.03
C: 0.4536198


SelfDrivingCar - SampleScene - PC, Mac & Linux Standalone - Unity 2020.3.16f1 Personal <DX11>
File Edit Assets GameObject Component Cinemachine Window Help
Hierarchy
SampleScene
McLaren F1
Directional Light
Track
Ground
Canvas
EventSystem
Directional Light (1)
Scene
Package Manager
Game
Inspector
Lighting
Project
Console
Project Settings
Time
Fixed Timestep 0.02
Maximum Allowed Timestep 0.3333333
Time Scale 1
Maximum Particle Timestep 0.03


INITIAL POPULATION: 60
CURRENT GENERATION: 0
CURRENT GENOME: 38
AVRAGE SPEED: 60
FITNESS BAR:


Initial Population: 60
Current Generation: 5
Current Genome: 0
Avrage Speed: 72
Fitness Bar:


Initial Population: 60
Current Generation: 5
Current Genome: 0
Avrage Speed: 72
Terminate
Successfully Trained!

6. Conclusion

In this project, we successfully simulate an environment using **Unity** where we will train a car to go around a track on its own by using a simple **Neural Network** and training it with a **genetic algorithm**. The Code is written in a very modular way which allows it to be used in various use cases in game development.

The project can still be improved. Currently, there is no meaningful obstacle in the track (like a pedestrian). For handling such more complex cases we will need a more robust algorithm, in this case, we can use the **NEAT** genetic algorithm. NEAT stands for NeuroEvolution of Augmenting Topologies and it should provide better results in more complex environments.

7. References

1. https://www.youtube.com/watch?v=C6SZUU8XQQ0&list=PL9FeLoYIHiTwyS18t1RUJQCKFy_abl3qe
2. P. Charoenkwan, S. Fang and S. Wong, "A Study on Genetic Algorithm and Neural Network for Implementing Mini-Games," 2010 International Conference on Technologies and Applications of Artificial Intelligence, 2010
3. Chuen-Tsai Sun and Ming-Da Wu, "Self-adaptive genetic algorithm learning in game playing," Proceedings of 1995 IEEE International Conference on Evolutionary Computation, 1995.
4. S. J. Louis and C. Miles, "Playing to learn: case-injected genetic algorithms for learning to play computer games," in IEEE Transactions on Evolutionary Computation, vol. 9, no. 6, pp. 669-681, Dec. 2005.
5. H. K. Lam, S. H. Ling, F. H. F. Leung and P. K. S. Tam, "Tuning of the structure and parameters of neural network using an improved genetic algorithm," IECON'01. 27th Annual Conference of the IEEE Industrial Electronics Society (Cat. No.37243), 2001
6. Trujillo, Leonardo, et al. "neat genetic programming: Controlling bloat naturally." Information Sciences 333 (2016).
7. Carson J. Genetic Algorithms: Advances in Research and Applications. Nova Science Publishers, Inc; 2017. Accessed December 5, 2021.
8. https://www.sicara.ai/blog/2017-08-29-was-darwin-great-computer-scientist
9. https://towardsdatascience.com/a-gentle-introduction-to-neural-networks-series-part-1-2b90b87795bc
10. https://towardsdatascience.com/neat-an-awesome-approach-to-neuroevolution-3eca5cc7930f
11. https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3
12. https://docs.unity3d.com/Manual/index.html
13. https://www.youtube.com/channel/UCYbK_tjZ2OrIZFBvU6CCMiA