2805ICT
System & Software Design
Milestone One
Weight 10%

Joshua Grylls

02/08/2019

Your first task is to implement in your favourite programming language and using your favourite programming tools a version of Minesweeper where scores are the time taken for a particular dimension of the board. However, this is just one phase of the overall challenge, and you should consider designing your software architecture and software components so other tasks are also achieved.

Requirements -

Req #	Requirement Description

Functional Requirements -

FR1	Clicking on a tile will reveal the contents of that tile. A tile can contain either a blank space, a mine, or a number. - Clicking a tile that contains a mine will cause the player to lose the game and the timer will stop. - Clicking a blank tile that has a mine in any of its surrounding tiles, will display the number of mines surrounding that tile as its contents. - Clicking a tile with a that contains a blank space or number will let the player continue. Revealing all blank tiles and numbers causes the user to win the game. When a blank space is revealed, adjacent/connected blank spaces tiles are also revealed.
FR2	Mines are randomly spread across the game board once the game starts or is restarted. The number of mines = length of the board + 1.
FR3	The user can press a restart key that causes the game to reset all tiles and change the location of the mines. The timer should also reset.
FR4	Right clicking a tile adds a ‘flag’ to it to indicate the player has marked it. This will act as a personal note for the player. Right clicking the tile again removes the flag. If the player flags all mines, the game is WON.
FR5	A timer will start when the user clicks the fist tile. The timer will stop once a mine has been revealed or game has been won. If the timer is lower than the high score when a player wins, update the high score.

Non-functional Requirements -

NR1	The board will be a minimum of 9x9 tiles.
NR2	There can be no more than four mines surrounding a blank tile (this will display a number 1, 2, 3 or 4)
NR3	Response time is no longer than .5 seconds when revealing the contents of a tile.
NR4	The user is only able to click one tile at a time
NR5	If the game has been lost or won, the player cannot continue exploring tiles.

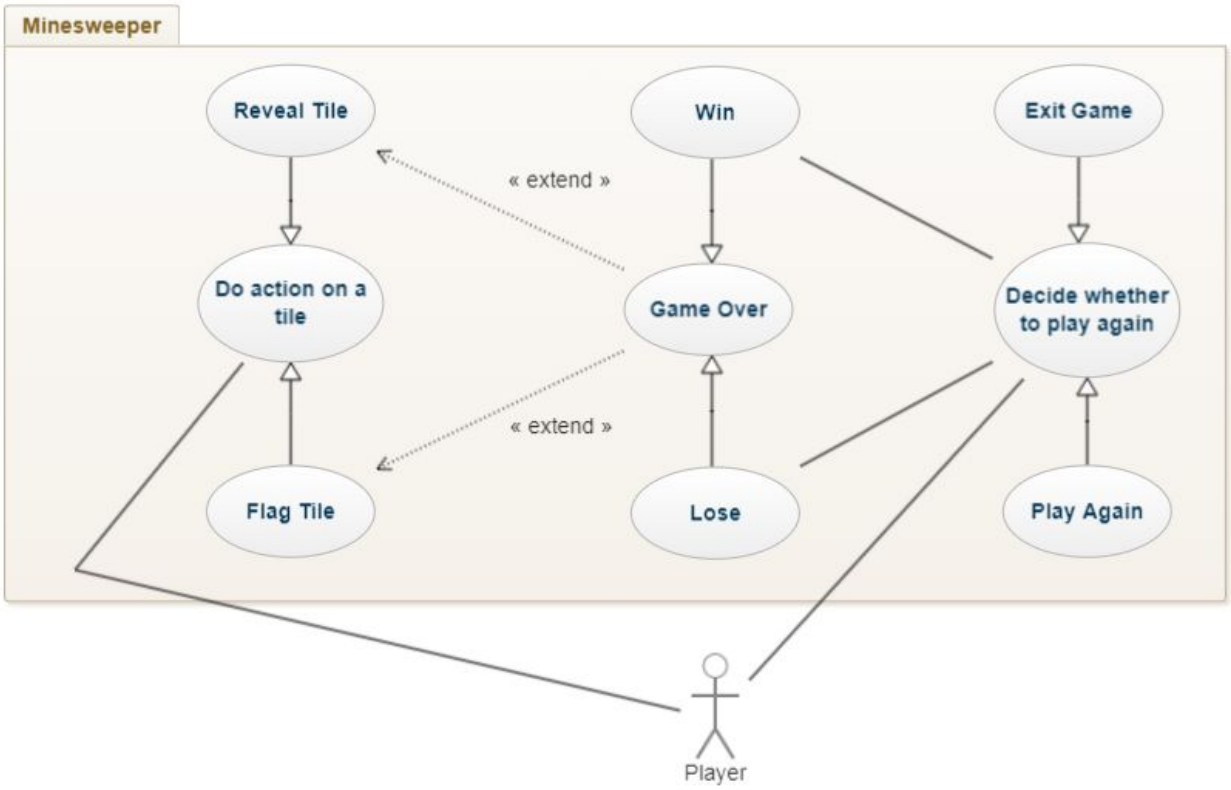
Constraints -

C1	The minesweeper game will have a maximum of one player
C2	The game can only be played on a desktop computer, but is not limited to one platform
C3	User must have 100mb of space free on their machine

Use Case FR1 -

Description - This functional requirement describes when a player is clicking a tile. The tile will either display a blank tile, a number or a mine. The user is only able to click on tile at a time.
Actors - The Actor in this use case is a player who is currently playing minesweeper.
Basic Flow - If a player clicks a tile containing a blank space, the blank tile is revealed and any connected tiles that are also blank will be revealed. Tiles containing a number will not search their surrounding tiles for blank spaces.
Alternate Flow 1 - If the user reveals a mine, the game is lost and the user needs to restart the game.
Alternate Flow 2 - If the user reveals a blank tile that has a mine in one of its surrounding tiles, a number will be revealed displaying how many mines it has in its surrounding tiles.
Alternate Flow 3 - If a player has won the game already, clicking on a tile containing a mine, the mine will reveal but the game will continue to display “You Win”.
Alternate Flow 4 - If a player has lost the game, clicking a tile containing a mine will do nothing, the tile will be unresponsive.

Use Case Diagram -



Project Risk -

Jumping into the main functionality of the minesweeper game can create a high risk of project failure. Dividing the project into phases allows each component of the game to be contain efficient code and reliable, consistent gameplay. Firstly the base of the minesweeper game will be developed - **Phase 1** includes the initial layout of the grid/board for all the tiles. This phase also includes getting the mines to randomly generate around the grid. One tile cannot contain two mines so checking will need to be involved when placing mines.

Phase 2 - This phase will focus on counting and displaying the number of tiles around a revealed tile. Once a tile is clicked it will need to firstly determine whether it is a mine or not. If the tile does not contain a mine, then search its surrounding 8 squares for any mines and display that number in the tile. If there is no mines surrounding that tile, do not display anything. There is a scenario where there is not a total of 8 tiles surrounding another tile such as in the top left corner of the grid. This tile only has 3 surrounding tiles. To solve this, an ‘inGrid()’ function will be made to determine if the tile is in the grid or not.

Phase 3 - This phase implements revealing multiple tiles when the user clicks on a tile that has no mines surrounding it. A recursive function can be used to check the contents of its surrounding tiles and decide whether or not to reveal them also. The statement from phase two can be reused to search all surrounding squares and then reveal them if they do not contain a mine. If the tile contains a number, then do not perform the recursive function again for that tile. The inGrid() function can be reused here to ensure the recursive function is not performed on tiles outside the game board.

Phase 4/ Final Phase - Once the game is playable, the game needs to be able to detect once the player has won or lost the game. The game will be ‘WON’ once the player has flagged (right clicked) all tiles containing a mine. The game will be ‘LOST’ if the player clicks on a tile containing a mine. A timer will also be implemented in this phase that starts when the user starts the game and stops when the game has been ‘LOST.’ The game will no longer be playable after winning or losing the game, so this phase will also implement a was to restart the game such as a key press, ‘ENTER’ for example.

Feasibility Report -

1. Introduction -

1.1 Minesweeper is a worldwide known game play throughout the last decade by player of all ages. The game originates from the 1960s, and has been written for many computing platforms in use today. Minesweeper is a single player game where the player is presented with a grid of tiles that can be clicked revealing either a blank space, mine or a number of how many mines are surrounding that tile. The aim of the game is to reveal all the tiles without mines, click a mine and the player loses.

1.2 The scope of this project is to create a game that replicates the famous 1960s minesweeper. This is a small scale project so the game must not be complex to ensure the product is developed on time to the clients standards.

1.3 Deliverables:

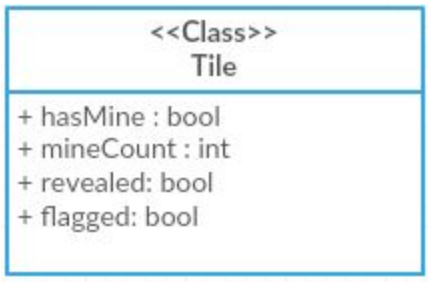
- Clicking a tile will reveal the contents of that tile (FR1)
- A tile can contain a mine, a number or will be blank (FR1)
- Numbers are displayed to show how many mines are in its surrounding tiles. (FR1)
- Clicking on a tile that contains a mine will lose the game (FR1)
- The game must be winnable (FR4)
- When an empty tile is revealed, connected empty tiles are also revealed. (FR1)

2. Feasibility Study

Splitting the project into phases allows comprehensive analysis of each requirement and constraint. The **first phase** is focused on creating the base of the game looking at requirement **NR1 and FR2**. The phase will be completed once the game is able to place mines around a 9x9 grid randomly each time the game is started.
Once the project reaches the **second phase**, the development of the contents of each tile will be implemented focusing on **FR1** as well as **NR2**. This phase will be concluded once the player is able to click a tile and the number of surrounding tiles with mines is displayed as its contents.
In **phase three** of the project, the functionality behind displaying blank tiles that are connected to a blank tile being clicked will be implemented (**FR1**). Once the player is able to click a blank tile and its connected blank tiles are also revealed, the fourth phase will be started.
The final phase of the project will add all three phases together to create a fun and playable game. The application should feel like you are playing a game. This phase focuses on requirements **FR1**, **FR3**, **FR4**, **FR5**, **NR3**, **NR5** and **C2**. This phase will add how the user can win and lose the game clicking a tile that contains a mine will lose the game. If the user right clicks (flags) all the mines on the board, then the user wins. If the game is lost or won, the user is able to press a key to restart the game. Once the game is started, a timer starting at zero will start to record how long the player takes to win the game.

The final phase of the project can be revisited to improve the overall feel for the game. These development phases will ensure the project will stay on track and remain in the dedicated budget and time frame. The phases are planned so that the most important requirements are handled first and the least important last. This allows enough time to work on the more complex tasks reducing the risk of failure.

Conceptual Design -



This class diagram represents one single tile in the grid. Each tile has a mine attribute to tell whether the tile contains a mine, a mineCount attribute to determine the amount of mines the tile is surrounded by, a revealed attribute to determine if the contents of the tile has been revealed and a flagged attribute to tell if the user has flagged the tile or not.

The tile class will be used in a 2d array, where each array will contain 9 tiles. The final layout of the game board/grid will look similar to below.