Spike Outcome Report

Number: 07

Spike Title: Emergent Group Behaviour

Personal: DANIEL NEWMAN, 6449921

Link to repository:

Goals:

The goal of this spike is to demonstrate how steering behaviours can be combined to create realistic grouping behaviours utilising autonomous vehicles.

Technologies, Tools, and Resources used:

Sublime Text 3 or equivalent Python supported IDE

Python 3.6.4

Tasks undertaken:

First we need to identify how each vehicle will determine its neighbours within a surrounding radius.

This can be done with a simple boolean value for all vehicles within the neighbours radius, this only needs to be done on an attribute change, such as enabling or disabling steering behaviours

```
#Steering behaviours
def tag_neighbours(self, radius):
    self.untag()
    for otherAgents in self.world.agents:
        to = self.pos - otherAgents.pos
        gap = radius + otherAgents.tag_radius
        if Vector2D.length_sq(to) < gap**2:
            otherAgents.tagged = True;
```

Then we define each of the steering behaviours we are going to use. Each steering behaviour loops through all the neighbours that have been tagged and applies a force that is proportional to all other neighbours.

```
def seperation(self):
    steering_force = Vector2D()
    for agent in self.world.agents:
        if(agent is not self and agent.tagged):
            toBot = self.pos - agent.pos
            steering_force += Vector2D.normalise(toBot)/Vector2D.length(toBot)
    return steering_force

def cohesion(self):
    centre_mass = Vector2D()
    steering_force = Vector2D()
    neighbour_count = 0
    for agent in self.world.agents:
        if agent is not self and agent.tagged:
            centre_mass += agent.pos
            neighbour_count += 1

    if neighbour_count > 0:
        centre_mass /= neighbour_count
        steering_force = self.seek(centre_mass)
    return steering_force

def alignment(self):
    avg_heading = Vector2D()
    neighbour_count = 0
    for agent in self.world.agents:
        if agent is not self and agent.tagged:
            avg_heading += agent.heading
            neighbour_count += 1
    if neighbour_count > 0:
        avg_heading /= neighbour_count
        avg_heading -= self.heading
    return avg_heading
```

Separation is designed so that it keeps a certain distance from its neighbours, this is done to try and prevent overlapping vehicles. However without a hard limit it is impossible to enforce.

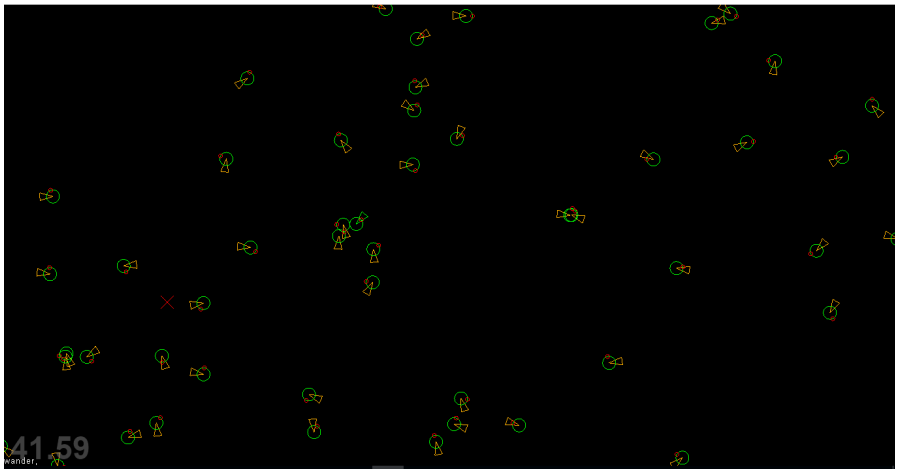
Cohesion is designed to put the vehicle at the centre of the group of neighbours, this helps keep the group together and vehicles on the outskirts of the radius from travelling away from the grouping

Alignment is designed to keep the vehicles in the group facing in the same direction. This is done by getting the average heading of all the neighbours and applying it to the vehicles

Now that we have created the steering behaviours we intend to use, we have to weight them so that one behaviour doesn't dominate over the other behaviours, which would lead to erratic behaviour. This is called the weighted truncated sum, and it is used to provide the steering force to the vehicle. The reason it is truncated is to prevent an extremely large force from being generated due to the grouping behaviours.

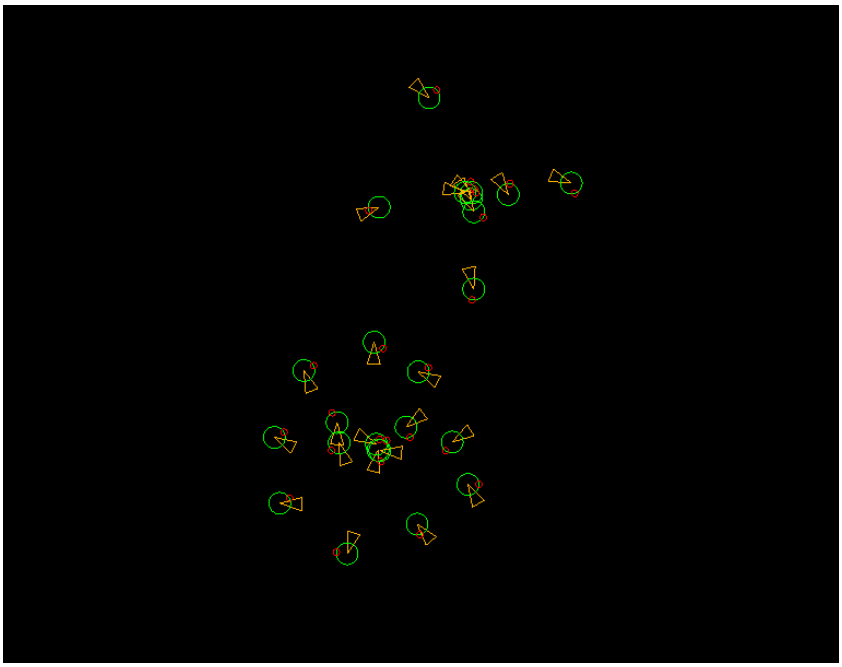
```
def calculate(self, delta):
    force = Vector2D()
    # calculate the current steering force
    force += self.wander(delta) * 0.5
    if(self.cohesion_toggle):
        force += self.cohesion() * GROUPING_VALUES['cohesion']
    if self.seperation_toggle:
        force += self.seperation() * GROUPING_VALUES['seperation']
    if self.alignment_toggle:
        force += self.alignment() * GROUPING_VALUES['alignment']

    self.force = force
    return force
```

What we found out:



No Grouping behaviours

Without grouping behaviours our vehicles behave as separate entities who will not group together of their own design



With Grouping Behaviours

Whereas with grouping behaviours employed we can simulate behaviours that effectively looks like the vehicles are designed to work together whilst maintaining their ability to remain autonomous