Project 1 Retrospective Write-up for “Tommy and da Boiiis”

Team Members

Nam “Tommy” Nguyen
Bailey K. Srimoungchanh
Branden Taylor
Grant L. Stewart

Git-Repo

https://github.com/BaileySri/Project_1/tree/master

Finish Date: September 16th, 2018

Meeting Log

- 8-31-18, Eaton 2
    - Tommy, Grant, Bailey, Branden
    - Determined Java was the language we wished to code our Project 1 in, exchanged githubs and planned our next meeting.
- 9-3-18, Spahr 1436
    - Tommy, Grant, Bailey Branden
    - Determined team name to be “Tommy and da Bois” and decided the GUI library we would use would be AWT, next meeting planned.
- 9-4-18, Spahr 1322
    - Tommy, Grant, Bailey, Branden
    - Decided to switch from AWT to Swing as it is strictly an upgrade on the AWT library. Decided on what files we would need to create and how to layout the menu. Divided the work amongst the team.
- 9-5-18, Eaton 2
    - Tommy, Grant, Bailey, Branden
    - Decided a GridLayout was the optimal way to layout the board
- 9-6-18, Spahr 1324
    - Tommy, Grant, Bailey, Branden
    - Outlined the Tile class and how it would interact with the Board class, designated a centralized Repo to hold our master branch.
- 9-6-18, Google Hangout
    - Tommy, Bailey
    - Helped Tommy download the editor and git bash. Forked the repository.
- 9-7-18, Eaton 2
    - Tommy, Grant, Bailey, Branden
    - Started writing the layout of the class files. General class creation occurred.
- 9-8-18, Google Hangout
    - Tommy, Bailey
    - Pushed the menu implementation into main repository.
- 9-11-18, Eaton 1005-D
    - Tommy, Grant, Bailey, Branden
    - Discussed the manner of which the board is controlled, decided on 2D array. Most of the classes were laid out, started reviewing our first interpretation of interaction between classes and further decided on how the board would operate.

- 9-13-18, Eaton 1018
    - Tommy, Grant, Bailey, Branden
    - Discussed the manner of board expansion and what final error handling and functions needed to be generated. Began cleaning up comments/syntax of code and started deciding which functions that backend had created would be implemented and which had become unnecessary.
- 9-14-18, Google Hangout
    - Tommy and Bailey
    - Implemented the win and lose conditions and refined the flag functions.
- 9-14-18, Spahr Library
    - Branden and Grant.
    - Discussed the win/lose disabling of the boards buttons, used Branden's setDisable() function in Game_Driver to solve the problem of users still being able to interact with the board after the game was finished.

Work Distribution

Work was split between frontend and backend with our classes being: Tile, Game_Drive, Menu and Board. Menu and Board served as our GUI front end framework for our minesweeper game and was worked on by Tommy and Bailey. Tile and Game_Driver was classified as the backend of our game and was worked on by Grant and Branden.

Challenges Faced

Some challenges faced when creating this minesweeper game was the collaboration between team mates. Our attempt to overcome collaboration difficulties was to constantly keep each other up to date on what we were working on and what we were implementing through a GroupMe conversation that had all the members of the team included. We experienced some of the typical project one difficulties when it came to the distribution of work and workflow as collaboration between team members started off slow only being in the team meeting but then changed to a regular daily basis over time. By meeting at least twice a week, on Tuesdays and Thursdays, we were at least able to keep members collaborating more consistently but still faced a period of time where we did not collaborate as much during the weekend. Instances of one individual waiting on another to finish a particular function before being able to proceed occurred and provided a form of roadblock at times until the function was resolved. We did not handle this in a organized fashion but instead decided to continue writing and cleaning up the code as the roadblocks occurred, eventually coming together to resolve errors arising not from merge conflicts but from different implementations. Communication was also seen as a challenge towards the finalization of the project as active documentation did not occur as smoothly as it should have and when an individual had to interact with a part of a class that they

did not write it would sometimes be difficult to interpret what was occuring. When situations like this occurred actively trying to communicate with the team member who wrote the function was the best option and informing that team member that documenting the current implementation would be helpful. Some challenges faced by some team members was getting used to how git works such as remotes, push/pull, resolving merge conflicts, and when to commit. The challenges brought by git would be resolved by asking a more experienced member of the team for assistance with Git and if not resolvable searching for solutions together would be the best option. Other challenges were in the code itself because our team initially challenged ourselves with using Java, which was new for most team members. Working together throughout the code creation and seeking assistance from each other in times where confusion would arise from writing Java was the best manner of which to approach writing Java, a language we as a team have little experience with.

Features left Undone

Some features talked about or left undone were:

- Not losing on first click
- Having a menu that presented beginning, medium, hard and custom options
- Possibly non-rectangular board generation (Think in the shape of a U)
- Providing a “Free Reveal Tile” option that allowed the user a desired number of free reveals that wouldn’t end the game if the tile was a mine1
- “Elapsed time” timer
- A potential plain text file creation that can be created/interpreted through Java to recreate a board which would act as a pseudo-save system to continue a game at a later date

What would we do different next time?

As the process of creating the classes that were determined as necessary for the minesweeper game it became more evident that we had not planned the interaction of the classes well. When one person was waiting for the information that would be passed from one class to another there were instances where both individuals had different understandings of what would be passed which would result in the classes needing to be modified to allow proper interactions. This issue was slightly more than a merge conflict as it required functions to be rewritten as necessary to allow the code to run properly. By improving our ability to layout what interactions will be happening more clearly and what values will be present at a given time the issue of having to change entire class structures will start lessening as we bring code together that will initially be able to interact.

Another approach that would be different next time is the division of work which became more prevalent as a difficulty. The work was initially divided into 4 classes for 4 individuals but these classes were not balanced and the differences in work load became more impactful on the workflow of the team. An example of this imbalance is comparing the Game_Driver class to that of the Menu class. The Menu class implementation was short as it was the GUI and not the logic behind the game but the Game_Driver class was large as it had to handle the controls for the minesweeper game. This imbalance was jarring on the workflow as it would create situations in which a team member would have to wait longer than another team member before getting feedback on if the function calls in their class would work correctly with another. For example matching parameters of a function that is called in one class but declared in another that was outlined but the parameters were not explicitly determined creating the situation in which a function call did not match its declaration. Another problem was that team members would have outdated repositories that would be several commits behind. When pushing the repository to the main one or popping the stash, there would need to be bulks of merge conflicts that would need to be resolved. An example of this was when one of the team members worked on one of the classes and afterwards he tried to commit and push it, but there were parts that needed to be resolved. A possible solution to this is to have everyone work on a branch rather than forking it. If we commit and push more frequently to our own branches, then everyone can be more actively updated.

Works Cited

MineIcon.png - https://www.iconspng.com/image/71827/black-cartoon-bomb