Assignment 2

SYSC 5104

Modeling the Formation of School of Fish

Mohammad Etemad

Department of System and Computer Engineering

Carleton University

November 2014

**Part I - Concept**

**1.1. Introduction**

Different animals that live in groups choose to do so for different reasons. Some may depend on each other for hunting prey, some may choose such life style to make finding a mate easier, some may enjoy one another’s company, and some find safety in numbers. Some types of fish normally live in groups to find safety in numbers.

Normally, schools of fish act as they do so that they can escape a predator. By forming a large group and confusing the predator, the fish can escape form this deadly danger. Almost all individuals in a school of fish are oriented in the same direction and normally move at the same speed. They predict their next move based on several different factors, some of which are how their surrounding fish move and how close is the predator [1]. Such behavior has also been observed in schools of fish once they want to feed or migrate. Hubbard et al. report that the school members may have a constant speed for some time, but then remain in the same position for a while. Also they mention that in a migration motion, the school normally stays close to the warm and cold water boundary [2].

Typically, once a fish group is formed they stay at the same position unless an event happens. This paper will focus on the formation of a fish school and will not consider any external factors that may occur such as: coming close to a predator, the boundaries of hot and cold water, or any other external event that may disrupt their normal motion.

**1.2. Model concept and basic definitions**

In this assignment, a model will be presented that shows how individuals of fish will form a school of fish. The model consists of three separate layers. Layer 1is the real environment where the fish will be swimming in. The fish will decide in which direction they will move based on the random direction assigned to them. To be able to represent and use the directions East, North, West, and South, each of these directions have been assigned a value. Table 1 shows the mapping between the directions and the integer value.

Table 1. The mapping between directions and integer values between 1 and 4

|  |  |
| --- | --- |
| Integer Value | Corresponding Direction |
| 1 | East |
| 2 | North |
| 3 | West |
| 4 | South |

To start the simulation, a number of fish have randomly been placed on the first layer. Figure 1 shows the first layer with randomly generated fish placed on the layer. These initial values will be considered throughout this report and the results that follow will be based on this initial state. The environment has been considered to be a 10 by 10 matrix.

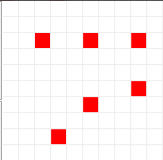


Figure 1. Initial state of the simulation

For the fish to start moving, the fish will look at their randomly generated direction which is stored in layer 2. Figure 2 shows the randomly generated numbers for the first step.

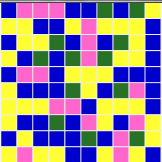
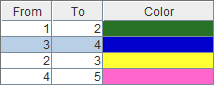
  


Figure 2. Initial state for layer 2 showing the randomly generated directions

These values will be stored in the second layer so in every step each fish will take the value on top of him and move based on the direction assigned to him.

The fish will move at random until a point that they see another fish in their field of view. Figure 3 demonstrates the field of view of fish. Once a fish sees another fish in its field of view, the fish will be attracted to one another no matter what their randomly assigned direction is. Since the field of view has been defined by 1 cell, to avoid collision the fish on the left will remain in its position while the fish on the right will move towards it. It is the same for vertical moves, the fish on the top will remain in its position while the fish below it will move towards it. On top of all of this, there is a rule that overrides all of these rules. If a fish is connected to a school, it will not move and the other free fish will move towards it, no matter if it has to move left or right, up or down.

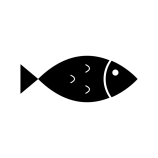


Figure 3. Field of view for fish

As demonstrated in figure 3, the fish cannot see the cell behind it. I order to be able to consider this in the model, an orientation for the fish should be defined. Layer 3 in the model represents the orientation of the fish. Basically, the orientation of the fish is the direction the fish was heading to one time step ago. So layer 3 will store a copy layer 2s data by a time delay of one time step. In order to make it easier to read, layer 3 will only keep the data if the random generated direction has been used for a fish. Figure 4 shows a screenshot of layers 1, 2, and 3 at a certain time.

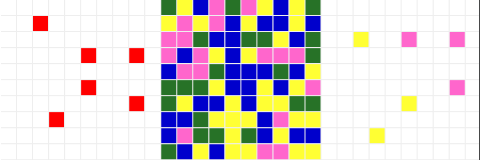


Figure 4. Layers 1, 2, and 3 at a certain time

**Part II – Formal Specification**

**2.1. Layers**

This model uses three different layers of size 10 by 10 each. The first layer will be the layer of interest where the fish are located. The second layer will contain randomly generated numbers which are mapped to directions. The third layer will contain a number showing the orientation of the fish. Figure 5 shows an overview of the layers. In the code, each zone corresponds to a layer.

Layer 3:   
Orientation of the fish

Layer 2:   
Direction of the fish for next move

Layer 1:   
Position of the fish itself

Figure 5. Demonstration of the layers of this model

**2.2. Neighbors**

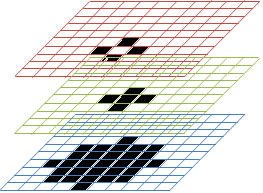
­The neighborhood considered for this model is an extended Von-Newman neighborhood. For the motion of the fish, the fish can only move in the Von-Newman neighborhood. Meaning that in each step, the fish can either go forwards, up, backwards, or down. The extension is used for collision avoidance and priority moving. The final neighborhood shape is demonstrated in figure 6.

Figure 6. Neighborhood shape

**2.2. Movement Rules**

The movement of the fish can be categorized in to two types of movement:

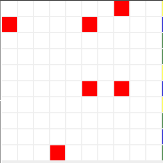
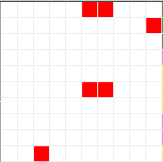
* Moving towards other fish
* Moving randomly

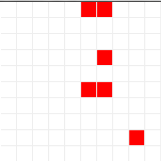
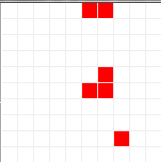
All fish are assumed to move randomly using their assigned random direction unless the fish can see another fish. If two fishes are in each other’s field of view, they will get together. The rules describing such behaviour are as follows. These rules will be executed in priority order:

1. If there is a fish on your left and you are not attached to a school, move left
2. If there is a fish on your right which is attached to a school, and you are not attached to a school, move right
3. If there is no fish on your left or right, and there is a fish above you and you are not attached to a school, move up.
4. If there is no fish on your left or right, and there is a fish below you which is attached to a school and you are not attached to a school, move down.

Some visual examples of the applied rules are demonstrated in figure 7. It should be noted that, in case of collision the priority is given to the fish moving horizontally. Also, moving left has more priority over moving right, and moving up has more priority over moving down. So if at the same time, two fish try to go to a cell, the priority will be given to the fish moving left. The other fish will remain in its spot to avoid collision.

For layer 2, random numbers will be generated in each step. Layer 3 will keep a copy of the previous number for each cell that has a fish below it (in layer 1).

   
State A State A+1

   
State B State B+1

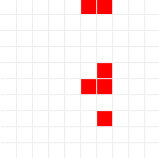
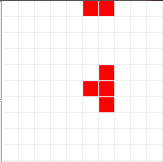
   
State C State C+1

Figure 7. Some sample states showing execution of rules

**2.2. Implementation of Rules**

The rules applied for the first layer are the main rules of this simulation. The first rule in this layer will cause a delay in the execution of the rules to follow. This is done so that the first official rule can be executed after layers 2 and 3 are populated.

[layer1]

%Delay execution of layer 1 rules

rule : { (0,0,0) } 100 { time < 200 }

The next set of rules will keep the position of the fish constant if the fish is attached to a school or if there is another fish that has priority for moving to a conflicted cell. The macros used in these set of rules are also presented.

%keep position if attached

rule : { (0,0,0) } 100 { (0,0,0) = 5 and (#Macro(Attached)) }

rule : { (0,0,0) } 100 { (0,0,0) = 0 and ((#Macro(VerticalAttached)) or (#Macro(HorizontalAttached))) }

rule : { (0,0,0) } 100 { (0,0,0) = 5 and (0,2,0) = 5 and (0,-2,0) != 5 and (#Macro(FarRightCellNotAttached))} %if you are the far left cell

rule : { (0,0,0) } 100 { (0,0,0) = 5 and (2,0,0) = 5 and (-2,0,0) != 5 and (#Macro(NoHorizontalMove)) and (#Macro(NoHorizontalMoveLowerFish)) and (#Macro(LowerFishNotAttached)) } %if you are the upper cell and no left or right move available

rule : { (0,0,0) } 100 { (0,0,0) = 5 and ((#Macro(UpperCellDontMove)) or (#Macro(LowerCellDontMove))) }

%Macro evaluating if any of the surrounding cells are attached to a group of fish  
#BeginMacro(Attached)  
((0,1,0)=5) or ((0,-1,0)=5) or ((1,0,0)=5) or ((-1,0,0)=5)   
#EndMacro  
  
%Macro checking to see if the fish can move vertically or if they are already in a school  
#BeginMacro(VerticalAttached)  
((((-1,0,0) = 5) and (((-1,1,0) = 5) or ((-1,-1,0) = 5) or ((-2,0,0) = 5))) and (((1,0,0) = 5) and (((1,1,0) = 5) or ((1,-1,0) = 5) or ((2,0,0) = 5))))   
#EndMacro

%Macro checking to see if the fish can move horizontally or if they are already in a school  
#BeginMacro(HorizontalAttached)  
((((0,-1,0) = 5) and (((1,-1,0) = 5) or ((-1,-1,0) = 5) or ((0,-2,0) = 5))) and (((0,1,0) = 5) and (((1,1,0) = 5) or ((-1,1,0) = 5) or ((0,2,0) = 5))))  
#EndMacro

%Macro evaluating to see if the right cell is not attached  
#BeginMacro(FarRightCellNotAttached)  
((0,2,0) = 5) and ((1,2,0) != 5) and ((-1,2,0) != 5) and ((0,3,0) != 5)  
#EndMacro

%Macro evaluating if no horizontal move is possible

#BeginMacro(NoHorizontalMove)

((0,-2,0) != 5) and ((0,2,0) != 5)

#EndMacro

%Macro evaluating if no horizontal move is possible for its lower cell

#BeginMacro(NoHorizontalMoveLowerFish)

((2,-2,0) != 5)and ((2,2,0) != 5)

#EndMacro

%Macro evaluating if the lower fish is not attached

#BeginMacro(LowerFishNotAttached)

((2,1,0) != 5) and ((2,-1,0) != 5) and ((3,0,0) != 5)

#EndMacro

%Macro avoiding collision if a fish tries to go left and a fish tries to go down - priority is given to horizontal moving fish

#BeginMacro(UpperCellDontMove)

((1,0,0) = 5) and (((1,1,0) = 5) or ((1,-1,0) = 5))

#EndMacro

%Macro avoiding collision if a fish tries to go left and a fish tries to go up - priority is given to horizontal moving fish

#BeginMacro(LowerCellDontMove)

((-2,0,0) = 5) and ((-1,1,0) = 5) and ((-1,-1,0) = 5)

#EndMacro

After this set of rules, the movement rules are implemented in the code. As explained before, the moves are executed on priority basis. The macros in the conditions take care of collision detection and priority execution. The code and the macros for these set of rules are as follows:

%Rules for fish separated by 1 cell

rule : { (0,1,0) } 100 { (0,1,0) = 5 and (0,-1,0) = 5 and (#Macro(RightCellNotAttached)) and ((0,1,2) != 1 or time <= 200) } % move left

rule : { (0,-1,0) } 100 { (0,-1,0) = 5 and (0,1,0) = 5 and (#Macro(RightCellAttached)) and ((0,-1,2) != 3 or time <= 200) } % move right

rule : { (1,0,0) } 100 { (-1,0,0) = 5 and (1,0,0) = 5 and (#Macro(NoHorizontalMoveLowerFish)) and ((1,0,2) != 4 or time <= 200) } %move up

rule : { (-1,0,0) } 100 { (1,0,0) = 5 and (-1,0,0) = 5 and ((#Macro(NoHorizontalMoveUpperFish)) and ((#Macro(DownCellAttached)) or (#Macro(HorizontalMoveLowerFish)))) and ((-1,0,2) != 2 or time <= 200) } %move down

%Macro evaluating if its right cell is not attached and can move  
#BeginMacro(RightCellNotAttached)  
((0,1,0) = 5) and ((1,1,0) != 5) and ((-1,1,0) != 5) and ((0,2,0) != 5)  
#EndMacro

%Macro evaluating if its right cell is attached and cannot move  
#BeginMacro(RightCellAttached)  
((-1,1,0) = 5) or ((1,1,0) = 5) or ((0,2,0) = 5)   
#EndMacro

%Macro evaluating if no horizontal move is possible for its lower cell  
#BeginMacro(NoHorizontalMoveLowerFish)  
((2,-2,0) != 5)and ((2,2,0) != 5)  
#EndMacro

%Macro evaluating if no horizontal move is possible for its upper cell  
#BeginMacro(NoHorizontalMoveUpperFish)  
((-2,-2,0) != 5) and ((-2,2,0) != 5)  
#EndMacro

%Macro evaluating if the fish below can make a horizontal move  
#BeginMacro(HorizontalMoveLowerFish)  
((1,-2,0) = 5) or ((1,2,0) = 5)  
#EndMacro

%Macro checking to see if its lower cell is already in a school  
#BeginMacro(DownCellAttached)  
((1,1,0) = 5) or ((1,-1,0) = 5) or ((2,0,0) = 5)   
#EndMacro

Finally, the last sets of rules are applied if the fish is alone and will be moving randomly. All rules have collision detection and are executed if a collision does not occur. The last rule will clear the cell if none of the rules are applicable to it. The random move rules will take the direction from the layer above them and act accordingly. These rules and their related macros are as follows:

%move randomly if alone

rule : { (0,1,0) } 100 { (0,1,1) = 3 and (0,1,0) = 5 and (#Macro(NoFishAroundLeft))} %move left

rule : { (1,0,0) } 100 { (1,0,1) = 2 and (1,0,0) = 5 and (#Macro(NoFishAroundUp))} %move up and collision avoidance

rule : { (0,-1,0) } 100 { (0,-1,1) = 1 and (0,-1,0) = 5 and (#Macro(NoFishAroundRight))} %move right and collision avoidance

rule : { (-1,0,0) } 100 { (-1,0,1) = 4 and (-1,0,0) = 5 and (#Macro(NoFishAroundDown))} %move down and collision avoidance

rule : 0 100 { t } %clear current cell

%Macro evaluating if there is no fish around so that it can move left

#BeginMacro(NoFishAroundLeft)

((2,1,0) != 5) and ((-2,1,0) != 5) and ((0,-1,0) != 5) and ((0,3,0) != 5) and ((1,1,0) != 5) and ((-1,1,0) != 5) and ((0,2,0) != 5)

#EndMacro

%Macro evaluating if there is no fish around so that it can move up

#BeginMacro(NoFishAroundUp)

((1,-2,0) != 5) and ((1,2,0) != 5) and ((-1,0,0) != 5) and ((3,0,0) != 5) and ((1,-1,0) != 5) and ((1,1,0) != 5) and ((2,0,0) != 5)

#EndMacro

%Macro evaluating if there is no fish around so that it can move right

#BeginMacro(NoFishAroundRight)

((-2,-1,0) != 5) and ((2,-1,0) != 5) and ((0,1,0) != 5) and ((0,-3,0) != 5) and ((-1,-1,0) != 5) and ((1,-1,0) != 5) and ((0,-2,0) != 5)

#EndMacro

%Macro evaluating if there is no fish around so that it can move down

#BeginMacro(NoFishAroundDown)

((-1,2,0) != 5) and ((-1,-2,0) != 5) and ((1,0,0) != 5) and ((-3,0,0) != 5) and ((-1,1,0) != 5) and ((-1,-1,0) != 5) and ((-2,0,0) != 5)

#EndMacro

The code for the other layers is demonstrated below. The second layer contains the direction of the fish for the next move, and the third layer contains the orientation of the fish.

[layer2]

%random direction generator

rule : {randInt(3)+1} 100 { t }

[layer3]

%orientation of the fish

rule : { (0,0,-1) } 100 { (0,0,-2) = 5 }

rule : 0 100 { t }

[nothing]

rule : { (0,0,0) } 100 { t }

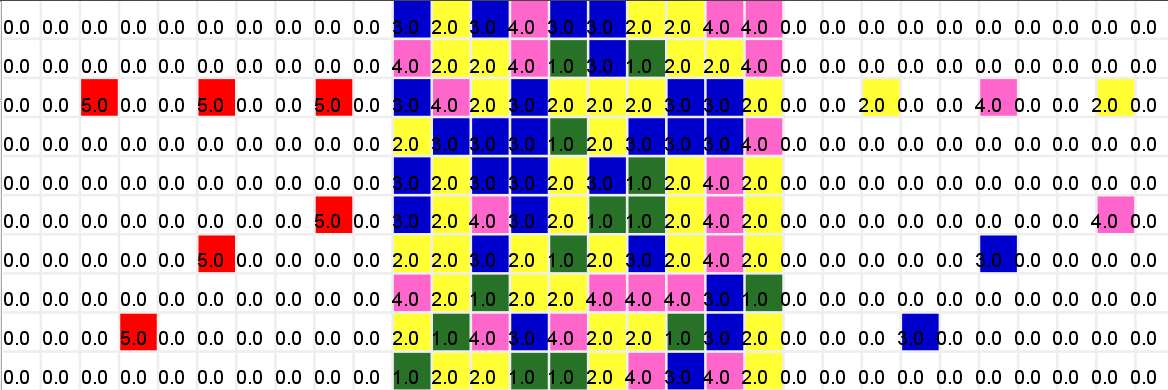
**Part III – Simulation Results**

To demonstrate the simulation results, different test scenarios will be presented. In all cases the conditions are as follows. Please note that the fish are assigned a value of 5 and are shown in red. Number 1 represents right direction, 2 is up, 3 left, and 4 is down with colors Green, Yellow, Blue, and Pink respectively:

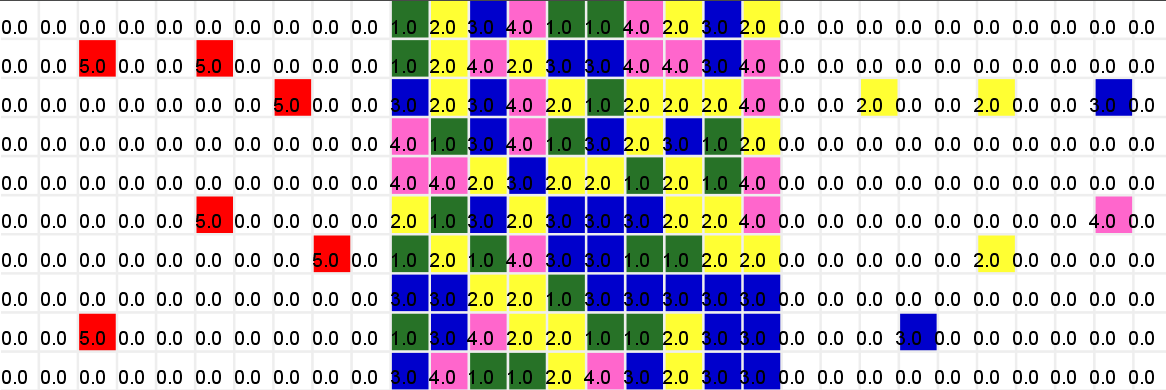
1. **Fish widely apart**

If the fish are widely spread apart, they will all have random movement until they get to see one another. The results are demonstrated below.

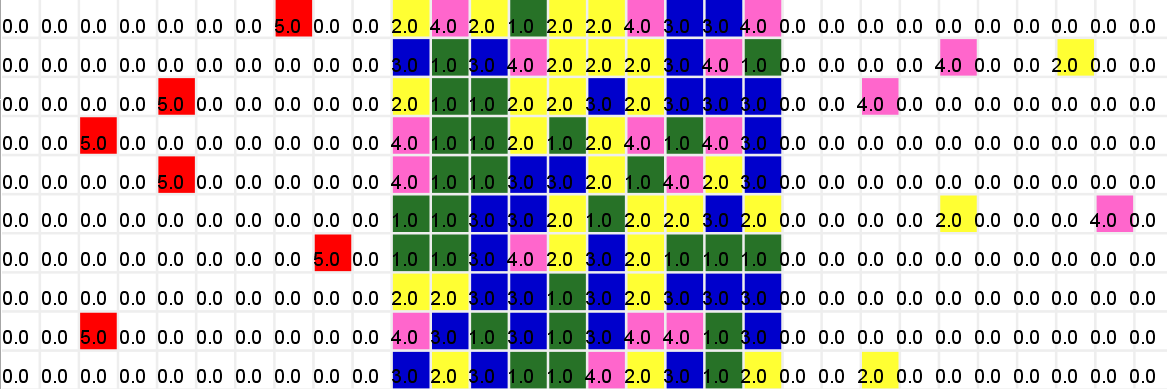
Initial state (after two time steps):



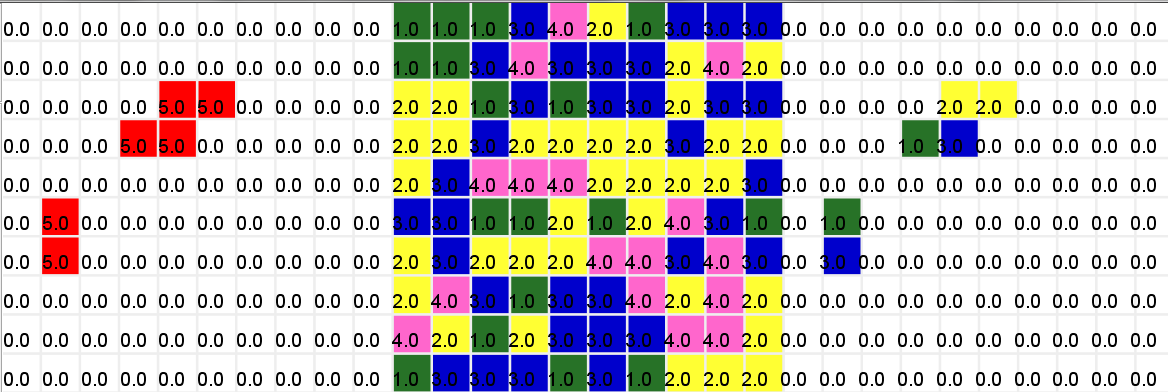
2nd state:



5th state:



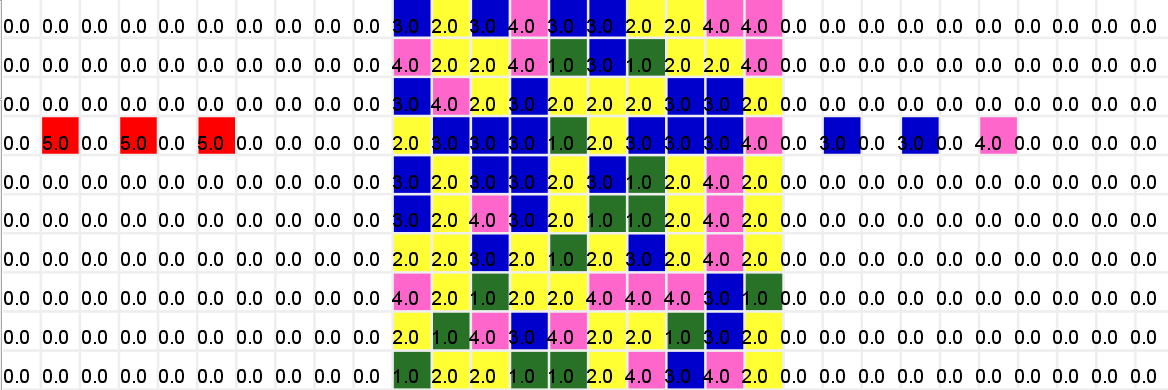
Final state:



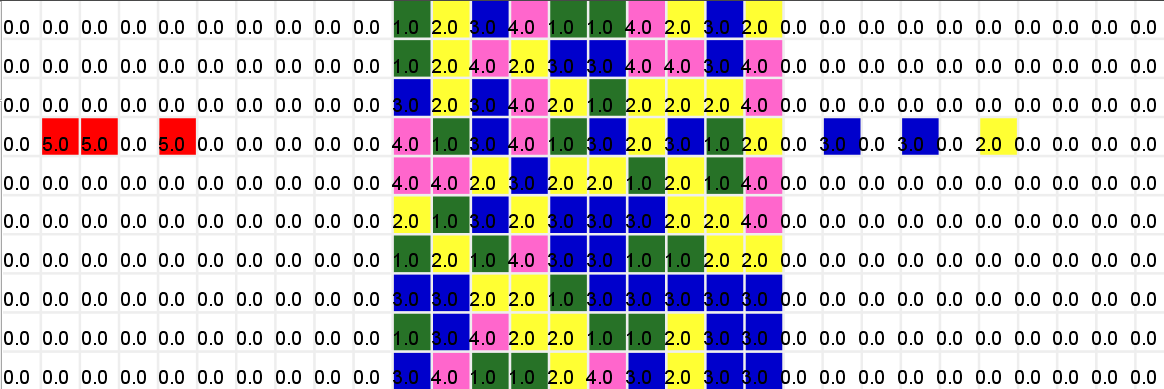
1. **Fish in one line**

If the fish are visible to one another they move towards each other using the rules defined.

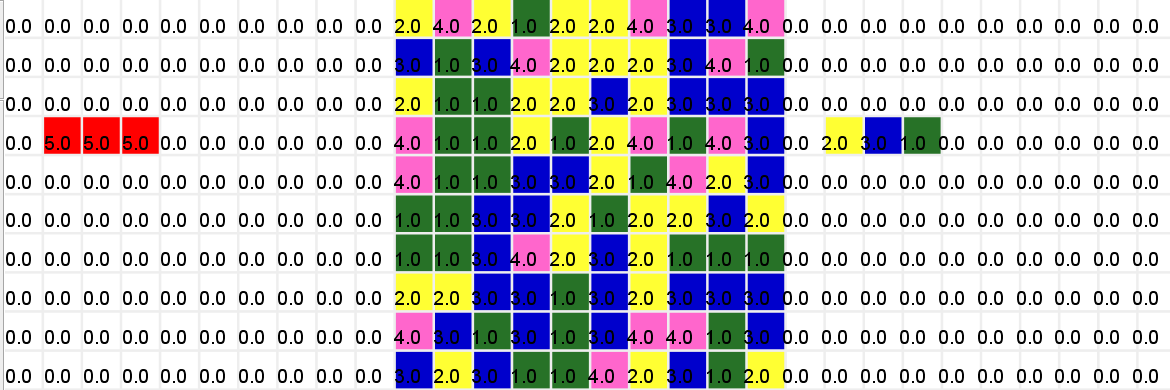
Initial state:



2nd state:



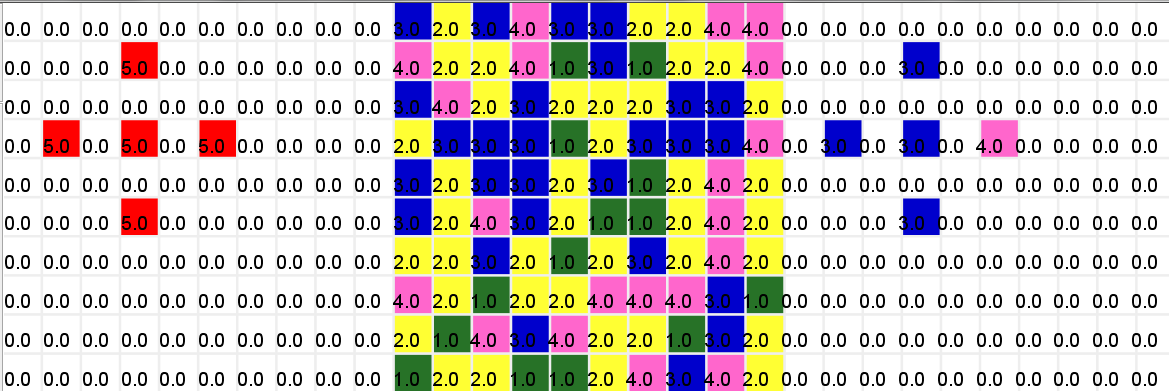
Final state:



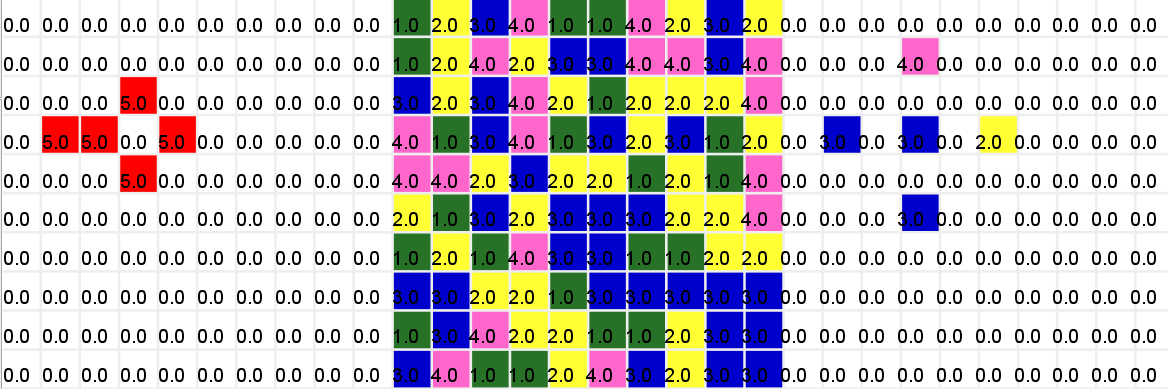
1. **Fish forming a cross**

If the fish are forming a cross with one cell in between, they should move towards one another, but collision avoidance comes into play.

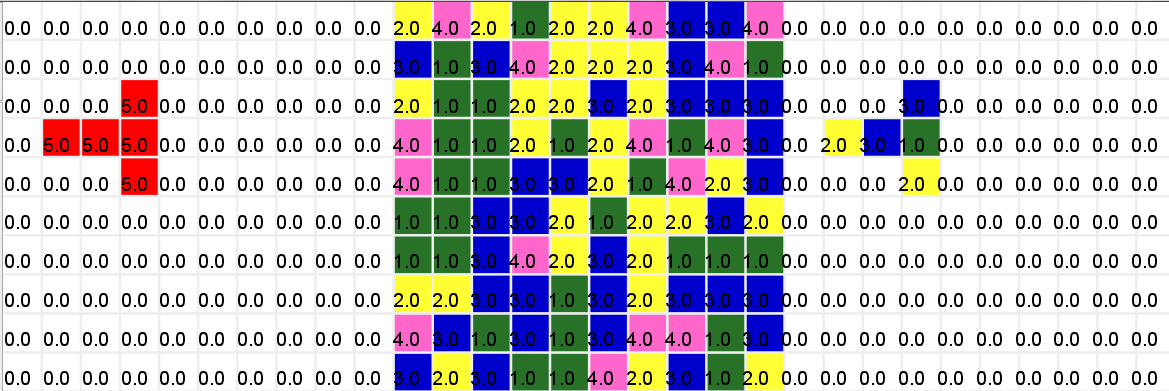
Initial state:



2nd state:



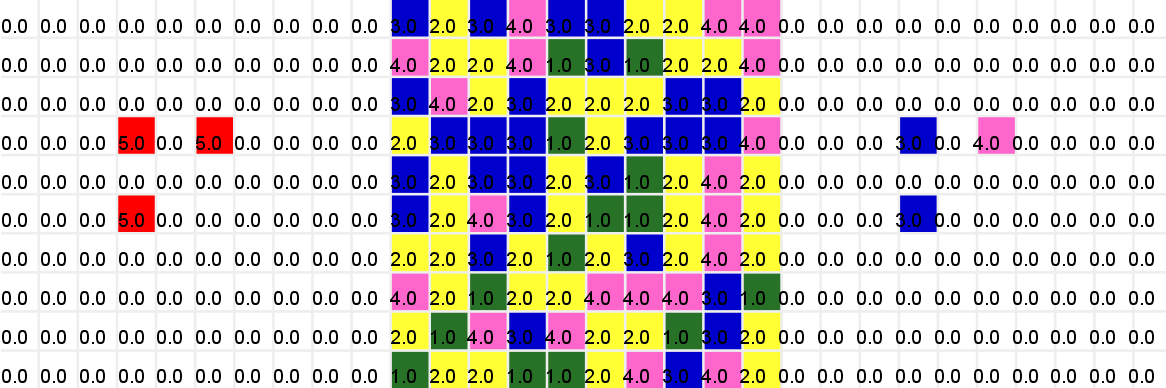
Final state:



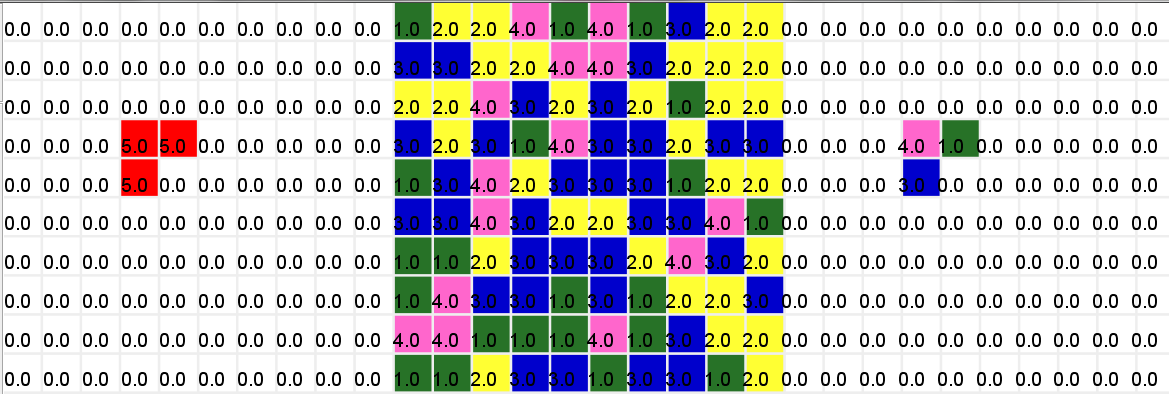
1. **Fish forming a right angle**

If the fish form a right angle, the top left fish should keep its position allowing other fish to come to him.

Initial state:



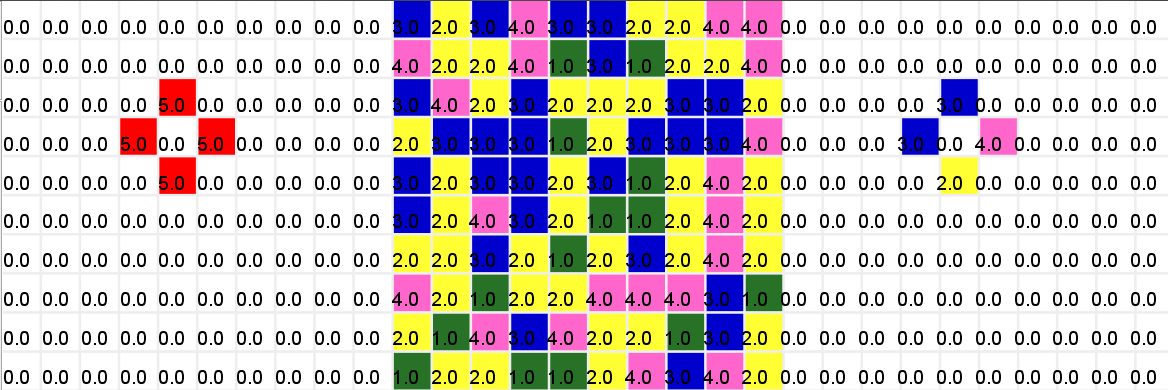
Final state:



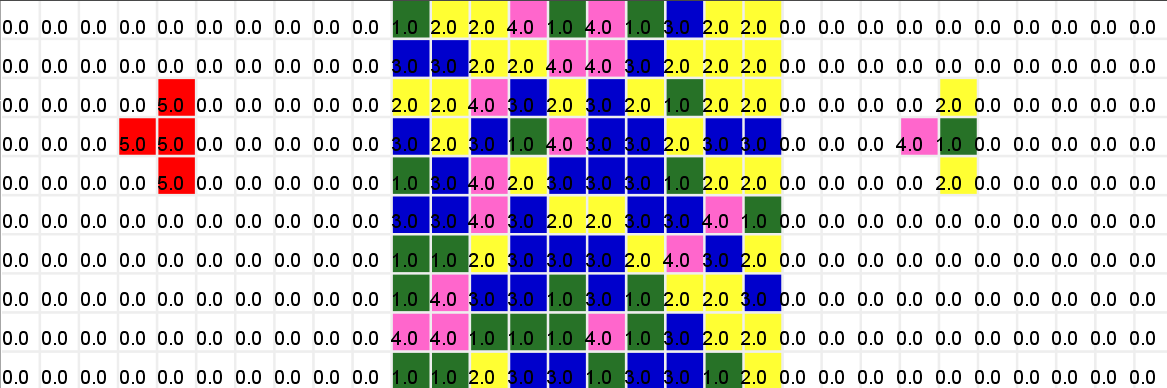
1. **Fish forming a diamond**

In this case, priority for moving is essential.

Initial state:



Final state:



**Future Work**

The current model implements the formation of schools of fish. Using the reference papers, a predator can be introduced to the system and different escape algorithms can be implemented.

**References**

|  |  |
| --- | --- |
| [1] | S. Hubbard, P. Babak, S. T. Sigurdsson and K. G. Magnússon, "A model of the formation of fish schools and migrations of fish," *Ecological Modelling,* vol. 174, no. 4, pp. 359-374, 2004. |
| [2] | M. Zheng, Y. Kashimori, O. Hoshino, K. Fujita and T. Kambara, "Behavior pattern (innate action) of individuals in fish schools generating efficient collective evasion from predation," *Journal of Theoretical Biology,* vol. 235, no. 2005, pp. 153-167, 2005. |