Dynamical and computational structures under the sea: modelling of fish motion

Timothy James Roberts
University of Manchester
School of Computer Science
Supervisor: Dr. Eva M. Navarro-López

***Abstract*—A simulation comprised of a multi-agent system based upon Couzin's Model of swarming/flocking is created to demonstrate the movement and behaviour of fish. The model used is a modified version of Couzin's Model in that two species of fish can interact with one another. The two species have their own parameters to determine how they react to their own species and the other species. The parameters can be altered in order to observe unique behaviours of the fish. These behaviours can then be compared to real fish behaviour to measure the effectiveness of the simulation. A future implementation would allow more parameters of the fish to be available.**

***Index Terms*—Dynamical Systems, Swarming, Couzin's Model, Multi-agent Systems, Vectors.**

CONTENTS

I. INTRODUCTION

This project is about simulating a dynamic model of fish swarming. The movement and behaviour of fish must be understood in order to replicate these characteristics in a computer simulation.

A. Fish Swarming

Shoals of fish exhibit a behaviour known as swarming in which multiple fish seemingly move together as one unit. Visually, swarms of fish can create very complex and unique shapes, which was an interest throughout this project. Fish in swarms have many advantages compared to solitary fish such as predator avoidance and a higher foraging success. This is due to the fact that if a fish detects something of interest, it will essentially transfer this information to the rest of the swarm through it's movement. A lone fish will react quicker to a predator than a swarm of fish ; however, the swarm will confuse the predator and allow the fish to conserve energy for other tasks [1].

Fig. 1. An example of fish swarming [2]. The fish will organise themselves into a cohesive group.

Fish will form a swarm by observing the other fish around themselves and moving similarly to them. An interference at one end of the swarm will usually be passed through the whole swarm. The fish will try to avoid colliding with other fish, by keeping a certain distance away from the others and they will move towards other nearby fish so as to increase their numbers. The fish also align themselves with nearby fish, so that a group of fish in the swarm will have a similar orientation. Fig. 1 shows how the fish align themselves. If you split up the image, the fish in each section will be pointing roughly in the same direction.

The structure of a swarm can be described to some extent, although it can be difficult due to the vast amount of fish. The most obvious is the swarm size. A swarm with many fish can produce more varied behaviour than a smaller swarm. Some swarms can contain upwards of 10 million fish [3].

Polarity describes the orientation of the fish as previously

mentioned. The polarity of the entire swarm can be calculated by first finding the average orientation of the fish. Then for each fish, find the angle between it's orientation and the swarm orientation. The average of all of these angles will be the swarm polarity [4].

A measure of how compact the swarm is, is the nearest neighbour distance. This involves finding the distances between each fish and it's closest neighbour and then averaging the results. Some fish, such as sardines, will group together to form a sphere like object known as a bait ball as a last ditch attempt to fend off predators.

More detailed knowledge of fish movement isn't necessary for this project. The report structure will be explained next.

B. Report Overview

The basic structure of this report is as follows. This section of the paper will provide a brief description of what was set out to be achieved and what the results show. The second section will focus on background information used to implement the simulation, with the third section explaining how the simulation was implemented. The fourth section will discuss an in-depth evaluation of the results and the final section will give some conclusions on what was achieved.

C. Objectives

The goals of this project were:

- To investigate different methods of modelling swarming.
  - Study the swarming behaviour of fish and see what mathematical models are available.
  - Choose a model that best suits fish swarming.
- To implement a swarming model.
  - Decide what language to use.
  - Design how to map the model to code.
- To modify the model to allow two species.
  - Decide how the model must be changed.
- To observe how the simulation changes with respect to the parameters.
  - Find settings to provide distinct behaviour.

These steps ensure the end result has no major flaws.

D. Summary of Results

The behaviour that was set out to be emulated, was that of the behaviour shown in one of Iain Couzin's papers [5]. The paper shows how different behaviours emerge as the parameters change. The three main behaviours shown, for one species, were a swarm, a torus and a polarised group. These behaviours were discovered in the simulation.

The second set of results were for two species. Due to the fact that there are lots of parameters to modify for two species, finding interesting behaviour was a challenge. The results show a groups of fish avoiding a polarised swarm of fish and it shows how some fish would be extremely cautious even when the fish they are avoiding have no interest in them. Another set of results show how the prey formed a structure similar to a bait ball previously mentioned. The prey would stay within a small area and move out of the way of the predators that were surrounding them.

II. Background Information

A. Multi-agent Systems

Due to the way fish interact with one another, the simulation was implemented as a multi-agent system. This is a computerised realisation of intelligent agents interacting in an environment. In this case, the fish are the agents and the environment is the ocean or an aquarium.

An intelligent agent is an autonomous entity, in the sense that it can perform actions on it's own without the need for user input. The agents will observe the environment through sensors ,which determine what specific information it will need, and then perform an action based on that information. Stuart J. Russell and Peter Novig defined five types of intelligent agents [5]:

- Simple reflex agents
- Model-based reflex agents
- Goal-based agents
- Utility-based agents
- Learning agents

Simplex reflex agents are the what was previously described. They only perform an action based on how the environment state is at that point in time and ignore anything that was observed before.

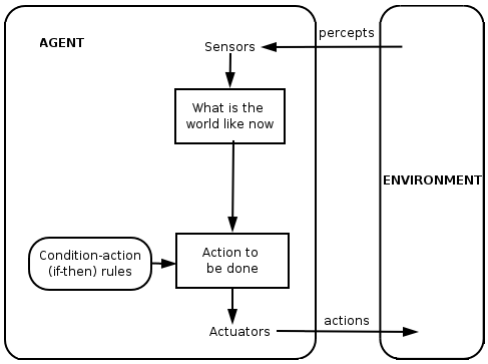

Fig. 2. A diagram of how a simple reflex agent works [6].

Model-based reflex agents are similar to reflex agents, except that they store a structure describing how the environment has evolved over time. This allows the agent to observe the environment that it can't see. It then acts the same as a simple reflex agent.

Goal-based agents and utility-based agents expand upon model-based reflex agents. Goal-based agents will use 'goals' to decide what action to take. The action that will be performed ill be the one that best suits one of the goals it wants to achieve. A utility-based agent uses a utility function in order the measure the desirability of the state of the environment. The action performed will be the one that produces the best utility of the environment state.

Finally, learning agents have a 'performance element', which determines which actions to take, and the 'learning element', which improves the performance element, thus ensuring that the learning agent will operate more effectively over time.

Intelligent agents have three characteristics, as specified in Michael Wooldridge's textbook [7]:

- Autonomy - Agents are autonomous to some extent.
- Local views - Agents won't have a global view of the system.
- Decentralisation - No agent is in control of the others.

It's unreasonable to suggest that an agent would have a full view of it's environment. In the context of fish, how would one fish know the positions and orientations of every other fish in it's swarm? It could only know information about the fish close to it and the ones it can see.

The multi-agent system used in the simulation will be a simple reflex agent as it best represents the behaviour of the fish.

B. Swarming Models and Algorithms

There exists several mathematical models and algorithms relating to swarming. Most of the models conform to notion that an agent has three rules associated with it.

- Align with the neighbours.
- Move towards the neighbours.
- Don't collide with the neighbours.

Tams Vicsek developed *Vicsek Model* in 1995, which modelled the behaviour of active matter. This model was explored more than others as it can be applied more easily to fish. It's rules are slightly different to the standard convention. The idea is that at each time step (t) of the simulation, an individual (i) at position r, will align itself with it's neighbours within a certain range. Random noise (η) is then added to this new angle of velocity (Θ).

$$\Theta_i(t + \Delta t) = \langle \Theta_j \rangle_{|r_i - r_j| < d} + \eta_i(t) \tag{1}$$

$\langle \Theta_j \rangle_{|r_i - r_j| < d}$ specifies the average directions of the neighbours of i withing the range d. The individual then moves a constant speed (v) in the new direction[8].

$$r_i(t + \Delta t) = r_i(t) + v\Delta t \begin{pmatrix} \cos \Theta_i(t) \\ \sin \Theta_i(t) \end{pmatrix} \tag{2}$$

This model, unlike others, is designed in 2-dimensions.

There are many other models and algorithms that apply to other animals such as bats, bees and ants. These models and algorithms can't be applied to fish as they, but are useful in other aspects.

The bat algorithm makes uses of echolocation, developed by Xin-She Yang in 2010 [18]. The bats send out pulses of sound with a varying frequency or wavelength and loudness, in order to detect prey. It will also change the emission rate and overtime find parameters that optimise it's prey detection.

Several of the different swarming algorithms relate to optimisation problems. With bees, it involves finding a flower to determine if it can provide the most nectar or pollen. For implementation, it mirrors finding a candidate solution and determining it's fitness.

The model used for this project is Couzin's Model.

C. Couzin's Model

The section uses information from Dr. Eva M. Navarro-López's lectures [10] and the *Nature* journal [11]. Couzin's Model was used as it is very simple to understand and each of the steps associated with it can be broken down into multiple sub-steps. It was designed by Iain Couzin as a variant to the Boids model. Couzin's Model represents an agent as having three zones around it, similarly to the three rules described in the previous sub-section. They are:

- Zone of repulsion (ZOR).
- Zone of orientation (ZOO).
- Zone of attraction (ZOA).

The zones work in the same way as the previous rules only that sizes of the zones are, from smallest to largest, ZOR, ZOO and then ZOA. The radii for each zone are expressed more formally:

$$0 > r_r > r_o > r_a \tag{3}$$

The zones can be disabled but it's standard for ZOR to be always active or else the agents will pass through one another. Each zone is modelled as a sphere. If an agent resides within the zone of another agent, the appropriate calculations are performed.

The model operates under time steps, t, with each time step progressing by τ. At each time step, an agent's (i) new direction, d_i, is given by:

$$d_i(t + \tau) \tag{4}$$

The direction vectors for each agent are normalised, to ensure each direction vector has the same length and therefore the same speed throughout a simulation.

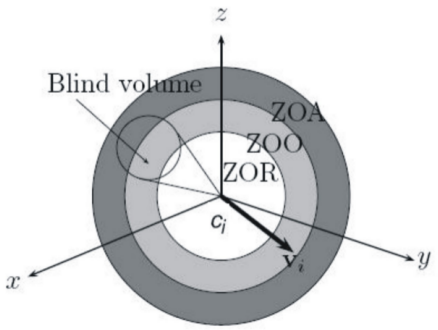

Fig. 3. A simple view of Couzin's Model showing the different zones and the blind volume[10]. c_i represents the position vector. v_i represents the direction vector.

The following equations relate to an agent's (i) new direction vector. The new direction for the zone of repulsion (ZOR) calculations (d_r) of i is given by:

$$d_r(t + \tau) = \frac{-\sum_{j \neq i}^{n_r} r_{ij}(t)}{|\, -\sum_{j \neq i}^{n_r} r_{ij}(t)\,|} \tag{5}$$

n_r represents the number of neighbours within the ZOR and r_{ij} represents the direction vector $\vec{ij}$ given by:

$$r_{ij}(t) = \frac{c_j(t) - c_i(t)}{|\, c_j(t) - c_i(t)\,|} \tag{6}$$

c represents the position vector of an agent.

Firstly, the direction vectors between i and any other agent within it's ZOR are calculated and normalised. These are all added together and multiplied by -1. This is because i needs to move away from the other agents. The result is then normalised again. The ZOR has the highest priority meaning that if n_r is greater than 0, only the ZOR calculation are performed. The ZOO and ZOA calculations are ignored regardless of however many neighbours might be in each zone. This is because the agents should never collide and must avoid doing so.

The zone of orientation (ZOO) calculations are a bit simpler. The new direction after ZOO calculations (d_o) is:

$$d_o(t+\tau) = \frac{\sum_{j=1}^{n_o} v_j(t)}{\mid \sum_{j=1}^{n_o} v_j(t) \mid} \tag{7}$$

The direction vectors of the neighbours withing the ZOO (n_o) are added together and normalised. The direction vector of i is also included as it avoids two agents from swapping direction vectors between each other. The new direction will be an average of the two.

The zone of attraction (ZOA) equation is almost identical to the ZOR equation. The only difference being that there is no multiplication of -1 applied to it. The new direction after ZOA calculations (d_a) is:

$$d_a(t+\tau) = \frac{\sum_{j\neq i}^{n_a} r_{ij}(t)}{\mid \sum_{j\neq i}^{n_a} r_{ij}(t) \mid} \tag{8}$$

If neighbours are within the ZOO and the ZOA, then the average between the two is found. This is done by adding $d_o(t+\tau)$ and $d_a(t+\tau)$ together and again, normalising the result.

There is also the notion of a 'blind volume'. In the context of fish, a fish won't be able to see behind itself so it shouldn't react to anything behind it. The blind volume is represented as a cone coming out of the back of the fish. Any fish within the cone won't be considered for any of the zone calculations. The interior angle of the cone can be decided, but shouldn't be any bigger than $100°$.

Finally there is the turning rate. When an agent calculates a new direction vector it shouldn't set it's direction vector to it straight away as it will produce a 'snapping' movement. The agent will immediately change it's direction and will look unnatural in the context of fish movement. The turning angle ensures that the new direction vector of an agent is set to one such that the angle between it and the original direction vector is no bigger than the turning angle. If the angle is larger, then the original direction vector is rotated towards the new direction vector by the degrees specified by the turning angle, otherwise the original direction vector is set to the new direction vector.

III. Implementation

The simulation was implemented in C using OpenGL and the OpenGL Utility Toolkit (GLUT). OpenGL provides the 3D graphics necessary for the simulation to look appealing and understandable. Couzin's Model requires several short functions in order to compute different attributes of vectors such as length and cross product and there is lots of OpenGL documentation that assisted with this.

A. Fish Variables

The fish are represented in the code as an array of structures.

TABLE I
FISH STRUCTURE VARIABLE INFORMATION

Variable Type	Variable Name
GLfloat[3]	*position_v*
GLfloat[3]	*direction_v*
GLfloat[3]	*next_direction_v*
int	*in_ZOR*
int	*in_ZOO*
int	*in_ZOA*

Table 1 shows the variable types and names included within the fish structure. The fish move about within a 'box' that has been defined with the variable *box_edge_size*. This is a positive float that is used to form the boundaries of the area creating a $2{*}$*box_edge_size*$*2$*box_edge_size*$*2$*box_edge_size* cube.

All of the vectors involved are in three dimensions. *Position_v* and *direction_v* are initialised to have random values within the area limits, but with *direction_v* being normalised to have each fish travel at the same speed. *Next_direction_v* is set to the zero vector since no zone calculations would have been performed yet and it avoids any errors for when they do. At the end of each time step, in this case whenever the next frame can be displayed, *direction_v* is updated using *next_direction_v*. This involves the turning angle, which will be explained later and *position_v* is updated using *direction_v* at this point as well.

In_ZOR, *in_ZOO* and *in_ZOA* are boolean values used to identify whether a fish has any neighbours withing it's ZOR, ZOO or ZOA respectively. If *in_ZOR* is true, then the ZOO and ZOA calculations can be ignored. Similarly, if *in_ZOO* is true, then the fish's *direction_v* is added to *next_direction_v*. It should become clear as to why it had to be done that way later. If any of the zone identifiers are true at the end of the time step, then *next_direction_v* is normalised and the zone identifiers are set to false.

When single species mode is on, the position of each fish is mapped to RGB values allowing the fish to change colour as they move throughout the scene. Aside from interesting aesthetics, it allows a user to easily see where different swarms have formed based purely on their colours. The equation in (9), shows how each dimension of the position vector maps to each of the RGB values.

$$rgb[i] = \frac{position[i]}{box\_edge\_size * 2} + 0.5 \tag{9}$$

B. Functions, Trigonometry and Vectors

The main function of the program is the *update_fish* function. This involves moving the fish and calculating the new direction vector through the zone calculations.

The pseudocode provided in algorithm one shows a simplified description of how *update_fish* operates for one species. The first step of the function is to move the fish. This involves altering the position vector, taking into account the turning angle, as well as making sure a fish stays within the area

boundaries. This function is aptly named, *move_fish*. The algorithm for *move_fish* is shown in algorithm two.

The rotating vector concept is implemented by having the direction vector rotate towards the next direction vector by the turning angle (Θ). The first part of this function involves checking to see if the angle between the direction vector and the next direction vector is less than Θ. If so, the direction vector is set to the next direction vector, otherwise the rotation is performed. The rotation calculation was done using *Rodrigues' rotation formula*, named after Olinde Rodrigues. The formula is stated as:

$$v_{rot} = v\cos\Theta + (k \times v)\sin\Theta + k(k \cdot v)(1 - \cos\Theta) \quad (10)$$

Given a vector, v, and an axis vector of which v will rotate around, k, the rotated vector, v_{rot} will be returned. This involves finding the cross product and dot product of k and v. The axis vector is determined by finding the vector perpendicular to $direction\_v$ and $next\_direction\_v$. This will mean $direction\_v$ will be rotated in the same plane as $next\_direction\_v$. However, since the dot product of perpendicular vectors equals 0, the formula can be shortened. In order to eliminate dead code, the formula actually used is:

$$v_{rot} = v\cos\Theta + (k \times v)\sin\Theta \quad (11)$$

Algorithm 1 The *update_fish* function

Input: Array of fish F be updated.
Output: The position vectors of each fish in F will be changed. The fish may receive a new direction vector.

```
call function to alter the positions of each fish
for all f ∈ F do
    initialise nextdirection vector
    perform ZOR calculations
    if no fish detected in ZOR then
        perform ZOO and ZOA calculations
    end if
end for
for all f ∈ F do
    if a fish is detected in ZOO then
        add direction vector to next direction vector
    end if
    if a fish is detected in ZOR, ZOO or ZOA then
        normalise nextdirection vector
    end if
end for
```

Algorithm 2 The *move_fish* function

Input: Array of fish F to be moved, the amount of fish and the turning angle Θ.
Output: Propagates each fish by it's direction vector and keeps fish within the area.

```
for all f ∈ F do
    if nextdirection vector is non-zero then
        if the angle between the direction vector and nextdirection vector ≤ Θ then
            direction ← nextdirection
        end if
    else
        rotate direction vector towards nextdirection vector by Θ
    end if
    add direction vector to position vector
    make corrections if position vector is outside the area
end for
```

Algorithm 3 The *update_in_ZOR* function

Input: A single fish $f*$, an array of fish F, the amount of fish and the ZOR range R_r
Output: Modified $nextdirection$ vector of $f*$

```
for all f ∈ F do
    if f* is not equal to f then
        D ← distance from f* to f
        if D < R_r and f is visible from f* then
            ZOR identifier set to 1
            Update nextdirection vector of f*
        end if
    end if
end for
```

Algorithm 4 The *update_in_ZOO_ZOA* function

Input: A single fish $f*$, an array of fish F, the amount of fish, the ZOO range R_o and the ZOA range R_a
Output: Modified $nextdirection$ vector of $f*$

```
for all f ∈ F do
    if f* is not equal to f then
        if f is visible from f* then
            D ← distance from f* to f
            if D < R_o then
                ZOO identifier set to 1
                add direction of f to nextdirection of f*
            else if D ≥ R_o and D < R_a then
                ZOA identifier set to 1
                update nextdirection vector of f*
            end if
        end if
    end if
end for
```

The code for the zone calculations, given in algorithm 3 and algorithm 4, is quite straight forward. Since the ZOR has the highest priority, the ZOR function is separate to the ZOO and ZOA. A fish mustn't have any other fish within it's ZOR, before checking the other zones.

To determine if a fish is visible from the other, the blind volume is implemented. A variable called the *blind_radian_segment* is created. This variable represents the area of space behind a fish, in which it can't see any other fish. The angle between a fish's direction vector and the direction

vector between itself and another fish is compared with the *blind_radian_segment* to see if the other fish is within the blind volume.

The rest of the functions implemented involve small functions to better manage the trigonometry and vector calculations involved and functions to improve the GUI.

C. Two Species

The *update_fish* function had to be changed in order to allow two species to interact with one another. The second species is activated by a simple boolean variable, *two_species*. A check to see if *two_species* is active is done throughout *update_fish* which causes each zone function to be called four times, so that species one can do calculations with species one and species two and so that species two can do calculation with species two and species one.

Each species has different zone ranges with regards to each species, meaning that the range of a zone species one has for species one might be different to the the range species two has for species one. This allows much more varied behaviour to be found. The two ranges for each zone are stored in an array. The first value is for species one, the second for species two.

The colours are also changed to allow a user to distinguish which species is which. Species one is orange and species two is blue.

IV. Results and Evaluation

The results from several experiments will now be evaluated, starting with one species activated, and then with two species activated. An accompanying screencast will provide video of the experiments.

A. Single Species Behaviour

TABLE II
Single Species Experiment Parameters

Parameter Name	Ex1 Values	Ex2 Values	Ex3 Values
fish1_count	400	400	400
turning_angle_spec1	5	5	5
ZOR_range_spec1[0]	2	2	2
ZOO_range_spec1[0]	0	10	20
ZOA_range_spec1[0]	40	40	40

When the zone of orientation is set to zero, as shown in Fig. 4, the fish all try to move towards the fish around them causing the swarm as a whole, to have no direction. The fish move around one another, with a clear lack of structure, apart from the fish wanting to stay near each other. The behaviour can be compared to that of sharks or other large fish, as show in Fig. 5. As sharks are higher up on the food chain, they don't have to rely on other sharks as much as they are better suited to fending off predators, which allows them to move more freely and independently.

If a predator comes to close to the swarm then they would move out of they way, creating a void in the swarm, shown in Fig. 6, similar to the fish shown in Fig. 7. This behaviour can

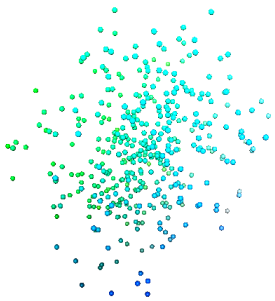

Fig. 4. Simulation results from experiment one in table two. An unorganised swarm.

Fig. 5. Sharks swarming with no coordination [15].

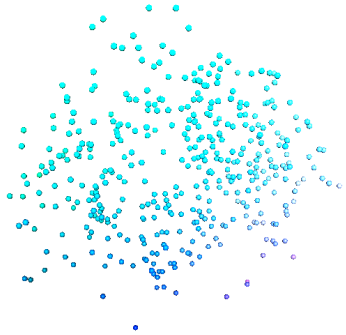

Fig. 6. Experiment 2 results. A torus shape is formed showing a lower concentration of fish in the centre. It is better observed on the screencast.

be briefly seen in fish when a predator approaches the fish as they are in a bait ball formation. The fish will move to create an opening for the predator to pass through.

Fig 8. shows how when the ZOO is high enough, the fish move in a common direction. A good example of this is shown in Fig 9. The high polarisation means that if any fish alters it's direction, due to an obstacle or predators, this change in direction will pass through the swarm and they will converge on a new common direction.

B. Dual Species Behaviour

The modified model allows for more parameters to change, to create lots of different behaviours.

Fig 10. shows behaviour similar to the bait ball structure. The cluster in the centre is the bait ball and the fish around the

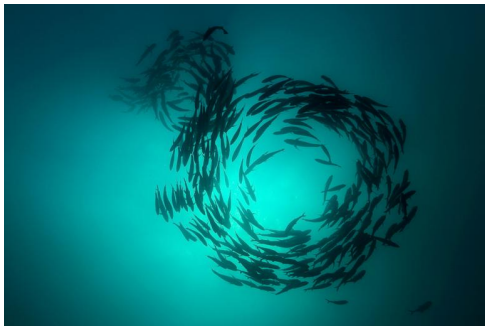

Fig. 7. A fish swarm forms a torus shape [13].

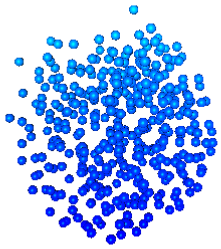

Fig. 8. Simulation results from experiment three in table two. A polarised swarm is formed.

Fig. 9. A polarised swarm of fish in reality [14].

TABLE III
DUAL SPECIES EXPERIMENT PARAMETERS

Parameter Name	Ex1 Values	Ex2 Values
fish1_count	100	300
turning_angle_spec1	4	5
ZOR_range_spec1[0]	2	2
ZOO_range_spec1[0]	10	20
ZOA_range_spec1[0]	20	30
ZOR_range_spec1[1]	8	0
ZOO_range_spec1[1]	0	0
ZOA_range_spec1[1]	30	0
fish2_count	300	200
turning_angle_spec2	8	5
ZOR_range_spec2[0]	5	30
ZOO_range_spec2[0]	0	0
ZOA_range_spec2[0]	0	0
ZOR_range_spec2[1]	1	2
ZOO_range_spec2[1]	0	10
ZOA_range_spec2[1]	40	30

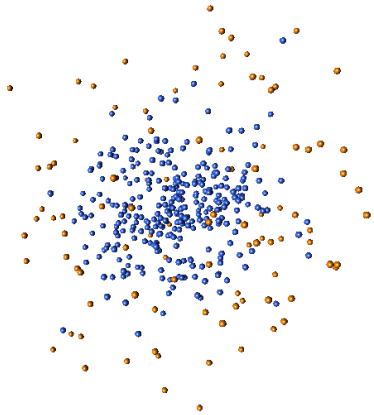

Fig. 10. Results from Experiment 1 of Table 3

Fig. 11. Fish form a bait ball, with dolphins preying on them [17].

outside are the predators. The centre fish will want to stay very close to one another and avoid the predators and the predators will dive into the bait ball. Fish closer to the centre have a greater chance of survival, although in reality, this strategy attracts more predators and causes more danger for the fish. Fig 11. showcases this behaviour in reality.

The results in Fig 13. show a polarised swarm on the left moving through the area, with the fish on the right staying well away from them, and reforming a swarm.This can be compared to a swarm of fish simply travelling through the ocean with no current interest in food, and the other fish are being very cautious and moving away with a large ZOR. Fig 14. shows a real life example of this.

There are millions of different parameter settings to choose from. The ones showcased here are chosen to show varied behaviours. Many of the settings are only slightly different to others.

V. CONCLUSIONS

The goal of this project was to simulate the movement of fish swarms, and this was accomplished to a large extent. Couzin's Model doesn't allow for fish to have different speeds, but for the purposes this project was made for, it performed well. The most difficult part of the project was implementing

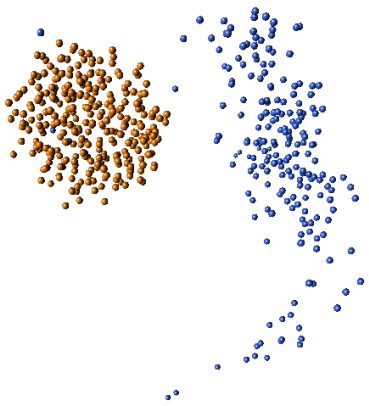

Fig. 12. Results from Experiment 2 of Table 3.

Fig. 13. The smaller fish stay well away from the larger fish [16].

the turning angle as designing a function to rotate a vector in 3D space was a challenge, however, *Rodrigues' rotation formula* reduced the complexity of this task significantly. If there was more time available, a robust method to view the scene would have been created as well as make the GUI more attractive.

Researching the movement of fish was vital for determining if the simulation accurately depicted fish swarming and footage of fish in the sea was used to see how they change their direction as a predator approaches. The way that fish move as if they were one organism was captured in the simulation, provided the parameters are reasonable. Having the ZOO and ZOA disabled would obviously create unnatural behaviour.

This project was very interesting and challenging to work on and the end result performs very well, even as a means of relaxation, due to the dynamic shapes and colours of the fish.

REFERENCES

[1] Mimus Polyglottus, "Mockingbird Tales: Readings in Animal Behavior" 2.1 Schooling in Fish, OpenStax CNX. 13 Jan 2011, Available at http://cnx.org/contents/0a9f2c0b-893f-48b4-98e9-a85af3aa4e45@5.1.

[2] Junji Takasago Photography

[3] Nicholas C. Makris, Purnima Ratilal, Deanelle T. Symonds, Srinivasan Jagannathan, Sunwoong Lee, Redwood W. Nero, "Fish Population and Behavior Revealed by Instantaneous Continental Shelf-Scale Imaging", Science, Volume 311, pp. 660-663, 2006,

[4] Viscido S, Parrish J, Grunbaum D, "Individual behavior and emergent properties of fish schools: a comparison of observation and theory", Marine Ecology Progress Series, Volume 273, pp. 239-249, 2004.

[5] Iain D. Couzin, "Collective cognition in animal groups", Trends in Cognitive Sciences, Volume 13 Issue 1, pp. 36-43, 2009.

[6] Stuart J. Russell, Peter Norvig, "Artificial Intelligence: A Modern Approach", (2nd ed.), Chapter 2, Pearson, 2003.

[7] https://en.wikipedia.org/wiki/File:Simple_reflex_agent.png, File:Simple reflex agent.png, 14/4/2016.

[8] Michael Wooldridge, "An Introduction to MultiAgent Systems", John Wiley & Sons. pp. 366, 2002.

[9] Tams Vicsek, Andrs Czirk, Eshel Ben-Jacob, Inon Cohen, Ofer Shochet, "Novel Type of Phase Transition in a System of Self-Driven Particles", Physical Review Letters, Volume 75, pp. 1226-1229, 1995.

[10] Craig Reynolds, "Flocks, herds and schools: A distributed behavioral model", Computer Graphics, Volume 21, pp. 25-34, 1987.

[11] Eva M. Navarro-López, "Advanced Algorithms II", Complex networks and collective behaviour, Lecture 7, 2015.

[12] Iain D. Couzin, Jens Krause, Nigel R. Franks, Simon A. Levin, "Leadership by numbers", Nature, Volume 433, pp. 513-516, 2005.

[13] http://news.nationalgeographic.com/content/dam/news/photos/000/389/38910.ngsversion.1441330227311.adapt.768.1.jpg, 25/4/2016.

[14] http://piximus.net/media/26656/fish-tornado-5.jpg, 25/4/2016.

[15] http://i.huffpost.com/gen/1487775/images/o-SHARKS-facebook.jpg, 28/4/2016.

[16] http://clikhear.palmbeachpost.com/wp-content/uploads/2013/06/first-for-CH4.jpg, 28/4/16.

[17] http://www.destination-scuba.com/images/baitball.jpg, 28/4/16.

[18] Xin-She Yang, "Nature Inspired Cooperative Strategies for Optimization", Studies in Computational Intelligence, Volume 284, pp. 65-74, 2010.