ICS 212 Homework 5

Space War

In this assignment, you'll implement a very simple space war simulation (my apologies if this is too destructive – I hope it's not. I am a big Trekkie, and it's all in fun).

● The basic purpose of the assignment is to gain experience with hierarchical constructors and destructors, and layered architectures.

● The "universe" in this assignment will be a 20x20 array, in which two opposing space ships can roam. Additionally, within this space, there may be asteroids, which can obscure the view of the two space ships. The two space ships carry photon torpedoes (I did warn you I was a Trekkie).

● The general idea is that the universe will contain a 20x20 array, each coordinate may contain one or both space ships and an asteroid (note it is okay for a cell to have both ships and an asteroid within a cell). Further, each ship has a torpedo tube, with a variable number of unspent torpedoes in reserve.

● Implement the typedef for a torpedo tube. This should be fairly simple; essentially it only has to describe how many torpedoes remain. This is an exercise, though, so please implement this as its own struct, even though it may seem excessive to put this one number in its own struct. It seems excessive because I am trying to keep the problem as simple as possible, to focus the assignment only on the layered architecture.

●Next, implement a constructor and destructor for the photon torpedo tube. The constructor should dynamically allocate a photon torpedo tube, initialize the number of torpedoes to some reasonable number, and return a pointer to the new struct. The destructor should free the memory.

●Next, you're going to need a typedef for a photon torpedo itself. This is for a photon torpedo in 'flight'. It simply needs to describe the direction in which the torpedo is traveling, e.g. up, down, left, or right. Implement the constructor and destructor for the photon torpedo.

● Finally, typedef the struct for a single coordinate in space. It should be able to contain a pointer to up to two space ships, it should have pointers for up to two photon torpedoes (i.e. in 'flight'), and it should have a way to indicate whether an asteroid is present at the given coordinate.

●Next typedef the struct for a spaceship. It's going to be really simple, too. Basically, it's going to contain nothing but a pointer to a torpedo tube. Again, I realize this is going to seem excessive. The reason it seems excessive is because I am trying to keep it simple enough that it doesn't become really confusing. Implement a constructor and destructor for the spaceship type. Note: this is going to be slightly more tricky, because the constructor for the spaceship first has to malloc()enough memory for its own struct, but then it has to *call the constructor* which you have just defined for the torpedo tube, above. Next, define the destructor for the spaceship. Observe, the spaceship destructor has to first call the destructor for the torpedo tube, above, and then it can free() the memory for its own main struct.

●Next, define the constructor for the game board, which should be a 20x20 array of the above structs. The constructor should initialize the structs, making all coordinates empty, except placing asteroids in a small number of spots (you can decide how many) and it should instantiate exactly two space ships in random locations.

● Once the universe has been instantiated, the game play should commence. This should consist of a loop. At each cycle in the loop (i.e. at each point in time):

1) Each ship should look linearly in all four directions. If it has a direct line of sight to the other ship, it will fire a photon torpedo at the other ship. Note that the ships cannot see through asteroids. Photon torpedoes travel only in a straight line once launched, and move two spaces in the array, for every unit of time. Torpedoes should fly harmlessly past asteroids, other torpedoes, and should disappear when they reach the edge of the 'universe'.

2) If a ship cannot see the other ship, then it will move one square in a random direction. (You'll have to decide what you want to do with regard to the edges of the array, i.e. wrap-around, or block movement). The ships can move up, down, left, right, or diagonally. They can also occupy the same space as an asteroid.

● Game play continues, until either a photon torpedo collides with a ship, or both ships run out of photon torpedoes. At the end of game play, you should call the universe's destructor function, and this should automatically destroy all of the dynamic memory for all data structures associated with the game.