## *Battle Boats Game*

## Contents

|  |  |  |
| --- | --- | --- |
| 1. | Introduction To the Game Contains all the information about what the game is about and what I hope to achieve in within making this game. | Pg: 2 |
| 2. | System Requirements Goes through all of the system requirements to run the game. | Pg: 3 |
| 3. | Instructions To Play Shows all the instructions on how to play with detailed and explained screen shots. | Pg: 4 |
| 4. | Pre-Development Plan Goes through all the details on how the game will be made with flow charts and hierarchy diagrams. | Pg: 5-6 |
| 5. | Development Of Game Shows the code and states what the more complex algorithms and functions will do and how they affect the rest of the program. | Pg: 7-10 |
| 6. | Bugs And Fixes States any bugs found and how I plan to fix them within the future. | Pg: 11 |
| 7. | Future Development Plan/ Conclusion Reiterates the idea of fixing the bugs I have stated, but also shows how I plan to improve the software if I where to continue with it. | Pg: 12 |

## Introduction

## Welcome To The Game

This is a game based of the idea of battle ships where, a player and a bot can interact with a user interface to play battle boats. It runs of the .net framework and is written purely in C++. The game has a built-in bot to play against that can become progressively harder, although the setting is not set to improve difficulty, the bot can become better at guessing where your boats are based upon where it has hit previously.

A picture containing background pattern

Description automatically generated

The game has sourced from the idea of a need for open-source basic battle ships game that developers can source from to create their first game engines. With a pre-coded Pc and player engine, a developer can analyse the code and get inspiration for their own projects. The average user however can play this game with very little system requirements, and ease of access.

## System Requirements

|  |  |  |
| --- | --- | --- |
| Requirement | Details | Download |
| .Net Framework or equivalent | All windows languages require the .Net framework for windows to translate the code. | [Download .NET Framework | Free official downloads (microsoft.com)](https://dotnet.microsoft.com/en-us/download/dotnet-framework) |
| Microsoft Visual C++ Redistributable (64 bit or greater) | Microsoft visual C++ allows for windows to translate the code compiled code into readable machine code. This is required as default else the program will fail to run. | [Latest supported Visual C++ Redistributable downloads | Microsoft Docs](https://docs.microsoft.com/en-us/cpp/windows/latest-supported-vc-redist?view=msvc-170) |
| For developers (A C++ compatible IDE and Compiler) | A C++ compatible IDE is essential for developers planning to add to the program. And C++ cannot be compiled by another language of complier. | [Visual Studio: IDE and Code Editor for Software Developers and Teams (microsoft.com)](https://visualstudio.microsoft.com/) |
| Windows 7 or greater | The program will not run off another OS. This Is because one of the imports relies on windows in order to grab the header file from. Without it, no re-rendering will occur and will disrupt progression through the game. | [Download Windows 10 (microsoft.com)](https://www.microsoft.com/en-gb/software-download/windows10) |

**Instructions**

This game is very simple to play. Firstly, you will be prompted with a start menu as you load in. With this you can select 3 options; Instructions, Play or Exit. To move between options, press the up and down arrows and to go back from the instruction’s menu press ‘Esc’. Once you have selected play, you will be asked to place your ships. Firstly, enter your boat number you would like to place, then type in the line stating, “Enter X”, Where you will enter what X you would like to place and “Enter Y” again where you will enter the Y you would like to place in. Once done so you will see your ship show up on the printed display. The next screens are display screens and help to let the computer render the PC engines decisions pregame to reduce in game latency. And the main game board shows your board and a computer board, again type the X and Y and you will hit the locations. ‘!’ a is hit and ‘X’ a is miss.

A picture containing table

Description automatically generated

For now, you cannot replace the boats once placed, I plan to add this within the future if it is not added already. But this menu just allows the user to interact and decide where they would like to place the boats. What’s more you must enter the numbers separately because there is no separating algorithm.

**Diagram

Description automatically generatedPre-Development Plan**

This flow chart shows the in game while loop I will develop that will run until told to return from an internal function. In reality, It will be much longer than this because it will need to detect if all the ships have been eliminated. But this function runs through hit players inputs to see if it hit a boat and will switch to the opposite player once complete. This does not however show the inputs from the user.

From this task I hope to achieve a working game with easy-to-use prompts and interface so an average user can interact with the game and play it enjoyably. This will ultimately come with less bugs and quality of life patches that will allow the user to interact easier. An example of what I want to get out of it is a working menu with prompts to go into multiple sections and an area to idle on. What’s more I would like to be able to make a game that the user can fully interact with and realize that this is a game based off battle ships. I also need to make sure that it is not too complex for the average user. Also, I need to make it so that a user would want to play my game and make it as unique as possible. For example, ‘cut scenes’ get the user to get a build up before or after the game. I will be able to see if my success criteria have been met by taking a survey of some sorts post development.

Text

Description automatically generated

This is the pseudocode for the flow chart above, this just reiterates that it is in a while loop in which I will call a game loop. And the current hit shows the cords of the current hit being set to either a, ! or X depending if it was a hit or miss. Then it re-renders by clearing and rendering. This also shows if a boat has been eliminated or not. However, in theory without the context of the functions it will not run

***Text

Description automatically generatedText

Description automatically generatedDevelopment Of Game***

If the current square contains any of the ships it will tally up. And if the total tally = 0 then there is no boats left and the player has won. This will run every turn to see if you have eliminated a ship or all the boats

Iterates through x and y going across 8 and down 1 every time for a total of 8 times to reach the end of the board.

States the players turn and if it is 1 which is the local players it will let that player play else it will run through the pc decision function contained within PCEgnine.h

This is some debugging code I used to see what was my tally and if it was adding up correctly

This is the miss print and it prints with a | and a “ ” so that it will align to print the board in the correct way

Takes the ascii value off 177 which is a custom character that was declared at the beginning of the main.cpp, this adds more character to the board in terms of making boats look more like boats.

This code iterates across the board from X axis. This is also contained with Y iterative loop that will iterate across the Y so it covers the whole board. This is repeated 8 times to reach cover the while board.

This section will outline all the details of the game development process. This will be a useful section for developers looking to expand upon the project. Each larger piece of code will be explained in as much detail as possible.

Keeps prompting the user for a location to attack at until the user has a valid location. Will only move on if they have not hit there before, or it isn’t out of range.

***Text

Description automatically generated***

Prompts the user for inputs to hit next and then the user will be displayed with an idea of where they hit.

Sees if the value is not = to 0. If it is not 0 there is something that is a boat therefor it has a hit on a boat.

Sets the values to a hit value and replaces the boat value, this allows for us to print an ! as we have a differing value.

Just states if all else then it is a direct miss and it will set all the values alike the direct hit. This can cause some bugs, so to fix it in the future I would create another else if condition and make another else to state a fatal error.

***A screenshot of a computer

Description automatically generated with medium confidence***

Says that the boat that has been placed by the pc is in a valid location now and it can now move on. And moves onto the next boat by adding one to the computer boat number,

Sets the value of the current boat to the pos on the board so we can identify the location and type of boat later in the game. Also sets this in the direction oi the boat.

Does the same again for each boat number until it generates enough number that it has placed all the boats in valid locations.

Says that the boat that has been placed by the pc is in a valid location now and it can now move on. And moves onto the next boat by adding one to the computer boat number,

***Text

Description automatically generated***

Clears the screen and sets the color to white and black forecolor.

States that if the winner was 1 then the bot wins and it will flash it and else it’s the player that won.

Allows for the screen to flicker from blank to a text, just adds to the effect and the immersivity of the game.

***Text

Description automatically generated***

Says that if the key pressed was down arrow, then the opt\_start screen will decrease by one and thus rendering the menu to be moving down by 1, if it has reached the bottom however it will reset to the top.

Getches to find the key that was pressed last

Shows that if the key pressed returns the console then it should take that as the enter key. This thus allows the user to enter a option.

**Graphical user interface, text, application, chat or text message

Description automatically generatedSystem Files And Purpose**

This allows for me as a developer to sort through my files and allow for each section to be split into their separate areas. For example, Pcengine.h hold all the code to allow the pc to place boats and place the boats to where they would like. There is a total of two custom engines I developed in this project, one of which is the bot engine the other in the local decision engine. The Main.cpp file is where the main sum of code is executed. If I wanted to multithread this program, I would have multiple cpp files to run multiple tasks at one time.

|  |  |  |  |
| --- | --- | --- | --- |
| ***Key Pressed*** | ***Ascii Value*** | ***Opt\_Start*** | ***Response*** |
| ***Down Arrow*** | ***80*** | ***+1*** | ***Moves Down One Option*** |
| ***Up Arrow*** | ***72*** | ***-1*** | ***Moves Up One Option*** |
| ***Enter*** | ***‘/r’*** | ***1*** | ***Enters Option and exits condition loop*** |

***Bugs And Fixes***

|  |  |  |
| --- | --- | --- |
| ***Bug*** | ***Priority*** | ***Fix*** |
| ***A Bug when entering the value of the cord and it is showing that it’s out of range when in fact its already in use*** | ***Minor*** | ***Just go over the collision detection in the localplayerengine that will decide if its out of range or not*** |
| ***When entering invalid cords it can display a fatal error message and will exit game*** | ***Major*** | ***Just need to add a function to allow detection of inputs of if it is within syntax of game build.*** |
| ***High resource Usage due to heavy rendering of repetitive code*** | ***Minor*** | ***Would need to fix up the GUI render title to allow for single render instead of repeating 3 times*** |
| ***When entering game it won’t allow the user to escape to the main menu without a fatal error*** | ***Major*** | ***Just allow game saving into memory and hold game positions then reload it when back in menu*** |

***Future Development & Conclusion***

To further develop this game, I would like to add multiple things that will allow the user to enjoy the app more. For example, rotation of selecting boats would allow for more games, developing this into a GUI like SDL2 or equivalent or even improving the cord selection system, for example the user can select their cords based upon letter, X eg: A1 ect. All of these would allow for the user to have a better experience whilst playing. However, with very limited time to develop a medium sized project like this with little major errors has made my success criteria fulfilled. Thus, all I have stated in my criteria I have achieved, and I was able to add more like exit winning menus and small changes like a white on black background.

However, coding this project like this in a programming language like C++ is tedious, but it will ultimately run better for a game that will require more efficient code. Python’s code would have been far shorter, but C++’s has proven to be far more efficient in low preforming systems. But I’m happy I was able to take the opportunity to improve my logical thinking within C++. And due to the local requirement, a game like this requires it makes it a great project to make you think about how we can immerse lines of code into a 2D reality. And I hope to take this knowledge I have gained to use it in my final project and potentially make a far more complex piece of software. And the basic development I have gained of GUIs I can use to make a 2D-3D engine later down the line. – Josh Lake.