MS1 Alpha: Project Report (self-evaluation)

Team Members

Max Pace (netID map438) • Arjun Shah (netID ans96) • Jerry Xu (netID jjx6)

CS3110 Fall 2021

Vision:

We don't think this part of our core vision has changed. We want to build a version of Minesweeper in OCaml. A change that we made from MS0 was relating to saving boards to and loading/reading boards from CSV or JSON files. We decided to reconsider this feature as a possibility for the next sprint because it would complicate the initial implementation, specifically with regards to encapsulation and information hiding. The rest of the key features that we specified in our project proposal are all still in our core vision.

Summary of progress:

Since the gameplay of Minesweeper is dependent on a board of square tiles, we designed our program to represent each tile as a record within a Square module. We wrote this module to represent the location data and to differentiate between discovered and undiscovered locations, actual mines, flagged locations, and mine-adjacency state. We also wrote a module for the gameboard storing all the squares using arrays and imperative features. This implementation largely preserved encapsulation and made developing either one at the same time much easier. The actual game of Minesweeper is normally played in a GUI, with colors and clickable tiles. However, for our rudimentary implementation at this stage, we are on the terminal, which is rather user-unfriendly and ugly when it is just black and white. We devised a way to print out our gameboard in the terminal with built-in guides and axes and with color, just like in the real game.

During our demo, we will demonstrate the abilities from above—displaying flagged and unflagged locations, digging up squares, and pretty-printing our boards with color. In addition, we will demonstrate the ability to generate a board randomly.

Max Pace, Arjun Shah, Jerry Xu
CS3110 FA21

Activity breakdown:

Jerry and Max both focused on the design of our data storage; specifically, Max completed most of the board module and Jerry completed most of the square module, helping out with the random generation and placement of mines. Both of these were designed with the help of Arjun as well. Arjun focused on the testing of these two modules.

- Max
    - His responsibilities specifically consisted of designing module structures, specifications, various testing, and prototyping early versions of pretty printing.
    - His delivered features include the Board compilation unit and the report.
    - Hours worked: 8
- Arjun
    - His responsibilities consisted of writing the test suite creation and implementation testing as well as cross-checking the individual module designs. His
    - His delivered features include comprehensive testing and code coverage, and contributing to the report.
    - Hours worked: 8
- Jerry
    - His responsibilities consisted of designing the module and project structure, testing the modules, and writing the report.
    - His delivered features include the square compilation unit, mine generation, and pretty-printing.
    - Hours worked: 8

Productivity analysis:

We were relatively slow when it came to working on the project initially, but we got the ball rolling pretty well as we started studying together for the exam, and we ramped up our meeting frequency by quite a bit after the exam. We did accomplish what we had planned, and we think that we still have our code under control and have managed the style well. Thus, we believe our estimates were pretty much on-point.

Max Pace, Arjun Shah, Jerry Xu
CS3110 FA21

Scope grade:

We achieved Excellent scope in this assignment. To begin with, we'll consider what we laid out for Satisfactory scope. Our plans stated that we would specify and implement the game board and square functionality. Specifically, we would have a representation of the data and the ability to dig/reveal squares, lay out mines (and dig a mine to end the game), flag mines, and count adjacent mines for each square. It is worth noting that our initial plans stated that we would be able to save boards to and load/read boards from CSV or JSON files, but we elected not to implement this at the moment since doing so would complicate much of the initial implementation and would have made testing much more difficult to start. Instead, we will consider doing this later on in the project if we still believe the feature is worthwhile, because it would help with automating our tests for user interfaces. Thus, since we achieved all goals we set out to, we achieved satisfactory scope.

We also achieved Good and Excellent scope since our initial plans were to be able to display a snapshot of the board's state and also to be able to pretty-print the board using color. We can print a depiction of the mine layout with the number of surrounding mines displayed for each square as well as a user-geared display, showing only revealed squares and flags. We incorporated numerical axes in the *x* and *y* directions and colorized the board to match the original game's design to the best of our ability. Therefore, we met our plans for good and excellent scope.

Goals for next sprint:

- ☐ Satisfactory: *Be able to instantiate an instance of a board and manipulate it according to user input via the terminal, which we must also be able to handle (without ever displaying exceptions to the user).*
- ☐ Good: *Be able to recursively reveal all surrounding tiles when a square is surrounded by zero mines (and do the same for those revealed squares if they have zero surrounding mines)*
- ☐ Excellent: *Handle win/lose cases by offering a chance to play again. Reveal where all mines were if the user loses. Begin a timer when the game begins and display the timer to the user. The user should automatically lose when the timer runs out.*