BATTLESHIP (project 2)

Project #3 - Due: Feb 27th, 2009, at the start of class

Purpose: To give the student C++ experience working with class construction and STL classes and functions. Also the purpose of this project is to have a little fun. (Programming groups of 2 are allows – but not mandatory.)

Introduction

The basic idea of this project is to design a console version of the famous game, Battleship. (check out this link for an online version of the game - http://scv.bu.edu/~aarondf/java/battleship.html) See appendix A for the basic rules of Battleship. Your program should allow a human player to play against a computer player. Your computer player can exhibit as much intelligence as you would like to give it – the simplest approach is for your computer player to select random places on the board. (However, it would be better to choose adjacent positions when an actual hit is reported.)

Object-oriented Design

The initial time and energy should go into the careful design of class creation and interaction. (board, ships, players, etc.)

Incremental Design Process

Here are the milestones to follow on your way to a complete Battleship game. If you run out of time and cannot finish, it is best that you stop immediately and thoroughly test and hand-in your partial solution. Remember to save version of your program as you are working on it – perhaps using the following milestone as versions.

Game Setup

- o Printing the Board (will be enhanced as you add ships, features, etc)
- o Placing Ships on the Board (Manual Placement first, then later attempt computer player placement)

Game Play

- o Inputing guesses from player
- o Generating random computer guesses
- o Determine if a guess hits a ship
- o Determine if a hit sinks a ship
- o Determine if a sunk ship ends the game

The worst thing that could be done on this programming assignment is to write the entire program, only to find out that many bugs exist, and that producing a working program would be impossible in the remaining time.

Random Number Generation

(see chapter 1 for information about working with random numbers)

```
#include <time.h>

…
  int iSecret

  /* initialize random seed: */
  srand ( time(NULL) );

  /* generate secret number:  1-10 */
  iSecret = rand() % 10 + 1;
```

Computer Player Intelligence

Your final program will somewhat be judged on the intelligence of the computer player – however I am not looking for professional grade intelligence. It should do at least a little more than simply choose a random position from the non-chosen positions.

Extra Credit Options (varies from 5 to 10% - depending on the results)

1. Cool GUI – use your favorite GUI library to add a visual component to your project.
2. Added Intelligence – give additional and advanced AI to your computer player.

Appendix A. How Battleship Is Played

Introduction:
Battleship is a two player game of sea warfare. You and your opponent each place 5 ships on the board, and you then alternate "firing" torpedoes at each others ships. The game is won when one player has destroyed/sunk all of the other players ships.

Setup:
Each player has two 10x10 boards to use, the ship board and the target board. The ship board is used to initially position your ships and record where the enemy has guessed. The target board is used to keep track of where you have guessed, and what the outcome of each guess was.

To set the board up, each player must place his five ships on the ship board. The ships have different sizes and can be placed either horizontally or vertically.

The five ships are:

* Aircraft Carrier – Length 5
* Battleship – Length 4
* Cruiser – Length 3
* Submarine – Length 3
* Destroyer – Length 2

The ships should be arranged so as to minimize the chances of the opponent sinking all your ships.

For a two player game, each player would set up his own ship board. Since your opponent for this assignment will be the computer, you should set up your own board, and have the computer randomly place his five pieces (legally of course).

Playing the game:
You play the game by each player calling out positions on the board and having the opponent tell them whether the position was a "hit" or "miss". You respond with "hit" if the position your opponent called out corresponds to a position on your ship board with a ship on it, otherwise, your respond "miss". A position on the board is indicated by a row and a column. The row is a letter between "A"and

"J"and the column is a number between "0"and "9"(really 1-10, but it will be easier if you use 0-9). The process of "firing" at the opponent is repeated until all the ships of one player have been sunk. Also every time one of your ships is sunk, you must tell your opponent which ship was sunk.

Your program should take care of determining the correct response of each move such as "hit", "miss" or "you sunk my battleship". Also, since your opponent is the computer, it should automatically make a move on its turn at a position that it hadn't guessed before.

Winning:
As stated before, the game is over when one player has sunk all the opponents ships.

Possible empty seas (before setup)

	1	2	3	4	5	6	7	8	9	10
A	.	.	.	.	.	.	.	.	.	.
B	.	.	.	.	.	.	.	.	.	.
C	.	.	.	.	.	.	.	.	.	.
D	.	.	.	.	.	.	.	.	.	.
E	.	.	.	.	.	.	.	.	.	.
F	.	.	.	.	.	.	.	.	.	.
G	.	.	.	.	.	.	.	.	.	.
H	.	.	.	.	.	.	.	.	.	.
I	.	.	.	.	.	.	.	.	.	.
J	.	.	.	.	.	.	.	.	.	.

Ship placement

	1	2	3	4	5	6	7	8	9	10
A	.	.	B	B	B	B	.	.	.	.
B	.	.	.	.	.	.	.	.	.	.
C	.	.	.	.	.	.	.	.	.	.
D	.	.	.	.	.	.	.	.	.	.
E	.	.	.	.	.	.	.	.	.	.
F	.	.	.	.	.	.	.	.	.	.
G	.	.	.	.	.	.	.	.	.	.
H	.	.	.	.	.	.	.	.	.	.
I	.	.	.	.	.	.	.	.	.	.
J	.	.	.	.	.	.	.	.	.	.