2. Flocking

The 4 rules of flocking are as follows:

2.1 Alignment

Steering towards direction of local flockmates. With this rule of thumb in mind, we need to position of local flockmates. Once we retrieved the position, we can have an if statement to see if a Boids position is within a defined range of neighbor Boids. Once we achieve the implementation of this if statement, then we can define our alignment force to be later applied to our Boids rigid body. This will ensure the Boids are steered towards their local flockmates.

2.2 Cohesion

Steering towards average position of local flockmates. With this rule of thumb in mind, again, we need the position of local flockmates. Luckily, we already have an if statement that checks whether a Boid is within a defined range of neighbor Boids. So, to reduce ambiguity, we can use this if statement and define or cohesion force in here. Note that cohesion needs the average position of the local flockmates which requires more calculations than alignment. For this reason, we divided the neighbors size to get the average position of the neighbors and applied the final cohesion force to our Boids rigid body.

2.3 Separation

Steering to avoid crowding local flockmates. With this rule of thumb in mind, we also need the position of the local flockmates. For this reason, we will define the separation logic under the same if statement we had for alignment and cohesion. Only, for separation, we need to steer away from local flockmates. For this reason, we retrieve the position difference between a Boid and its surrounding neighbors. Later, we use this position difference as our steering point to steer away from the local flockmates.

All the forces above are later applied to Boid rigid body and achieves the result of flocking.

2.4 Collision Avoidance

This is achieved with the same logic as separation rule above since we need to avoid colliding with the obstacles in the scene by moving the Boids away from the obstacles. An additional if statement has been created for the obstacles because we needed to iterate the obstacles positions that were passed from Generate Obstacle function above. Once the obstacle positions were retrieves, I used this to steer the Boids away from them to avoid possible collisions.