# Project 1 IT3708

Flocking and Avoidance With Boids

Iver Jordal

## Implementation

The simulation is programmed in Python, with pygame for the user interface. I've used the concept of Object Oriented Programming. The main class is **Flock**. This is the class that initializes the graphics (**Gfx**), a number of boids and a number of predators. When that is done, it starts running the update and render loop. For each iteration in the update loop, update is called on each **Boid** and **Predator** instance. Then each Boid and Predator instance calculates and applies all forces that should act on itself. Boid, Predator and **Obstacle** are classes with *x* and *y* attributes, according to where they should be drawn in the UI. Boid and Predator also have *dx* and *dy* for describing their velocity. Boid, Predator and Obstacle have draw methods that take in the drawing library and the screen object. In each draw iteration draw() is called on all those instances. The Gfx instance takes care of initializing the screen and registering keyboard and mouse interaction. Whenever a keyboard or mouse event occurs, Gfx notifies its event listeners. The Boid class listens to keypresses that should change boid parameters (boid separation, boid alignment, boid cohesion, predator separation, obstacle avoidance) and Predator listens to predator-related keypresses. Obstacle listens to mouse interactions and adds/removes obstacles accordingly.

### How forces are calculated

#### Separation

The separation force pushes a boid away from another boid. The scalar of the force is $\frac{1}{1+d}$ where $d$ is the euclidean distance $\sqrt{(x_1 - x_2)^2 + (y_1 + y_2)^2}$ between the two boids. This means that when two boids are really close, the force becomes really strong. When they are far away from each other, the separation force is not so strong. This is inspired by how magnets repel each other.

#### Alignment

This force is obtained by averaging the velocities of other boids within the neighbour distance threshold. Basically, just sum the velocity vectors for the neighbour boids and divide by the number of neighbour boids. Finally, there's a normalization step that divides the force by its vector length. Thus the alignment force becomes a unit vector with a direction that is the average direction of the neighbour boids' movement.

#### Cohesion

The cohesion force is the force that pulls a boid towards the average position of neighbour boids. The average position is obstained by summing the positions of the neighbour boids and then dividing by the number of neighbour boids. The cohesion force is then found by subtracting own boid position from the average position of neighbour boids.

#### Obstacle avoidance

Obstacle avoidance is a combination of two forces: The first force is simply a separation force as explained before. The second force is a steering force that is specific to obstacle avoidance. It is only applied when the obstacle is in front of the boid, as illustrated by the large semicircle in Figure 1:

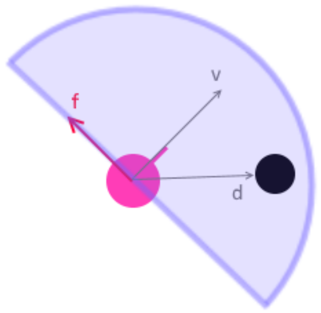

*Figure 1: A boid's obstacle avoidance force*

In Figure 1, $v$ is the velocity vector, $d$ is the unit vector that points towards the obstacle and $f$ is the resulting force. $f$ is always normal to $d$. In other words, it is $v \pm \frac{\pi}{2}$. Plus or minus is chosen based on the angle difference between $v$ and $d$. Basically, the concept is to steer left when an obstacle is detected to the right, and vice versa. The scalar of $f$ is computed in the same way as for the separation force, i.e. $\frac{1}{1+d}$

### Fleeing from predators

The force that makes a boid flee from predators is just a separation force, as explained before, but with a larger neighbour radius and a larger scalar.

### Speed regulation

Boids and predators have a default speed which they always strive to achieve. Whenever a boid or a predator has a speed that deviates from the default speed, it either brakes or accelerates. This is done by calculating the difference between the actual speed and the default (desired) speed and setting the current speed to $speed_{current} - 0.7 * \Delta speed$, so that the current speed approaches the default speed. In a way, the speed is damped and becomes stable when the current speed equals the desired speed.

## Description of emergent behavior

**Scenario 1** (low separation, low alignment, high cohesion): Boids fly close to each other. The alignment is not very stable.

**Scenario 2** (low separation, high alignment, low cohesion): The alignment is very stable. The space between boids is normal.

**Scenario 3** (high separation, low alignment, low cohesion): There's a lot of space between each boid. Alignment is in flux.

**Scenario 4** (low separation, high alignment, high cohesion): The formation is compact, as in scenario 1, but the alignment is much more stable.

**Scenario 5** (high separation, low alignment, high cohesion): Normal spacing between boids. Alignment is not so stable.

**Scenario 6** (high separation, high alignment, low cohesion): Looks pretty much like scenario 3, as in lots of space between boids, but the alignment is much more stable.