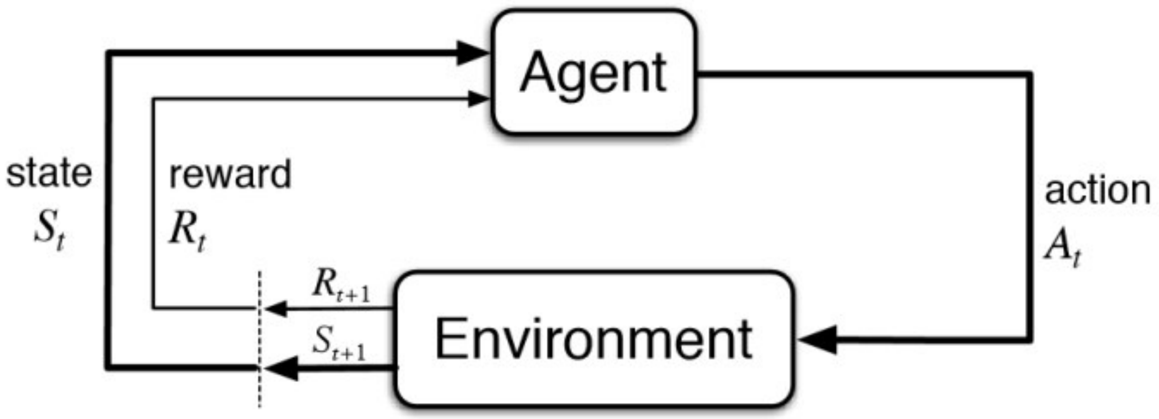
DOCUMENTATION

KAI – SELF DRIVING CAR AI

Introduction

This is a deep reinforcement learning project made using the Unity ML-Agents library. The main goal of the project is creating and training an intelligent agent that is capable of autonomously navigating the track using two actions: accelerating and steering.

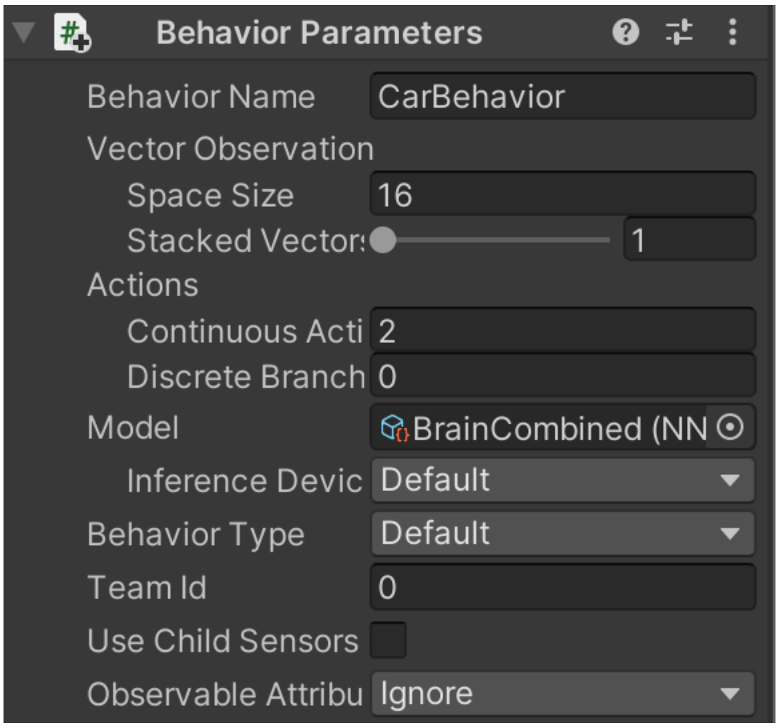


In essence, reinforcement learning consists of the following steps:

1. The agent performs an action
2. That action changes the state of the environment
3. The agent takes in the following:
    a. Observations about the environment after the action has been performed
    b. Reward signals that promote wanted behavior and penalties for bad behavior
4. The agents tries to draw conclusions from its inputs and adapt his behavior in order to maximize the reward it receives

The Unity ML-Agents library allows us to easily integrate deep reinforcement learning and 3D environments created in Unity, which is the reason why it was chosen for the project. The algorithm used for the learning process is called PPO (Proximal Policy Optimization).

Actions & Observations

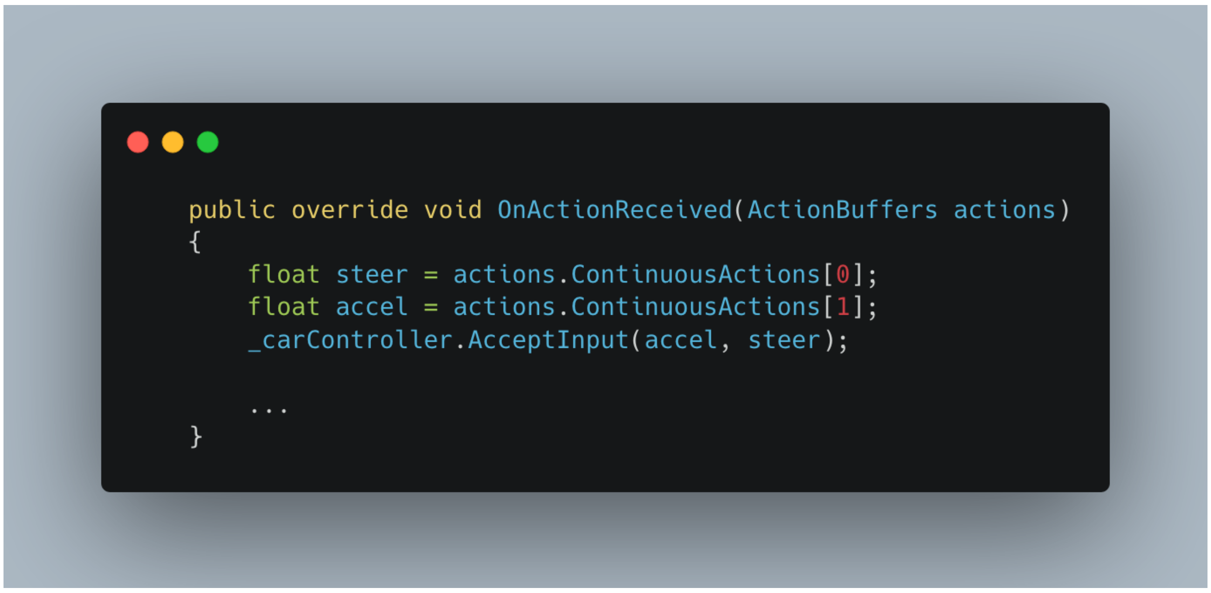


Behavior Parameters

Behavior Name	CarBehavior
Vector Observation	
Space Size	16
Stacked Vector:	1
Actions	
Continuous Acti	2
Discrete Branch	0
Model	BrainCombined (NN
Inference Devic	Default
Behavior Type	Default
Team Id	0
Use Child Sensors	
Observable Attribu	Ignore

This component can be found in the unity project editor. As can be seen from the picture above, this is where we define actions that the agent can take, as well as observations about its environment.

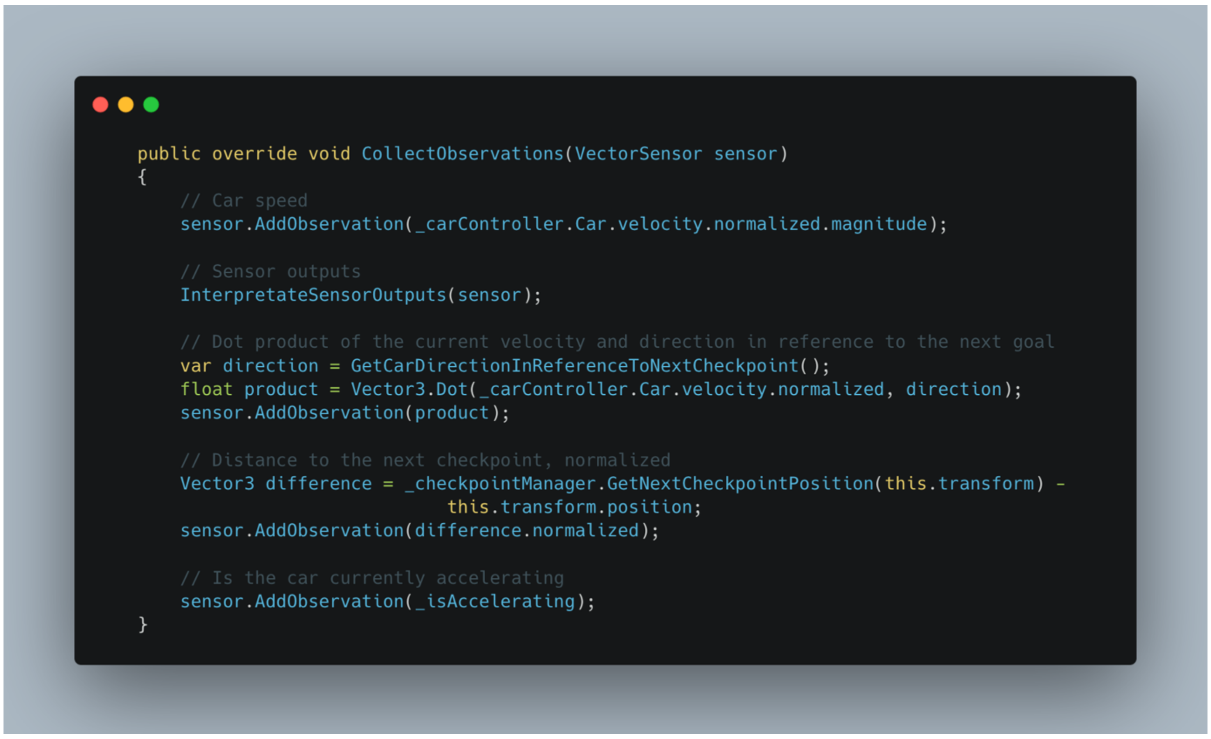
Actions are represented in a continuous space of size two – meaning that the agent can take two actions and the values of those actions are always between -1 and 1. In this case, these action represent accelerating (that can also be negative) and steering (negative values correspond to left steering). Actions that the agent has chosen can be read and applied wherever needed, as is shown in the code segment below.



```
public override void OnActionReceived(ActionBuffers actions)
{
    float steer = actions.ContinuousActions[0];
    float accel = actions.ContinuousActions[1];
    _carController.AcceptInput(accel, steer);

    ...
}
```

Observations represent a way for the agents to perceive its environment. In our project, the observation space is of size 16 and each entry is represented by a floating point number. The observations can be set by overriding the appropriate method, as is shown in the code segment below.



```
public override void CollectObservations(VectorSensor sensor)
{
    // Car speed
    sensor.AddObservation(_carController.Car.velocity.normalized.magnitude);

    // Sensor outputs
    InterpretateSensorOutputs(sensor);

    // Dot product of the current velocity and direction in reference to the next goal
    var direction = GetCarDirectionInReferenceToNextCheckpoint();
    float product = Vector3.Dot(_carController.Car.velocity.normalized, direction);
    sensor.AddObservation(product);

    // Distance to the next checkpoint, normalized
    Vector3 difference = _checkpointManager.GetNextCheckpointPosition(this.transform) -
                         this.transform.position;
    sensor.AddObservation(difference.normalized);

    // Is the car currently accelerating
    sensor.AddObservation(_isAccelerating);
}
```

The two methods shown in the two code segments above are the core methods for reinforcement learning when using the Unity ML-Agents library. This project can be used as a base and be easily extended by new functionality, using the already written code as a reference. For more information about the library and its usage, use the following url: https://github.com/Unity-Technologies/ml-agents