LINKÖPING UNIVERSITY

MODELLING PROJECT

TNM085

Simulation and Visualization of a School of Fish Using OpenGL

Author:
Jakob Bertlin
David Tran
Anton Sterner
Rasmus Ståhl

Supervisor:
Dr. Anna Lombardi

2017-03-12

Abstract

Swarm behavior is a widely researched area but without many conclusive answers. Different kinds of swarms have different kinds of behaviors, and in this project the purpose is to simulate the schooling and shoaling patterns of fish using a Particle swarm optimization algorithm. Each individual fish has a set of rules which depends on its position relative to its neighbors and a universal target. To gain an insight into how the algorithm should be implemented, the first testing was done in MATLAB, which has extensive support for calculating and displaying large numbers of positions. The first version of the algorithm included moving particles towards a common goal, while staying together as a group and avoiding collision with neighboring particles.

The final program was written using C++ in the CLion IDE and OpenGL was used to generate the graphics. Furthermore the the algorithm was tuned to include other factors that affect their movement such as their tendency to turn from their current path. An ocean environment was also made, as well as making the fish's tail move back and forth to make the visualization more realistic. In the final program the fishes follow a single target which is controlled using the mouse and keyboard, giving the user the ability to observe different behavior like schooling and shoaling seamlessly.

The fish school movement patterns implemented resemble the results of various research documents. There are no explicit conclusions of how fish move and what defines good fish movement, so the fine tuning of the algorithm was made solely based on observations of the swarm rather than quantified values. Even if the fine tuning of the algorithm was based on values defining good movement as opposed to observation, all that really matters is that the resulting animation looks close to a real school of fish.

Table of contents

List of Figures

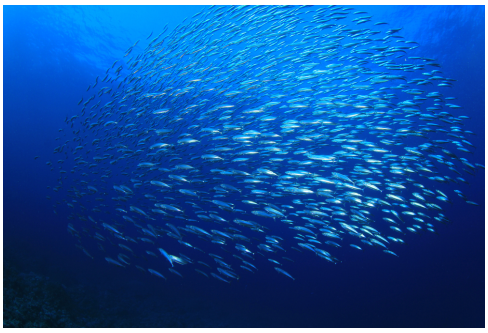
Chapter 1

Introduction

1.1 Background

The behavior of collective living organisms has always been a fascinating subject. An interesting question is how a group of species orient themselves in their surroundings when searching for food or avoiding predators. The question is if an intelligent leader is required to coordinate the group or if every individual organism contributes to the orientation of the group. This has lead to many investigations about the behavior patterns of different species such as ants, birds and fish.

According to the *The International Union for Conservation of Nature and Natural Resources*[1] there are roughly 30,000 different species of fish in the ocean. Different kind of species have different kind of behaviors. The term *school* of fish describes fishes that move parallel to their nearest neighbor and in the same direction. Usually a school of fish is of the same species, however, a *shoal* of fish is different. A shoal of fish is a group of fish that stays connected, forming a social group. Their orientation is rather random compared to a school of fish. The distance to the nearest neighbor and species in the shoal can vary widely.



(a) School of fish

(b) Shoal of fish

Figure 1.1: Figures of the behaviors of fish

1.2 Purpose

The purpose of this project is to simulate and visualize the schooling and shoaling patterns of fish. A simulation of the behavior of fish will be implemented in MATLAB and visualized using OpenGL, through a particle swarm optimization algorithm.

Chapter 2

Simulation and visualization

2.1 Particle Swarm Optimization

The simulation implemented in this project is not based on a real physical system. It is based on the optimization algorithm *particle swarm optimization*. The algorithm was originally developed to mimic the movement patterns of flocking birds and schools of fish [2]. It was later realized by its inventors, Russell Eberhart and James Kennedy, that the algorithm could also be used for numeric optimization. In this report, the simulation is true to the origins of the algorithm, simulating the schooling of fish. The goal of this simulation is not just to make sure the fish find the goal as quickly as possible, but also to make the school move like one would in real life. In the scientific paper *Simulating The Collective Behavior of Schooling Fish With A Discrete Stochastic Model* [3] written by Alethea Barbaro, Bjorn Birnir and Kirk Taylor, the authors set up some rules for each individual fish which tells them how they should behave with respect to their neighboring fish. The basics of this is shown in figure 2.4. If a neighboring fish approaches the zone of repulsion, the fish at the center is compelled to move away from its neighbor, preventing the collision. However, if a neighboring fish crosses into the zone of attraction, the fish moves toward its neighbor. A zone of attraction and repulsion are used in the algorithm implemented in this project, while the zone of orientation is not needed as all fishes are moving toward the same goal, without need for orientational adjustments.

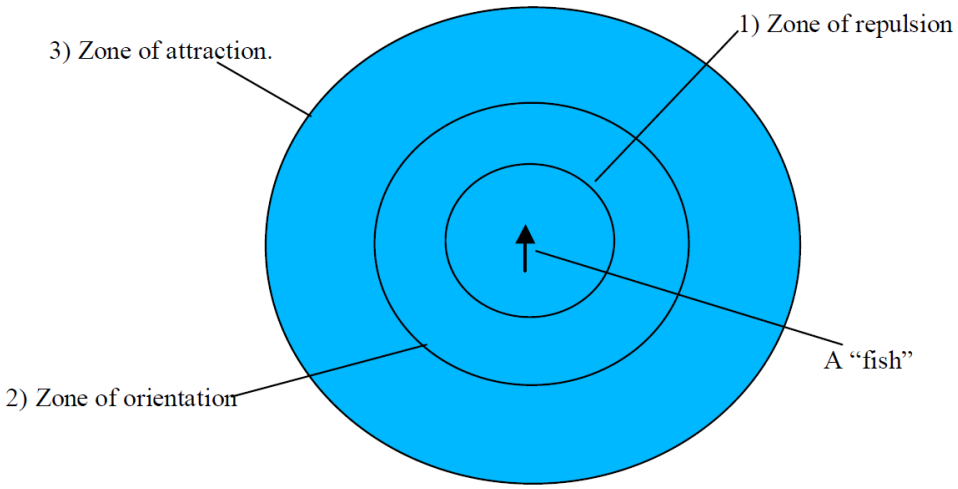


Figure 2.1: Zones of attraction, repulsion and orientation of a fish

2.1.1 Numerical method

The algorithm itself is a numerical method, which solves an optimization problem by moving particles in a search space. In this case, the problem to solve is the position of the mouse cursor. That is, the closer a particle is to the cursor, the smaller is its *fitness value*. The other particles will gain velocity towards the particle having the smallest fitness value. That makes all particles move in roughly the same direction. Pseudo code for the basics of the PSO algorithm can be seen in algorithm 1.

Algorithm 1 Pseudo code, particle swarm optimization

```
For each particle
    Initialize particle
Do until maximum iterations
    For each particle
        Calculate fitness value
        If the fitness value is better than personal best value
            personal best = current fitness value
        If personal best is better than global best
            global best = personal best
    For each particle
        Calculate particle velocity vector using global best
        Calculate new particle position using the velocity
```

The final algorithm used in the simulation is defined in 2.1 and 2.2.

$$\begin{aligned}\frac{dx}{dt}x_i(t+1) = \omega\frac{dx}{dt}x_i(t) + (SlowdownC_1r_1(G - x_i(t)) - \\ -RepulsionC_2r_1\frac{x_i(t) - x_{G1}(t)}{|x_i(t) - x_{G1}(t)|} + Slowdown(C_3r_1(x_i(t) - x_{G2}(t))) * Turnrate\end{aligned} \tag{2.1}$$

$$x_i(t+1) = x_i(t) + \frac{dx}{dt}x_i(t+1)dt \tag{2.2}$$

In 2.1 and 2.2, $\frac{dx}{dt}x_i$ is the velocity vector of a particle and x_i is the position of a single particle. The position $x_i(t+1)$ is calculated by adding the velocity vector to the previous position. G is the calculated "global best" position, which is the position of the particle which is closest to the target. r_1 is a normalized random number and x_{G1} and x_{G2} is the position of the nearest- and second nearest neighbor, respectively. $Slowdown$ is a coefficient which is used to slow down the velocity of a particle in the direction of the target and in the direction of the particle's second nearest neighbor, if its nearest neighbor enters the repulsion zone. The coefficient $Repulsion$ is used when the nearest neighbor of a particle enters its repulsion zone, which makes the particle move away from its neighbor. $Turnrate$ is a variable that which makes the particle change direction quicker, the farther away it is from the target. The coefficients C_1, C_2 and C_3, are explained in figure 2.2.

```
FishAmount = 200;
InitialSpeed = 0.1;      % 0 - 1
C1 = 1.5;      % Globalbest coefficient
C2 = 0.2; % Neighbour avoid coefficient
C3 = 0.3;    % Neighbour attraction coefficient
dt = 0.05;    % Time step
REPULSION_RADIUS = 0.1; % Zone of repulsion
SPEED_CAP = 0.3;
TURN_RATE = 0.04;
```

Figure 2.2: Declaration of algorithm parameters.

The main purpose of including the three terms which include C_1, C_2 and C_3 is to display the particles so that they all move toward the target, while avoiding their nearest neighbor and not straying away from the collective particle system.

As can be seen in the equations, the ”personal best” value from algorithm 1 is not used. This is because the target which the particle's seek is continuously moving, so a personal best position would not be of any use, it would cause the particles to get stuck when the goal moves further away.

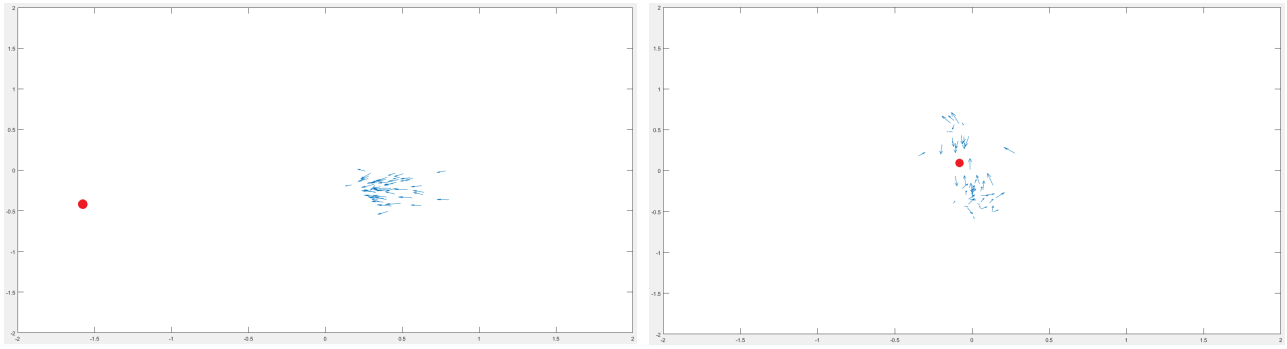
2.2 MATLAB simulation

The algorithm was first implemented and simulated in MATLAB. The algorithm in the simulation is slightly less refined than the final OpenGL implementation, as the behavior of the particles was difficult to evaluate using only two dimensional plots in MATLAB. Since there is no obvious optimal value for a fish school simulation, the values of these coefficients have been determined based on visual interpretation only.

The position and velocity of each particle is stored in a particle struct. The particle struct has three fields, position, velocity and cost. Every iteration of the initialization loop generates a starting velocity and a random starting position in three dimensions x, y, z for each particle.

In the main loop, the global best cost is set to infinity at the start of each iteration. That is because the particles are to follow the mouse cursor position, and not get stuck at a certain position.

In figure 2.3 the MATLAB simulation is shown. The particles are displayed as arrows which point in the direction in which they move. The red dot is the target.



(a) Snapshot of the fish schooling toward the target (b) Snapshot of the fish shoaling the target

Figure 2.3: Snapshots of the Matlab animation of the school

2.3 OpenGL visualization

The final visualization was written in C++ using the Jetbrains CLion IDE for cross-platform support. The algorithm used in the MATLAB simulation was a rough approximation of the desired movement pattern, so the algorithm has been tweaked in the OpenGL visualization to better resemble the movements of real fish. For displaying graphics on-screen mainly OpenGL and various libraries supporting OpenGL were used. Swarm behavior was implemented as two C++ classes, one class handling the entire swarm and another class handling each individual fish and its movements. The class containing the entire swarm consist of a user defined amount of fishes/particles. To enhance the realism of fish movements many techniques for smoothing their turning rate was used such as increasing a fish's tendency to face its target as the distance between them grows larger. The camera position is fixed in one position at all times. As of now there are no constraints to where the target and the fishes can move to. This can result in moving the target outside of the screen boundaries.

3D models used were downloaded from tf3dm.com [4] and then utilized for rendering and animation through OpenGL. The same fish model was used for all the fish, and a model of a loaf of bread was used to represent the target.

2.3.1 Fish motion with sine wave

When rendering the fishes, a sine wave along the fish is used to manipulate the vertex positions of the fish object, where the amplitude of the sine wave depend on the velocity of the fish. This makes the tail of each fish flap back and forth, and it looks a lot more natural and correlated to its movements.

2.3.2 Environment

The environment was done using a simple box mesh with a blue texture to represent the ocean. The color changes to a darker shade toward the bottom of the screen to somewhat mimic the ocean depths.

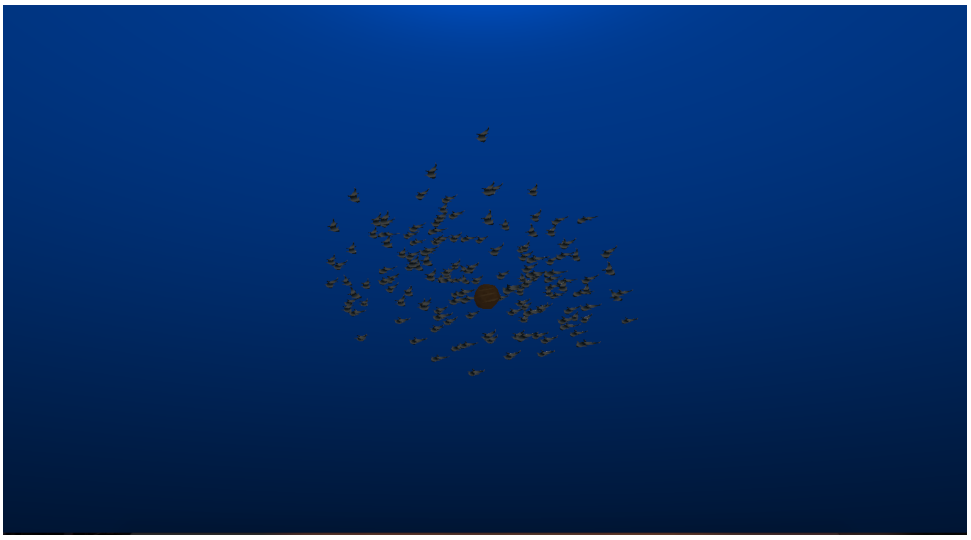


Figure 2.4: Ocean environment

2.3.3 Requirements and control

The visualization does not require a powerful GPU or CPU to run it. It was created on various laptops using specifications as low as 5th generation Intel i5 processors using their integrated graphics, which should be considered the minimum requirements.

To start the program a simple .exe file is executed. The file depends on two shader files (fragmentshader.glsl, vertexshader.glsl) that have to be placed in the same directory as the executable. When the program is started mouse movements are used to move the target up,down,left and right. To move the target closer or further away in the scene use W and S on the keyboard.

Chapter 3

Analysis and discussion

3.1 Discussion

As described in section 2.2 there is no obvious way of quantifying the success of the implementation other than by visual interpretation. If however this was to be done, there would have to be research done to extract information about the movement patterns of real schools of fish. That would then have to be interpreted mathematically so that real numerical data could be used for comparison. If such data would be available, perhaps the distance between the fishes at different positions could be used in the simulation to evaluate whether the particles are moving in a realistic manner, based on the comparison of the data from a real school.

Something that could also be used to evaluate the swarm behavior is to analyze data on how quick and how much each fish turns based of real fish and use it as an indicator for how realistic the fish moves in the simulation. But these values still does not mean anything if the fish does not truly look like it is moving realistically. That is why the results and algorithm of this project has almost entirely been based of images and video of real fish movements.

The best way of evaluation may be to compare the simulation with footage of a real school of fish and to not at all use values to determine the accuracy as it can cause the developer to chase values that does not truly represent fish behavior.

3.2 Further development

There is a lot of functionality and improvements that could be added to the program in the future. For example, adding an object that the fish are programmed to avoid, such as a shark. This could be implemented rather simply by basing it on the original algorithm, but reversing the direction of velocity of the fish when the predator approaches.

Another addition could be to include the ability for the user to change the values of the coefficients in real time to see their effects directly. A small window could be implemented with sliders for changing these values. This could be used to increase the realism of the fish school as it can be difficult to observe changes in behavior of the swarm when closing and opening the program every time a variable is modified. The ocean environment could be implemented with a surface textured animated mesh to represent the ocean surface. Crepuscular rays could also be included to emphasize the underwater lighting effect.

The computational performance of the visualization is not optimized as of now. This could be improved by utilizing multi-threading and using models with a lower polygon count.

Chapter 4

Conclusion

The final result of the animation demonstrates a clear similarity to real schools of fish. The fish all swim towards the target, they do not collide with each other and they stay together as one unit. The simulation shows that it is possible to replicate the behavior of fish using quite simple algorithms.

The absence of numerical evaluation based on data collected from real schools of fish may make the final result less legitimate, but for a small project like this, the visual evaluation method may be sufficient. In a larger project the simulation should be based on data collected instead of only visual interpretation.

Bibliography

[1] http://factmonster.com, Estimated Number of Animal and Plant Species on Earth. 2007. [cited: 2017 Mars 3]. Available from: http://www.factmonster.com/ipka/A0934288.html

[2] mnemstudio.org. Particle Swarm Optimization. John McCullock; 2012. [cited: 2017 Mars 3]. Available from: http://mnemstudio.org/particle-swarm-introduction.htm

[3] Alethea Barbaro, Bjorn Birnir, Kirk Taylor. Simulating The Collective Behavior of Schooling Fish With A Discrete Stochastic Model. Funded by: The National Science Foundation and The Research Fund of The University of Iceland; 2006. [cited: 2017 Mars 3]. Available from: https://www.mrl.ucsb.edu/sites/default/files/mrl_docs/ret_attachments/research/KTaylor.pdf

[4] TF3DM. 2017. [cited: 2017 Mars 11]. Available from: http://tf3dm.com/