Starling Flight and Flocking

Sushant Rathi, Shashwat Shivam

May 3, 2018

Abstract

This paper deals with observing and trying to simulate the patterns and behaviour of Starling bird flocks flight using mathematical properties.

Contents

1 Starling rules

1.1 Separation

A crucial behaviour for a simulation is to prevent objects intersecting. This is achieved with a repulsive force between any pair of birds which are too close to each other. Currently we have taken a weighted average of all these forces to calculate net change in velocity.

1.2 Cohesion

Here each bird considers its nearby neighbours, but instead of trying to move away from any boid which is too close, this rule calculates the average position as an influence for boids to move towards.

1.3 Alignment

Similar to the implementation of the cohesion rule, birds inspect their radius of neighbours, and try to align themselves with the average of their neighbours velocity. Without this rule, the result can allow for groups of birds to fly at cross-purposes through each others paths, and create unrealistic simulations.

1.4 Target following

One feature that we observed in starling flights is that they regularly group together at some spot (that seems random to the observer) and disperse. We have simulated this by randomly spawning target locations for each bird (that are close to each other) and dispersing them after some time (by adjusting weights for other rules).

2 Efficiency

One area where we realised we could improve our algorithm was where we were finding out the neighbouring birds. We have set a radius, which can be considered as a sphere of influence, and only birds inside this sphere influence the accelerations determined by each rule. The naive approach to this is an O(n2) algorithm which iterates over every bird and calculates distance to determine whether it is a neighbour or not. Here are some improvements we came up with-

2.1 Parallelism

We ran the method to update neighbours on a separate core (in an infinite loop), this led to a significant reduction in lag for large number of birds as this part does not stall with updation of positions and rendering.

2.2 Improved data structure

We experimented with using a better data structure for storing the positions- we first considered a grid based approach however found it to be memory intensive, as each block of grid whether occupied or not will take up space. We finally settled on octree, as it offers an excellent alternative as the granularity of a region depends on number of particles in it, and we can get very fine division of space in without taking up too much memory.