Team 8

Names:

Ahmed Youssef [2507690@dundee.ac.uk](mailto:2507690@dundee.ac.uk)

Arvin Shafiei 2503389@dundee.ac.uk

Usama Fakhar 2530869@dundee.ac.uk

AC22005 Assignment 1 – C# Grid Game Report

Minesweeper Game

Minesweeper is a classic grid game that involves the players uncovering hidden mines without detonating them. Our aim was to create a functional game that allows the user to switch between different difficulty levels while playing against the computer as an opponent. Making the game interactive as much as possible and having an appealing user interface for the user was something that we tried to achieve.

The code is organized into methods that handle specific and different aspects of the game, such as grid initialization, mine placement, button clicks, computer’s turn and game over scenarios. We began by designing the basic structure of the game by creating the grid for the game. The game grid itself is represented using a two-dimensional array of buttons and the presence of mines is tracked by another two-dimensional array ‘mines’.

For a user-friendly interface, we added a menu strip at the top where the player can choose the game difficulty they would like to proceed with as well as an about section to explain the game briefly. At the bottom there is another section that shows two scores, your current score as well as the highest score you have achieved during this game. Buttons were used to represent grid cells, and their click events were handled to reveal the content of the corresponding cell.

Core game logic was implemented, such as mines placement, adjacent mine counting and recursive cell revealing. The game maintains a score system where the player collects points for each safe cell that he uncovers. The computer opponent that plays alternate turns was added to make the game more challenging and interesting for the players.

The game has three difficulty levels: Easy, Medium, and Hard. The difficulty level determines the number of mines, score multipliers and the computer’s chance to ignore mines. In ‘Easy’ there are 10 mines, and each safe cell adds 1 point to the player’s score. ‘Medium’ increases the challenge with 20 mines and a score multiplier of 2. ‘Hard’ sets the skills level to max with 30 mines and a score multiplier of 3. The AI opponent implemented evaluates whether to click on a cell with a chance (aiBombChance) of ignoring a mine. The computer opponent attempts to explore areas with fewer adjacent mine, increasing its chances of survival.

One significant challenge that we’ve encountered was resetting the grid after the player’s turn and the computer’s turn. Resetting the grid involves disposing existing buttons, reinitialising the grid, and placing new mines.

Future possible enhancements:

Introducing a settings menu to allow the user to have more control over the game such as changing the grid size. Choosing custom grid sizes beyond the current fixed size 10x10, this would involve adjustments to the UI layout and game logic. This will allow more players with different preferences to engage with the game. The visual appeal of the game could also be improved by introducing more animations, for mine explosions for example. Having smoother transitions between turns could also enhance the overall user experience. Sound effects would also be a good addition to the game and would add some sound to the game since it has no sounds currently.

In conclusion, the code structure covers key game aspects, including grid setup, mine placement, and AI turns. The three difficulty levels provide varied challenges, while overcoming challenges in grid resetting. The Minesweeper implementation sets the stage for potential improvements, ensuring an enjoyable experience for different player preferences.