# CMSC 435/634: Introduction to Computer Graphics

## Assignment 5
## Boids
## Due Novemeber 20, 2017 @ 11:59 PM

## The Assignment

The assignment is to write a program that performs a simple crowd (Boid) simulation. The wikipedia page has a number of useful links about the model and the original paper is linked here and is fairly easy to read. Boids should react to other boids that are visible, i.e. that are within some distance. I have provided you with code for a KDTree to help speed up these neighbor queries. Boids have three goals

1. Collision Avoidance: avoid collisions with nearby flockmates.
2. Velocity Matching: attempt to match velocity with nearby flockmates.
3. Flock Centering: attempt to stay close to nearby flockmates.

An executable of my program is at:

```
~adamb/public/Fishtank/fishtank sample.in sample.out
```

### Viewer

I have pushed an openGL based viewer to your repositories. This will compile on gl as is and on mac with a one line change to the makefile. Its syntax is

```
viewer sample.out
```

it will loop the animation. Press 'q' to quit.

### Note about food

Reacting to food is extra credit, for the basic assignment you can ignore everything about food below. You will need to parse those lines of the input, but you can ignore them and just write 0 size to the output.

### Input

The first line of the input file contains a number of parameters of the boids system:

```
size neighbor_radius num_neighbors mass collision centering velocity hunger damping dt length
```

size is the size of the boid. This is useful for telling if the boid has consumed food. neighbor_radius is the distance the boid can see, any boid farther than neighbor_radius is ignored. num_neighbors is the maximum number of neighboring boids to process. mass is the mass of the boid. collision, centering, and velocity are weights for these various forces. hunger is a weight for the force that draws boids to the nearest food. damping is a "mass proportional damping" constant. Multiply the velocity by this constant every timestep. dt is the timestep size. length is the length of the animation.

After that boids and food are specified. The next line has the number of boids, nboids, followed by nboids lines with two slVector3, the first is position, the second is initial velocity. Then the number of pieces of food, nfood, followed by nfood lines with two SlVector3s (again, position followed by velocity) and a float that give the time that the food should be added to the system.

### Output

The first line is the number of frames, nframes. This is followed by nframes of frame data. Each frame consists of boids and food. The first line of a boid segment contains the number of boids, nboids, followed by nboids lines with position and velocity (similar to the input). The food segment has the number of pieces of food, nfood, followed by nfood lines with the food position (the viewer ignore velocity).

### Details about my setup.

- I found weighting the collision avoidance force by the inverse sqaure of the distance between boids (as suggested by Reynolds) to work well.
- I weighted the flock centering force by the distance, so that it is larger the farther a boid is from the center of its neighbors.
- My Boids are constrained to a box from [-0.5,0.5]x[-0.25,0.25]x[-0.125,0.125] (when they leave the box I flip their velocity).
- I included "mass proportional damping;" I multiply the velocity by 0.999 every timestep.
- I will add more details to this page as questions come in.

### Extra Credit

For 30 points of extra credit, feed the Boids with a particle system. Don't ignore the food in the input. When the time in your simulation passes the time that the food should become active, add the food to the system. The food falls slowly through the scene with small random perturbations to the x and z components of the velocity. There is an additional force on the Boids that draws them to the food and when they get close the food is eaten and disappears.

For up to 40 additional points, extend your program with additional phenomena or forces. Be creative. Points will be awarded based on how cool the grader thinks your enhancements are. There are lots of ideas for extensions in the various links on the wikipedia page and here

## What to turn in

Turn in this assignment electronically by pushing your source code to your `proj5` GIT directory by 11:59 PM on the day of the deadline. We will be looking for multiple checkins documenting your development process.

As always, double check that you have submitted **everything** we need to build and run your submission, but **no** generated files (.o's, executables, or images). Be sure to include a Makefile that will build your project when we run '`make`', and a `readme.txt` file telling us about your assignment. Do not forget to tell us what (if any) help did you

receive from books, web sites or people other than the instructor and TA.

(This page: www.csee.umbc.edu/~adamb/435/proj5.html)