# 

OCR GCE A

COMPUTER SCIENCE PROJECT

H446-03

Name: Oliver Milligan

Candidate Number: <INSERT CANDIDATE NUMBER>

Gordonstoun School: 71216

Title of Project: <Hello world>

H446-03 – Project CONTENTS

Table of Contents

[A. Analysis 3](#_Toc499540517)

[Interview 8](#_Toc499540518)

[Features of proposed solution 10](#_Toc499540519)

[Hardware and Software Requirements 10](#_Toc499540520)

[Requirements Specification 11](#_Toc499540521)

[B. Design 12](#_Toc499540522)

[Systems diagram 13](#_Toc499540523)

[Proposed Screen Designs and usability features 13](#_Toc499540524)

[Detailed summary of the process including key structures and variables 15](#_Toc499540525)

[Algorithms 18](#_Toc499540526)

[Test Data for development 24](#_Toc499540527)

[Test data for beta testing 26](#_Toc499540528)

[C. Developing the coded solution (“The development story”) 30](#_Toc499540529)

[D. Evaluation 30](#_Toc499540530)

[Project Appendixes 31](#_Toc499540531)

# A. Analysis

##### Description of Program

My solution to the problem given is to make a classic battleships style game with an implementation of AI. This will use the standard 15x15 game board along with the standard game ships. This include an Aircraft Carrier, which is 5x1 blocks, a battleship – which is 4x1 blocks, a destroyer – which is 3x1 blocks, a submarine, which is 3x1 blocks and a minesweeper – which is 2x1 blocks. In keeping with the traditional game rules, the ships may only be placed either horizontally or vertically across the grid squares. Usually there are 2 game grids displayed, one representing the players ships and opponent’s hits/misses and the other displaying the opponent’s grid and the players hits/misses. Only one player can take a turn at shooting at any one time and with each shot the turn alternates between players. The winner is the one who destroys the opponent’s ships first.

With regards to AI. The problem lies with getting the computer to act as a human would playing the game, that being whenever you score a hit, you try either one of up, down, left or right to see which way the ship lies. Consequently, once you find where the ship lies then continuing to shoot along that axis until the ship is destroyed. If you start missing along that same axis, the computer must know to retrace its steps and try 1 square before the first shot in order to destroy the ship.

##### Controls

My Battleships game will be controlled by primarily a mouse/trackpad. This will be used when selecting where players ships will be placed on the boards, as well as selecting a square to fire at on the enemy’s board.

##### Target Audience

The target audience of my Battleships game will primarily be to kids/teenagers between the age of 8-18. However, due to the simplicity and popularity of the game battleships, this will make it easy to play and understand for users from any age group. The reason for primarily choosing children/teenagers as my target audience is due to the fact that many children start using phones and computers from a very early age, meaning that they have a good grasp on modern technology and therefore can pick new games up and understand them very quickly compared to the over 65 age group – who didn’t grow up with this and subsequently may not be able to play and enjoy the game to its full potential.

#### Justification of how problem can be solved by computational methods

##### Thinking Abstractly

Due to the target audience of my project being primarily children or teenagers, my project will be greatly abstracted in order to keep it easy to use for every user, one such example of this is going to be a scoring system.

* In game scoring system: Players will be awarded a certain number of points if they score a hit and depending on which ship they hit (Battleships will be awarded the highest number of points, then destroyer, then submarine etc.). This will allow comparison between both players of the game. Bonus points will also be awarded for landing multiple hits in a row. This is abstraction as in war whether winning or losing would be based on the physical ships lost, whereas in the game it merely disappears of the board and you are rewarded with points.
* High score table: After the game has finished, a high score table will be shown with a username for each player, allowing comparison between all players of the game. This is abstraction due to the fact that usernames represent two warring factions and the scores represent a battle won or lost.
* End of game: When the player loses all of their ships, a splash screen will replace the game board with text telling the player whether they have won or lost.
* Pausing the game: This will allow users to take a break if needed to grab a drink or go out and come back to the game at a later time – giving the user more freedom. This is abstraction as you can’t leave a battle instantly in war time.

##### Thinking Ahead

* Thinking back to the stakeholders or target audience for my project, the controls need to be simple so that everyone from tech savvy teenagers to OAPs can use my program. As a result of referring to this I have decided that the input to interact with my program is the mouse due to the simplicity of using it. Anyone with basic computer skills can navigate an operating system using a mouse, which will hopefully allow all age groups to take advantage of my program. The mouse will be used to navigate menus and to select the square to target when playing the actual game. Thinking ahead allows the developer to assess the usability of the program before the design or development stage of the program
* I believe it is also important to design the program as modular, this is so to avoid programming the same thing multiple times and instead to reuse already written code, this will save time when in the development stage.

##### Thinking Procedurally

* Thinking procedurally is the process of breaking a program down into more manageable chunks, this can be done through a variety of ways such as modular programs, decomposition and top down approach to program design. I personally am using both of the aforementioned techniques.
* One method of implementing these is using subroutines (procedures and functions) in order to split up the program into smaller simpler pieces of code. These will also allow the developer – in this case myself – to very easily reuse code as you can just copy the function and change the parameters being parsed depending on the situation. An example of the use of this will be the target selection and hit animation for both the player and the computer will be exactly the same, therefore we can reuse the code for this in both sections of the program due to a modular program layout.

##### Thinking logically

* Thinking logically is all about the logical decisions inside the program, these usually fall under one of two programming structures; iteration or branching. Iteration iterates through the code until the condition in the loop is fulfilled, whereas branching runs the code that satisfies the conditions given in the statement.
* An example of where iteration will be used is to keep giving players another turn until all ships are sunk, at which point it will have satisfied the condition and the program will move on. An example of where branching will be used is the main and pause menus, depending on which button is clicked, a different piece of code will be run for start game, options or quit in the main menu and resume or exit to main menu or desktop in the pause menu.

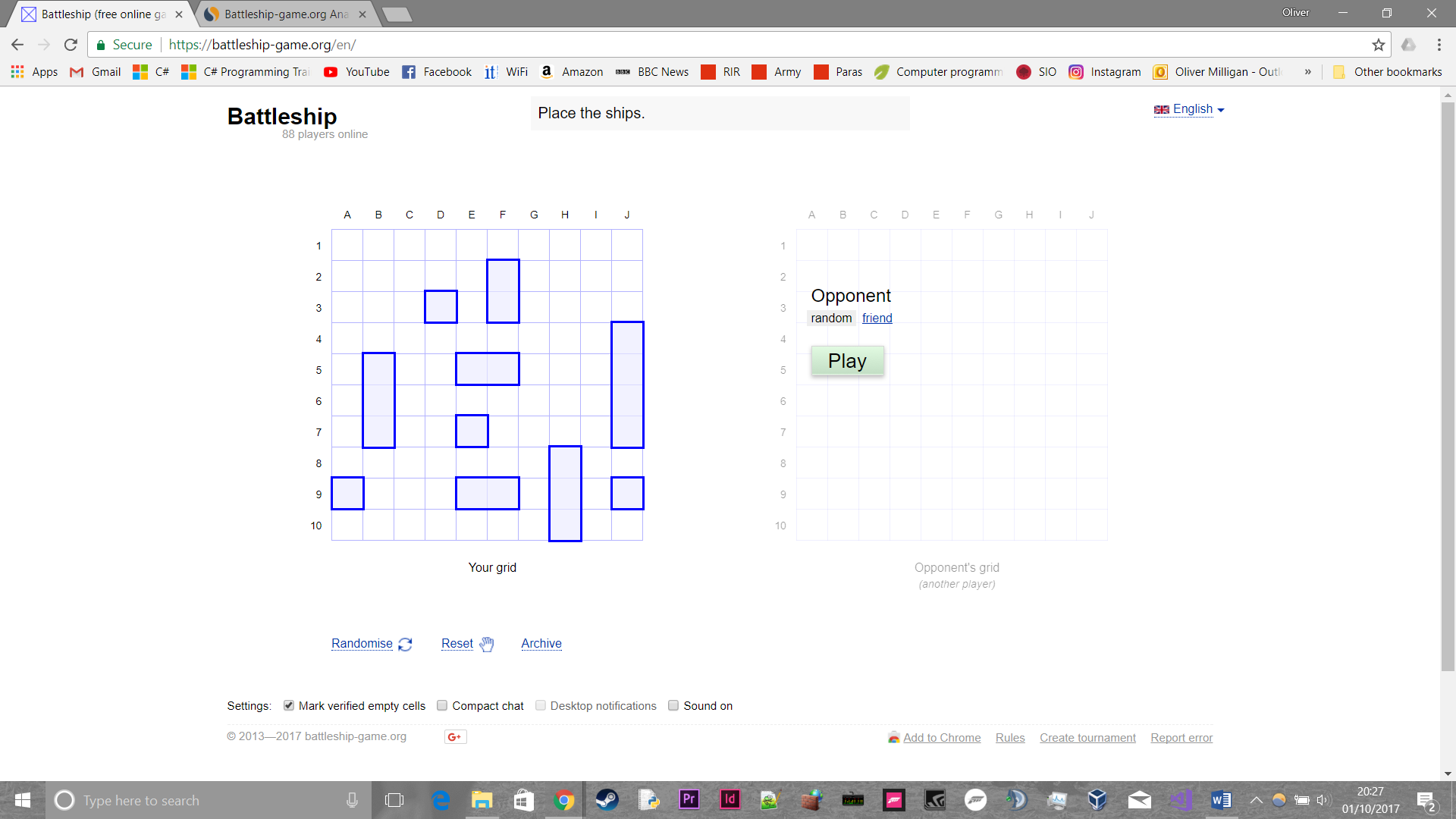
##### Thinking Concurrently

* Thinking concurrently is to determine the parts of the program which can be run at the same time and also to think of the pros and cons to the above mentioned concurrent activity.
* An example of where this could be used is both players selecting the position of their ships on the board at the same time or while switching turn the scoreboard updates.

#### Research

##### Analysis of a similar solution

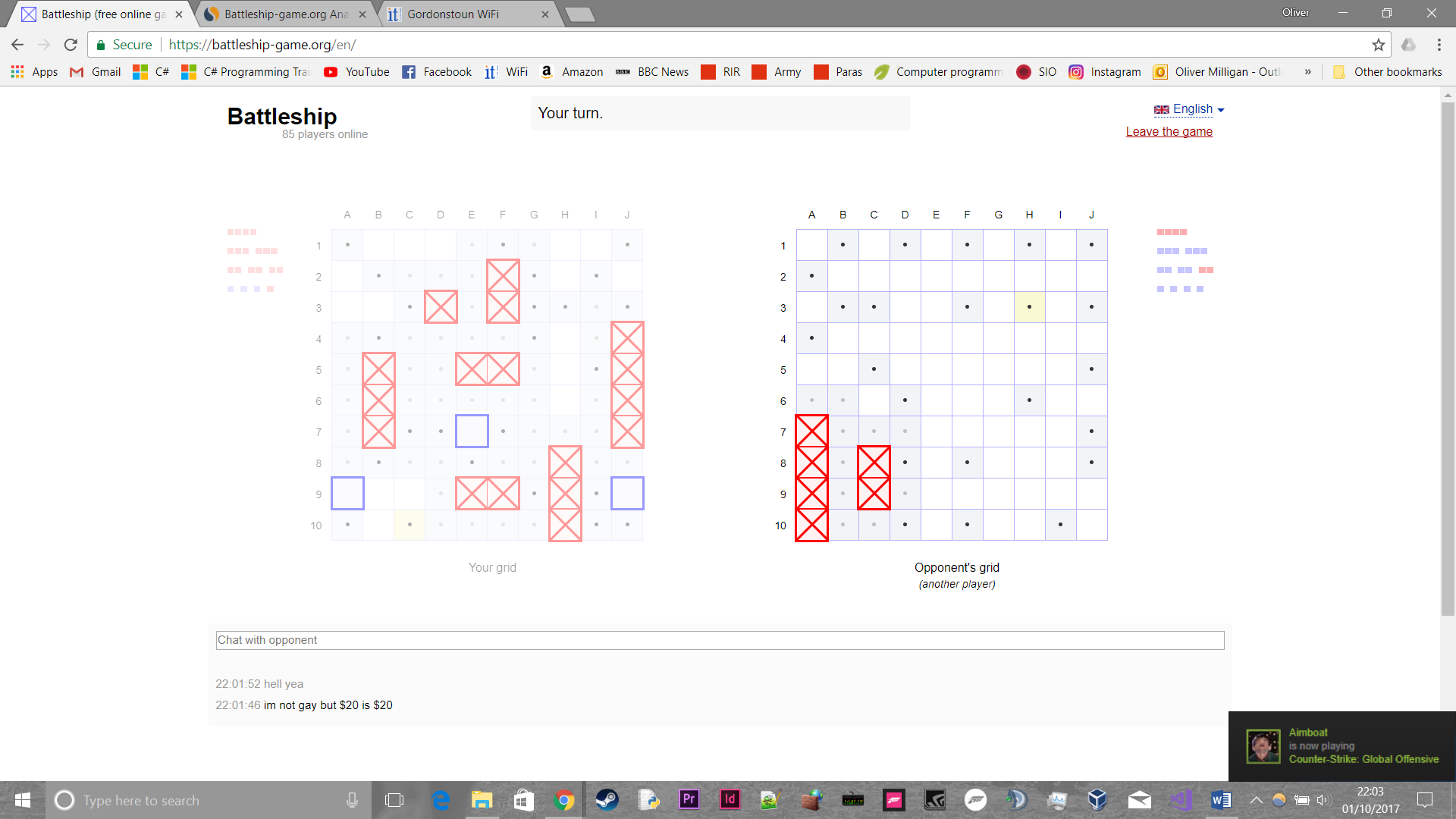
A popular online simple 2D version of battleships, found at <https://battleship-game.org>



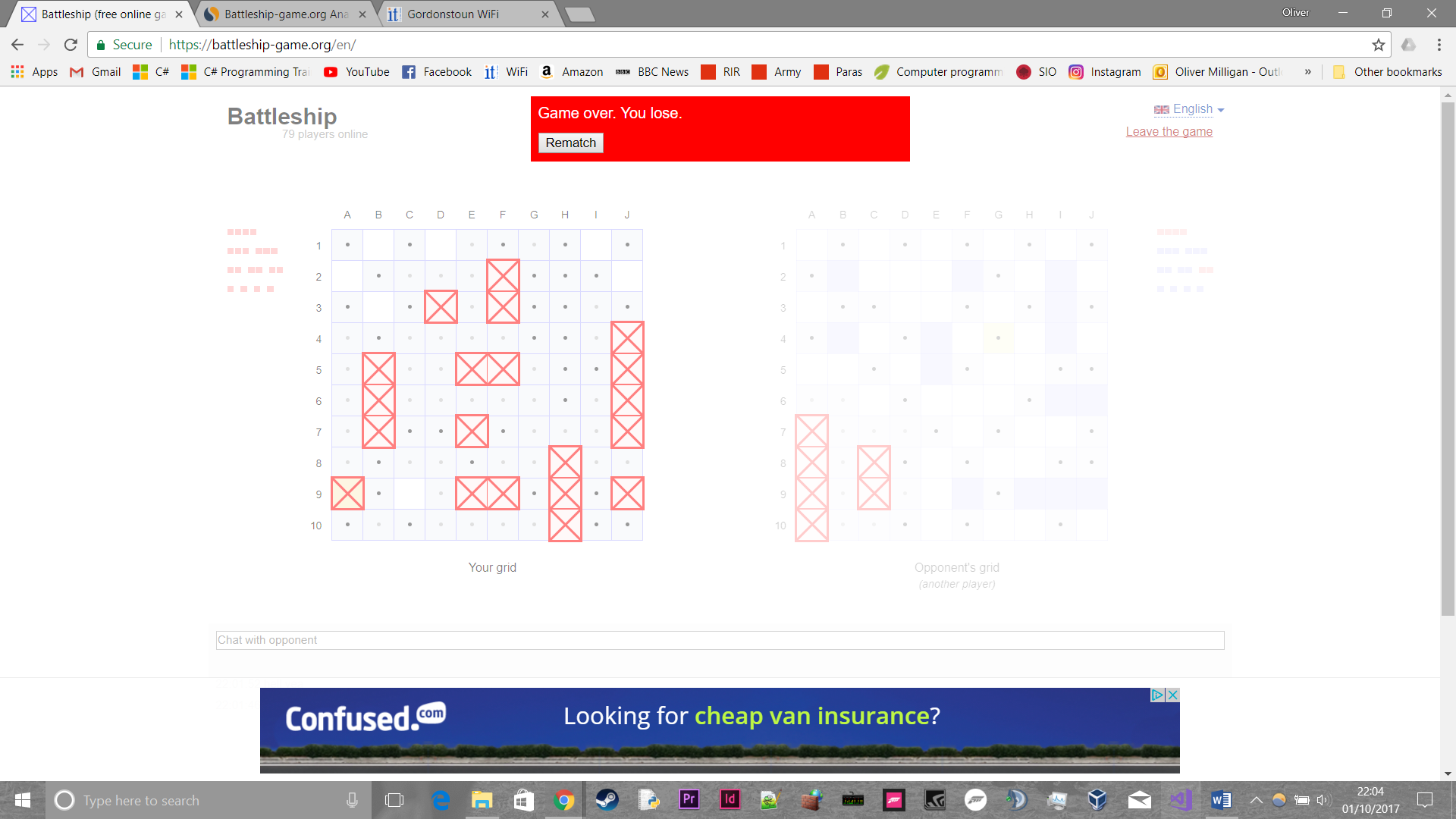
This version of battleships follows the traditional idea for battleships that it is a guessing game for two players. It is played on a board on which the players' fleets of shipsare marked. The locations of the fleet are concealed from the other player. Players alternate turns calling shots at the other player's ships, and the objective of the game is to destroy all of the opposing player's fleet.

Battleship is known worldwide as a pencil and paper game which dates from World War I. It was published by various companies as a pad-and-pencil game in the 1930s, and was released as a plastic board game by Milton Bradley in 1967. This version is online and multiplayer.

* **Start/Welcome Screen:** The screen that greets the user is very simple to understand, a key concept that I would like to have in my program. Instructions for what to do are seen at the top of the screen. The ships are pre-set in a random location, the user can either move the ship themselves or randomise the location with the button below the grid. Finally, on the right-hand side, the option to start the game with a random or friend is given.



* **In game:** Here we can see the very simple method they use to illustrate hits, misses and the number of ships left. Hits are a red cross through the square, whereas misses grey out the square and put a black dot in the middle of the square. The players ships are shown to the left and right of respective players as can be seen in the photo above, the number of dots indicates what type of ships are left. As this is a multiplayer PvP game, there is the ability to chat with your opponent at the bottom. There is a leave the game option in the top right and again the instructions are listed at the top of the screen – telling the player who’s turn it is.



* **End Game:** After the game is won or lost (in this case I lost) a banner is displayed at the top. Due to the fact I lost, the banner is red and states ‘Game over. You lose’. There is an option to rematch your opponent or else it takes you back the main menu to have a game with another player.
* **Mechanics:** The game in this case is played on a 10x10 board with 10 ships. The game is quite fun and addictive in this format, the simplicity of it is key. The graphics as previously mentioned are very simple and 2D, this allows it to be easily picked up by anyone who browses the web and happens to play the game. The size of the board relates to how long the game lasts, in this case about 2 minutes, which is very manageable for anyone who has a break in work or simply wants to have some fun. When you score a hit, the tiles around the hit are also greyed out, eliminating the squares on the board quickly, which can lead to tense endings.

###### Pros

* **Simplicity:** As outlined above in the analysis, it is very easy to use and understand. The graphics make it not an eye sore and add to the effect of the simple design. This has given me a good idea of how to make my program look as to keep people interested and entice them to play
* **Implementation of original concept:** I particularly like the smaller than usual size of the board (10x10). This makes the games short but interesting and saves time if you want to place your own ships. The number of ships in the game is also good, and there is enough space to keep the game interesting with the set number of ships. In my implementation of the game, I will reduce the number of ships that take up only one square (minesweepers) as this is my least favourite part of the game, as you can see by the above image in the end of the game there is only random one square parts of the board left free meaning you just click on all these and hope to land a hit.

###### Cons

* **No scoreboard or high score table**: I believe that having a scoreboard and high score table makes the game more competitive and would allow the target audience to compare their scores and encourage them to play more to get a higher score. This would be displayed on the losing screen when the red banner is currently displayed in the game.
* **No pause option:**  I would also like to have in my implementation a pause option that enables players to take a quick break, possibly only 30 seconds that they would get one of each match.

### Interview

##### Plan

My goal from the initial interview is to gauge how fun/exciting the game is to a stakeholder and anyways in which they think the game can be improved upon in order to make it more engaging to play. It will give me a second opinion on the likes and dislikes and how any changes should be implemented. Also, to find out do they agree with my current analysis and pros and cons of the program and is the game layout of the game suitable for multiple platforms.

##### Stakeholder

Name: Fin McCallum

Age: 17

Occupation: Student at Gordonstoun

**Why is Fin a suitable stakeholder?**

Fin is an avid gamer and plays a variety of games in his spare time on both his PC and phone including flappy bird, Dota 2, Candy crush and angry birds. He is also in the top of the age range of my target audience and so can give an accurate opinion of an average user of my application

##### Transcript

Q: Have you played battleships before?

A: Yes

Q: What did you like most about it?

A: Being able to move the pieces around the board and the customisability of it

Q: What where your first impressions of the online game?

A: Easy to use, the rules are simple and easy to understand

Q: Is there any way you would improve it?

A: Slightly longer games

Q: Do you find it easy and fun to play?

A: Yes, and it’s quite addictive

Q: What makes it addictive?

A: The quick games and the rivalry between the 2 players

Q: Do you think a scoreboard helps the competitive nature of the game?

A: Yes, I want to play the game again get to the top the scoreboard

Q: What is your opinion on the graphics?

A: They are simple and abstracted, it helps users to understand the game

Q: Is the game board a suitable size?

A: Yes, it helps to keep the game quick and doesn’t drag on

Q: Do you think the game lasts too long or short?

A: I think it’s just right, any longer and it would get boring

Q: What platform do you think would suit this game the best?

A: Desktop and mobile would both be great options for this game

Q: Do you agree with my pros and cons of the game?

A: Yes, I think they are just about spot on

##### Analysis

### Features of proposed solution

* The game will be played on a 10x10 game board
* There will be 10 placeable ships to put on the board
* The menus will be navigated and game interacted with solely with the mouse and not a keyboard
  + The interaction with the game will primarily done using Left Click or Mouse Button 1
  + Moving the mouse moves an on-screen cursor through the menus or on the board in game
  + The square to be targeted should be indicated by a change in colour or outlined in a different colour
  + In game if ‘P’ is pressed it should bring up the pause menu
* The game should last roughly 2 minutes
* The layout should be simple and easy to understand by people of all ages
* Although simple, the design of the game, especially the loading screen should be eye catching and entice people to want to play the game
* When launching into a game, there will be a splash screen and there should be options to start, change colour scheme from light to dark, view high scores and to quit
* Before the game the player should be able to place their own ships or be given a random layout
* During the game there should be an option to pause, which will last for 30 seconds
* When a ship is hit it should be clearly marked and indicated when destroyed on both players screens.
* When a shot is missed the relative square should be greyed out on both players screens.
* A scoreboard should be visible for both players during the game
  + Points should be awarded for a hit, with a larger amount for destroying the ship
  + Bonuses should be awarded for multiple hits in a row
* If possible to implement there should be a multiplayer option
* Players should be able to play against AI
  + The computer (or AI) should randomly choose a square to target
  + When it scores a hit, it should randomly choose a square around the one hit to target
* Instructions should clearly be displayed at the top of the screen so that new players can get a grasp for the game quickly
* Players should take it in turns to target a square, if a player scores a hit, they are given another turn
* In terms of graphics, either squares or sprites of ships will be used, if ships are used then animation of hits will be possible

### Hardware and Software Requirements

##### Hardware

|  |  |
| --- | --- |
| Item | Reason |
| Mouse | So that the user is able to navigate through the menus, and to interact with the in game boards |
| Keyboard | So the user is able to use the pause function |
| Monitor | Allows the user to view the game |
| Hard Drive with 1MB of space | So the user can store any sprites or effects/animations for the game, it is very simple so not much space will be needed |
| RAM – At least 2GB | To allow the user to use a web browser to play the game |
| CPU – Pentium/AMD equivalent or above | Suitable for any operating system that would allow the user to access the game |

##### Software

|  |  |
| --- | --- |
| Software | Reason |
| Windows XP or later, MACOS X or later, Linux, Flash, iOS, Android or Windows Phone 7 or later | Compatible operating systems with Monkey X |
| Monkey X Run time library | So that the user doesn’t have to download the Monkey X language to play the game |
| To develop the game: |  |
| Monkey X language and IDE | To allow you to program in the language and compile the code |
| Monkey X compatible libraries | To add more advanced features to the language that other users of the language have developed that add features such as improved graphics |

### Requirements Specification

|  |  |
| --- | --- |
|  |  |
| Requirement | **Justification** |
| The game will feature 10 ships for each player | This was agreed between myself and the stakeholder that this is the right amount for an enjoyable game |
| The board will be 10x10 units in size | This is the right amount of squares in order to keep the games short and sweet |
| The game will feature a semi AI element in order to make it feel like you are playing against a human when you are not | I discussed with Fin that this was the right approach to go as it would make the experience more enjoyable |
| There should be a scoring system and timer | This will help keep the game competitive and keep people wanting to play the game more to beat their times |
| The score should have a multiplier | This will reward players for scoring multiple hits and help to vary the scores on the high score table |
| The mouse should be used for control | As Fin mentioned this will keep the controls easy to use for a wide age range and ability of users |
| Have a main menu to introduce players to the game | This has been proven by a large number of online games to be the most suitable opening page on your game, this has to be eye catching in order to entice users to play. |
| Have a section for controls on the main menu | This will allow first time players of the game Battleships to get a grasp for the rules and structure of the game, which should allow them to fully enjoy the game. |
| Appeal to all ages | I believe that due to the popularity of the board game, it is important to keep the game both new and updated for those who have never played before – but also still recognisable for those who played the board game when they were younger. |
| Navigation and control done with the mouse or trackpad | This goes along with my appeal to all ages, this is due to the fact that the mouse is very easy and intuitive to use by anyone who picks it up. Navigation is done by using the left mouse button, this is the standard button to navigate through operating system, therefore it should be intuitive to navigate through the game menus. |
| Change the colour of the squares upon a ship getting hit | This will make it so that everyone intuitively knows that this means they have scored a hit, linking back to the ease of use requirement |
| Have a pause function, accessible by pressing ‘P’ | This will allow players to stop the clock when playing the game, so they can take a break mid game without affecting the time that will go onto the high scores table. The P key is the universal key for pausing in this style of game, however the users will be reminded of this in the controls page in the main menu |
| Allow users to choose a username | This allows users to customise their game experience and gives an alternative to using their real name |

# B. Design

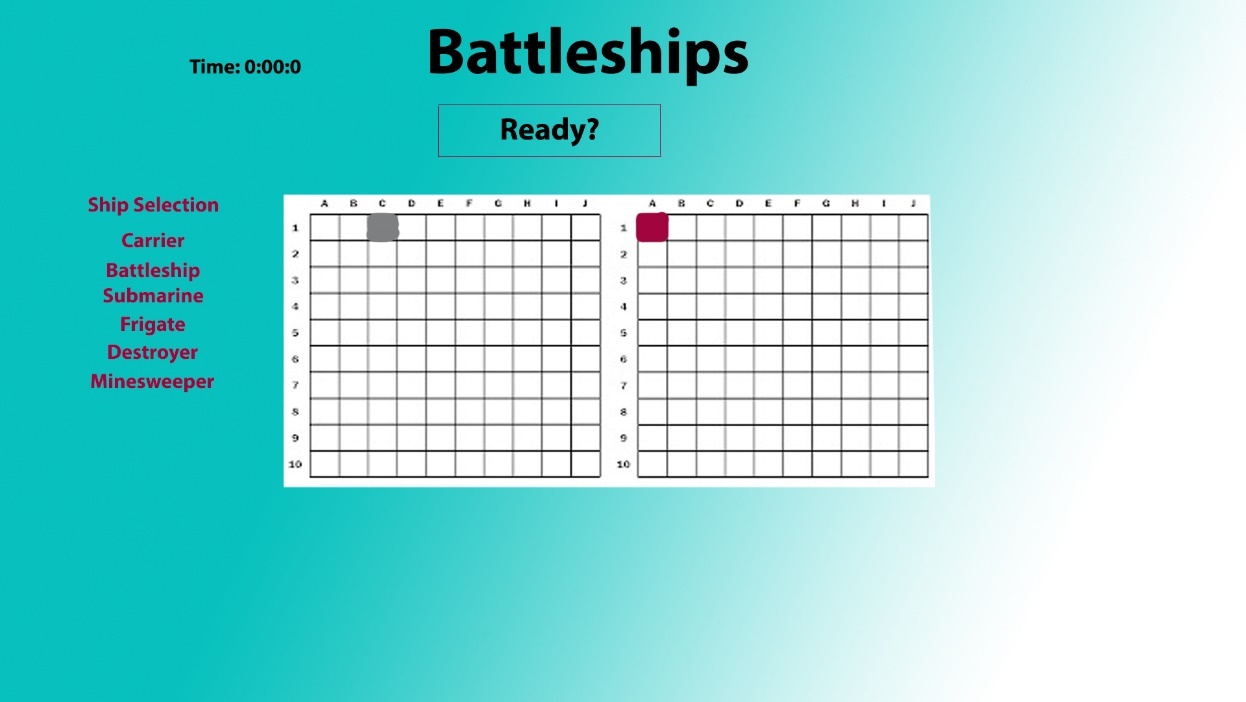
<See H446-03 Project Advice Booklet for help and guidance of what must go here.>

## Systems diagram

### Proposed Screen Designs and usability features



**Splash Screen design, simple and relevant image behind of a WW2 era battleship, with the game title overlaid onto the image**

****

**Basic game board design, with the colours shown that represent a miss/hit on the players and opponents board. This is the pre-game board where the player places ships on their board, and presses the ready button to change the game state to playing.**



**Main menu simple design, where users can choose what they want to do first, each option was thought out in the requirements spec**

### Detailed summary of the process including key structures and variables­­­

**[Key variables to be completed upon coding]**

#### Main menu

###### Start Game

* Displays on screen button
* When pressed, leads to game setup
* Allows for game initiation
* Navigation of the menu will be done via the mouse and mouse cursor

###### Game Preload

* Initialization of all variables needed in playing and paused game states
* Loads in all graphics (sprites etc.), sound effects and music from respective folders
* Displays a static loading splash screen to inform the user that the game is loading
* Changes game state to playing

###### Instructions/controls

* Informs user of how to navigate through menus and game board, such as the following:
* Navigation in menus and in game is done by the on-screen mouse cursor
* To pause the game at any time, press the “P” key
* To place ships, click on the ship you would like to place, then move it around the board using the mouse. Orientation can be done with the keys “Q” and “E”. Once a suitable location is found, press the left mouse button to place it.

#### Main Game

###### Loading/Setup of Game

* Loads the game board onto the screen
* Displays the game screen to display to the user the game is ready to setup
* All ships not placed are displayed to the left of the board
* When clicked, this allows the ship to be placed on the board
* If the ship is placed off the board, the squares that are off the board will be displayed in red and if tried to be placed, the ship will go back to the left of the board
* When all ships have been placed on the board, a button that says “Ready?”, when clicked a countdown of 5 seconds will be displayed to let the user know the timer has been started

###### Sound/Music

* Songs will be randomly chosen/shuffled from a folder in the game files
* Depending on user feedback, there may/may not be a sound that is played upon scoring a hit

###### Ships

* Start with 10 ships
* Once placed at the start of the game, the x and y coordinates are stored in an array, and the ship can no longer be moved
* Each one varies in x coordinate size, but are 1 y coordinate long
* Upon getting hit, the square representing the ship will turn red
* When destroyed, a cross will appear over the red squares
* A win is awarded to the player who destroys all of the other players ships first

###### Player Inputs

* As discussed in the controls section, interaction is primarily done using a mouse/trackpad
* Interaction is done by clicking on menu items, and pressing squares on the game board that the player would like to strike
* In order to make this easier the mouse cursor will not disappear whilst in the game

###### End Game

* Once a win is awarded, the end game screen will be displayed
* The user will be asked to enter in a username, which will be stored as a variable
* This username along with their time will be stored in the high scores file to display in the high scores table
* The high score table will display the username, score and time of the top 10 best players to add a competitive nature to the game

###### Game States

* Playing
  + Game is initialized
  + Loads relevant game resources (graphics, sounds, sprites)
  + Plays music at an ambient level in the background
  + Sets timer to 0/Resumes if coming from paused game state
* Paused
  + Activated upon press of the “P” key during the playing game state
  + Displays pause menu
  + If “P” or Play button in menu is pressed, resume playing game state
  + Pauses music
  + Pauses game timer
  + Game remains initialized
  + If menu button pressed, quit the game and load menu
  + If exit button pressed, exit game to desktop
* Menu
  + Displays the main menu, as detailed previously
  + Allows for exit to desktop and to start playing game state

###### Scoring

* Players are awarded +100 score for a hit
* A multiplier is awarded when consecutive hits are scored (x2 for 2 hits, x3 for 3 etc.)
* This multiplier is lost when a strike is missed
* A timer in the layout of a digital clock is displayed on screen in order to let users know how they are doing in comparison to high scorers
* Users are allowed to choose a username of up to 15 Characters to represent themselves on screen, which is stored as a variable

###### A.I

* A.I will be represented in my code by semi/pseudo random guesses made by the computer verses the player
* This will make the user feel like they are playing against a real player
* The A.I should make guesses in the game places that humans do at the start of the game, i.e. in and around the corners of the board – if there are no ships there, it should randomly select squares to strike
* Upon scoring a hit, the A.I will choose a square surrounding the ship to strike, until it finds the orientation of the ship, at which point it will add respective x or y coordinates on to destroy the ship

### Algorithms

START

###### Scoring

Set score, multiplier time=0

Do strike coords==ship coords.

**Yes**

Multiplier+=1

Score=score+(100\*multiplier)

Does

endgame==True

**No**

Multiplier=0

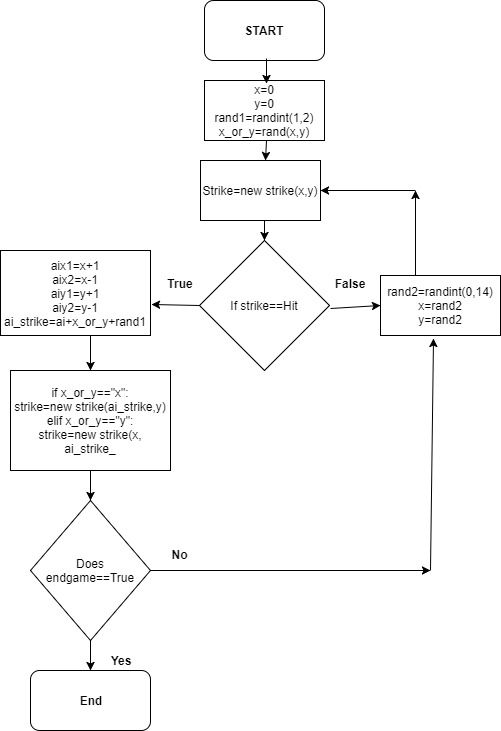
**Yes**

highscore.write(score, time)

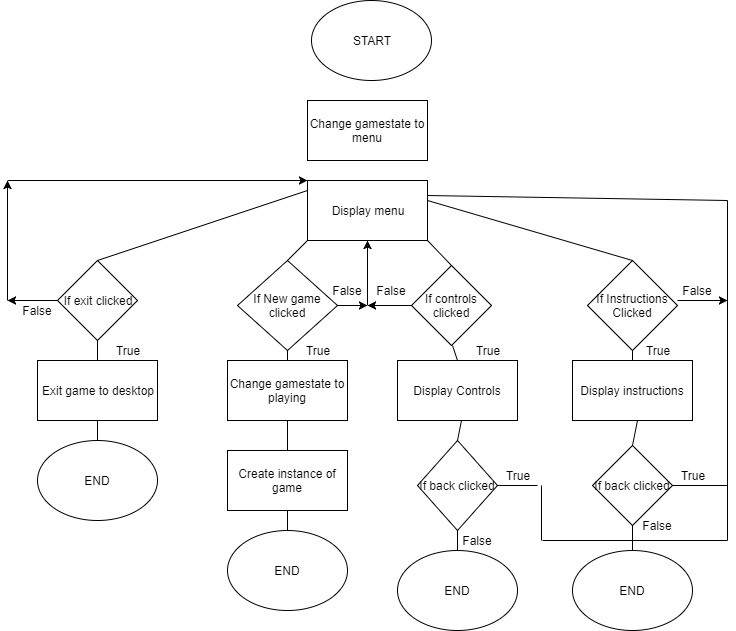
**No**

End

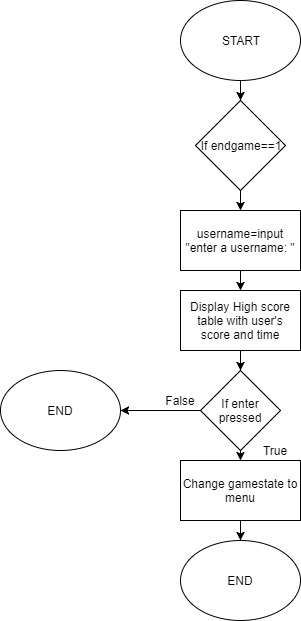
###### A.I



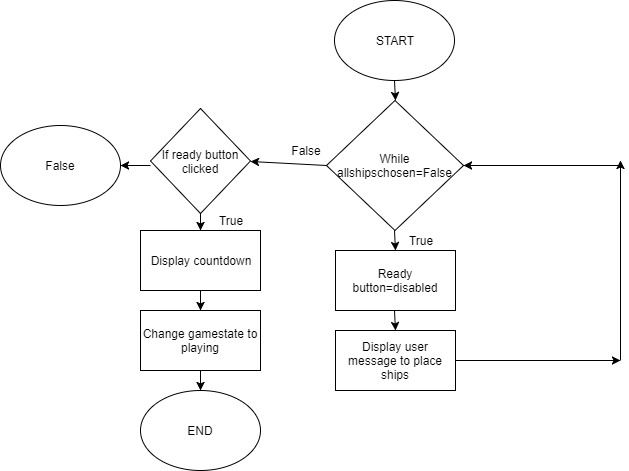
###### Main menu



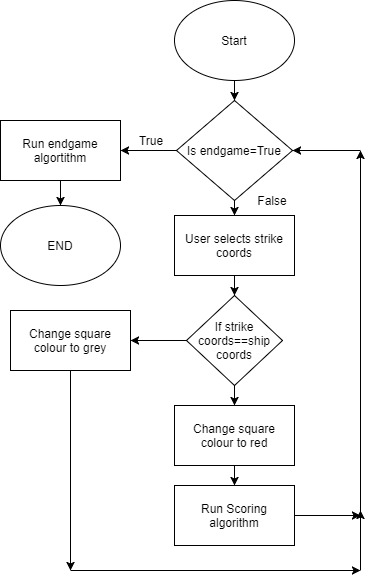
###### End game



###### Start game validation



###### Hit detection



### Test Data for development

Testing for beta/acceptance testing is done via black box testing (monitoring the inputs and outputs of the program)

**Username validation**

|  |  |
| --- | --- |
| Input | Validity |
| Length of 15 | borderline |
| length of 0 | invalid |
| length of 7 | valid |

**Menu navigation**

|  |  |
| --- | --- |
| INput | Validity |
| “P” | **Invalid** |
| Mouse button 1 (on BUTTOn) | **valid** |
| “esc” | **Borderline** |
| Mouse button 1 (not on button) | **Invalid** |
| USE mouse to move cursor around screen | **VALID** |

**Pause (in game)**

|  |  |
| --- | --- |
| Input | Validity |
| “P” | **Valid** |
| “MOuse button 1” | **Borderline** |
| “r” | **Invalid** |

**MENU FUNCTIONS**

|  |  |
| --- | --- |
| INPUT | VALIDITY |
| Click New game (MB1) | **Valid** |
| Click controls (MB1) | **VALID** |
| CLICK INstructions (MB1) | **VALID** |
| CLICK new game (mb2) | **Invalid** |
| Click instructions (mB2) | **INVALID** |

**PRE-GAME**

|  |  |
| --- | --- |
| INPUT | VALIDITY |
| Click Ready button (all ships placed) | **Valid** |
| Click ready button (not all ships placed) | **INVALID** |
| ‘P’ | **INVALID** |

**Menu ITems (instructions/Controls)**

|  |  |
| --- | --- |
| Input | VALIDITY |
| Click return | **valid** |
| Click on screen (not on any buttons) | **Borderline** |
| ‘P’ | **Invalid** |

**Pause**

|  |  |
| --- | --- |
| Input | validity |
| ‘P’ | **Valid** |
| ‘7’ | **Invalid** |
| MB1 | **Invalid** |

**Scoring**

|  |  |
| --- | --- |
| INPUT | Validity |
| HIT | **VAlid (increases score)** |
| 2 Hits | **Valid (Increases SCore)** |
| Miss | **Invalid (doesn’t increase score)** |

### Test data for beta testing

**Main Menu**

|  |  |  |
| --- | --- | --- |
| Test | Explanation | Expected Outcome |
| Splash screen appears | Upon game launch and compiling, splash screen should appear | Splash Screen appearing |
| MEnu APPearing | When any key pressed on splash screen, menu appears | Main menu appears |
| Cursor to navigate the menus | The cursor should move around the screen when user moves mouse | Cursor moves |
| Exit button Closes Game | Game quits to desktop when exit button clicked | Game quits to desktop |
| Display INstructions | When instructions clicked in menu, the instructions should be displayed | Instructions screen displayed |
| Display controls | When controls clicked in menu, the controls screen should be displayed | Controls Screen displayed |
| Back button in (in menu items) | When clicked, takes user back to menu | menu screen displayed |

**Pre-game**

|  |  |  |
| --- | --- | --- |
| Test | EXplanation | Expected Outcome |
| Clicking ready button | **when clicked, should start a countdown to begin game** | **Display countdown, then load into game** |
| Ship placement | **Ships should be able to be moved around the board with the mouse** | **Ships should be able to be selected and placed on board** |
| Clicking ready button when all ships not placed | **This should not start game and display a message to user telling them they need to place all ships before starting the game** | **Error message and no countdown** |

**Scoring**

|  |  |  |
| --- | --- | --- |
| Test | Explanation | Expected outcome |
| Hit | **A hit should increase the score by 100 points** | **Score increase of 100** |
| Timer | **Timer should start upon game loading** | **timer starts from 0:00** |
| MUltiple hits | **Multiple hits should increase the score by 100\*(the amount of hits)** | **Score increase of 100\*multiplier** |
| Score table update | **When game is finished, the high score table should be updated with the users result** | **Users score entered into high score table** |
| High score table displayed | **When the game is finished, the high score table should be displayed** | **high score table displayed upon game ending** |
| Timer stops | **When game ends, timer should be stopped** | **Timer stops at game ending** |
| Username entry | **USer should be able to enter a username at the end of the game** | **Prompt to enter a username** |

**Sounds**

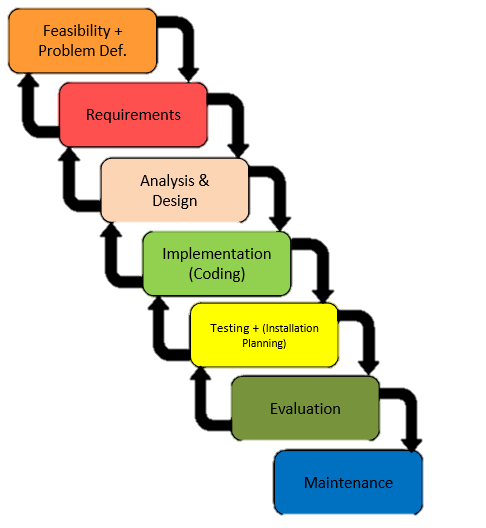
|  |  |  |
| --- | --- | --- |
| Test | Explanation | Expected Outcome |
| Music plays on start | **When game is launched, background music should start** | **music plays on game start** |
| Change of music on game win | **victory music should be played if the player won** | **Music change on game ending** |
| Change of music on Game Loss | **Defeat music should be played if the player lost** | **Music change on game ending** |

**Game functions**

|  |  |  |
| --- | --- | --- |
| Test | Explanation | Expected Outcome |
| Hit | **When a ship is hit, it should change the square to a red colour** | **square that was chosen by user turns red** |
| Miss | **When a miss occurs, the square selected should be greyed out** | **Square chosen turns grey** |
| A.i (Hit) | **The AI should choose squares around the hit to try and destroy the whole ship instead of continuing to choose random squares** | **A.I player chooses squares around original hit to target** |
| A.I (miss) | **A.i Player should choose random squares around the board, as well as targeting corners where players usually place ships** | **A.I player chooses random squares to target until scoring a hit** |
| Game win | **Game winning message should be displayed and end game function run** | **end game function run with win message** |
| Game loss | **Game losing message should be displayed and end game function run** | **End game function run with loss message** |
| Movement | **Mouse cursor should move when user moves mouse** | **Mouse cursor is able to be moved around screen** |

### SDLC

**Waterfall Model**

The waterfall model (as shown in the diagram to the right) is useful during the initial stages of design and specification, due to the fact you are working off of only 1 set of requirements which are not changing. For the coding or implementation of the requirements, a different approach is needed which has constant evaluation, as opposed to only one evaluation stage at the end of the project

**Agile Methodologies**

**Requirements**

**Plan**

**Design**

**Develop**

**Release**

**Monitor**

**/Maintain**

Agile methodologies are especially useful in the development and testing phases of the project. This is due to the fact that it is constant evaluation, and allows communication with the stakeholder to ensure that any improvements that need to be made are implemented into the final product.

### Proposal sign off

##### interview with stakeholder

Q: Do you like and find relevant the screen designs at this stage?

A: Yes, they look promising

Q: Would you make any improvements to the screens at this stage?

A: Yes, the background image could be improved

Q: Are there any features that you want implemented that have not been represented in the systems diagram?

A: No, everything crucial to the core game has been represented

# C. Developing the coded solution (“The development story”)

##### Iteration 1

The first iteration is the creation of the game board and placement of the ships onto the game board

// Constants -setup of ships

var CONST = {};

CONST.AVAILABLE\_SHIPS = ['carrier', 'battleship', 'destroyer', 'submarine', 'minesweeper'];

# D. Evaluation

<See H446-03 Project Advice Booklet for help and guidance of what must go here.>

# Project Appendixes

Insert as many project appendixes as you need for your project.

These might include, but are not limited to:

* Complete Code Listing (ESSENTIAL)
* Interview Transcripts
* Meeting notes
* Observation notes or questionnaires