EECS 448 Project 1 Retrospective

Runtime Terrors (Abby Davidow, Anissa Khan, Grant Schnettgoecke, Jacob Swearingen, Chongzhi Gao)

Meeting Log

9/6	Spahr	Everyone	Decided on C++ and started thinking about design layout
9/9	Spahr	Everyone	Came up with the goal to finish initial design layout and divide up work
9/9	1005C Eaton	Everyone	Divided work by classes to create
9/10	Fishbowl	Anissa and Grant	Worked on Game class
9/11	LEEP Atrium/ Fishbowl	Anissa, Abby, and Grant	Talked about Game class function and new ship class
9/12	Fishbowl	Abby and Jacob	Talked and worked on Ship class
9/13	Fishowl	Everyone	Worked on ship class and game class
9/16	Spahr	Everyone	Discussed plan for work during lab time
9/16	1005C Eaton	Everyone	Worked together on setup and run and their helper functions
9/18	Spahr	Abby, Anissa, Grant, George	Need to finish some last touches
9/20	Fishbowl	Everyone	Worked on testing, documentation, and reflection

Division of Labor

For the Battleship project, we decided to divide the game into four classes: main, Game, Board, and Ships. The Board class was written by Abby. Anissa and Grant primarily worked on the Game class with some assistance from Jacob, George, and Abby. Jacob created the Ships class with assistance from Abby. George worked on the main.cpp file. George and Jacob worked on the Player class, which was later removed from the project as it was primarily used only as a way to access the player boards. In hindsight, it may have been worth keeping the Player class in the project but was ultimately too costly to add back. Everyone documented the classes and methods they created, and Abby generated the html documentation using doxygen. We all wrote this retrospective.

Challenges

Turning Git into an asset instead of a roadblock took some time to accomplish. Merge conflicts and their ascii remains often caused problems with compilation. Sometimes merge conflict remains were accidentally pushed to Github causing further confusion.

On GitHub, our team created a total of 2 branches for Game class and Ships class. The Game class was originally created to work primarily on that class. Ships class was created and merged into master quickly, but as the project proceeded, the Game class branch became the main branch for new changes, with everyone contributing to it, which led to more merge conflicts when pushing to that branch. The Game class branch ended up acting as the master branch until the end of the project when we merged the Game class branch with the true master branch. Going forward we think it would be wise for individuals to work on separate branches unless they are working on the same class. One difficulty that we haven't figured out how to navigate is: how to work on an individual branch but also get all the updates from other team members' branches that may contain vital class updates needed to test one's own

branch. Overall, we are all much more comfortable using Github but definitely need more practice.

A good balance for the division of labor was also difficult to find. We divided up the responsibility generally by classes, but some classes were bigger than others so the work was not distributed as equally as it could have been. Since the project was relatively small scale, it didn't require too many classes, with some classes being smaller than others. We tried to find more methods for people to write in the longer game class, but that proved difficult because of the others' need to use and test the methods as they wrote them.

It was also a challenge to divide work in such a way that each person's responsibilities were not dependent on others' work. For example, we found it difficult to test the game class without having the board class and helper methods in place.

Having a larger group made communication difficult. We found that at times it was easier for a few team members to meet last minute after a class to go over certain features or class additions. The outcomes of these meetings were generally very productive, but ultimately did not include all team members. We began using GroupMe more towards the end of the project, which began improving communication.

Features

Just about every feature that we wanted was implemented into the demo version. We wanted to mainly focus on getting the game to work properly first and then if we had extra time, add additional features. One feature that we could have added is sound to the game. For example, every time a player gets a hit, it would make a cool destruction sound. All the user interaction with our game is through the console, one thing we considered doing was some research into building a graphical user interface to make the user experience more enjoyable. We did however use some clever ascii art that displays in the terminal whenever the game starts and when it's the next player's turn. With this being said, we are happy with our final version and what we have accomplished in time frame that we were given.

Retrospective

Overall, we think our project was a success measured by our primary deliverable--the game. Our team was able to complete the functionality of the game with multiple days left before the due date allowing plenty of time for testing and bug fixing. We do however think there are many improvements that can be made in terms of our group dynamics--division of labor, and communication being a few examples.

When we planned the outline of classes for this project, we were not specific enough in the function and methods needed in each class. This led to an imbalance of work between team members. For future projects, we need to brainstorm methods, hours of work, and lines of code needed to implement each class in order to properly assign a similar amount of responsibility to each team member. If we would have tried to really dive into what would be included in the Game class, we could have foreseen how many methods it would need. The numerous check methods in Game could have been placed into another class focused on verifying values and inputs.

Going forward, we know that it is necessary that we improve our method of communication. Sometimes a couple members of the group were able to discuss the project during other classes, which were constructive, but the outcomes failed to be announced to everyone. We determined that it would be best to give a thorough summary of these meetings to all members via email, messaging, or in person during class, so everyone remains on the same page and in the loop. One area of communication in which succeeded was scheduling in-person meetings. We managed to have almost everyone meet together outside of class every couple of days. These meetings were very productive and helped us update our to-do list as we completed tasks.