Alex Lewandowski

Project 1 Diary

2/8/19

Today Andrew and I met and took on the roles of project owners. We discussed what we want in a Battleship game. The most important requirement for both of us was to have a playable game in its most basic form. That describes something that could be text-based, but our second requirement was that it have at least some minimal graphical interface. It should have the ability to generate scores and declare a winner. We liked the concept of powerups that could be earned by sinking a ship in one series of hits with no misses. To start with, we'd like a simple hotseat, pass and play, style game, but we could incorporate networked gameplay in the future. We'd like to give it a humorous theme. We decide on Battle Sushi, with sushi ships and wasabi bombs. Eventually, we would like to add different food-based themes the users can pick from. A few ideas are pizza with anchovy bombs, ice cream Sundays with cherry bombs, and bratwurst with sauerkraut bombs.

2/20/19 AM

I got SFML up and running in Visual Studio, which is a chore I am getting faster at doing! I pushed battleship.cpp containing sample SFML code. I drafted and pushed a rough class diagram for Andrew and I to review and edit in class later today.

2/20/19 PM

Andrew and I spent an hour and a half pair programming. We reviewed the class diagram and removed the fleet parent class. We'll be lucky to get one type of fleet finished for this project and in a work situation it would be easy to add a parent class and more food-based fleets later. We set up barebones frameworks for most of the classes. We don't know how to test with classes yet, which we need to figure out in a hurry.

….

I spent another hour and a half at home refamiliarizing myself with SFML, finishing up the function that places the ships based on a start coordinate and a direction, and trying to figure out catch with classes. It's late and I'm a bit fuzzy so I think I'll have better luck in the morning.

2/21/19

Andrew and I met for another pair programming session but ended up spending most of the time trying to make work with git, Clion, and Visual Studio. VS only allows you to run one main per project. Figuring out how to add multiple projects to VS, set them both up for SFML, and configure the solution to run both mains took a bit of time. We were working for more than 2 hours, and I don't think we added anything to the project.

2/22/19 PM

Andrew and I met for another hour and a half. We put our Project Owner hats back on and revisited our expectations. Given an ever-shrinking time line and the trouble the developers ran into setting up catch, SFML, and git, the project owners decided a text-based game was a better place to start than a graphical interface. Putting our developer hats back on, we removed SFML and tried to push the changes. We accidentally added an IDE project file but noticed before committing it. We Googled how to undo the add and found advice saying to use git reset. Apparently, that was the wrong thing to do because we were no longer able to push anything after that. We spent ½ hour trying to correct it, following all the advice that popped up in git bash when trying to push, but with no success. We finally threw our hands up in the air, made a new repo, and pushed all our docs to it. I know git is supposed to super safe and idiot-proof, It's just not Alex-proof. The past two days have been very unproductive and frustrating. Hopefully, we will make a lot of progress this weekend so we have something decent to hand in on Wednesday.

…

I spent a ½ hour at Hamme Pool working on the Ship class. I wrote and passed tests for Ship:getcoords() and Ship:getLen(). Due to our troubles getting everything running well in the IDEs, we wrote those functions before the tests. We are now in a place where we can write the tests first, in proper TDD form.

…

I spent a few hors and finally started to feel like I was making good progress. Writing tests and functions, failing and passing, them, and moving on to the next. We decided to put our classes into a single header for now. We can break them up if it gets too unwieldly.

2/23/19

I worked on the project all day long, from morning until early morning of the 24th. At this point I am over my 15 hours, but we spent a ridiculous amount of time getting everything working properly with git and our IDEs, with a few false starts along the way. Given that, I am probably just at about 15 hours of actual coding. I started off the day in good TDD form. As the program grew, I inevitably came across issues that required refactoring. In some of these situations I found myself making several small edits to various functions, and I feel like that led to losing my grip on the good TDD practices I had started off with. In some of those cases, I just edited the existing tests to match the new state of things, but there are probably some things that should have tests but don't. I also didn't know how to use catch for functions that take user input, so I didn't write tests for those functions. I see the value of TDD and want to learn how to adhere to it better as projects grow in size and complexity.

In terms of the state of the game, I got a lot done! Phew. At this point, we have a functioning Battleship game. It still lacks any kind of ascii or GUI representation of the board, so you can only play with paper and pen in hand. The important thing is that you can play, and someone can win. Andrew couldn't work on it today, but he is going to pick it up on the 24th and work on adding ascii boards that respond to the game.

2/24/19

I worked on a few minor bug fixes, but today was Andrew's day to pick up the ball and run with it, which he did. We now a have a basic board printing to the screen, and by the end of the day you could see the ships get added as each player placed them. I spent about 2 hours working on things today.

2/26/19

We have met our initial short-term goal of developing a functioning Battleship Game. The project owners would like a humorous, food related, graphical interface, and we would like to speak to them about the possibility of writing the GUI in Python as it would reduce development time. We believe it would suit their needs nicely while keeping costs down.

I revisited our class diagram. The project evolved as we worked and there were a few things to update. While we didn't rely to heavily on the class diagram once we started coding, it was a very useful tool in the early stages of development. It was a great way to clearly illustrate ideas and communicate about the design of the program. It can be so easy to misinterpret a spoken description. Pictures can really help clarify things.

Now we are just have to put together a Powerpoint and practice our presentation.