Jagunmolu B Oke

[Email address]

Abstract

[Draw your reader in with an engaging abstract. It is typically a short summary of the document.   
When you’re ready to add your content, just click here and start typing.]

Racing simulation (driver behaviour)

Artificial Intelligence for Games

Table of Contents

[Project Overview 2](#_Toc155334660)

[AI-Driven Behaviour Characteristics 2](#_Toc155334661)

[Car Model 2](#_Toc155334662)

[Decision Making 3](#_Toc155334663)

[Finite State Machine 3](#_Toc155334664)

[Awareness (perception) 4](#_Toc155334665)

[Fuzzy Logic and Set 4](#_Toc155334666)

[Obstacle Avoidance 6](#_Toc155334667)

[Pathfinding 9](#_Toc155334668)

[Appendix 10](#_Toc155334669)

# Project Overview

This project is about racing cars, where AI embodies these car model and have their own unique behaviours for analysis and understanding the play space and make swift decision in a fun and playful manner. These behaviours include and not limited to Acceleration and Breaking, Environmental Awareness, Turning behaviour, Offensive behaviour, Defensive behaviour, avoidance behaviour and reactive behaviours.

This project aims to create an AI-driven behaviour model for a racing simulation using Unity neglecting fully car dynamics. These would be achieved using Applied mathematics (majorly Vector calculation, mathematical functions) and AI techniques (revolving around decision making then pathfinding and Autonomous Agents) to model these systems.

## AI-Driven Behaviour Characteristics

Primarily these would include two unique AI-driven behaviours:

1. Aggressive Racer: This is a behaviour considered reckless behaviour where the driver takes more risks. Below is a list of characteristics:
   1. Constantly seeks to overtake.
   2. Aggressive in manoeuvres.
   3. Constant pressure on an opponent (forcing them to error and close proximity to the opponent).
   4. If behind, constantly aiming to catch up by aggressive overtaking and risky moves.
2. Cautious Racer: This is a behaviour considered defensive behaviour where the driver takes less risks. Below is a list of characteristics:
   1. Avoids risky overtaking.
   2. Avoids collision.
   3. Priorities a safe proximity to opponents.
   4. Sacrificing speed (Brakes early to Corners).
   5. Adjust speed based on potential collision and unpredictable opponent behaviour.

## Car Model

A diagram of a car system

Description automatically generatedThe car used in this project is a basic car model based on unity physics wheel collider neglecting other vehicle physics dynamics.

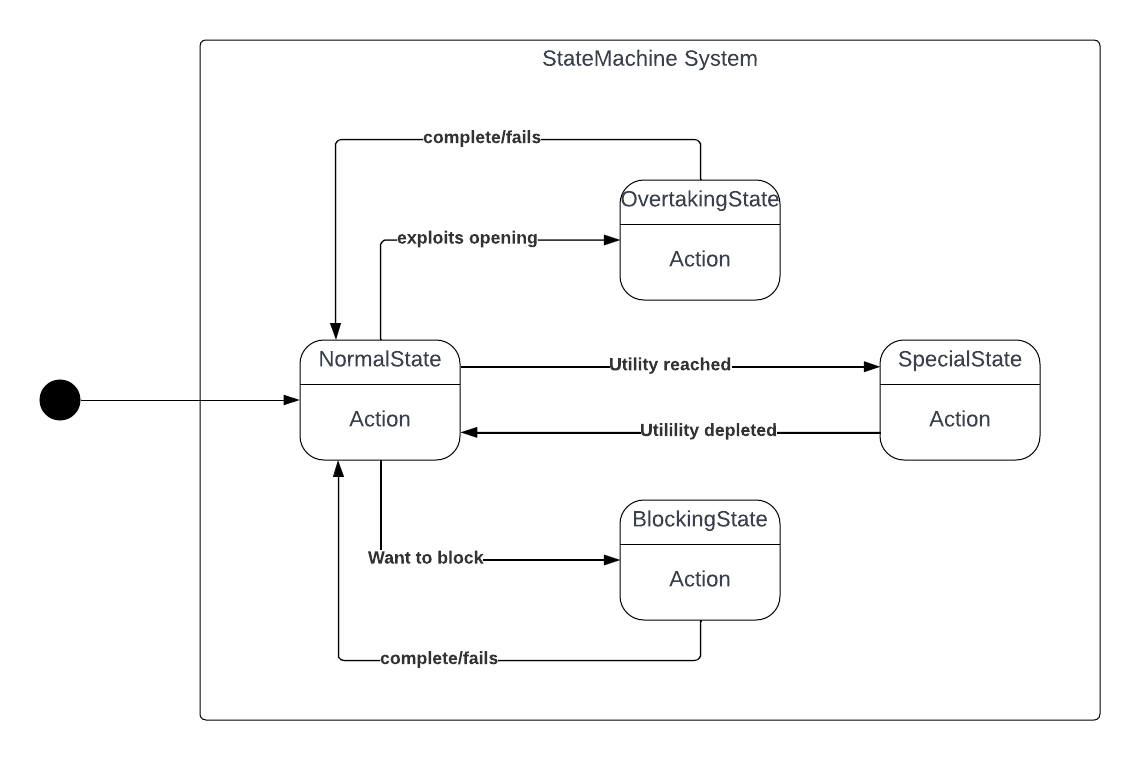
# Decision Making

Decision-making is primarily the backbone of creating a believable AI model behaviour, where the model can act and react based on its internal or external knowledge which the process in turn could affect the model's internal state and external environment state.

## Finite State Machine

The Finite State Machine (FSM) is the abstraction of multiple components of a system to a finite number of states, where only one can exist at a time and transitions are made between those states based on predetermined conditions.

FSM is used to model and manage the behaviours of the drivers; the following is an illustration of the use of FSM for a basic driver model.

 The fig above shown the UML diagram of the state interaction for a basic driver model, when conditions are check using current gameplay situations with Boolean logic to decide a condition is meant to transition between state. But Special State use utility values to decide switch for example the Aggressive Racer special state is Aggressive state ranging from 0 – 10. Which increases or decreases over time based on the game state. If a threshold is meet it switch to the state or out of the state. Below is the table illustration of the model.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Opening (OnStart) | Special Utility < Threshold | Special Utility > Threshold | Potential Block (OnStart) | OnComplete |
| Normal | Overtake | - | Special | Block | - |
| Overtake |  | - | Special |  | Normal |
| Block |  | - | Special | - | Normal |
| Special | Overtake (Possible Ram) | Normal | - | Block (Ram) | - |

Using FSM to model behaviours in games could get a bit complicated if the system being handled gets larger and more complex. Switching from one state to another, having multiple conditions to switch to another state. For a racing AI system, behaviours that are not large FSM will suffice to represent those behaviours (Tomlinson & Melder, 2014).

Overtaking state, a trail for ghost (potential predicted position), Dot product is used to calucte is car is ahead or behind positions.THIS HAS CHANGED, to steering to the most opponent dominant side and create a path with minimum of three nodes.

TALK ABOUT PATH IN PATHFINDING: the calculation what is consider to generate a path vector maths.

HERE: use the overtake opptunitity to talk about the advantage of better FSM over hardcoded enum you could have enter and exit state which could be used for enhancing the system. In the enter state (IT S NOT CALLED STATE BY THE WAY) will when you enter variable could be initiated check if required vslues are set for proper functioning of state like SpeedAdjustment is defined first in the enter state if each state have uniques difference. For exit, could be used for state to complete what is going on and get ready for next state, for example in this current when exiting to overtaking state the overtaking the analysis done for overtaking are stored in as struct data (SHOW IMAGE) that would be sent to a overtake state on success if not fails and return back to current state.

## Awareness (perception)

Awareness and perception are crucial factors for video game characters when making decisions. These factors help define what the character knows or doesn't know so that they can make informed choices. The AI collects information about the play space using specific variable calculations, as well as analysis through raycasting and vector calculations.

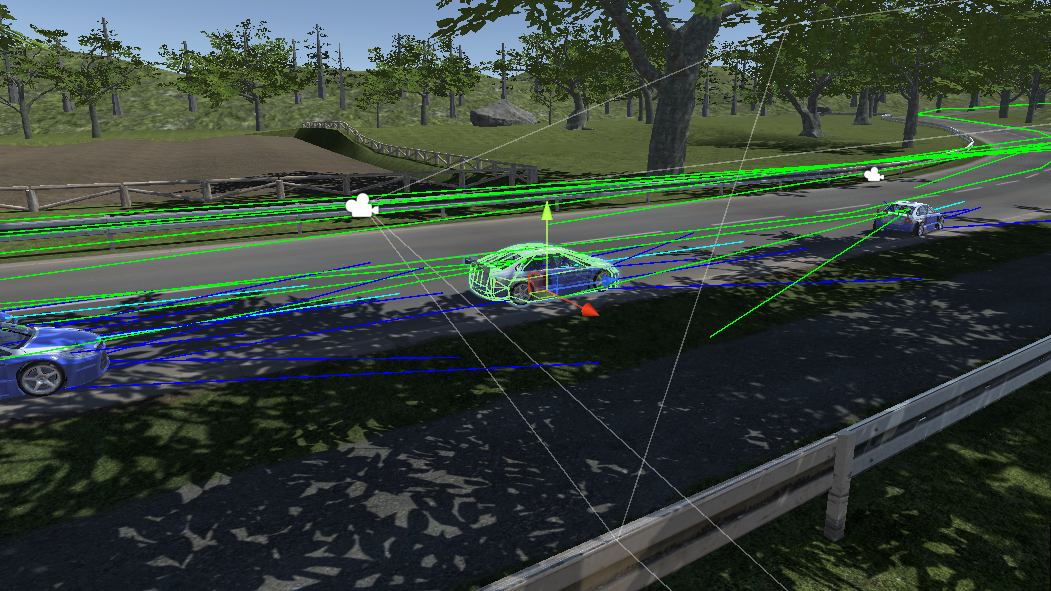
Raycasting is used to predict potential collisions and opponents ahead of the AI. Different types of raycasting are utilized to retrieve information about the play space, such as distance to collision, opponent information, and distance to next corners. Various types of collision raycasts are used for analysis, including single rays, single rays with parallel whiskeys, and single rays with cone angle whiskeys. Please see Appendix G, H and I for more details.

Figure : Multiple Raycasting

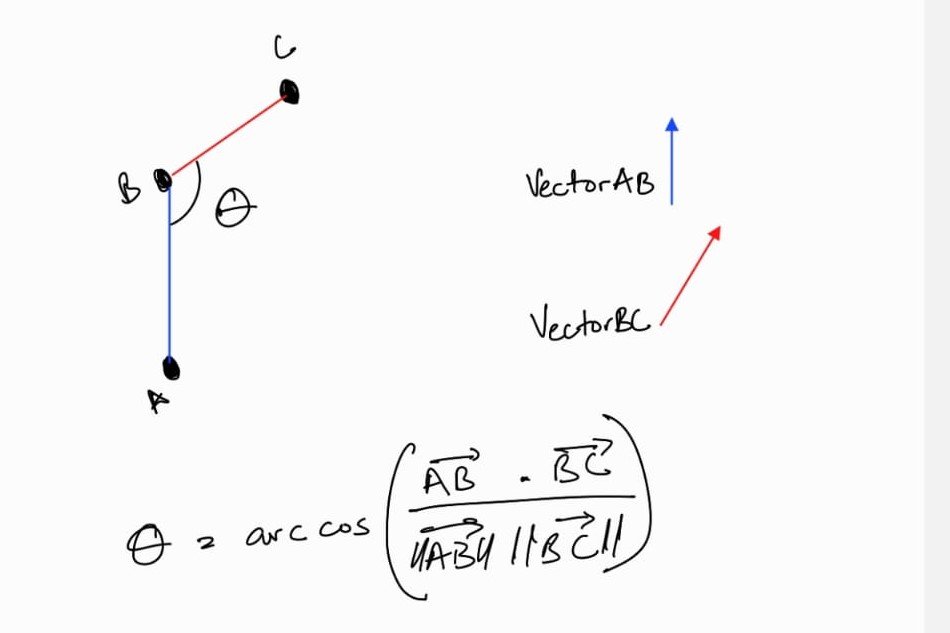
Vector calculations are used to check distance, direction to points, angles between points, and more. One of the main uses of vector calculations is to calculate the distance to the next corner node, which is possible because the pathfinding system is node-based.

Figure : Three points angle calculation

This is achieved by testing the angle between three points (A, B, and C) to determine if there is a corner. The vectors A to B and B to C are calculated, and a dot product is performed to get the angle between them. The steepness between the two vectors defines the angle of the corner. Please see Appendix J for more details.

Talk about dynamic change in the side whisky angles, found the document before but forgot where.

Where all this helps the AI better analysis and understand the play space.

## Fuzzy Logic and Set

Fuzzy Logic is a mathematical tool that enables working with grey areas, not just black and white. It allows for modelling concepts such as “how true”, “pretty big” and more instead of just “true or false” as in traditional Boolean logic. AI characters can have deeper reasoning ability. (McCuskey, 2000).

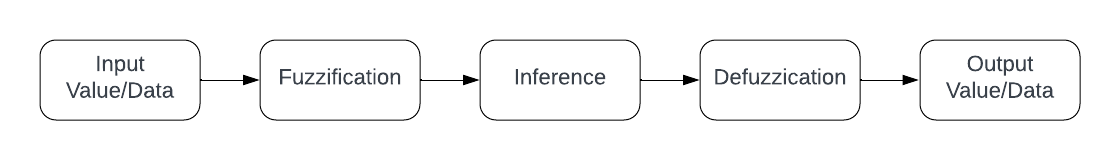
 Fuzzy logic has been used to model the speed adjustment for each driver based on the distance to obstacles or corners and the current speed of travel. This approach aims to create unique driver understanding to distance and current speed in their own unique ways, and a decision will be made accordingly.

Figure : Driver Fuzzy System Process

The Figure above provides shows the fuzzy logic process used to make speed decision for racing agents. The process steps through the following steps:

1. The **INPUT VALUE/DATA**: hard values thar provide specific information. The process takes in speed, distance, and the allowance for both speed, and distance. Please see Appendix D for more details.
2. **Fuzzification:** where the system processes and determines the degree to which the input data belongs to a fuzzy set.

For the Driver Fuzzy System, the set is called DecisionRating contains members of High, Average and Low. To rate a data (current speed and distance), the average is aggregated:

To save time with complex mathematical calculations, the Animation Curve please see Appendix C, a unity tool was used to define the degree of member (DOM) to Decision Rating Set. This helps to create a unique behaviour across drivers by easily adjusting the animation curve corresponding to the members of DecisionRating.

To rate the DOM is each member, by the use of the method Evaluate() to the corresponding animation curve.

Degree of x = animation curve Of. Evaluate(ratio).

Where x is the member of DecisionRating, The degree of high, medium and low defines the belongs of the input data to the members. GOING TO HAVE AN OF THE SET AND ILLUSTRATION

1. **Interference:** where the system decides what the input data means, using 1 or 0 after fuzzification based on some provided rule set.

This model is to interpret “Big”, “Medium”, or “Small” and uses the fuzzified Data to decision to interpret the input data (current speed and distance) as Big, Medium, or Small. Please see Appendix A, for the Fuzzy Rule Set struct.

Big if the input data's highest DOM is high.

Medium if its DOM is average.

And Small if its DOM is low.

1. **Defuzzification:** where the system processes the output from the fuzzified set and Interference to a single data, computer-friendly information based on some extra rule set.

This model outputs the data as a single enum variable see Appendix E, SpeedAdjustment. The table below shows the relationship based on Speed and Distance to determine the SpeedAdjustment state.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Small Speed | Medium Speed | Big Speed |
| Big Distance | Floor it | Speed Up | Maintain Speed |
| Medium Distance | Speed Up | Maintain Speed | Slow Down |
| Small Distance | Speed Up | Slow Down | Brake Hard |

1. **Output:** The data is then ready to use by driver. For this current program the SpeedAdjustment is used as a mini–HardCoded State machine (Hierarchical State Machine) as it is layered within the FSM system. The Basic function of each state is listed below:

|  |  |
| --- | --- |
| Brake Hard | Full brake (1.0f), no acceleration |
| Slow Down | 0.5f brake, no acceleration |
| Maintain Speed | No brake, no acceleration |
| Speed Up | No brake, 0.5f acceleration |
| Floor it | No brake, full acceleration (1.0f) |

# Obstacle Avoidance

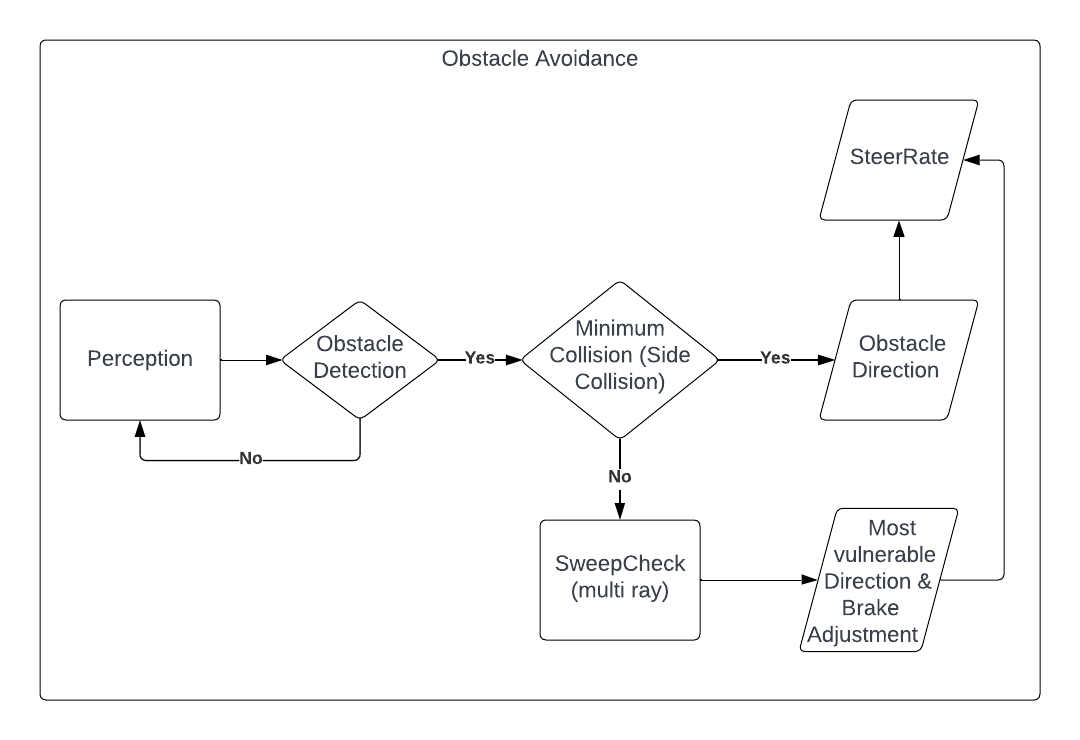
 Predicting collision or threat is a very useful thing to be implemented into an AI System, that involves more than one AI agent (DaGraca, 2017). The Obstacle Avoidance System is heavily reliant on the Awareness set up for the system.

Figure :Obstacle Avoidance System Flow.

The Figure shows the step for individual AI driver models process avoiding threat for decision making.

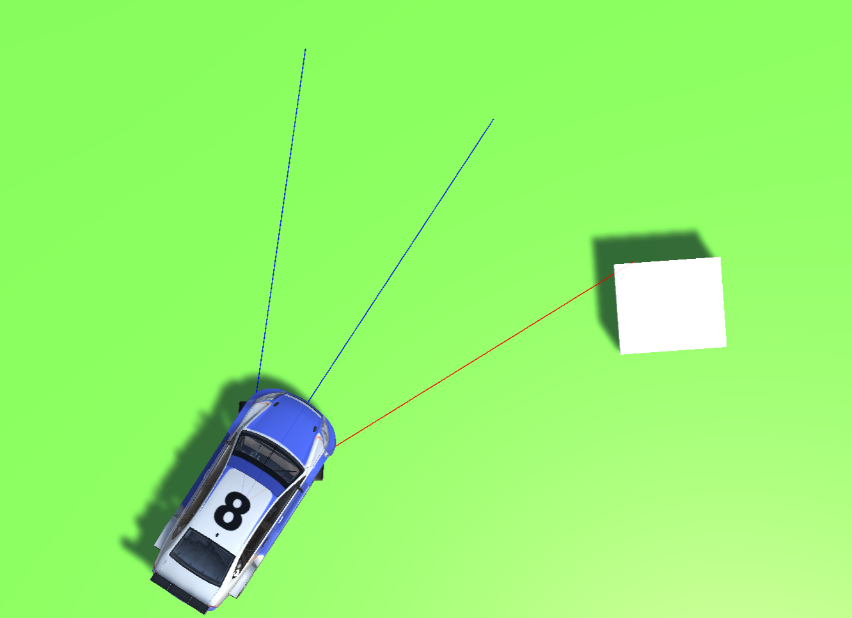
1. During **Perception:** The system collects information about the environment using the awareness techniques.

Figure : Environmental Awareness

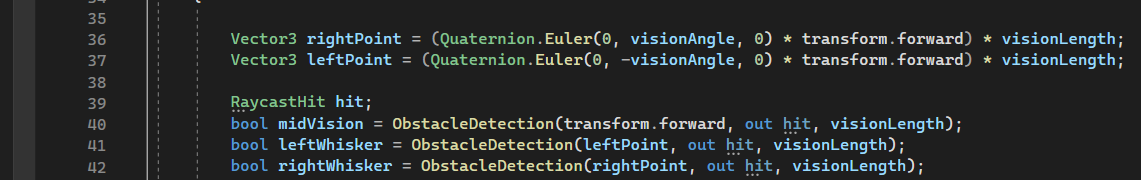
The single ray and two angled whiskies are used to detect where obstacles are for potential collisions.

Figure : Collision Ray Construction

The figure above shows the mathematical construction of the rays based on the forward of the vehicle and returns a “true” Boolean value on detection.

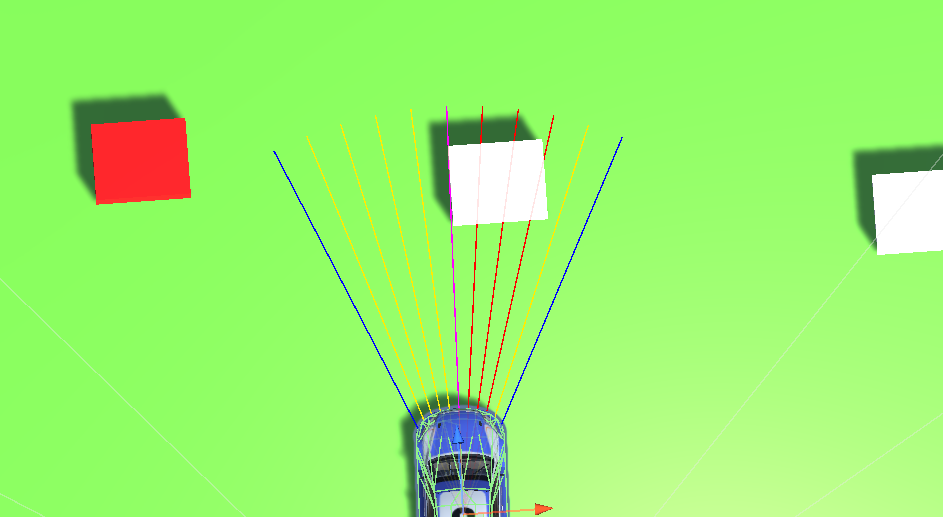
1. If **Obstacle is Detected:** when one of the raycast returns true, the minimum collision is decided based on the if the obstacle is on the two side whiskeys. A Maxmum collision is predicted if the middle ray returns true, which triggers a **SweepCheck**.
2. **SweepCheck:** SweepCheck returns the direction of the most danger as integer -1 or 1 for left and right respectively. The check is performed by shooting multiple rays from left to right within the side whiskeys.

Figure : Sweep Rays for Collision Detection

From Figure 4 above the yellow-ray indicates false collision and red for true collisions. The direction of danger is computed based on the side with the most true collision. See Appendix F, for the sweep technique algorithm.

# Pathfinding

The pathfinding for the driver AI, is defined by using traditional Waypoints systems where the nodes are carefully hand around the track circuit which can be use to guide the driver around.

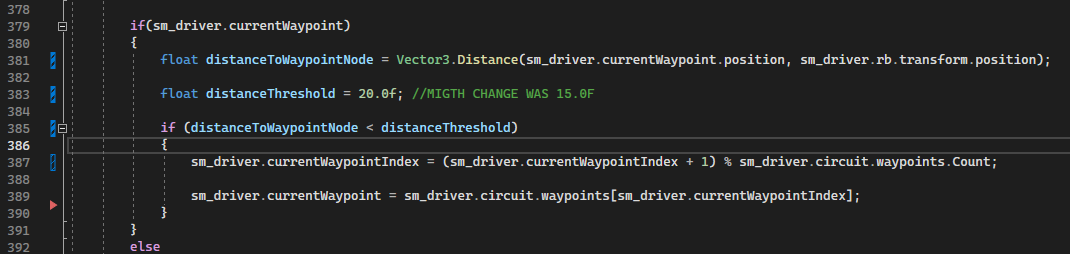
The node are stored in a list of transforms, and the drivers can keep track of current node using the index of that node in the waypoint. Then if the driver have reached a specified threshold to the node the current node integer value is incremented by one.

Figure : Update Current waypoint Index.

In the Car Engine, a function Move() with takes in parameters acceleration (0-1), brake (0-1) and steer(-1 to 1) is used to navigate the vehicle along the track with the current waypoint transform and some angle calculations the driver determine the steer rate and calls the Move function. Please   
See Appendix K, L and M for more details.

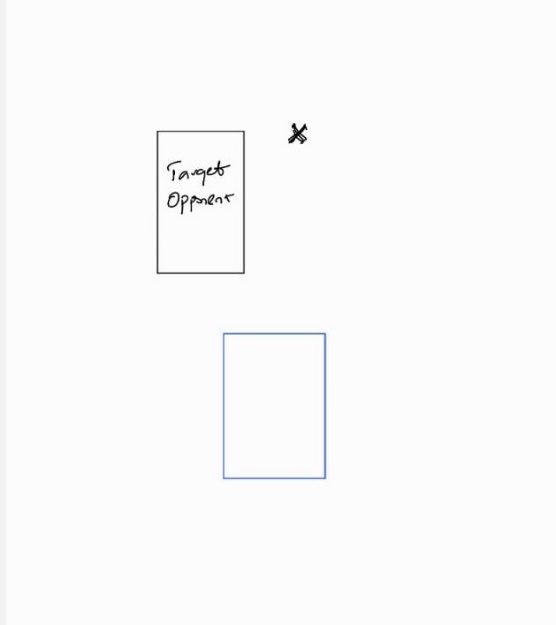
Overtaking Path, for overtaking a four node instantiatous path is calculated based on the main circuit path. The first node point is decided based on the dominant side and some offset to the opponent overtakes the target.

Figure : Most Dominant Side

For the other three points, the current waypoint is taken into consideration and the first node is compared with it and the instant direction of flow.

The instant direction of flow = next waypoint node – current waypoint node.

If the first node, is behind the current waypoint node use the instant direction of flow else calculate the next direction of flow. See the illustration below.

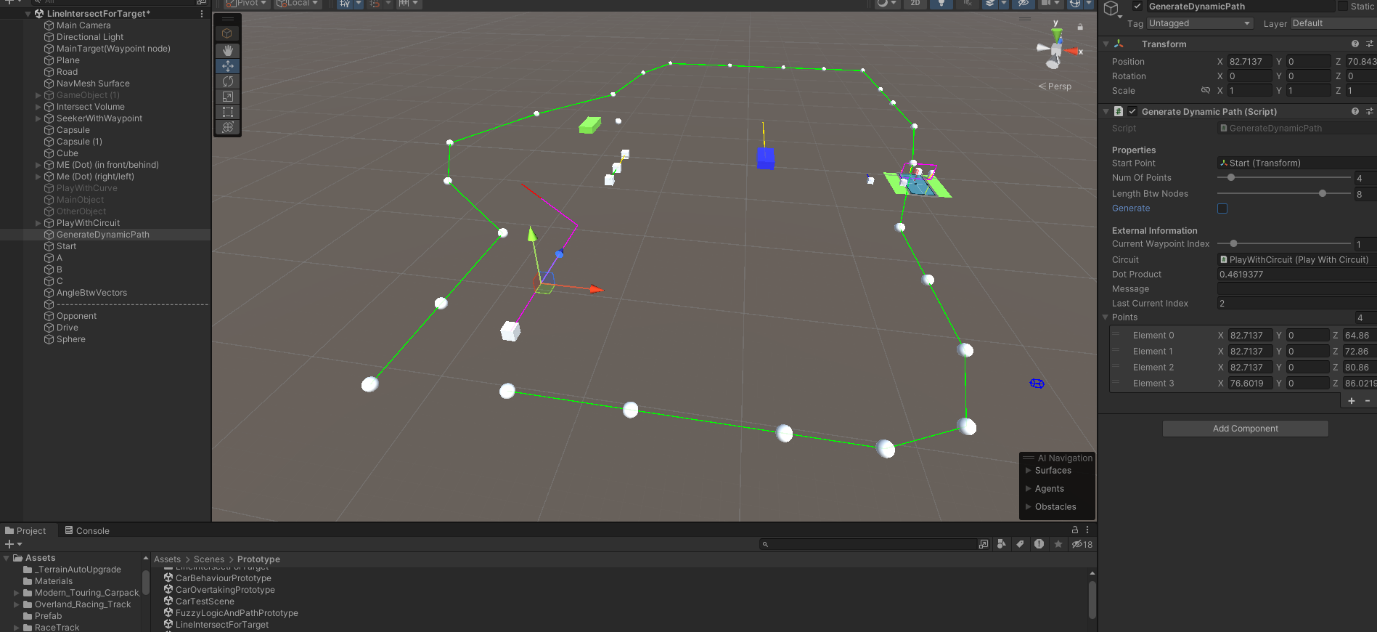
The Green line illustrates the flow of the main path, and the magenta line illustrates the dynamically created path mimicking the flow of the main path. Please see N, O and P for more details

Figure : Dynamic Path Creation

The design/level design of the track is very important

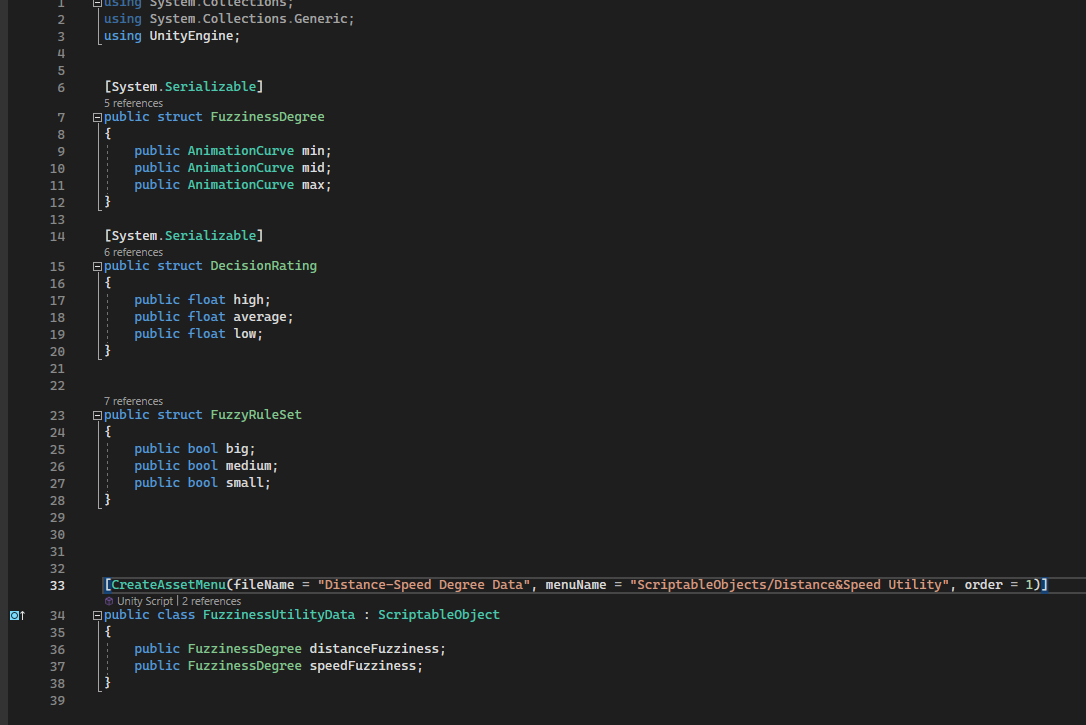
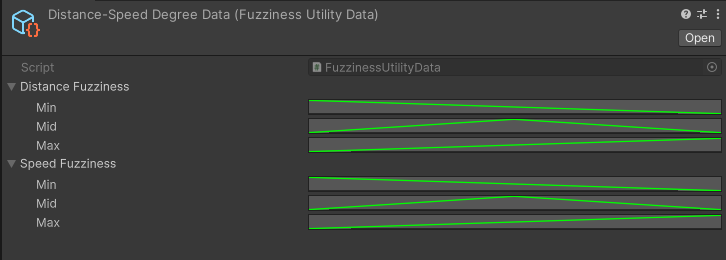
This week (18/12 – 24/12) – Complete a refactorize FSM, with some Obstacle avoidance and (Utility AI / fuzzy logic and set).

Next week (25/12 – 31/12)– BT, with some Obstacle avoidance and Utility AI (if possible)

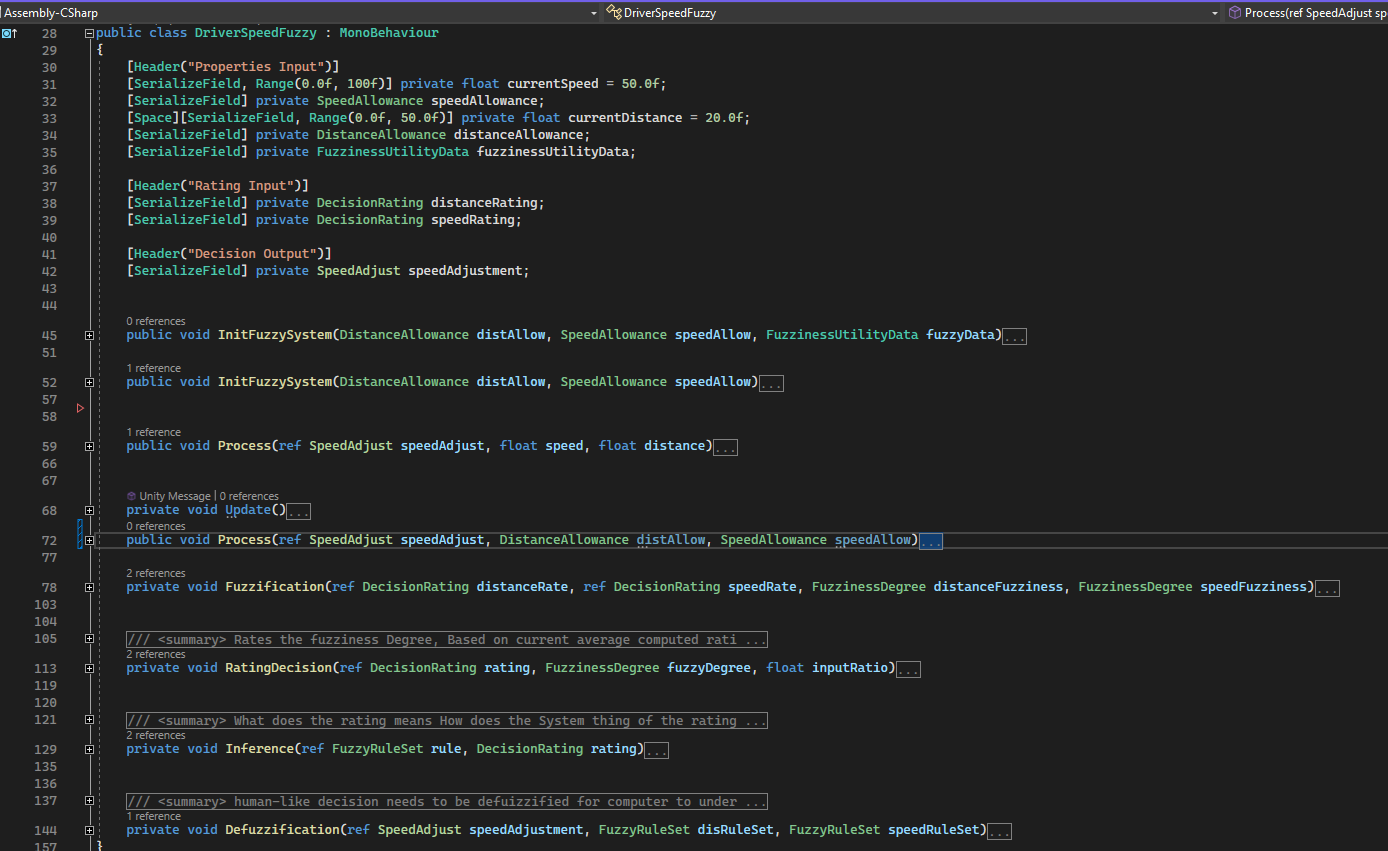
Week After the next week (01/01 – 07/01)– if not (fully integrate FSM together BT), then polish with Utility AI and better pathfinding. Look into PID or Rule Based System (if possible)

(08/01 – 10/01) – Extra stuffs

# Appendix

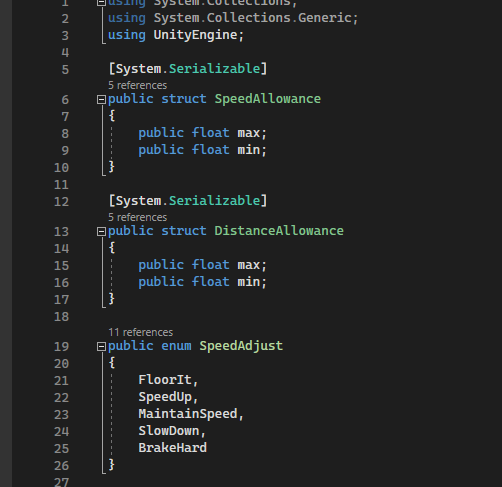
Appendix A

Appendix B

A graph with green lines

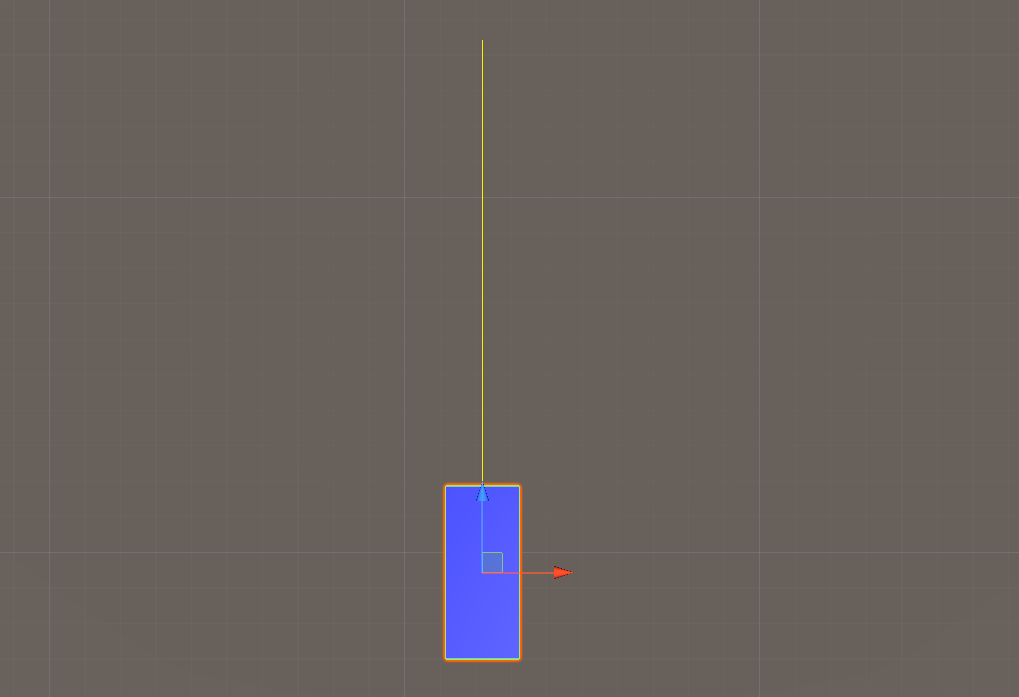
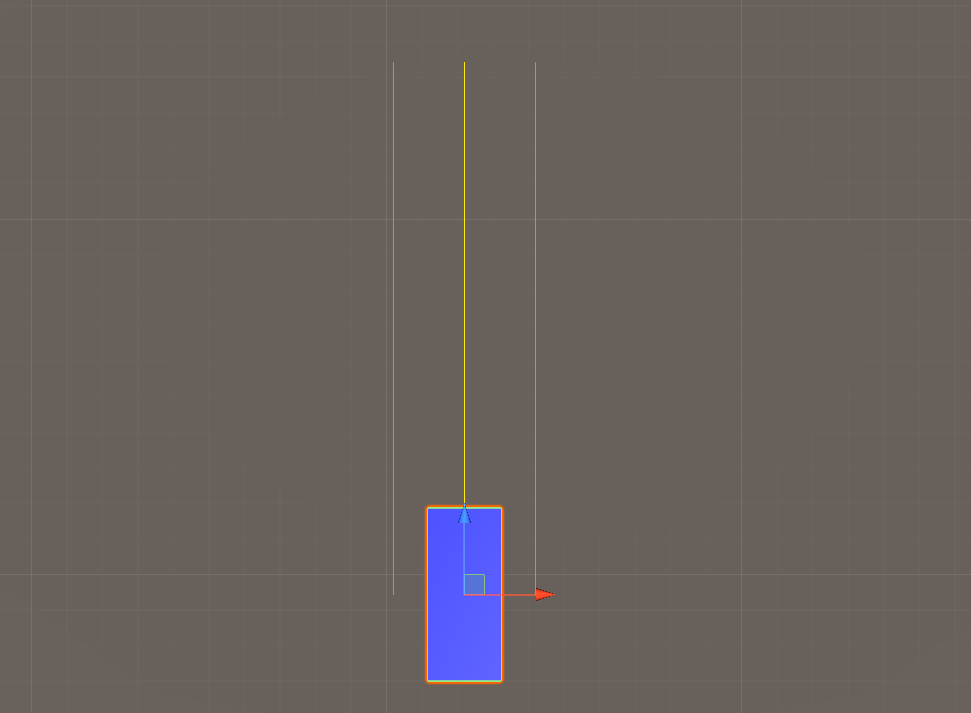
Description automatically generatedAppendix C

Appendix D

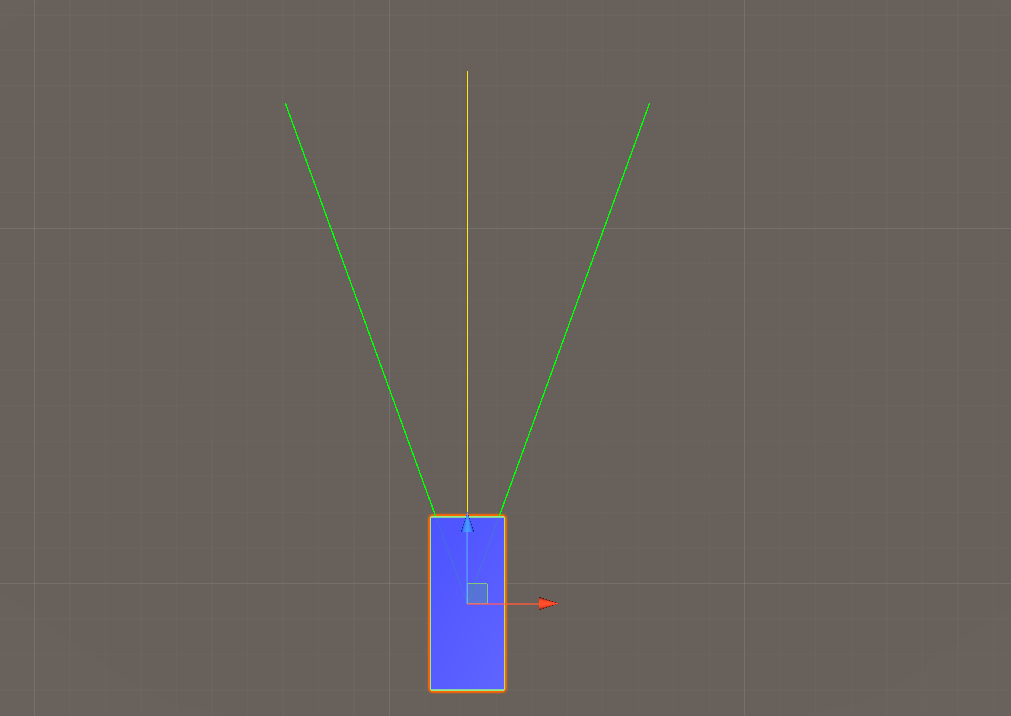


Appendix E

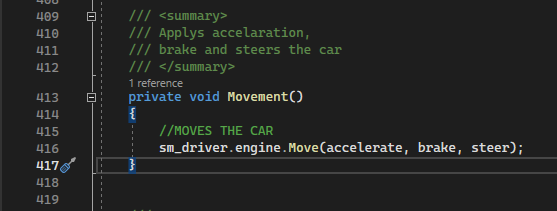
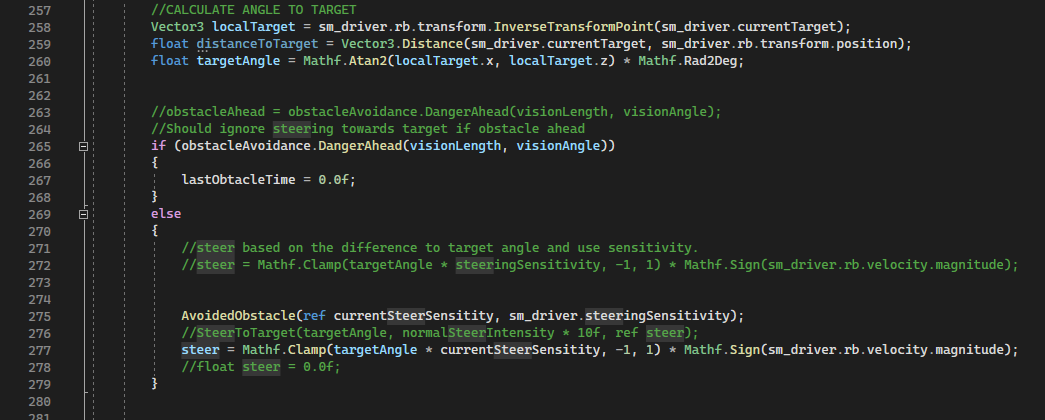
Appendix F

Appendix G

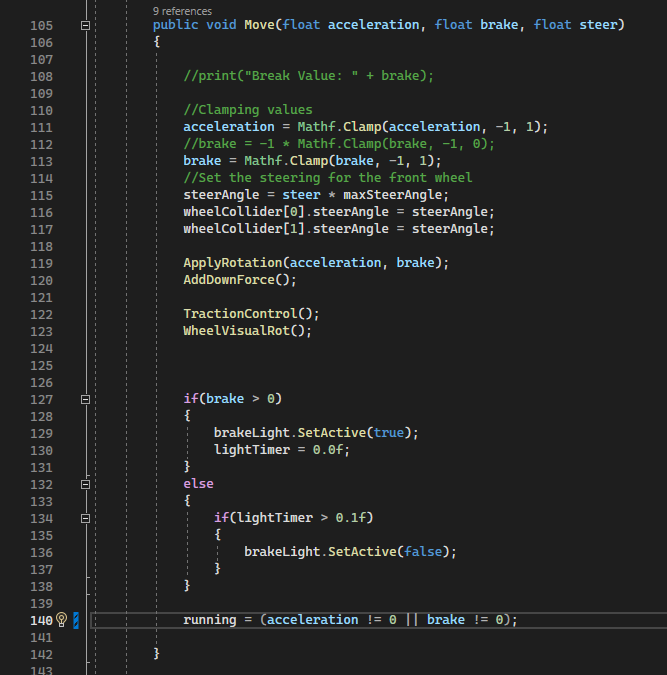
Appendix H

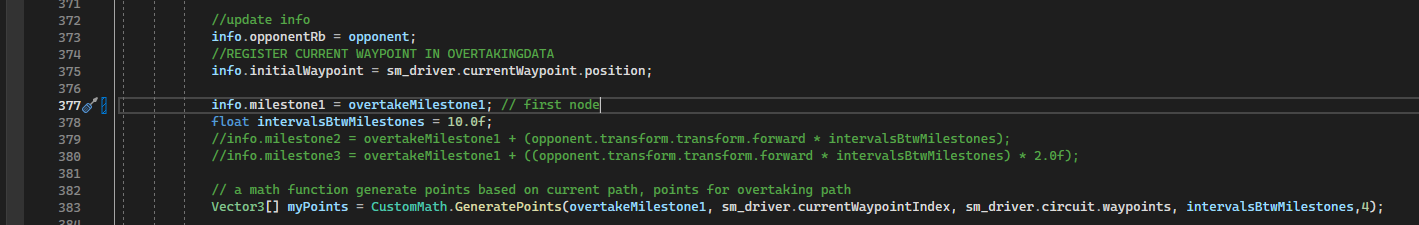
Appendix I

Appendix J

Appendix K

Appendix L

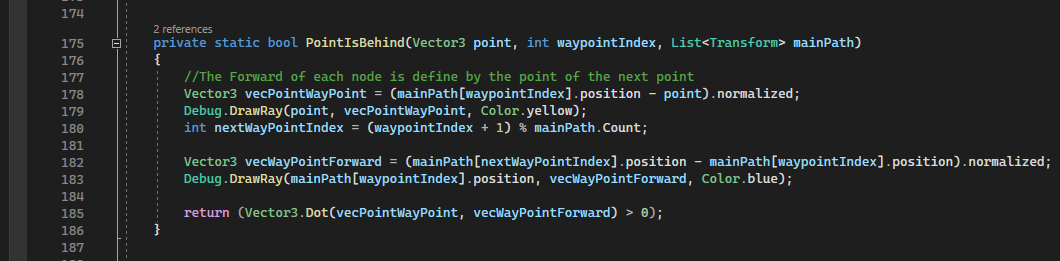
Appendix M



Appendix N

A computer screen shot of text

Description automatically generatedAppendix O



Appendix P