Minesweeper

CS 224

1 Introduction

For this assignment you will write a Qt application using the MVC design pattern that allows the user to play Minesweeper. Section 2 gives a brief overview of how Minesweeper is played. Section 3 lays out the requirements for the interface, and what everything needs to do. Section 4 gives some advice on how to set up your model. Section 5 gives some suggestions on how to complete the assignment. Scoring information is in Section 6, and what to submit is covered in 7

This assignment will be submitted via GitLab. As with the previous assignment, I'll be looking for evidence that you're using git reasonably well.

2 Playing Minesweeper

To quote Wikipedia:

> The player is initially presented with a grid of undifferentiated squares. Some randomly selected squares, unknown to the player, are designated to contain mines. Typically, the size of the grid and the number of mines are set in advance by the user, either by entering the numbers or selecting from defined skill levels, depending on the implementations. (In the Microsoft variant, this is limited to 30 times 24 with 667 mines.)
>
> The game is played by revealing squares of the grid by clicking or otherwise indicating each square. If a square containing a mine is revealed, the player loses the game. If no mine is revealed, a digit is instead displayed in the square, indicating how many adjacent squares contain mines; if no mines are adjacent, the square becomes blank, and all adjacent squares will be recursively revealed. The player uses this information to deduce the contents of other squares, and may either safely reveal each square or mark the square as containing a mine. [1]

3 Interface Requirements

Your application must present a grid of QPushButtons, or equivalent, representing the Minesweeper game itself. You must also present the number of bombs in the puzzle. The user must be able to click on widgets in the grid, which 'uncovers' the contents of the square. When the user clicks an empty square, you do not need to reveal adjacent empty squares. You must also present a method for starting a new game, e.g. by using a menu.

[1] `https://en.wikipedia.org/wiki/Minesweeper_(video_game)`

At the minimum, you must provide a 10x10 grid, containing 10 bombs. You may produce the interface using any (Qt-based) method you wish, so long as grid cells are square and you do not use Qt Designer or a similar tool. See Figure 1 for an example of the interface.

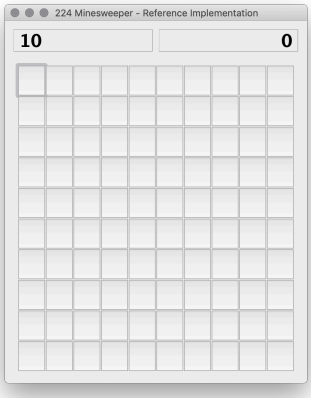

Figure 1: The overall user interface

4 Model

You are required to use the Model-View-Controller design pattern for this assignment. To that end, you will need to have a model class which represents the actual state game, and have your interface (probably in a class of its own) interact with the model. Precisely how you structure the model is up to you, but the following functionality is probably a good starting point:

newGame Start a new game, with new random bomb placements. That function should probably:

1. Initialize your game grid(s)
2. Add an appropriate number of bombs
3. Figure out adjacent bomb counts

reveal Reveal a particular square in the grid

getSquare returns the current state of an indicated square (revealed or not, adj. bomb count)

getMoveCount returns the number of moves made

getGameState indicates the current state of the game (won, lost, in progress)

5 Suggestions

You are free to build the interface however you'd like (so long as you do not use Qt Designer). This section presents some suggestions for how to lay it out, and some other useful tips that you can use or not, as you would.

By all means, start from the PyQt demos we've done in class.

5.1 Laying Out Controls

You'll probably want a QGridLayout for laying out the main grid of squares. To get an interface that looks like the example:

1. Create a QVBoxLayout
2. Add to it a QHBoxLayout containing two QLabels
3. Add the QGridLayout to the QVBoxLayout

Note that the second label in the example interface is a move counter, and is not required.

5.2 Grid Squares

You'll need some way to identify which grid square was clicked. Qt provides a properties system that allows you to add extra information to a widget. You can use that to keep track of which row/column a grid square is in.

When you create the button, you can set the properties like this:

```
button.setProperty("myRow", row)
button.setProperty("myCol", col)
```

Then, in your clicked method, retrieve the row and column like so:

```
clicked = self.sender()
row = clicked.property("myRow")
col = clicked.property("myCol")
```

5.3 2D Arrays

You can construct two-dimensional lists in Python, but it's a bit more complicated than in other languages. Essentially, you need to construct a list of lists. Assuming `rows` and `cols` are the desired size of your 2d list, the following code will produce a list of lists of that size, initialized to 0:

```
grid = [[0]*(cols) for i in range(rows)]
```

6 Scoring

Programs which do not run will receive a 0. The same Git requirements from previous assignments are in effect. The assignment is worth 35 points, broken down as follows:

	GitLab
5	Multiple commits at reasonable junctures
5	Reasonable commit messages

5	General Style and organization
10	Interface looks correct
5	Proper model/view breakdown
5	Minesweeper is playable

Additionally, you may earn some bonus points as follows:

1	When an empty square is revealed, reveal all squares adjacent to it recursively
1	Allow the user to 'flag' bombs
1	Keep track of the number of moves the player has made
1	Include a timer which counts from the first click on the grid
1	Allow the board to be resized

I will consider other functionality for bonus points as well, to a maximum of 5 bonus points.

7 What to submit

You'll be submitting your project using Gitlab. Begin by creating a project named `CS224-Mine` and getting it checked out and ready to use. Do your development work as usual, checking things into git as appropriate. You will probably want to break your program into three files (a main file, a file for the window, and one for the model), but this is not required. In addition to your source code, provide a README file which includes:

- How to run your program
- Anything you'd like considered for bonus points

Once your project works to your satisfaction, made sure it's up-to-date in GitLab and add me as a reporter. Also leave a note in Canvas that your program is ready to grade.