서 울 대 학 교
SEOUL NATIONAL UNIVERSITY

Programming Assignment Report #1

soheon yoo (2021-22823)
yumk0717@snu.ac.kr
Seoul National University - Advanced Graphics (Spring 2023)

Abstract

Boids(Bird-oid) Model developed by Craig Reynolds is basic crowd simulation model[1]. In this Assignment, Boids model was implemented using python and three.js graphics API. Three rules which is basic to Boids Model are separation, alignment and cohesion rule. Additionally goal seeking, predator avoidance and obstacle avoidance were implemented in this project.

1 Introduction

Crowd simulation is the process of simulating the behavior and movement of Crowd which includes large number of character or entities. It is widely used in game engine and other graphics related industry. In simulation deciding behavior of each entities is microscopic model and deciding behavior depending on time and environment is macroscopic model. Boids model developed by Craig Reynolds in 1987 is microscopic model [1]. Boids model use simple few rules, but simulates behavior of flock very well.

2 Methods

Following model was implemented using python code. Position and velocity of boid was calculated each step and was rendered using THREE.js API. boid was rendered as red cone and predator was rendered as blue cone. Goal was rendered as green material and obstacle was rendered as white material.

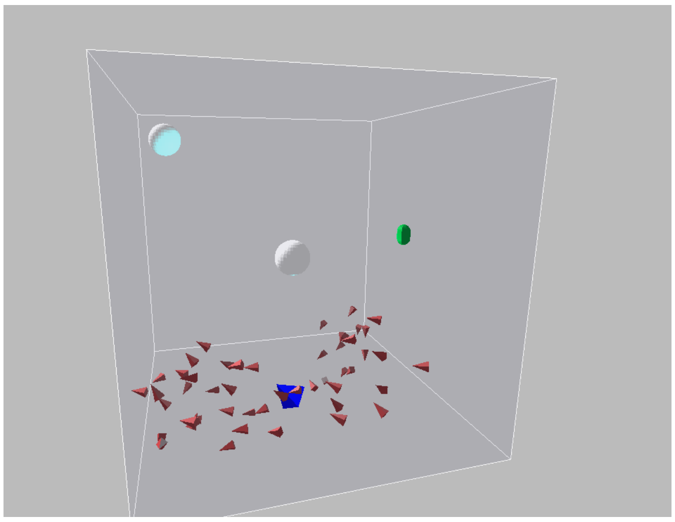

Figure 1: **Boids model system sample**

3 Model explanation

At every step calculate velocity of each boid using various rules. Elementary rule in Boids model is cohesion, alignment and separation. This rules resemble real worlds characteristic of birds or fishes. A boid(one element) in flock tend to move to center of group. And a boid has tendency to move in same direction with friend and want to have some private space. Also to make some behavior pattern in flock, other rules can be added to algorithm.I added goal seeking pattern and avoidance pattern.

```
Initialize states;
while true do
    for Each boid b_i : boids do
        $N_b = FindNeighbor(b)$;
        for each rule k do
            $v\_i = v\_i + w_k * rule_k(b, N_b)$;
        end
        $p\_i = p\_i + \delta t * v\_i$;
    end
end
```

Algorithm 1: Boids Model algorithm

3.1 Cohesion rule

First rule in Boids model is Cohesion rule. Animal in flock tend to move to center of group for some reason. Each boid find neighbor's mass center and change velocity toward center. When finding neighbor, consider visual range of boid. If distance between two boid is larger than visual range each boid cannot affect each other. Cohesion factor can steer the tendency of cohesion in flock.

```
$Rule\_cohesion(b\_i, N_i)$
$p = 0$;
for boid b_k in $N_b$ do
    $p = p + p\_k$;
end
if $|N_b| == 0$ then
    return 0;
end
$p = p/|N_b|$;
$v = (p - p\_i)$;
$v = cohesion\_factor * v$;
return v;
```

Algorithm 2: Cohesion rule

3.2 Alignment rule

Boid tend to move in the same direction with friend. So in this rule boid change direction similarly to neighbor.

```
$Rule\_Alignment\ (b\_i, N_i)$
$v = 0$;
for boid b_k in $N_b$ do
    $v = v + v\_k$;
end
if $|N_b| == 0$ then
    return $0$;
end
$v = v/|N_b|$;
return $align\_factor * (v - v\_i)$;
```

Algorithm 3: Alignment rule

3.3 Separation rule

A boid in group want a litte distance away from other boids. If not, boids can collide. So velocity from this rule is vector away from another close boid. It can be attained easily by subtracting position vector. separation velocity is proportional to distance between two boids. It seems unnatural that boids that are apart far have higher velocity from this rule. But this algorithm can make smooth acceleration. And smooth acceleration is key to natural simulation.

```
$Rule\_Separation\ (b\_i, N_i)$
$v = 0$;
for boid b_k in $N_b$ do
    $d = p\_i - p\_k$;
    if $|d| <= Separation\_dist$ then
        $v = v + d$;
    end
end
return $separation\_factor * v$;
```

Algorithm 4: Separation rule

3.4 Goal seeking rule

Cohesion, alignment and separation rule is enough to emulate real world flock. But adding some other rule can make more realistic system. Goal seeking is tendency to go some region. For example bird flock tend to go to river where have lots of food and water. By adding some velocity to boids we can make behavior of goal seeking. In figure 1 tendency to find goal was set zero. If you set up parameter higher flock will go to green goal region.

```
$Rule\_Goal\ (b\_i)$
$v = 0$;
$d = p\_goal - p\_i$;
if $|d| <= VisualRange$ then
    $v = v + d$;
end
return $goal\_factor * v$;
```

Algorithm 5: Goal seeking rule

3.5 Predator avoidance

Predator stalk boids in this system. Behavior of predator in this modeling is just following mass center of flock.

Listing 1: Rule for predator

```
def update_pred():
  global boidsN, boidsP , PredPos , PredShape, PredV , PredPower
  mass_center =np.zeros(3)

  for i in range(boidsN):
    mass_center=mass_center+boidsP[i]

  mass_center= mass_center/boidsN
  PredV=PredV+(mass_center−PredPos)*PredPower

  maxV=2
  if np.linalg.norm(PredV) > maxV:
    PredV = PredV*maxV/np.linalg.norm(PredV)

  PredPos=PredPos+PredV
  return
```

Rule for predator avoidance is similar to separation mechanism. If distance between boid and predator is less than visual range, plus vector(away from predator) to boid velocity. I think survival needs is very strong, so magnitude of velocity for rule was identical regardless of distance to predator.

```
$Rule\_Predator\ (b\_i)$
$v = 0$;
for predator b_k in Predators do
    $d = p\_i - p\_k$;
    if $|d| <= VisualRange$ then
        $v = v + d/|d|$;
    end
end
return $predatoravoid\_factor * v$;
```

Algorithm 6: Predator avoidance rule

3.6 Obstacle avoidance

Obstacle avoidance pattern is simple to implement. It is very similar to goal seeking algorithm. But goal_factor should be minus, so boid try to get away from obstacle.

4 Discussion

Real world flocking animal have speed limit. Also most of them cannot stay or cannot move very slowly(ex. bird or fish). Birds cannot fly like helicopter. They need to move everytime, so in this system velocity saturation was done. Max veolcity and Min velocity was set, and every velocity of boids are saturated between min and max like code below.

Listing 2: Velocity saturation

```
def saturate():
  global boidsN, boidsV
  for i in range(boidsN):
    if np.linalg.norm(boidsV[i])>maxVel:
      boidsV[i]= boidsV[i]*maxVel/np.linalg.norm(boidsV[i])
    elif np.linalg.norm(boidsV[i]) < minVel:
      boidsV[i]=boidsV[i]*minVel/np.linalg.norm(boidsV[i])
  return
```

For bounding in some range two options are possible. One is hard bounding and the other is soft bounding. Hard bounding is not accepting boid to go beyond range. If position of boid is beyond range change velocity toward center immediately. Soft bounding is accepting boid flying out of range a little. When boid cross the line, boid start to steer to center. Soft bounding seems natural in eye because boid does not bounce off at boundary.

Predator in this system is very stupid.They just follow mass center of total boid. In case there are two flocks, predator don't follow a flock. It fly toward center of two flocks(empty space). If you want to implement more smart predator following mass center of visible boids is more wise strategy. Of course it would take more calculation in CPU.

References

[1] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," *SIGGRAPH Comput. Graph.*, vol. 21, p. 25–34, aug 1987.