**INTERNATIONAL UNIVERSITY**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**



**DEPARTMENT OF INFORMATION TECHNOLOGY**

**SUBJECT**

**DATA STRUCTURES AND ALGORITHMS**

**LECTURER**

**TRẦN THANH TÙNG**

**GROUP 4:**

**Lê Minh Tấn ITITIU18236**

**Bùi Nguyễn Thiện ITITIU18238**

INDEX

1. OVERVIEW
2. OBJECTIVES
3. Implementation
4. FEATURES

1. Game rules

2. Goals

3. Algorithms and Data Structures

1. BUILDING SOFTWARE – DESIGNS

1. UML Diagram

2. GUI

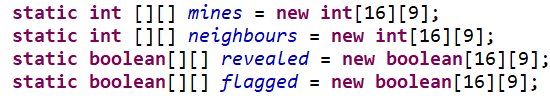
1. DEMO INSTRUCTION
2. Future progress
3. CONCLUSION
4. REFERENCES
5. OVERVIEW

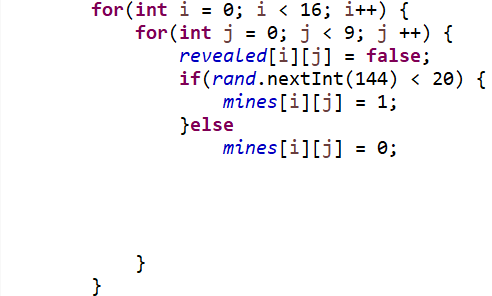
In this report, we will introduce you full vision that how our game works, and what Data Structures & Algorithms we use for this game. Firstly, the game named Minesweepers, like any others Minesweepers that Microsoft had developed, the rules and gameplay are the same. But we create this game depend on the knowledges we learned in school such as Queue to generate 2D arrays,…This game made by Java. Hopefully the Professor will satisfied.

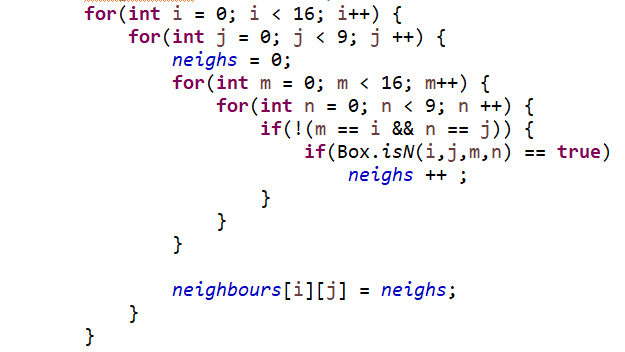
1. OBJECTIVES

### At the beginning, our target is to make a 3D version of Minesweeper by using Unity, but then we had some problem in using Unity and C#, so we decided to come back and made a basic version of Minesweepers. After finishing this project and finish this course, our goal is to make sure we have fully knowledge about Data Structures and Algorithms, a penchant for problem solving and thinking ability.

1. Implementation
   * We use aray for our project.
   * We stick with the simplest.







1. FEATURES
2. Game rules.

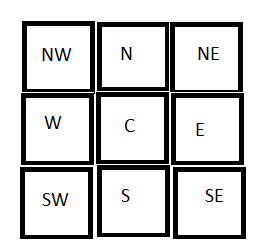
Our rule is very simple

* Use left click button on the mouse to select a space on the grid. If you hit a bomb, you lose.
* The numbers on the board represent how many bombs are adjacent to a square. The bombs could be above, below, right/left, or diagonal to the square.
* Put a flag where you think a bomb is, so you can avoid clicking that spot.

1. Goal.

A good game for relaxing and practice logical thinking ability.

1. Algorithms and Data Structures



N.W   N   N.E

              \   |   /

               \  |  /

            W----Cell----E

                 / | \

               /   |  \

            S.W    S   S.E

        Cell-->Current Cell

        N -->  North

        S -->  South

        E -->  East

        W -->  West

        N.E--> North-East

        N.W--> North-West

        S.E--> South-East

        S.W--> South-West

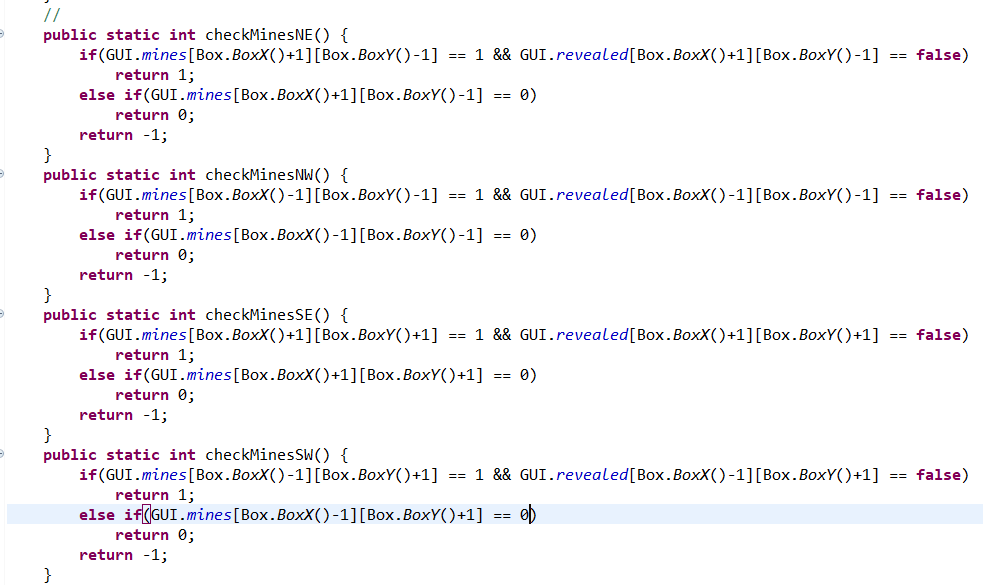
Explore Algorithm:

* We use North,East,South,West for this algorithm.

- We click open Cell it will open that Cell, and it will check around the cell has mines.

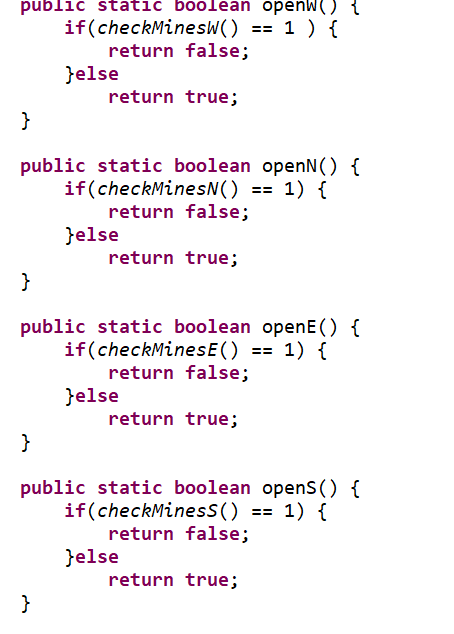


Check N, E, W, S

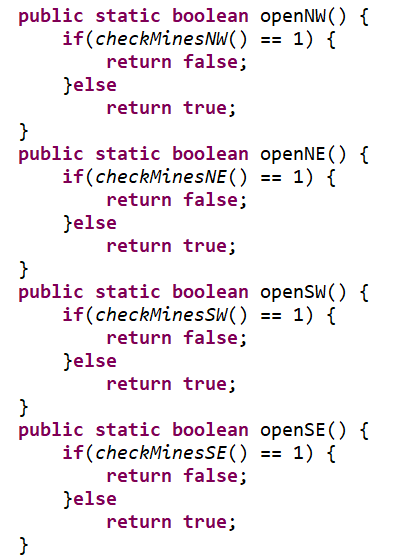


Check NE, NW, SW, SE

* Then it won’t open that mines or if it’s not mines, it will open.



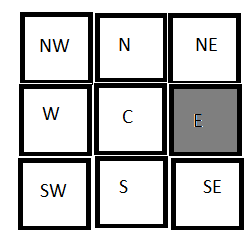
Open N, E, W, S



Open NE, NW, SW, SE

-If North, East, South, West, North West, North East, South West, South East have mines, they won’t open that cell.

Ex:



Gray won’t open because it is mine.

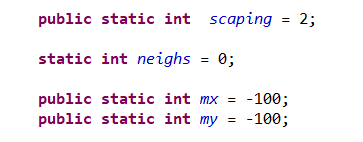
V. BUILDING SOFTWARE AND DESIGN

1. UML Class Diagram

Diagram, schematic

Description automatically generated

2. GUI



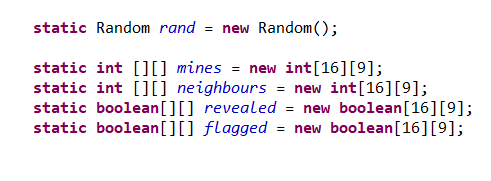


Fig 2.1: Create interface, main board and set status

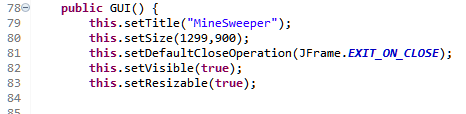


Fig2: Set size, title, canvas

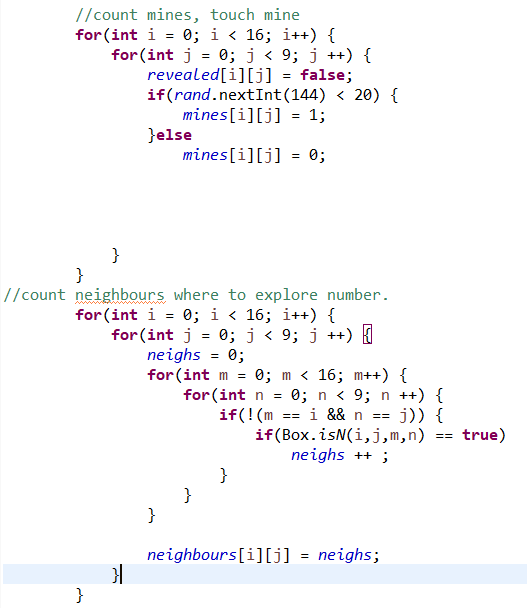


Fig 2.2: Create bomb, neighbours means number of bomb around the square, revealed is number in square.

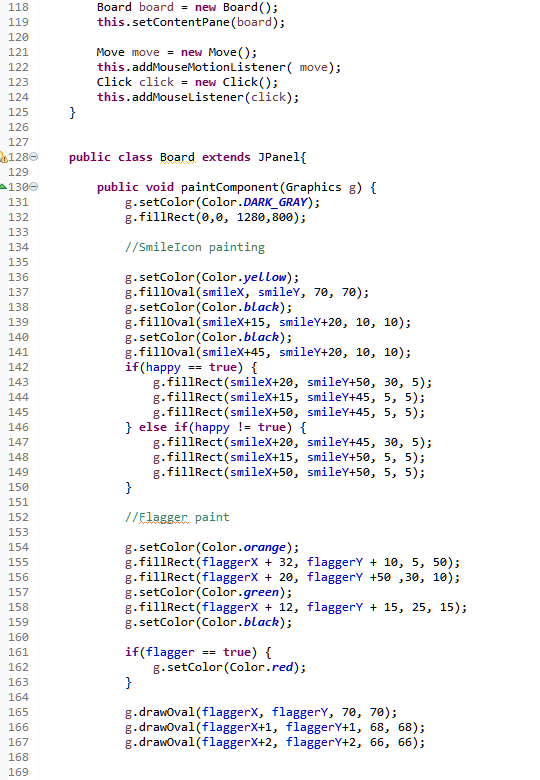


Fig 2.4: Create main interface.

We set color dark\_grey to fill in the board in order to make it more colorful and has a better look for users. Set color = yellow so whenever you move the mouse to a square, it will highlighted by yellow color. Flagger paint is some function we color the flag.

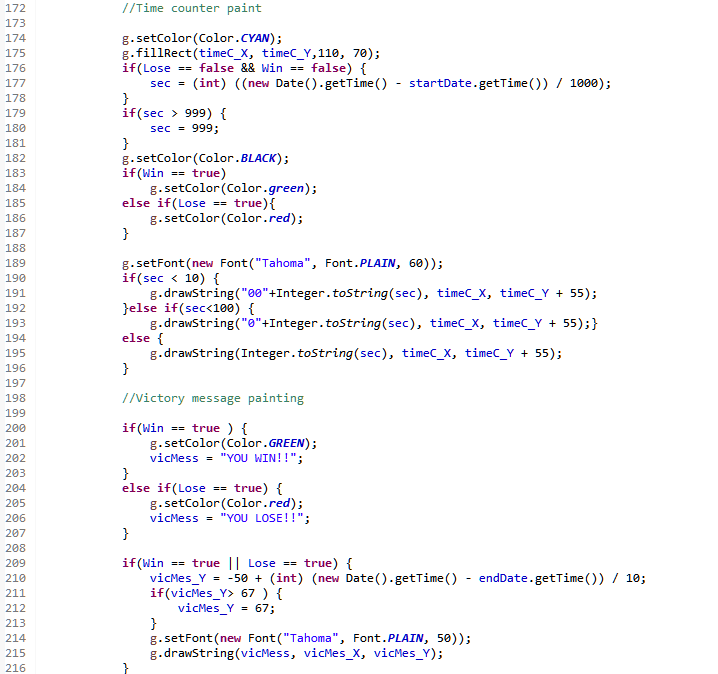


Fig 2.5: Time count and win message

We set time in the right corner that player can observe that they spend how much time in playing game. If times > 999s, we set the times equals 999s.

Black and green color represent color of time and a frame that cover the number of time to look easier.

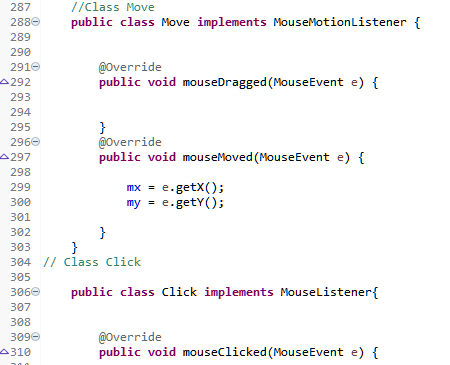


Fig 2.6: Create Mouse interaction

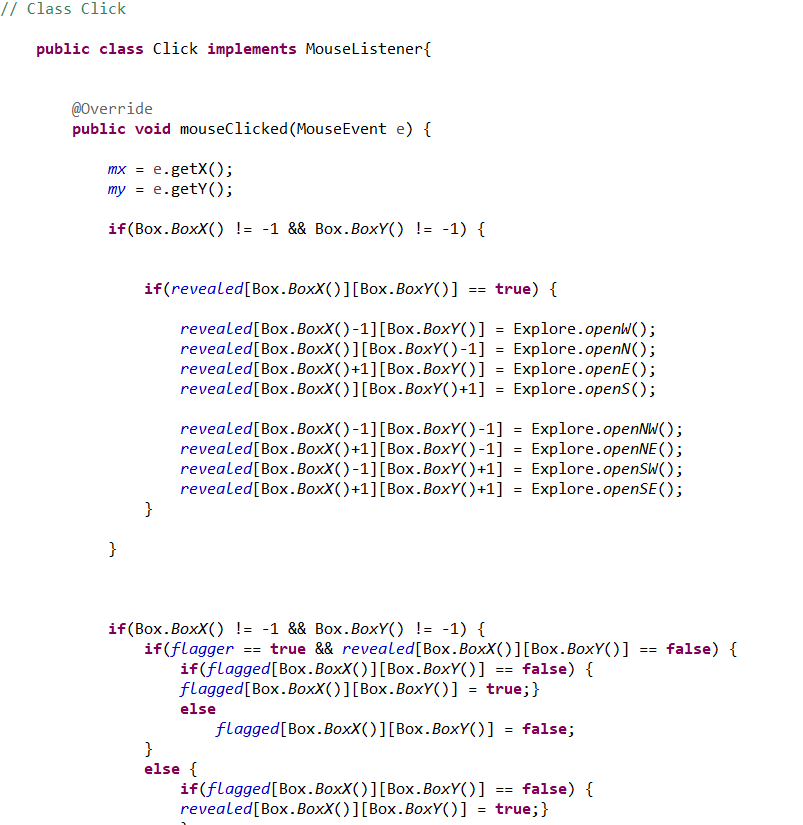




Fig 2.7: Click button

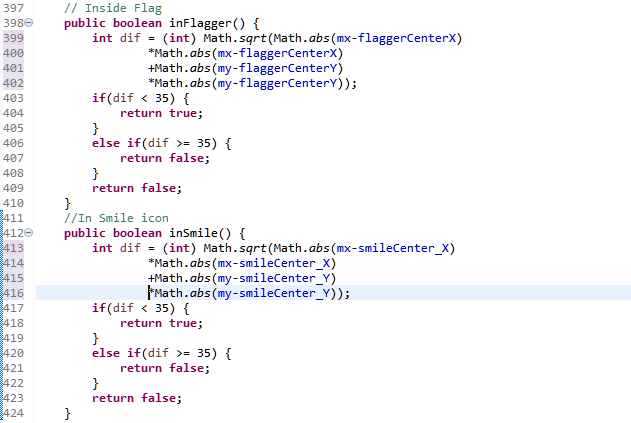


Fig 2.8: Create flag icon and smile icon in main board

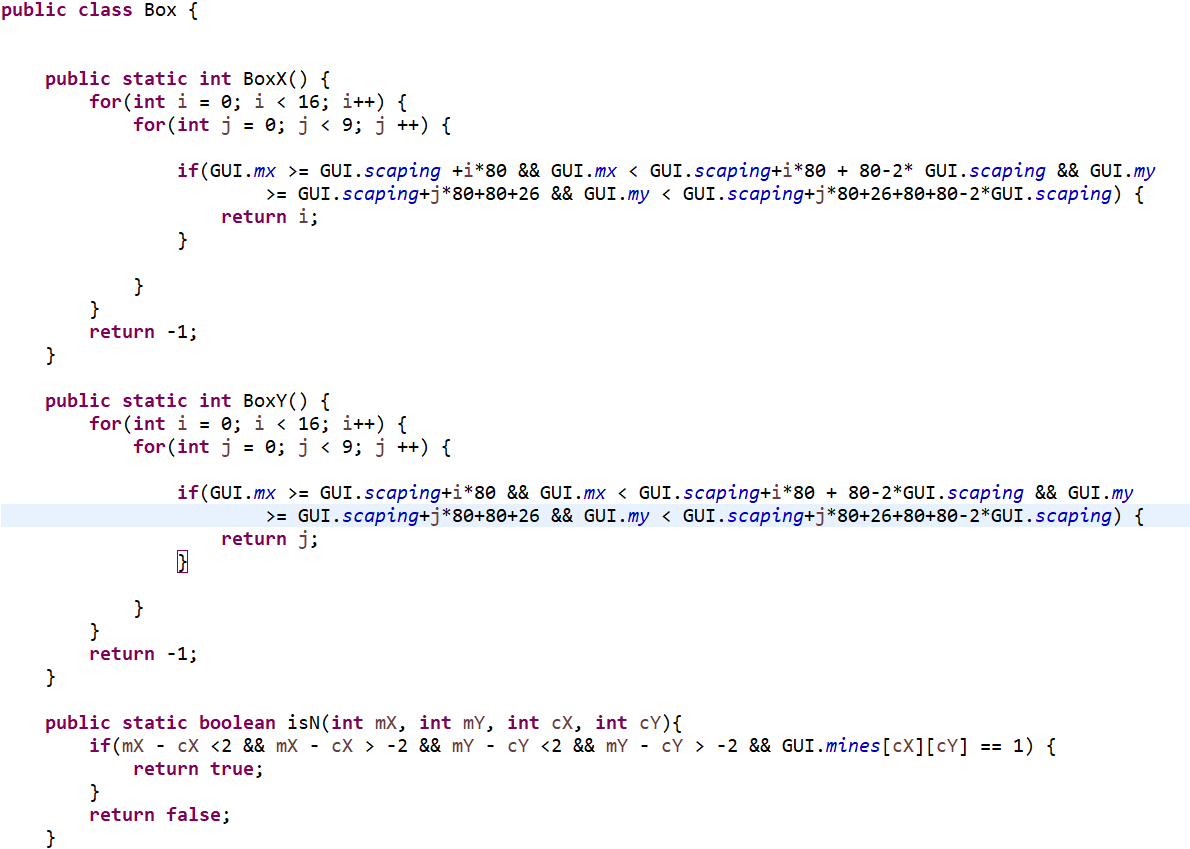
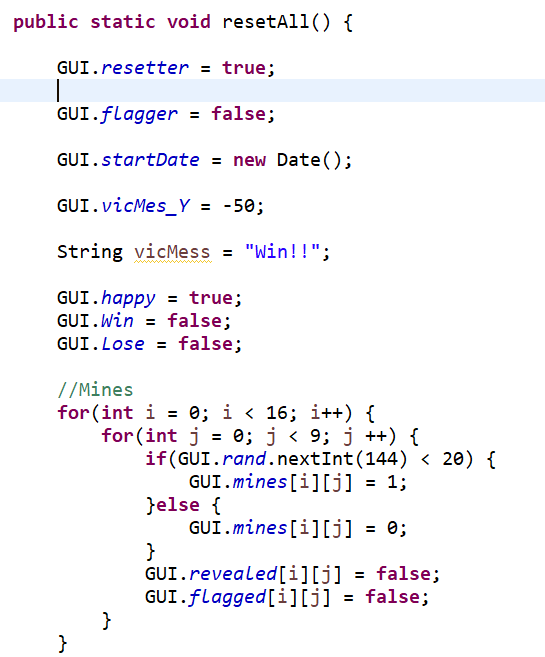


Fig 2.9: Position of the cell



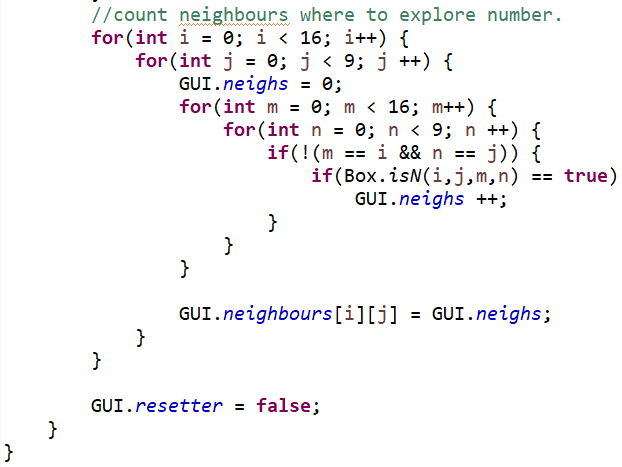


Fig 2.10: Reset the game

VI. DEMO INSTRUCTION

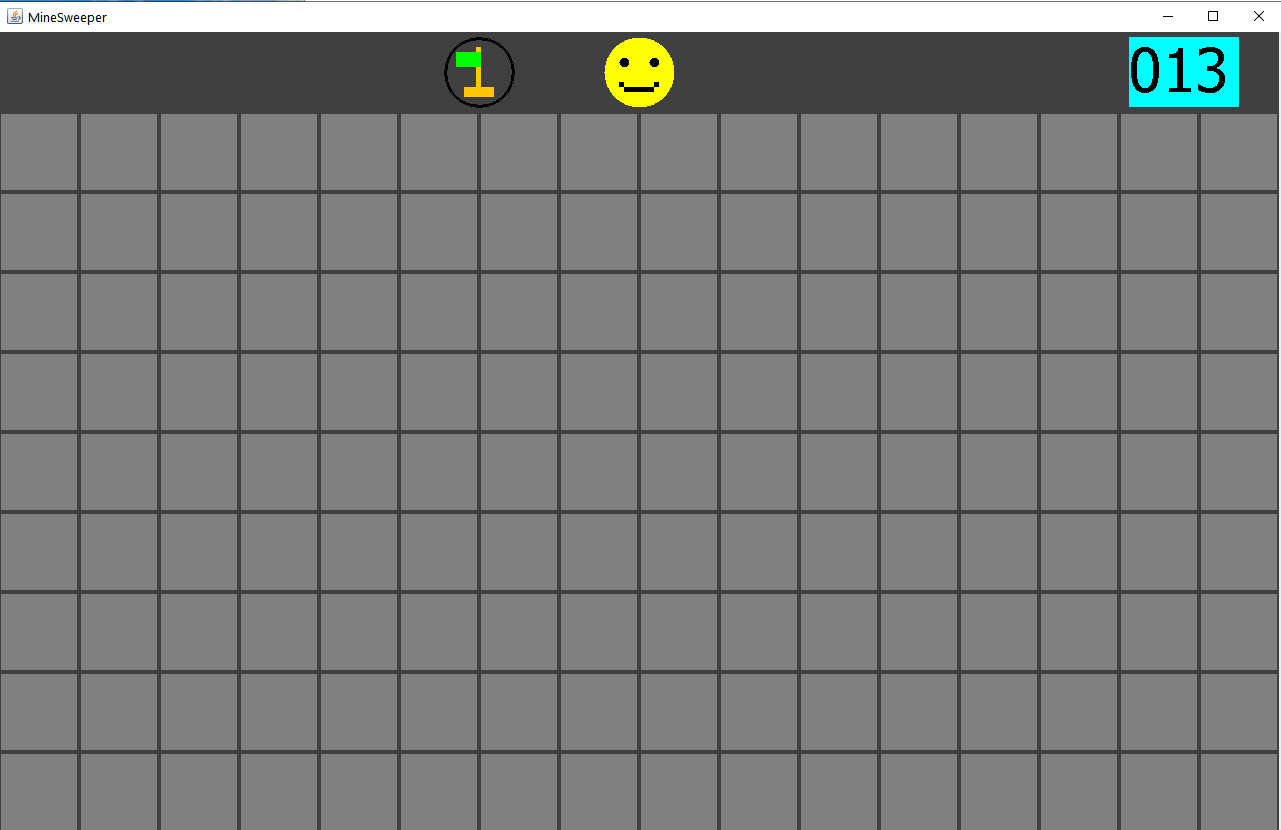
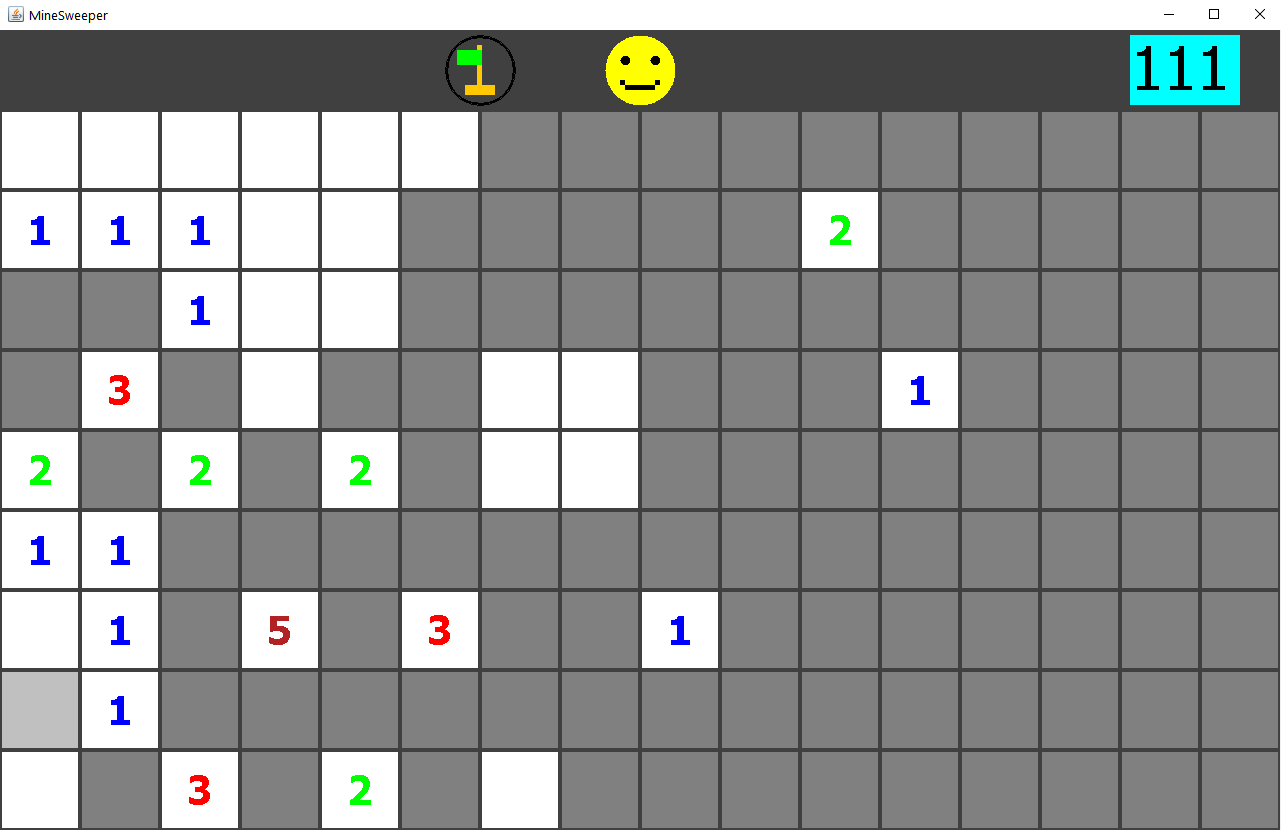


Fig 1: Game’s interface

Fig 2: Left click to play and avoid the bombs

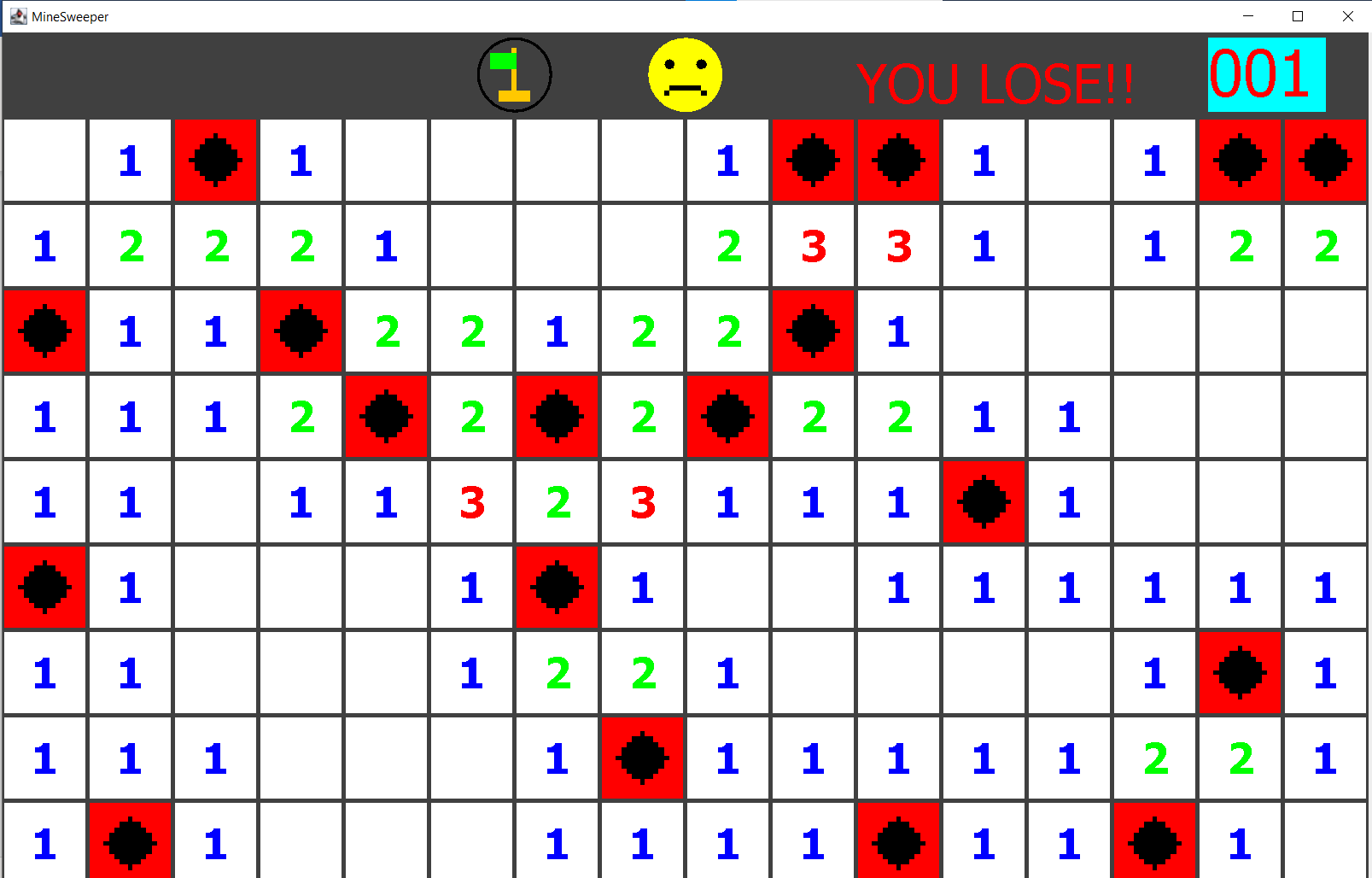


Fig 3: When you hit a bomb, the game will auto generate all the cells.



Fig 4: Find all the numbers and avoid bomb to win

VII.Future progress

We have some bugs and not perfect algorithm, so we will update,

* Add undo, redo feature for our game.
* Add easy, hard, extrem mode.
* Use C# instead Java for unity 3D.

VIII.CONCLUSION

After invented this game, we realize that it is hard to put Algorithms to code and fix bugs. Another issue is that our codes are so complexity, but we do not know how to code clean so it look very disorders, hopefully in the future we can fix it and make a new version of Minesweeper to everyone who like the origin game

VII. REFERENCES

Youtube: <https://www.youtube.com/watch?v=RFpJp62ZoY8&list=PLGxHvpw-PAk6QvPw0fYe8bks31GRKvymK>

Document:

https://www.geeksforgeeks.org/cpp-implementation-minesweeper-game/