Meeting Log

- September 24, 2018
  - Place: Spahr Classroom
  - Work: We decided on the division of work for the project, with Josh implementing the cheat mode and Eric and Zak working together on the bot. We didn't establish any timeline for these tasks, but none should take more than a few hours.
- October 2, 2018
  - Place: Phone call/Spahr Library
  - Work: Decided on some implementation details and got the inherited project working. This required some effort, since we ran into issues with the previous team's use of the JavaScript module system. We noticed that they used NPM and didn't push their package.json, so there may have been a run script in there that would have worked. However, we got everything to work well enough with a simple Python server.
- October 4, 2018
  - Place: Spahr Library
  - Work: Eric and Zak planned out the algorithm for the bot. We aren't using machine learning, but are instead going to develop a simple algorithm that mimics the way a human would play the game. It will give each unclicked space a probability of being a mine and click on the space with the lowest probability. After doing that, it will check if any spaces are certainly bombs and flag them.
- October 6, 2018
  - Place: Jayhawk Towers
  - Work: Eric and Zak finished implementing the bot. It wins effectively 100% of its games when the board is under 1/10 bombs, most of its games when the board is between about 1/10 and 1/5 bombs, and very few games above that. We also updated the documentation with our team details.

Division of Work

The division of work on this project was fairly simple, as there weren't many required tasks. Besides that, the choice was obvious for us based on our existing skills. Josh, having the most experience with frontend development, was the obvious choice to implement cheat mode. His experience would allow him to find the easiest way to implement cheat mode without changing too much of the existing code. Eric and Zak, being interested in machine learning and artificial intelligence, were the obvious choices to implement the bot. This part of the project ended up requiring a similar amount of code to cheat mode, but we decided two of us should collaborate on the bot to allow for the best possible algorithm.

Challenges

Our most significant challenge by far was figuring out how to run the existing project. We suspect the previous team ran their code using NPM, as their repository included the package lock and node modules folder. However, they didn't push their package.json, so we didn't know what, or even if, they used an existing build tool to run their website. After messing around with the modules for a short time - the project seemed to work better if we did away with the module system - we decided to use Python's SimpleHTTPServer to serve the page locally. The previous team's code ran successfully when hosted on GitHub pages, so we were satisfied with this somewhat-hacky fix to use during development. However, Eric ran into yet another issue with MIME types: the Python server wasn't assigning the correct MIME type to the JavaScript files, so his browser refused to execute them. This was easily fixed by using Edge instead of Chrome, which worked but made for a very unpleasant development experience.

Additionally, something seemed to be wrong with the previous team's use of the JavaScript module system, as any addition of new module imports caused the existing code to break. Eric spent a very-frustrating two hours trying to get the import for the bot class to work somehow, but eventually gave up and decided to put the code in one of the existing files. While not exactly the best design in terms of modularity, this worked fine for the project, as it was quite short-term. In addition, the bot ended up requiring a fairly small amount of code, so the less-modular design had very little effect overall.

Missing Features

The feature that was missing that the team had discussed implementing the bot using Artificial Intelligence and Reinforcement learning using Tensorflow.js. However, upon examination that the team had little to no experience in Tensorflow.js they opted for an algorithmic Minesweeper bot. The team had examined the possibility of implementing the bot using machine learning, though they also found the time constraint for the project to be limiting. Having no experience with machine learning, we didn't think we could implement a performant algorithm in just two weeks. However, our current Minesweeper bot is not perfect for larger and more complex cases, only performing well up to about 1/10 of the board being bombs, and we believe that the implementation of some form of machine learning could substantially improve the results of the bot. We could, for example, start with our current algorithm, but in cases where it can make no good decision revert to the learned model. This could help in cases like at the beginning of the game, when the bot is searching for an open area with no bombs. A learned model might be able to make more varied moves, thereby finding the open spaces faster and more reliably. After our current algorithm finds a large open area and exposes some "0" tiles, it can perform fairly reliably and wouldn't need as much help from the learned model.

Retrospective

The team found that they were able to able to effectively complete the project before the deadline with few hiccups. One thing that would likely change would be to start working through the codebase at an earlier point of time. Due to conflicts and other commitments, we did not start working on the project immediately after the start date and as a result started analyzing the code at a later point in time. If we had spent some time initially to look over the previous team's code then we may have had time to develop a more robust solution to issues we found with the previous code. Additionally, we could have implemented more features beyond the bot and cheat mode; for example, we could have added an input to change the speed of execution of the bot. Currently it waits one second between each move to allow the user some time to look at and mentally process the board - without this delay, the game would pass in an instant and the user would have no idea what happened. It may have been useful to allow the user to change the delay between each bot move if they wanted to analyze the bot's strategy more thoroughly or collaborate with the bot.