9/24/18

All members present at the meeting in Class (Kyle, Devin, Connor, Ian, & Daniel)

Plan

- Slack Channel to communicate design so we don't have to wait
- Work on design, cheat mode implementation
- Put a player on the board to walk around use the arrow keys walks around in real time. Most likely an alternative mode
- Application file making it a jar file instead of having to run it every time

10/1/18

All members present at the meeting in Class (Kyle, Devin, Connor, Ian, & Daniel)

Plan

- More fleshed out settings design
- Bomb animations and bomb sounds

10/3/18

All members present at the meeting in Class (Kyle, Devin, Connor, Ian, & Daniel)

Plan

- **On Thursday get together to make sure everyone is working on their part**
- **Go back to the drawing board for project 3 ideas**

Project 3 ideas

1. **Create a Skype-like clone**
2. **Inventory trading system between two clients**
3. **RPG Game/Smash**
4. **Web-based/useful/original/social**
5. **A web crawler that searches Amazon to see if reviews are legit.**

10/5/18

All members present at the meeting in Class (Kyle, Devin, Connor, Ian, & Daniel)

Plan

- Finish sounds tonight
- Implement a better user interface
- Start narrowing down ideas for project 3

Description of how work was split between teammates

The RepoMen team decided to have work split up and assigned to each person. Ian user interface, Connor changing mine algorithm, Kyle cheat mode, Daniel bomb animations, and Devin sounds. These were the things that we wanted to accomplish during this project and we thought that the work split up pretty decently so we decided that each of us would take a specific part. We then decided that when we figured out how to do each of our individual parts we would get together and try to combine them all to get everything we wanted up and running.

Challenges and how they were overcome or dealt with

Over the course of the project, we only ran into a few hiccups. Most notably, the codebase we inherited was written in Java and the team as a whole was not very familiar with using Eclipse. So, at first, it was difficult getting the project to compile and run because we didn't know how to create packages to assemble the project. However, after reading a few guides and reconfiguring settings, we were able to get up and running. Another issue we ran into was animating the bombs on the tiles. The original team had created all of the tiles using jButtons and simply changed the background image on them. This was not conducive to running animations in Java, so instead, we had to create .gifs of our bomb animation and use the same practice of replacing images. The problem with this was using .gif files broke the game if you restarted in the middle of them playing. To fix this bug, we ended up adding a delay to the reset button so it would start working after 2 seconds of clicking the button. This allowed the entire animation to play and only introduced a small delay in gameplay.

We also had some challenges with the changing mines. We had a problem where the mines were going to spots that were already clicked even though they shouldn't have been. We

ended up having to revamp how we were doing it in order to fix it. There was also a bit of issue when we were combining everything together especially with the cheat mode and the changing mines. Since we have the cheat mode in it's own window we needed to get it to update each time the mines changed which proved to be a bit of a challenge that we ultimately ended up solving.

Any features that did not make the demo version

We wanted to implement a better user interface for the project. So the user did not have to click on so many different pop-up boxes. Have the flags, bombs, and cheat mode all displayed at the top of the minesweeper game. We wanted to add a sound for the flag placement but we did not find a good sound file for it. We also had the idea of adding an animation of a guy that goes around to each of the tiles and have him interact with the tiles. While this was a fun idea we couldn't figure out how to implement it so we ended up scrapping it. Another thing we didn't get around to making was an executable file so that you didn't have to run it in a program like eclipse. We figured this would be nice and make it easier to use but it ended up getting pushed back for more important things.

Retrospective on what the team would have done

As a team, we had some good ideas that were able to be implemented independently of one another. Because of this, we decided to split the work up with each person working on their own specific feature. Some of these features, like sound and animation, were developed on the same branch because they were related. Alternatively, we could have worked on a few bigger modifications to the project, together, instead of more individual features. We also probably should have implemented the features together earlier in the process to have been less time constrained to fix bugs that popped up when combining different elements.