Final Project

Algorithm and Programming

Project Name: “Minesweeper”

Student Name: Jessica Angela Huang

Student ID: 2602213031

Class: L1BC

Table of Contents

A. Description

I. Introduction

As soon as I heard that there is a final project for Algorithm and Programming, I started to consider what python game I should do. Then, I browsed for a simple Python project that I think would enjoy doing and will fulfill the final project's requirements. In the end, I came across a game that I used to play in my high school to pass the time, called Minesweeper. It was a very simple game that I really enjoyed back then. Minesweeper is a logic-based single-player computer game played on a rectangular board. The objective is to find a specified number of randomly placed "mines" in the shortest possible time by clicking on "safe" squares while avoiding mines. The game ends when the player clicks on a mine. This game is a really good time killer and uses a bit of brain power but not too much that makes your brain hurt.

II. The function of this program

The function of this program is to run the game, which can be played offline with just the use of python and pygame.

B. Design

I. Parts of the program and requirements

Requirements:

- PyGame: used to design the overall game and visualize it into a window.
- Datetime: used to get current date and time
- "from random import randint" : a statement which imports the randint function from the random module, allowing you to generate random integers. The randint function takes two parameters, start and end, which specify the range of integers you want to generate.

II. Function of each part of the program

Firstly, I define all the variables like game variables

```
# Game Variables
GRID_SIZE = 10
CELL_SIZE = 26
MINE_COUNT = 20
TIME = 0
```

GRID_SIZE = the size of the grid
CELL_SIZE = to cize of cell
MINE_COUNT = to set the number of mines (aka bomb)
TIME = time played
And also define the color so it can be the exact color that i want

```
black   = (   0,   0,   0)
white   = ( 255, 255, 255)
lightgrey = (238, 233, 233)
green   = (   0, 100,   0)
blue    = ( 100, 149, 237)
darkblue =(   0,   0, 205)
red     = ( 255,   0,   0)
purple  = (  25,  25, 112)
darkred = ( 102,   0,   0)
iceblue = (   0, 204, 204)
grey    = ( 128, 128, 128)
```

Next, I calculate the variables for the set up

```
MENU_BAR_HEIGHT = 30
MENU_BAR_WIDTH = GRID_SIZE * CELL_SIZE
MAX_SCREEN_HEIGHT = GRID_SIZE * CELL_SIZE + MENU_BAR_HEIGHT
MIN_SCREEN_HEIGHT = MENU_BAR_HEIGHT
MAX_SCREEN_WIDTH = MENU_BAR_WIDTH
MIN_SCREEN_WIDTH = 0
MAX_COL_COUNT = GRID_SIZE - 1
MAX_ROW_COUNT = GRID_SIZE - 1
MENU_BUTTON_X = 1
MENU_BUTTON_Y = 1
GAME_STATE = "setup"
```

C. Implementation

I. Classes Diagram

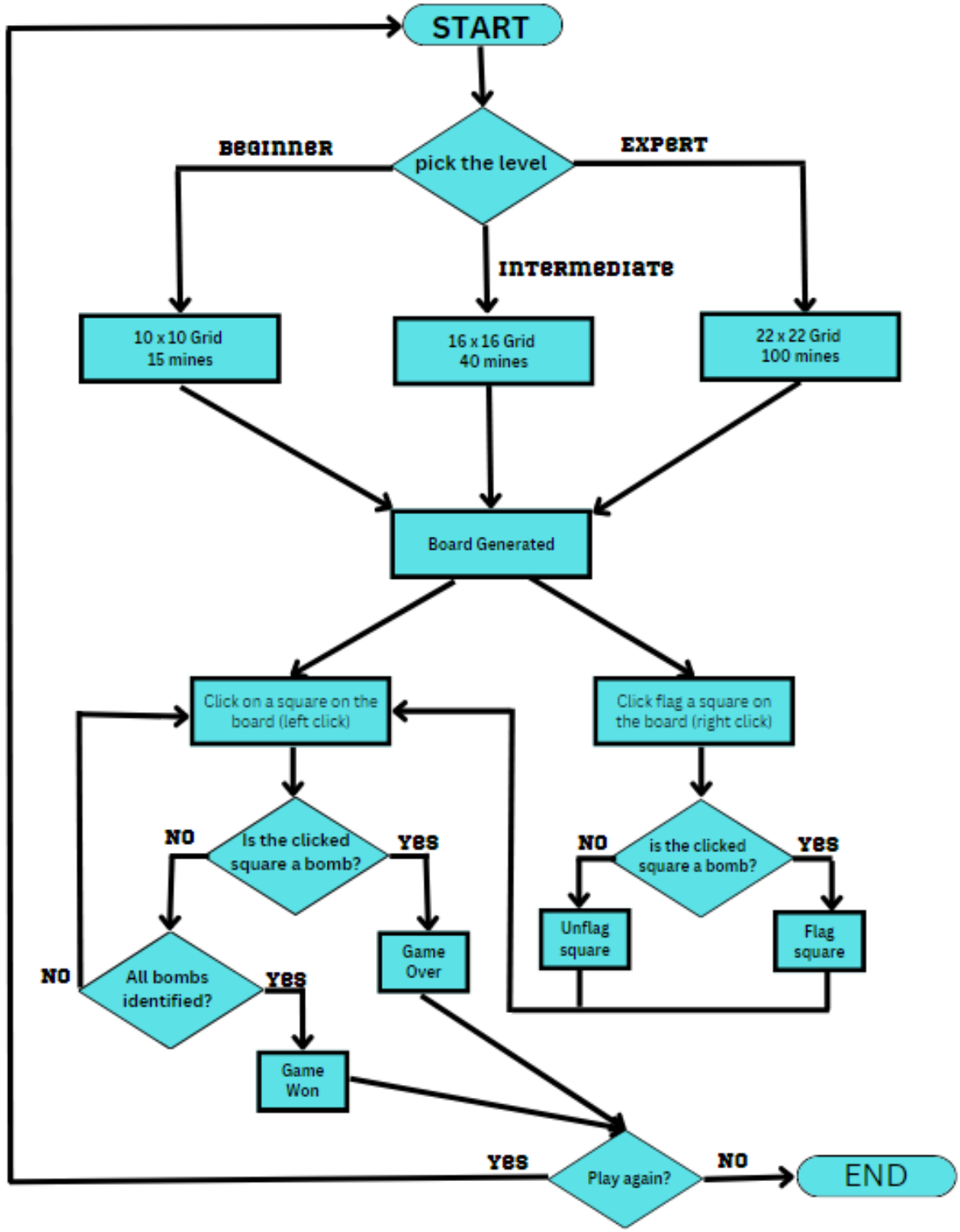
```
class boardSpot(object):
    value = 0
    selected = False
    mine = False

    def __init__(self):
        self.selected = False

    def __str__(self):
        return str(boardSpot.value)

    def isMine(self):
        if boardSpot.value == -1:
            return True
        return False
```

This Class is for a boardSpot object.
The class has three class-level variables: value, selected, and mine. value is initialized to 0, selected and mine are both initialized to False.
The __init__ method is called when a new boardSpot object is created, and sets the selected attribute to False.
The __str__ method returns the string representation of the value attribute when the class is printed.
The isMine method returns True if the value attribute is equal to -1, otherwise it returns False.
It looks like this class is being used to represent a spot on a board game, with a value, selected status, and mine status.

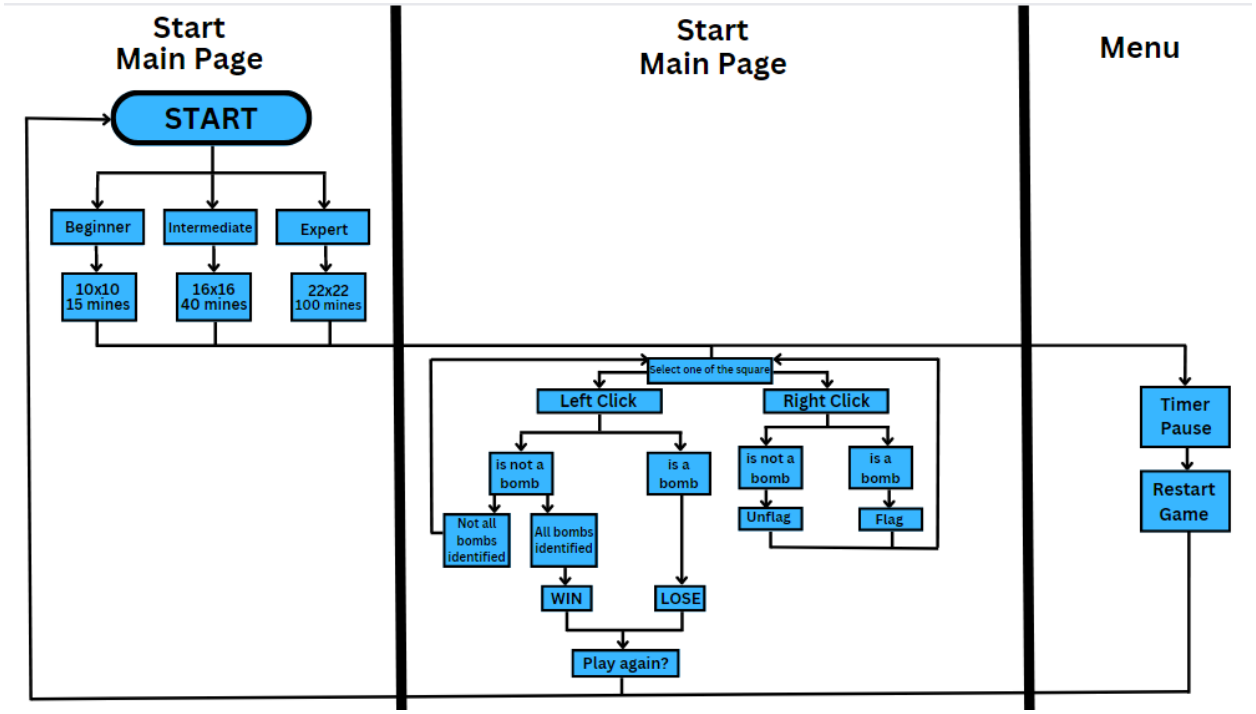
```
class boardClass(object):
    def __init__(self, m_boardSize, m_numMines):
        self.board = [[boardSpot() for i in range(m_boardSize)] for j in range(m_boardSize)]
        self.boardSize = m_boardSize
        self.numMines = m_numMines
        self.selectableSpots = m_boardSize * m_boardSize - m_numMines
        i = 0
        while i < m_numMines:
            x = random.randint(0, self.boardSize-1)
            y = random.randint(0, self.boardSize-1)
            if not self.board[x][y].mine:
                self.addMine(x, y)
                i += 1
            else:
                i -= 1
```

The "boardClass" class represents the game board. The class has an __init__ method that is run when an object of the class is created. This method takes in two arguments, m_boardSize and m_numMines, which represent the size of the board and the number of mines on the board, respectively. The method initializes the board attribute as a 2-dimensional list of boardSpot objects, with m_boardSize number of rows and columns. It also sets the boardSize and numMines attributes to the input values, and calculates the number of selectable spots on the board. It then uses a while loop to randomly place m_numMines number of mines on the board by adding mines to the board attribute at random x and y coordinates, ensuring that the spot does not already have a mine.

II. Use-case Diagram



III. Activity Diagram



IV. Extensibility

1. Comment lines are used to further explain the functions of each function and method and the purpose of each of them to help other programmers.
2. Variables use meaningful identifiers to help other coders to understand easier.
3. Class diagram and flowchart to make it easier for others to understand and to easily identify features.

V. Explanation of all the functions made and used

I commented on all the uses of each function in my main.py to make it easier to understand.

D. Lessons learned

I. Error Handling and Debugging

I feel that it can be very frustrating and confusing having to solve a problem especially if it were to be a silly mistake, for example that i forgot to add a symbol somewhere, which may cause a big error in my code. Even so, I found my own way to figure out what's wrong with the code. Firstly, I will check the line in which the error is occurring. Then I check my code for typos because maybe I accidentally missed a parenthesis or comma while writing. Sometimes this may take a while because I might not be aware of it. If everything is fine, then I check what kind of error is in my code as there is some problem with my code's logic. Another way is that when I run the code, usually there will be an explanation of the error in my terminal, which I usually try to solve the problem based on the terminal instructions. If I can't fix the problem, will maybe ask my friend to help me out.

E. Evaluation

I. Does the program work properly?

Yes, it does as long as you have the requirements, the program should be running well and then you can enjoy playing the game again and again without having to run the program again and again to reset the game as you can just restart the game from the menu.

II. Future Improvements that can be done

I want to make the mainpage better, because I feel that the main page is too simple and that if possible I want to make a much more complex game than this, but because of the limited amount of time, I guess this game is quick and easy to make.

F. Evidence of working program

I made a readme program in my github with screenshots of evidences of the working program