Spike: Emerge Group Behaviour
Title: Emerge Group Behaviour

Author: Tran Duc Anh Dang | 103995439

Goals / deliverables:

Create a group agent steering behaviour simulation that is able to demonstrate distinct modes of emergent group behaviour. In particular, the simulation must:

- Include cohesion, separation and alignment steering behaviours
- Include basic wandering behaviours
- Use a weighted-sum to combine all steering behaviours
- Support the adjustment of parameters for each steering force while running

Technologies, Tools, and Resources used:

List of information needed by someone trying to reproduce this work

- Python 3+
- Built in Python libraries.
- IDE or Code Editor (Visual Studio Code)

Tasks undertaken:

- Install Python: Download and Install Python 3+ via https://www.python.org/downloads/
- Set up a code editor or IDE: Download and install a python compatible ide or code editor such as Visual Studio Code, PyCharm
- Open and familiarize with the code by reading through, paying attention to the comments that had been made.
- Run the code: Execute the code and observing the output.

What we found out:

agent.py

```
76      def calculate(self, delta, close_neighbors):
77          if self.mode == 'separation':
78              combined_force = self.separation(close_neighbors) * self.separation_weight
79          else:
80              wander_force = self.wander(delta)
81              cohesion_force = self.cohesion(close_neighbors)
82              alignment_force = self.alignment(close_neighbors)
83
84              combined_force = (wander_force * self.wander_weight +
85                                cohesion_force * self.cohesion_weight +
86                                alignment_force * self.alignment_weight)
87
88          return combined_force
```

Snipped

In the calculate function, it determines the force that will be applied to the agent based on the mode. Especially, Separation mode will help the agents to de-attach from each other. Otherwise, in wander will be

a combination force of wander force, cohesion force and alignment force to help them combine and travel together in the same direction.

agent.py

```
182  def cohesion(self, close_neighbors):
183          center_of_mass = Vector2D()
184          count = 0
185
186          for agent in close_neighbors:
187              center_of_mass += agent.pos
188              count += 1
189
190          if count > 0:
191              center_of_mass /= count
192              return center_of_mass
193          else:
194              return Vector2D()
195
196      def separation(self, close_neighbors):
197          if not close_neighbors:
198              return Vector2D()
199
200          # Assuming you have a function to find the closest agent
201          closest_agent = self.closest(close_neighbors)
202          closest_agent_pos = closest_agent.pos
203          target = (2 * self.pos - closest_agent_pos)
204          to_target = target - self.pos
205
206          return to_target
207
208      def alignment(self, close_neighbors):
209          average_heading = Vector2D()
210          count = 0
211
212          for agent in close_neighbors:
213              average_heading += agent.heading
214              count += 1
215
216          if count > 0:
217              average_heading /= count
218              average_heading -= self.heading
219              return average_heading
220          else:
221              return Vector2D()
```

Snipped

Cohesion function will calculate the centre of mass of the neighbouring agents and returns the direction towards the centre of mass. This will allow the agent to move in groups and stick together.

Separation will find the closes agent among the neighbours and calculates a possible force required to pushes agent away from the closest neighbour. This behaviour helps the agent to avoid each other and maintain a certain distance.

Alignment function will calculate the average heading of the neighbouring agents and returns the difference between the average heading and agent's heading. This will align the direction of the agent with its neighbour.

Wander function is the function that been given out in the last few task and this function creates random behaviour, it first adds small random vector to the agent's current target position, then projects this new vector back onto a unit circle and adjusts the length of the vector to the same radius wander circle. Finally moves the target into a position ahead of the agent and seeks towards new target.

Overall, these behaviours have met the requirement of this task, which allow the agent to interact with its environment and other agents in different way.