# Flocking Simulation

Ethan Lao

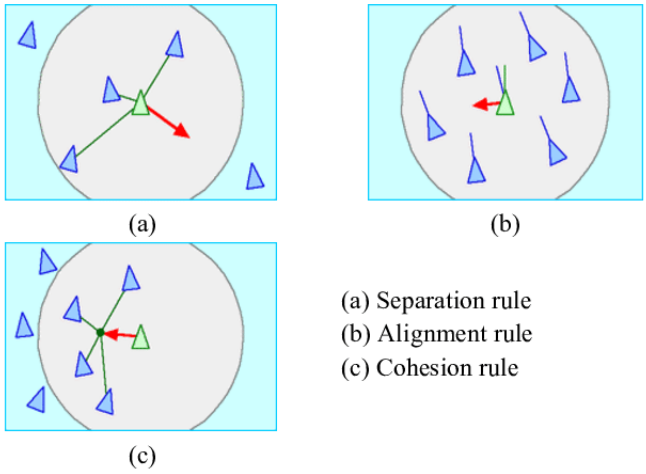
May 2022

## 1 Algorithm

### 1.1 Standard Boids

In a flocking simulation, we simulate how a flock of objects, like birds, could move in a natural way. The standard technique to do this is the Boids algorithm, where each boid (bird) moves to the next step according to the following 3 parameters:

- Separation: steer to avoid crowding nearby boids
- Alignment: steer towards the average velocity or direction of nearby boids
- Cohesion: steer to move towards the average position of nearby boids




At each timestep, we calculate the average velocity and position of nearby boids to obtain vectors for alignment and cohesion. In addition, we average the directions of nearby boids towards this boid to obtain the separation vector. Finally, all 3 of these vectors are passed into a weighted sum to get the force and direction applied towards this boid.

Deciding what is considered nearby is another parameter, a boid's visual range. This affects both alignment and cohesion, where the average velocity and position is taken only only for the boids within this boid's visual range. To note, this does not apply to separation since a boid would not necessarily want to separate from every boid it sees, but rather only boids within a constant radius (smaller than the visual range) of this boid.

### 1.2 Extensions

In addition to these 3 parameters, I also added 3 more for standard boids:

- Momentum: steer towards the boid's previous heading direction; this smooths out the path of a boid, and allows a boid to keep moving when there are no boids in it's visual range.
- Light Attraction: steer towards the light in the scene

- Fear: steer away from predators (see the Predators section below)

The visual range applies to both the light attraction and fear. These 3 parameters are integrated in the weighted sum.

### 1.3 Predators

Predators are a variant of the standard boid, but they don't necessarily obey all the standard rules. The only parameters that apply to them are separation (from other boids) and momentum. A predator has an additional factor of steering towards the closest non-predator boid within it's visual range.

## 2 Implementation

This entire project is coded in JavaScript/TypeScript and displayed through a React app. The boid graphics are implemented using D3.js, a graphing library to manipulate svgs directly.

### 2.1 Code Structure

The bulk of the code is separated into 2 main files, `FlockingGui.tsx` and `FlockingSim.tsx`.

`FlockingGui.tsx` sets up all the parameters the user can specify, as well as drawing the actual simulation. In addition, there are 3 buttons. One plays/pauses the simulation, one resets the simulation while keeping all parameters, and the last resets the simulation and restores all parameters to their default values.

The simulation is done in 3d, but the graphics are in 2d. The position of each boid is projected onto an x-y plane. To represent the z-axis, each boid changes size according to how far it is from the "camera" by changing the radius of the circle representing the boid. I used D3.js to create an svg, and used circles and lines to represent each boid (the circle for it's position, and lines for it's direction). D3.js provides the feature of transitioning objects, where the position of a boid can be interpolated linearly over a certain time. This smooths out the animation and allows us to use a larger timestep for the actual simulation.

`FlockingSim.tsx` contains the code that actually determines the current positions and directions of all boids at each step. There are 3 classes in the file. The first is a custom `vec3` class that is used to represent a position and direction. There is a `Bird` class which contains a boids location, velocity, group number (see Groups below), and whether it is a predator or whether it is dead. Finally, the `FlockingSim` class contains all information about all boids in the scene. All boids are initialized with a random position and location within the scene, as well as a random group number.

The majority of the boids algorithm implementation is handled in the `getNextStep()` function in the `FlockingSim` class. It updates the position and direction of all boids in the scene, then returns the updated information.

### 2.2 Groups

Groups can be thought of to represent a species of bird, and each boid is colored according to it's group number. I added this to simulate how multiple flocks could interact with each other. The number of groups is determined by the user, and members of each group only flock with others in the same group. However, the rules of separation still apply between all boids.

Each group is also given a light, or a destination point, to help keep the flocks moving and interacting with each other. These destination points are randomly generated. Once a flock's average position is within a constant distance to the light, the light gets moved to a new position. This feature is disabled when there is only one group. As a note, predators are represented in red and don't count towards the group number.

### 2.3 Other Comments

#### 2.3.1 Boundaries

Since we want the birds to stay in view, we create a constant-sized box for the boids to stay within. A boid bounces off a "wall" of the box according to an inverse function when the boid is within a certain margin of the box: $\frac{-\text{margin}*1000}{\text{dist}-\text{margin}} - 1000$. If we find that a boid is outside of the box, an "infinite" force is applied in the direction of the axis violated (but will ultimately be limited by the maximum speed).

#### 2.3.2 Speed Limit

Before a boid's final velocity is determined, I cap it with a speed limit constant to prevent the simulation from blowing up. But for predators, this speed limit is determined by user input.

#### 2.3.3 Predator Interaction

A predator is always designated in red. It has it's own visual range parameter, as well as it's own maximum speed. When a predator is within a constant radius of a standard boid, the boid dies. This is represented by freezing the boid and fading it's opacity. When there are no boids within the predator's visual range, the predator will simply coast in the same direction and bounce off walls because of it's momentum parameter.

#### 2.3.4 Hover Light

When the user's mouse hovers over the simulation, a light is moved to the position of the mouse. Since there is a not a way for the user to specify the position on the z-axis, I simply place this light in the middle (depth-wise) of the box. Birds will flock to this light like all other lights in the scene, and the tendency to do so is weighted by the same light attraction parameter determined by the user.

## 3 References


- https://en.wikipedia.org/wiki/Boids
- https://eater.net/boids
- https://medium.com/fragmentblog/simulating-flocking-with-the-boids-algorithm-92aef51b9e00