[Return to APCS Home](http://docs.google.com/index.html)

|  |  |
| --- | --- |
| **Category:** General  **Writer(s):** James Y. | **Marine Biology Case Study**  Okay, so the Marine Biology Case Study is less fun than a barrel of monkeys, actually it's probably the least fun you've ever had in your life. I know it was for me. However, you need to know it to get a decent score on the AP test at the end of the year. I'm going to explain the main functions and objects of the program without actually discussing the code. Using my explanation you should be able to look at the code and understand exactly what it's doing. My suggestion is to start at the FishSim.cpp main programming and work your way back to the other sections of the program as you go. So, without further adieu, I present the least entertaining thing ever, but it's got to be done.  The Marine Biology Case Study has 7 main objects that get called from the main program. FishSim.cpp contains the main programming of the entire program. The first thing it does is it opens the fish.dat file. The information in the fish.dat file is what creates the environment. All the information is stored in two columns, the first is for row information, and the second for the columns. The first row of information gives the dimensions of the environment. All rows after the first are fish locations to be initialized when the environment is created. This means that the first numbers in each of the two columns must be the largest because you can't initialize fish outside of the environment. For example, the following would initialize a 5X5 matrix with four fish in their given locations.  5 5  0 1  2 4  3 2  4 4  The environment takes this information like any other document, first it finds the dimensions of the world, and then it calls addFish for each fish it reads. If you want to change the number of fish in the program or the size of the environment, then you would do it here. After the environment has been initialized, display.show sends it to the screen of the user. The FishSim.cpp then asks the user how many steps should be executed. For each number of steps the user asks for, the program calls simulate.step, and then it displays it on the screen again.  Simulate obviously does a whole lot of work, because these fish aren't moving on their own, and it is the only call made before it is displayed again. Simulate calls the Environment.AllFish function that creates a list of all the fish in an explicit list called fishlist. This list arranges all the fish in order from left to right, then top to bottom. So, it searches the first row one column at a time from left to right, then it moves on to the next column and so forth. It is this order that controls the order that the fish will get to move. By changing the AllFish function you can change the order in which the fish move. So, if your environment looked like this:  A \_  B C  Then, the only possible result after one step would be:  B A  C \_  This is because the first fish in fishlist is A, then B, and finally C. Since the fish can't move at the same time or be in the same location, this is the only possible result.  I seem to have gotten ahead of myself because the fish won't do anything until you tell them to, but at least you know the order they will move in. After Simulate gets the fishlist, it takes each fish and tells it to move. Move is a function in the Fish object that calls Position by using the EmptyNeighbors function. This function sends the location of the fish to the Position object which then calls the functions North, South, East, and West. These four functions take the position that the fish knows it's in (Yes, there is a function in the fish object that allows a fish to store its own position, so why do you have to ask Position? So you can get neighboring positions and the positions of surrounding fish as well.) and returns the four adjacent positions. If the call to position had more options than N, S, E, and W, then it would change the way that the fish moved. Say for example NW was an option, or SE, there would be more places that the fish could move.  After the locations of each neighbor of the fish is returned it sends these four locations to the Neighborhood object. This object will find out whether there is a fish in these four locations by using the IsEmpty function. It will then return which of the positions are open to the fish. If there are any positions available, RandGen is called and chooses which of these positions to move to.  This new location is changed in every place that a fish's location can be stored. It sends the new location to Update in the environment object so that it can update the myworld positions. Since the fish objects also know their own locations, those must be updated as well.  After everything has been updated, Simulate takes the next fish and repeats the process. After each fish has had a chance to move, the first step is over and the new locations of all the fish have been changed in myworld. Display then shows myworld on the screen of the user. This is considered to be one step. After each step has been made the program is done.  Now that you know how the program works, you can go back and read the code yourself and follow along the different paths that are called. You can also go into fish.dat and change the environment settings or go into the code itself and mess around with whatever you want. By now you should know where to go to change the environment, the order fish are called, the directions they can move, and anything else that is done in the program. Good luck and have fun experimenting with the possibilities.  Multiple Choice Questions:  1) If you wanted to change the order in which fish were called, what part of the program would you have to change?  a) Display.Show  b) Fish.Move  c) Environment.AllFish  d) Do what with who?  2) If you wanted to change the directions in which fish could move, what part of the program would you most likely change?  a) Position.EmptyNeighbors  b) Fish.Move  c) Environment.AllFish  d) Equal Rights for everyone!!!  3) What controls the number of steps taken?  a) fish.dat  b) FishSim.cpp  c) Adolf Hitler  d) Adolf Hitler if he was still alive and was the user of this program  4) Where are the fishes identifications stored after a fish has moved from its starting spot?  a) Dog Tags  b) Position.Emptyneighbors  c) Fish.ID  d) Environment.Update  5) From the following fish locations in the 2X2 matrix shown what is a possible outcome after one step has been completed?  A \_  B C  a) A C  B \_  b) B A  \_ C  c) B A  C \_  d) No movement, fish union is on strike again  6) What controls the initial size of the matrix as well as the number and locations of the fish?  a) fish.dat  b) Environment.myworld  c) Environment.AllFish  d) The user  7) What do the following lines of programming do if they were found in the fish.dat file?  9 9  5 6  0 1  2 3  a) creates a matrix with 99 square units and 6 fish  b) creates a matrix with 99 square units and 3 fish  c) creates a 9X9 matrix with 6 fish  d) creates a 9X9 matrix with 3 fish  8) What kind of list are the fish stored in before they are sent to simulate?  a) an implicit apvector  b) an explicit apvector  c) an apmatrix  d) tuna can apvector  Answers:  1) C; Environment.AllFish is where the Fishlist is created, and that is the explicit list that holds all the fish in the order that they will be called.  2) A; Position.EmptyNeighbors controls the different adjacent positions that will be returned to the Fish.  3) D; Whoever is using the program determines the number of steps taken. That's the second or third thing done in the FishSim.cpp driver program  4) C; Fish.ID is the function that allows each fish to store its own identification, this way when fish cross paths you can tell which one is which.  5) C; Since A is the first fish to move, and there's only one open adjacent position, it has to go there. Once A has moved to the right, there is an open space above B that it can fill. The space vacated by B is the only spot that C can go into.  6) A; Fish.dat is the written file that contains the two column information that tells the environment how big it will be, how many fish there will be, and where they will be located.  7) D; The first line tells you the environment will be a 9X9 apmatrix and there are 3 more lines, each of which contains the starting position of a fish, meaning there will be 3 fish.  8) B; An explicit apvector. The list they are referring to is the fishlist. The fishlist doesn't contain any of the open spaces in the matrix, making it an explicit list. If it did contain all the spaces then it would be implicit, but it's not. |
|  | [Return to APCS Home](http://docs.google.com/index.html) |