ESE 224 Spring 2015

Course Project Description

Over the course of this semester, you learn various coding concepts in C++ that are applicable to programming in general. This class project is designed to show you how many of these concepts can be used together in a single application and will test your familiarity with programming these concepts. This project is intended for 2-4 people groups.

We will provide a rigid skeleton guide which you MUST follow, but how to implement components in the skeleton is your decision. This means that any public method signature should not be modified (its return type, name, and parameters). However, you are free to remove, add, or edit any private methods you feel necessary. The private methods are included as a guideline to structuring and organizing your code. The private variables should also not be modified, as much of the public functions rely on these private variables. This means we (i.e., TAs) should be able to use our own GUI code to run your minesweeper class, and your GUI class to run our minesweeper. Following this skeleton allows us to show you how a program is typically structured while allowing for your own creativity to shine. You should describe your algorithm/implementation in comments in the code as well as in a final report.

Grading will be roughly divided into 80% for the actual program, 20% for a final report, and an extra 20% for bonus. The actual program should be functional and free of basic errors to obtain credit. Extra points for style will be given for succinct, smart, and readable (spaces and indentation) code as well as good documentation and comments. Every submission should include ALL relevant header files (*.h) and source code (*.cpp) as well as the report. These should compile on GCC without exception. Production code that is unable to compile is obviously unacceptable in industry, and the same mindset should apply here.

The project report should describe the efforts that were put into the project as well as show your familiarity with the program. Things that should be included and mentioned in the project report include: a description of the program architecture, any algorithms written (searching data structure, etc), and a description of which team member completed which tasks.

Project Description

The project is Minesweeper. This game will be implemented on the console/terminal for simplicity. A version of the game can be found at minesweeperonline.com. A description of the game rules can be found on wikipedia. There should a main function to run the (character-based not graphical) gui and a class representing the playing board at minimum.

Required functionality

1. Gui is easy to understand and user friendly; should not be too complicated (text/console based)
2. Any and all data structures used should be appropriate for their uses
3. There should be no possibility of turn 1 game over (the first box selected is a mine)
4. Tiles are correctly revealed on click (surrounding tiles for 0 mines tile and single tile for all else; refer to wiki and game example for clarification)

5. Mines are randomly placed on field
6. Read an initial mine placement input file from a given file path. A test file will be provided at a later date.
7. Correctly detect end game conditions. There are three possible end game states at the end of a turn, and two should end a game (WIN, LOSS. Ongoing is a state, but should not end the game)
8. Should handle any possible errors

Generated board size should not be hard coded (m x n board size), but the common size:mine ratios are 9x9:10 mines, 16x16:40 mines, 16x30:99 mines.

Project Report

1. Project description: Should discuss and describe feature implementations and overall functionality. Algorithms and implementation tricks should be described and discussed. This is the place to show off whatever clever tricks you pulled to get the project done.
2. Class/Method Description: Description of class/method implementation. This should cover skeleton functions, skeleton classes, and any other code you added.
3. Notes: There should be a description of how the program should behave. Any user assumptions made should be noted here as well.

Bonus

1. Include a scoring system that tracks # of wins, # of losses, win/loss ratio as well as current/longest win/loss streak. Should be viewable from gui.
2. Add code to store the above in a file. The file saved should be in a human readable format (ie. I can open this saved file and understand the statistics)
3. There should be no possibility of turn 1 game over (the first box selected is a mine). Mine placement should still be random.

This can be enabled by working on the bonus source (which inherits from minesweeper) and calling a bonus object instead of a minesweeper object.

Submission Requirement

All submissions will need required header files (*.h), source code (*.cpp), and a project report in (any common document file format, .docx, .doc, .pdf etc.). A **simple** README file should also be included detailing how to start and use the program. The report should detail the architecture of the program, class hierarchies and relationships, general functionality of classes, interfaces, and methods, and usage of class-learned concepts.

Starting Instructions:

First, please understand the game mechanics of Minesweeper before attempting. This project will replicate the following flow in the flow chart.

Step 1:

Understand the given skeleton structure. The idea here is to separate GUI code from game state/interaction code. This is achieved by performing all GUI in main.cpp and maintaining the game state inside minesweeper.cpp. It may be most logical to begin coding the functions as they appear in logic flow (the flow chart). Beginning in the minesweeper class may clear some confusion you have about the program flow. You should deal with each function one by one, as many of the functions rely on helper private functions to operate. The design reasons for this vary, you may ask a TA if you disagree or do not understand how the function was designed. Feel free to add functions that you may need to achieve the required functionality. Follow the program flow in the minesweeper class based on the comments in the given skeleton as well as the program flow to start coding.

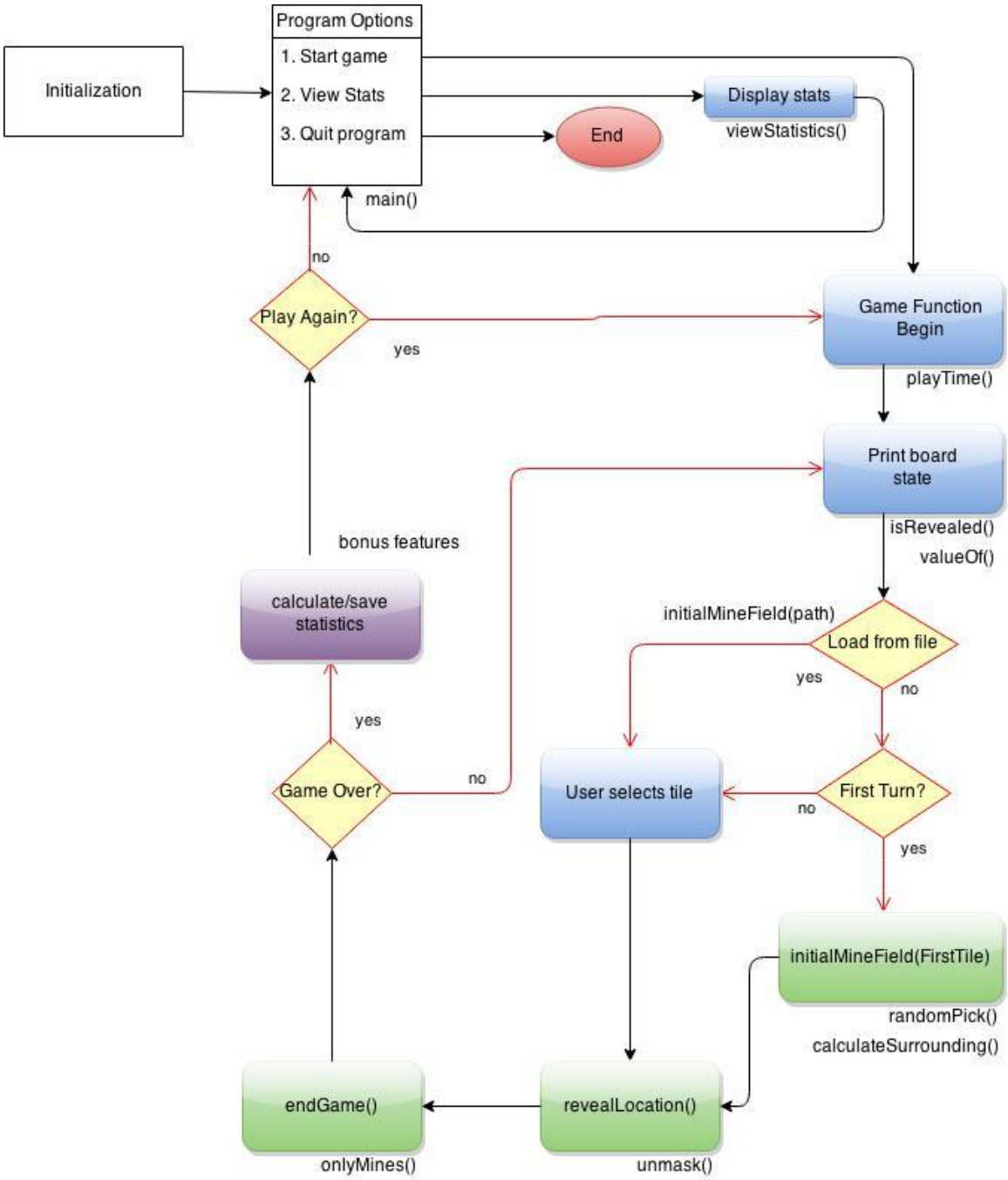
Step 2:

After understanding the minesweeper class and its functionality, features, and design, you can create a GUI for it. The GUI handles all user interaction and passes them along to the minesweeper class. The GUI should be on the console in text; no need for fancy pop up windows and pictures. The GUI should gather user input, pass it to minesweeper, and display the minesweeper output to the user. If you do not intend to implement the bonus features, you may skip the purple box in the program flow chart.

User interaction for gameplay should consist of three inputs. The first parameter should represent a left/right mouse click in minesweeper games with GUI, where left click reveals a tile and right click marks a tile as possible mine. The second will be the column #/X axis value, similarly, the third will be row#/Y axis value. These parameters should be tokenized using a ‘,’. So if the user were to mark the top left tile (0,0), the input would be “r,0,0”.

Step 3:

Bonus features can be implemented after the above is completed. These should basically be implemented after the minesweeper class is complete, especially since bonus will inherit minesweeper. Bonus has additional functionality and member variables to minesweeper. Keep in mind that even though bonus inherits minesweeper, you will need to edit/add to you GUI to accommodate the additional functionality of your bonus features.



White = main()

Blue = playTime()

Green = minesweeper.cpp

Purple = bonus

Functions in boxes should actually be called, which in turn call on the annotated functions to achieve their intended functionality.