CT6GAMAI - Report

By UP814853

Contents

Contents

[Introduction 1](#_Toc26786487)

[Analysis 2](#_Toc26786488)

[Pathfinding 2](#_Toc26786489)

[Steering Behaviours 2](#_Toc26786490)

[State Machines 2](#_Toc26786491)

[Reflection 2](#_Toc26786492)

[Conclusion 2](#_Toc26786493)

[References 2](#_Toc26786494)

# Introduction

Racing simulations in games are AD-HOC systems which simulate real life drivers based on factors created by the programming team such as personality, vehicle type, race track and more. These features are the primary focus of a development team when creating a single player racing game.

The problem this project focused on was creating a racing game where AD-HOC where Artificial Intelligence would race each other around a circuit and consider strategy based on their personality. The main problems would be steering behaviours, state machines and pathfinding.

Creating realistic Artificial Intelligence for racing games is important, you want the Artificial Intelligence to be able to race each other but in a fair manner on the player and themselves. The Artificial Intelligence that is raced against can either make or break a game.

Forza Horizon 4 (Turn 10 Studios, Playground Games, 2018) and F1 2019 (Codemasters, Swordfish Studios, 2019) are two of the most popular and state of the art racing games both having advanced racing Artificial Intelligence which can be considered to be state of the art. Forza Horizon 4 (Turn 10 Studios, Playground Games, 2018) even has a system in place where the driver profiles are built using a Learning Artificial Intelligence system (Forza Support, 2019).

Both of these games are considered to be state of the art as they successfully implement fun and challenging Artificial Intelligence system within their games which can react to the best of players and can be modified by the players within menus.

For the artefact focused on getting the Artificial Intelligence to go around the track and be able to pit when they need fuel based on desires. With the 3 main systems being Steering Behaviours, Pathfinding and State Machines.

Epsilon is a top down racing game which has pre-set roads or tiles that can be placed down to create a track.

# Analysis

## Pathfinding

Pathfinding can be done in multiple algorithms each used for different systems. These are Breadth First Search, Depth First Search, Dijkstra’s algorithm and A\*

For the pathfinding A\* Search algorithm (Nillson, Hart, & Raphael) was chosen. The technique was chosen as it is the quickest way to search a graph speeding up Dijkstra’s algorithm (Buckland, 2004). It is also the standard within video games being the most optimal technique currently.

This technique was chosen as it was a quick way to get the AI to find their way across the track and made them behave in a more realistic way choosing the optimal path to race on and worked on any track.

It was implemented by having the AI ray cast down and get their source node, they would then get the next checkpoint they wanted to go to and ray cast to get the target node. From there the algorithm would search and find a route and return it. The ASTAR method was abstracted so that I couldn’t break the implementation and would only need to hand in the source and target node for it and it would be able to search a given graph. The implementation of A\* was inspired by the implementation in Programming Game AI by example (Buckland, 2004) and the systems were adapted from this book and modified it to work within set parameters and code

The Nav graph was generated by creating a grid, this grid was set by an area e.g. 100 x 100. The nodes would be placed, they would then be given a “Selector” which would look above it and see if there are any roads, if there isn’t it deletes its self, if there is it checks for an obstacle or wall on the road, if there is either of these obstacle’s it deletes its self. This implementation of the Nav Graph saves unnecessary checking for walkable tiles and instead only checking the tiles it can walk on and decreases the run time.

## Steering Behaviours

Steering behaviours are techniques to control the AIs movement in certain conditions. Techniques can include obstacle avoidance, wall avoidance, evade, seek, join, flock, wander and more. The techniques were adapted from Programming Game AI by example (Buckland, 2004) and were changed to work within Units. I used steering behaviours as they are used in F1 2019 (Codemasters, Swordfish Studios, 2019)

These techniques calculate a velocity which is added to a velocity sum and returned to the AI. In a racing game these behaviours were crucial to get correct otherwise the vehicles would crash into each other or have unrealistic behaviours. For Epsilon the following techniques are used within the game. Seek, Overtake and Obstacle Avoidance. Seek works by getting the target position then subtracting the normalised vehicle position and multiplying it by the vehicle max speed, it is then subtracted by the vehicle velocity. The overtake function simple checks for any AI in a projected cube, if it finds some it runs an algorithm to get that AI and calculate a force to go around it much like obstacle avoidance. I found that returning x as z and visa versa lead to greater results. Obstacle avoidance works in the same way.

The steering behaviours would be added to the velocity sum which would tell the AI what way they need to accelerate the order it is added in is important, seek needs to be last as all the other calculations are more important. If seek was first the AI would rather go towards the target then overtake. Commonly obstacle avoidance is off as the pathfinder detected and pathed around these obstacles anyway so overtake was the only other force.

## State Machine

The AI logic could have been done using 3 systems, State machines, Fuzzy logic or goal driven/behaviour trees.

The project used State machines as they would be the most effective towards the game. The AI would simply assess their desires using utility theory and then the highest desire was chosen and executed this way I laid the foundations to be able to have dynamic profiles (Buckland, 2004). As the AI was just driving around a track I only needed 3 desires, Driving, Overtake and Pit. Each were controlled by their driver profile with the default AI only wanting to overtake if they meet certain conditions and pitting if they estimate they do not have enough fuel to finish the race.

State machines were chosen as they were simple to implement and design (Yannakakis & Togelius, 2018) and as the project was small there was no need for more complicated systems such as Fuzzy Logic or goal driven systems (Yannakakis & Togelius, 2018).

The state machine was designed within 3 states it could be in, each state was unique but all of them drove the car around the track ensure the AI still raced. The 3 states would be controlled by their desire which was updated in their driver profile.

State machines allowed the implementation to be simple and available to be abstracted to be used by different driver profiles with scope of having profiles like aggressive, defensive, strategic and default

# Reflection

## Main Results

The goal of epsilon was to create a project which would have a basic AD-HOC racing system for a PC game. The AI behave in a set pattern according to the driver profiles given to them and the vehicles they are racing. The game has a set amount of random generation within it for the vehicle parameters. The project will simulate the race however with breaking not being implemented in the vehicle with the highest speed would be the one to win the race most of the time. The AI drive to the route and follow it well and the state machine controls their desires and pits them when they are needed. The faster AI will always try and overtake the slower AI to win the race.

## Issues

Pathfinding in racing games is the incorrect implementation, the project should of stead just used steering behaviours to simulate the race, pathfinding is and has caused the biggest issue with the project with the AI trying to overtake each other if they miss a seek checkpoint and its not removed they may turn around or just suddenly turn left essentially given up that position. This has created unforeseen moments in races where due to this incorrect implementation the AI have weird behaviour. Often in the project the AI would turn around or not race at all because the pathfinding was not behaving.

Pathfinding would cause the most lag and whenever the game freezes its due to A\* and there being to many nodes however removing pathfinding would break most systems in the game. Pathfinding and nodes is also causing the main issue with AI turning around or going in the wrong direction this is because I cannot remove nodes behind the AI as it would not corner well.

The state machine could have been done with fuzzy logic and would have been a better implementation. Using fuzzy logic for this project would have allowed the developers to write better logic for the AI to follow during the races and implemented techniques such as a racing lines and team mates and ignoring orders and allowed for a more diverse AI system then state machines. This could have also been done with behaviour trees as well.

The main limitation of this project was time. Due to the project having a 2-month time period and conceptually created with a month till hand in the project was created to be a foundation of a racing game however is missing implementations such as defending, random items, shortcuts, different driver profiles, randomly generated tracks and procedural generation of the game world.

In extreme cases the AI like to dart diagonally into a walk and just slam into its multiple times. Developers theorise this is caused by the steering behaviours not avoiding walls as the pathfinding done it for them and when they slammed into a wall there is no programming implemented for them to rescue themselves.

## Other Implementations

Steering behaviours – the ways that certain steering behaviours are implemented is messy and unclean with the code being cost inefficient and can lead to lag. All the systems of these behaviours should be re written with better implementations designed for the racing game.

One improvement that could be made would be with the code length and structure, certain parts of the code is repeated and messy with implementations in the just created phase and nothing cleaned up or optimised.

# Conclusion

In conclusion with 2 months the project successfully created a racing simulator where 8 AIS race to a lap counter, pit and use created techniques to race each other. The AI use obstacle avoidance to overtake each other, state machines to decide what they are doing and pathfinding to find out where they are going. The project could have been done better with more time and on reflection certain systems where not the best implementation however for what the project needed and with the time given Epsilon is a racing game with small bugs which can be ignored.

# References

Buckland, M. (2004). *Programming Game AI by example.* Plano: Wordware Publishing, Inc.

Codemasters, Swordfish Studios. (2019, June 19). F1 2019. *F1 2019*. Southam, Warwickshire, United Kingdom: Codemasters.

Forza Support. (2019). *Drivatars*. Retrieved from Forza Support: https://support.forzamotorsport.net/hc/en-us/articles/360005302934-Drivatars

Nillson, N. J., Hart, P. E., & Raphael, B. (n.d.). A\* search algorithm. *A\* search algorithm*.

Rusildo. (2015, September 3). Wander steering behaviour in 3D. Retrieved from https://gamedev.stackexchange.com/questions/106737/wander-steering-behaviour-in-3d

Turn 10 Studios, Playground Games. (2018, September 2018). Forza Horizon 4. *Forza Horizon 4*. Redmond, Washington, United States Of America: Xbox Game Studios.

Yannakakis, G. N., & Togelius, J. (2018). *Artificial Intelligence and Games.* Berlin: Springer.