# REINFORCEMENTDRIVE

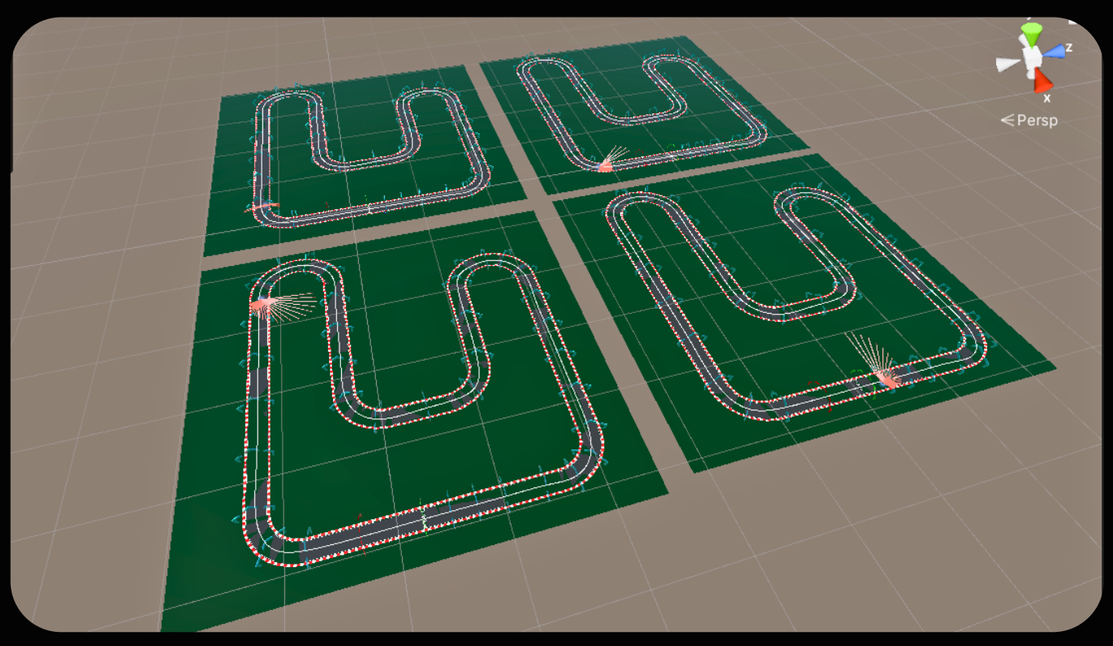
Daniel McMorrow | Software & Electronic Engineering | G00343022

Ollscoil Teicneolaíochta an Atlantaigh

Atlantic Technological University

## Introduction

This project is focused on creating a racing game with a unique twist, by integrating Unity Game Engine and ML-Agents plugin. It uses a Reinforcement learning method called Proximal Policy Optimization (PPO) to teach the AI how to drive a car in the game. PPO helps the AI learn from what it knows and discover new techniques. Players can race against the AI, which learns to go faster around the track while avoiding crashes with the player's car. As the AI uses PPO, it keeps getting better at racing, offering players an exciting and challenging experience.




## Training, Rewards & Object Detection

- Designed a reward system using a racepath, checkpoints, and invisible walls on the track.
- Checkpoints provided positive rewards, while collisions with invisible walls incurred negative rewards.
- Enhanced the agent's perception with raycasts, allowing it to detect checkpoints, walls, and track boundaries.
- Created multiple instances of the training track to accelerate the agent's learning process.
- Implemented a step timeout to penalize the agent for taking too long to reach the next checkpoint, resetting it to a random checkpoint if necessary.
- These improvements allowed the AI racer to learn from its actions and become a more skilled competitor in the racing game.

## Technologies Used

**Unity Engine:** A powerful game engine & development environment used to create games and applications.

**ML-Agents Plugin:** Enables machine learning to train intelligent agents within the game.

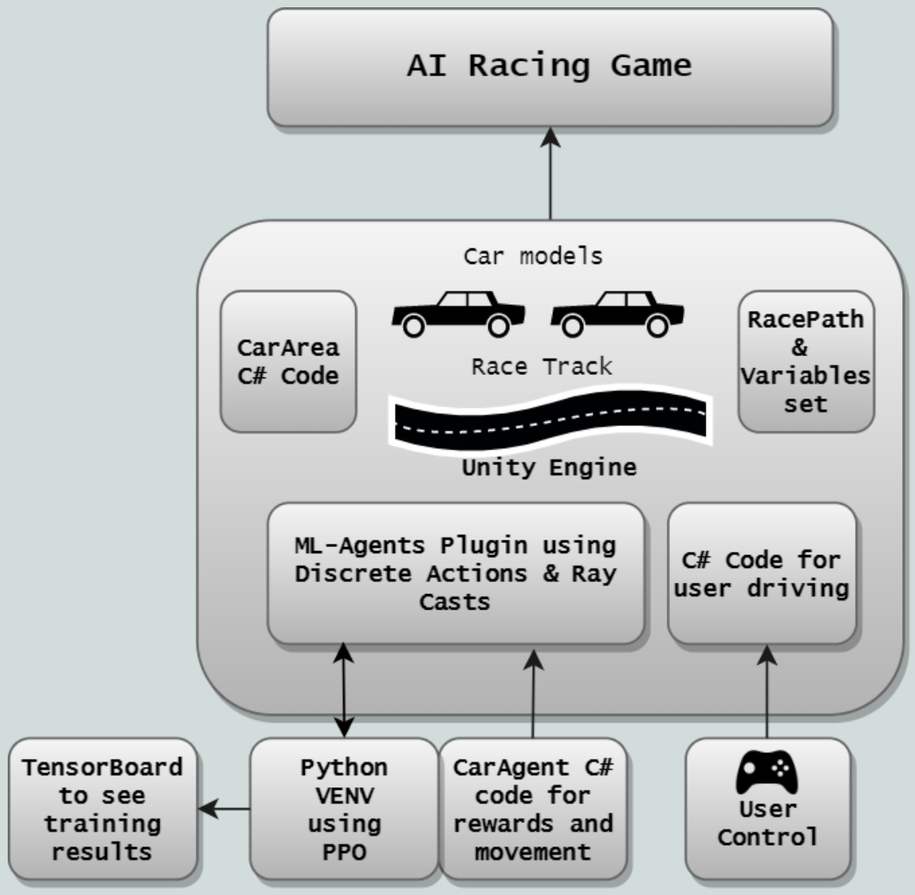
**Proximal Policy Optimization (PPO):** A reinforcement learning algorithm used to train the AI agent, balancing exploration and exploitation for efficient learning.

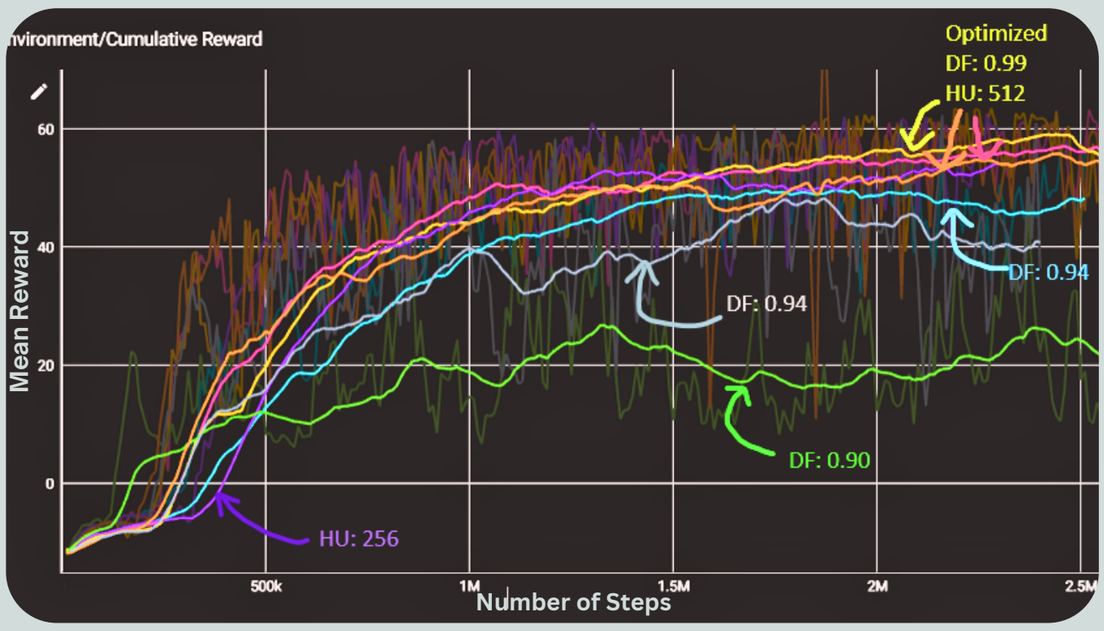
**TensorBoard:** A visualization tool used to monitor the training process of the AI agent.

**Python:** Used to configure the training process and define the learning algorithm.

**Visual Studio IDE & C#:** Used to write necessary C# code for game creation.

**Raycasts:** Provide the AI agent with information about its environment.




Graph of learning improvements with parameter changes

## Results

Initially, the AI agent struggled on the track, often making poor decisions and failing to follow the optimal racing line. However, by adjusting hyperparameters like hidden units and discount factor (gamma), the agent's learning improved. Balancing exploration (trying new actions) and exploitation (leveraging its current knowledge), the agent adapted its behavior and enhanced its racing skills.

These adjustments led to increased mean rewards per step, indicating that the agent was completing different tracks faster and with fewer mistakes. The progress highlights the reinforcement learning approach's effectiveness, with the learning curve eventually leveling out, as seen in the TensorBoard graph output.