EECS 448

FRAGILE DEVELOPMENT

MINESWEEPER - CODE SWAP

October 8, 2018

Benjamin Streit

Blake Rieschick

Evan Brown

Application Overview

This implementation of Minesweeper stays true to the original Windows 3.1 version, aside from the addition of mines that will move upon cell reveal as well as a cheat mode. The application is built using Python, and Tkinter for a GUI.

Delegation of Labor

To manage the project, YouTask, a team task management tool was used. Epics for Project I Requirements, Old Code Clean-up, Cheat Mode, Moving Mines, and GUI were created. From here, stories, bugs, feature requests, etc. were created and each team member selected tasks they deemed themselves suited for and assigned it to their user account.

A specific breakdown of labor performed is as follows:

Benjamin Streit	Cheat Mode Stopwatch class Old code clean-up
Blake Rieschick	Moving Mines
Evan Brown	Project I Requirements GUI

Challenges

The team ran across two main challenges over the course of completing the project. Firstly, parsing through the inherited code-base. The code consisted of superfluous methods that were never used throughout the application, as well as several methods accepting superfluous arguments. Examples include a method for printing the game board to the console, despite the existence of another more robust print-game-board method already being in use, as well as methods accepting and using arguments that already existed as member variables of the class and thus could have been accessed natively. Such occurrences were adjusted accordingly in order to produce more readable and scale-able code for the team to work with and build upon.

The second main issue the team had to overcome was the addition of a custom feature. This feature is that of moving mines, meaning that whenever the user interacts with the board (aside from calling upon cheat mode), all or some of the remaining mines move to a new location on the board. The moving of the mines themselves proved to be relatively simple, but the updating of the necessary state variables concerning the grid in which the entire board is housed proved to be less than simple. Issues of now outdated cell information being used for interactions post mine-move highlighted this, and the move-mine function had to be updated accordingly. Thankfully, after further analysis of the code, swapping around the order in which methods were called to update state variables during the move-mine function provided a resolution, the final order being in which a method call to update grid values was called after all other associated methods.

Feature Backlog

Overall, all but one envisioned enhancement made it to the demo-ready build before the code freeze date. During brainstorming of what custom feature the team should implement, talks of a high-score system came up. In the end, the team decided upon the feature of moving mines as the previous project saw implementation of a high-score system. However, this idea remains appealing as adapting the previous model from NodeJS to the new platform of Python would have certainly been an interesting challenge. This route would have also served as a novel means of improving upon previous design and thought processes concerning code design and implementation.

In Retrospect

Seeing as this project revolved around the inheritance of another team's code-base, in retrospect, the team would have been better served under the adoption of a more structured and robust testing system for the inherited code. With meeting project I requirements being a graded portion of this project, all project I features were quickly accounted for, and all project I bugs were identified by hand. This is not an ideal practice, nor an industry practice, and thus will not be the team's strategy moving forward. A robust testing environment must be created and put to use before building upon an inherited code-base to ensure integrity of the code prior to making any additions or deletions. This is inline with industry practices and would behoove the experience of working with someone else's code, and will be kept in mind for all future endeavors.

MEETING LOG

```
## Meeting #1
Date: Friday, September 21, 2018

Time: 11:30 - 11:50 AM

Location: In-Class

  * Decided upon Epics necessary for project
  * Discussed GUI frameworks to use

## Meeting #2
Date: Monday, September 24, 2018

Time: 11:30 - 11:50 AM

Location: In-Class

  * Decided upon custom addition
  * Discussed GUI frameworks
  * Discussed division of labor

## Meeting #3
Date: Wednesday, September 26, 2018

Time: 11:30 - 11:50 AM

Location: In-Class

  * Discussed work each team member has done
  * Discussed new Epic for cleaning up the inherited code base
  * Discussed work for each member to be done before next meeting
  * Discussed next meeting for later this week

## Meeting #4
Date: Friday, September 28, 2018

Time: 11:30 - 11:50 AM

Location: In-Class

```

```
  * Discussed issue with moving mines feature
  * Discussed next meeting for the following week
  * Discussed next steps after moving mines is resolved
  * Discussed current progress on example GUI implementation.

## Meeting #5
Date: Monday, October 1, 2018

Time: 11:30 - 11:50 AM

Location: In-Class

  * Discussed moving mines implementation
  * Discussed continuing work on GUI
  * Discussed implementing a stopwatch
  * Discussed next meeting time and location

## Meeting #6
Date: Tuesday, October 2, 2018

Time: 1:00 - 2:50 PM

Location: Spahr Library

  * Discussed work done since last meeting
  * Discussed moving mines implementation
  * Discussed continuing work on GUI
  * Discussed revamping documentation implementation

## Meeting #7
Date: Friday, October 5, 2018

Time: 11:30 - 11:50 PM

Location: In-Class

  * Discussed work done since last meeting
  * Discussed GUI implementation
  * Discussed documentation
  * Discussed final activities
```