

# BATTLESHIP

## 2.2 Executive Summary

This is a fully functioning human vs computer version of the popular game battleship. The computer has been programed to guess intelligently based on previous successful hits. The program recognizes when ships are sunk and declares a winner win a player has sunk all of the opponent's ships.

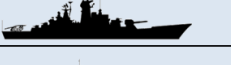

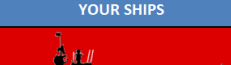
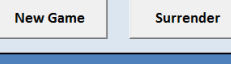






### BATTLESHIP

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





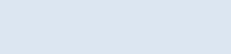






YOUR SHOTS FIRED : 17

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

#### ENEMY SHIPS



#### YOUR SHIPS



## 2.3 Implementation & Documentation

### Features

- **Random Ship Placement**
  - Random position and orientation
  - Ships are always placed completely on the board
  - Ships do not overlap
- **Making Guesses**
  - Player makes a guess by selecting a cell on the top grid
  - Guesses can only be made by selecting an empty cell
  - A red "X" appears if a ship was hit
  - A black "X" appears if it was a miss
  - After the human guess, the computer makes a guess
- **Sinking a Ship**
  - Ships retain their identity as groupings of linear cells
  - When all cells in a ship are hit, the ship is sunk
  - When a ship is sunk the corresponding picture is highlighted in red
- **Ending the Game**
  - The game ends when all of one player's ships are sunk
  - Computer ships are revealed
  - A dialog announces the winner and the number of shots taken
  - No more guesses can be made
- **Statistics**
  - Game keeps track of the number of shots fired and ships sunk
- **New Game**
  - Clears all human and computer guesses
  - Resets sunk ships
  - Randomizes ship configurations on both board
  - Resets Statistics
- **Surrender**
  - Reveals the computer ships
  - If desired the player can still make guesses
- **Computer Player Acts Intelligently**
  - Computer makes random guesses until it hits a ship
  - After a computer hits a ship it makes adjacent guesses
  - Once the computer determines the ship's orientation, it guesses along that plane
  - If the computer knows the orientation and misses before sinking the ship, it will guess on the opposite end of the un-sunk ship
  - Once the computer has hit both ends of the ship it begins guessing randomly again

## 2.4 Discussion of Learning

Through this project I became quite **fluent in VBA**. In previous projects I began to learn VBA, but since projects were short and spread out I didn't have sufficient experience to master topics. This project had sufficient length to provide lots of VBA exposure. This project came out to 25 functions/sub routines and 428 lines of code.

In programing a battleship mastered **2D arrays** in VBA. I created four 2D arrays - one for each set of ship placements and one for each set of guesses. I learned how to iterate through theses arrays and check for complex conditions.

I felt pleased with the **ship placement code** that creates random ship configurations. This code attempts to place a ship in a random orientation and position. The for loops ensure that the ship do not extend off the edges of board. Before the new ship is placed, the program checks the current board configuration to make sure that the new ship does not overlap an existing ship. If the ship overlaps, the place\_ship() function is called again until a valid placement is found.

I am also proud of the **intelligent computer guessing**. This code composes about a third of the code base. When the computer hits a ship, the computer will fire at adjacent cells. Once the computer determines the orientation of the ship it will preceed to fire sequentially along that line of cells. If the computer misses it will jump to the other side of the ship and fire sequentially until it misses. The figure on the right shows the order of guess the computer will make after hitting a ship.

In this project I learned a lot about **formatting cells and styles** with VBA. I had to do a lot of formatting changes to indicate to the user the state of the game. I worked for a while to **change the color of the ship**

**silhouettes** when they were sunk; from online forums it looks like there aren't strong image recoloring options available in VBA. Eventually I settled on giving the silhouettes transparent backgrounds and highlighting the background cells in red when the ship was sunk.

Lastly, I saw the importance creating many **simple functions** and commenting them well. As my code began to grow, large functions quickly became confusing. I had to work to extract as many simple functions as I could from the monolithic ones. As I did this I found the code much more understandable and much easier to reuse. This especially came in handy since the computer player could reuse many of the functions made for the human board.

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

## 2.5 Assistance

I did not receive substantial assistance on this project