Max Pace, Arjun Shah, Jerry Xu
CS3110 FA21

MS0 Charter: Project Proposal

Team Members

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

Meeting schedule

We are all running busy schedules, so we will structure and schedule our meetings as follows:

- We will commit to one guaranteed meeting every weekend
    - This meeting will alternate in location physically and virtually every other week
    - We will tentatively schedule for Saturdays at 1PM.
    - We will communicate, reschedule and agree on times over group messaging if conflicts arise
- Meetings during the week will be scheduled on demand as necessary with the goal that there will be at least two in-week meetings per week.
    - The best days for this will be Tuesday and Thursday - most will be scheduled on Tuesday and Thursday
    - These will be held virtually by default, with the option to switch to a physical setting if pair programming, for example, is desired.
    - Not all team members are expected to attend in-week meetings because of conflicts
    - We will communicate, reschedule and agree on times over group messaging

Max Pace, Arjun Shah, Jerry Xu
CS3110 FA21

Project - Minesweeper

Summary:

Core vision
We want to build a version of Minesweeper in OCaml.

Key features

- Actively handle user input
- Reveal and flag specific tiles upon request (and reveal surrounding tiles when appropriate, such as flood-filling all empty squares when revealed)
- Randomly generate and keep track of the state of a full board of tiles upon starting the game
- Calculate the number of surrounding mines of any tile and display it
- Be able to seek out and “auto-discover” when clicking on a tile that has its adjacent sum full of expected mines
- Terminal interface to start to be ported to a GUI

Max Pace, Arjun Shah, Jerry Xu
CS3110 FA21

Roadmap:

MS1:

Main goal: Implement basic game board to print to terminal

- ☐ Satisfactory: *Have a game board and be able to represent mines, discovered squares, flagged squares, and undiscovered squares in memory, read and write board to a JSON file.*

The board files that our game engine will save to and load from for testing will be formatted in CSVs. In OCaml, we will use the OCaml CSV package to write to and read from these files (maintained by Github user Chris00).
The gameplay of Minesweeper is dependent on a board, which is a Cartesian grid. We can call each integer coordinate pair a location (or interchangeably, a square). Each location can be either discovered or undiscovered. An undiscovered location can contain either a mine or nothing. An undiscovered location can also be flagged by the user if they think that the location contains a mine. A discovered location is one opened up by the user (this will come later, but a discovered location cannot contain a mine, or the user will have lost). If a discovered location does not have any mines on its 8 adjacent locations, it will be considered a non-mine-adjacent location. Discovered locations that are adjacent to mines will be considered a mine-adjacent location. We will create a way to represent this data in code, and to differentiate between discovered and undiscovered locations, actual mines, flagged locations, and mine-adjacency state.

- ☐ Good: *Display a snapshot of game board; be able to calculate adjacent numbers on the game board*

At this point in the project, we will be displaying the state of the board in the command line. We will primarily be using the print_string and print_endline library functions for this purpose. Being able to display the state of the game in any shape requires a usable implementation of the model from which we can extract the necessary data. Ensuring we can do this is critical to being able to scale up the interface for full functionality and user interaction. We will need to be able to retrieve our stored data from satisfactory scope and differentiate between discovered and undiscovered locations, actual mines, flagged locations, and mine-adjacency state when displaying information to the user. Furthermore, we will need a way to calculate the number of adjacent mines to a mine-adjacent location and present that to the user, because that is the primarily logic behind which the user plays the game.

- ☐ Excellent: *Pretty-print game board with numbered axes and colors*

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 when it comes to trying to select tiles to execute commands on. It is also rather ugly when it is just black and white. We will devise a way to print out our terminal with built-in guides and axes and also create a way to color the numbers just like in real life.

MS2:

Main goal: Make game playable and maintain game state

- Perform operations via the terminal
- Be able to mark squares as mines and dig up empty squares
- Implement losing operation
- Start the player on an empty square and generate a valid board to start
- Allow the user to create their own sized game boards
- Seek out non-mine-adjacent squares and flood fill when a new one is discovered
- “Autocomplete” discovery of adjacent locations on a mine-adjacent square whose number has been exceeded by the number of adjacent flagged locations.

MS3:

Main goal: Improve user interface

- Either convert existing the terminal game to GUI
    - (or make the terminal game more robust, depending on overall progress as the development of the game progresses)
- GUI should have the ability to click on the board directly, interact and discover, and provide the user with clear information about the state of the game in a pleasant manner