ParBoids

Catelen Wu
cw3223

Ethan Wu
ew2664

1 Background

`Boids` ("bird-oids") is an artificial life program simulating the flocking behavior of birds developed by Craig Reynolds in 1986. It is an example of emergent behavior and swarm intelligence: each boid agent follows only a simple set of rules, but the interactions between them give rise to complex and unpredictable behavior mimicking that of flocks or herds of animals found in nature.

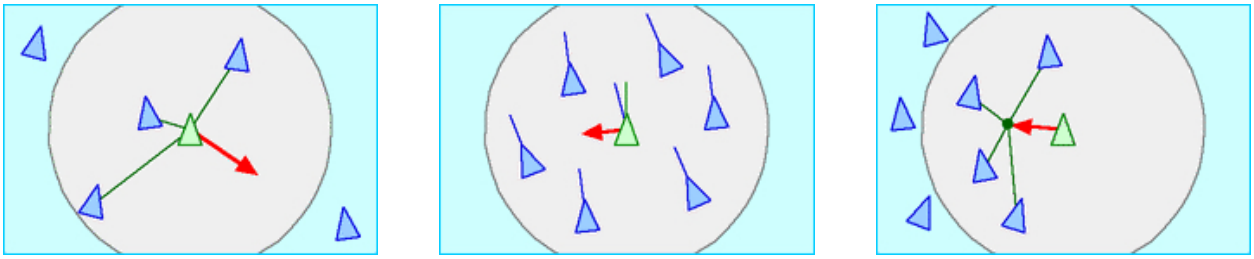

Figure 1: Steering forces of separation, alignment, and cohesion (left to right)

Each boid is subject to three steering forces: **separation**, **alignment**, and **cohesion**. (Other rules can also be added to simulate more complex behavior, such as follow-the-leader or obstacle avoidance.) These forces are computed based on the relative positions and velocities of each boid to its local flockmates. Other boids beyond a certain radius are ignored. Then, every time step, the position and velocity of each boid is updated by a weighted sum of the three steering forces. The simulation is run for a fixed number of time steps, or indefinitely. See Algorithm 1 below for specifics.

2 Project Goals

Our goal for this project is to implement a sequential `Boids` simulation in Haskell as our baseline, then to parallelize the program. In particular, we see one big opportunity for parallelism: the position and velocity of each boid is updated every time step, but each update is independent of others, so the program can likely be sped up by executing these updates in parallel. A few challenges we anticipate in the project:

- Finding an even way of splitting up the boids for each core, because the work that is required to update each boid can vary depending on the flock density in its vicinity
- Developing a graphical display in Haskell, which we don't have experience with

We also anticipate finding the local flockmates of each boid to be a big bottleneck in the program. Depending on the results of our first parallelization attempt, we may also try to implement a better data structure (e.g., a quadtree or octree) to store the boids based on their positions.

3 Appendix

Below is the rough pseudo-code for Reynolds's flocking algorithm.

Algorithm 1 Reynolds's Flocking Algorithm

```
for i ← 1, n do
    for each b ∈ B do
        nbs ← neighbors(b, B)                    ▷ Get neighbors within a certain radius
        f_s ← separation(b, nbs)
        f_a ← alignment(b, nbs)
        f_c ← cohesion(b, nbs)
        V(b) ← V(b) + k_s f_s + k_a f_a + k_c f_c    ▷ Update boid velocity
        X(b) ← X(b) + V(b)                       ▷ Update boid position
    end for
end for

procedure SEPARATION(b, nbs)                     ▷ Compute separation force
    s ← 0
    for each b_o ∈ nbs do
        s ← s − (X(b_o) − X(b))
    end for
    return s
end procedure

procedure ALIGNMENT(b, nbs)                      ▷ Compute alignment force
    v ← 0
    for each b_o ∈ nbs do
        v ← v + V(b_o)
    end for
    v ← v/len(nbs)
    return v − V(b)
end procedure

procedure COHESION(b, nbs)                       ▷ Compute cohesion force
    c ← 0
    for each b_o ∈ nbs do
        c ← c + X(b_o)
    end for
    c ← c/len(nbs)
    return c − X(b)
end procedure
```