EECS 448 – Software Engineering I

Project 01

(Two-Player Battleship Game)

Documentation

Group 08:
Junyi Zhao
Zhenzhou Wang
Mir Shazil Faisal
Huzaifa Zahid

1. **Introduction:**

Battleship is a two-player game wherein both players secretly place 1 to 5 ships on a 10x10 grid.
Each player takes turns and announces where on the opponent's grid they wish to fire.
The opponent then announces whether one of his ships was hit and if the whole ship is sunk or not.
The first player to sink all the opponent's ships wins.

2. **Background:**

Initially our team decided on making the game in Python and so the duties were evenly divided upon the group members. Huzaifa and Mir were responsible for the backend programming and Zhenzhou and Junyi were responsible for the frontend. However, after a couple of days of trying to familiarize themselves with Python, Huzaifa and Mir decided to stick to C++ for the backend programming because they were most familiar with that language. Additionally, after having written the code in C++, they can bind that code to Python which would allow successful linking of the backend to the frontend and would allow the game to function without any issues.

The following links were used as guidelines for binding the C++ code to Python:

https://pybind11.readthedocs.io/en/stable/basics.html

https://github.com/pybind/pybind11

Once the repository is cloned (or pulled after any updates, if the repository was cloned previously), going into class_example folder inside Backend and writing the following line of code on the terminal

>>> pip3 install .

will install the python's package manager in that folder and running the following command shows that the binding worked successfully:

>>> python3 test.py

Further functions can be added/changed in the test.py file to play the game. The following commands show the sequence of commands to be passed:

>>> git clone https://github.com/hzahid99/EECS448_Project01.git

>>> cd Backend/Backend/class_example

>>> pip3 install .

>>> python3 test.py

3. **Code Details:**

The backend code is broken into two classes mainly, the Board class and the Executive class.

The Board Class is responsible for the following tasks:

- Creating and updating the 10x10 grid.
- Placing and keeping track of where the ships have been placed.
- Printing the boards (battleship grids) with and without ships on them.
- Validating that legal placement of ships and hits are made.
- Checking whether a ship has been destroyed or not.
- Updating the grid after hits.

The Executive class is responsible for the following tasks:

- Taking inputs from the users.
- Taking care of the commands/instructions outputted on the terminal.
- Creating both the players.
- Calling methods from the Board class according to the need.

The main.cpp file just creates an Executive object and calls the run() method which makes the game work.

The makefile simply compiles and links all the files together.

Junyi Zhao and Zhenzhou Wang worked on the frontend of the project. Junyi's code works on the first part of the GUI - placing ships. The code allows players to choose how many ships they will enter, and to enter the ships at the place they wish.

The second part of the game (playing) is done by Zhenzhou. His code shows the ship boards and allows the user to interact and play with it. Players click on the target they want to fire at, and the GUI gives feedback accordingly, indicating whether they hit the ship or missed.

The individual files are heavily commented and show what each part of the code is supposed to be doing.

4. Conclusion

Conclusively, even though our individual components work, we were unable to create a GUI based implementation of the game and ended up with a terminal-based game instead.