Boids

Prasanth & Suresh

May 11, 2018

Abstract

Boids implementation and mathematical modelling is discussed below. Implementation describes about the defining the the rules which boids follow in the flock. Mathematical modelling tells about the mathematics which lies underneath the rules.

1 Introduction

Boids is a program which is used to depict the simulation of the flocking of birds in the real life. Careful observation suggests that the real world flocking can be simulated by following certain simple rules (will be discussed in detail) . The efforts are taken to walkthrough the rules to be followed and the mathematics behind it.

2 Background

Boids follow some simple rules to remain in the flock , avoid the obstacles and remain in the boundaries.The simplest implementation of boids require only three rules.The rules which are to be followed are as follows:-

1. Alignment
2. Cohesion
3. Separation

2.1 Alignment

Boids try to change their position so that it corresponds to the average heading of the other nearby boids.

2.2 Cohesion

Every boid trys to move towards the average heading of the other nearby boids.

2.3 Separation

Every boid attempts to maintain a reasonable amount of distance between itself and any nearby birds, to prevent overcrowding. The implementation of the boids is discussed below.

3 Procedure

Boid is an object which has a position and velocity. So , in the three dimensional coordinate system it has three coordinates x,y,z. So the position and velocity are the objects of class vec3 - used to represent a 3 coordinate vector. There are certain constants like the minimum separation(MIN_DISTANCE), and maximum scope(MAX_DISTANCE) for each boid which suggests the neighbours.There is a flock which is used to represent the collective nature of the boids. So all the boids correspond to a single flock. So if a new boid is generated then it is added to this current flock. There are certain constants like MIN_VELOCITY and MAX_VELOCITY which confirms

that boids always move some velocity above minimum velocity and also donot cross the maximum velocity.These boids are generated in the certain environment which has boundries. The constants related to these boundaries are window width,height and depth. These boids follow certain rules which are the functions which take some parameters and add the returned result to the velocity of respective boids.In order to increase/ decrease the strength of the some rules.
The flow of the working is as follows. The boids are instantiated with random velocities at random positions in the space and move along the velocity vector.Between each frame of rendering, the boids follow each rules and the velocities and other attributes of the boid are updated.

4 Mathematical Modelling

The mathematics and the working process is as follows. First initialize the boids at random positions ,with random velocity. Assume that the position and velocity vectors of the boid b_i are p_i and v_i respectively .Next apply the rules for each boid. Every rule contributes to a fractional change in the magnitude and direction of velocity of the boid . The rules applied are as follows:-

4.1 rule1

Inorder to apply the **alignment** rule, the average of the velocity vectors of the nearby boids will be taken and will be multiplied by a factor which indicates the weight of that rule.Let b_1 to b_i boids are near to the present boid , then

$$rf_1 = \frac{v_1 + v_2 + \cdots + v_i}{i}$$

4.2 rule2

The **cohesion** rule works as follows. Since the boids try to move towards the average heading of the nearby boids, this depends upon the positions of these nearby boids. So the center of mass of these nearby boids is taken.
Let b_1 to b_i boids are near ot the present boid,then

$$rf_2 = \frac{p_1 + p_2 + \cdots + p_i}{i}$$

4.3 rule3

In **separation** rule, the boids try to steer away to avoid crowding. This is done by averaging the difference in the position vectors of the nearby boids with itself.
Let b_1 to b_i boids are near ot the present boid(b_j),

$$rf_3 = p_j - \frac{p_1 + p_2 + \cdots + p_i}{i}$$

4.4 rule4

For applying the boundary conditions, and the minimum , maximum velocity constraints , a small rule factor can be added to the velocity vector of the boid to get it back into the boundaries.

4.5 Bringing all together

In a similar way , some more rules could be included to achieve more complex movement.Let the contribution by those rules be rf_i.
Inorder to increase or decrease the influence of the rule on the velocity and other attributes of the boid , certain weights are multiplied before it is actually added to the velocity.Let the weights associated with rules be W_i.If b_j is the boid with which we are dealing with then,

$$v_j = v_j + rf_1 * W_1 + rf_2 * W_2 + \cdots + rf_i * W_i$$

The position can now be foundout by the formula , where frameTime is the time taken between the execution of two consecutive frames.

$$p_j = p_j + v_j * frameTime$$

References

[1] https://www.opengl.org/discussion_boards/showthread.php/166981-bouncing-ball