

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM
KHOA ĐIỆN-ĐIỆN ĐIỆN TỬ



HCMUTE

TIỂU LUẬN GIỮA KỲ

LẬP TRÌNH VỚI PYTHON TỰA
GAME MINESWEEPER

MÃ MÔN HỌC: PRPL218164_23_2_09

NHÓM THỰC HIỆN: Nhóm

GV: ThS. Nguyễn Duy Thảo

Tp. Hồ Chí Minh, tháng 5 năm 2024

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM
KHOA ĐIỆN-ĐIỆN ĐIỆN TỬ



HCMUTE

TIỂU LUẬN GIỮA KỲ

LẬP TRÌNH VỚI PYTHON TỰA

GAME MINESWEEPER

MÃ MÔN HỌC: PRPL218164_23_2_09

NHÓM THỰC HIỆN: Nhóm

GV: ThS. Nguyễn Duy Thảo

Tp. Hồ Chí Minh, tháng 5 năm 2024

DANH SÁCH NHÓM THAM GIA VIẾT TIỂU LUẬN

HỌC KỲ II, NĂM HỌC 2023-2024

Nhóm (Lớp thứ 2, tiết 10-12)

Tên đề tài: Lập trình với Python tựa game Minesweeper

ST T	HỌ VÀ TÊN SINH VIÊN	MÃ SỐ SINH VIÊN	TỶ LỆ % HOÀN THÀNH	Ký tên
1	Hoàng Ngọc Dung	23139006	100%	
2	Hồ Bảo Việt	23139050	100%	
3	Trần Hữu Dương	23139009	100%	
4	Đoàn Minh Duy Bình	23139005	100%	
5	Cao Như Ý	23139052	100%	

Ghi chú:

- Tỷ lệ% = 100%
- Trưởng nhóm: Hoàng Ngọc Dung

Nhận xét của giáo viên:

.....

.....

.....

.....

.....

.....

.....

Ngày tháng năm

Giáo viên chấm điểm

MỤC LỤC

CHƯƠNG 1: TỔNG QUAN	4
1.1. Giới thiệu	4
1.2. Mục đích	4
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT	5
2.1. Tổng quan về PyGame	5
2.2. Các khái niệm cơ bản về giao diện trong PyGame	5
2.3. Thuật toán đệ Quy	6
2.4. Một số thành phần, phương thức sử dụng	7
2.4.1. Class button: tạo nút nhấn cho giao diện	7
2.4.2. Class piece: Các thao tác trên ô của bảng	7
2.4.3. Class solver: Thao tác di chuyển trên bảng	7
2.4.4. Class Board: Tạo lập bảng (sinh bảng mìn và số lân cận)	8
2.4.5. Class game: quản lý gameplay	8
CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ GAME	8
3.1. Cấu trúc quản lý file	8
3.2. Các chức năng của hệ thống	9
3.3. Thuật toán MinesweeperGame	11
CHƯƠNG 4: KẾT QUẢ CHƯƠNG TRÌNH	15
PHỤ LỤC	18

CHƯƠNG 1: TỔNG QUAN

1.1. Giới thiệu

Minesweeper được tạo ra bởi Curt Pollak, một lập trình viên của Microsoft vào năm 1985. Trò chơi Minesweeper là một trò chơi logic nổi tiếng, nhiệm vụ của người chơi phải tìm ra các ô trống mà không có mìn, dựa vào các số hiển thị xung quanh. Tựa game giải đố phổ biến trên thế giới.

1.2. Mục đích

Nhóm thiết kế tựa game Dò Mìn (MineSweeper) với giao diện thân thiện và dựa trên ngôn ngữ Python đã học cùng với thư viện Pygame và PyAutoGUI (một thư viện Python cho phép điều khiển chuột và bàn phím, cũng như các tác vụ tự động). Một số tính năng đã được thiết kế

- Trong lần đầu tiên, người dùng chọn 1 ô bất kì trong bảng. Sau đó tiến hành tạo bảng mìn để đảm bảo người dùng sẽ không thua khi mới bắt đầu chơi.
- Chế độ đặt cờ cho những ô người chơi dự đoán là sẽ có bom.
- Số lượng bom được đặt sẽ $\leq 50\%$ số lượng ô
- Chế độ chơi : Dễ (4x4) , Trung bình (9x9), Khó (19x19)

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1. Tổng quan về PyGame

Pygame là một thư viện mã nguồn mở được viết bằng Python dành cho việc phát triển trò chơi điện tử. Nó cung cấp các công cụ để tạo đồ họa, âm thanh, xử lý sự kiện và nhiều tính năng khác cần thiết để tạo ra các trò chơi hoàn chỉnh. Pygame được viết bởi Pete Shinnars thay thế cho chương trình PySDL sau khi quá trình phát triển dự án này bị đình trệ. Chính thức phát hành từ năm 2000, Pygame được phát hành theo phần mềm miễn phí GNU Lesser General Public License. Pygame có thể chạy trên nhiều nền tảng và hệ điều hành khác nhau.

2.2. Các khái niệm cơ bản về giao diện trong PyGame

Reference object (RO) chính trong Pygame(những đối tượng được sử dụng để quản lý các tài nguyên trò chơi):

- **Surface:** Surface là RO được sử dụng để lưu trữ dữ liệu đồ họa.Surface có thể được sử dụng để vẽ hình ảnh, văn bản và các thành phần giao diện khác.Các điểm ảnh (pixel) của Surface object có thể được thay đổi bằng cách gọi các hàm vẽ của Pygame (pygame.draw)và sau đó hiển thị chúng trên màn hình, nhưng không bao gồm viền cửa sổ, thanh tiêu đề và các nút. Sau khi vẽ xong tất cả các thành phần cho một frame, cần sử dụng hàm *pygame.display.flip()* để cập nhật nội dung của Surface object lên màn hình.
- **Image:** Image là RO thường được sử dụng trong Pygame cho hiển thị hình ảnh lên màn hình: tải tệp hình ảnh từ ổ cứng bằng cách sử dụng hàm *pygame.image.load()*

Sự kiện (Event)

<code>pygame.event.get()</code>	Lấy danh sách các sự kiện đã xảy ra
<code>event.type</code>	Loại sự kiện (ví dụ: KEYDOWN, MOUSEBUTTONDOWN).

Âm thanh

<code>pygame.mixer.init()</code> <code>sound.play()</code>	Khởi tạo hệ thống âm thanh. Phát âm thanh
<code>pygame.mixer.Sound("filename")</code>	Tải tệp âm thanh và tạo Sound object.

Phương thức tạo và quản lý các thành phần giao diện người dùng

Phương thức	Ý nghĩa
<code>pygame.display.set_mode(width, height)</code>	Tạo ra một Surface màn hình với kích thước được chỉ định.
<code>pygame.Surface.fill(color)</code>	Tô màu cho Surface với màu được chỉ định.
<code>pygame.font.Font.render(text, antialias, color)</code>	Tạo ra một Image từ văn bản với phong chữ và màu được chỉ định.
<code>pygame.display.flip()</code>	Cập nhật màn hình trò chơi.
<code>pygame.init()</code>	Khởi tạo Pygame trước khi sử dụng các chức năng của nó.
<code>pygame.quit()</code>	Thoát Pygame và dọn dẹp các tài nguyên.

Bề mặt hiển thị (Display Surface)

<code>screen = pygame.display.get_surface()</code>	Lấy đối tượng Surface đại diện cho màn hình hiển thị.
<code>screen.fill(color)</code>	Tô màu toàn bộ màn hình hiển thị.

Hình ảnh (Image object)

<code>pygame.image.load("filename")</code>	Tải tệp hình ảnh và tạo Image object.
<code>image.get_width():</code>	Lấy chiều rộng của Image object.
<code>image.get_height()</code>	Lấy chiều cao của Image object.
<code>image.convert()</code>	Chuyển đổi định dạng màu của Image object.

2.3. Thuật toán đệ Quy

Thuật toán đệ quy là một kỹ thuật lập trình trong đó hàm tự gọi chính nó. Thuật toán này có thể được sử dụng để giải quyết các bài toán phức tạp một cách hiệu quả bằng cách chia nhỏ bài toán thành các phần nhỏ hơn và lặp lại cho đến khi đạt được điều kiện cơ bản. Giải các câu đố hoặc quản lý trạng thái trò chơi thường đòi hỏi tư duy đệ quy

và trong Pygame, thuật toán đệ quy có thể được sử dụng cho nhiều mục đích khác nhau, bao gồm:

- Vẽ các hình dạng phức tạp: thuật toán đệ quy để vẽ cây Fractal, hoa tuyết Koch, ...
- Tạo các hiệu ứng đồ họa: thuật toán đệ quy để tạo hiệu ứng nổ, hiệu ứng tia sáng, ...
- Di chuyển các đối tượng trong game: thuật toán đệ quy để di chuyển các nhân vật theo đường cong, di chuyển các viên đạn theo quỹ đạo,..

2.4. Một số thành phần, phương thức sử dụng

Self trong Python là đại diện cho thể hiện của lớp(class). Bằng cách sử dụng từ khóa “self”, chúng ta có thể truy cập các thuộc tính và phương thức của lớp trong python. Vì vậy nó liên kết các thuộc tính với các đối số đã cho. Class sử dụng: **game , board , button , piece , solver.**

2.4.1. Class button: tạo nút nhấn cho giao diện

- Thuộc tính: image, x_pos, y_pos, font, base_color, hovering_color, text, rect, text_rect
- Phương thức: cập nhật xử lý và thay đổi màu khi user chọn button
 - Update(): tham số (self , screen)
 - Checkforinput(): tham số (self, position)
 - Changecolor(): tham số (self , position)

2.4.2. Class piece: Các thao tác trên ô của bảng

- Thuộc tính: hasbom, arround, clicked, flagged, neighbors
- Phương thức: Cập nhật danh sách các ô lân cận và thiết lập số lượng mìn xung quanh, Gắn cờ, Bom có hay không.
 - getNumAround(): tham số (self), getClicked(): tham số (self)
 - getFlagged(): tham số (self), getNeighbors(): tham số (self)
 - toggleFlag(): tham số (self), handleClick(): tham số (self)
 - setNumAround(): tham số (self),setNeighbors(): tham số (self, neighbors)

2.4.3. Class solver: Thao tác di chuyển trên bảng

- Thuộc tính: board
- Phương thức: Move(), openUnflagged(), flagAll()

- Move(): tham số (self)
- openUnflagged(): tham số (self, neighbors)
- flagAll() : tham số (self, neighbors)

2.4.4. Class Board: Tạo lập bảng (sinh bảng mìn và số lân cận)

- Thuộc tính: size, board, won, lost, prob
- Phương thức:
 - Update_board(): tham số (self , board , index)
 - Getboard(): tham số (self), Getsize(): tham số (self)
 - getPiece(): tham số (self , index)
 - handleClick(): tham số (self , piece , index)
 - checkwon (): tham số (self), getwon(): tham số (self)
 - getLost(): tham số (self)
 - setNeighbors(): tham số (self)
 - addToNeighborList(): tham số (self, neighbors , row , col)
 - setNumAround(): tham số (self)

2.4.5. Class game: quản lý gameplay

- Thuộc tính: sizeScreen, screen, piecesize, loadPictures, solver
- Phương thức
 - loadPictures(): tham số (self)
 - run(): tham số (self), draw(): tham số (self)
 - getImageString(): tham số (self, piece)
 - handleClick(): tham số (self , position , flag , k)
 - win(): tham số (self), lose() : tham số (self)

CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ GAME

3.1. Cấu trúc quản lý file

```

├── main.py # file chính build game
├── Scripts
│   ├── board.py # tạo bảng mìn
│   ├── game.py # quản lý gameplay
│   ├── piece.py # xử lý thao tác từng phần tử của bảng
│   ├── Solver.py # chuyển động di chuyển chuột của user
│   └── States.py # state win, lose, option, menu

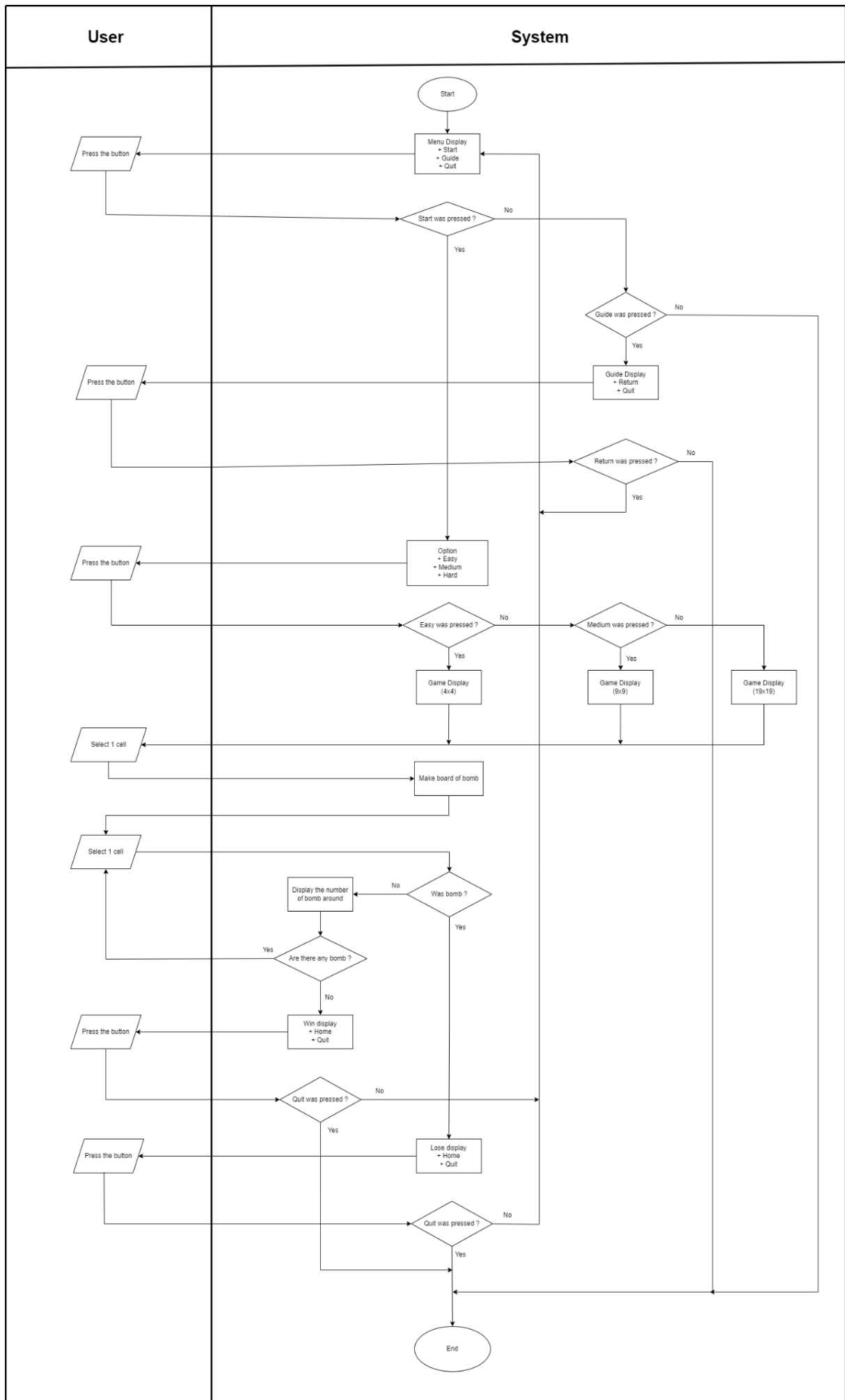
```

└─ bnt.py # UI for create a button
└─ music
└─ images

3.2. Các chức năng của hệ thống

Tổng quan chức năng	Mô tả
MainMenu	Giao diện bắt đầu với 3 nút: Start, Guide, Quit
Option	Tạo ra menu options của trò chơi gồm 3 nút : easy , medium , hard
Guide	Hướng dẫn người chơi
Gameplay	Bảng thử thách để user chơi game
Win	Hiển thị người chơi chiến thắng
Lose	Hiển thị người chơi thua cuộc
Chức năng GamePlay	Mô tả
Virtual_Board	Tạo bảng rỗng không có bom
Update_Board	Tạo bảng có bom từ vị trí mà user chọn Số lượng bom \leq 50% số lượng ô
Flagged	Chế độ đặt cờ cho những ô người chơi dự đoán là sẽ có bom

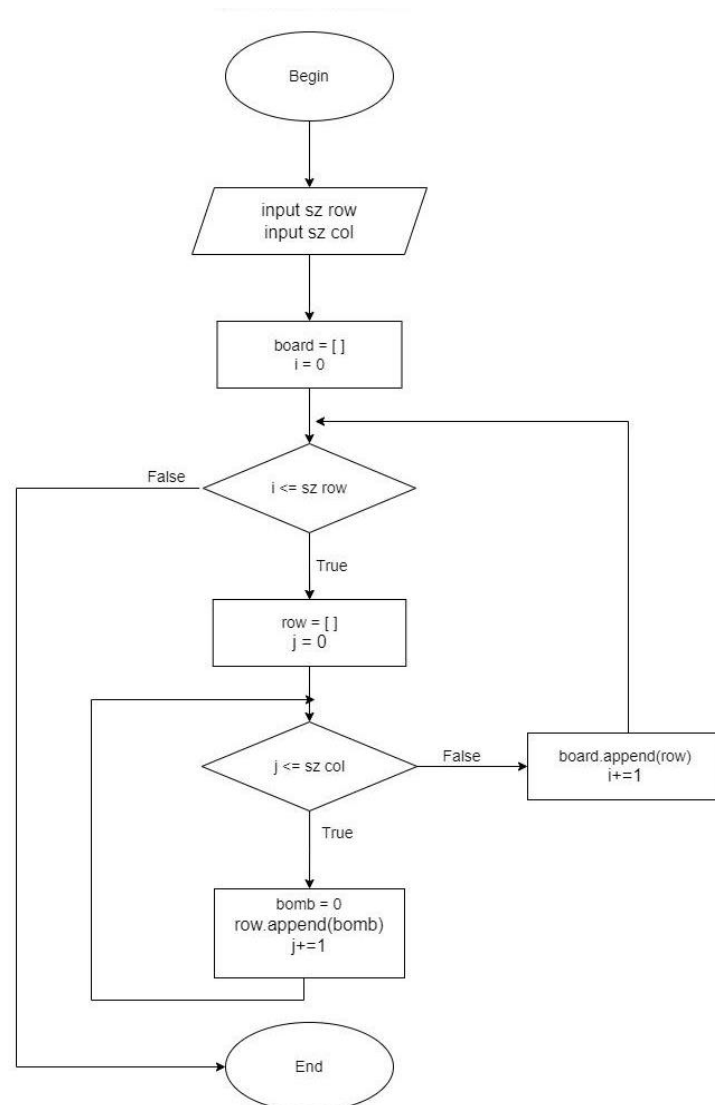
Lưu đồ tổng quan hoạt động



3.3. Thuật toán MinesweeperGame

- Tạo bảng rỗng khi bắt đầu (sinh mìn)

```
def __init__(self, size, prob): # tạo bảng rỗng ban đầu
    self.size = size
    self.board = []
    self.won = False
    self.lost = False
    self.prob = prob
    for row in range(size[0]):
        row = []
        for col in range(size[1]):
            bomb = 0
            piece = Piece(bomb)
            row.append(piece)
        self.board.append(row)
```

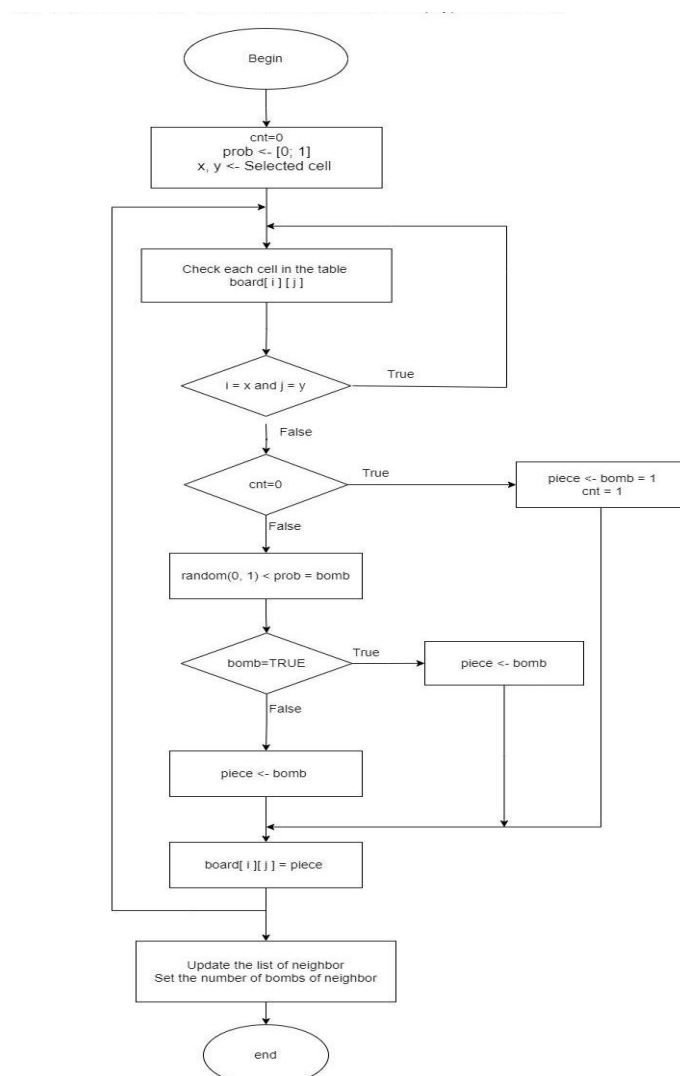


- Sau khi user nhấn vào 1 ô bất kì trong bảng , tiến hành update bảng với số lượng bom và số lân cận (True : ô có mìn, False : ô không có mìn)

```

cnt = 0, sz = len(self.board)
for i in range(sz):
    for j in range(sz):
        if (i == index[0] and j == index[1]): # skip first click
            continue
        else:
            if cnt <= 1 :
                bomb = 1 # bảng luôn có ít nhất 1 bom
                cnt += 1
            else: bomb = random.random() < self.prob #xác suất
50% so với các bảng
            piece = Piece(bomb)
            self.board[i][j] = piece
self.setNeighbors()
self.setNumAround()

```



- Thêm danh sách 8 ô lân cận gần kề của ô (row,col) đang được xét

```
def addToNeighborsList(self, neighbors, row, col):  
    for r in range(row - 1, row + 2): # Duyệt 8 ô xung quanh  
        for c in range(col - 1, col + 2):  
            if r == row and c == col: # trừ vị trí hiện tại  
                continue  
            if r < 0 or r >= self.size[0] or c < 0 or c >=  
self.size[1]: # kích thước size[0] hàng , size[1] cột  
                continue  
            neighbors.append(self.board[r][c]) # add to list
```

- Với mỗi ô có các danh sách hàng xóm gần kề cập nhật các số

```
def setNumAround(self):  
    num = 0  
    for neighbor in self.neighbors:  
        if neighbor.getHasBomb():  
            num += 1  
    self.around = num
```

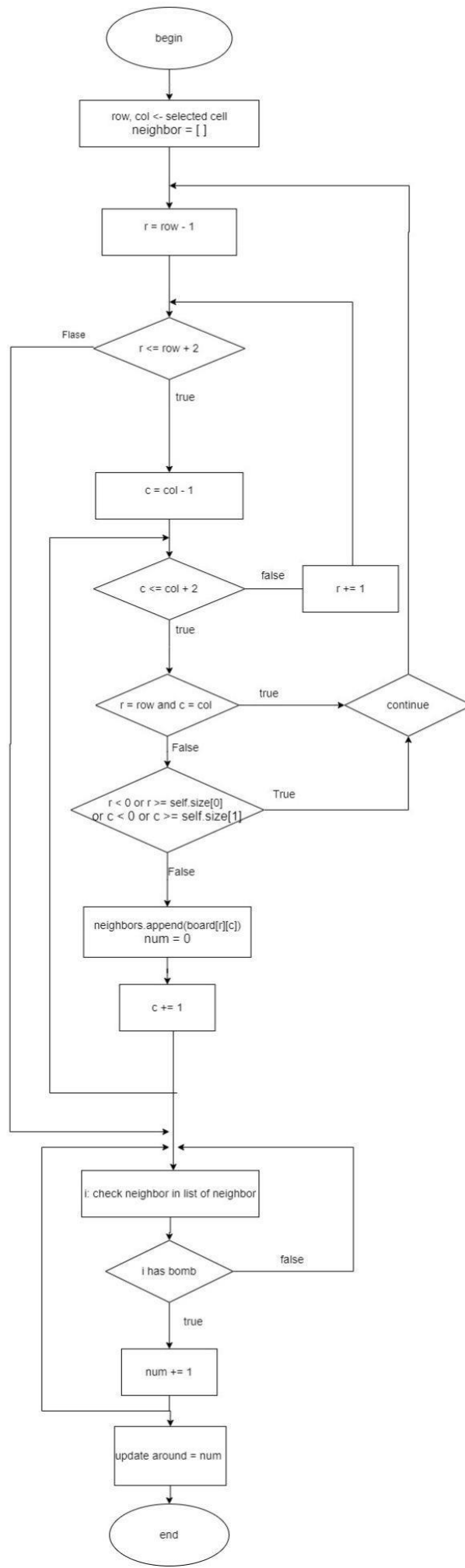
- Khi user playgame : nếu click vào ô không có mìn xung quanh

```
if piece.getNumAround() == 0: # loang rộng nếu ô click không có bom  
    for neighbor in piece.getNeighbors():#xung quanh ko có bom  
        self.handleClick(neighbor, False)  
if piece.getHasBomb():# gặp bom -> thua  
    self.lost = True  
else:  
    self.won = self.checkWon()
```

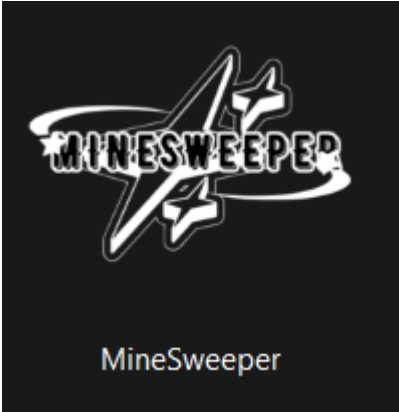
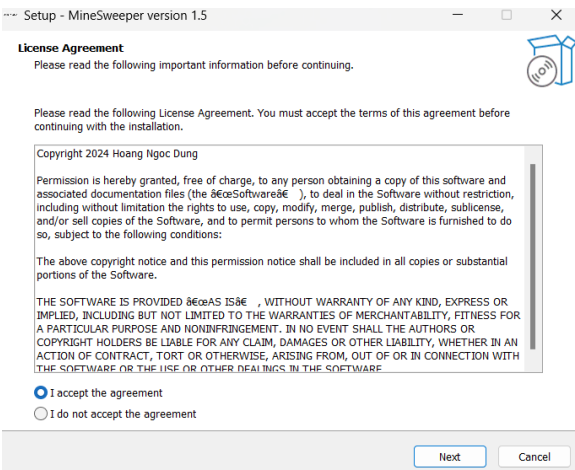

- Kiểm tra thắng :


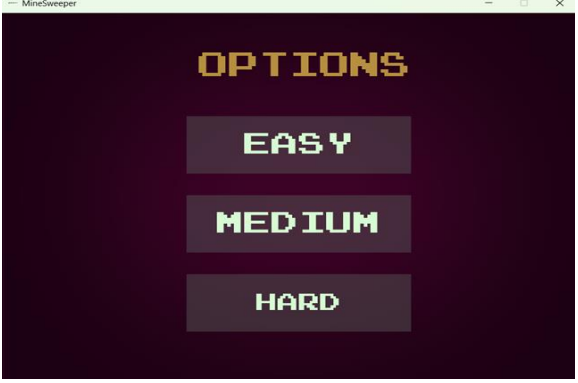

```
def checkWon(self):  
    for row in self.board:  
        for piece in row:  
            if not piece.getHasBomb() and not piece.getClicked():  
# nếu không có bom và chưa được chọn thì chưa thắng  
                return False  
    return True
```

THIẾT LẬP SỐ LƯỢNG BOM CHO CÁC Ô LÂN CẬN XUNG QUANH

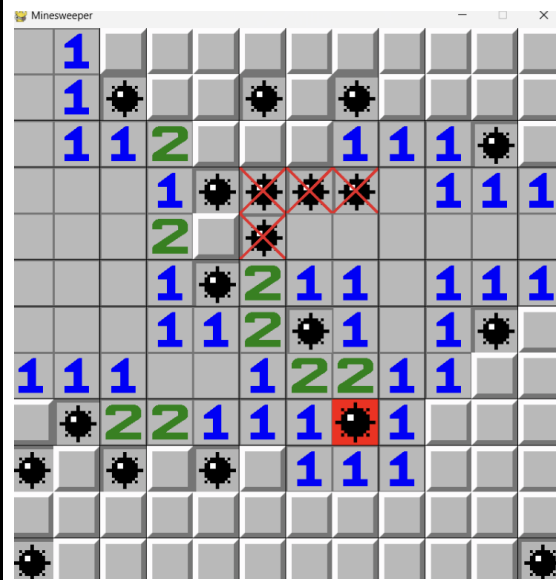


CHƯƠNG 4: KẾT QUẢ CHƯƠNG TRÌNH

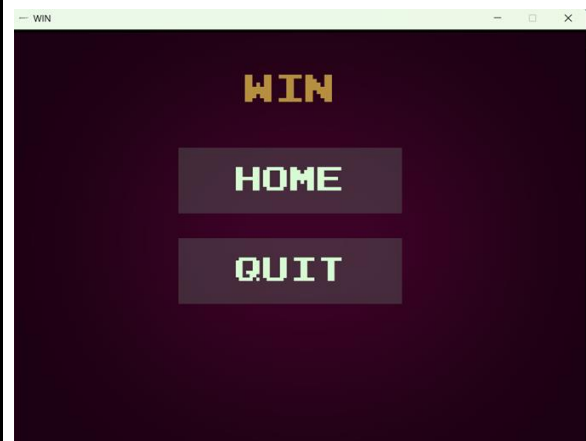
Application	
Releases	
Loading State	

<p>MainMenu</p>	 A screenshot of a game window titled "WIN". The background is dark purple. At the top, the word "MINESWEEPER" is written in a yellow, pixelated font. Below it, three dark grey buttons with white text are stacked vertically: "START", "GUIDE", and "QUIT".
<p>Options</p>	 A screenshot of a game window titled "MineSweeper". The background is dark purple. At the top, the word "OPTIONS" is written in a yellow, pixelated font. Below it, three dark grey buttons with white text are stacked vertically: "EASY", "MEDIUM", and "HARD".
<p>Guide</p>	 A screenshot of a game window titled "MINESWEEPER". The background is dark purple. At the top, the word "MINESWEEPER" is written in a yellow, pixelated font. Below it, a light grey rectangular box contains the following text: "Minesweeper is also known by another name called Mine Detection or Bomb Removal. Coming to Minesweeper the player's task is to open all the squares without clicking on the squares containing mines, and if you click on a square containing a mine, you will lose immediately. You can right-click to place a flag." At the bottom of the screen, there are two dark grey buttons with white text: "QUIT" on the left and "RETURN" on the right.

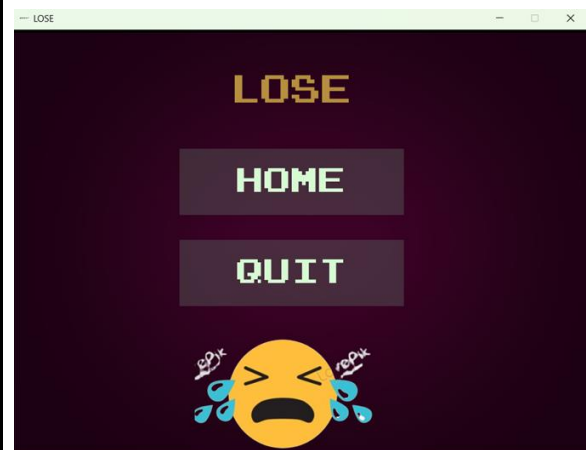
GamePlay



Win



Lose



PHỤ LỤC

Source Code: [giunzz/MineSweeper Python: pygame \(github.com\)](https://github.com/giunzz/MineSweeper)

Application: [MINESWEEPER GAME - Google Drive](#)