EECS 448

Project 1 Battleship Retrospective

https://www.pexels.com/photo/group-hand-fist-bump-1068523/

Team: **Return Awesome**

Ethan Brenner

Kyle Kappes-Sum

Connie Li

Malena Schoeni

Archana Ramakrishnan

Log of all meetings

1. **9/6/2019. In lecture.**
    a. Attendees: Archana Ramakrishnan, Kyle Kappes-Sum, Ethan Brenner, Malena Schoeni, Connie Li
    b. Decided to make the project a webpage using JavaScript/HTML/CSS.
2. **9/9/2019. In lecture.**
    a. Attendees: Archana Ramakrishnan, Kyle Kappes-Sum, Ethan Brenner, Malena Schoeni, Connie Li
    b. Discussed plans for our JS object-oriented program design (grid and ship objects), interface design.
3. **9/9/2019. In lab.**
    a. Attendees: Kyle Kappes-Sum, Connie Li
    b. Initialized project repo and started master log. Mocked up class/object structure and interface structure to pitch to entire group at next meeting.
4. **9/9/2019. Spahr Library**
    a. Attendees: Archana Ramakrishnan, Kyle Kappes-Sum, Ethan Brenner, Malena Schoeni, Connie Li
    b. Decided on class/object structure, divided tasks mostly by object class
5. **9/10/2019. LEEP2.**
    a. Attendees: Kyle Kappes-Sum, Ethan Brenner, Malena Schoeni, Connie Li
    b. Started work on our respective tasks, using JS class declarations.
6. **9/12/2019. LEEP2.**
    a. Attendees: Ethan Brenner, Connie Li
    b. Figured out how to merge remote branches properly.
7. **9/13/2019. In lecture, scrum.**
    a. Attendees: Archana Ramakrishnan, Kyle Kappes-Sum, Ethan Brenner, Malena Schoeni, Connie Li
    b. Scrum, discussed progress and future plans.
8. **9/13/2019. Fish Bowl**
    a. Attendees: Archana Ramakrishnan, Ethan Brenner, Malena Schoeni
    b. Discussed class design, UI elements, user interaction.
9. **9/14/2019. LEEP2**
    a. Attendees: Archana Ramakrishnan, Kyle Kappes-Sum, Ethan Brenner, Malena Schoeni, Connie Li
    b. Modified class definitions (merged Fleet class functionality into Admiral), clarified which functionality belongs to which class, resolved naming/labeling conventions for coordinates. Made progress on our respective class methods.
10. **9/16/2019. In lecture, scrum.**
    a. Attendees: Archana Ramakrishnan, Kyle Kappes-Sum, Ethan Brenner, Malena Schoeni, Connie Li

    b. Scrum, discussed progress and future plans. Showed ship placement UI prototype to Dr. Gibbons.

11. 9/16/2019. In lab.

    a. Attendees: Kyle Kappes-Sum, Connie Li
    b. Began writing/running testing code, starting with testing Ship & Grid.

12. 9/16/2019. Spahr Library

    a. Attendees: Archana Ramakrishnan, Kyle Kappes-Sum, Ethan Brenner, Malena Schoeni, Connie Li
    b. Continued front end and GUI development; continued testing Grid; completed testing of Ship.

13. 9/20/19. Eaton.

    a. Attendees: Kyle Kappes-Sum, Ethan Brenner, Malena Schoeni, Connie Li
    b. Impromptu scrum, post-lecture. Discussed recent bugs, fixes, testing, and features still in progress.

14. 9/20/19. Fish Bowl

    a. Attendees: Archana Ramakrishnan, Ethan Brenner
    b. Continued front end/ GUI development; bug troubleshooting

15. 9/21/19 LEEP2

    a. Attendees: Archana Ramakrishnan, Ethan Brenner, Kyle Kappes-Sum, Malena Schoeni, Connie Li
    b. Hack-a-thon style code as much as possible -- connect back end and front end, test and fix game functionality, resolve session storage issues, general bug fixes.

16. 9/22/19 Watkins Scholarship Hall

    a. Attendees: Archana Ramakrishnan, Malena Schoeni
    b. Worked on hit/miss interface and what happens when a ship is sunk. Final bug fixes

How work was split between teammates:

In one of our first meetings, we diagramed out how we wanted the battleship game to work and what classes we would have and what their role would be. The classes we decided to have were Ship, Fleet, Grid, Admiral, and Executive. From there, we distributed the individual classes: Connie worked on Ship and Fleet, Kyle worked on Grid, Malena worked on Admiral, Ethan worked on Executive, and Archana worked on the UI/UX and the HTML tables that would correspond to the Grid objects.

As we went along, we made some changes, including getting rid of the Fleet class and adding a “setup” page to the game. Additionally, some of our roles adjusted to accommodate workflow and any problems that arose. For example, Connie ended up coding a lot of Admiral, and Malena switched to working on the interfacing in gui.js for placing ships. In addition to everyone doing smaller-scale testing continuously while writing our working in our respective areas, once the majority of necessary functionality was completed in the lower level classes, Kyle and

Connie started doing comprehensive, general testing of existing code in order to find and fix as many issues as possible before starting to integrate the front-end and back-end. Overall, we worked well on our own files and helped each other as needed.

In addition to working during lab and on our own outside of class, we had two major hack-a-thon sessions on 9/14 and 9/21. In both of these sessions, everyone was present, which allowed us to ask each other for help, advice, and to make sure that all of our classes and functions were going to work together.

Challenges and how they were overcome or dealt with:

1. Team Challenges
    a. **Let's Git-Kraken**: At the start of the project we decided to use GitKraken to provide a nice, graphical representation of our commit history. For the most part, the way work was divided up amongst team members prevented merges from conflicting, but on a few occasions master would be merged into a personal branch, cascading issues going forward for future merges. Most of these errors stemmed from a combination of GitKraken "refreshing" improperly, a misunderstanding of what "merge *branchName* into master" vs. "merge master into *branchName*" means, or pushing changes to the master branch instead of to our personal branches by mistake.
2. Archana
    a. **COMMITment issues:** I should commit more regularly. Overall, I had trouble understanding how Git works, made a couple of mistakes with merging wrongly. The team had to go through quite a bit in understanding how to revert the mistakes. My teammates were extremely patient and helpful in helping me understand Git, and we used GitKraken to have a clear visual understanding of all our versions.
    b. **Styling Issues:** Malena and I had trouble with placing the numbers and letters around the boards. We should have thought of it in the beginning, but we worked around it by using asides and putting elements in separate divs.
3. Malena
    a. **Placing ships user experience and interface:** I knew for placing ships that I wanted to be able to mouse over a cell and see where the ship would be and see if it was legal or not. The `onmousemove`, `onmouseout` and `onclick`, gave me a lot of problems and were really hard to write since they involved hovering over one cell, but changing the color of multiple cells horizontally or vertically. After I finally got all the colors to work the problem was that there was no way to place multiple ships, or switch players to allow the second player to place ships. After a lot of angsting and trial and error I finally came up with a solution that involved turning the placeShip function into a recursive function that would call itself after a user clicks.
4. Kyle

    a. **Javascript for Dummies**: I had not previously worked in javascript at all so I wasn't familiar with all of the differences compared to other languages I had worked in. It is also very weakly typed which I was not used to which combined with the peculiar methods of privatization caused a few bugs early on. Since it is designed to run no matter what, it can have errors occur downstream of the actual issue causing it to be hard to debug.These issues were mitigated by reading documentation and asking my teammates a lot of questions. I think I have a general preference towards languages that are more syntactically ordered rather than the seemingly random state of javascript due to its odd evolution.
5. Connie
    a. **Working with code written by other people**: There were a few cases where I had some trouble understanding code that my teammates had implemented in a certain way, especially when I thought, based on our discussions while planning, that it would be implemented in "my" way. For example, I misunderstood how Grid's updateCell method worked and wrote two functions in Admiral which used an ordinary Grid getter instead of updateCell, which temporarily broke the game. After Kyle explained why updateCell needed to be called, I rewrote the functions to use updateCell while still retaining the functionality that I needed to add to Admiral.
    b. **Having a poorly-defined role**: At the beginning of the project, I volunteered to implement Ship and Fleet, not realizing that that would take much less time and effort than it took for the other classes to be implemented. After I finished writing Ship/Fleet, I found myself needing to continuously look for things to do next: general code testing, asking my teammates whether I could help them take on any tasks (difficult because of 5a above), and later front-end/back-end integration and bug fixing. Though I am happy to do whatever is needed to help complete the project, having essentially an undefined role for much of the project sometimes left me unsure whether I was contributing as much to the project as I would like to.
6. Ethan
    a. **Executive vs. GUI**: My main struggle during the project was planning/adapting the executive class to fit the needs of the program. Initially, executive was the sole connection between the UI and the backend. As the project moved forward, an additional "GUI" class was introduced that handled much of the user-interface JS, shifting the role of executive. This change to a "mid-level" class was the perfect bridge for taking fire-events from the user and storing them in the grids on back end. Learning to be flexible was key to getting the project completed in a timely manner.

Features that didn't make the demo version:

1. **Private Variables:** We would like to have worked with private class fields. We could not do this because we needed to use JSON.stringify to store data in session storage so that the ship placement could be passed from the initial setup page to the gameplay page. However, calling JSON.stringify on an object with private fields creates an empty object, so we had to look for a workaround. A workaround that would have allowed us to continue to use private class fields would be to put the full functionality of the game (setup and gameplay) on a single webpage so that storing objects in session storage would not be necessary. However, due to time constraints, we decided the best option would be to return to using public class fields.
2. **Add a feature that prevents refreshing:** Currently, if you refresh after placing a missed ship, it reloads the whole game, allowing you to fire somewhere else.
3. **Better alignment of the columns and row headers on the table:** We created the tables for placing ships and firing without the A-H and number headers originally. We got this to work with our game interface and then realized we had forgotten to label the tables. Since adding a new row and column in the table broke our code, we opted to make separate tables for the headers. These headers are not quite lined up with the cells in the grid. If we had more time, we would have made these line up better, or ideally, have put them in the grid from the start.

Known issues: (We ran out of time to troubleshoot and fix these.)

1. **Cell 8H**: For some reason you cannot place a 1x1 ship in this cell, and the cell does not highlight when you hover the mouse over it.
2. **On hover over existing ship in horizontal orientation during placement:** when hovering the mouse over a position that is illegal because the ship goes off the map, normally the entire space that would be occupied by the ship should turn red. However, when placing a ship in horizontal orientation, hovering the mouse over certain positions near the right side of the board which are already occupied does not cause any cells to turn red. When the size of the ship you are placing is *N*, the width of the area where this error occurs is *N - 1*.

Retrospective on what the team would have done differently

1. **Document for method names, variables, constants, etc:** It might have improved our work experience to have had created a doc where we stored all of our variable names and method names. Since we needed to call functions that someone else wrote in a different class, this would have helped us organize and make sure that we were calling the correct thing.
2. **Could have made the GUI look better:** Gui.js was our least planned file, and it is also the largest. Compared to all of the other files it is the least organized.
3. **Consistent styling:** This is actually something we briefly discussed when starting the project but we didn't do the best job of keeping it consistent. For example, gui.js has indents of 2 spaces while most of the rest are all 4 spaces. Also, some of us put our

braces in the same line as a function/if/while etc. while others put braces on their own line.

4. **Get more done in the first week**: We spent a lot of time planning which was critical for design, organizing and making sure that everything would work in the end. However, in addition to the planning, we could have gotten more work done early on to prevent coding so much/so late the Saturday-Sunday before code freeze.

Bloopers :)

- "*You're a web developer*, Harry" -- Connie
- "I forgot to add my TODO list to the TODO list!" -- Ethan
- "Archana has commit-ment issues." -- everyone
- *air horn sounds in the distance* -- Ethan
- "I wish this program would throw *more* errors." - Kyle