Modeling the Learning Ability of Fish in an Artificial Fish Simulation using Reinforcement Learning

G. M. T. C. Galahena[1], S. P. Wimalaratne[2]
[1,2] University of Colombo School of Computing
Colombo, Sri Lanka,
tcg@ucsc.cmb.ac.lk[1], spw@ucsc.cmb.ac.lk[2]

Abstract

Fish display a considerable amount of learning skills in activities like foraging and defense (ex: locations and quality food patches, areas where certain predators are in, etc.). But most of the existing models of fish behavior do not simulate the learning patterns of fish. It makes those models less realistic. This research tries to fill that gap by creating a model with learning ability which is more similar to the actual behavior of the fish. The main focus of this research will be on the learning involving in foraging and defense of the fish. The system will be a comprised of multiple agents to represent fish and each agent will act individually. The senses and the locomotion abilities of the agents in the simulation will be generalized representations of actual fish. And the learning will be simulated using machine learning algorithms.

1. Introduction

Behavioral studies of fish have been carried out in both Animal Cognition and Computer Science fields. In animal cognition, many experiments have been done in order to find the learning pattern of fish and the factors which contribute to those learning patterns. In computer science, especially in the fields of artificial life and artificial intelligence there has been a number of researches conducted in order to find better ways of simulating the behavior of fish.

Fish are the most ancient form of vertebrates which have been living on earth for 500 million years [10]. Because of that, early scientists considered their behavior as a series of fixed behavior patterns released when exposed to appropriate stimuli. Nevertheless, over the last few decades, researchers realized that fish exhibit a rich array of sophisticated behavior and that learning plays a pivotal role in the behavioral development of fish. For an example, fish perform a considerable amount of learning when they are engaged in activities like foraging and defense [1][2]. They learn things like locations and quality food patches and areas where certain predator is in. They can even rank pray and match probability of finding food in certain locations.

If we turn towards computer science, we can see that simulating behavior of the fish is a popular subject in the fields of Artificial Life and Artificial Intelligence. In both of these fields, common behaviors of fish like swarming are being studied and simulated using a number of techniques.

There are a number of applications for these simulations. The main uses of the simulations are games and other entertainment systems such as Virtual Aquariums. Addition to that these simulation are used for researches in areas like animal cognition and biology.

2. Background

Researches in Artificial Life and Virtual Reality have developed many models and applications to simulate the behavior of various animals. Especially, a number of methods and algorithms have been developed in simulating collective behaviors of animals such as fish, birds and insects. They are mostly used in game development, special effects in movies, entertainment (ex: virtual aquariums), etc.. These researches are mostly focused on maintaining the realism and obtaining the optimum performance of the simulations.

There are three main types of approaches in modeling animal behavior.

A Mathematical models

Early studies of behavior used mathematical models in simulating and understanding the particular behaviors like fish schooling. Most often these models use simple rules in guiding individual animal.

The 'boids' computer program created by Craig Reynolds in 1986 is one of the earliest programs that uses these kind of rules [3]. It uses three rules to simulate the behavior of each fish (Figure 1). They are,

- Separation - avoid crowding neighbors (short range repulsion)
- Alignment - steer towards the average heading of neighbors
- Cohesion - steer towards the average position of neighbors (long range attraction)

With these three simple rules, the flock moves in an extremely realistic way, creating complex motion and interaction that would be extremely hard to create otherwise.

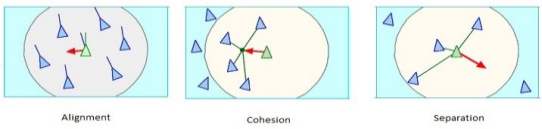

Figure 1: The Three Rules of Boids Model

B Evolutionary models

Evolutionary computing a subfield of artificial intelligence which involves in continuous optimization and combinatorial optimization problems [5]. The difference of these algorithms is that they are mostly influenced by biological mechanisms of evolution. These methods are also used in simulating animal behavior since they can be used to study the evolution of behaviors such as swarm in certain animals.

Mostly these algorithms use genetic algorithms. A genetic algorithm (GA) is a search heuristic that mimics the process of natural selection proposed by Charles Darwin [6]. It finds solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover. In simulations, these algorithms allow models of animals to undergo an artificial evolutionary process through a number of generations and observe the behavior changes in the simulation. These simulations have been used to investigate a number of hypotheses on the evolution of animal swarming behavior such as the selfish herd theory,[7] the predator confusion effect,[8] and the dilution effect [9].

C Agent-Based Methods

In agent based approach each individual in the population is represented by an intelligent autonomous agent. This agent can simulate the actions such as movements, communication foraging, etc. of the particular individual. These individual agents behave independent of other agents while interacting with other agents. The collective behaviors such as swarming and schooling will be emerged through the contribution of actions of each individual agent.

A notable research in simulating fish which uses this particular approach is the research done by Terzopoulos and Tu [4]. It contained a dynamic biomechanical muscular movement model, photo-realistic texture mapping, accurate sensory abilities, a model of desires, and a decision tree based action selection mechanism.

When we consider mathematical models and evolutionary models, it is difficult to use them in tasks such as simulating learning. But agent based models can be used easily in such tasks. Other than that, this approach can be used to model each agent individually that improve the variability of different individual characters. So, agent based models will be the best approach for this particular research.

3. Analysis and Design

The fish show a considerable amount of learning abilities in foraging and in predatory defense [1][2]. When considering foraging fish can learn to identify food through stimuli and also rank them according to the reinforcement they get. When considering defense the fish can identify and avoid the areas which the predators are in.

The system consists of two components 'Agent' and the 'Environment'. The agent is the representation of a fish in the virtual environment and the environment is the virtual space containing obstacles, food, predators, etc.. The agent takes environment data from its sensors and act according to them. Figure 2 shows the abstract architecture of the simulation.

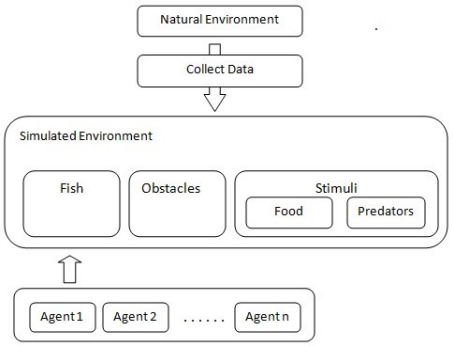

Figure 2: Abstract Architecture of the Simulation

a. Fish (The Agent)

The agent is the representation of a fish in the simulation. Since that it has the ability to simulate the behavior of a fish. Figure 3 will show the overview of an agent.

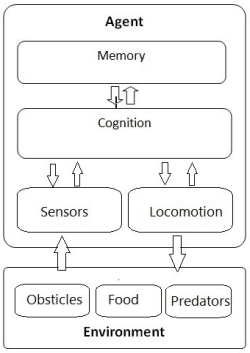

Figure 3: Overview of an Agent

An agent contains several modules that simulate different aspects of the fish.

1) Locomotion

This module controls the movements of fish. The movements which are simulated would be movements towards food, movements away from predators, swarming and collision detection.

The boids algorithm is used in the simulation of schooling behavior [3]. It provides a simple yet realistic behavior pattern to the agents. But if more complex behaviors are required another algorithm can also be plugged into this module. It uses three rules to simulate the behavior of each fish.

Collision avoidance of this model consists of two major components as steer to avoid local collisions with neighbors, and steer to avoid collisions with obstacles. The range of perception of the fish is used as a threshold to detect potential collisions with obstacles and other agents.

2) Sensors

This module simulates the sensory organs that a fish used to take input from the environment. These senses can be visual and chemical since they are the generic senses, which most of the fish have. Figure 4 shows the field of vision of an artificial fish [4].

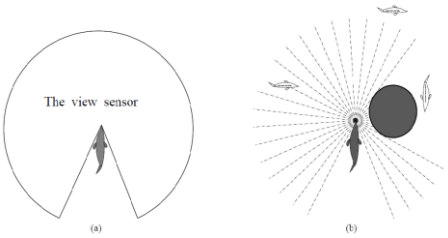

Figure 4: The field of vision of an artificial fish [4]

3) Cognition

This module is in charge of making decisions and learning. This module takes information from memory, sensor module and takes decisions and control locomotion module according to them. Also, this module learns by the reinforcements it gets and stores the new knowledge in the memory module.

An agent has to take decisions on two aspects, foraging and Defense. In each aspect the agent has to do both learning and making decisions according to what it learns.

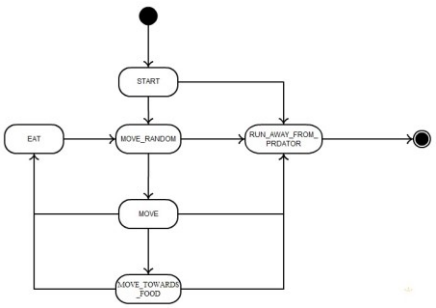

Figure 5: The state machine of an agent

An agent selects its actions according to a finite state machine. The actions of the agent are performed through a loop until the simulation is over or until the agent is destroyed. Figure 5 shows the state machine of an agent.

The functions of each state are as below,

- START – Initial state. When the simulation starts, all the agents are in this state.
- MOVE – Move according to the learned data towards the known food locations. An agent sets into this state when its hunger level is raised above a threshold value.
- MOVE_RANDOM – Move in a school. An agent sets to this state when its hunger level came below the above mentioned threshold. Although the direction of the school is random, it tent to avoid the known predator areas and move around food areas with higher probability. To achieve this probability of choosing between a totally random move and a move from learned data is set to a value decided by experimenting.
- EAT – Simulate the consumption of food. An agent sets into this state when it a food patch appeared in its field of perception. The hunger level is reduced in each time unit that an agent is in this state.
- RUN_AWAY_FROM_PREDATOR – Move away from a predator. An agent sets into this state when a predator appeared in its field of perception.
- MOVE_TOWARDS_FOOD - Move towards a food patch when it appeared in the field of perception of an agent.

Learning is applied in every state of the fish. Temporal difference learning algorithm is used to simulate the learning ability of the fish. It is specifically used for this task because it can be adopted into performing a similar learning process to an organism like a fish. It is also an algorithm majorly influenced by theories of animal learning studied by psychologists [15].

Temporal difference learning (SARSA) is a prediction method. It has been mostly used for solving the reinforcement learning problem. Reinforcement learning allows software agents to automatically determine the ideal behavior within a specific context, in order to maximize its performance. Simple reward feedback is required for the agent to learn its behavior; this is known as the reinforcement signal [13]. Fig 6 shows the action feedback loop of reinforcement learning.

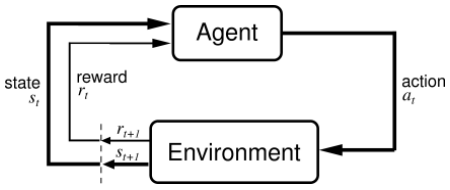

Figure 6: Feedback loop of reinforcement learning [11]

In reinforcement learning, temporal difference learning methods can be used to estimate value functions. The Value Functions are state-action pair functions that estimate how good a particular action will be in a given state, or what the return value is for that action expected to be. Value functions are denoted by V^{Π} (the value of a state under policy Π) [12].

If the value functions were to be calculated without estimation, the agent would need to wait until the final reward was received before any state-action pair values can be updated. Once the final reward was received, the path taken to reach the final state would need to be traced back and each value updated accordingly. This can be denoted as,

$$V(s_t) \leftarrow V(s_t) + \alpha[R_t - V(s_t)] \qquad (1)$$

Where s_t is the state visited at time t, R_t is the reward after time t and is a constant parameter. On the other hand, with TD methods, an estimate of the final reward is calculated at each state and the state-action value updated for every step of the way. This can be denoted as,

$$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)] \qquad (2)$$

Where r_{t+1} is the observed reward at time t+1.

4) Memory

This model contains the data structures that are required to store and retrieve data. The data are stored by the cognition module which is generated by the learning process.

To apply the learning algorithm to this particular problem, the environment is divided into 50 cells, each with a size of 20x20x20 units (Figure 7). The objective of this is to create a location to store a utility value for each location that an agent is in. So the cognition module can update these utility values according to the learning algorithm when it learns and use them to calculate the best direction it should take when it is traveling.

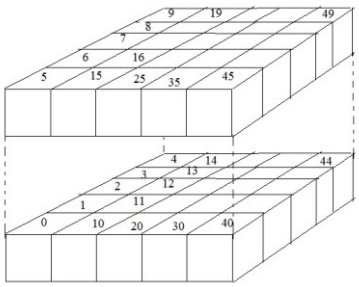

Figure 7: Cell distribution of the environment

The cells aren't given just a single utility value. A value is given for each cell for each time unit in the time cycle. So the value distribution of the cells will be changed through the time cycle also and give an agent a chance to learn about food and predator locations changing through time.

b. Environment

The environment is the module where the fish are interacting with. This module contains the locations and properties of the obstacles, stimuli, predators and other agents. The environment will be modeled as a grid to make it easier for the agents to keep a presentation of the stimuli, predator or obstacle locations. The grid size will be decided depending on the realism of the simulation and the performance of the simulation.

The stimuli contain the position, color and the reinforcement that it gives to the fish. Also the stimuli will appear in the environment within a specific time period. The above properties of the stimuli will be modeled to match the conditions in the real life to maintain the realism of the simulation.

4. Implementation

Implementation of this simulation is created using OpenSteer [16] which is an open source library designed for simulating steering behaviors of autonomous characters like human, animals, other types of creatures or vehicles.

The basic vehicle implementation provided by the toolkit is used to simulate the movement of an agent (fish). First a given number of agents will be initialized with a random position. Then a vehicle will be created for each agent with the same position as the agent. Then at each time step the agent status will be updated and each vehicle position will also be updated according to the agent's new position. Figure 8 will show how agents are represented in the simulation.

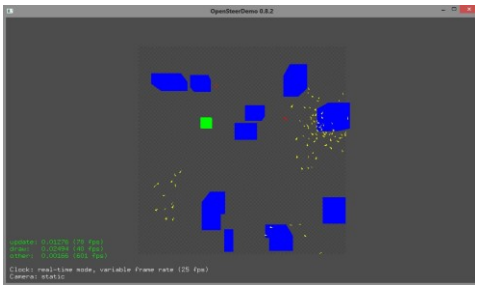

Figure 8: Prototype application

The area of the environment is the weight and the height of 100x100 pixels and the height is 40 pixels. Obstacles in the environment are represented by cubes which can be placed on the floor of the environment to make different settings similar to the natural environment.

The food is represented by smaller cubes, place on the floor of the environment. They have different colors and reinforcement values according to their color. Predators are also represented using vehicle implementation provided by the toolkit. They have random movement around specified locations and within a certain boundary. Figure 9 shows the movement area of a predator.

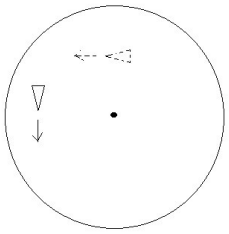

Figure 9: The area which a particular predator moves

Figure 8 shows the overall simulation. Figure 10 shows a moment where the fish are gathered around food patch and figure 11 shows the side view of the simulation.

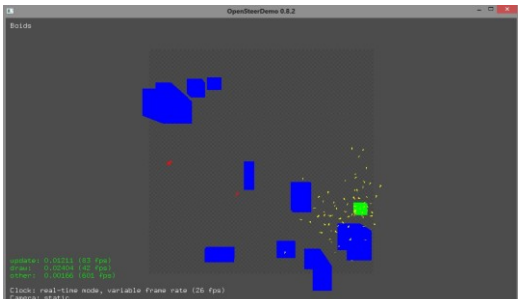

Figure 10: Prototype application – Fish are gathered around a food patch

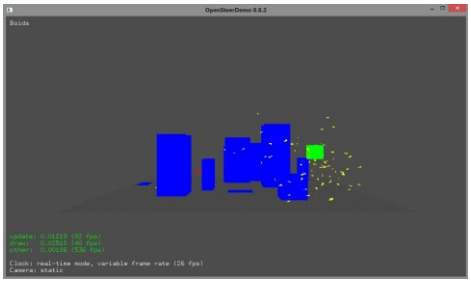

Figure 11: Prototype application - Side view

5. Evaluation

The evaluation of this kind of a research can be categorized into two sections as performance evaluation and user evaluation. Evaluation of this particular project mainly concerns about the performance evaluation because of the lack of expertise in this domain and the time constraints. The performance evaluation is carried out to evaluate the behavior simulation model by changing the conditions such as the number of agents, the number of food patches and the number of predators. The objective is to identify areas that required optimization to reduce memory consumption and execution time.

a. Performance Analysis Methodology

Entire application cannot be effectively used for the performance evaluation. Therefore, only few selected functions are considered here and those functions are selected according to their impact on the simulation. The performance analysis concerns about main functionalities of the application such as initialization, drawing and updating. This simulation application is profiled using Very Sleepy 0.82 [14] which is a C/C++ CPU profiler for Windows systems. All the measurements are taken from a machine with configurations of Intel Corei5, 2.3 GHz processor with 4GB RAM.

Performance analysis is conducted under three sections,

i. Analyses performance by changing the number of fish.

- Initially the number of fish is set to 50, number food is set to 1 and the number of predators are set to 2. The analysis is focused in three main functions known as environment() which is used to initialize each agent, food and predator draw() which handles OpenGL functions and finally the update() function.
- The performance test is conducted separately for samples with 200, 210, 250, 300, 350 and 500 hornets.
- The application inclusive time for above mentioned functions are measured in each sample.

ii. Analyses performance by changing the number of food locations.

- Initially the number of fish is set to 200, number food is set to 1 and the number of predators is set to 2. Then the analysis focus in three main functions environment() conduct initialization, draw() function and update() function.
- The test is conducted separately for samples with 1to 10 food locations.
- The application inclusive time for above mentioned functions are measured in each sample.

iii. Analyses performance by changing the number of predators.

- Initially the number of fish is set to 200, number food is set to 1 and the number of predators is set to 1. Then the analysis focus in three main functions environment() conduct initialization, draw() function and update() function.
- The test is conducted separately for samples with 1to 10 predators.
- The application inclusive time for above mentioned functions are measured in each sample.

Furthermore, since profiling cannot be conducted for more than 500 agents, frame rates for updating and drawing are also considered in order to evaluate the performance of the application.

b. Analysis

The application inclusive time for each function are considered for this analysis. This gives the time spent in the specific function and its children without the time spent in the kernel. The application exclusive time, which the individual time spent in the function without its children, was not considered since cohesion of functions.

Even though the application runs well up to 1000 fish, it was hard to take measurements for samples of more than 500 fish while profiling. Therefore, in performance analysis 18 measurements are taken for six samples (200, 210, 250, 300, 350 and 500 fish) and plotted the best fitting curve by using the curve fitting tool provided with the MATLAB application.

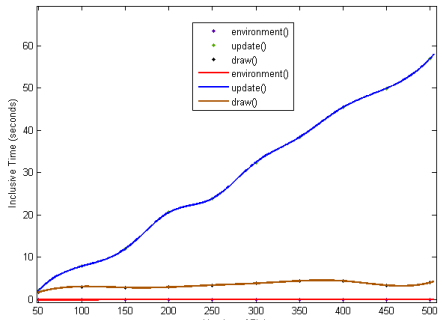

Figure 12: Inclusive times vs. the number of fish.

The graph in figure 12 illustrates the inclusive times that corresponds to the number of fish. It includes the measurements for the three functions which mentioned earlier. As illustrate in the graph, time spent on the initialization (environment() function) and draw() function doesn't exhibit a significant variation up to 500 fish. But the update() functions show a significant change with the increase of the number of fish.

Although the draw() function doesn't increases much it shows slight fluctuations along with the number of fish. On the other hand the update() function also shows similar fluctuations while it is increasing. This may have been due to the fact that 'MOVE' state of a fish gives a one of two outcomes according to a given probability. These outcomes may require different time spans to execute.

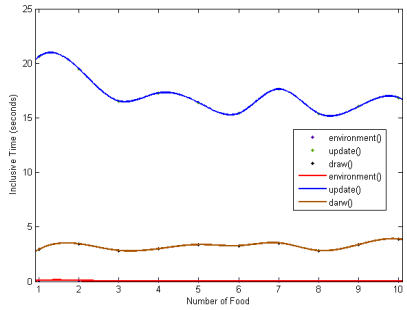

Figure 13: Inclusive times vs. the number of food patches.

The graph in figure 13 shows the inclusive times which correspond to the number of food patches. As you can see from the graph the initializing doesn't take much time to execute and also doesn't have a much increase or decrease. The draw() function also doesn't have a much increase or decrease in inclusive time it is still larger than the time taken by the initialization.

On the other hand the update function has much larger inclusive time, and also it has an overall slight decrease. The reason for this decrease is that the increase of the number of food patches increases the probability of finding a food patch by a fish within its range of perception without moving far. This makes it less likely to choose the best path from its learned data. So it saves the time spent on quarrying its memory. The fluctuations in the both update() and draw() functions are due to the same reason mentioned in the number of fish vs. inclusive time test.

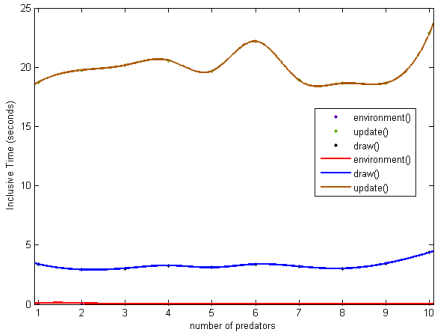

Figure 14: Inclusive times vs. the number of predators

The graph in the figure 14 shows the inclusive times that correspond to the number of predators. This also doesn't show much increase or decrease in initialization and draw() function. But the update() function shows significant fluctuations along the way. This may have the same reason described in the above case. But in here the predators are also moving and it makes the graph to take much more random variation than the above case.

If conclude the result that obtained from analyzing the profiling report, update() function is the most critical function that consume more CPU time. Therefore, the update() function has to be optimized in order to achieve better performance.

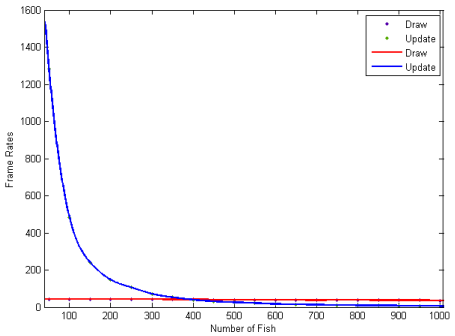

Figure 15: Frame Rates vs. the number of fish.

The second performance test was conducted by considering the frame rate varies according to the change in number of agents. The graph in fig 15 shows the results of the frame rate evaluation. These results also indicate that the performance variation is more tide with the update process than the drawing process. So from this frame rate evaluation, it can be concluded that updating process is the part that need to be optimized in order to get a better performance.

6. Conclusion

Designing a realistic model for simulating the behavior of the fish is a complex task which requires considering many aspects. For a moderately realistic simulation fish can be modeled as automatons which acts on a simple set of rules. But to create a more realistic simulation, higher order behaviors of fish such as learning should be included in the behavior model. In order to fulfill that requirement this paper presents a behavioral model which can simulate learning ability of fish. But considering the time and resources available for the research the scope is limited to the simulation of the learning involved in foraging and defense of the fish only.

The learning aspect in this model is designed using the temporal difference learning. It is a reinforcement learning algorithm which provides a less computationally complex learning mechanism which was ideal for this research. The algorithm was able to learn the locations of food and predators and guide the agents in finding food and escaping predators. Also the algorithm was implemented way that it can also learn the appearance of the food and predators relative to time. Addition to that, the agents are provided with a mechanism for them to find new food locations by performing a certain percentage of their movements randomly. These movements are modeled using Boids model which is a well-known flocking behavior model.

When implementing the application, there were many problems that had to be solved. Finding a better tool for implementing the application was the major issue that involved in the research and had to improve graphics related and mathematical skills in order to carry out the research. With the time constraints, the implementation is done as a prototype by using simple objects to represent fish, obstacles, predators and food patches in the virtual environment. The prototype was built using OpenSteer libraries (open source) that provide basic functionalities for creating a multi-agent based simulation. Even though using simple objects may impact on the user attraction, it has been somewhat successful in giving some idea about their behavior pattern.

When consider the evaluation of this research, finding expertise to conduct a user evaluation was involved as the main issue. Therefore, only performance analysis was carried out to measure the performance of important functions. Finally end of this research; we have to conclude that it has not achieved the goal 100% due to the unavailability of the sufficient amount of experimental data to make the simulation more accurate. So the simulation only demonstrates an approximation of the learning ability of the fish. Further improvements can be used to solve this issue.

7. Future Work

The learning skills of the agents in this model only show an approximation of the learning skills of the actual fish. So the issue that needs the attention is to find sufficient experimental data in order to modify the model to give more accurate output. Other than that the model consider only the generic behavior of the fish. So, looking for ways in which the model can be improved in order for it to simulate particular species of fish is also a task that should be looked into in the future.

In addition to the foraging and defense fish show learning in other types of behavior such as Mate Choice. So extending the model to simulate other types of behavior beside the foraging and defense, is also important to consider when conducting further research. Also, the communication between the fish which includes social learning is also need to be included in future versions of this model.

Acknowledgment

I would like to thank all who gave the guidance and support to make this research successful.

References

[1] Warburton K, Hughes, R , Learning of Foraging Skills by Fish , Fish Cognition and Behavior, 2006, p.10 – 35.

[2] Kelley J. L, Magurran A. E, Learned Defences and Counterdefences in Predator–Prey Interactions , Fish Cognition and Behavior, 2006 p.36 – 58.

[3] Reynolds, C. W. (1987) Flocks, Herds, and Schools: A Distributed Behavioral Model, in Computer Graphics, SIGGRAPH '87 Conference Proceedings, 2006, p. 25-34.

[4] Tu, X. Artificial Animals for Computer Animation: Biomechanics, Locomotion, Perception, and Behavior. Ph.d thesis, University of Toronto, 1996.

[5] D. Simon. Evolutionary Optimization Algorithms. Wiley, 2013.

[6] Mitchell, Melanie (1996). An Introduction to Genetic Algorithms. Cambridge, MA: MIT Press.

[7] Olson RS, Knoester DB, Adami C (2013). "Critical Interplay Between Density-dependent Predation and Evolution of the Selfish Herd". Proceedings of GECCO 2013.

[8] Olson RS, Hintze A, Dyer FC, Knoester DB, Adami C (2013). "Predator confusion is sufficient to evolve swarming behaviour". J. R. Soc. Interface.

[9] Tosh CR (2011). "Which conditions promote negative density dependent selection on prey aggregations?". Journal of Theoretical Biology

[10] Brown C, LalandK, Krause J, Fish Cognition and Behavior , Fish Cognition and Behavior, p.1 –9, 2006

[11] McClelland, James L. , Explorations in Parallel Distributed Processing: A Handbook of Models, Programs, and Exercises. 2 ed , Vol . 2014

[12] "Reinforcement Learning". RetrievedJanuary , 2014 Available: http://www.cse.unsw.edu.au/ ~cs9417ml/RL1/tdlearning.html

[13] "Reinforcement Learning". RetrievedJanuary , 2014 Available: http://reinforcementlearning.ai-depot.com/Intro.html

[14] Mitton, R , "Very Sleepy". Retrieved January , 2014 Available: http://www.codersnotes.com/sleepy/

[15] Andrew G. Barto (2007) Temporal difference learning. Scholarpedia, (11):1604

[16] C. W. Reynolds,"'OpenSteer: Documentation." [Online] Available: http://opensteer.sourceforge.net [Accessed: November 13, 2014].