Exercise – Flocking – Part 1

In this exercise we'll be implementing 3 new steering behaviours as IBehaviour objects. The pseudo code supplied here includes weighting of the returned forces. This is not necessary, but creates a very typical flocking behaviour and bypasses the need to implement a *Weighted Truncated Running Sum with Priority* system. Feel free to adjust this to your liking.

Exercise 1:

Based on the lecture notes, Implement a Separation behaviour that implements the IBehaviour class/interface.

```
//var for aggregating forces
separate = zero

foreach(agent b in neighbourhood)
{
    push = agent.position – b.position
    push *= ? //1 if close, 0 if far
    separate += push
}

//get average separation force
separate /= neighbourhood.size

forceToApply = separate – agent.velocity
//etc...
```

Exercise 2:

Based on the lecture notes, Implement an Alignment behaviour that implements the IBehaviour class/interface.

```
//var for aggregating forces
align = zero

foreach(agent b in neighbourhood)
{
    head = b.velocity
    head *= ? //1 if close, 0 if far
    align += head
}

//get average separation force
align /= neighbourhood.size

forceToApply = align – agent.velocity
//etc...
```

Exercise 3:

Based on the lecture notes, Implement a Cohesion behaviour that implements the IBehaviour class/interface.

```
//var for aggregating forces
cohere = zero

foreach(agent b in neighbourhood)
{
    pull = b.position – agent.position
    pull *= ? //0 if close, 1 if far
    cohere += pull
}

//get average separation force
cohere /= neighbourhood.size

forceToApply = cohere – agent.velocity
//etc...
```

Exercise 4:

Let's create a Boid object that implements/inherits the Agent class so we can see these at work.

- Give the Boid class a 'neighbourhoodRadius' value and the ability to know about other Boid objects within that area around itself.
    - Hacky method for testing purposes: a static collection of Boids within the Boid class which you push to and pop from via the Boid's constructor/deconstructor. Then simply loop through and do a distance check on all Boids to find neighbours.
- Create a small swarm of boids and ensure they behave in an appropriate way.
- Try adding some wander force to the boids to make them change direction a little more.

Challenge Exercise:

1) Implement a BoidLeader who exerts more influence over other nearby boids. Make the BoidLeader controllable via keyboard input and enjoy steering your swarm around!
2) Add collision avoidance to the Boid object so they don't move though physical objects in their environment.

References:

- A great tutorial for revision on flocking can be found here.
- The original paper on flocking by Craig Reynolds can be found here.