FRIteza

Predator-Prey Simulation Using Boids Model

Matija Ojo, Miha Krajnc, Marko Adžaga, and Janez Kuhar

Collective behaviour course research seminar report

November 12, 2023

Iztok Lebar Bajec | Associate Professor | mentor

ToDo: Add abstract.

Collective Behaviour | Boids | Simulation | Prey-Predator | Escape patterns

Introduction

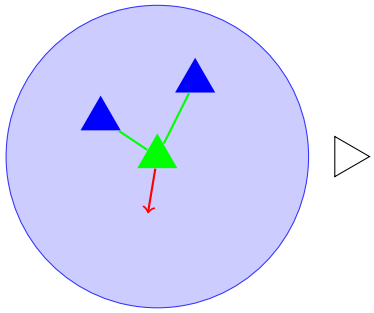
In 1987, Reynolds [2] proposed a simple algorithm to model the flocking behavior of birds, herding of sheep, and similar phenomena, known as the Boids (Bird-oid objects) model. In contrast to controlling the interactions of the entire flock, the Boids simulation focuses on dictating the behavior of each individual boid. Despite consisting of a few simple rules, this algorithm produces complex and lifelike behaviors similar to those observed in nature.

Our paper centers on the implementation of a predator-prey behavior utilizing a Boids simulation.

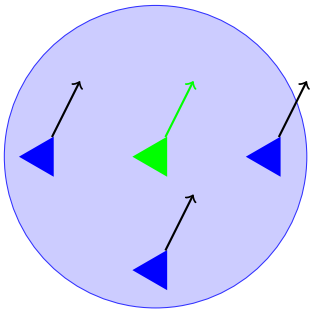
Methods

As a starting point, a basic boids model has been implemented, which is based on three simple rules:

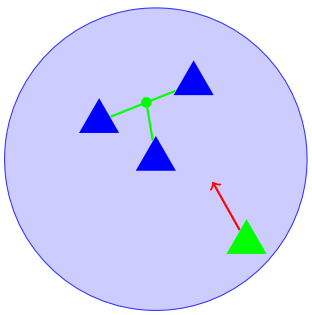
1. Avoid collisions.



2. Maintain the same heading and speed as the neighboring boids.



3. Gravitate toward the center of the flock.



Procedural generation of a tropic island and coral reef

In computer graphics there is frequent need for displaying large vistas of natural looking terrain. Designing such terrain by hand is typically time consuming. With procedural generation, on the other hand, larger areas of natural looking terrain can be generated with or without minimal intervention in a relatively short time. In this work we present a process of procedural generation of a tropical island with the associated corral reef. We start by generating a heightmap for the base terrain. The heightmap is then transformed by simulating the processes of hydraulic and thermal erosion to achieve a more natural look of the terrain. As coral reefs often grow around tropical islands, we also simulate their growth as part of the last step. Real-time visualization is enabled during the simulation, so that one can observe the evolution of the terrain. Here we dynamically apply textures to the terrain based on its local characteristics. The result is a natural looking model of the textured tropical island and corral reef.

Procedural generation | Terrain generation | Thermal and hydraulic erosion | Coral reef | Simulation | GPU

Basic boids implementation. Each boid B has the following properties:

1. position - a vector in $\mathbb{R}^2$, denoted by $position(B)$,
2. velocity - a vector in $\mathbb{R}^2$, denoted by $velocity(B)$,
3. acceleration - a vector in $\mathbb{R}^2$; used exclusively for the internal computation of the boid's velocity and not for behavioral logic.

With regards to behavioral logic, we also assign the following attributes to all boids:

1. perception radius (denoted by r_P)
2. separation radius (denoted by r_S, also note that $r_S < r_P$)

The *Euclidean distance* (1) is used for computing the distance between boids.

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2};\ p,q \in \mathbb{R}^2 \quad [1]$$

To avoid the computation of the costly square root of a real number, we utilize an equivalent formula (2):

$$d(p,q)^2 = (p_1 - q_1)^2 + (p_2 - q_2)^2;\ p,q \in \mathbb{R}^2 \quad [2]$$

Each tick of the simulation loop updates the direction of a boid, which is then applied to its acceleration, determining the actual velocity for all boids.

The direction for collision avoidance, also known as *separation*, for boid B is computed by the following formula (3):

$$direction = \sum_{i=1}^{n} position(B) - position(B_i) \quad [3]$$

where i-th boid B_i is a boid such that: $d(position(B), position(B_i))^2 < r_S^2$. This effectively means that boid B will move away (in the opposite direction) from the boids which are too near (closer than the specified r_S).

The direction for *alignment* for boid B is computed as (4):

$$direction = \left(\frac{\sum_{i=1}^{n} velocity(B_i)}{n} - velocity(B) \right) / 8 \quad [4]$$

where i-th boid B_i is a boid such that: $d(position(B), position(B_i)^2 < r_P^2)$ (i.e., B_i is B's neighbour). First, the average velocity of all neighboring boids is computed (denoted v_{avg}). The velocity of current boid B is then subtracted from v_{avg}, such that a vector in the direction from $velocity(B)$ to v_{avg} is obtained. Adding such a vector to the $velocity(B)$ would result in $velocity(B)$ being equal to the v_{avg}. Since this result is not desired, direction is lastly divided by a constant 8.

The direction for *cohesion* for boid B is computed like so (5):

$$direction = \left(\frac{\sum_{i=1}^{n} (position(B_i) - position(B))}{n} \right) / 100 \quad [5]$$

where i-th boid B_i is a boid such that: $d(position(B), position(B_i)^2 < r_P^2)$ (i.e., B_i is B's neighbour). In this formula a centroid to the neighboring boids is computed, which is then divided by 100. The reasoning behind the constant 100 is that on every update, each boid would move 1% towards the centroid.

Results

ToDo: Add Results.

Discussion

ToDo: Conclusion/Discussion.

CONTRIBUTIONS. ToDo: division of work.

Bibliography

1. Papadopoulou M, Hildenbrandt H, Sankey DW, Portugal SJ, Hemelrijk CK (2022) Emergence of splits and collective turns in pigeon flocks under predation. *Royal Society Open Science* 9(2):211898.
2. Reynolds CW (1987) Flocks, herds and schools: A distributed behavioral model in *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*. pp. 25–34.