**Artificial Intelligence Final Project Report - Battleship**

**The Game:**

Battleship is a game, generally played by two players, where each player has five ships: aircraft carrier, battleship, submarine, cruiser and destroyer. Each player chooses where to place their ships on the game board. Each ship takes up a certain number of spaces (coordinates) on the game board. The game board is a ten by ten grid with rows A - J and columns 1 – 10. Each player has two boards. One of the boards shows their ships. On this board, the player records the moves made by the opponent. The other board starts as a blank representation of the opponent’s ships. The players take turns listing coordinates (i.e. A1) to target on the opponent’s game board. If the move is a hit, the opponent will say so and the player records the move on his representation of the opponent’s board with the appropriate mark, generally a red peg for a hit and a white peg for a miss. If the player misses, the opponent also announces this and it becomes the opponent’s turn. If all the coordinates of a ship are found by a player, the ship is “sunk.” Once all the ships have been “sunk” on either the player’s or the opponent’s board, the game is over and the person with ships left un-sunk is declared the winner.

**Play Styles:**

There are a few different play styles that a player can use in battleship. The first and most simple is random guessing. As the name suggest, random guessing is a style in which each move made by the player is completely random. This style has no intelligent and is not very effective, however, each style of playing battleship, incorporates random guessing at some point.

The most common play style is hunt and target. In this style, the player starts off performing random moves. Once the player finds a ship and the opponent declares a hit, the player knows that above, below, left or right the rest of the ship must lie. The player then targets these areas for the next few attacks. When a second hit is found, the player should continue down the line until they miss a shot. Most players then switch to the opposite side to be sure the entire ship has been destroyed.

The next play style is known as Parity. This is similar to hunt and target, with one exception. Instead of starting with random moves, the player moves on every other coordinate (A1, A3, B2, B4, etc…). This strategy works because the smallest ship is at least two spaces, thus, if every other spot is hit, you’re guaranteed to hit every ship at least once. Once a ship is hit, the strategy moves back to the target mode as described in the hunt and target style.

The final and most advances style of play is the probability density function. This method uses an algorithm to determine the best move to make next, based on the possible locations of remaining opponent ships. The first move is random. Then, at each turn, the algorithm determines all the possible locations where a ship could fit and builds a density graph which details the likelihood of the presence of a ship at any location.

**Our Project:**

The project we chose to do is build an intelligent version of the classic game Battleship. Our goal was to create a game that is nearly impossible to beat (without using the same algorithm). We set up our game as a standard game of battleship, player verses computer. The game is set up just as described above (two boards, five ships, etc…). As stated, our goal was to create a difficult game, so the obvious choice for play style was the probability density function. This algorithm far outplays the other play styles.

**Design:**

We created our project using C# in Visual Studios.

**Results:**

We set up a test with the program to run multiple games back to back and output data determined from the games. The results showed the amount of moves the computer had to make before destroying all of the ships on the board. We ran six different tests and detailed the result. The first test was on 100 games. This test had a mean move number of 44. The median move number was also 44. The smallest amount of moves the computer made in a game was 23, while the largest was 70. The next test was to simulate 200 games. The results were very similar to the initial 100 game run. The mean and median were both 44, just as shown in the 100 game run. The smallest amount of moves decreased by one in this simulation, bringing the total down to 22. The largest also decreased by one, making it 69. The next test we ran through the simulation was 500 total games. This time the results changed slightly more than the previous run. The mean for this test was 43 total moves, one less than previously observed. The median was down to 42 moves, a two move drop. The smallest move number stayed the same at 22, whereas the largest dropped again down to 67. Although the changes weren’t significant, it seemed at this point that with the more games played, the results were lower. The next run was 1000 games. These results were the opposite of what we previously predicting, mostly increasing overall. The mean and median move numbers both increased back up to 44. The smallest number of moves in a game was the only statistic to decrease this run, decreasing by one down to 21. The largest amount of moves in a game, like the mean and median, increased back up to 70. The results were very similar to the first run, with most of the values the same, the only exception being the smallest number of moves in a game, which was smaller by two turns. Next, we ran a simulation of 5000 games. At this point we expected our results to be pretty consistent, as they have been with previous runs. The mean was, again, 44 moves. The median was down to 43 moves. The most interesting statistic we received in this run was the smallest number of moves needed for the computer to destroy all of the enemy ships. It is important to note that a perfect game (only hitting the ships from start to finish and never missing a single shot) is 17 moves. In this simulation, we had a game be completed in just 18 moves. That means the computer only missed a single shot in the entire game. This is an extremely impressive result. The largest number of moves taken stayed at 70 for this simulation. The final simulation we ran was 10,000 games. As we expected, the results didn’t vary too much in this run. The mean and median were both 43. The smallest move number was again an impressive 18 and the largest number was 71.

Overall, the results from the simulations were quite consistent. The average number of moves it took the computer to destroy all the ships stayed around 44. When comparing this to statistics found online about the number of moves it takes for a game to complete using the other styles, these results are very impressive. With the smallest number of moves being as low as 18, this gives promise that with this algorithm, a perfect game could possibly be achieved.

**Sources:**

http://www.datagenetics.com/blog/december32011/index.html