



**BINUS UNIVERSITY**

**BINUS INTERNATIONAL**

Assignment Cover Letter

(Individual Work)

**Student Information:**

**Surname:** Claudio

**Given Name:** Alexandro Joe

**Student ID Number:** 2501963160

**Course Code :** COMP6047001

**Course Name :** Algorithm and Programming

**Class** : L1AC  
MCS

**Lecturer** : Jude Joseph Lamug Martinez,

**Type of Assignments:** Term Final Project

**Submission Pattern**

**Due Date** : 17 January 2022      **Submission Date** : 17 January 2022

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer's instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

**Plagiarism/Cheating**

BiNus International seriously regards all forms of plagiarism, cheating, and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity, and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

**Declaration of Originality**

By signing this assignment, I understand, accept, and consent to BiNus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student:

# TABLE OF CONTENTS

## I. Overview

- A. Genre
- B. Game Methods
- C. Visual
- D. Objective

## II. Program Implementation

- A. Screen
- B. Cells (button)
- C. Mines
- D. User Input (left click and right click), Win Statement, Lose Statement
- E. Cell Counter

## III. References

# I. Overview

## A. Genre

The game that I created for this final project is minesweeper, however minesweeper is not a new game that people are not familiar with. It is an old game that people used to play on an old computer. Minesweeper's genre is single player puzzle video games.

## B. Game Methods

In this game, the player will be guessing which cell is not mine. They can use the left click on the mouse to reveal the cell that they want to reveal. As far as I know when I played this game as a kid, I don't remember that there was a right click function to click the cell, so in this game I added a right click function to mark the possible mine so that users can play the game with strategy and not just clicking on the boxes randomly.

## C. Visual

Since it was released in 1992, I can tell that this game display is pretty simple. The top area is for the title of this game, which is minesweeper. I use the left area of the display to show how many cells are left that are not a mine. The main part of this game is in the right area where I put 49 cells that users can click and play with.

## D. Objective

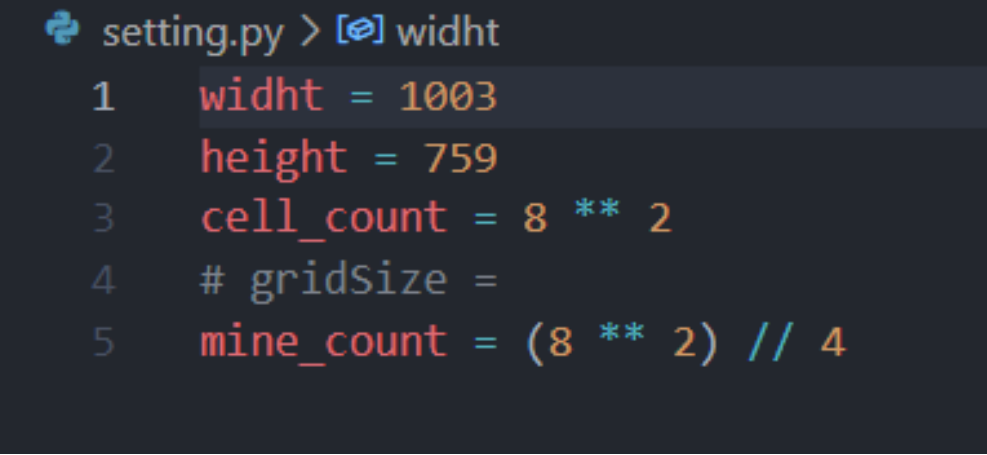
The main goal of this game is to reveal as many cells without limited time and avoiding mines. When a user starts this game, they can immediately press the cell that they want to, if it is not a mine the cell will show a number. The number shows how many mine(s) are in the area of the cell that the user clicked. If the user presses the cell that is a mine, there will be a message display showing that they lost this game, however if they can open the cell with the same amount of the mine (which I set  $\frac{1}{4}$  from the total cells) there will be a message showing that they win the game. The right click function is set to make the user predict the cells around

them. I found that this right click function made me think more and carefully click on the cells when I was trying out this game.

## II. Program Implementation

### A. Screen

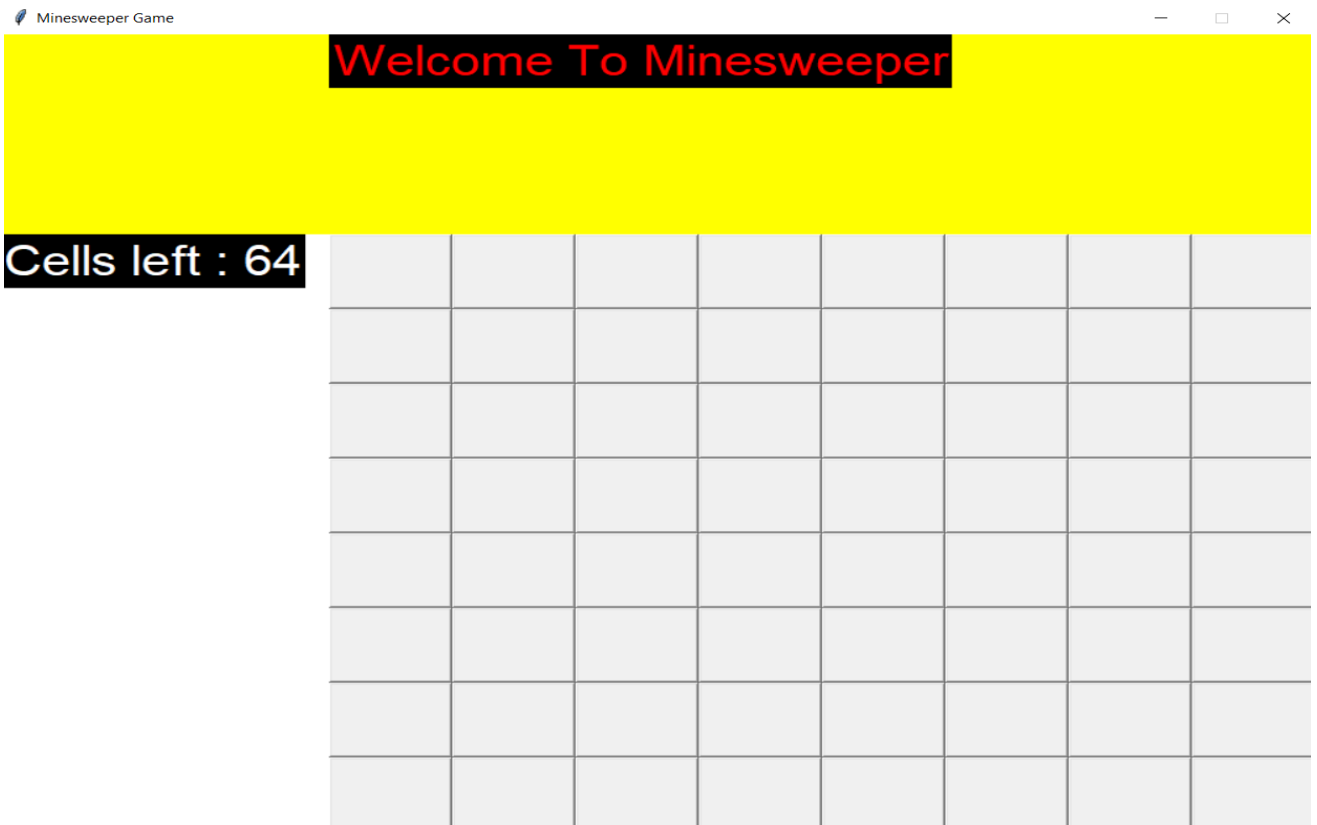
In this project I use the Tkinter GUI toolkit. For the display screen of the Tkinter I set it width = 1003, height = 59.



```
setting.py > [🔍] widht
1  widht = 1003
2  height = 759
3  cell_count = 8 ** 2
4  # gridSize =
5  mine_count = (8 ** 2) // 4
```

I set the width, height, cell\_count, and mine\_count on a different file so that the main file and the file that I use to run the logic will be clear, because I use those to store my numbers. Why there is no rounding on the width and height, that is because I did trial and error to make the display perfect for the screen.

The screen divided into 3 parts and I will show the exact area with a picture below



The yellow part is the top\_frame variable, the white part where I put the Cells Left is on the left side, and the rest is for the cells that the user can play.

```
utils.py > percentage_widht
1 import setting
2
3 def percentage_height(percentage):
4     return (setting.height / 100) * percentage
5
6 def percentage_widht(percentage):
7     return (setting.widht / 100) * percentage
```

I made another file called utils to store these functions. These functions will calculate the percentage of the width and height. I got the exact number of height and width by importing the setting file. I also separate this file because I think that it will make the

other file neat and also to avoid hard coding by doing mathematical functions straight on the main file.

```
root.configure(bg="blue")
root.geometry(f'{setting.widht}x{setting.height}')
root.title("Minesweeper Game")
root.resizable(False, False)
```

From the image above, I just want to explain some of it. From the top I use the background as blue but there is no blue colour on the game screen that is because the background is already overridden by the code that is on the image below. Geometry function on the second line is to change the width and height of the screen itself, the exact value of those already stored inside setting file and I just import it. Because I want to avoid the user resizing the game window, I use the resizable function. I set both width and height to false so that it makes the window not resizable.

```
top_frame = Frame(root, bg= "yellow", width = utils.percentage_widht(100), height = utils.percentage_height(25))
top_frame.place(x = 0, y = 0)

left_frame = Frame(root, bg = "white", width = utils.percentage_widht(25), height = utils.percentage_height(75))
left_frame.place(x = 0, y = utils.percentage_height(25))

center_frame = Frame(root, bg = "violet", width = utils.percentage_widht(75), height = utils.percentage_height(75))
center_frame.place(x = utils.percentage_widht(25), y = utils.percentage_height(25))
```

These variables are set to override the root.configure variable that I mentioned in the previous paragraph. For the example to explain the percentage I use top\_frame, I set the width 100% from the screen so that it fills the width of the screen, for the height I set it to 25% of the screen an. For the coordinate I use the .place function so that the user can set the exact coordinate to start the frame.

## B. Cells (button)

I made this part title's as cells (button) because this game's main attraction is similar to cells, but it is basically a button that the user can press with left click or right click to interact with the buttons.

```
def create_button(self, position):
    btn = Button(
        position,
        width = 12,
        height = 4
    )
    btn.bind('<Button-1>', self.left_click )
    btn.bind('<Button-3>', self.right_click )
    self.cell_button_object = btn
```

First thing to do when I want to create a cell is to make a function. I use the Button command from Tkinter to immediately create a button. In this function I will set the width and the height to 12 and 4. On the last 3 lines of the code we can see bind functions, the bind functions are used to deal with the events. In this image it deals with user input of left click and right click, which will be explained further for left and right click. At the last line of the code I make the self.cell\_button\_object is equal to btn which I already set, I use the self.cell\_button\_object inside the init functions.

```
for x in range(8):
    for y in range(8):
        c = Cell(x, y)
        c.create_button(center_frame)
        c.cell_button_object.grid(column = x, row = y)
```

Continuing the cells making step, this is the nested loop that I use to make all the cells on the screen. At the create\_button I pass the center\_frame to give the information of the location. After that I used the grid method, I think that it is very simple to use and I pass x as column and y as the row. After these few lines it generates an 8x8 grid.



## C. Mines

```
@staticmethod
def random_mines():
    selected_cells = random.sample(Cell.all, setting.mine_count)
    for selected_cell in selected_cells:
        selected_cell.mine = True
```

Static method is a method that does not belong to each instance, beside it belongs globally to the class. To randomise the mine I use a random.sample function, it is an inbuilt random module that returns a particular length list of items and in this case is from a list (Cell.all is a list). For this random I set how many mines that I want to randomise by setting it on the setting file.

```
setting.py > ...
1  width = 1003
2  height = 759
3  cell_count = 8 ** 2
4  mine_count = (8 ** 2) // 4
```

After using the random.sample function I use for loops, to make the attribute .mine from false into true. Below is the upper section of this file showing that mine attribute was set to false and it turns into true after going from the for loop.

```
def __init__(self, x, y, mine = False):
    self.mine = mine
    self.candidate_mine = False
    self.opened = False
    self.cell_button_object = None
    self.x = x
    self.y = y
```

## D. User Input (left click and right click), Win Statement, Lose Statement

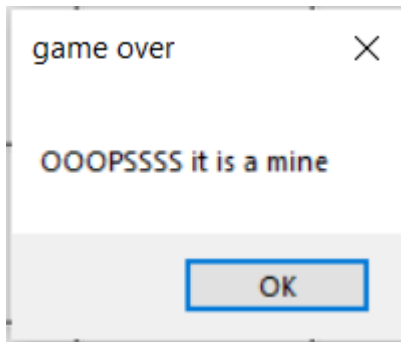
This game will take user input by clicking the right click or the left click on the mouse. Right click will “mark” the cell that is suspicious for being a mine, and of course the left click is to reveal the cell.

```
def left_click(self, event):
    if self.mine:
        self.showMine()
    else:
        if self.cells_around_mines == 0:
            for x in self.cells_around:
                x.showCell()
        self.showCell()
        if Cell.cell_count == setting.mine_count:
            ctypes.windll.user32.MessageBoxW(0, 'YESSSS you won this time, another try ?', 'game over', 0)
        self.cell_button_object.unbind('<Button-1>')
        self.cell_button_object.unbind('<Button-3>')
```

Talking about the left click first, if the user clicks on the cell that is mine then it will run the showMine function.

```
def showMine(self):
    self.cell_button_object.configure(bg="red")
    ctypes.windll.user32.MessageBoxW(0, 'OOOPSSSS it is a mine', 'game over', 0)
    sys.exit()
```

This showMine function will display a red colour as the background of the cell that we clicked. After we press the mine, the game will stop and it will show a dialog box that I import from ctypes library and the last thing that we want to do is, if the user presses “OK” it will exit the screen.



Going back to the first image of this part, there is an else statement, with if and for loops. Basically the program will tell how much mine is around the cell that the user clicked (if it is not a mine), but then there is a special case. If the cell that the user clicked shows 0 mine around it, then it will open one cell on right, left, up, down, diagonal top right, diagonal top left, diagonal bottom right, and diagonal bottom left. This is the code to implement the logic.

```
@property
def cells_around(self):
    surround = [
        self.cell_by_axis(self.x - 1, self.y - 1),
        self.cell_by_axis(self.x - 1, self.y),
        self.cell_by_axis(self.x - 1, self.y + 1),
        self.cell_by_axis(self.x, self.y - 1),
        self.cell_by_axis(self.x + 1, self.y - 1),
        self.cell_by_axis(self.x + 1, self.y),
        self.cell_by_axis(self.x + 1, self.y + 1),
        self.cell_by_axis(self.x, self.y + 1)
    ]

    surround = [cell for cell in surround if cell is not None]
    return surround
```

But then there will be an error if the user clicks on the most left cell or I can say at the corner. That is because there will be either no right, left, up, down, diagonal top right, diagonal top left, diagonal bottom right, and diagonal bottom left. If it is happening, it will return none on the list, which I don't want to happen. So 2 last line of the image is to make the list don't have the none value.

Next up is the win statement, if the cell left is equal to the mine that I already set, then there will be a dialog box showing that the user won the game. Below is the code

```
if Cell.cell_count == setting.mine_count:  
    ctypes.windll.user32.MessageBoxW(0, 'YESSSS you won this time, another try ?', 'game over', 0)
```

```
self.cell_button_object.unbind('<Button-1>')  
self.cell_button_object.unbind('<Button-3>')
```

For the last 2 lines on the left click function is responsible to cancel the event. So that after the user did the left click and the user tried to right click again, the colour of the background won't change.

```
def right_click(self, event):  
    if not self.candidate_mine:  
        self.cell_button_object.configure(bg = 'orange')  
        self.candidate_mine = True  
    else:  
        self.cell_button_object.configure( bg = 'SystemButtonFace')  
        self.random_mines = False
```

The right click function will make the right clicked cell become orange to mark it and return it to True, because I set it to False.

## E. Cell Counter

```
@staticmethod
def create_cell_counter_label(position):
    lbl = Label(
        position,
        bg= "black",
        fg= "white",
        text=f"Cells left : {Cell.cell_count}",
        font = ("Comic Sans", 30)
    )
    Cell.cell_label = lbl
```

Cell counter will count how many non mines cells that are left on the game, in this function I also use the static method same like the random mines function. Label is a widget that we can display a text or image inside of it. Inside of the text there is cell\_count function that is responsible for displaying the cell number

```
class Cell:
    all = []
    cell_label = None
    title = None
    cell_count = setting.cell_count
```

I imported the cell\_count from other file (setting.py) and make a variable on this file

```

def showCell(self):
    if not self.opened:
        Cell.cell_count -= 1
        self.cell_button_object.configure(text=self.cells_around_mines)

        if Cell.cell_label:
            Cell.cell_label.configure(text=f"Cells left : {Cell.cell_count}")

        self.cell_button_object.configure(
            bg = 'SystemButtonFace'
        )
    self.opened = True

```

In this class, I want to check if the cell is open or not, if it is not open then it will reduce the cell count by 1, show on the cell how many mines around the cell, and also give the latest update of the cell left. After that I want to set the background colour to be grey. On the last line of the code it will return the value True because I set it False at first.

```

@property
def cells_around_mines(self):
    counter = 0
    for cell in self.cells_around:
        if cell.mine:
            counter += 1
    return counter

```

Cells around mines function is used to count how many cells that are mines on the area of the cells, the area of the cell is already explained on page 10.

# References

- Modules:
  - a. Tkinter
  - b. Random
  - c. Sys
  - d. Ctypes
- Video References
  - a. <https://www.youtube.com/watch?v=Ejw7Lc9zlyU&t=45s>
  - b. <https://www.youtube.com/watch?v=XTT8mXwIGpQ>
  - c. [https://www.youtube.com/watch?v=\\_Aycjxb9Pkg&t=522s](https://www.youtube.com/watch?v=_Aycjxb9Pkg&t=522s)