1305ENG – Engineering Programming Project Report

Author: Ethan Watman, S5289668

Introduction

The project 'Battle Ships' is based on the original board game. The aim of the game is to sink your opponent's ships first resulting in the last player standing winning. In this project, I implemented modifications that both added realism to the shooting mechanism and added further complexity to the strategy and mind games involved. The addition of realism was achieved through the implementation of wind which has a random variety of strengths and directions which is complete. The last addition of stealing opposing players' ships and swapping them with your own is also complete. This project was developed, compiled and run within the program Code Blocks'.

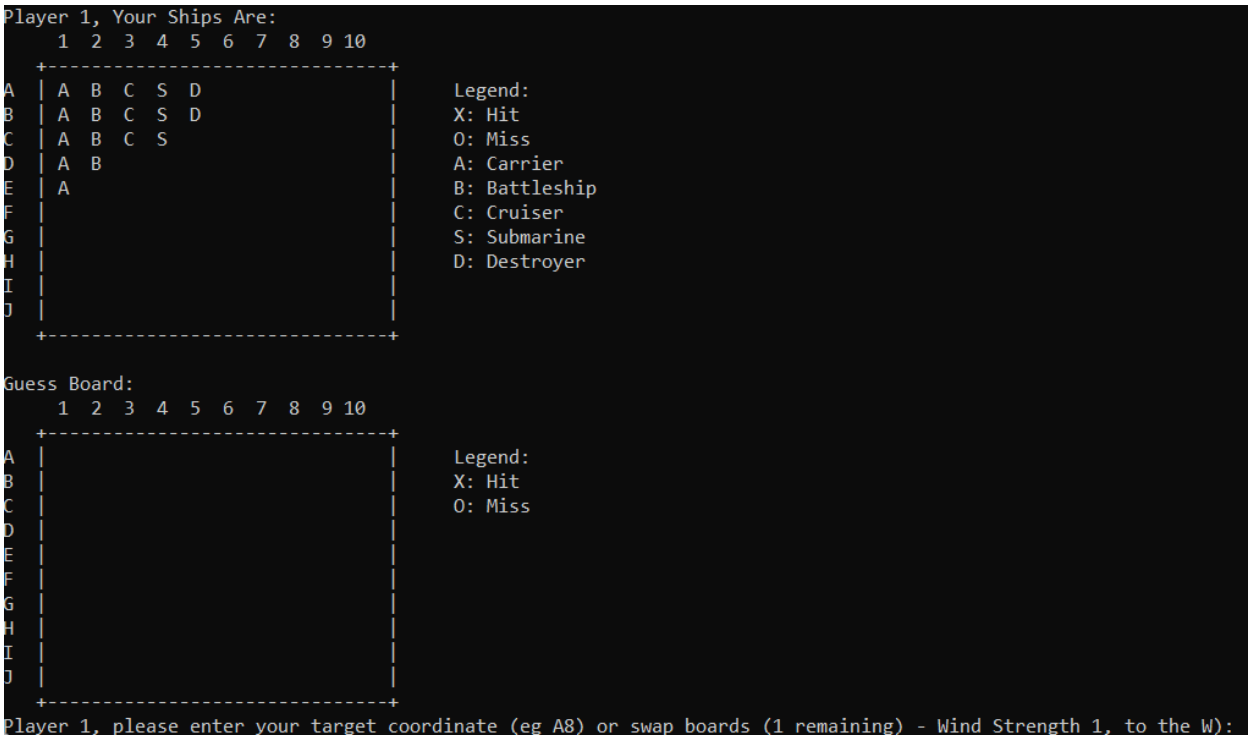
Program Features

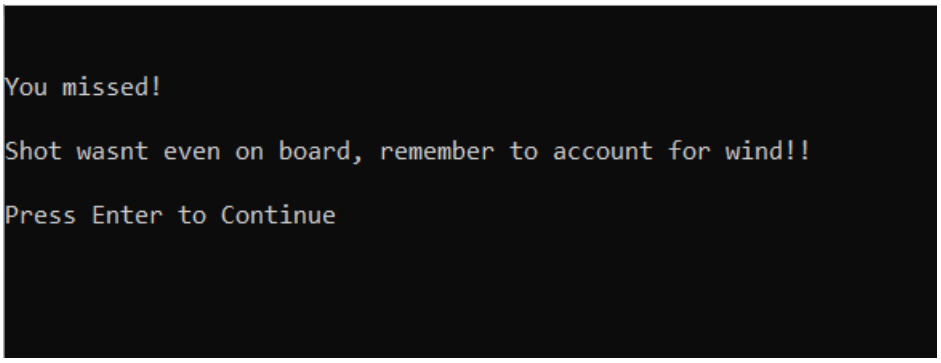
The list of program features that I developed and implemented effectively are listed below:

Feature 1 – Shooting Realism Mechanism

Once the game official starts, before the player is prompted to shoot, a random algorithm selects both wind strength and direction. Once these values are selected, the player is prompted to select a coordinate. Within the same prompt line, the player is shown the value of wind strength and direction. Once the player inputs the target coordinate, the new target position is calculated based on the direction of the wind and the strength. The algorithm then checks if the input coordinate is valid, and the game continues as prior.

The algorithm used was based on the rand() function built within c. To calculate the wind strength a rand() function. To create a realistic environment the wind strength went from 0-3, but if the wind strength was equal to 3, it was set to 0. This was done to create a more accurate game environment as 50% of the time wind strength is 0, 25% of the time is 1, and the last 25% of the time is 2. The wind direction is also set through a rand() function. The same values are used in the rand() function but depending on the value the direction is set to one of the 4 points of the compass (N, E, S, W). A collection of if statements are then used to calculate the new value of the target coordinate. This is then passed through the rest of the existing algorithm. Overall to achieve this a few key modifications needed to be made to the existing template. The main modification is the changed order of operation. The wind strength and direction are only incorporated once the original coordinates are passed to the get coordinates function. Once this is done and it returns valid, the target coordinate is changed and checked if it is still within the BOARD_SIZE which is defined in the header file. If it is out of bounds, it calls a unique end-of-turn function which reminds the player to account for the new realistic shooting mechanism and changed the player's turn. This modification creates and adds a new and unique aspect to the game and the fundamental design.



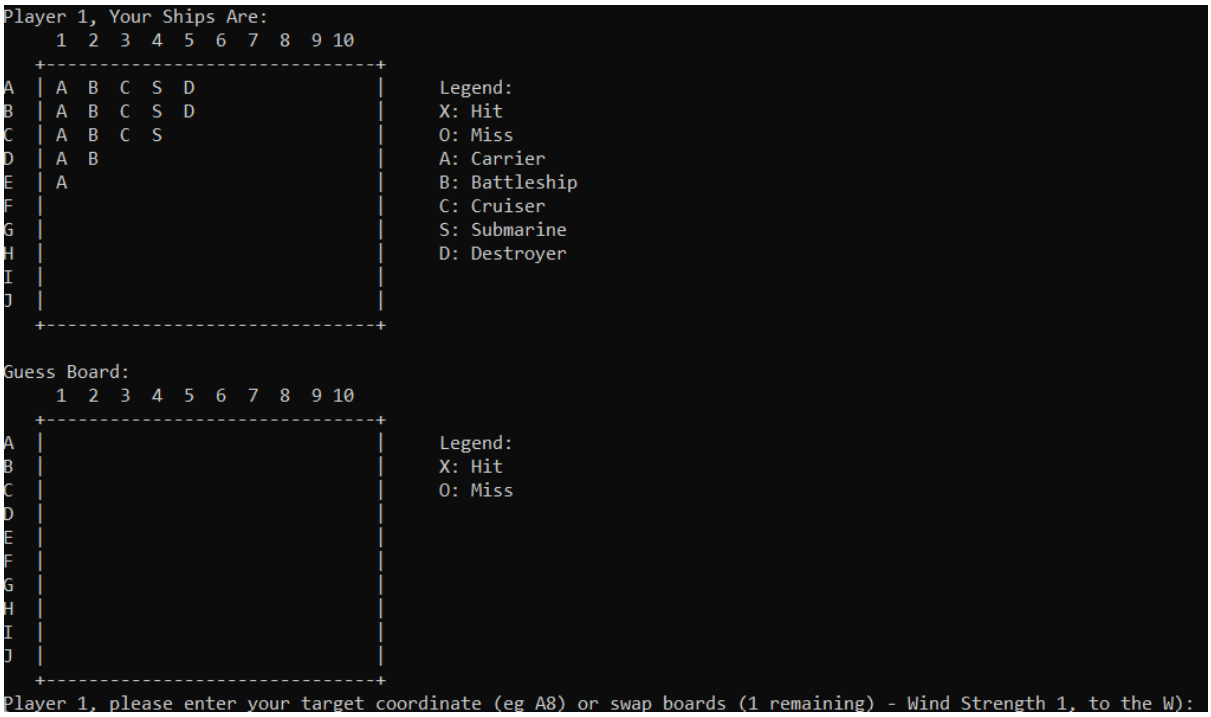


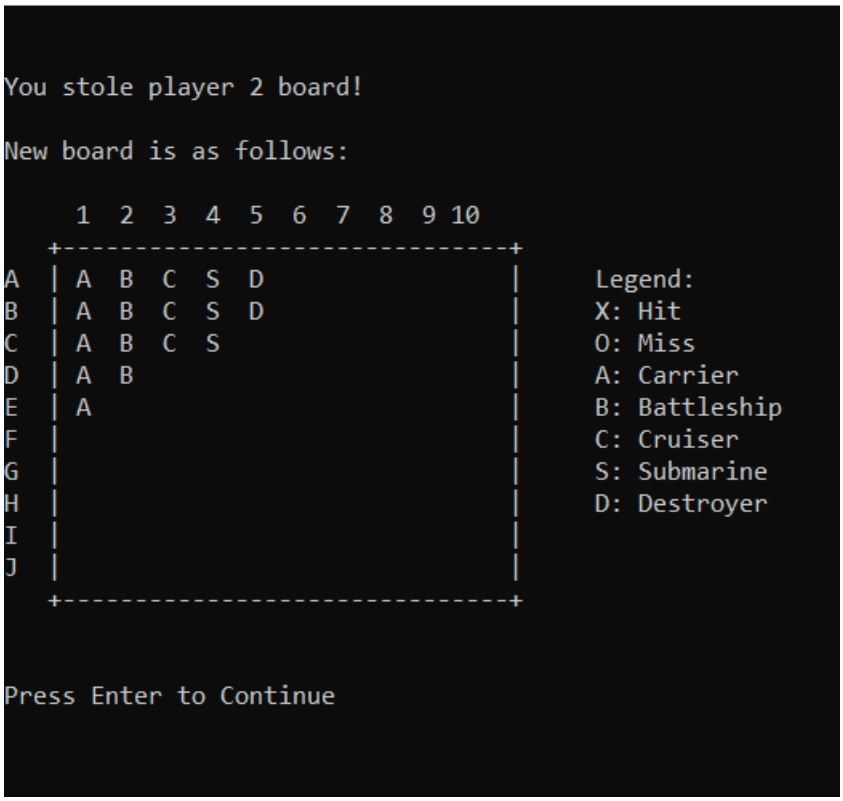
Feature 2 – Stealing Opponent Ships

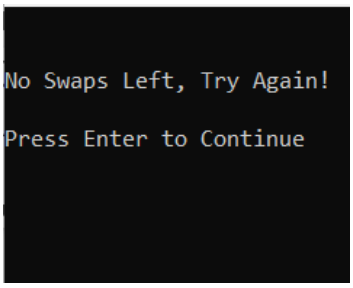
As the game is played, the player will be given the option to either shoot at the enemy's ships or swap the board. If the 'swap' is entered instead of a coordinate, the function checks if the player can swap boards and if true, will reverse the current state of the gameboard. A message will then print to the screen showing the current player their new board. The turn is then changed and player 2 is told that the boards were swapped. The game continues as normal, but each player can only swap boards once per game.

I created this addition to the game via the use of strcmp(). Strcmp enables the comparison of strings easily and quickly. In the play game function when the player is prompted to shoot, a string is taken which originally was used for the coordinates. I modified the program to first check if the string is equal to a saved string ("swap") and if true outputs 0. This output is then passed into a set of if statements. If the string is not equal to 0 the game proceeds as normal with no modification to the game. If the string is equal to 0, depending on which player turn it is currently, the current player board is reset to the current opponent board and the same goes for ships. This is done through a process similar to the original function, which sets the structures player board, player ships, etc… each turn. The difference is the incorporation of a call swap and player swap variable. The player swap variable is used to see what the player called the board, as there is a different structure to set depending on what the player calls swap. The second variable player swap checks if the player has any swaps left. Each player at the start of the game is given 1 swap, this is held in two separate

variables, one for player 1 and another for player 2. When a player calls swap, the variable is misused by 1, this enables the change of total swaps allowed. If no swaps are left, it prints an error message and prompts the user to enter target coordinates. After this, a final variable is used in an if statement, as if the player board is already swapped and a player wants to swap it back, a different set of structures is needed. This is done through the use of a variable called swap. Swap is originally set to 0 and when called is increased by 1, and if it's called and equal to 1, is then set to 0. By using this variable, I was able to check if the board is already swapped or not and fill the structures accordingly. After this, a simple print function is used to state the success of the swap and the current opponent board is printed showing the player's new ship arrangement. The turn then changes, and the new player is told the boards were swapped. The game continues as normal from here.







Program Completeness

The current project is almost completed. There are only 2 minor errors that are not crucial to the game's function. The first is the inability to print what ship the player hit in the play game function. The second error is also related to this as when the ship is sunk, I am currently unable to print what ship was sunk. For the two implemented features there are no current bugs and or errors, but the functions and algorithms created can be both streamlined and cleaned up. This would create more efficient and readable code. The major limitation of the currently implemented features is readability. With each turn, it is currently hard to read the wind strength and direction as it's at the bottom right of the screen. This could be improved with a clearer message before the game boards are drawn. If this is implemented, it would overcome the current limitation. Overall, the project can be considered completed with no bugs and or errors as due to the above areas of concern were removed as they were not vital to the game.

Conclusion

The project 'Battle Ships' that I coded and modified with the implementation of new features was completed and functioning. The introduction of wind to the shooting mechanism added a unique aspect of realism to the game. Also, the introduction of swapping player boards introduced a higher level of both competitiveness and strategy. These features were coded into the core project template resulting in a seamless and complete game environment.