TRƯỜNG ĐẠI HỌC SÀI GÒN
KHOA CÔNG NGHỆ THÔNG TIN

PHÁT TRIỂN PHẦN MỀM MÃ NGUỒN MỞ

Xây Dựng Game Hex Bằng Python

GVHD: Từ Lãng Phiêu
SV: Lâm Tuấn Long - 3121560047
Nguyễn Hào Phong - 3121560070

TP. HỒ CHÍ MINH, THÁNG 5/2024

Mục lục

LỜI MỞ ĐẦU

Hiện nay, ngành Công nghệ thông tin đang phát triển nhanh chóng và ứng dụng của nó lan rộng trong mọi lĩnh vực cuộc sống, cả trên toàn cầu và ở Việt Nam. Công nghệ thông tin là một phần không thể thiếu, đóng vai trò quan trọng trong công cuộc công nghiệp hóa, hiện đại hóa và phát triển đất nước. Việc ứng dụng khoa học và công nghệ vào đời sống và công tác là vô cùng thiết yếu. Sự kết hợp giữa công nghệ thông tin và truyền thông là một yếu tố quan trọng trong hoạt động của các công ty và tổ chức. Công nghệ thông tin và truyền thông góp phần thay đổi suy nghĩ và lối tư duy của con người, giúp con người trở nên năng động và kết nối nhanh chóng ở bất kỳ đâu, từ đó tăng cường hiệu quả và năng suất làm việc.

Python là một ngôn ngữ lập trình hướng đối tượng cao cấp, được sử dụng để phát triển website và các ứng dụng đa dạng. Python được tạo ra bởi Guido van Rossum và phát triển trong dự án mã nguồn mở. Với cú pháp đơn giản và dễ hiểu, là lựa chọn hoàn hảo cho người mới học lập trình. Python là 1 ngôn ngữ phổ biến trong việc phát triển các phần mềm mã nguồn mở miễn phí, dễ sử dụng và linh hoạt . Pygame là thư viện mã nguồn mở trên ngôn ngữ Python dùng để lập trình video games, chứa đầy đủ các công cụ hỗ trợ lập trình game như đồ hoạt, hoạt hình, âm thanh, và sự kiện điều khiển kết hợp cùng thư viện Socket có thể tạo nên các game multiplayer đơn giản và dễ hiểu

Trong quá trình tìm hiểu, chúng tôi rất quan tâm đến các ứng dụng game được phát triển và lập trình bằng Python sử dụng thư viện Pygame. Pygame là một bộ công cụ tiện ích trong ngôn ngữ lập trình Python và đã tạo ra nhiều trò chơi huyền thoại từ thời ban đầu. Do đó, chúng tôi đã quyết định sử dụng thư viện Pygame của Python, kết hợp cùng thư viện socket để xây dựng trò chơi Hex (board game) chơi được trên 2 máy, nhằm thấy rõ sức mạnh của nó.

LỜI CẢM ƠN

Chúng em xin chân thành cảm ơn Khoa Công nghệ thông tin, trường Đại học Sài Gòn đã tạo điều kiện cho chúng em được thực hiện đồ án môn học "Phát triển phần mềm mã nguồn mở". Chúng em cũng xin chân thành cảm ơn thầy Từ Lãng Phiêu đã giảng dạy cho chúng em những kiến thức cần thiết cho môn học này. Những kiến thức đó đã giúp cho chúng em rất nhiều trong quá trình làm đồ án báo cáo môn học.

Chúng em xin chân thành cảm ơn quý Thầy cô trong Khoa đã tận tình giảng dạy và trang bị cho chúng em những kiến thức cần thiết trong thời gian qua. Mặc dù nhóm đã cố gắng hoàn thành đồ án môn học với tất cả nỗ lực của từng thành viên trong nhóm, nhưng đồ án chắc chắn không tránh khỏi những thiếu sót nhất định, rất mong nhận được sự cảm thông, chia sẻ và tận tình đóng góp chỉ bảo của quý Thầy Cô.

1 TỔNG QUÁT

1.1 Lý do chọn đề tài

-Hex game là một trò chơi cổ điển.
-Luật chơi rất đơn giản nhưng mà có chiều sâu về tính chiến thuật.
-Trò chơi dễ dàng tái tạo lại bằng ngôn ngữ python dựa trên thư viện pygame.
-Vận dụng được tính chất lập trình hướng đối tượng cùng lập trình socket.

1.2 Mục tiêu và phạm vi đề tài

1.2.1 Mục tiêu

- Mục đích:
+ Nắm chắc được được kỹ năng và kiến thức về lập trình.
+ Tìm hiểu về thư viện Pygame và socket trong ngôn ngữ lập trình Python.
+ Củng cố, áp dụng, nâng cao kiến thức đã được học.
+ Nắm bắt được quy trình làm game cơ bản.
- Mục tiêu:
+ Vận dụng được tính chất của lập trình hướng đối tượng.
+ Sử dụng thư viện Pygame vào việc xây dựng game. + Sử dụng thư viện socket kết nối 2 máy với nhau để chơi game

1.2.2 Phạm vi

- Xây dựng game Hex (board game) trong python sử dụng thư viện pygame và kết nối 2 người chơi bằng thư viện socket.

2 GIỚI THIỆU VÀ HƯỚNG DẪN CÀI ĐẶT

2.1 Giới thiệu

2.1.1 Giới thiệu Python

Python là một ngôn ngữ lập trình mã nguồn mở, hướng đối tượng cấp cao,mạnh mẽ và thông dịch. Do Guido van Rossum tạo ra và lần đầu ra mắt vào năm 1991.

Ban đầu, Python được phát triển để chạy trên nền Unix. Nhưng rồi theo thời gian,Python dần mở rộng sang mọi hệ điều hành từ MS-DOS đến Mac OS, OS/2, Windows, Linux và các hệ điều hành khác thuộc họ Unix. Mặc dù sự phát triển của Python có sự đóng góp của rất nhiều cá nhân, nhưng Guido van Rossum hiện nay vẫn là tác giả chủ yếu của Python. Ông giữ vai trò chủ chốt trong việc quyết định hướng phát triển của Python.

Python luôn được xếp hạng vào những ngôn ngữ lập trình phổ biến nhất.

- Ưu điểm

+ Là ngôn ngữ có hình thức và cấu trúc rõ ràng, cú pháp ngắn gọn

+ Có trên tất cả các nền tảng hệ điều hành

+ Tương thích mạnh mẽ với số lượng thư viện khổng lồ

+ Tốc độ xử lý cực nhanh, có thể tạo ra những chương trình từ những script siêu nhỏ tới những phần mềm cực lớn

- Nhược điểm

+ Không có các thuộc tính như: protected, private, public, không có vòng lặp do...while và switch...case

+ Tốc độ xử lý vẫn kém hơn Java và C++

2.1.2 Giới thiệu thư viện Pygame

Pygame là một bộ mô-đun Python đa nền tảng được thiết kế để viết trò chơi điện tử. Nó bao gồm đồ họa máy tính và thư viện âm thanh được thiết kế để sử dụng với ngôn ngữ lập trình Python.

Pygame sử dụng thư viện Simple DirectMedia Layer (SDL), với mục đích cho phép phát triển trò chơi máy tính trong thời gian thực mà không cần cơ chế bậc thấp của ngôn ngữ lập trình C và các dẫn xuất của nó. Điều này dựa trên giả định rằng các chức năng đắt tiền nhất bên trong trò chơi có thể được trừu tượng hóa khỏi logic trò chơi, do đó có thể sử dụng ngôn ngữ lập trình bậc cao, chẳng hạn như Python, để cấu trúc trò chơi.

Các tính năng khác mà SDL không có bao gồm toán học vectơ, phát hiện va chạm, quản lý đồ họa 2d, hỗ trợ MIDI, camera, thao tác mảng pixel, chuyển đổi, lọc, hỗ trợ phông chữ freetype nâng cao và vẽ.

Các ứng dụng sử dụng pygame có thể chạy trên điện thoại và máy tính bảng Android với việc sử dụng Bộ phụ pygame cho Android (pgs4a). Âm thanh, rung, bàn phím và gia tốc kế được hỗ trợ trên Android.

Một số hàm thông dụng được sử dụng trong pygame:

pygame.init()	Khởi tạo môi trường Pygame. Phải được gọi trước khi bắt đầu sử dụng bất kỳ chức năng Pygame nào.
pygame.display.*set_mode*((width, height))	Tạo ra một cửa sổ đồ họa với kích thước được chỉ định và trả về một đối tượng cửa sổ.
pygame.image.load("image.png")	Tải hình ảnh từ tệp hình ảnh.
pygame.mixer.Sound("sound.wav")	Tải âm thanh từ tệp âm thanh
pygame.event.get()	Lấy tất cả các sự kiện từ hàng đợi sự kiện. Sự kiện có thể là sự kiện đầu vào như phím bấm, chuột, vv.
pygame.quit()	Kết thúc môi trường Pygame. Cần được gọi trước khi chương trình kết thúc.
pygame.draw.polygon(surface,color,points, width)	Vẽ một hình đa giác lên bề mặt (surface) với màu viền (color) có độ dày (width) dựa tập hợp các điểm(points)
pygame.time.Clock()	Tạo một đối tượng đồng hồ để quản lý thời gian trong game.
pygame.time.Clock.tick(fps)	Giữ thời gian giữa các khung hình không vượt quá giới hạn fps.
pygame.display.*set_caption*("title")	Đặt tiêu đề cho cửa sổ game.
pygame.display.flip()	Cập nhật cửa sổ và hiển thị những thay đổi đã thực hiện.

Bảng 1: Các hàm thông dụng trong pygame

2.1.3 Giới thiệu thư viện socket

Một socket là một end-point của một liên kết giữa hai ứng dụng. Socket cho phép giao tiếp trong 1 tiến trình, giữa những tiến trình trên cùng 1 máy hoặc giữa nhiều máy với nhau.

Trong hệ thống mạng có rất nhiều ứng dụng bao gồm chương trình khách và chương trình chủ ở 2 hệ cuối(client và server) khác nhau. Sau khi được kích hoạt, một tiến trình khách và chủ được tạo và mục đích của socket được sử dụng là để giúp 2 tiến trình này có thể truyền thông với nhau dễ dàng

Python cung cấp module socket giúp chúng ta dễ dàng thực hiện kết nối client server để giao tiếp với nhau thông qua giao thức TCP/IP.

Để có thể sử dụng được trước tiên ta phải import module socket vào chương trình

```
1 import socket
```

Tạo một socket có kiểu kết nối ipv4 và giao thức TCP/IP

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

Các phương thức thường được sử dụng trong thư viện:

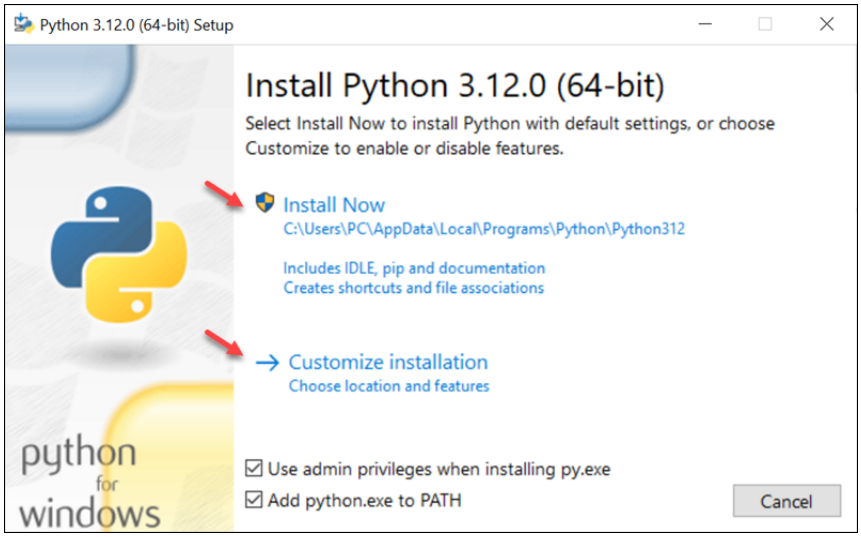
socket.bind((HOST, PORT))	Đăng ký tên cho socket, ràng buộc địa chỉ vào socket.
socket.listen(2)	Cho socket đang lắng nghe tới tối đa 2 kết nối.
client, addr = socket.accept()	Khi một client gõ cửa, server chấp nhận kết nối và 1 socket mới được tạo ra. Client và server bây giờ đã có thể truyền và nhận dữ liệu với nhau.
socket.connect((HOST, PORT))	Client tiến hành tìm kiếm server để kết nối với địa chỉ cho trước
data = client.recv(1024)	Nhận gói dữ liệu
$str_data = data.decode("utf8")$	Giải mã gói dữ liệu vừa nhận
socket.send(bytes(msg, "utf8"))	Gửi dữ liệu thông qua giao thức TCP
socket.close()	Đóng kết nối

Bảng 2: Các hàm thông dụng trong socket

2.2 Hướng dẫn cài đặt

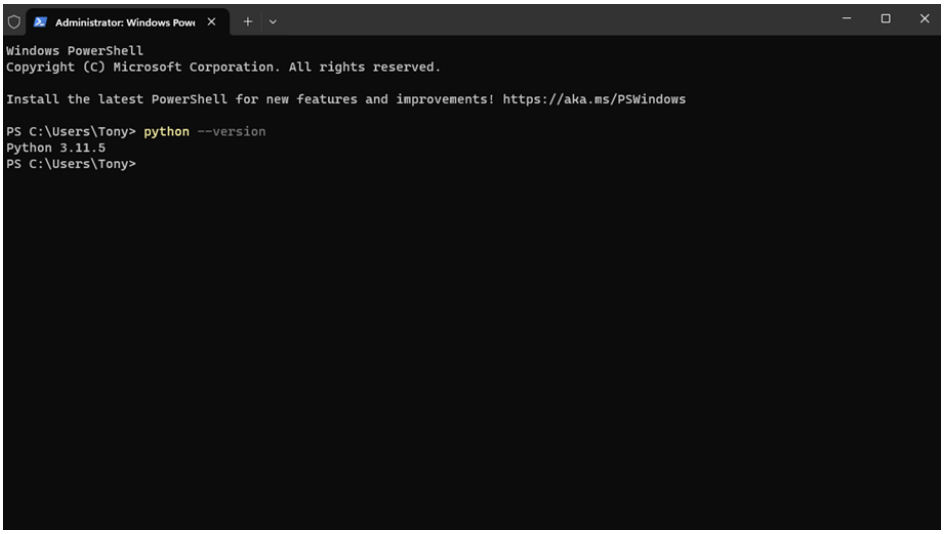
2.2.1 Cài đặt môi trường

Đầu tiên hãy truy cập vào python.org và chọn phiên bản tương ứng với hệ điều hành của máy.
-Chọn Add Python 3.12 to PATH → Install Now



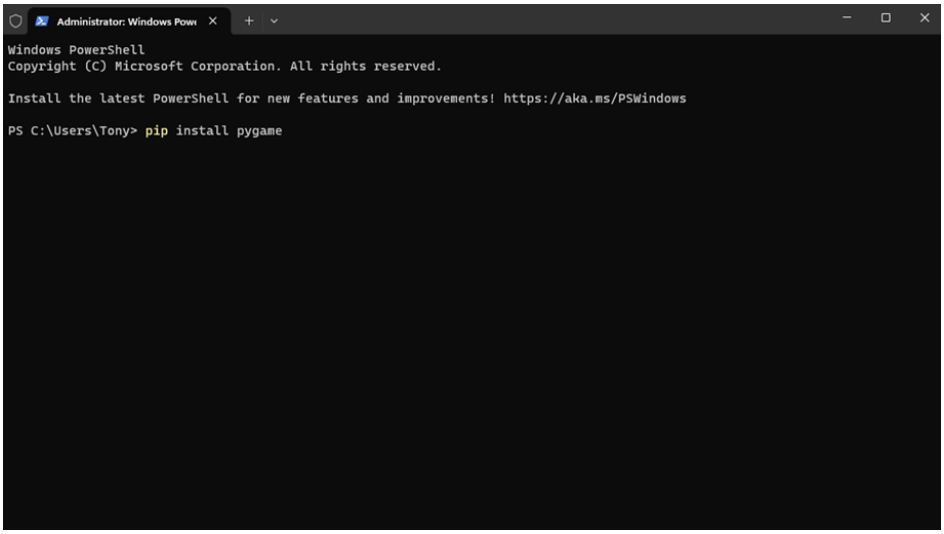
Hình 1: Cài đặt Python

-Bạn mở cmd từ thanh tìm kiếm trên window -> gõ: python –version -> enter



Hình 2: Cách xem version Python

- Cũng tại cửa sổ Command Prompt này, mình sẽ cài đặt thư viện pygame
- Gõ: pip install pygame



Hình 3: Cài đặt thư viện pygame

IDE sử dụng: Visual Studio Code hoặc Pycharm -3

2.2.2 Cài đặt ứng dụng và thực thi

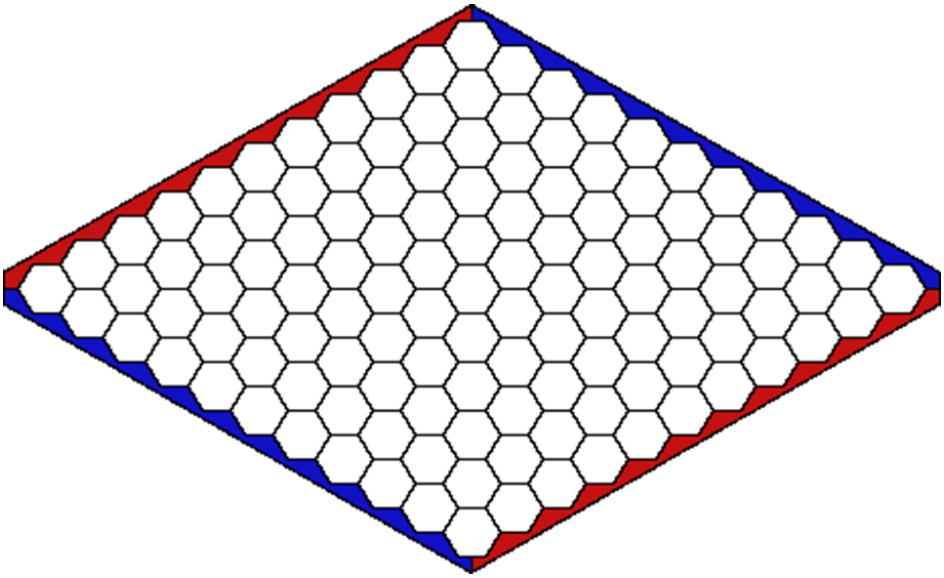
- Xem file README.md trên github ứng dụng

3 XÂY DỰNG GAME HEX VỚI THƯ VIỆN PYGAME

3.1 Giới thiệu game Hex

Hex là một trò chơi bàn cờ chiến lược trừu tượng dành cho hai người chơi, trong đó người chơi cố gắng nối các mặt đối diện của một bảng hình thoi làm bằng các ô lục giác. Hex được phát minh bởi nhà toán học kiêm nhà thơ Piet Hein vào năm 1942 và sau đó được John Nash khám phá lại và phổ biến.

Theo truyền thống, nó được chơi trên bảng hình thoi 11×11, mặc dù các bảng 13×13 và 19×19 cũng rất phổ biến. Bảng bao gồm các hình lục giác được gọi là tế bào hoặc hexes. Mỗi người chơi được chỉ định một cặp mặt đối diện của bảng, họ phải cố gắng kết nối bằng cách luân phiên đặt một viên đá màu của họ lên bất kỳ lục giác trống nào. Sau khi đặt, những viên đá không bao giờ được di chuyển hoặc loại bỏ. Một người chơi giành chiến thắng khi họ kết nối thành công các bên của họ với nhau thông qua một chuỗi các viên đá liền kề. Hòa là không thể trong Hex do cấu trúc liên kết của bảng trò chơi.



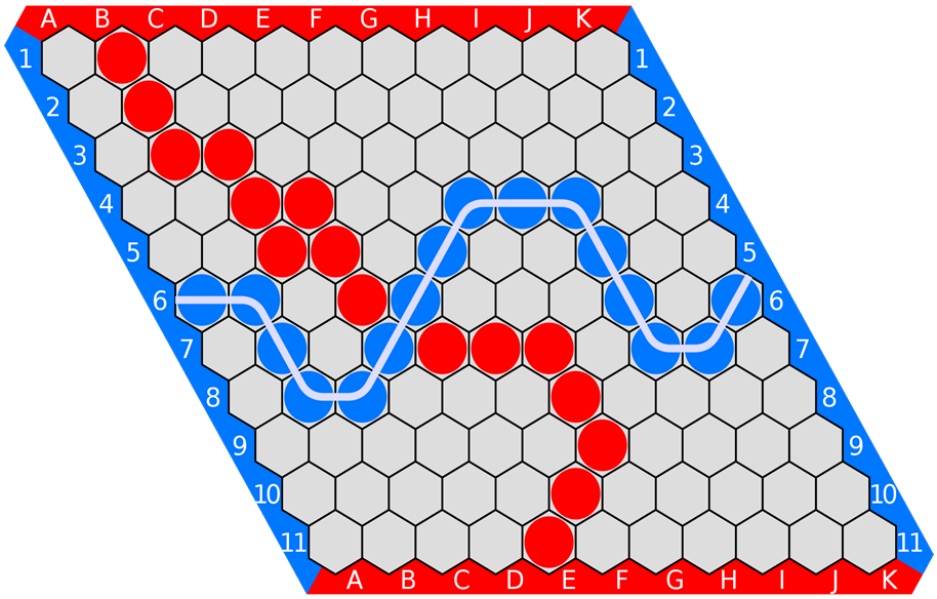
Hình 4: Ví dụ về 1 hex board

Mặc dù sự đơn giản của các quy tắc của nó, trò chơi có chiến lược sâu sắc và chiến thuật sắc nét. Nó cũng có nền tảng toán học sâu sắc liên quan đến định lý điểm cố định Brouwer, matroids và kết nối đồ thị. Hex cũng có thể được chơi bằng giấy và bút chì trên giấy đồ thị được cai trị lục giác.

3.2 Quy tắc chơi game

Hex được chơi trên một lưới hình thoi gồm các hình lục giác, thường có kích thước 11×11, mặc dù các kích thước khác cũng có thể. Mỗi người chơi có một màu được phân bổ, đỏ và xanh thông thường, hoặc đen và trắng. Mỗi người chơi cũng được gán hai cạnh bảng đối diện. Các hình lục giác trên mỗi góc trong số bốn góc thuộc về cả hai cạnh bảng liền kề.

Người chơi thay phiên nhau đặt một viên đá màu của họ lên một ô duy nhất trên bảng. Quy ước phổ biến nhất là Đỏ hoặc Đen đi trước. Sau khi đặt, đá không được di chuyển, thay thế hoặc loại bỏ khỏi bảng. Mục tiêu của mỗi người chơi là tạo thành một con đường kết nối của những viên đá của riêng họ liên kết hai cạnh bảng của họ. Người chơi hoàn thành kết nối như vậy sẽ thắng trò chơi.



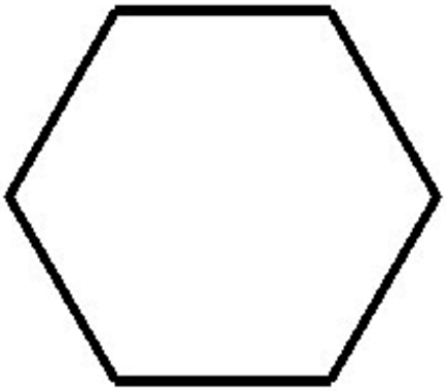
Hình 5: Người chơi xanh chiến thắng vì đã kết nối 2 cạnh bảng xanh lại với nhau

3.3 Ứng dụng thư viện pygame xây dựng trò chơi Hex

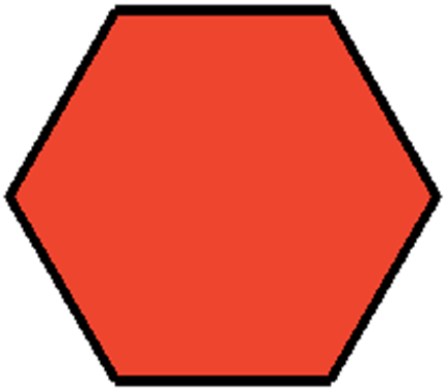
3.3.1 Các lớp quan trong của trò chơi

1. Lớp Hexagon(lục giác):

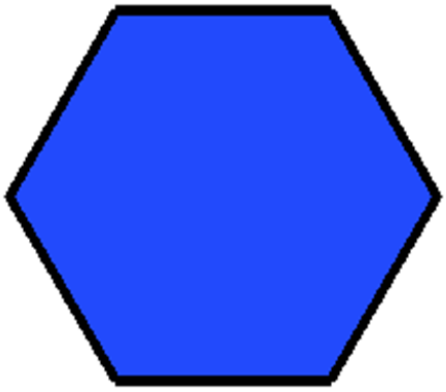
-Là đối tượng chính để tương tác của trò chơi
-Gồm có 3 trạng thái:



Hình 6: Chưa bị chiếm đóng

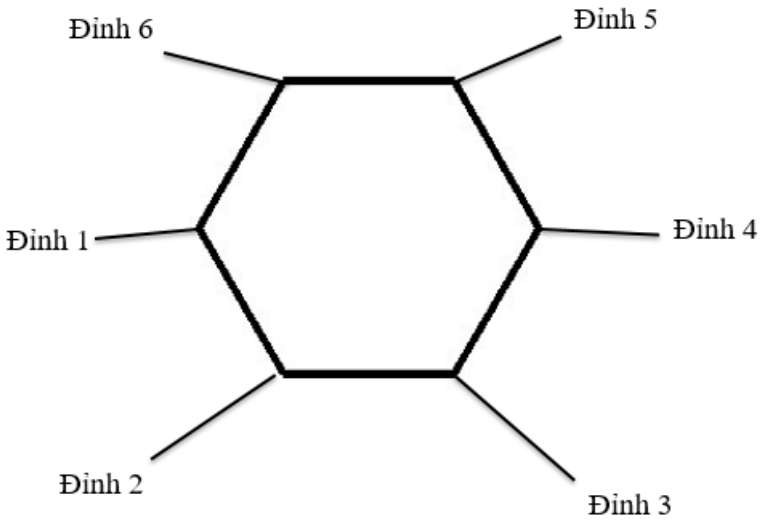


Hình 7: Bị người chơi 1 chiếm đóng



Hình 8: Bị người chơi 2 chiếm đóng

- Cách vẽ: sử dụng hàm pygame.draw.polygon(surface, color, point, width) để vẽ lục giác không màu. Trong đó:
+surface: là bề mặt cần để vẽ lục giác
+color: màu viền lục giác, ở đây mặc định là màu đen
+width: Độ dày của đường viền
+point: tập hợp các đỉnh của lục giác theo các tọa độ sau:

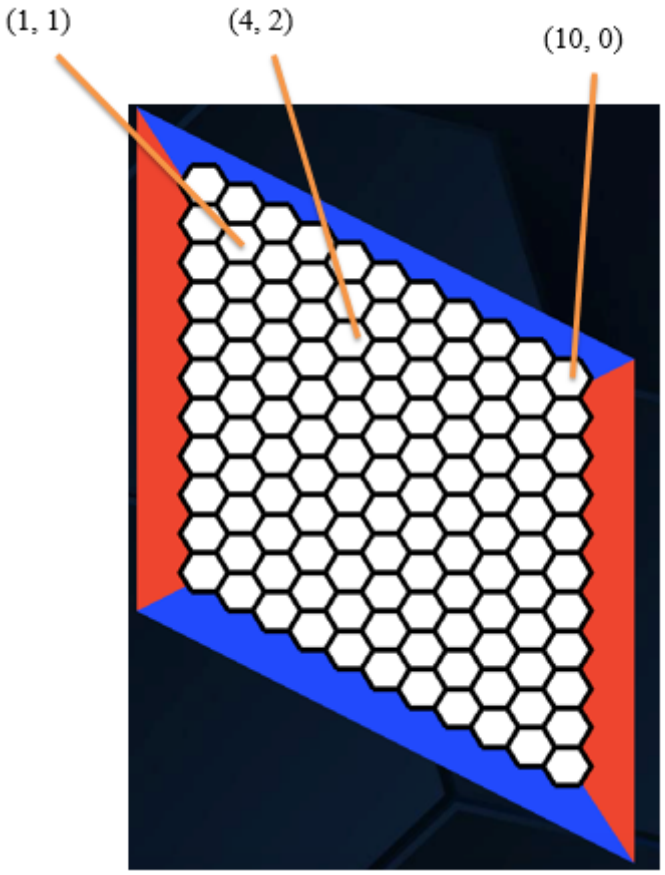


Hình 9: Các đỉnh lục giác theo thứ tự

· (x, y): Đỉnh 1
· (x + 3 * halfRadius, y + minimalRadius): Đỉnh 2
· (x + 3 * halfRadius, y + minimalRadius): Đỉnh 3
· (x + 4 * halfRadius, y): Đỉnh 4
· (x + 3 * halfRadius, y - minimalRadius): Đỉnh 5
· (x + halfRadius, y - minimalRadius): Đỉnh 6

*Với: x, y là số bất kỳ
halfRadius là nửa độ dài 1 cạnh lục giác
minimalRadius là bán kính đường tròn nội tiếp lục giác

2. Lớp Board(bảng):
   - Là lớp chứa đường viền 2 màu cùng với các ô lục giác có cùng số dòng và số cột
   - Mỗi ô lục giác đều được gắn toạ độ tương ứng với vị trí trên bảng



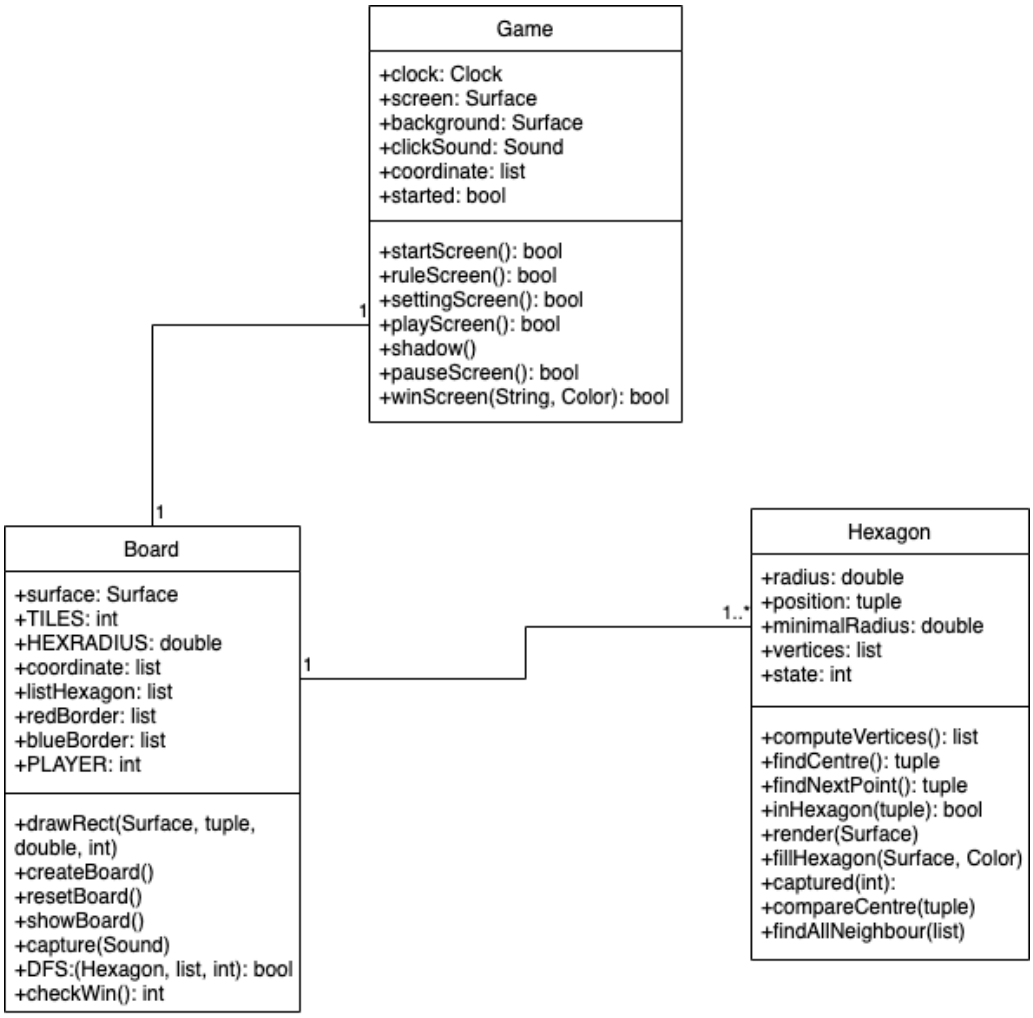
Hình 10: Ví dụ về tọa độ trên bảng

3.3.2 Thuật toán sử dụng

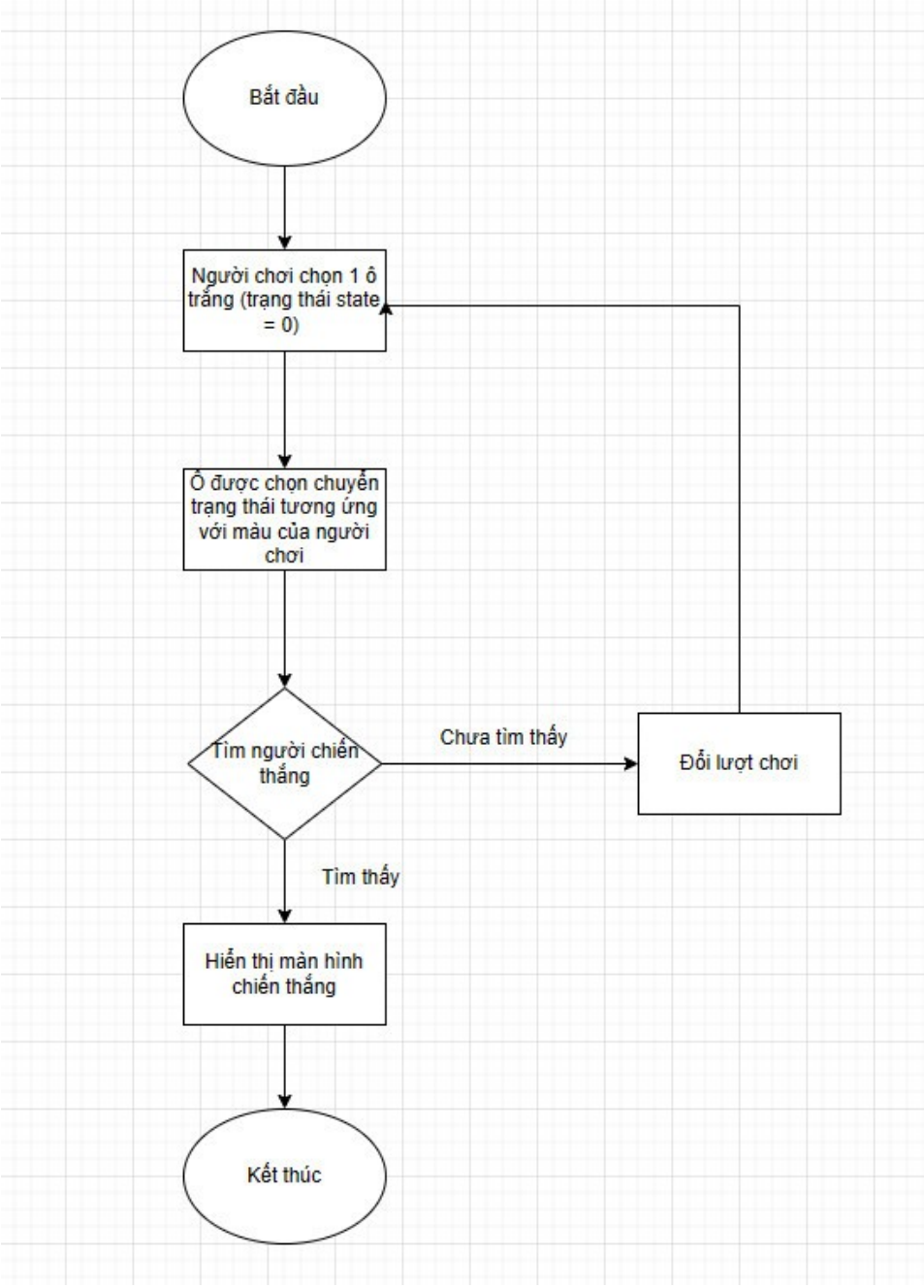
- Ứng dụng thuật toán DFS(Tìm kiếm theo chiều sâu) để tìm đường đi nối giữa 2 cạnh bảng đối diện với nhau, từ đó tìm được người chiến thắng
- Các bước thực hiện:

Khởi tạo stack theo class deque và đưa vào ô giáp viền cần xét
Khởi tạo list visited để đánh dấu các ô đã xét
while stack không rỗng:
    Lấy ô đầu stack ra khỏi stack để xét. Nếu stack rỗng thì trả về
False
    If ô đang xét nằm trong danh sách ô đích
    Trả về True và kết thúc thuật toán, ngược lại thì tiếp tục
    Đưa ô đang xét vào visited
    Tìm danh sách các ô kề cạnh ô đang xét
    If ô trong danh sách vừa tạo có trạng thái cùng màu
với người chơi đang xét và không nằm trong visited
    Đưa ô đó vào trong stack
If stack rỗng:
Trả về False và kết thúc thuật toán

3.3.3 Các sơ đồ



Hình 11: Sơ đồ lớp



Hình 12: Flowchart chơi trò chơi

3.3.4 Source code các lớp quan trọng

const.py

```
W = 1000
H = 800
STARTPOS =(W/3, H/4.5)
BLACK =(  0,   0,   0)
WHITE =(255, 255, 255)
RED   = (255,   0,   0)
BLUE  = (  0,   0, 255)
GOLD =(255,215,0)
ORANGE =(255, 165, 0)
FPS =30
```

+W: Chiều rộng màn hình chơi
+H: Chiều dài màn hình chơi
+STARTPOS: Toạ độ điểm bắt đầu vẽ lục giác đầu tiên
+BLACK: Chỉ số RGB của màu đen
+WHITE: Chỉ số RGB của màu trắng
+RED: Chỉ số RGB của màu đỏ
+BLUE: Chỉ số RGB của màu xanh
+GOLD: Chỉ số RGB của màu vàng kim
+ORANGE: Chỉ số RGB của màu cam
+FPS: Số khung hình trên 1 giây

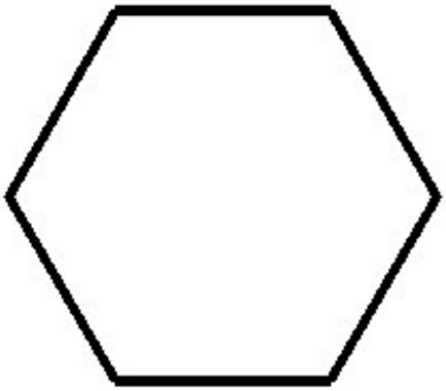
Hexagon.py

```
import math
from const import *
import pygame as pg
from pygame import gfxdraw
```

+Các thư viện và module cần import

```
class Hexagon:
  def __init__(self, radius, position) ->None:
        self.radius =radius
        self.position =position
        self.minimalRadius =math.cos(math.radians(30)) *radius
        self.vertices =self.computeVertices()
        self.centre =self.findCentre()
        self.state =0
```

+Hàm khởi tạo gồm các thuộc tính:
· self.radius: Bán kính đường tròn ngoại tiếp lục giác
· self.position: Vị trí bắt đầu vẽ lục giác(góc trái)
· self.minimalRadius: Bán kính đường tròn nội tiếp lục giác
· self.vertices: List chứa tọa độ của đỉnh lục giác
· self.centre: Tọa độ tâm của lục giác
· self.state: Trạng thái hiện tại của lục giác



Hình 13: Lục giác có state = 0

```
def computeVertices(self):
    x, y = self.position
    halfRadius = self.radius /2
    return [(x,y),
            (x + halfRadius, y + self.minimalRadius),
            (x + 3 * halfRadius, y + self.minimalRadius),
            (x + 4 * halfRadius, y),
            (x + 3 * halfRadius, y - self.minimalRadius),
            (x + halfRadius, y - self.minimalRadius) ]
```

+Hàm tìm toạ độ các đỉnh: trả về 1 list chứa tọa độ các đỉnh của đối tượng này theo thứ tự:
- (x, y): Đỉnh góc trái
- (x + 3 * halfRadius, y + self.minimalRadius): Đỉnh góc trái dưới
- (x + 3 * halfRadius, y + self.minimalRadius): Đỉnh góc phải dưới
- (x + 4 * halfRadius, y): Đỉnh góc phải
- (x + 3 * halfRadius, y - self.minimalRadius): Đỉnh góc phải trên
- (x + halfRadius, y - self.minimalRadius): Đỉnh góc trái trên

```
def findCentre(self):
    x, y = self.position
    return (x + self.radius, y)
```

+Hàm tìm tâm lục giác: trả về tọa độ tâm lục giác

```
def findNextPoint(self):
    return self.vertices[2]
```

+Hàm tìm vị trí lục giác kế cạnh: trả về đỉnh toạ độ bắt đầu vẽ của lục giác kế cạnh(góc phải dưới của lục giác này)

```
def inHexagon(self, point):
    distance =math.dist(self.centre, point)
    return distance <self.minimalRadius
```

+Hàm kiểm vị trí có nằm trong phạm vi của lục giác: trả về true nếu vị trí đang xét nằm trong phạm vi lục giác này, ngược lại trả về false

```
def render(self, screen):
    pg.draw.polygon(screen, BLACK, self.vertices, 5)
```

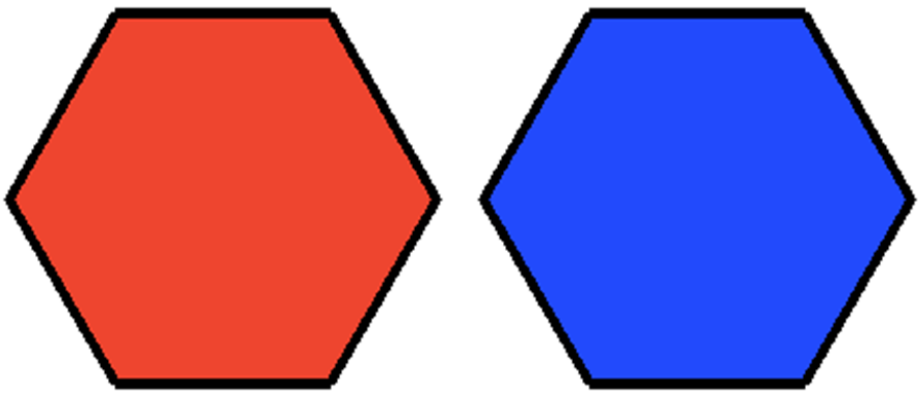
+Hàm vẽ lục giác: vẽ lục giác viền đen lên màn hình dựa trên toạ độ các đỉnh

```
def fillHexagon(self, screen, color):
    gfxdraw.filled_polygon(screen, self.vertices, color)
    self.color =color
```

+Hàm tô màu lục giác: tô màu cho lục giác hiện tại với color được truyền vào từ tham số

```
def captured(self, player):
    self.state =player
```

+Hàm thay đổi trạng thái: thay đổi trạng thái dựa trên player nào chiếm đóng được đối tượng này



Hình 14: (a) Lục giác có state = 1 (b) Lục giác có state = 2

```
def compareCentre(self, point):
    x, y =self.centre
    a, b =point
    return x ==a and y ==b
```

+Hàm so sánh: kiểm tra xem điểm đang xét có trùng với tâm lục giác không. Nếu có trả về true và ngược lại là false

```
def findAllNeighbour(self, listHex):
     neighbour =[]
     for hexagon in listHex:
          distance =math.dist(self.centre, hexagon.centre)
        if math.isclose(distance, 2*self.minimalRadius, rel_tol=0.05):
           neighbour.append(hexagon)
     return neighbour
```

+Hàm tất cả các lục giác kế cạnh: trả về list các lục giác kế cạnh lục giác này(có khoảng cách giữa 2 tâm lục giác xấp xỉ 2 lần bán kính đường tròn nội tiếp)

Client1_Board.py/Client2_Board.py: Class board của mỗi client tương ứng

```
from collections import deque
import tkinter
from tkinter import Tk
import pygame as pg
from const import *
from Hexagon import *
import socket
import threading
```

+Import các thư viện và module cần thiết

```
class Client1_Board:
    def __init__(self, surface, size):
        self.surface =surface
        self.TILES =size
        self.HEXRADIUS =H/(20 +self.TILES*2)
        self.coordinate =[[0 for i in range(self.TILES)] for j in range(self.TILES)]
        self.listHexagon =[]
        self.redBorder =[]
        self.blueBorder =[]
        self.PLAYER =1
        self.turn =True
        self.player1 ="player1"
        self.player2 ="player2"
        self.interupt =False

        self.sock =socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

+Hàm khởi tạo gồm các thuộc tính:

· self.surface: Bề mặt vẽ board được gán bằng tham số surface

· self.TILES: Số lục giác trên 1 dòng được gán bằng tham số size

· self.HEXRADIUS: Độ dài 1 cạnh lục giác được tính bằng Chiều cao màn hình chơi/(20 + số lục giác trên 1 dòng)

· self.coordinate[][]: Mảng 2 chiều chứa toạ độ của toàn bộ lục giác trên bảng

· self.listHexagon[]: Mảng chứa toàn bộ đối tượng lục giác trên bảng, dùng khi cần xét từng đối tượng một trên bảng

· self.redBorder[]: Mảng chứa các lục giác giáp đường viền đỏ có toạ độ (self.TILES – 1, y)

· self.blueBorder[]: Mảng chứa các lục giác giáp đường viền đỏ có toạ độ (x, self.TILES – 1)

· self.PLAYER: Số hiệu của người chơi. Client1 có số hiệu 1(tương ứng màu đỏ) và Client2 có số hiệu 2(tương ứng màu xanh)

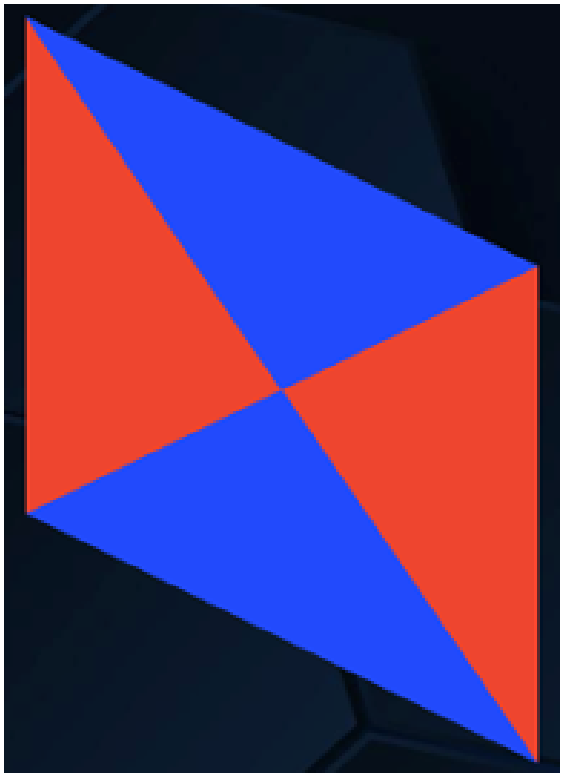
· self.player1/2: Biến chứa nickname của client1 và client2

· self.interupt: Biến gián đoạn cuộc chơi. Mặc định để là False, khi chuyển thành True thì

tương ứng đối phương đã rời khỏi cuộc chơi · self.sock: Khởi tạo socket để giao tiếp với server

```
def drawRect(self, surface, pos, height, boardSize):
    x, y =pos
    pg.draw.polygon(surface, RED, [(x,y), (x, y+height), (x+height-height/boardSize,
                                                y+height/2), (x+height-height/boardSize
                                                ,y+height/2+height)])
    pg.draw.polygon(surface, BLUE, [(x,y), (x+height-height/boardSize, y+height/2), (x,
                                                y+height), (x+height-height/boardSize,
                                                y+height/2+height)])
```

+Hàm vẽ viền: Dùng vẽ các đường viền bên ngoài các lục giác. Các đường viền này thực chất là 2 hình tam giác đối đỉnh màu xanh và 2 hình tam giác đối đỉnh màu đỏ ghép lại với nhau tạo thành 1 hình thoi, khi vẽ các lục giác lên sẽ che đi phần giữa của hình



Hình 15: Bảng khi chưa vẽ các lục giác

```
def createBoard(self):
      x, y =STARTPOS
      for i in range(self.TILES):
          distance =0
          for j in range(self.TILES):
              self.coordinate[i][j] =Hexagon(self.HEXRADIUS, (x, y+distance))
              distance +=self.coordinate[i][j].minimalRadius *2
              self.listHexagon.append(self.coordinate[i][j])
              if i ==self.TILES-1:
```

```
                self.redBorder.append(self.coordinate[i][j])
            if j ==self.TILES-1:
                self.blueBorder.append(self.coordinate[i][j])
        x, y =self.coordinate[i][0].findNextPoint()
```

+Hàm khởi tạo bảng: Lần lượt khởi tạo các đối tượng lục giác và gán chúng cho toạ độ của bảng. Các đối tượng sẽ có khoảng cách giữa 2 tâm lục giác bằng 2 lần bán kính đường tròn nội tiếp lục giác. Trong hàm này ta cũng đồng thời gán các đối tượng lục giác tương ứng cho 2 list redBorder và blueBorder

```
def resetBoard(self):
        for hexagon in self.listHexagon:
            hexagon.state =0
        self.sock.send(bytes("{P1}", "utf8"))
        self.sock.close()
```

+Hàm reset bảng: Chuyển trạng thái của tất các lục giác thành 0(màu trắng), đồng thời gửi tín hiệu quay lại màn hình chính đến server và đóng kết nối

```
def showBoard(self):
        x, y =STARTPOS
        self.drawRect(self.surface,(x-2*self.coordinate[0][0].minimalRadius, y -4 *
                                                    self.coordinate[0][0].minimalRadius),
                                                    self.coordinate[0][0].minimalRadius *
                                                    (self.TILES+2) *2, self.TILES)
        for col in range(self.TILES):
            for row in range(self.TILES):
                if self.coordinate[col][row].state ==0:
                    self.coordinate[col][row].fillHexagon(self.surface, WHITE)
                    self.coordinate[col][row].render(self.surface)
                    if self.coordinate[col][row].inHexagon(pg.mouse.get_pos()):
                        if self.PLAYER ==1:
                            self.coordinate[col][row].fillHexagon(self.surface, RED)
                            self.coordinate[col][row].render(self.surface)
                elif self.coordinate[col][row].state ==1:
                    self.coordinate[col][row].fillHexagon(self.surface, RED)
                    self.coordinate[col][row].render(self.surface)
                else:
                    self.coordinate[col][row].fillHexagon(self.surface, BLUE)
                    self.coordinate[col][row].render(self.surface)
```

+Hàm vẽ board: Gọi hàm vẽ viền trước rồi lần lượt tô màu từng lục giác ứng với state của nó. Xét từng lục giác để tô màu lục giác đang chứa vị trí hiện tại của con trỏ chuột theo màu của client.

```
def capture(self, sound):
        for col in range(self.TILES):
            for row in range(self.TILES):
                if self.coordinate[col][row].inHexagon(pg.mouse.get_pos())\
                    and self.coordinate[col][row].state ==0 and self.turn ==True:
                        if self.PLAYER ==1:
                            pg.mixer.Sound.play(sound)
                            self.coordinate[col][row].captured(self.PLAYER)
                            send_data ='{},{},{},{},{}'.format('GAME',col, row, True,
                                                                self.PLAYER).encode()
                            self.sock.send(send_data)
                            self.turn =False
```

+Hàm chiếm đóng: Gọi hàm captured của đối tượng để chuyển trạng thái khi người chơi click vào ô có state = 0. Sau khi thực hiện sẽ gửi tín hiệu (GAME, toạ độ lục giác chuyển state, biến chuyển lượt đi) đến server

```
def DFS(self, start, finish, player):
        stack =deque()
        stack.append(start)
        visited =[]
        while len(stack):
            current =stack.pop()
            for hexagon in finish:
                if current is hexagon:
                    return True
            visited.append(current)
            listNeighbour =current.findAllNeighbour(self.listHexagon)
            for hexagon in listNeighbour:
                if hexagon not in visited and hexagon.state ==player:
                    stack.append(hexagon)
        return False
```

+Thuật toán DFS(Tìm kiếm theo chiều sâu): Tìm kiếm đường đi nối giữa 2 đường viền đối diện nhau. Các tham số truyền vào gồm có:

· start: Ô bắt đầu xét, có toạ độ (0, y) với player 1 và (x, 0) với player 2

· finish: List các ô ở vị trí đích(list redBorder đối với player 1 và list blueBorder đối với player 2)

· player: Trạng thái các ô cần tìm để tạo đường đi Các bước tìm kiếm được thực hiện theo các bước sau:

· B1: Khởi tạo stack theo class deque và đưa vào lục giác start cần xét

· B2: Khởi tạo list visited để đánh dấu các ô đã xét

· B3: Nếu stack không rỗng thì lấy ô đầu stack ra khỏi stack để xét. Nếu stack rỗng thì trả về False

· B4: Nếu ô đang xét nằm trong list finish thì trả về True và kết thúc, ngược lại thực hiện bước 5

· B5: Đưa ô đang xét vào visited

· B6: Tạo ra 1 list các ô kế cạnh bằng cách gọi hàm findAllNeighbour của đối tượng đang xét

· B7: Xét từng ô trong list vừa tạo, ô nào có state = biến player đã truyền vào và không nằm trong visited thì đưa vào stack

· B8: Quay lại bước 3

```
def checkWin(self):
        for row in range(self.TILES):
            if self.coordinate[0][row].state ==1:
                if self.DFS(self.coordinate[0][row], self.redBorder, 1):
                    return 1
        for col in range(self.TILES):
            if self.coordinate[col][0].state ==2:
                if self.DFS(self.coordinate[col][0], self.blueBorder, 2):
                    return 2
        return 0
```

+Hàm tìm người chiến thắng: Tìm người chiến thắng bằng cách gọi hàm tìm kiếm DFS với:

· Ô bắt đầu có toạ độ (0, y) và list redBorder là các vị trí đích đối với người chơi 1

· Ô bắt đầu có toạ độ (x, 0) và list blueBorder là các vị trí đích đối với người chơi 2
Nếu tìm được người chiến thắng thì trả về số của người chơi đó, ngược lại thì trả về 0

```
def create_thread(self, target):
        thread =threading.Thread(target=target)
        thread.daemon =True
        thread.start()
```

+Hàm tạo thread: Tiến hành tạo luồng phụ với hàm được đưa vào và cho chạy trong background

```
def receive_data(self):
        while True:
            message =self.sock.recv(1024).decode()
            if( str(message).startswith("{back}") ):
                self.interupt =True
                break
            elif( str(message).startswith("{start}") ):
                self.begin_button["state"] ="normal"
            elif( str(message).startswith("{quit}") ):
                self.begin_button["state"] ="disabled"
            elif( str(message).startswith("PLAYERNAME") ):
                data =message.split(",")
                self.player1 =str(data[1])
                self.player2 =str(data[2])
            else:
                if( str(message).startswith("GAME") ):
                    data =message.split(",")
                    x, y =(data[1], data[2])
                    if(bool(data[3]) ==True and int(data[4]) ==2):
                        self.turn =True
                        self.otherCapture(x, y, data[4])
                else:
                    try:
                        self.msg_list.insert(tkinter.END, message)
                    except:
                        print('An error occurred!')
```

+Hàm nhận phản hồi: Tạo vòng lặp vô tận liên tục tiếp nhận phản hồi từ phía server và giải mã. Sau đó thực hiện hành động ứng với tín hiệu từ phía server:

· back: Tín hiệu cho thấy đối phương đột ngột dừng cuộc chơi, chuyển biến self.interupt thành True và kết thúc vòng lặp

· start: Tín hiệu báo đã đủ người chơi trong room, nút "Bắt đầu chơi" trong room sẽ xuất hiện

· quit: Tín hiệu báo đối phương đã thoát khỏi room, nút "Bắt đầu chơi" sẽ bị disabled

· PLAYERNAME: Tín hiệu gửi đến khi room có đủ người chơi, chứa nickname của các client trong phòng

· GAME: Tín hiệu chứa toạ độ ô lục giác đối phương vừa chiếm giữ. Gọi hàm otherCapture() với tham số là toạ độ được gửi đến

· Nếu không thuộc bất kỳ tín hiệu nào trên thì phản hồi gửi đến là tin nhắn trong chatbox. Hiển thị tin nhắn trong chatbox của giao diện phòng chat

```
def otherCapture(self, x, y, player):
        self.coordinate[int(x)][int(y)].captured(int(player))
```

+Hàm đối phương chiếm đóng: chuyển trạng thái của lục giác ở toạ độ từ tham số truyền vào

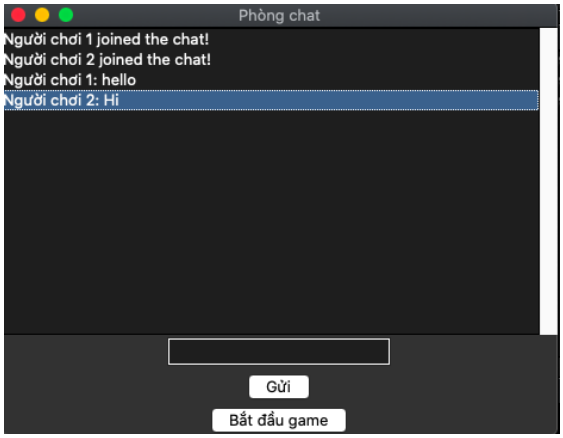
sang màu của đối phương

```
def waiting_connection(self):
        HOST ="0.0.0.0"
        PORT =5000
        while True:
            try:
                self.sock.connect((HOST, PORT))
                self.create_thread(self.receive_data)
                self.create_chat_UI()
                return True
            except Exception as e:
                pass
```

+Hàm đợi kết nối: Thực hiện kết nối đến server. Sau khi kết nối thành công sẽ tạo 1 luồng mới cho receive_data() và mở giao diện phòng chat

```
def create_chat_UI(self):
        self.top =Tk()
        self.top.title("Phng chat")

        messages_frame =tkinter.Frame(self.top)
        self.my_msg =tkinter.StringVar() # For the messages to be sent.
        self.my_msg.set(" Nhp  nickname  ca   bn ")

        scrollbar =tkinter.Scrollbar(messages_frame) # To see through previous messages.
        # this will contain the messages.
        self.msg_list =tkinter.Listbox(messages_frame, height=15, width=50,
                                                    yscrollcommand=scrollbar.set)
        scrollbar.pack(side=tkinter.RIGHT, fill=tkinter.Y)
        self.msg_list.pack(side=tkinter.LEFT, fill=tkinter.BOTH)
        self.msg_list.pack()
        messages_frame.pack()

        entry_field =tkinter.Entry(self.top, textvariable=self.my_msg)
        entry_field.bind("<Return>", self.send)
        entry_field.pack()

        send_button =tkinter.Button(self.top, text="Gui", command=self.send)
        send_button.pack()

        self.begin_button =tkinter.Button(self.top, text="Bat dau game", command=self.on_closing)
        self.begin_button.pack()
        self.begin_button["state"] ="disable"

        self.top.protocol("WM_DELETE_WINDOW", self.on_closing)
        tkinter.mainloop() # for start of GUI Interface
```

+Hàm tạo phòng chat: Phòng chat gồm 1 chatbox, 1 ô input nhập tin nhắn, 1 button gửi tin nhắn và 1 button bắt đầu game



Hình 16: Giao diện phòng chat

```
def send(self, event=None): # event is passed by binders.play
        msg = self.my_msg.get()
        self.my_msg.set("") # Clears input field.
        self.sock.send(bytes(msg, "utf8"))
        if msg =="{quit}":
            self.top.destroy()
```

+Hàm gửi tin: Thực hiện gửi tin trong ô input đến server. Nếu tin được gửi có tín hiệu {quit} thì đóng giao diện phòng chat và bắt đầu chơi

```
def on_closing(self, event=None):
        """This function is to be called when the window is closed."""
        self.my_msg.set("{quit}")
        self.send()
```

+Hàm đóng giao diện: Set input tin nhắn thành tín hiệu {quit} và gọi hàm send()

```
def quit_game(self):
        self.sock.send("{quit}".encode("utf-8"))
        self.resetBoard()
```

+Hàm thoát trò chơi: Gửi tín hiệu quit đến server và gọi hàm resetBoard()
room_server.py: server của trò chơi

```
import socket
import threading
import time
```

+Import các thư viện và module cần thiết

```
host = ''
port = 5000

server =socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind((host, port))
server.listen(2)
```

+Thực hiện khởi tạo server với host và port cho trước, chỉ cho phép tối đa 2 client kết nối tới

```
clients =[]
nicknames =[]
players =[]
```

+clients: Danh sách các client đang trong phòng chat
+nicknames: Danh sách nickname của các client đang kết nối
+players: Danh sách các client đang chơi

```
def broadcast(message):
    for client in clients:
        client.send(message)
```

+Hàm phát tin: Gửi tin nhắn đến toàn bộ client đang ở phòng chat

```
def gamebroadcast(message):
    while len(clients) !=0:
        continue
    for player in players:
        player.send(message)
```

+Hàm phát tin trong cuộc chơi: Gửi tin đến toàn bộ client đang chơi game. Tin sẽ không được gửi trừ phi không còn ai trong phòng chat

```

def handle(client):
    while True:
        message =client.recv(1024).decode()
        print(message)
        if( message !="{quit}" ):
            if( str(message).startswith("GAME") ):
                print(message)
                gamebroadcast(message.encode('utf-8'))
            elif( str(message).startswith("{P1}")):
                print("Player 1 left the game")
            elif( str(message).startswith("{P2}")):
                print("Player 2 left the game")
            else:
                index =clients.index(client)
                nickname =nicknames[index]
                messages =f'{nickname}: ' +message
                broadcast(messages.encode('utf-8'))
        else
            try:
                index =clients.index(client)
                nickname =nicknames[index]
                clients.remove(client)
                players.append(client)
                broadcast(f'{nickname}    vo   chi !'.encode('utf-8'))
            except ValueError as e:
                index =players.index(client)
                nickname =nicknames[index]
                players.remove(client)
                broadcast(f'{nickname}    thot !'.encode('utf-8'))
                time.sleep(0.01)
                broadcast("{quit}".encode("utf-8"))
                if( len(players) ==1 ):
                    (players.pop()).send("{back}".encode("utf-8"))
```

+Hàm xử lý tin nhắn từ client: Tạo 1 vòng lặp vô tận liên tục nhận tin nhắn từ phía client và giải mã chúng. Dựa vào tín hiệu tự client gửi tới sẽ có cách xử lý tương ứng:

· GAME: Gọi hàm gamebroadcast với tham số là tin vừa được gửi đến

· P1: In thông báo player 1 đã rời phòng

· P2: In thông báo player 2 đã rời phòng

· quit: Xử lý khi client rời phòng

- Nếu client thoát phòng chat thì sẽ lấy socket của client đó ra khỏi tập clients[] và thêm vào tập players[] đồng thời gửi tin nhắn cho client còn lại trong phòng

- Nếu client thoát giữa cuộc chơi thì sẽ lấy socket của client đó ra khỏi tập players[] và thêm vào tập players[] và gửi tin nhắn thông báo đã thoát đến client đang trong phòng chat hoặc gửi tín hiệu gián đoạn cho client đang chơi

· Nếu không có tín hiệu nào thì truyền tin nhắn đến các client trong phòng chat

```
def receive():
    while True:
        client, address =server.accept()
        print(f'Connected with {str(address)}')
        nickname =client.recv(1024).decode('utf-8')
        nicknames.append(nickname)
        clients.append(client)
        print(f'Nickname of client is {nickname}')
        broadcast(f'{nickname} joined the chat!'.encode('utf-8'))
        thread =threading.Thread(target=handle, args=(client,))
        thread.start()
        if( len(clients) ==2 ):
            time.sleep(0.01)
            broadcast(f"PLAYERNAME,{nicknames[0]},{nicknames[1]}".encode('utf-8'))
            time.sleep(0.01)
            broadcast("{start}".encode("utf-8"))
```

+Hàm nhận kết nối từ client: accept yêu cầu kết nối từ client, thông báo client đã vào phòng chat khi nhận được nickname và cho chạy thread xử lý tin nhắn cho client đó. Khi phòng chat có đủ người chơi sẽ phát tín hiệu cho phép bắt đầu trò chơi

```
print('Server is listening...')
thread =threading.Thread(target=receive)
thread.start()
```

+Chạy server khi file được gọi

Game.py: Duy trì mọi hoạt động của giao diện

```
import sys
import time
import pygame as pg
from os import path
from pygame.locals import QUIT
from const import *
from Client1_Board import *
from Client2_Board import *
from TextButton import *
from PlayerTag import *
from tkinter import Tk
```

+Import các thư viện và module cần thiết

```
board =None
```

```
    tiles =11
    def __init__(self, type):
        root =Tk()
        root.destroy()
        pg.init()
        pg.display.set_caption('Hex game')
        self.clock =pg.time.Clock()
        self.screen =pg.display.set_mode((W, H))
        file_path =path.join(path.dirname(__file__), "other")
        self.background =pg.image.load(path.join(file_path, "bg1.jpg")).convert_alpha()
        self.clickSound =pg.mixer.Sound(path.join(file_path, "click.wav"))
        self.started =False

        self.type =type
```

+Hàm khởi tạo gồm các thuộc tính:

· Biến board thuộc lớp Board dùng để tạo bảng

· Biến tiles là số lượng ô lục giác trên 1 hàng, có giá trị mặc định bằng 11

· pg.init(): Khởi động các hàm trong thư viện pygame

· pg.display.set_caption(): Đặt tiêu đề cho màn hình game

· self.clock = pg.time.Clock(): Tạo đối tượng Clock để theo dõi thời gian

· self.screen = pg.display.set_mode(W, H): Đặt độ lớn của màn hình game theo hằng số W và H

· file_path: tạo đường dẫn đến thư mục other bằng module path

· self.background: Load background game từ đường dẫn file_path

· self.clickSound: Load hiệu ứng âm thanh khi click vào 1 đối tượng từ đường dẫn file_path

· self.started: Xác định đã bắt đầu game chưa, mặc định là False

· self.type: Xác định client của giao diện từ tham số truyền vào

```
def startScreen(self):
        start =True
        title =TextButton(self.screen, "HEX GAME", (W/2, H/4), 100, GOLD)
        startGame =TextButton(self.screen, "Bat au choi", (W/2, H/2), 60, WHITE)
        rule =TextButton(self.screen, "Luat choi", (W/2, H -H/2 +100), 60, RED)
        buttons =[startGame, rule]
        while start:
            self.screen.fill(WHITE)
            self.screen.blit(self.background, (0, 0))
            self.clock.tick(FPS)
            for event in pg.event.get():
                if event.type ==QUIT:
                    pg.quit()
                    sys.exit()
                if event.type ==pg.MOUSEBUTTONDOWN:
                    if startGame.click():
                        pg.mixer.Sound.play(self.clickSound)
                        self.started =True
                        start =False
                        if(self.type =="client1"):
                            self.board =Client1_Board(self.screen, self.tiles)
                            self.board.createBoard()
                        else:
                            self.board =Client2_Board(self.screen, self.tiles)
                            self.board.createBoard()
                        self.playScreen()
                        return True
                    elif rule.click():
```

```
                    pg.mixer.Sound.play(self.clickSound)
                    self.ruleScreen()
                    return True
            title.printText(100)
            for b in buttons:
                b.render()
            pg.display.flip()
```

+Màn hình bắt đầu: Khởi động khi chạy ứng dụng, gồm có: -Click vào “Bắt đầu chơi” sẽ tiến hành tạo board và chuyển sang màn hình chơi -Click vào "Luật chơi" sẽ chuyển sang màn hình luật chơi

```
def ruleScreen(self):
        rule =True
        title =TextButton(self.screen, "Luat choi", (W/2, H/5), 60, RED)
        line1 =TextButton(self.screen, "2 nguoi choi thay phien nhau chon 1", (W/2, H/3), 30,
                                                    WHITE)
        line2 =TextButton(self.screen, " o trang tren board de chiem ong.", (W/2, H/3 +50), 30,
                                                    WHITE)
        line3 =TextButton(self.screen, "Nguoi choi nao tao duoc 1 duong lien ket", (W/2, H/3 +
                                                    100), 30, WHITE)
        line4 =TextButton(self.screen, "2 canh doi dien nhau co mau duong vien", (W/2, H/3 +150),
                                                    30, WHITE)
        line5 =TextButton(self.screen, "ung voi mau cua nguoi choi o se la nguoi chien thang",
                                                    (W/2, H/3 +200), 30, WHITE)
        back =TextButton(self.screen, "Quay lai", (W/8, H -H/15), 60, WHITE)
        text =[line1, line2, line3, line4, line5]
        while rule:
            self.clock.tick(FPS)
            self.screen.fill(WHITE)
            self.screen.blit(self.background, (0, 0))
            for event in pg.event.get():
                if event.type ==QUIT:
                    pg.quit()
                    sys.exit()
                if event.type ==pg.MOUSEBUTTONDOWN:
                    if back.click():
                        pg.mixer.Sound.play(self.clickSound)
                        self.started =False
                        rule =False
            for t in text:
                t.printText(30)
            title.printText(80)
            back.render()
            pg.display.flip()
```

+Màn hình luật chơi: Khởi động khi click vào nút “Luật chơi” ở màn hình bắt đầu. Click vào "Quay lại" sẽ quay lại màn hình bắt đầu

```
def playScreen(self):
        """ Mn   hnh   chi """
        play =True
        connect =False
        while play:
            self.clock.tick(FPS)
            self.screen.fill(WHITE)
            self.screen.blit(self.background, (0, 0))
            for event in pg.event.get():
                if event.type ==QUIT:
                    self.board.quit_game()
```

```
                sleep(0.1)
                pg.quit()
                sys.exit()
            if event.type ==pg.MOUSEBUTTONDOWN:
                self.board.capture(self.clickSound)
        if(connect ==False):
            self.waitScreen()
            time.sleep(1)
            red =PlayerTag(self.screen, self.board.player1 +" TURN", (W/10, H/15), 30, RED)
            blue =PlayerTag(self.screen, self.board.player2 +" TURN", (W/10, H/15), 30, BLUE)
            connect =True
        self.board.showBoard()
        if (self.board.turn ==True and self.board.PLAYER ==1) or (self.board.turn ==False and
                                                                  self.board.PLAYER ==2):
           red.render()
        else:
           blue.render()
        if  self.board.checkWin() ==1:
            self.winScreen(str(self.board.player1) +" win", RED)
            play =False
        if  self.board.checkWin() ==2:
            self.winScreen(str(self.board.player2) +" win", BLUE)
            play =False
        if (self.board.interupt ==True):
            self.interuptingScreen()
            play =False
        pg.display.flip()
```

+Màn hình chơi: Khởi động khi click “Bắt đầu chơi” ở màn hình bắt đầu. · Khi chưa kết nối sẽ chuyển qua màn hình chờ và gọi hàm tạo giao diện phòng chat · Sau khi click "Bắt đầu chơi" ở giao diện phòng chat, màn hình sẽ hiển thị giao diện chơi · Khi 1 người chơi thoát giữa ván đấu thì người chơi còn lại sẽ chuyển sang màn hình gián đoạn · Sau mỗi lần chiến đóng, hàm checkWin() của bảng sẽ được gọi để kiểm tra người chiến thắng. Giao diện sẽ chuyển sang màn hình chiến thắng của người chơi đạt điều kiện chiến thắng

```
def shadow(self):
       shadow =pg.Surface((W, H))
       shadow.set_alpha(200)
       self.screen.blit(shadow, (0, 0))
```

+Hàm đổ bóng cho màn hình

```
def waitScreen(self):
       pause =True
       cont =TextButton(self.screen, "Dang doi bat dau...", (W/2, H/2), 80, GOLD)
       buttons =[cont]
       while pause:
           for event in pg.event.get():
               if event.type ==QUIT:
                   pg.quit()
                   sys.exit()
               if event.type ==pg.MOUSEBUTTONDOWN:
                   if cont.click():
                       pg.mixer.Sound.play(self.clickSound)
                       self.screen.fill(WHITE)
                       self.screen.blit(self.background, (0, 0))
                       return True
           self.screen.blit(self.background, (0, 0))
           self.shadow()
```

```
            for b in buttons:
                b.render()
            pg.display.flip()
            if( self.board.waiting_connection() ==True):
                self.screen.fill(WHITE)
                self.screen.blit(self.background, (0, 0))
                return True
```

+Màn hình đợi: Khởi động khi người chơi chưa được kết nối đến server. Màn hình này khởi động cùng lúc vs giao diện phòng chat. Khi rời khỏi phòng chat thì sẽ quay lại màn hình chơi

```
def interuptingScreen(self):
        pause =True
        if(self.type =="client1"):
            cont =TextButton(self.screen, self.board.player2 +" da thoat", (W/2, H/2), 80, GOLD)
        else:
            cont =TextButton(self.screen, self.board.player1 +" da thoat", (W/2, H/2), 80, GOLD)
        back =TextButton(self.screen, "Quay lai", (W/8, H -H/15), 60, WHITE)
        buttons =[cont, back]
        while pause:
            for event in pg.event.get():
                if event.type ==QUIT:
                    pg.quit()
                    sys.exit()
                if event.type ==pg.MOUSEBUTTONDOWN:
                    if back.click():
                        pg.mixer.Sound.play(self.clickSound)
                        self.started =False
                        self.board.resetBoard()
                        return True
            self.screen.blit(self.background, (0, 0))
            self.shadow()
            for b in buttons:
                b.render()
            pg.display.flip()
```

+Màn hình gián đoạn: Khởi động khi có người chơi thoát giữa ván đấu. Màn hình thông báo nickname người chơi đã thoát cho người chơi còn lại

```
def winScreen(self, txt, color):
        win =True
        winner =TextButton(self.screen, txt, (W/2, H/10), 80, color)
        back =TextButton(self.screen, "Quay lai", (W/8, H -H/15), 60, WHITE)
        while win:
            self.clock.tick(FPS)
            self.screen.fill(WHITE)
            self.screen.blit(self.background, (0, 0))
            self.board.showBoard()
            self.shadow()
            for event in pg.event.get():
                if event.type ==QUIT:
                    pg.quit()
                    sys.exit()
                if event.type ==pg.MOUSEBUTTONDOWN:
                    if back.click():
                        pg.mixer.Sound.play(self.clickSound)
                        self.started =False
                        win =False
                        self.board.resetBoard()
                        return True
```

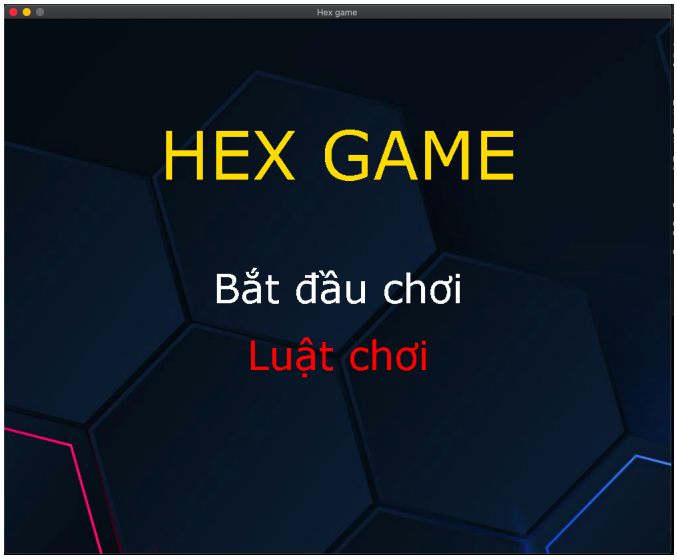
```
            winner.printText(80)
            back.render()
            pg.display.flip()
```

+Màn hình chiến thắng: Khởi động khi có người chơi thoả điều kiện chiến thắng **client1/client2.py**

```
import pygame as pg
from Game import *

game =Game("client1") /Game("client2")
while True:
    if not game.started:
        game.startScreen()
    else:
        game.playScreen()

    pg.display.flip()
```

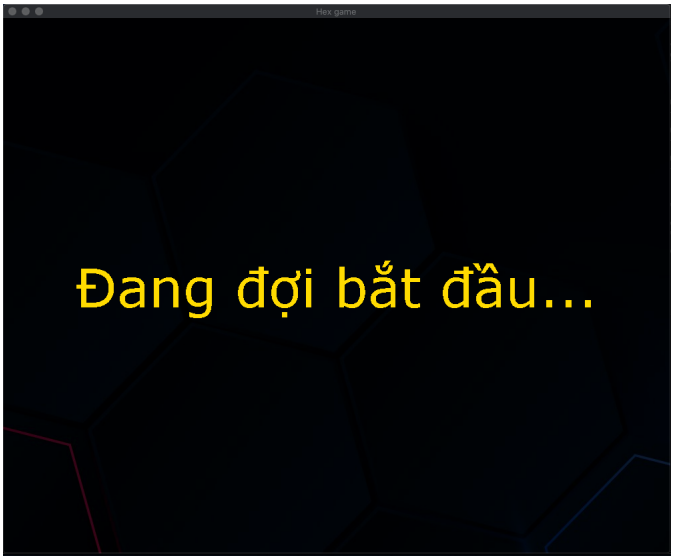
+Class khởi động client của trò chơi. Tham số truyền vào đối tượng Game sẽ quyết định thứ tự màu của client đó

3.3.5 Giao diện

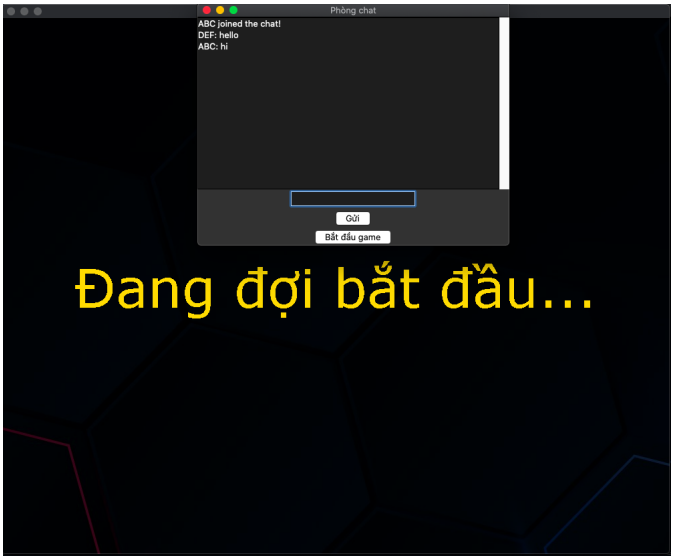


Hình 17: Màn hình bắt đầu

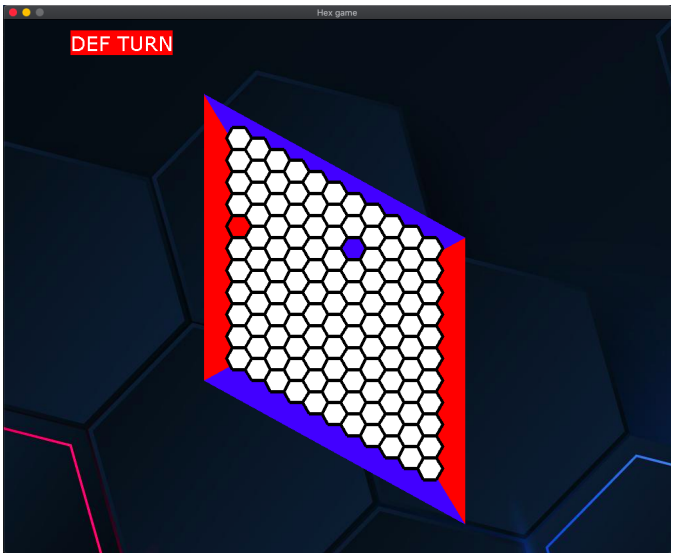
Hình 18: Màn hình luật chơi



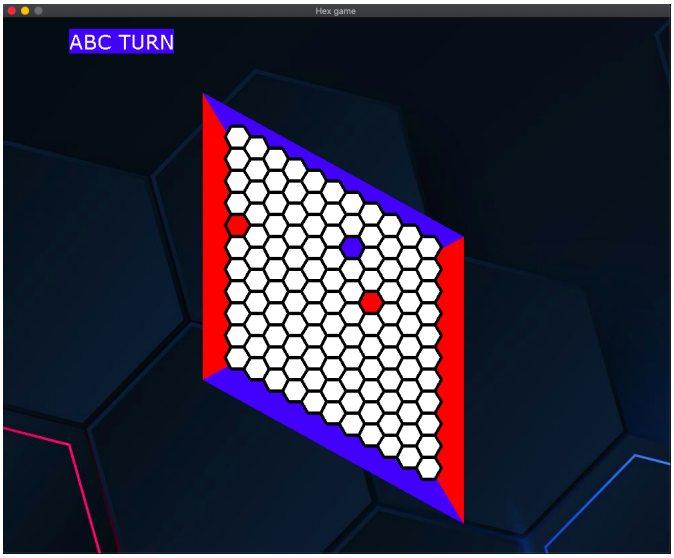
Hình 19: Màn hình đợi



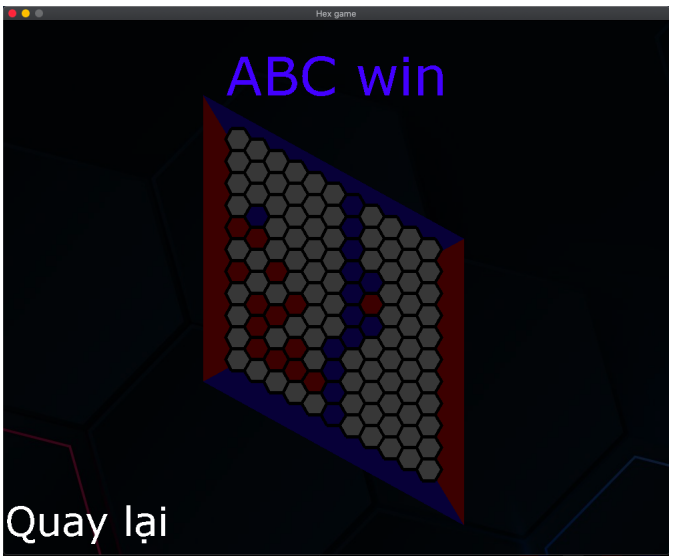
Hình 20: Giao diện phòng chat



Hình 21: Màn hình chơi 1



Hình 22: Màn hình chơi 2



Hình 23: Màn chiến thắng

Hình 24: Màn hình gián đoạn

4 KẾT LUẬN

4.1 Đánh giá

-Ưu điểm:
+Có đầy đủ các hoạt động cơ bản của trò chơi
+Giao diện đơn giản và dễ hiểu
+Dung lượng cài đặt chương trình nhỏ
-Nhược điểm
+Giao diện người dùng còn sơ sài
+Hiệu năng có thể bị giới hạn
+Chưa có hỗ trợ chơi với máy
+Phải thực hiện đúng trình tự các bước trong hướng dẫn mới có thể chơi. Thực hiện sai có thể dẫn đến đóng băng ứng dụng
+Còn nhiều lỗi chưa được sửa chữa

4.2 Hướng phát triển

-Thêm hỗ trợ chơi với máy
-Mở rộng thêm tính năng tuỳ chỉnh số ô chơi
-Cải thiện giao diện
-Thêm nhạc nền và các hình ảnh giúp game sinh động hơn
+Đơn giản hoá các bước setup

5 TÀI LIỆU THAM KHẢO

1.Hex (board game) - Wikipedia
2.Python (ngôn ngữ lập trình) – Wikipedia tiếng Việt
3.https://codelearn.io/sharing/lap-trinh-game-co-ban-voi-pygame
4.https://github.com/ANDREYDEN/Hex-Game
5.https://codelearn.io/sharing/lap-trinh-socket-voi-tcpip-trong-python
6.Slide bài giảng của thầy Từ Lãng Phiêu