ML-Racing

Project Engineering

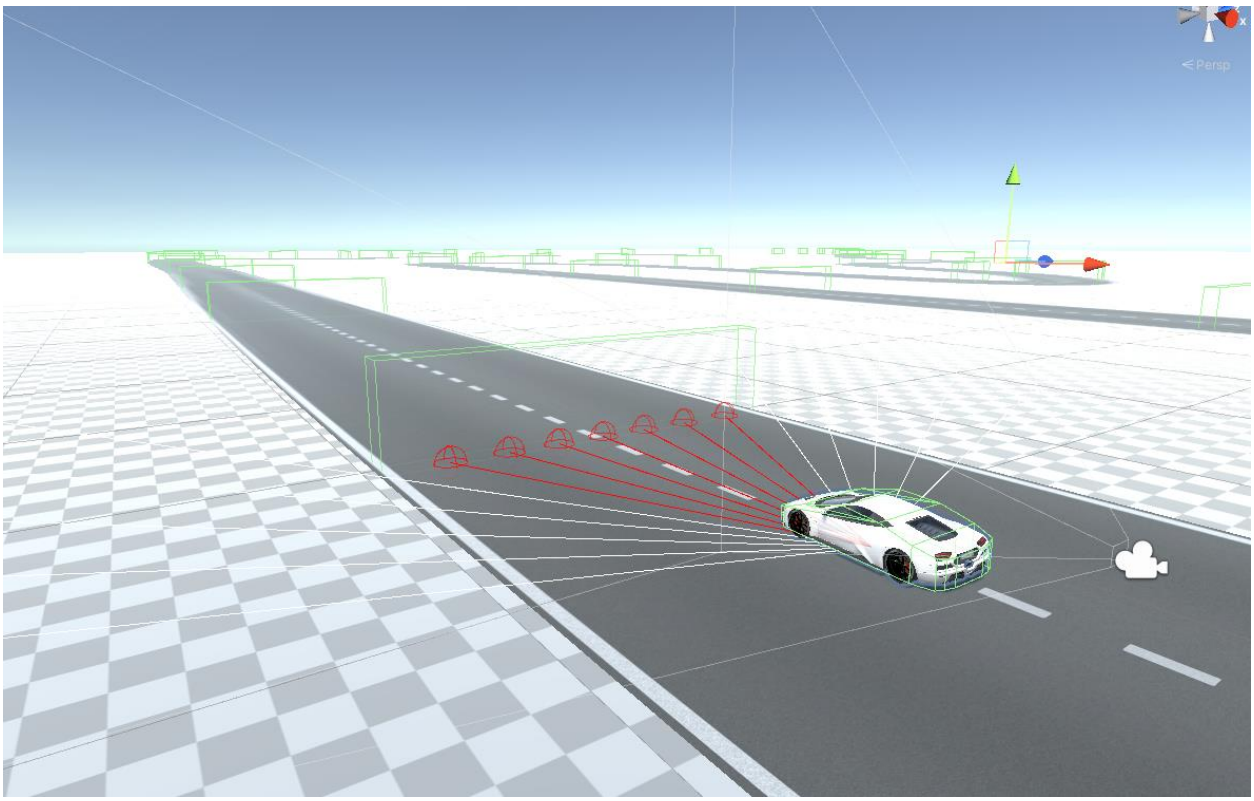
Year 4

Conor Dempsey

Bachelor of Engineering (Honours) in Software and Electronic Engineering

Galway-Mayo Institute of Technology

2021/2022


< Persp

Declaration

This project is presented in partial fulfilment of the requirements for the degree of Bachelor of Engineering (Honours) in Software and Electronic Engineering at Galway-Mayo Institute of Technology.

This project is my own work, except where otherwise accredited. Where the work of others has been used or incorporated during this project, this is acknowledged and referenced.

_____

Acknowledgements

I would like to acknowledge and thank Michelle Lynch, Paul Lennon, Niall O'Keefe and Brian O'Shea for their continued support in working on this project. Without there support many of the goals I have achieved in this project would not have been possible.

Table of Contents

1 Summary

The goal of ML-Racing is to create a game using Unity and several plugins in the unity registry including ML-Agents that use machine learning to train an AI/Agent to drive a car around any track without specific track knowledge or user input. The agent will learn over time when given inputs for forward movement and turning movement how to navigate any track, this will create a neural network model that can be used in another game environment to compete against players. The game will allow a user to race against AI, the user controller their car using input keys and the AI controlled using the neural network created using reinforcement learning.

Some of the main features of the project will be the integration of machine learning into a learning environment, this learning environment will be one scene in unity used to train Agents to drive. The next scene will be a playable level that allows the user to play against the trained Agents. The game will have one track and one car to begin with a single car controller used to drive all cars.

I plan to start by creating a track and car model using assets in the unity store. Once these are complete I will; bring them both into a scene and begin coding scripts to control the car and the checkpoints needed to show the Agents where to move. Once complete I will add a reward system and add the ML-Agents components to the car Agent, this will then allow me to use training and create neural networks.

The main technologies used will be the Unity game development engine, ML-Agents, C# scripts and python.

Of the goals I set out I have completed the integration of the car and track models into a game environment, these models include scripts to control the car and correctly identify if the user is navigating the track correctly. I have not completed the training environment although I have learnt a great deal about machine learning and of MLAgents.

The main conclusions I can draw from this project are that I will explore the topics covered in this project as future career paths and also that I have learnt a vast deal of knowledge on topics I previously had no experience with.

Ollscoil Teicneolaíochta an Atlantaigh
Atlantic Technological University

ML-Racing

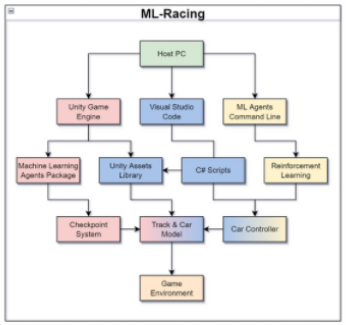
A Unity Racing Game With Machine Learning

Name: Conor Dempsey
Course: BENG(H) In Software and Electronic Engineering
Module: Project Engineering
Supervisor: Michelle Lynch

Introduction

ML Racing is a racing game designed in the Unity Game engine that incorporates machine learning to create an AI that will race competitively against the player. The game applies a neural network created using MLAgents, a software designed by Unity, by using reinforcement learning to teach the kart over time how to navigate a set track.

Architecture Diagram



Summary

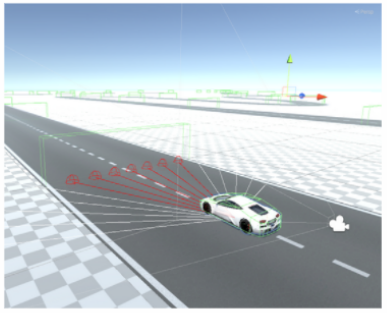
Using C# scripts the game implements colliders which are the targets for the kart to reach. Each time the kart reaches a collider it is rewarded points for doing so and when it fails it is deducted points giving a reward/penalty system. If left for long enough the kart will go through hundreds of scenarios to try to gain the highest reward possible.

Technologies Used

TensorFlow
unity
Visual Studio Code
blender
PyTorch
python

Results and Conclusion

The Program works as expected and over time the standard of reward increases from being between 1.0 - 2.0 all the way to 100 - 110. The results can also be seen when the neural network is applied to a kart for game purposes. Without the neural network applied the kart will aimlessly drive in circles, but once applied the kart will drive around the track almost as quickly as the player would.



QR Code to GitHub

3 Introduction

3.1 Scope

The main **goal** of my project is to firstly create a high-quality game environment including tracks based on real life Grand Prix tracks and models designed around real-life cars. I had planned to design my own models at first using blender however this was far more complex than I had first anticipated. The next goal I hope to complete is to create a realistic car controller that includes controls for forward, reverse and turning. After some research I found out that the physics of a car can be programmed into a car controller including grip, torque, drag etc., I hope to incorporate some of these physics.

To follow up on the above paragraph some of the main **deliverables** will include a completed tutorial on ML-Agents and to have a car model and track completed by the end of 2021. In 2022 I hope to deliver version 1 of the game which will take the car and track models and add a controller to the car, creating a playable racing game. Next, I hope to deliver a training environment for the AI in my game, allowing me to crate neural networks. The final deliverable will be the completed game.

Some of the **constraints** I have considered is my limited knowledge of Unity and machine learning, I have planned accordingly, making sure I spend most of semester 1 working on research and tutorials on these topics. I also have the constraint of not having the software needed available in college. I will investigate this in the future.

3.2 Motivation

The motivation for this project came from a long-time interest from gaming and game development, I always wondered how games like Forza Horizon 4 and F1 2021 were created and how exactly the AI was able to race competitively against a player. After some research into game engines, I decided to opt with using Unity, one of the packages that are included in the unity registry is ML-Agents, this is a software that uses another software aspect I have been interested in for a long time, Machine Learning. After reading about AI and specifically seeing Tesla have done with their autonomous cars, I decided I wanted to incorporate this into my project. Once I had decided this, the idea for ML-Racing, a racing game using machine learning came to mind.

4 Background

In this section I will introduce some background on the main components of my project and any relevant information on each.

4.1 Game development

Game Development is the art of creating games and describes the design, development, and release of a game. It may involve concept generation, design, build, test, and release. (*What Is Game Development?*, n.d.). Game development can include programming, sound design, visual design, and game design. All of these are used together to create an interactable program allowing user input which translates to output on screen.

4.2 Unity

Unity is a 3D and 2D game engine developed by Unity Technologies that allows game developers to use several tools to create games. (*What Is Unity? – A Guide for One of the Top Game Engines – GameDev Academy*, n.d.) The engine can create 3D and 2D objects that can be programmed to complete several tasks using C# scripts, these scripts can also be applied to models that are imported into the library. These models can be designed in Blender or downloaded online. Another feature of Unity is the large catalogue of open-source assets in the Unity store, ranging from models to programs this can be a useful for beginners to Unity.

4.3 Machine Learning

Machine learning is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention. (*Machine Learning: What It Is and Why It Matters | SAS UK*, n.d.) Machine learning works in a similar way to the human brain, by gaining knowledge and understanding through training environments it can carry out any task when given a set of inputs and specific goals.

4.4 ML-Agents

ML-Agents is an open-source toolkit that can be downloaded from the unity registry that allows games to serve as a training environment for intelligent agents that can be then brought back into game environment as fully trained AI. The toolkit requires the Unity package that can be downloaded from the Unity registry, and a python API that can be used to initiate the learning process.

5 Project Architecture

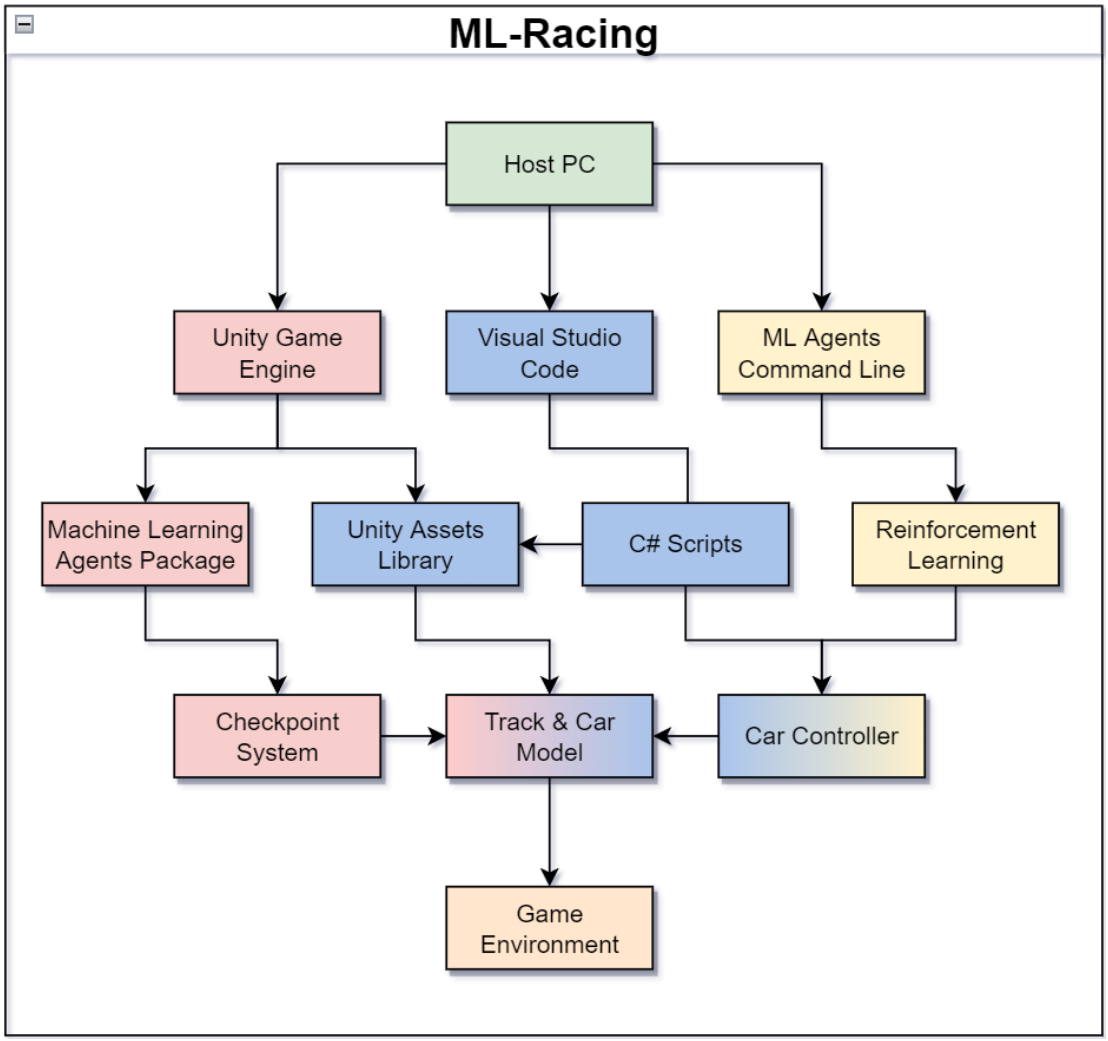


Figure 5-1 Architecture Diagram

6 Project Plan

Project Timeline

Conor Dempsey
Project

Project Start:	Mon, 9/20/2021
Display Week:	1

TASK		PROGRESS	START	END
First Semester Q1				
1	Research project ideas	100%	9/20/21	9/27/21
2	Research Unity / Unreal	100%	9/27/21	10/3/21
3	Research Machine Learning	100%	10/3/21	10/7/21
4	Download and test game engines	100%	10/7/21	10/13/21
5	Project Proposal	100%	10/13/21	10/20/21
First Semester Q2				
Task 1	Further research	80%	10/20/21	11/12/21
Task 2	Develop game environment / mechanics	20%	11/4/21	11/25/21
Task 3			11/5/21	11/25/21
Task 4	Integrate machine learning into game		11/25/21	12/5/21
Task 5	Christmas demo preperation		12/5/21	12/10/21
Second Semester Q1				
Task 1	Refine game environment / mechanics		1/18/22	1/30/22
Task 2	Improve machine learning algorithm		1/30/22	2/9/22
Task 3	Work on game level to include AI		2/9/22	2/21/22
Task 4	Improve user experience		2/21/22	3/3/22
Task 5	Search for bugs/issues		3/3/22	3/17/22
Second Semester Q2				
Task 1	Resolve any remaining issues		3/17/22	3/24/22
Task 2	Poster		3/24/22	3/31/22
Task 3	Final report		3/31/22	4/7/22
Task 4	Video		4/7/22	4/14/22
Task 5	Final Presentation		4/14/22	4/21/22

Link To extended Gantt Chart: Simple Gantt Chart1.xlsx

7 Unity Code

7.1 Car Controller

The car controller script is used to drive the car forward and backwards and turns the car left and right while implementing physics like acceleration and torque. The script takes inputs from the directional keys and space bar.

"GetInput()" is used to get the inputs from the user, the Input.GetAxis function is part of the Unity Engine directive and collects inputs from the UP and DOWN keys represented by vertical input and the LEFT and RIGHT key represented by horizontal input. Finally Using the space key, we can input brakes.

```
private void GetInput()
{
    horizontalInput = Input.GetAxis(HORIZONTAL);
    verticalInput = Input.GetAxis(VERTICAL);
    isBreaking = Input.GetKey(KeyCode.Space);
}
```

"SetInputs()" is used to send data to the ML-Agents script, this will allow the agent to set the vertical and horizontal input mentioned above.

```
public void SetInputs(float verticalInput, float horizontalInput)
{
    this.verticalInput = verticalInput;
    this.horizontalInput = horizontalInput;
}
```

"HandleMotor()" is used to rotate the wheel colliders which will move the car forward. motorTorque is also part of the Unity Engine directive and is multiplied by the vertical input which can give values of either 1 or -1 (forward or reverse). This is only applied to the front wheel colliders to make the car front wheel drive.

```
private void HandleMotor()
{
    frontLeftWheelCollider.motorTorque = verticalInput * motorForce;
    frontRightWheelCollider.motorTorque = verticalInput * motorForce;
    currentbreakForce = isBreaking ? breakForce : 0f;
    ApplyBreaking();
}
```

"ApplyBreaking()" is similar to HandleMotor(), it will apply the breaking force of 1 or 0 (breaking or not breaking) and slow the wheel colliders down until eventually they stop.

```
private void ApplyBreaking()
{
    frontRightWheelCollider.brakeTorque = currentbreakForce;
    frontLeftWheelCollider.brakeTorque = currentbreakForce;
    rearLeftWheelCollider.brakeTorque = currentbreakForce;
    rearRightWheelCollider.brakeTorque = currentbreakForce;
}
```

"HandleSteering()" is used to turn the wheels and will multiply the max steer angle (the max angle the wheels will turn) by the horizontal input. this will then apply this to the front colliders.

```
private void HandleSteering()
{
    currentSteerAngle = maxSteerAngle * horizontalInput;
    frontLeftWheelCollider.steerAngle = currentSteerAngle;
    frontRightWheelCollider.steerAngle = currentSteerAngle;
}
```

"UpdateSingleWheel()" will visually change the position of a single wheel by using the GetWorldPose function.

```
    private void UpdateSingleWheel(WheelCollider wheelCollider, Transform wheelTransform)
    {
        Vector3 pos;
        Quaternion rot
; wheelCollider.GetWorldPose(out pos, out rot);
        wheelTransform.rotation = rot;
        wheelTransform.position = pos;
    }
```

7.2 Single Checkpoint System

The Single Checkpoint script is added to a 3D cube object that acts as a goal for the car when running under the training environment, the car will aim to make its box collider hit the cubes box collider. This script also references the multi checkpoint script which I will go into more detail about below.

"OnTriggerEnter()" will look be called when the cars box collider collides with the checkpoint box's box collider, once this happens it will send this data to the multi checkpoint object.

```
    private void OnTriggerEnter(Collider other)
    {
        if (other.TryGetComponent<Collider>(out Collider collider))
        {
            trackCheckpoint.PlayerThroughCheckpoint(this);
        }
    }
    1 reference
    public void SetTrackCheckpoints(TrackCheckpoint trackCheckpoint)
    {
        this.trackCheckpoint = trackCheckpoint;
    }
}
```

7.3 Multi Checkpoint System

"Awake()" is called when the program is run, it firstly creates a Transform (game object) that will find the game object in the project under the name "checkpoints", this game object contains all the checkpoint objects on the track. A new list is created to store all the single checkpoint objects, this will be used later in the script. It also sets each checkpoint for the AI to identify in the training environment using SetTrackCheckpoints().

```
private void Awake() {
    Transform checkpointsTransform = transform.Find("Checkpoints");

    checkpointSingleList = new List<CheckpointSingle>();
    foreach (Transform checkpointSingleTransform in checkpointsTransform) {
        CheckpointSingle checkpointSingle = checkpointSingleTransform.GetComponent<CheckpointSingle>();

        checkpointSingle.SetTrackCheckpoints(this);
        checkpointSingleList.Add(checkpointSingle);
    }
    nextCheckpointSingleIndex = 0;
}
```

“PlayerThroughCheckpoint()” takes in a single checkpoint object and if this checkpoint matches the order of the list created in awake() it will be identified as a correct checkpoint it will output “correct in the console and increment the checkpoint index by 1 and then get the remainder of the total checkpoints in order to allow the car to do more than one lap. If the player or AI misses a checkpoint it will be marked as incorrect, and the index will stay the same until the correct checkpoint is passed first.

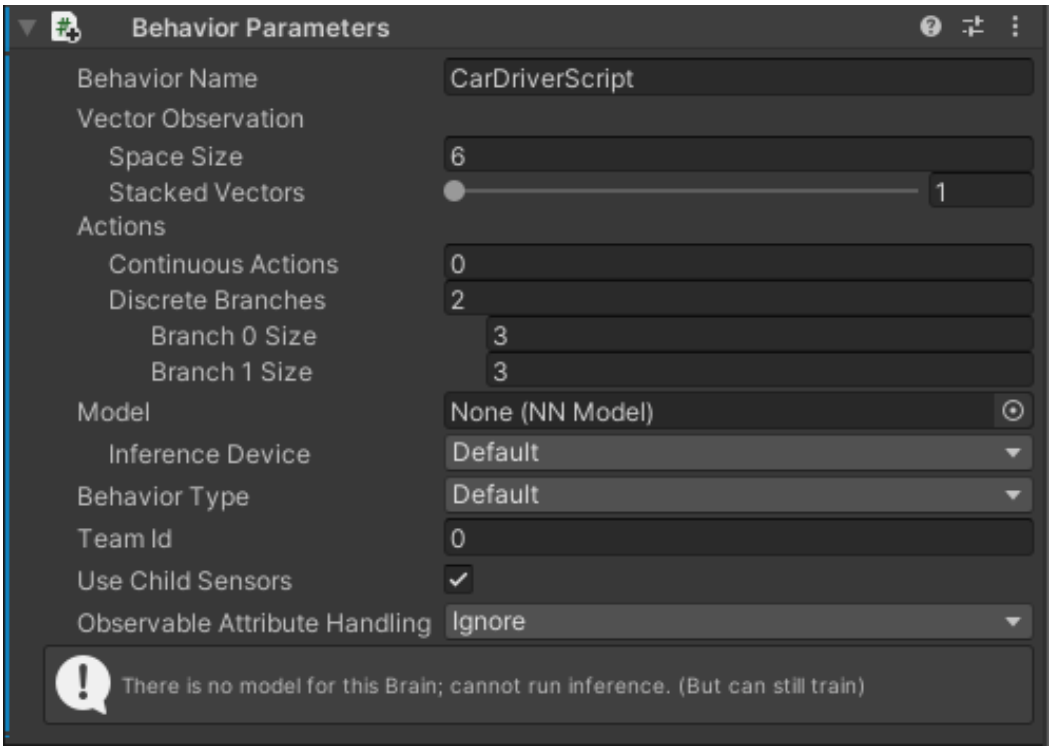
```
public void PlayerThroughCheckpoint(CheckpointSingle checkpointSingle) {
    if (checkpointSingleList.IndexOf(checkpointSingle) == nextCheckpointSingleIndex) {
        Debug.Log("Correct");
        nextCheckpointSingleIndex = (nextCheckpointSingleIndex + 1) % checkpointSingleList.Count;
        OnPlayerCorrectCheckpoint?.Invoke(this, EventArgs.Empty);
    }
    else {
        Debug.Log("Incorrect");
        OnPlayerWrongCheckpoint?.Invoke(this, EventArgs.Empty);
    }
}
```

8 ML-Agents

ML-Agents comes with premade scripts that will handle a lot of the machine learning training. By adding these scripts onto the car object, it will apply this too the car, the car can then be switched between training and inference.

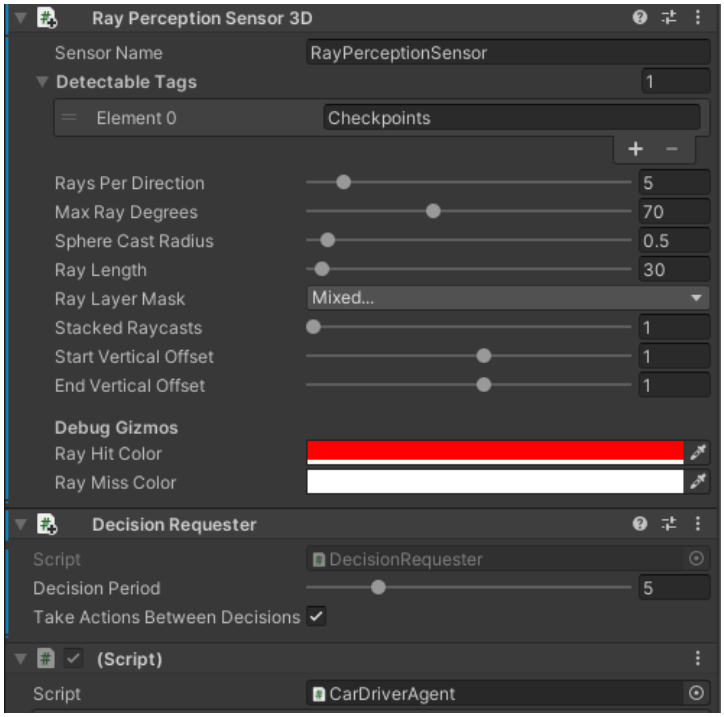
Behaviour Parameters is used to set up the training of the AI, vector observation will assign a vector of a set number of branches and sizes depending on the number of inputs. For example, in my case there is 2 inputs, forward/reverse and turning, these both have three possible values, 1, 0 and -1. As you can see below there are 2 branches for the 2 inputs and each branch is of size 3 for each value. These branches will be using during the decision-making process.

Model is where the neural network is stored after it is created when this is added, and the car is set to inferencing this will allow the agent to exit training and drive the car itself.



“Ray Perception Sensor 3D” is a useful script that will automatically cast raycasts that can be used to detect the checkpoints. Any game tag under the detectable tags will be picked up by these sensors, multiple tags can be added to this allowing the optional functionality in my project to detect walls set up as track limits, this will keep the car within track regulations. The sliders seen below are to change the position, length, and covered radius of the sensors. The sensors can also be visualized as shown in figure 8-1 below.

The CarDriverAgent will handle the inputs taken for driving and turning and will drive the car by itself. I will go into more detail about this script below. The script Decision Requester will decide what actions to take in CarDriverAgent.



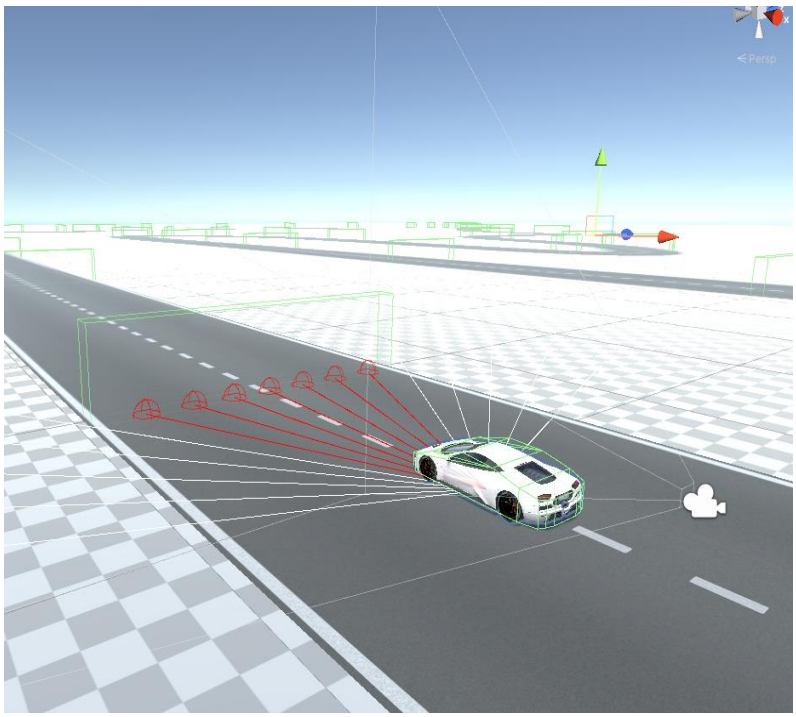


Figure 8-1 Raycast Sensors

9 Conclusion

To conclude, I have met most of the objectives I set out to complete at the start of this project, after some extensive research into machine learning I have learnt the fundamentals of this topic and would be more than confident in working on machine learning in future projects.

ML-Racing is now fully functional as a game, upon launching the program the player can take control of the car model using controls defined in my scripts. The user can drive the car around a set track created using a unity package called EasyRoads, this track includes checkpoints that are displayed in the console upon passing each one in order. If the player skips a checkpoint the console will display this also.

The car controller is bug free and seems to preform well even over long periods of time, although the car physics aren't an exact simulation, with the code I have compiled it gives the feeling of a real-life driving experience and for a first version of the game the mechanics hold up against some of its competition.

As for the machine learning aspect of the project the game currently does not have the required functionality to train AI although the scripts and assets needed are all available within the project. With some further work and more research on using the MLAgents functions and libraries to create the missing code needed I believe I will be able to implement this functionality.

Overall, the project has been a success, with most of my deliverables I set out in the begging being achieved. I have learnt a vast amount of knowledge on each topic in this project including game development, machine learning, game design and C#. I will be looking at game development as an option for a future career and at the very least I plan to expand on this project.

10 References

Machine Learning: What it is and why it matters | SAS UK. (n.d.). Retrieved April 21, 2022, from https://www.sas.com/en_ie/insights/analytics/machine-learning.html

What Is Game Development? (n.d.). Retrieved April 20, 2022, from https://www.freecodecamp.org/news/what-is-game-development/

What is Unity? – A Guide for One of the Top Game Engines – GameDev Academy. (n.d.). Retrieved April 21, 2022, from https://gamedevacademy.org/what-is-unity/