Exercise – Flocking part 2

In this exercise we’ll be implementing *Formations* and *Reciprocal Collision Avoidance.*

Exercise 1 - Formations:

Based on the lecture notes, Implement a FormationAgent that derives from Agent. The FormationAgent will update the Agent’s target position and then allow the base Agent to run IBehaviors that move it towards that location. Feel free to research alternative formation solutions.

FormationAgent: Agent

{

squadMates = null

leader = null

isLeader = false

offset = Vector2(0,0)

Initialise(squadMates, isLeader)

{

this.squadMates = squadMates

this.isLeader = isLeader

if (!this.isLeader)

{

this.leader = // find leader agent from squadMates

// provided squad members are initialised // in formation

offset = this.leader.position - base.position

}

}

override Update()

{

if (this.isLeader)

{

//normal update using base.targetPosition or user input, etc

base.Update()

}

else

{

//transform offset by leader's current transform

base.targetPosition = leader.transform \* offset

base.Update()

}

}

}

Exercise 2 – Crowd Simulation:

Create a CrowdAgent that derives from Agent. The CrowdAgent will override the Agent’s Update method so that it may constrain the final velocity before it is applied to the Agent’s position.

CrowdAgent: Agent

{

override Update()

{

steeringForce = //business as usual, recommend use of virtual methods

base.velocity += steeringForce \* deltaTime

base.velocity = ComputeNewVelocity(base.velocity)

base.position += base.velocity \* deltaTime

}

}

Next, download the RVO2 library from: <http://gamma.cs.unc.edu/RVO2/downloads/> and integrate it into your project so that your CrowdAgent can make use of it. The main challenge here will be restructuring the library code to suit your project. Alternatively your teacher may give you the source code to the Crowd Demo to reference (written in Unity/C#).