Міністерство освіти і науки України

Донецький національний університет імені Василя Стуса

Факультет інформаційних та прикладних технологій

ЗВІТ

з дисципліни «Технології Програмування»

Творчє Завдання

Виконав студент 2 курсу

Групи Б21_Д/121_1_Б

Колібабчук Д. І.

Перевірив:

Антонов Ю.С.

https://github.com/JeffTheK/Saper

Правила Гри

Гравець відкриває комірки, намагаючись не відкрити комірку з міною. Відкривши комірку з міною, він програє. Якщо під відкритим осередком міни немає, то в ній з'являється число, що показує, скільки осередків, що є сусідами з щойно відкритою, «заміновано» (у кожному варіанті гри сусідство визначається по-своєму); використовуючи ці числа, гравець намагається розрахувати розташування мін, проте іноді навіть у середині і наприкінці гри деякі осередки все ж таки доводиться відкривати навмання. Якщо під сусідніми осередками теж немає мін, то відкривається деяка «не замінована» область до осередків, де є цифри. "Заміновані" комірки гравець може помітити, щоб випадково не відкрити їх. Відкривши всі «не заміновані» осередки, гравець виграє.

Python та Kivy

Гра розроблена на мові Python за допомогою графічної бібліотеці kivy. Я обрав Python для розробки тому, що це доволі проста мова. Kivy – крос-платформна open-source бібліотека яка дозволяє швидко будувати графічни додатки і так само швидко упакувати її в бінарні файлі. Також вважаю плюсом що додаток на kive можна запакувати як для декстопу так і для мобільних телефонів. Kivy також має свою мову для розмітки графічних елементів.

Source Code

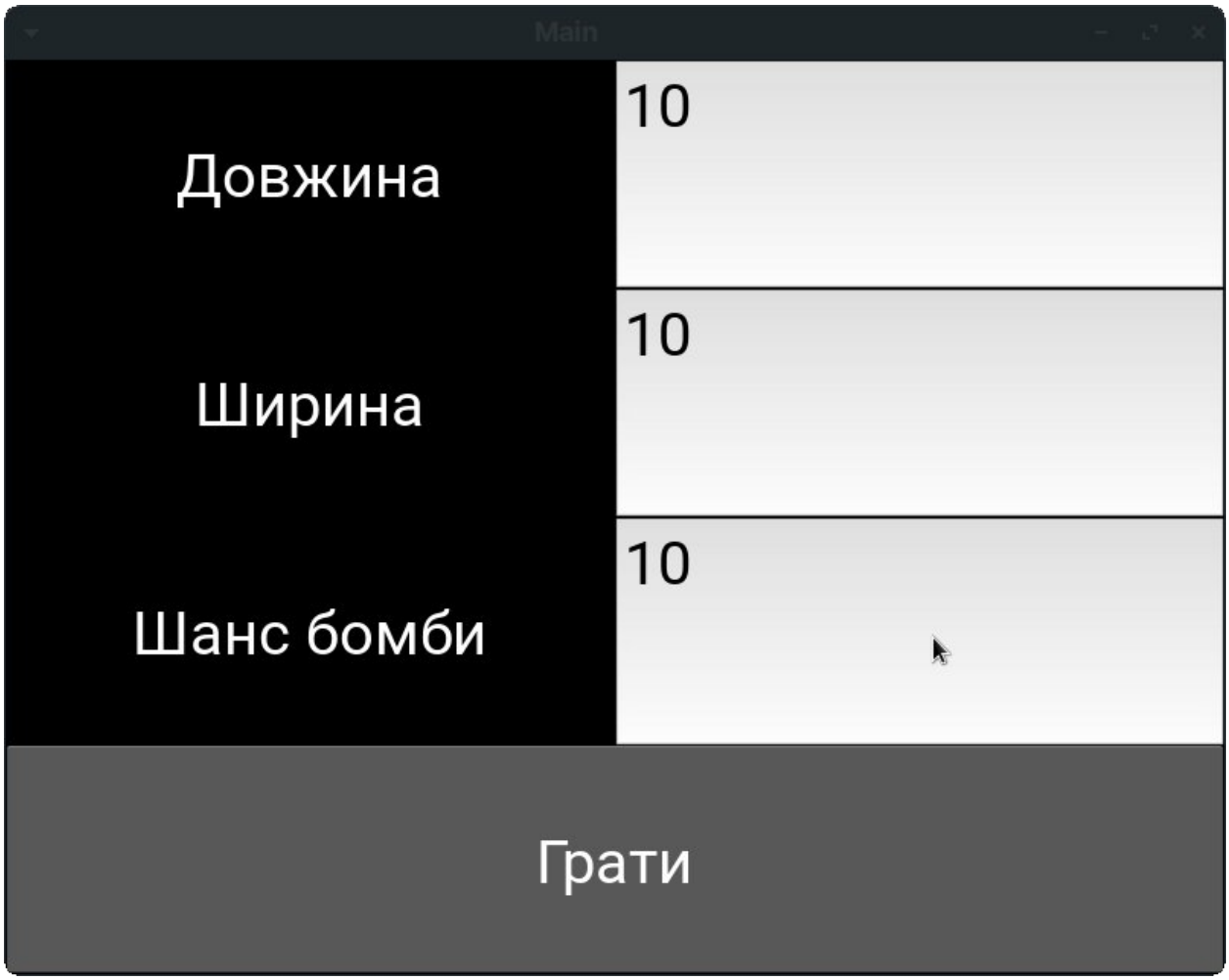
Код останної версії знаходится за посиланням в GitHub (https://github.com/JeffTheK/Saper)

Як запустити або побудувати гру в себе

Перед запуском перевірте що у вас встановлено kivy. Щоб запустити гру запустіть файл src/main.py або введіть команду make run якщо у вас є GNU Make.

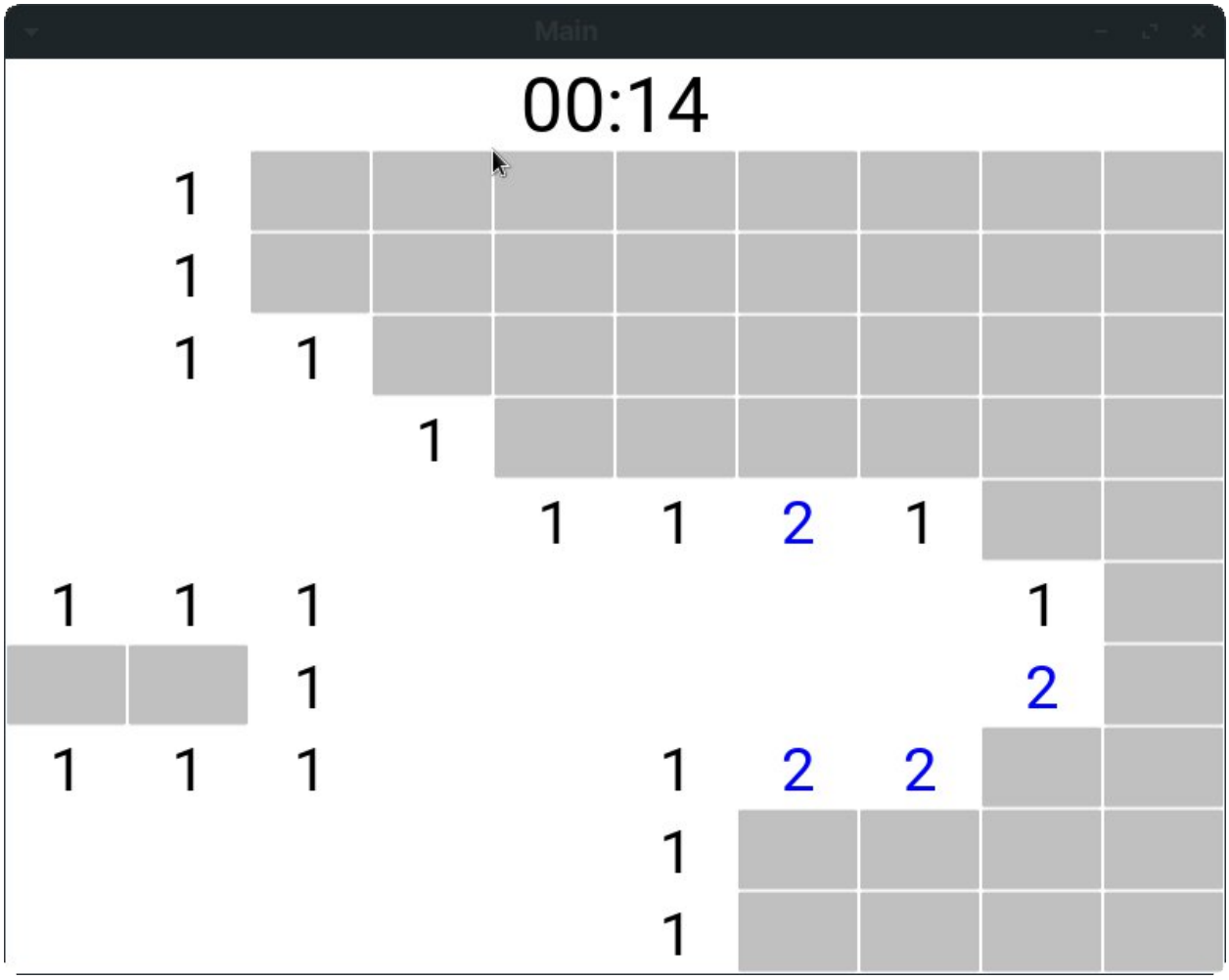
Ця інформація також присутня в README репозиторію.

Головний Екран



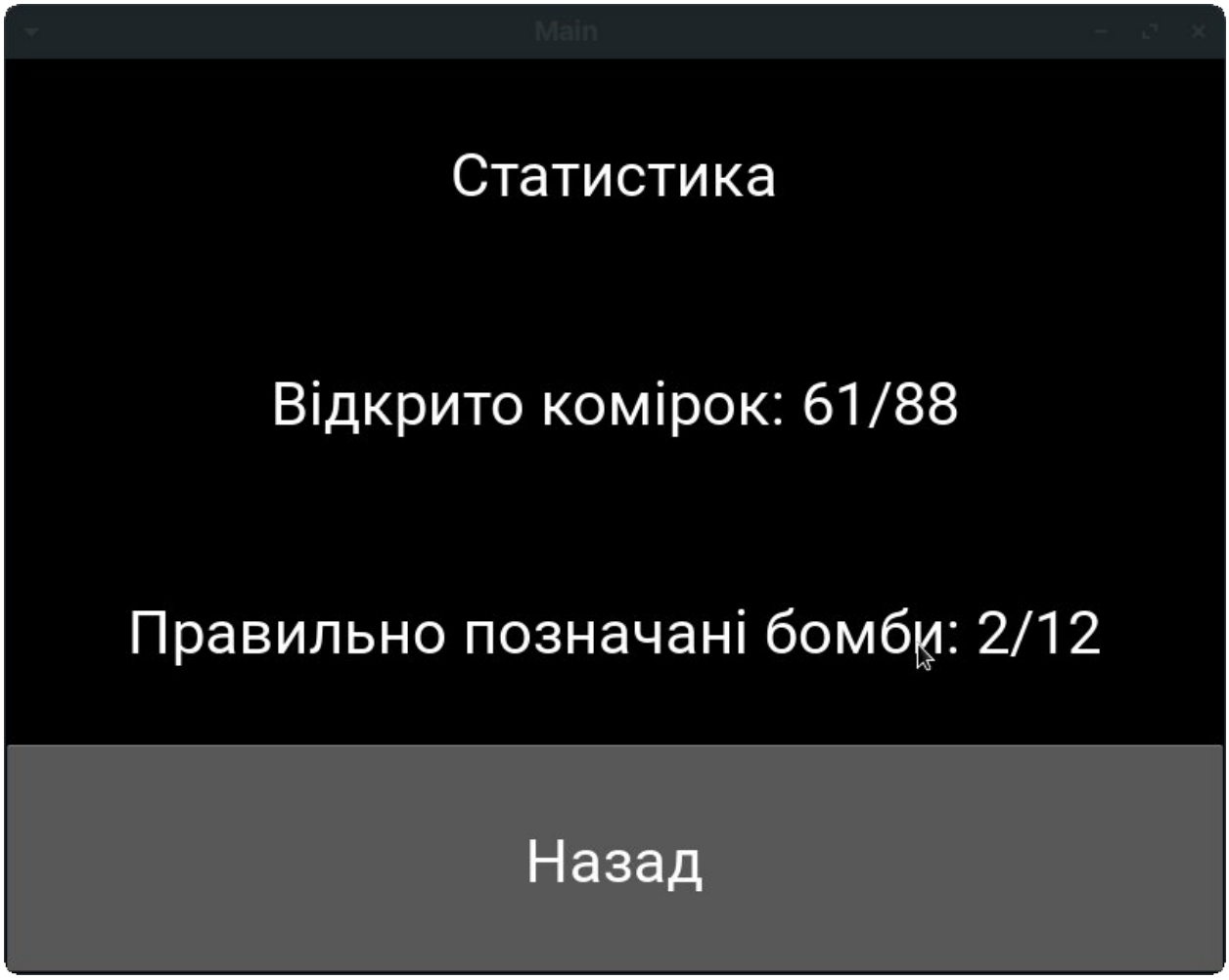
Перше що ви побачите при запуску, це екран де можна налаштувати параметри поля. Натисніть “Грати” для початку гри

Екран Поля



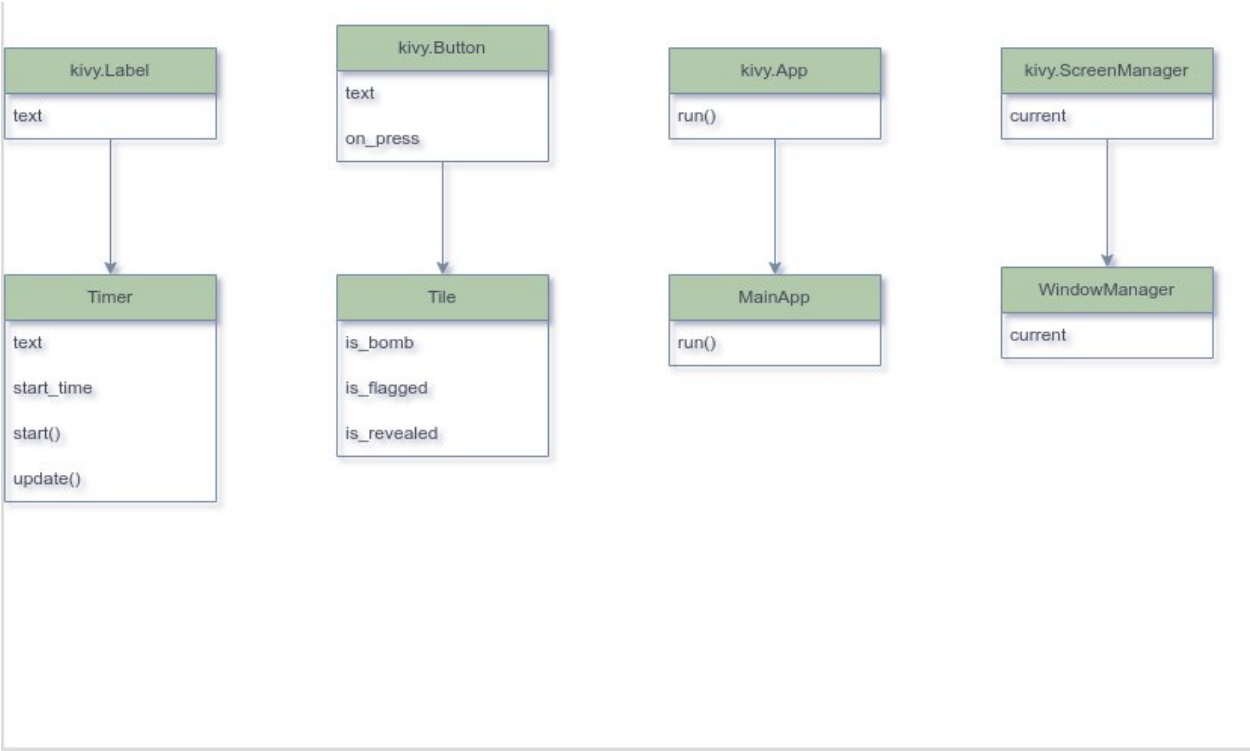
Далі ви переходите до самої гри. Натиснувши лівою кнопкою можете відкрити комірку. Натиснувши правою кнопкою ви ставите флаг як мітку на комірці. Якщо ви відкриваєте комірку з бомбою то програєте та переходите до останнього екрану.

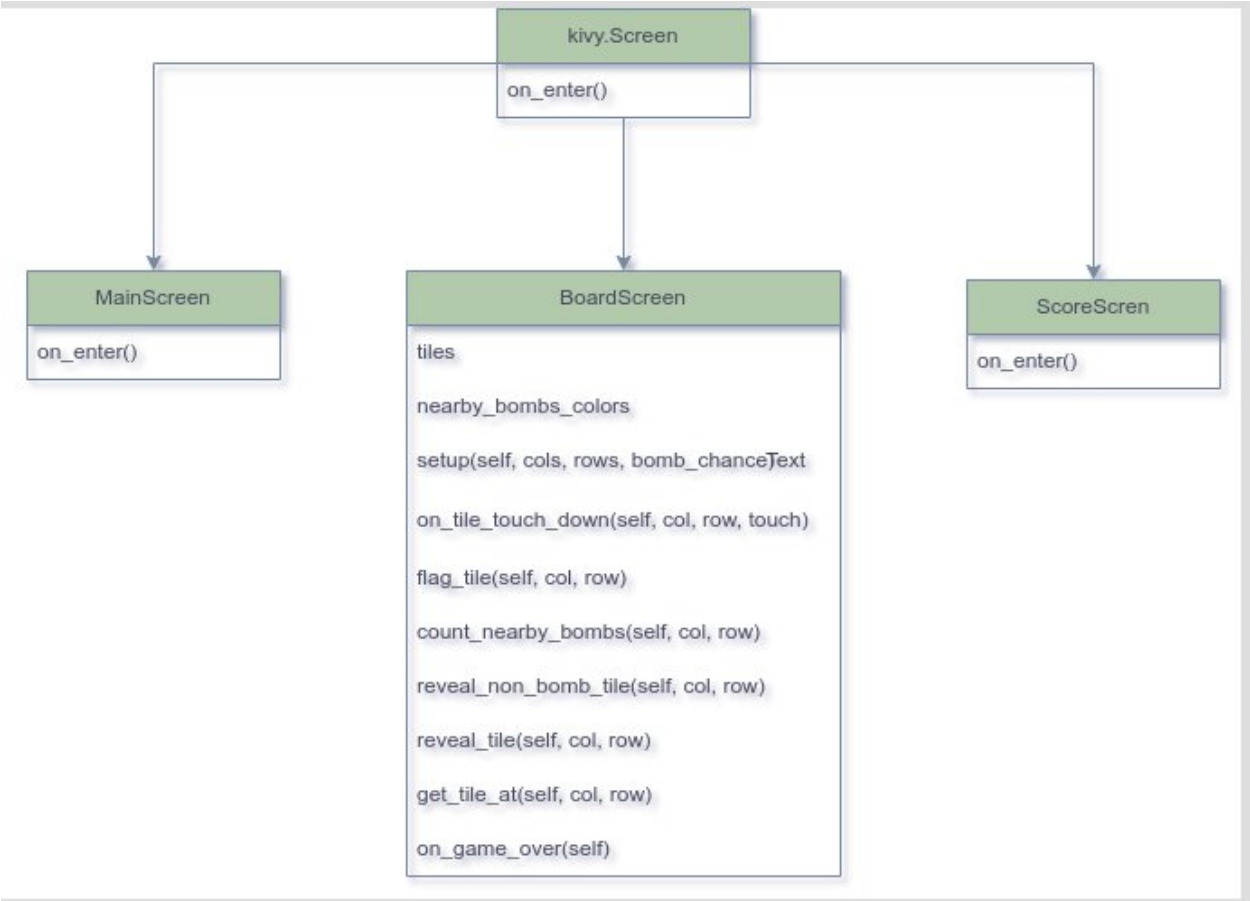
Екран Статистики



Останній екран де вам виводится статистика після гри. Натиснувши назад можна повернутися до першого екрану.

Діаграма Класів





Опис програмного коду

(https://github.com/JeffTheK/Saper)

В main.kv на екрані MainScreen кнопку Грати зв’язано до переходу в BoardScreen і запуску BoardScreen.setup()

```
Button:
    text: "Грати"
    font_size: global_font_size1
    on_press: app.root.current = "board"; app.root.get_screen("board").setup(int(height_input.text), int(width_input.text),
```

Метод setup() спочатку видаляє попередню сітку комірок якщо вона є. Потім починає створювати нову сітку. Залежності від шансу бомби, комірка може бути створена з бомбою. Також при створенні комірки, так як клас Tile походить від Button, то ми зв’язуємо його метод on_touch_down() який викликається при натисканні, до методу MainScreen.on_tile_touch_down(pos, touch)

```
def setup(self, cols, rows, bomb_chance):
    if self.tiles != None:
        for tile in self.tiles.values():
            self.ids.layout.remove_widget(tile)
        self.tiles = {}
    self.ids.layout.cols = cols
    self.ids.layout.rows = rows
    self.ids.timer.start()
    self.bomb_chance = bomb_chance
    bombs_count = 0
    for row in range(rows):
        for col in range(cols):
            is_bomb = random.randrange(0, 100) <= self.bomb_chance
            if is_bomb:
                bombs_count += 1
            tile = Tile(is_bomb)
            tile.bind(on_touch_down=lambda _, touch, pos=(col, row): self.on_tile_touch_down(pos, touch))
            #tile.text = str(col) + " " + str(row)
            self.tiles[(col, row)] = (tile)
            self.ids.layout.add_widget(tile)
    self.score = Score(cols * rows, bombs_count)
    self.score.flagged_tiles = 0
    self.update_bombs_left_label()
```

Метод on_tile_touch_down(self, pos, touch) Перевіряє яка з кнопок миші була натиснута. І викликає відповідний метод.

```
def on_tile_touch_down(self, pos, touch):
    tile = self.get_tile_at(pos)
    if not tile.collide_point(*touch.pos):
        return

    if touch.button == "left":
        self.reveal_tile(pos)
    elif touch.button == "right":
        self.flag_tile(pos)
```

Метод flag_tile(pos) ставить/видаляє флаг на комірці.

```
def flag_tile(self, pos):
    tile = self.get_tile_at(pos)

    if tile.is_revealed:
        return

    if tile.is_flagged:
        tile.is_flagged = False
        tile.remove_widget(tile.icon)
        self.score.flagged_tiles -= 1
        if tile.is_bomb:
            self.score.correctly_guessed_bombs -= 1
    else:
        tile.is_flagged = True
        icon = Image(source="icons/flag.png", size=(tile.width / 1.5, tile.height / 1.5))
        icon.pos = (tile.x + tile.width / 2 - icon.width / 2, tile.y + tile.height / 2 - icon.height / 2)
        tile.icon = icon
        tile.add_widget(icon)
        self.score.flagged_tiles += 1
        if tile.is_bomb:
            self.score.correctly_guessed_bombs += 1
```

Метод reveal_tile(pos) викликається коли гравець хоче відкрити комірку. Якщо це бомба то викликаємо метод on_game_over(). Якщо ні то викликаємо reveal_non_bomb_tile(pos)

```
def reveal_tile(self, pos):
    tile = self.get_tile_at(pos)
    if tile.is_flagged:
        return
    if tile.is_bomb:
        tile.background_color = (1, 0, 0, 1)
        App.get_running_app().root.get_screen("board").on_game_over()
        tile.is_revealed = True
    else:
        self.reveal_non_bomb_tile(pos)
```

Метод on_game_over() переходить до екрану статистики

```
# Викликається при поразці
def on_game_over(self):
    App.get_running_app().root.get_screen("score").score = self.score
    App.get_running_app().root.current = "score"
```

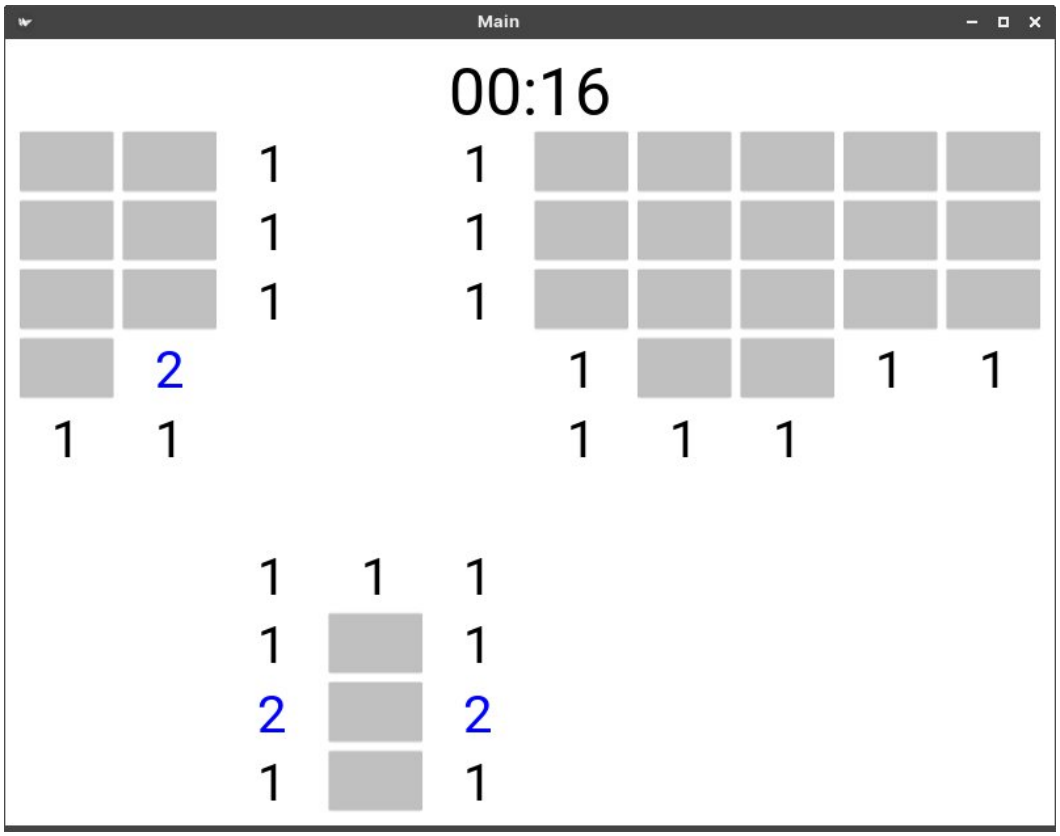
Метод reveal_non_bomb_tile(self, pos) Викликається якщо натиснута комірка не має бомб. Рекурсивно викликається на сусідах. Тобто якщо ви відкрили комірку без бомб то сусідні комірки без бомб теж відкриваються

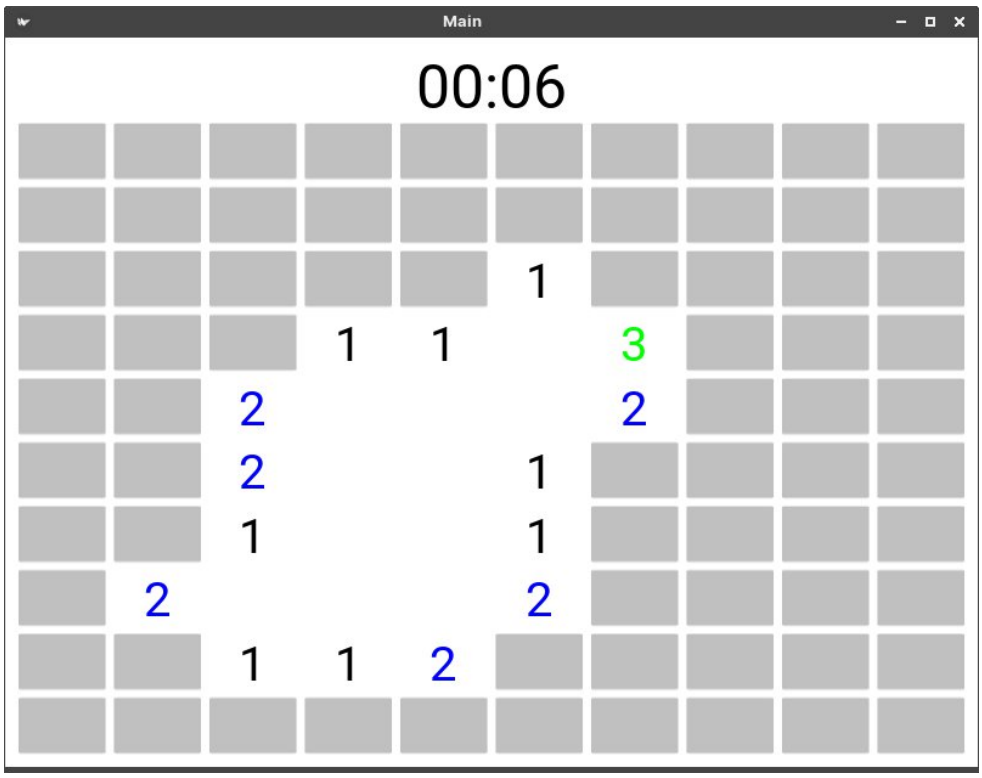
```
def reveal_non_bomb_tile(self, pos):
    col = pos[0]
    row = pos[1]
    positions = [
        (col + 1, row),
        (col - 1, row),
        (col, row + 1),
        (col, row - 1),
    ]

    tile = self.get_tile_at(pos)
    if tile.is_bomb or tile.is_revealed:
        return

    tile.is_revealed = True
    tile.background_color = (0, 0, 0, 0)
    nearby_bombs_count = self.count_nearby_bombs(pos)
    if (nearby_bombs_count > 0):
        tile.text = str(self.count_nearby_bombs(pos))
        tile.color = self.nearby_bombs_colors[nearby_bombs_count]
    else:
        for pos_ in positions:
            if self.get_tile_at(pos_) != None:
                self.reveal_non_bomb_tile(pos_)
    self.score.cleared_tiles += 1
    self.check_for_win()
```

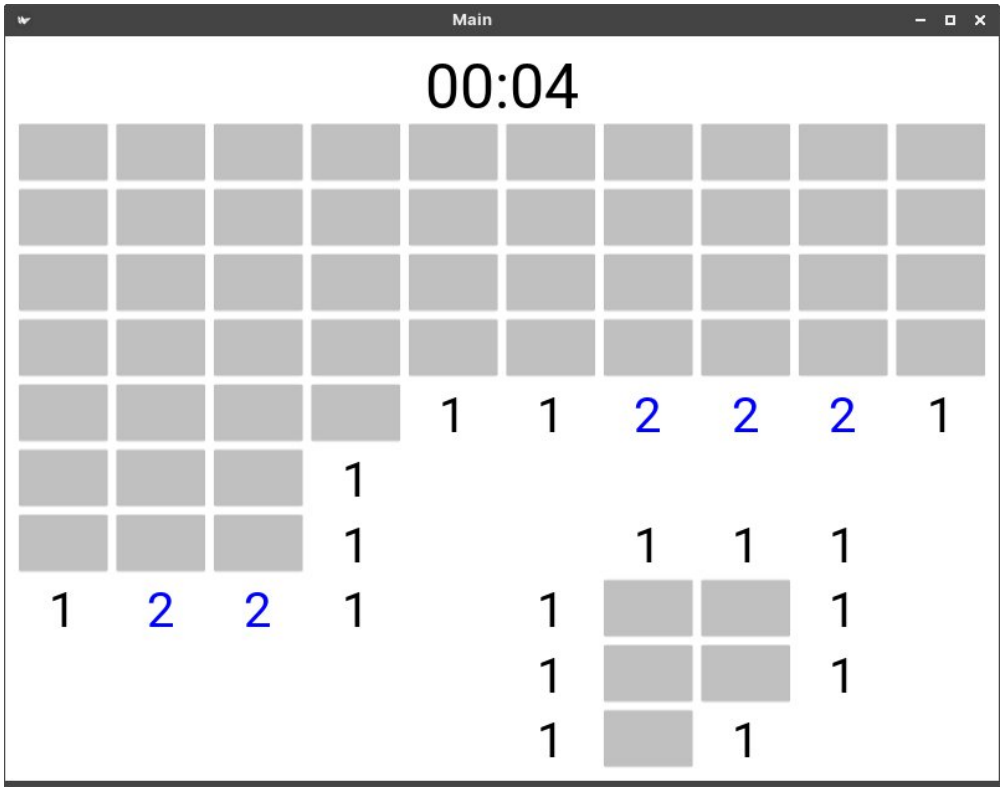
Метод count_nearby_bombs(pos) Повертає кількість сусідніх комірок з бомбами

```
def count_nearby_bombs(self, pos) -> int:
    col = pos[0]
    row = pos[1]
    print("kek" + str(col) + " " + str(row))
    count = 0
    positions = [
        (col + 1, row),
        (col - 1, row),
        (col, row + 1),
        (col, row - 1),
        (col + 1, row + 1),
        (col - 1, row - 1),
        (col + 1, row - 1),
        (col - 1, row + 1)]
    for pos_ in positions:
        print(pos_)
        if self.get_tile_at(pos_) != None and self.get_tile_at(pos_).is_bomb:
            count += 1

    return count
```

Скріншоти




Main
00:06


Main
00:04