Project 1 Retrospective Write-up for “Big SegFault Energy”

Team Members:

Chance Penner
Haonan Hu
Markus Becerra
Sarah Scott
Thomas Gardner

Git Link:
https://github.com/MarkusBecerra/BattleShip

Date finished: September 20, 2019

Group Meeting log:

Meeting 1:
09/06/2019 @ 11:47AM to 11:50AM
Location: Eaton 2
All in attendance.
Agenda:
*Chose C++ as language for project.
*Set next meeting date.

Meeting 2:
09/07/2019 @ 2:40PM to 3:30PM
Location: Alcove 1326 in LEEP2
All in attendance.
Agenda:
*Brainstorming
*Ran through simulation
*Started Class Creation (pseudocode)
*Set up potential next meeting.

Meeting 3:
09/09/2019 @11:40AM to 11:50AM
Agenda:
*Discussed changes we made since last meeting
*Planned to meet in lab in two days

Meeting 4:
09/11/2019 @ 9AM to 10:50AM
Location: Eaton Hall in a Lab Room
All in attendance
Agenda:
*Created Board and Player classes

Meeting 5:
09/13/2019 11:30AM to 11:50AM
Location: Eaton Lawr2
All in attendance.
Agenda:
*Discussed when to meet next and what to focus on

Meeting 6:
09/14/2019 @ 2:30PM to 4:00PM

Location: LEEP2 1322
All in attendance.
Agenda:
*Outlined classes and their functions
*Added class functions
*Mapped out the flow of the program, step by step

Meeting 7:
09/16/2019 @ 4:00 PM to 6:00PM
Location: LEEP2 1322
Haonan and Chance
Agenda:
*try and catch implementation
*Get the board operating without ships

Meeting 8:
09/18/2019 @ 9:00AM to 10:50AM
Location: Eaton 1005C
All in attendance
Agenda:
*Everyone works on Ship set up together
*Now ships can be correctly placed on the board
*Some methods left in Ship class need be defined(*****Important*****)

Meeting 9:
09/20/2019 @ 10:00AM to 11:00AM
Location: Eaton 1005A
Chance, Thomas, Sarah, Markus
Agenda:
*Worked on input verification
*Worked on ship class keeping track of destroyed ships

Meeting 10:
09/20/2019 @ 12:00PM to 2:00PM
Location: LEEP2
Chance, Thomas
Agenda:
*Worked on damage counter and game over
*Boat verification input

Meeting 11:

09/20/2019 @ 4:10PM to 5:30PM
Location: LEEP2
Chance, Thomas
Agenda:
*Completed game over functionality
*Add input verification for coordinates

Meeting 12:
09/21/2019 @ 3:00PM to 6:13PM
Location: LEEP2
Chance, Markus
Agenda:
*Fixed existing bugs
*Fixed newly created bugs
*Fixed output placement
*Added some features

Meeting 13:
09/21/2019 @ 6:13PM to 7:13PM
Location: Eaton
Agenda:
*Checked Valgrind and fixed ALL MEMORY LEAKS AND ERRORS

Work Distribution:

Our project is formed by 4 classes: Ship class, Board class, Player class and Executive class, we split work by class, Chance is mainly in charge of Board class, he created default board layout and helped other people with their methods implementation as well. Thomas is doing Executive Class, which prints the user interface, main menu and battleship game rule. Haonan implemented Player Class, it controls player's movement like shoot or get shot by other player, he also did documentation for the group. Sarah helped doing players shooting function and make sure board is being updated. Markus did most of Ship related methods and put the ship tracker on board.

Overall, while some did stick to mainly their classes, each person had to work is multiple classes at a time. We did end up splitting the work fairly evenly in terms of amount of functions implemented and code written, but in terms of splitting up by classes, that did not really happen, which was fine. It was more comfortable and helpful having people work on similar things at the same time in order to get the best implementation, rather than forcing each person to work on an entire class by themselves.

Challenges:

First of all, since this is our first group project, everyone is excited about it and they shared a lot of brilliant ideas for setting up the game, and here I mean a lot! We just have too many thoughts about this game, which increased difficulty for everyone staying on the same page. For example, we nearly spent most of time discussing what features we should add into the game in one of our meeting, which was super fun, because everyone got involved. But at the end of meeting, we barely had any thoughts in common and didn't have a specific plan for game setting up.

Secondly, due to the different schedule, we cannot meet as frequent as we can. Our common schedule would be Saturday for about 1 to 2 hours, which could be relatively short to work together. Instead, we decided to meet as small group, it helps a lot. During a small group meeting, people work on problems that they have on their coding. Thanks to it, we solved a lot of critical problems in our program and kept the game functional.

Collaboration between team members is definitely one of the challenges. We are all new to the GitHub platform, we made a lot mistakes when we were using it. We have to constantly keep each other up to date on what we were working with. Every time someone is trying to push their work to master branch, he/she will text other group members in GroupMe and avoid merge conflict. It is a little annoying at the first time, because you have to stop your work and pull others' work first, but we all get used to it.

Unfinished Features:

We had a lot features that we wanted to add into our games at the second meeting, here is some of them:

1) Player name customization
2) Different colors for each ship
3) Play again feature for after a player wins

Features to be added after demo

Features we knew we would not have time for, but would gladly implement after due date:

1) We wanted to add sound effect to ship actions
2) Player can control their ships with WASD
3) High score output file for lowest turns to win per game mode
4) Turn timer to make players choose quickly
5) Game mode for having another turn each time you successfully hit a ship
6) VS Computer mode

Would have done different:

After our first project, we realized that we still have a lot to improve. Communication between team members must be more frequent and detailed. We need some well-planned ideas, not just saying what the end product should do, it can save a lot of time during the meeting. Also, we need to focus on basic functionality of the program first, once we achieved that, then we can move on to next stage, which is making the program looks fancy.

Another approach that would be different is the division of work. For this program, we initially have four classes, but we have 5 people in the group, so splitting work and make everyone is doing same amount of job is really hard. One example for this imbalance is the Board class. It has the most methods and the most complicated implementation. No one in our group wanted to do it alone, but once everyone else picked their job, the Board class became left over. And then we had to work on this class together, which became a chaos situation. On next project, we need to spend more time on splitting tasks, class-based work distribution should not be accepted. It could cause imbalanced work and conflict.

Another idea to help with the workflow is to create branches in Git, rather than always pushing to the master branch. There was one instance where Chance wanted to push some code to the Executive class, but first had to pull what Sarah had added to the class. Because they both were in the master branch, Chance pulled Sarah's changes and that merged hers with his, making it nearly impossible to tell whose code was who's. Had Chance made an experimental branch first, he could have pushed his changes without any conflicts, and had a 'save state' of his code to reference when then merging into the master branch.

Works Cited:

Press Enter to continue feature:
https://stackoverflow.com/questions/903221/press-enter-to-continue

Colored text output in terminal:
https://stackoverflow.com/questions/2616906/how-do-i-output-coloured-text-to-a-linux-terminal

Converting character to integer:
https://stackoverflow.com/questions/5029840/convert-char-to-int-in-c-and-c

Converting a string to uppercase:
https://stackoverflow.com/questions/735204/convert-a-string-in-c-to-upper-case

Checking valid input with a string:
http://www.cplusplus.com/reference/stdexcept/out_of_range/