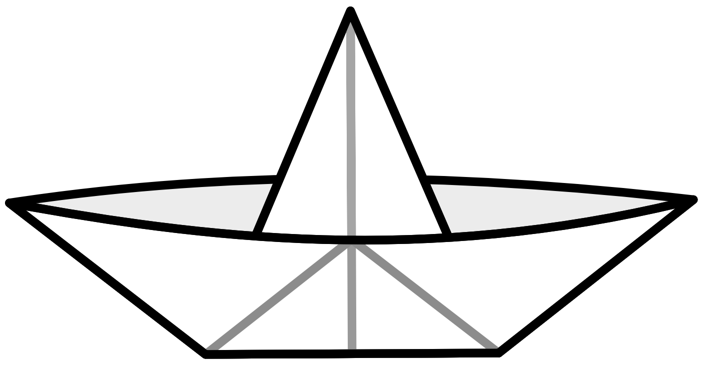


Battleboats

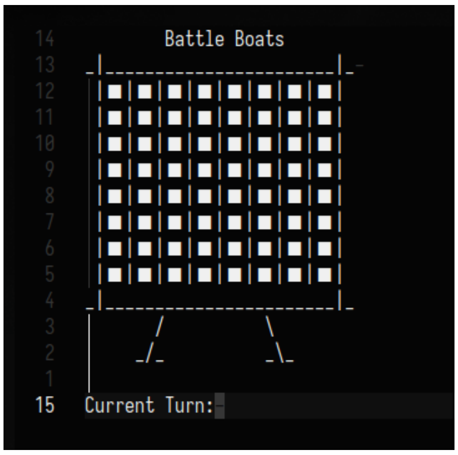
Zayaan Azam

Contents

1. Project Objectives

1. Create A Menu that allows the User to start a new game, load a pre-existing game, read the instructions or exit.
2. The program will validate and ensure that the inputs do correspond to a existing option and will capture the keypress, not the input itself.
3. Upon the 'New Game' option being selected, the user should be prompted to place an amount of boats (with a specified rotation) on a grid of size that is specified in the Constants.cs file.
4. The boats will be validated in order to ensure they do not get placed over each other, do not exceed the bounds of the grid and, have a valid rotation.
5. The program will, alongside the player, create a 'FleetMap' for the player that is used for computational purposes.
6. The program will then go on to generate the fleet for the computer following the same restrictions as the players boats.
7. The program will similarly create a 'FleetMap' for the computer that is used for the same purposes.
8. The program will then start the turn cycle, beginning with the player turn.
9. for each turn cycle, the player will choose a position to target.
10. said target is validated to ensure that it has not been used before.
11. this target will then be checked for, successively, a Hit > Sink > Victory.
12. The computer will then take their turn, performing the same targeting and checking cycle as the player.
13. The players grid will then be output to the player, showing the results of the computers turn.
14. this process will then be repeated until either victory is achieved for either person, where the game will end, or until escape is pressed.
15. At any point during the game loop, if the player were to press 'Escape', the game will save to disk and then display a in-game Menu
16. this menu will allow the player to either continue the game, read the instructions or save and exit the game.
17. as in the main menu, if the instructions option is selected the program will display the "Instructions.txt" file located in ./Documentation/ to the player
18. the player may then exit the instructions by pressing escape, returning to the menu.
19. the game will be able to be controlled via either the arrow, or WASD Keys, as detailed in the instructions

2. Planning



```
           Battle Boats
_|-----------------------|_
 |■|■|■|■|■|■|■|■|
 |■|■|■|■|■|■|■|■|
 |■|■|■|■|■|■|■|■|
 |■|■|■|■|■|■|■|■|
 |■|■|■|■|■|■|■|■|
 |■|■|■|■|■|■|■|■|
 |■|■|■|■|■|■|■|■|
 |■|■|■|■|■|■|■|■|
_|-----------------------|_
       /          \
     _/_          _\_

Current Turn:
```

Figure 1: Initial Mockup

3. Documented Design

3.1. Function Heirarchy

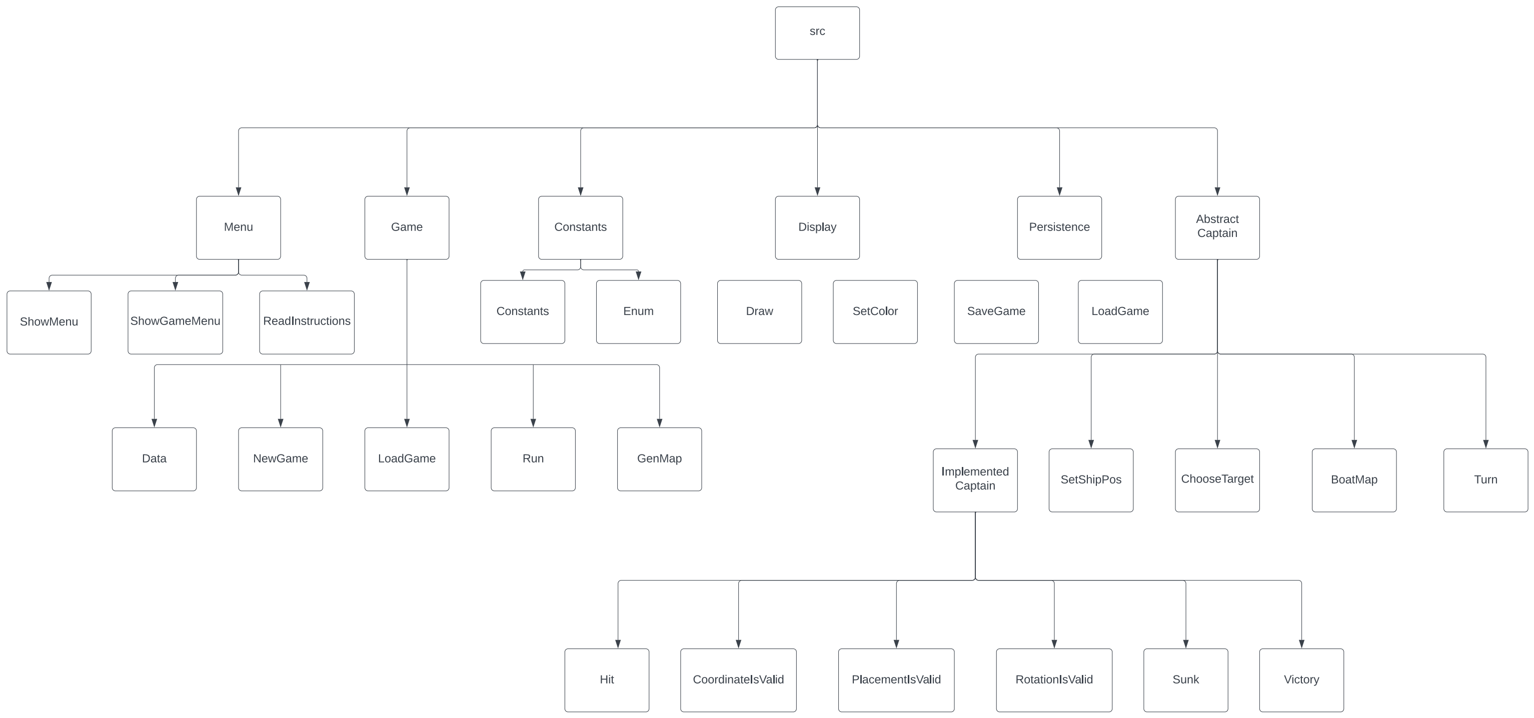


Figure 2: Software Heirarchy, In order of src > files > classes > functions. Better view can be seen in ./Documentation/Assets/heirarchy.svg

3.2. Configurable Parameters (Constants)

There are a few configurable parameters that will globally change the way the game will function. these consist of the following:

- width: the width of the playable game space
- height: the height of the playable game space
- window width: the width of the terminal, used for visual purposes, only works on windows
- window height: the height of the terminal, used for visual purposes, only works on windows
- Fleet: a list of the struct boat, used to determine whats boats are placeable for the player / computer. these can be changed and will work throughout the program, and is defined by the quantity of each boat, and the total length of said boat.

3.3. Data Structures

Tile: an enumerable, used primarily for rendering / logical purposes in order to better store the map data. Consists of the following:

- Empty1 & Empty2: logically the same, used for denoting that the cell is empty. The reasoning for 2 is to create the slight visual effect of a wave, by having a light and dark colour for the wave.
- Boat: used to denote where a boat has been placed
- Wreckage: replaces a boat, when the individual boat has been entirely hit
- Hit: Where a boat has been hit by a shot
- Miss: where a shot has missed any boats
- Using: no technical purpose, used to visually show what cell is currently being occupied by the 'cursor'

Boat:

- a struct used in the , that stores the quantity and length of each individual boat. This is used to determine what boats can be placed and the size of them.

BoatMap:

- a struct used in the , that stores, for each boat, the seed coordinate, length, rotation, and sunk status.
- This is used primarily to determine whether a boat is either sunk or if victory has been achieved.

Data:

- a class, that holds all data that would be needed to start a new game.
- this is done for easier parameter calling and function passing, whilst also being useful for the saving and loading, due to the way newtonsoft serialises json.

Captain:

- an abstract class, used to create a form of 'template' that is then further used to create both the player and computer implementations.
- An abstract class is useful as it allows you to define abstract functions; ie functions that are just declared and not initialised. it also allows you define shared private functions, that can only be accessible by classes that inherit this abstract.

3.4. Imports

Newtonsoft.json: used for elegant serialisation and deserialisation. It allows you to convert even custom types into json compared to systems solution

3.5. Saving / Loading

Done via newtonsoft.json. instead of manually creating my own formatting for saving and loading it is done through standard json syntax. All required data is put into the Data class earlier in the program and upon serialisation, it is only that class that is saved to the file (that is specified in Constants.cs).

4. Technical Solution

4.1. Foreword

The C# Code & All Documentation can be accessed at the Git Repo and further is attached in a zip file to the teams assignment. Below listed is each file with its commented code and, a brief explanation on the purpose of the file. Note that there exists a somewhat extensive version control history, however due to an incident involving a lack of patience and --force, the history is cut off abruptly around the time of a somewhat major rewrite. The code is not copied here as that would add an additional 18 pages to this document for no real benefit, as it would be a better experience to read the code through your preferred IDE. (vim).

4.2. Files

4.2.1. Program.cs

Compiler entry point, Initialises code and provides disclaimer depending on OS.

4.2.2. Menu.cs

Allows player to decide what they wish to do and learn rules of the game.

4.2.3. Constants.cs

Serves as the 'settings' for the game, also declares key types used throughout the project.

4.2.4. Abstracts.cs

Creates an abstract class implementation that is then used in Player.cs & Computer.cs

4.2.5. Persistence.cs

Provides saving and loading functionality using json

4.2.6. Display.cs

Frontend for the project, outputs results of the other files

4.2.7. Game.cs

Provides the game initialisation and loop.

4.2.8. Player.cs

Players implementation of the Captain abstract.

4.2.9. Computer.cs

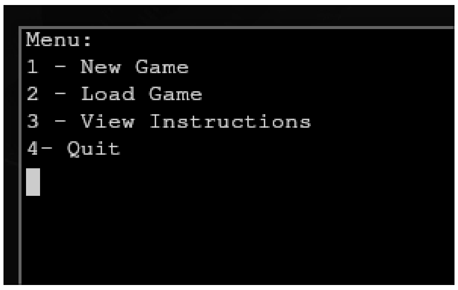
Computers implementation of the Captain abstract.

5. Testing

5.1. Foreword

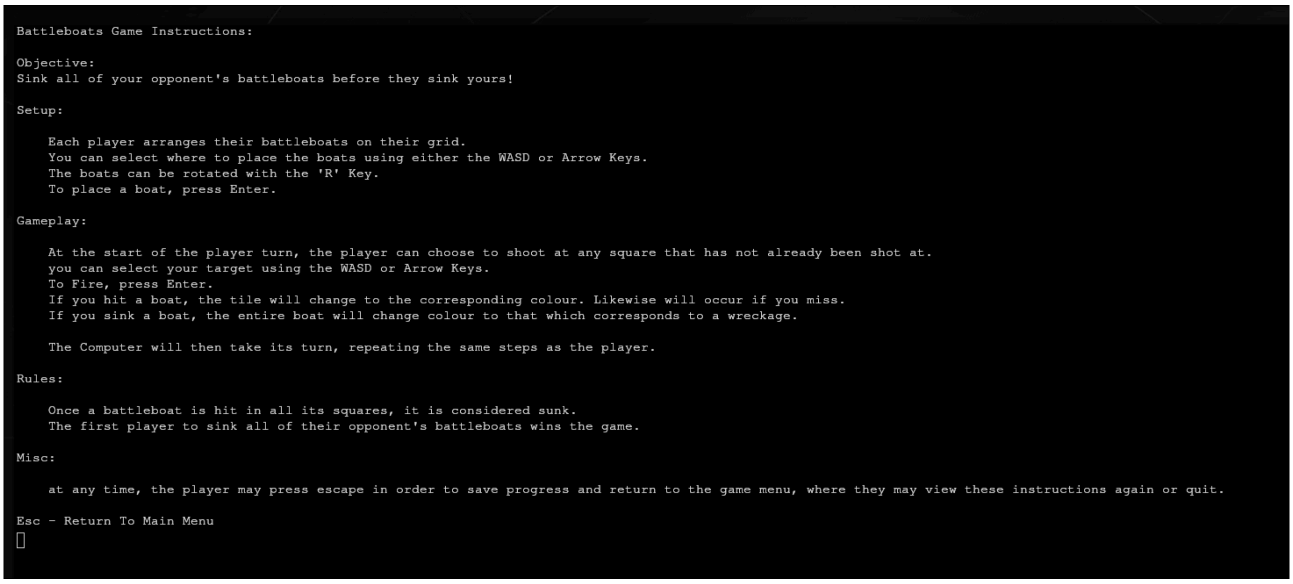
All testing is completed on Arch Linux, with an up to date set of kernel packages. it is completed on the kitty terminal emulator and is done using the default base16 colour scheme. the terminal font is monospace and should not affect testing. There is only 1 piece of code that should operate differently per platform and that has been validated by an external party.

5.2. Menu



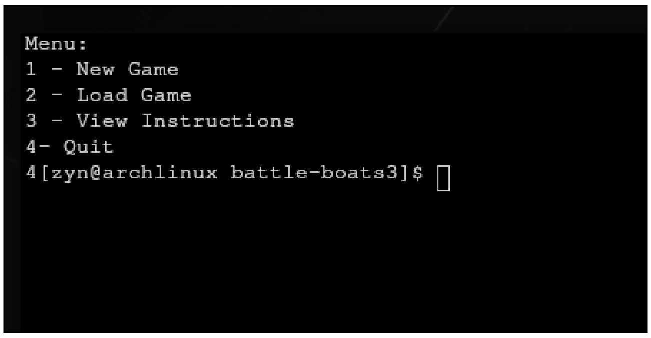
```
Menu:
1 - New Game
2 - Load Game
3 - View Instructions
4- Quit
█
```

Figure 3: Menu Outputting correctly and not reacting to invalid inputs



```
Battleboats Game Instructions:

Objective:
Sink all of your opponent's battleboats before they sink yours!

Setup:

    Each player arranges their battleboats on their grid.
    You can select where to place the boats using either the WASD or Arrow Keys.
    The boats can be rotated with the 'R' Key.
    To place a boat, press Enter.

Gameplay:

    At the start of the player turn, the player can choose to shoot at any square that has not already been shot at.
    you can select your target using the WASD or Arrow Keys.
    To Fire, press Enter.
    If you hit a boat, the tile will change to the corresponding colour. Likewise will occur if you miss.
    If you sink a boat, the entire boat will change colour to that which corresponds to a wreckage.

    The Computer will then take its turn, repeating the same steps as the player.

Rules:

    Once a battleboat is hit in all its squares, it is considered sunk.
    The first player to sink all of their opponent's battleboats wins the game.

Misc:

    at any time, the player may press escape in order to save progress and return to the game menu, where they may view these instructions again or quit.

Esc - Return To Main Menu
▯
```

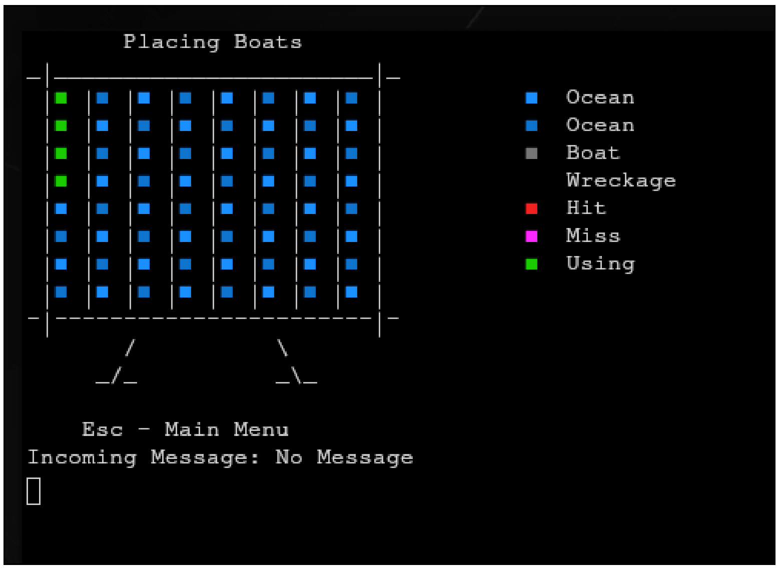
Figure 4: Instructions



```
Menu:
1 - New Game
2 - Load Game
3 - View Instructions
4- Quit
4[zyn@archlinux battle-boats3]$ ▯
```

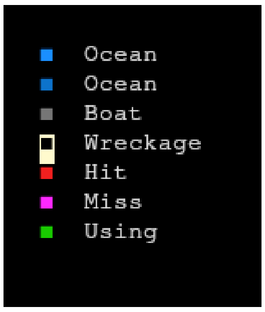
Figure 5: Exiting, can be seen returning to shell with no errors

5.3. Rendering



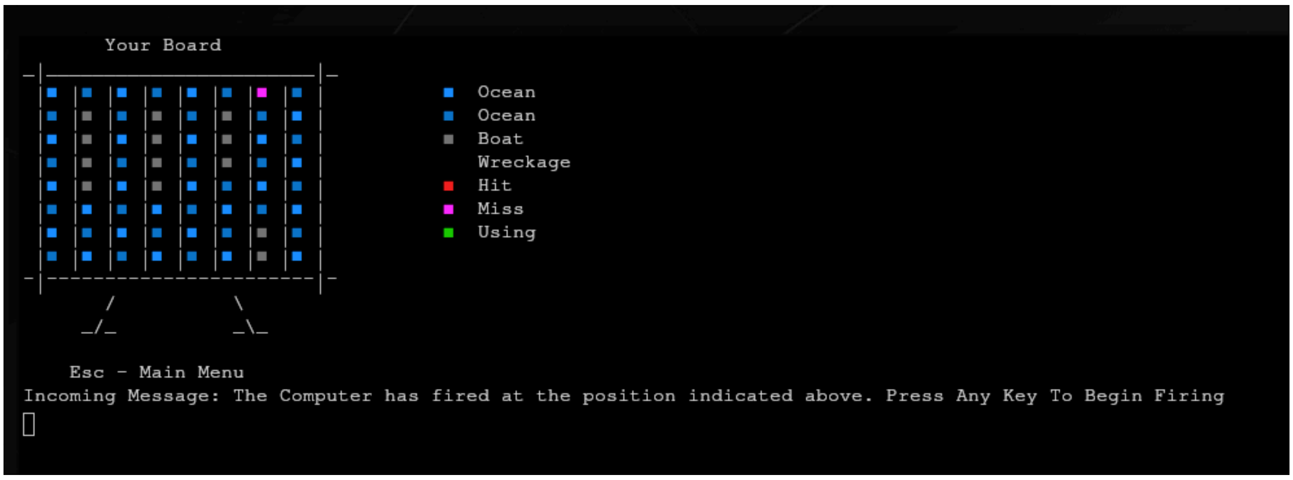
```
        Placing Boats
_|_______________________|_                  🟦 Ocean
 |🟩 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |            🟦 Ocean
 |🟩 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |            ▨ Boat
 |🟩 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |               Wreckage
 |🟩 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |            🟥 Hit
 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |            🟪 Miss
 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |            🟩 Using
 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |
 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |
_|-----------------------|_
      /           \
    _/_           _\_

    Esc - Main Menu
Incoming Message: No Message
▯
```

Figure 6: Displaying Key & Grid



```
🟦 Ocean
🟦 Ocean
▨ Boat
□ Wreckage
🟥 Hit
🟪 Miss
🟩 Using
```

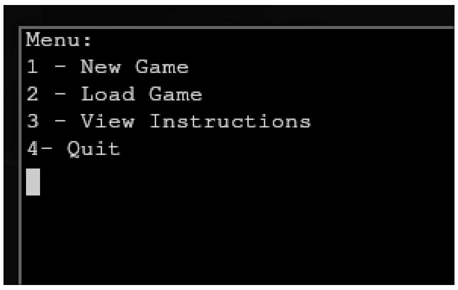
Figure 7: Example with all possible colours being displayed



```
        Your Board
_|_______________________|_                  🟦 Ocean
 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟪 |🟦 |            🟦 Ocean
 |🟦 |▨ |🟦 |▨ |🟦 |▨ |🟦 |🟦 |              ▨ Boat
 |🟦 |▨ |🟦 |▨ |🟦 |▨ |🟦 |🟦 |                 Wreckage
 |🟦 |▨ |🟦 |▨ |🟦 |▨ |🟦 |🟦 |              🟥 Hit
 |🟦 |▨ |🟦 |▨ |🟦 |▨ |🟦 |🟦 |              🟪 Miss
 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |            🟩 Using
 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |▨ |🟦 |
 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |▨ |🟦 |
_|-----------------------|_
      /           \
    _/_           _\_

    Esc - Main Menu
Incoming Message: The Computer has fired at the position indicated above. Press Any Key To Begin Firing
▯
```

Figure 8: Showcase Of Results of Computers Turn

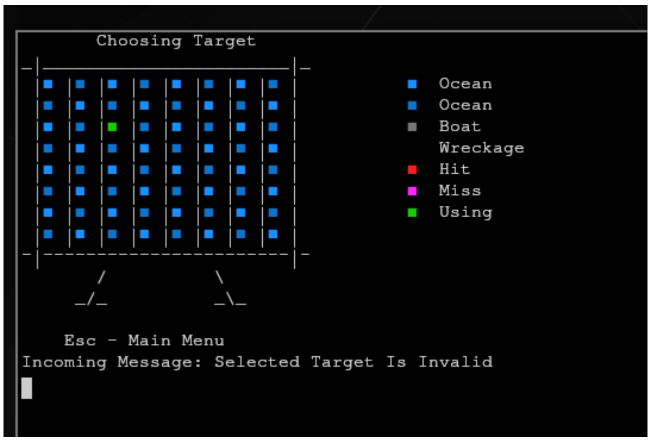
5.4. Inputs



```
Menu:
1 - New Game
2 - Load Game
3 - View Instructions
4- Quit
```

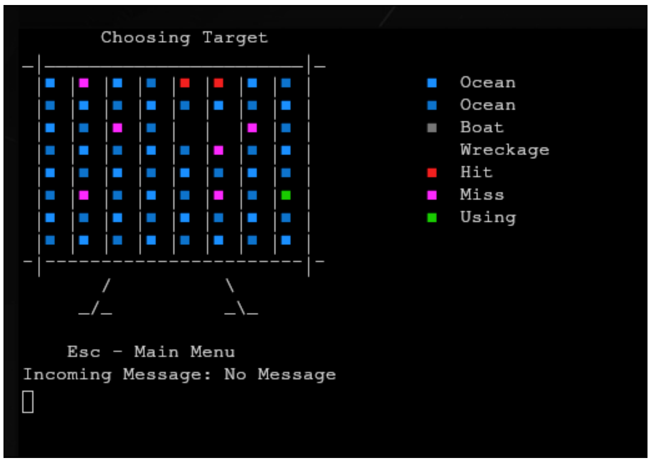
Figure 9: Extraneous Inputs Not Doing Anything, (Difficult to show via image)

5.5. Targeting



```
        Choosing Target
-|______________________|-
 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |        🟦  Ocean
 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |        🟦  Ocean
 |🟦 |🟦 |🟩 |🟦 |🟦 |🟦 |🟦 |🟦 |        ⬛  Boat
 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |            Wreckage
 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |        🟥  Hit
 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |        🟪  Miss
 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |        🟩  Using
 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |
-|----------------------|-
       /           \
     _/_           _\_

    Esc - Main Menu
Incoming Message: Selected Target Is Invalid
```

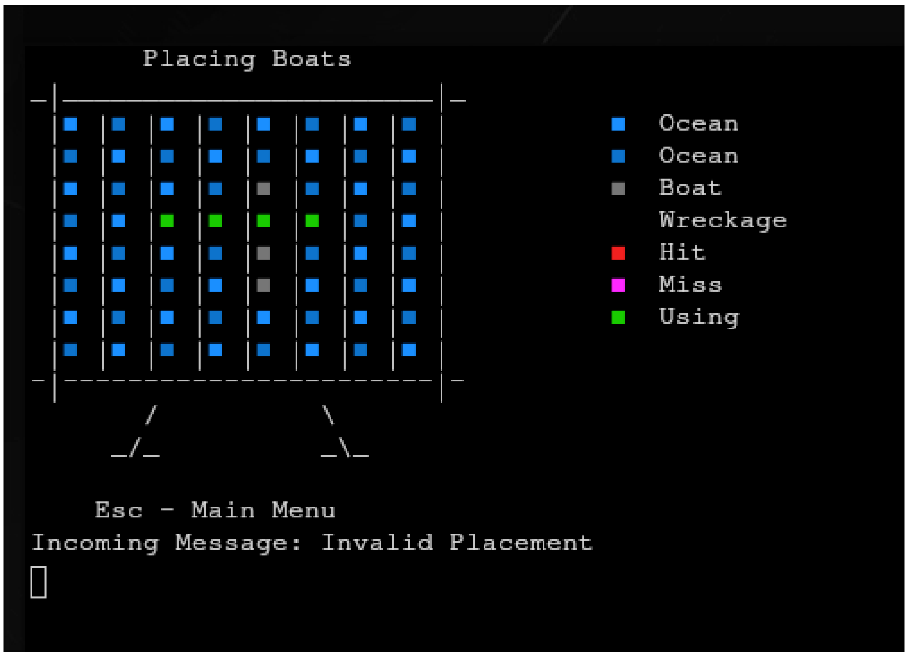
Figure 10: Validation of if target is valid, can be seen by message. Note that the highlighted cell is a miss (is clearly known when actually playing)



```
        Choosing Target
-|______________________|-
 |🟦 |🟪 |🟦 |🟦 |🟥 |🟥 |🟦 |🟦 |        🟦  Ocean
 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |        🟦  Ocean
 |🟦 |🟦 |🟪 |🟦 |   |   |🟪 |🟦 |        ⬛  Boat
 |🟦 |🟦 |🟦 |🟦 |🟦 |🟪 |🟦 |🟦 |            Wreckage
 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |        🟥  Hit
 |🟦 |🟪 |🟦 |🟦 |🟦 |🟪 |🟦 |🟩 |        🟪  Miss
 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |        🟩  Using
 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |
-|----------------------|-
       /           \
     _/_           _\_

    Esc - Main Menu
Incoming Message: No Message
```

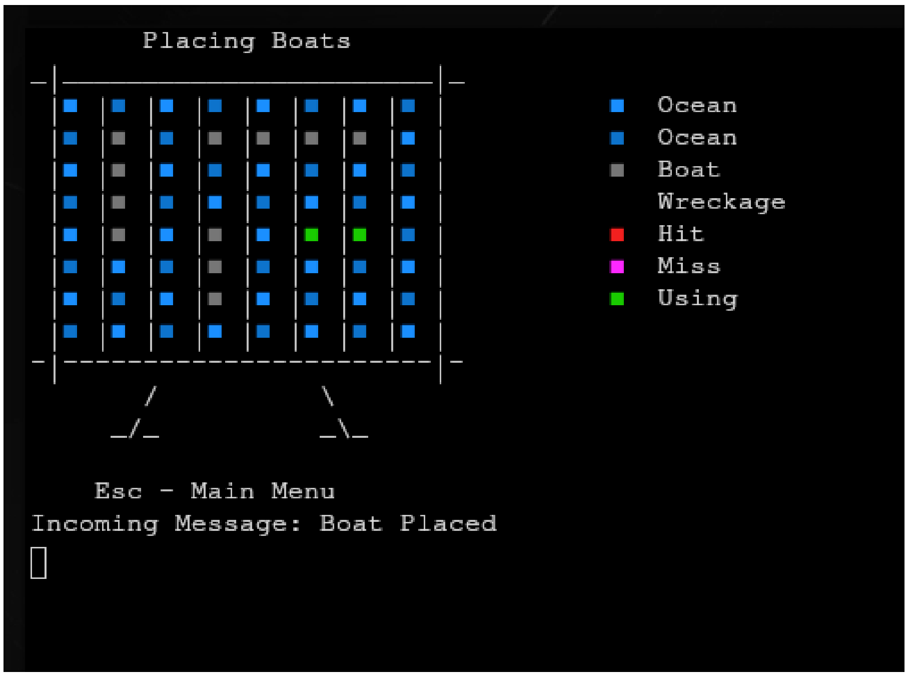
Figure 11: Target Hit, Miss & Wreckage (Sunk)

5.6. Placing



```
        Placing Boats
 _|________________________|_
  |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |          🟦 Ocean
  |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |          🟦 Ocean
  |🟦 |🟦 |🟦 |🟦 |▨ |🟦 |🟦 |🟦 |          ▨ Boat
  |🟦 |🟦 |🟩 |🟩 |🟩 |🟩 |🟦 |🟦 |            Wreckage
  |🟦 |🟦 |🟦 |🟦 |▨ |🟦 |🟦 |🟦 |          🟥 Hit
  |🟦 |🟦 |🟦 |🟦 |▨ |🟦 |🟦 |🟦 |          🟪 Miss
  |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |          🟩 Using
  |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |
 -|------------------------|-
       /           \
     _/_           _\_

    Esc - Main Menu
Incoming Message: Invalid Placement
```

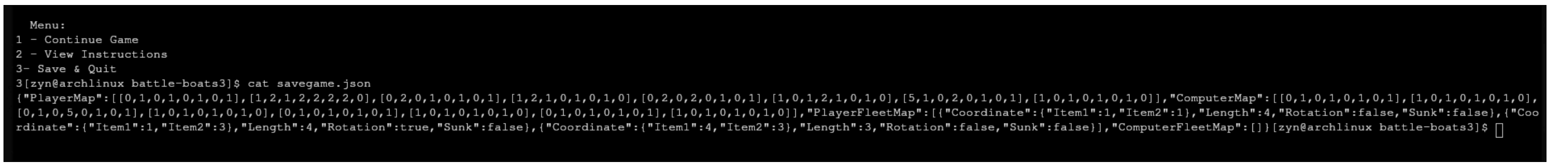
Figure 12: Overlap Detection, shown in message



```
        Placing Boats
 _|________________________|_
  |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |          🟦 Ocean
  |🟦 |▨ |🟦 |▨ |▨ |▨ |▨ |🟦 |          🟦 Ocean
  |🟦 |▨ |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |          ▨ Boat
  |🟦 |▨ |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |            Wreckage
  |🟦 |▨ |🟦 |▨ |🟦 |🟩 |🟩 |🟦 |          🟥 Hit
  |🟦 |🟦 |🟦 |▨ |🟦 |🟦 |🟦 |🟦 |          🟪 Miss
  |🟦 |🟦 |🟦 |▨ |🟦 |🟦 |🟦 |🟦 |          🟩 Using
  |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |🟦 |
 -|------------------------|-
       /           \
     _/_           _\_

    Esc - Main Menu
Incoming Message: Boat Placed
```

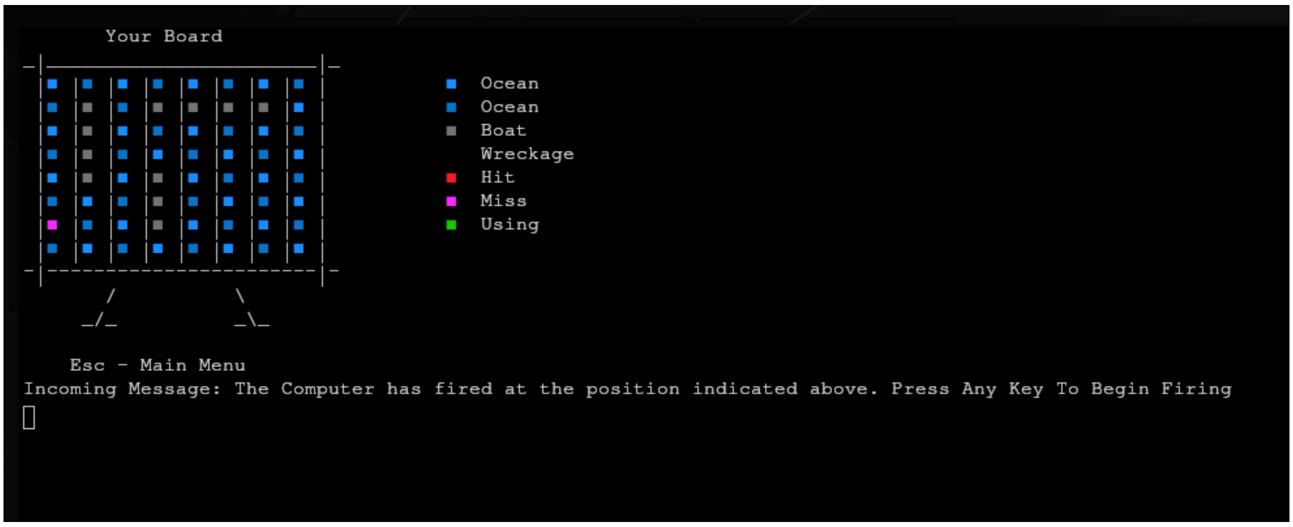
Figure 13: Differing Sizes & Rotation

5.7. Persistence



```
  Menu:
1 - Continue Game
2 - View Instructions
3- Save & Quit
3[zyn@archlinux battle-boats3]$ cat savegame.json
{"PlayerMap":[[0,1,0,1,0,1,0,1],[1,2,1,2,2,2,2,0],[0,2,0,1,0,1,0,1],[1,2,1,0,1,0,1,0],[0,2,0,2,0,1,0,1],[1,0,1,2,1,0,1,0],[5,1,0,2,0,1,0,1],[1,0,1,0,1,0,1,0]],"ComputerMap":[[0,1,0,1,0,1,0,1],[1,0,1,0,1,0,1,0],
[0,1,0,5,0,1,0,1],[1,0,1,0,1,0,1,0],[0,1,0,1,0,1,0,1],[1,0,1,0,1,0,1,0],[0,1,0,1,0,1,0,1],[1,0,1,0,1,0,1,0]],"PlayerFleetMap":[{"Coordinate":{"Item1":1,"Item2":1},"Length":4,"Rotation":false,"Sunk":false},{"Coo
rdinate":{"Item1":1,"Item2":3},"Length":4,"Rotation":true,"Sunk":false},{"Coordinate":{"Item1":4,"Item2":3},"Length":3,"Rotation":false,"Sunk":false}],"ComputerFleetMap":[]}[zyn@archlinux battle-boats3]$
```

Figure 14: Saving the game via menu & outputted savefile



```
       Your Board

        🟦 Ocean
        🟦 Ocean
        ⬜ Boat
           Wreckage
        🟥 Hit
        🟪 Miss
        🟩 Using

    Esc - Main Menu
Incoming Message: The Computer has fired at the position indicated above. Press Any Key To Begin Firing
```

Figure 15: Loading, below save file corresponds to this game which can be observed if read

```
{"PlayerMap":[[0,1,0,1,0,1,0,1],[1,2,1,2,2,2,2,0],[0,2,0,1,0,1,0,1],
[1,2,1,0,1,0,1,0],[0,2,0,2,0,1,0,1],[1,0,1,2,1,0,1,0],[5,1,0,2,0,1,0,1],
[1,0,1,0,1,0,1,0]],"ComputerMap":[[0,1,0,1,0,1,0,1],[1,0,1,0,1,0,1,0],
[0,1,0,5,0,1,0,1],[1,0,1,0,1,0,1,0],[0,1,0,1,0,1,0,1],[1,0,1,0,1,0,1,0],
[0,1,0,1,0,1,0,1],[1,0,1,0,1,0,1,0]],"PlayerFleetMap":[{"Coordinate":{"Item1":
1,"Item2":1},"Length":4,"Rotation":false,"Sunk":false},{"Coordinate":{"Item1":
1,"Item2":3},"Length":4,"Rotation":true,"Sunk":false},{"Coordinate":{"Item1":4,
"Item2":3},"Length":3,"Rotation":false,"Sunk":false}],"ComputerFleetMap":[]}
```

Listing 1: Save File Itself

6. Evaluation

Overall I think my project has solidly met or exceeded the points stated in Objectives. I also believe I completed the extensions outlined in the briefing. Throughout the development of the project there were no major problems that needed solving, with the primary issue I ran into being rendering the player 'cursor' in real time. If given the chance to do the project again, I would consider redo-ing / improving upon the following things:

- Advancing the computers targeting AI, such as making it consider the previous shots and whether they were hit and misses, or even placing the boats in a more 'strategic' way, potentially with preset patterns.
- Create a nicer, more detailed User Interface
- Create a inbuilt method to adjust the 'Settings' (the constants), so that the user can better adjust the gameplay to suit them.
- create a scaling UI, as, whilst all the technical parts 100% scale correctly, some of the visual flair does not.
- Optimisation of some particularly egregious 1 liners, primarily in some if statements.