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](#_Toc154827844)

[AI-Driven Behaviour Characteristics 2](#_Toc154827845)

[Car Model 2](#_Toc154827846)

[Decision Making 3](#_Toc154827847)

[Finite State Machine 3](#_Toc154827848)

[Awareness 3](#_Toc154827849)

[Fuzzy Logic and Set 3](#_Toc154827850)

[Behaviour Tree 3](#_Toc154827851)

[Obstacle Avoidance 4](#_Toc154827852)

[Fuzzy Logic and Set 5](#_Toc154827853)

[Pathfinding 6](#_Toc154827854)

# 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. 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.

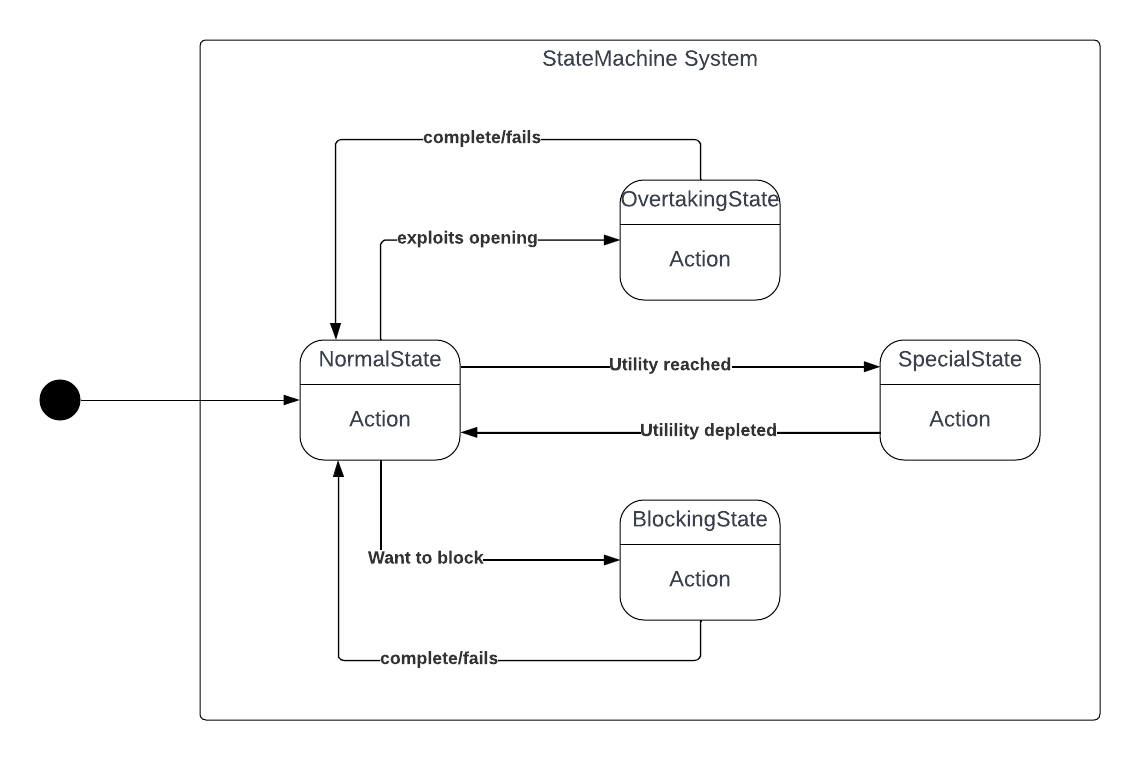
# Decision Making

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

## Awareness (perception)

The decision to switch to it is important for the AI model to understand and interpret its environment and make use of the data to make decisions.

This data/information is received through analysis by the use of multiple collision line raycasting using vector mathematics to define the line function for a better understanding of the play space to be sent to the AI system for interpretation.

Different type of collision raycast were used to analysis the play space, which includes:

1. Single ray: from the center of each car forward (mainly used for distance detection)
2. Single ray with parallel whiskey: Is where three ray, one from the center forward and two whiskeys on the side to the center of the main ray to check for possible overtaking.
3. Single ray with cone angle whiskeys: Used for opponent awareness an possible collision

## Fuzzy Logic and Set

The use of Fuzzy Logic allows to work with grey area, not just black and white. And also helped to make the drivers abit uniques in behaviours. It is used as “to what degree” or “how much”. Unlike the traditional Boolean logic which defines true or false, with fuzzy logic allows to mathematical model size concept such as “how true”, “pretty big”, “medium”, “really small”. With a system embedded with fuzzy logic AI characters can have deeper reasoning to traditional “true or false”. (McCuskey, 2000)

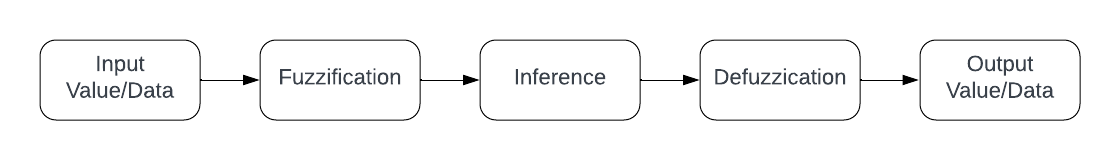
 Fuzzy Logic is used to design and develop the speed adjustment for each driver based on the distance to obstacles or corners and the current speed of travel. The aim is to use fuzzy logic to create unique driver understanding to distance and current speed in their own unique ways and a decision will be made based on their understanding.

Fig: Fuzzy Process

The above Fig is the fuzzy logic use in make the decisions for the racing agents,

1. The **INPUT VALUE/DATA**: Are hard values that tells the system specific inforamtions, for the Driver Speed System see Appendix D, The process method takes in speed, distance and also the allowance for both speed and distance.
2. **Fuzzification:** is where the system processes and determines the degree to which the input data belongs to the fuzzy set.

For this Car AI Behaviour the set which is called DecisionRating with members High, Average and Low. To rate a current speed, the average is aggregated:

With the help of Unity AnimationCurve see Appendix C, to save time with complex mathematics calculations. Animation Curve is used to define the degree of member (DOM) in the DecisionRating Set. Which also creates a uniques behaviour across the drivers by easy adjusting the animation curve.To rate the degree of member of the ratio, the Evaluation method is called on corresponding AnimationCurves.

Degree of x = animationCurve.x.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 member.

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

This model is to interpret “Big”, “Medium”, or “Small” see Appendix A, for the Fuzzy Rule Set struct. Using the Fuzzified Data, Speed or Distance is:

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:** Where the data is ready to use by AI. 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

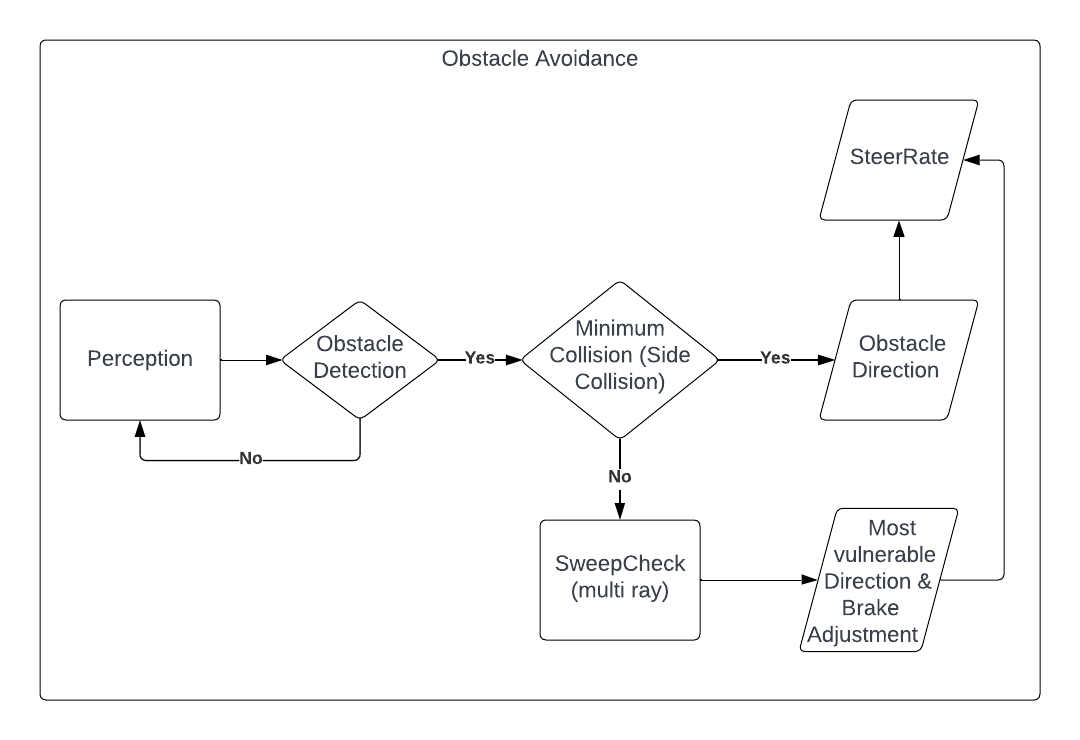
 Predicting collision or threat is a very useful thing to be implemented into an AI System, that involves more than one AI agent (DaGraca, 2017). The Obstacle Avoidance System is heavily reliant on the Awareness set up for the system.

Figure :Obstacle Avoidance System Flow.

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

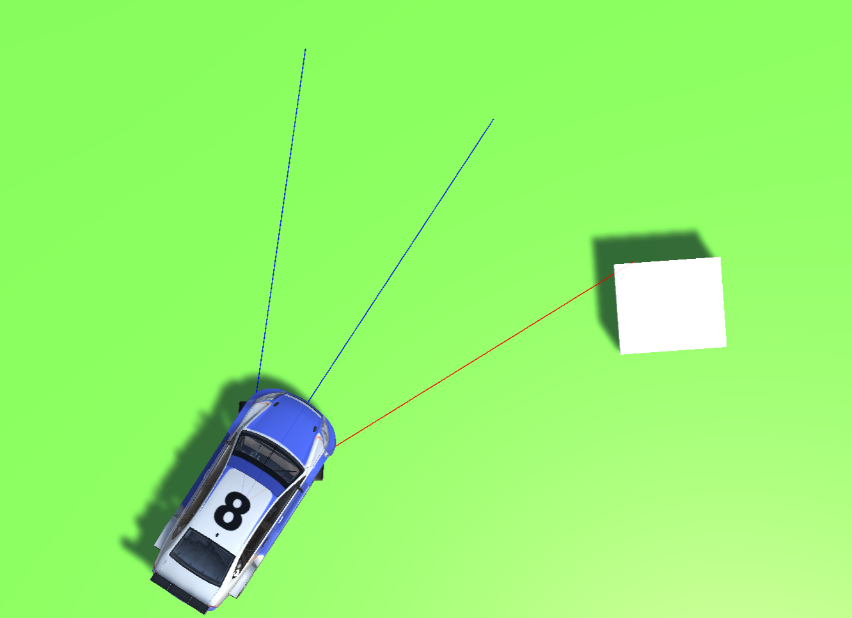
1. During **Perception:** The system collects information about the environment using the awareness techniques.

Figure : Environmental Awareness

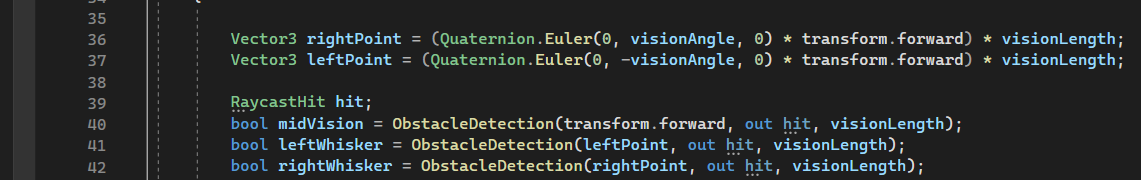
The single ray and two angled whiskies are used to detect where obstacles are for potential collisions.

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.

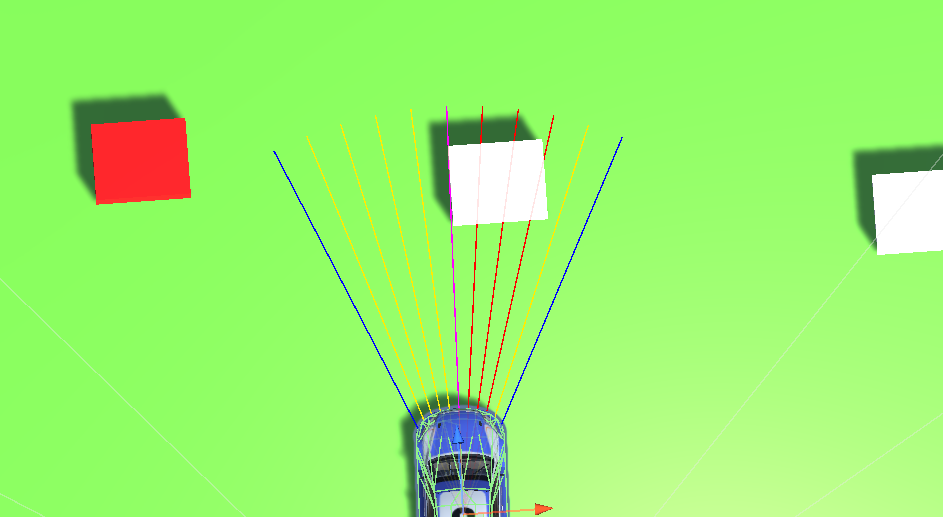
1. If **Obstacle is Detected:** when one of the raycast returns true, the minimum collision is decided based on the if the obstacle is on the two side whiskeys. A Maxmum collision is predicted if the middle ray returns true, which triggers a **SweepCheck**.
2. **SweepCheck:** SweepCheck returns the direction 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 design/level design of the track is very important

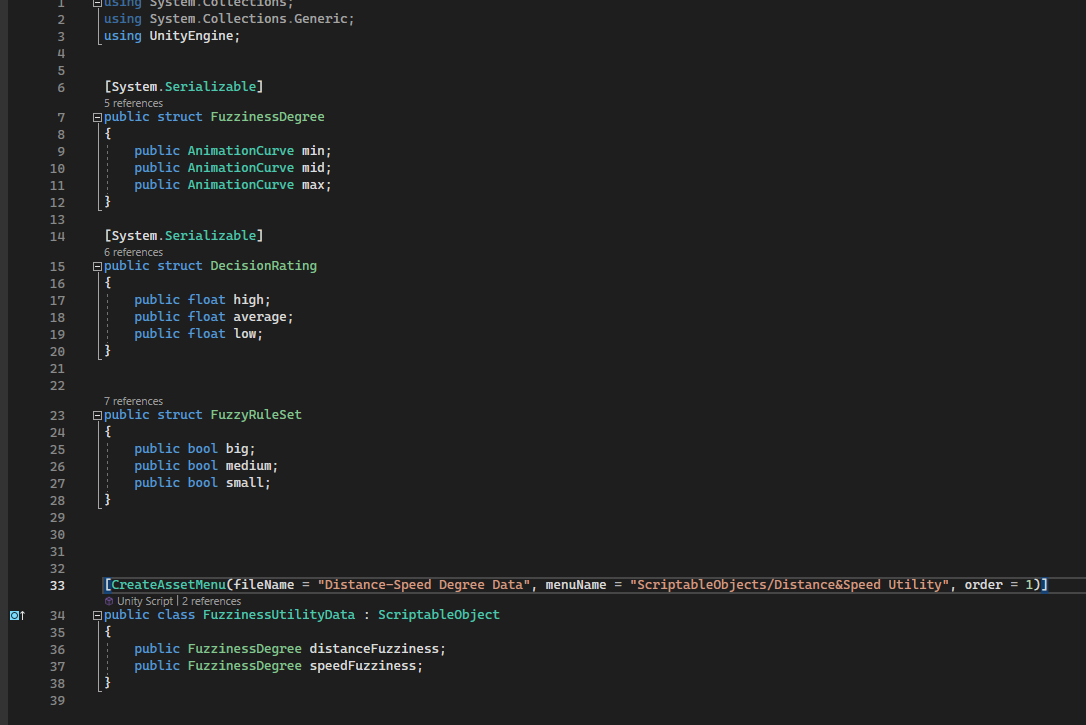
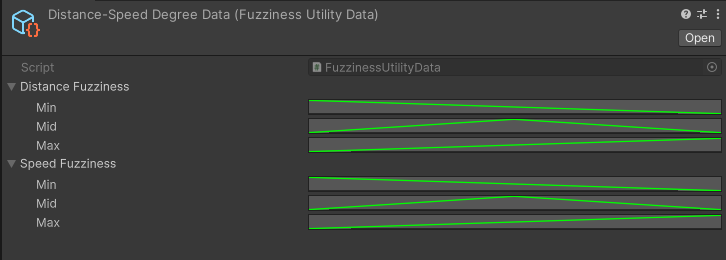
This week (18/12 – 24/12) – Complete a refactorize FSM, with some Obstacle avoidance and (Utility AI / fuzzy logic and set).

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

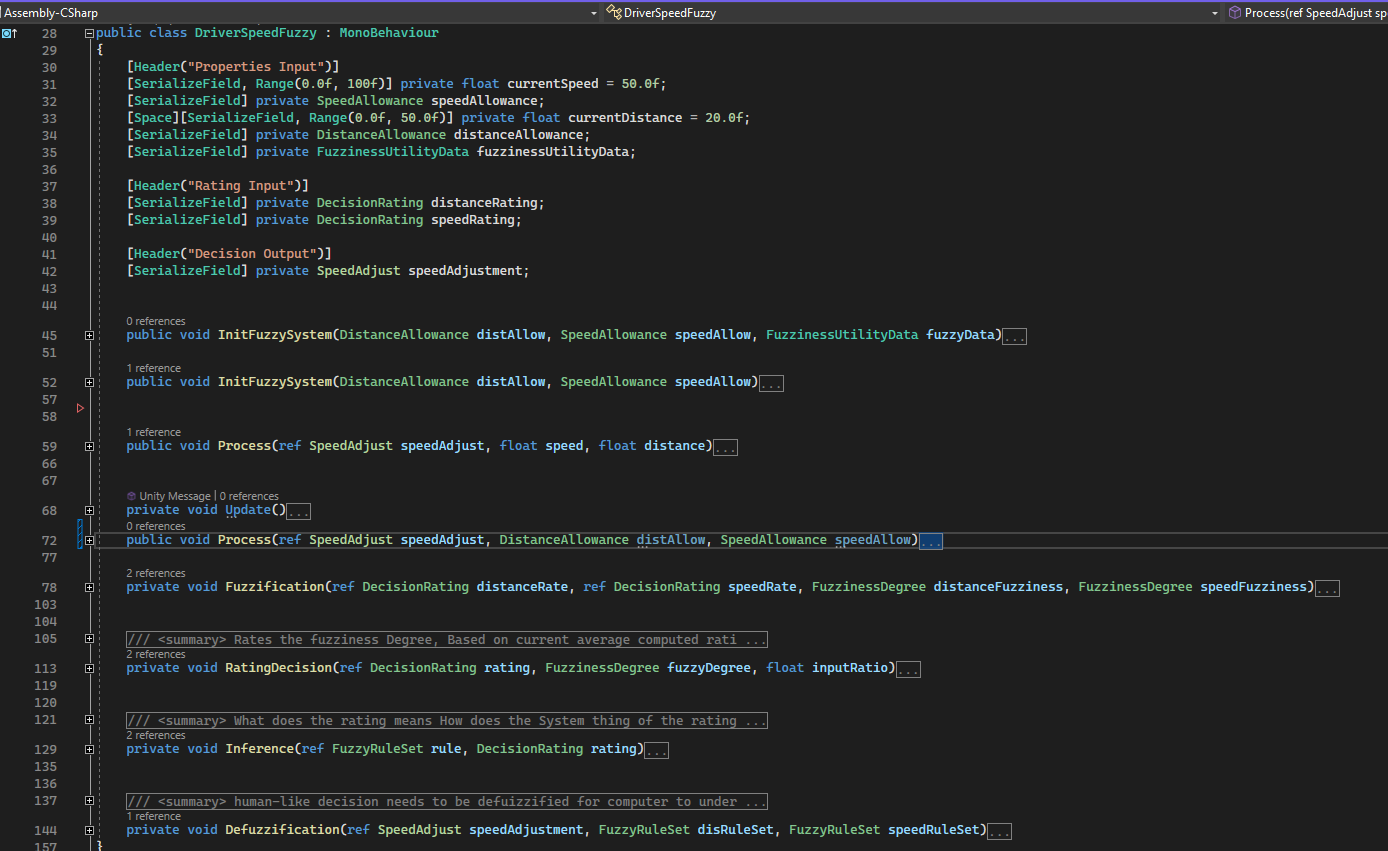
Week After the next week (01/01 – 07/01)– if not (fully integrate FSM together BT), then polish with Utility AI and better pathfinding. Look into PID or Rule Based System (if possible)

(08/01 – 10/01) – Extra stuffs

# Appendix

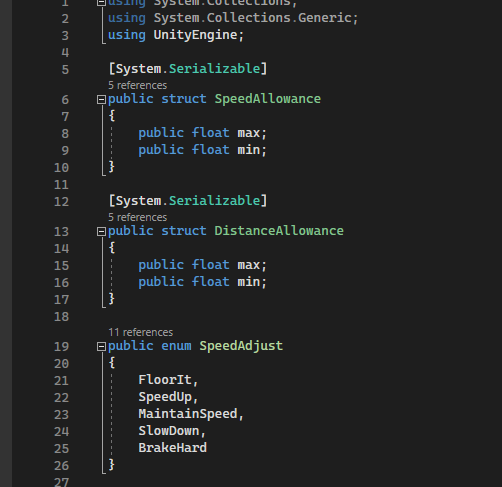
Appendix A

Appendix B

A graph with green lines

Description automatically generatedAppendix C

Appendix D



Appendix E

Appendix F