Battleboats Project

Alexander Tahiri

Contents

1. Objectives

1. The program will present the user with a menu that allows the user to play a new game, resume a game, read instructions for the game or quit the game. The program will verify that the given menu option is valid. The program should accept an abbreviated version of the option (`"p"`), or the full version (`"play"`)
2. When the user selects the menu option to play a new game, the user will be prompted for 5 ship directions and coordinates. After entering each coordinate, the program will display the new fleet grid.
3. The program should be able to generate the computer's fleet grid by randomly selecting 5 directions and coordinates within the grid.
4. The program should verify that the player and computer generated ships do not overlap, and are within the grid.
5. The program should display the user's fleet grid and target tracker after each turn.
6. Each turn, the program should first prompt the user for their target coordinates. The program should verify these coordinates are within the grid and have not been used before. If there is a ship in the computer's fleet grid at the given coordinates, a hit should be shown at that position on the user's target tracker, otherwise a miss should be shown.
7. After the user's turn, the program should randomly select a coordinate within the grid that the computer has not used before. If there is a ship in the user's fleet grid, a hit should be shown.
8. The program should repeat the user and computer turns until all the ships in either of the fleet grids have been hit. The winner (the player that still has ships remaining), should be displayed to the user, and the program should return to the menu.
9. After each turn the program should save the current state of the game to a file.
10. When the user selects the menu option to resume a game, the state of the game should be loaded from a file and start the current player's turn.
11. When the user selects the menu option to read instructions, the program should display a clear explanation of the game and return to the menu.

2. Documented Design

2.1. Data Structures

The player and computer grids are stored using a 2D arrays of the enum `Cell`, which has 4 states:

- `Empty` - There is no ship at this position.
- `Miss` - The enemy has fired at this position, but there is no ship.
- `Ship` - There is an intact ship at this position.
- `Hit` - The enemy has hit a ship at this position.

The types of ships are stored in an array of the struct `Ship`, which has the following members:

- `int num` - The number of this type of ship.
- `int len` - The length of this ship.
- `string name` - The name of this ship. Used for instructions menu.

Coordinates are represented as tuples, and are zero-based. (`"B4" = (2, 3)`).

2.2. Constants

- `int SIZE` - The size of the grids (default 8). Values above 9 will not be rendered correctly, however this is not in the project scenario.
- `Ship[] SHIPS` - An array of the types of ships. The `Ship` struct is defined in 2.1. Data Structures.

- `string SAVE_FILE` - A path to the game's save file (default `"game.bin"`). The file is saved in a custom binary format (2.5. Save File Format) and created by the program after each turn.
- `bool SHOW_ENEMY_SHIPS` - If the program should display enemy ships on the target tracker (default false). Used for debugging.

2.3. Functions

Main

Parameters: `None`
Returns: `None`
Entry point. Shows the menu to the user until `"q"/"quit"` is entered.

Play

Parameters: `Cell[,] playerGrid, Cell[,] pcGrid`
Returns: `None`
Main gameplay loop. Expects both grids to be populated.

InputShips

Parameters: `None`
Returns: `Cell[,]`
Prompts the user to input their ships, returns their grid.

GenShips

Parameters: `None`
Returns: `Cell[,]`
Generates a grid with random ships.

VerifyAndPlace

Parameters: `Cell[,] grid, (int, int) pos, bool vert, int size`
Returns: `bool`
Verifies if a ship at position `pos`, length `size`, placed vertically if `vert == true` would fall within the size of the grid and not intersect other ships. Places the ship and returns true if the position is valid.

Fire

Parameters: `Cell[,] grid, (int,int) pos, bool player`
Returns: `bool`
If the cell at pos is `Miss/Hit` already, returns false, otherwise replaces `Empty` with `Miss` and `Ship` with `Hit` and returns true. Always prints a message for hit/miss, and prints 'coordinate already picked' if `player == true`.

DisplayGrid

Parameters: `Cell[,] grid, string title, (int, int) offset, bool player`
Returns: `None`
Prints out the given grid with the given offset relative to the current cursor. Will only display ships if `player == true`, and misses if `player == false`.

WriteGrid

Parameters: `Cell[,] grid, BinaryWriter file`
Returns: `None`
Write the given grid to a file. See 2.5. Save File Format for more details.

ReadGrid
Parameters: `BinaryReader file`
Returns: `Cell[,]`
Read a grid from the given file. See 2.5. Save File Format for more details.

ParseCoord
Parameters: `string coord`
Returns: `(int,int)`
Parse the given string (case insensitive) as a coordinate (`"B5" -> (1, 4)`). Verifies if the coordinate is within (`SIZE, SIZE`). Returns `(-1, -1)` if the coordinate is invalid.

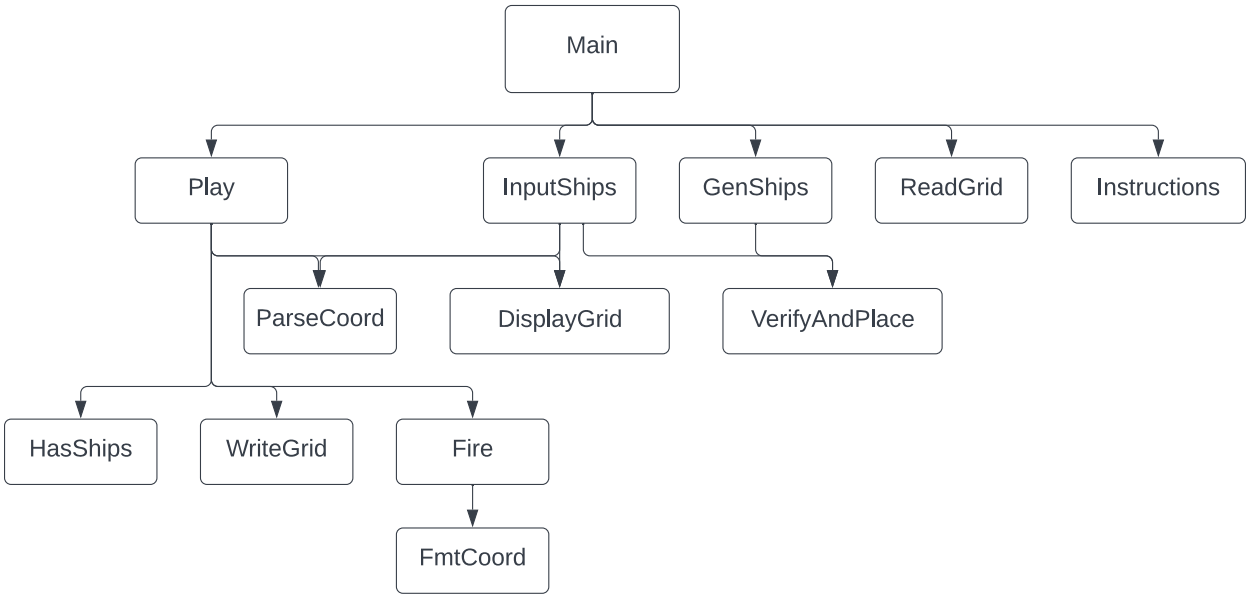
FmtCoord
Parameters: `int x, int y`
Returns: `string`
Returns a string representing the given coordinate (`(2, 3) -> "D4"`).

HasShips
Parameters: `Cell[,] grid`
Returns: `bool`
Returns true if there are ships in the given grid.

Prompt
Parameters: `string`
Returns: `string`
Display the given prompt and read 1 line from the console.

Instructions
Parameters: `None`
Returns: `None`
Prints out instructions for the game.

2.4. Function Hierarchy



2.5. Save File Format

The game save contains the player grid, then the computer grid, both saved in row-major order, using 1 byte for each `Cell`. The byte representation of each `Cell` is automatically generated by C#, starting at 0 and increasing by 1 in the order defined in 2.1. Data Structures.

3. Technical Solution

The C# source code for the project is availible on GitHub. The `"Program.cs"` file is also attached to the Teams assignment.

4. Testing

4.1. Menu

```
- Battleboats -
p - play a new game
r - resume a game
i - read instructions
q - quit
enter menu option:
unknown option ''
- Battleboats -
p - play a new game
r - resume a game
i - read instructions
q - quit
enter menu option: h
unknown option 'h'
- Battleboats -
p - play a new game
r - resume a game
i - read instructions
q - quit
enter menu option: i
instructions:
 setup your fleet grid by entering coordinates (C3, E5, ...)
 choose the direction of longer ships by entering h/v for horizontal/vertical.
 pick coordinates on the target tracker to attack
 the grid will either show hit (X) or miss (-).
 the first player to destroy all the enemy ships wins.
 ship types:
 2 x destroyer (1 cell(s))
 2 x submarine (2 cell(s))
 1 x carrier (3 cell(s))
- Battleboats -
p - play a new game
r - resume a game
i - read instructions
q - quit
enter menu option: q
```

4.2. Inputting ships

```
- Battleboats -
p - play a new game
r - resume a game
i - read instructions
q - quit
enter menu option: p
place your ships
fleet grid:
  A B C D E F G H
1 ~ ~ ~ ~ ~ ~ ~ ~
2 ~ ~ ~ ~ ~ ~ ~ ~
3 ~ ~ ~ ~ ~ ~ ~ ~
4 ~ ~ ~ ~ ~ ~ ~ ~
5 ~ ~ ~ ~ ~ ~ ~ ~
6 ~ ~ ~ ~ ~ ~ ~ ~
7 ~ ~ ~ ~ ~ ~ ~ ~
8 ~ ~ ~ ~ ~ ~ ~ ~
placing ship 1/5, length 1
enter ship coord: c3
fleet grid:
  A B C D E F G H
1 ~ ~ ~ ~ ~ ~ ~ ~
2 ~ ~ ~ ~ ~ ~ ~ ~
3 ~ ~ # ~ ~ ~ ~ ~
4 ~ ~ ~ ~ ~ ~ ~ ~
5 ~ ~ ~ ~ ~ ~ ~ ~
6 ~ ~ ~ ~ ~ ~ ~ ~
7 ~ ~ ~ ~ ~ ~ ~ ~
8 ~ ~ ~ ~ ~ ~ ~ ~
placing ship 2/5, length 1
enter ship coord: a9
invalid coordinate
enter ship coord: 4b
invalid coordinate
enter ship coord: e3
fleet grid:
  A B C D E F G H
1 ~ ~ ~ ~ ~ ~ ~ ~
2 ~ ~ ~ ~ ~ ~ ~ ~
3 ~ ~ # ~ # ~ ~ ~
4 ~ ~ ~ ~ ~ ~ ~ ~
5 ~ ~ ~ ~ ~ ~ ~ ~
6 ~ ~ ~ ~ ~ ~ ~ ~
7 ~ ~ ~ ~ ~ ~ ~ ~
8 ~ ~ ~ ~ ~ ~ ~ ~
placing ship 3/5, length 2
enter ship direction (h/v): h
enter ship coord: f2
fleet grid:
  A B C D E F G H
1 ~ ~ ~ ~ ~ ~ ~ ~
2 ~ ~ ~ ~ ~ # # ~
3 ~ ~ # ~ # ~ ~ ~
```

```
4 ~ ~ ~ ~ ~ ~ ~ ~
5 ~ ~ ~ ~ ~ ~ ~ ~
6 ~ ~ ~ ~ ~ ~ ~ ~
7 ~ ~ ~ ~ ~ ~ ~ ~
8 ~ ~ ~ ~ ~ ~ ~ ~
placing ship 4/5, length 2
enter ship direction (h/v): v
enter ship coord: d8
invalid placement
fleet grid:
  A B C D E F G H
1 ~ ~ ~ ~ ~ ~ ~ ~
2 ~ ~ ~ ~ ~ # # ~
3 ~ ~ # ~ # ~ ~ ~
4 ~ ~ ~ ~ ~ ~ ~ ~
5 ~ ~ ~ ~ ~ ~ ~ ~
6 ~ ~ ~ ~ ~ ~ ~ ~
7 ~ ~ ~ ~ ~ ~ ~ ~
8 ~ ~ ~ ~ ~ ~ ~ ~
placing ship 4/5, length 2
enter ship direction (h/v): v
enter ship coord: d7
fleet grid:
  A B C D E F G H
1 ~ ~ ~ ~ ~ ~ ~ ~
2 ~ ~ ~ ~ ~ # # ~
3 ~ ~ # ~ # ~ ~ ~
4 ~ ~ ~ ~ ~ ~ ~ ~
5 ~ ~ ~ ~ ~ ~ ~ ~
6 ~ ~ ~ ~ ~ ~ ~ ~
7 ~ ~ ~ # ~ ~ ~ ~
8 ~ ~ ~ # ~ ~ ~ ~
placing ship 5/5, length 3
enter ship direction (h/v): h
enter ship coord: d5
fleet grid:         tracker grid:
  A B C D E F G H     A B C D E F G H
1 ~ ~ ~ ~ ~ ~ ~ ~   1 ~ ~ ~ ~ ~ ~ ~ ~
2 ~ ~ ~ ~ ~ # # ~   2 ~ ~ ~ ~ ~ ~ ~ ~
3 ~ ~ # ~ # ~ ~ ~   3 ~ ~ ~ ~ ~ ~ ~ ~
4 ~ ~ ~ ~ ~ ~ ~ ~   4 ~ ~ ~ ~ ~ ~ ~ ~
5 ~ ~ ~ # # # ~ ~   5 ~ ~ ~ ~ ~ ~ ~ ~
6 ~ ~ ~ ~ ~ ~ ~ ~   6 ~ ~ ~ ~ ~ ~ ~ ~
7 ~ ~ ~ # ~ ~ ~ ~   7 ~ ~ ~ ~ ~ ~ ~ ~
8 ~ ~ ~ # ~ ~ ~ ~   8 ~ ~ ~ ~ ~ ~ ~ ~
enter target coord:
```

4.3. Resuming

```
- Battleboats -
p - play a new game
r - resume a game
```

```
i - read instructions
q - quit
enter menu option: r
fleet grid:         tracker grid:
  A B C D E F G H     A B C D E F G H
1 ~ ~ ~ ~ ~ ~ ~ ~   1 ~ ~ ~ ~ ~ ~ ~ ~
2 ~ ~ ~ ~ ~ # # ~   2 ~ ~ ~ ~ ~ ~ ~ ~
3 ~ ~ # ~ # ~ ~ ~   3 ~ ~ ~ ~ ~ ~ ~ ~
4 ~ ~ ~ ~ ~ ~ ~ ~   4 ~ ~ ~ ~ ~ ~ ~ ~
5 ~ ~ ~ # # # ~ ~   5 ~ ~ ~ ~ ~ ~ ~ ~
6 ~ ~ ~ ~ ~ ~ ~ ~   6 ~ ~ ~ ~ ~ ~ ~ ~
7 ~ ~ ~ # ~ ~ ~ ~   7 ~ ~ ~ ~ ~ ~ ~ ~
8 ~ ~ ~ # ~ ~ ~ ~   8 ~ ~ ~ ~ ~ ~ ~ ~
enter target coord: c3
hit C3
computer's turn:
hit D5
fleet grid:         tracker grid:
  A B C D E F G H     A B C D E F G H
1 ~ ~ ~ ~ ~ ~ ~ ~   1 ~ ~ ~ ~ ~ ~ ~ ~
2 ~ ~ ~ ~ ~ # # ~   2 ~ ~ ~ ~ ~ ~ ~ ~
3 ~ ~ # ~ # ~ ~ ~   3 ~ ~ X ~ ~ ~ ~ ~
4 ~ ~ ~ ~ ~ ~ ~ ~   4 ~ ~ ~ ~ ~ ~ ~ ~
5 ~ ~ ~ X # # ~ ~   5 ~ ~ ~ ~ ~ ~ ~ ~
6 ~ ~ ~ ~ ~ ~ ~ ~   6 ~ ~ ~ ~ ~ ~ ~ ~
7 ~ ~ ~ # ~ ~ ~ ~   7 ~ ~ ~ ~ ~ ~ ~ ~
8 ~ ~ ~ # ~ ~ ~ ~   8 ~ ~ ~ ~ ~ ~ ~ ~
enter target coord: d5
miss D5
computer's turn:
miss G8
fleet grid:         tracker grid:
  A B C D E F G H     A B C D E F G H
1 ~ ~ ~ ~ ~ ~ ~ ~   1 ~ ~ ~ ~ ~ ~ ~ ~
2 ~ ~ ~ ~ ~ # # ~   2 ~ ~ ~ ~ ~ ~ ~ ~
3 ~ ~ # ~ # ~ ~ ~   3 ~ ~ X ~ ~ ~ ~ ~
4 ~ ~ ~ ~ ~ ~ ~ ~   4 ~ ~ ~ ~ ~ ~ ~ ~
5 ~ ~ ~ X # # ~ ~   5 ~ ~ ~ - ~ ~ ~ ~
6 ~ ~ ~ ~ ~ ~ ~ ~   6 ~ ~ ~ ~ ~ ~ ~ ~
7 ~ ~ ~ # ~ ~ ~ ~   7 ~ ~ ~ ~ ~ ~ ~ ~
8 ~ ~ ~ # ~ ~ ~ ~   8 ~ ~ ~ ~ ~ ~ ~ ~
enter target coord:
```

4.4. Win Condition

```
- Battleboats -
p - play a new game
r - resume a game
i - read instructions
q - quit
enter menu option: r
fleet grid:         tracker grid:
```

```
  A B C D E F G H     A B C D E F G H
1 ~ ~ ~ ~ ~ ~ ~ ~   1 ~ ~ ~ ~ ~ ~ X X
2 ~ ~ ~ ~ ~ # # ~   2 ~ ~ ~ ~ ~ ~ ~ ~
3 ~ ~ # ~ # ~ ~ ~   3 ~ ~ X ~ ~ ~ ~ ~
4 ~ ~ ~ ~ ~ ~ ~ ~   4 X ~ ~ ~ ~ ~ ~ ~
5 ~ ~ ~ X # # ~ ~   5 ~ ~ ~ - ~ ~ ~ X
6 ~ ~ ~ ~ ~ ~ ~ ~   6 ~ ~ ~ ~ ~ ~ X X
7 ~ ~ ~ # ~ ~ ~ ~   7 ~ ~ ~ ~ ~ ~ X ~
8 ~ ~ ~ # ~ ~ ~ ~   8 ~ ~ ~ ~ ~ ~ ~ ~
enter target coord: h4
hit H4
you won
- Battleboats -
p - play a new game
r - resume a game
i - read instructions
q - quit
enter menu option:
```

5. Evaluation

Overall, my project was successful and met all of the criteria outlined in 1. Objectives, including the extensions. I didn't encounter any major problems whilst implementing the program. If I were to revisit the project, I would make the following improvements:

- Computer targeting could use a more complex algorithm that considers the previous hits, attempting adjacent cells.
- The user interface could be made more visually appealing with colours and ASCII art, or even a GUI.
- The error handling on file loading on file loading could be improved.