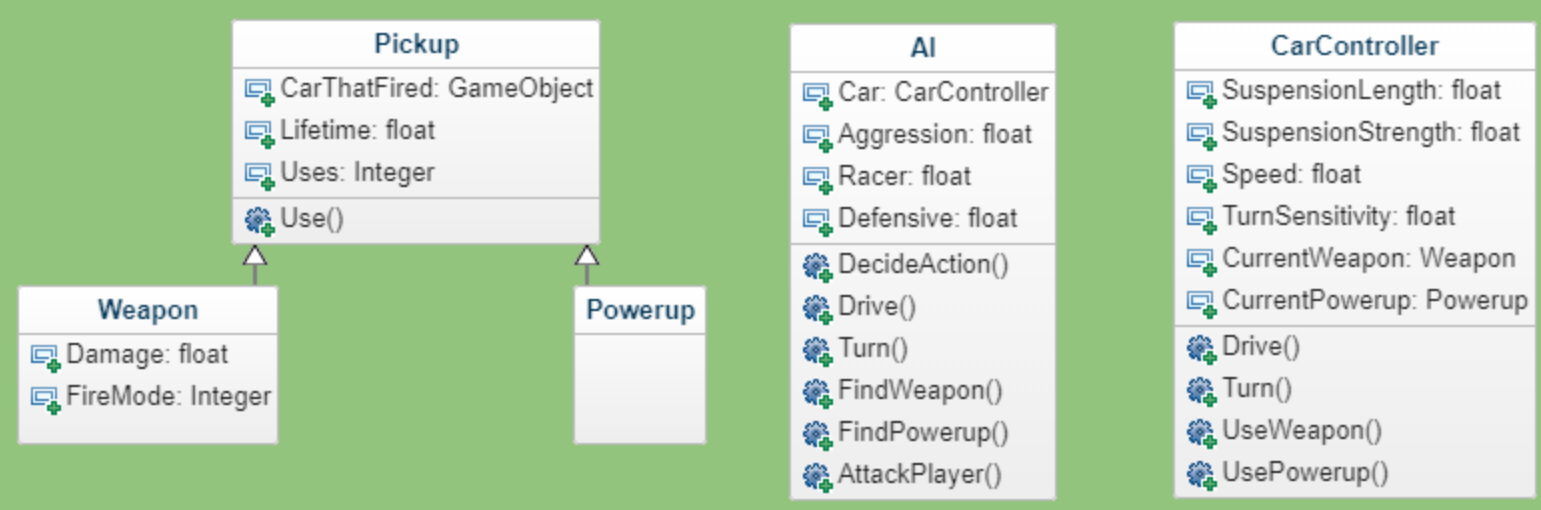# Karts & Krafts Racing AI

Jamie Covell

What is the problem?:
The problem that I am approaching is a battle racing AI for my team project. This AI will need to be able to find it's way round the track and interact with the players via steering the physics based car controller and using weapons and powerups around the map

Why does the problem need solving?:
The problem needs solving because the game is local multiplayer. If you are trying to play the game on your own then it would be very dull to play. So there will need to be AI in the game to keep it interesting for the player. The AI will need to behave in a way that is similar to that of a real player to make it feel like a challenge.

Pickup
- CarThatFired: GameObject
- Lifetime: float
- Uses: Integer
- Use()

Weapon
- Damage: float
- FireMode: Integer

Powerup

AI
- Car: CarController
- Aggression: float
- Racer: float
- Defensive: float
- DecideAction()
- Drive()
- Turn()
- FindWeapon()
- FindPowerup()
- AttackPlayer()

CarController
- SuspensionLength: float
- SuspensionStrength: float
- Speed: float
- TurnSensitivity: float
- CurrentWeapon: Weapon
- CurrentPowerup: Powerup
- Drive()
- Turn()
- UseWeapon()
- UsePowerup()

What is my solution?:
My initial solution to this problem was to create a GOAP system with STRIPS planning to create some advanced agents that could seemingly have different personalities to race and interact with the players in different ways.

GOAP: Goal Oriented Action Planning is a system in which the agent has a specific set of goals that are prioritised in the order that the agent tries to complete them. The agent also has a set of actions that it can do. The actions are comprised of a set of preconditions and postconditions. The agent can find a series of actions to take to complete its goal

STRIPS: Stanford Research Institute Problem Solver is a planning system that is used to find the goals of an agent given a domain and the world conditions.

I decided not to go with a GOAP system because it seemed to be an over complication for the problem i was trying to solve. I think I can achieve a similar result by using a simpler Utility-based AI system.

Utility-based AI: The Utility based AI will assign a cost/Weight to each action that it can take and will pick the option with the lowest cost/highest weight. Biases can also be added to each of these actions to force the agent to favour certain actions over others.

For my system I will be implementing 3 different biases. Aggression, Racer and Defensive.

The Aggression bias will favour actions that will go towards killing other players such as picking up weapons or shooting players.

The Racer bias will favour actions that go towards the agent completing the race such as steering in the correct direction and avoiding obstacles.

The Defensive bias will favour actions that go towards conserving the agents position and health, such as picking up powerups, avoiding combat and obstacles.

What went well?
I think the physics based steering for the cars has gone well. The cars can all steer towards the next checkpoint in the track. The cars do not avoid obstacles in front of them yet. I will need to implement a raycast or trigger in front of the car to check for obstacles and make the car reverse and turn if there is one.
I have also so far been unable to implement the Utility-based AI. This is mainly due to research on how to implement it but I will be adding this in soon to give the AI more complexity and personality