TRƯỜNG ĐẠI HỌC SÀI GÒN
KHOA CÔNG NGHỆ THÔNG TIN

PHÁT TRIỂN PHẦN MỀM MÃ NGUỒN MỞ

Đề tài:

Phát triển game Ultimate Tic Tac Toe dùng được cho 2 người chơi

GVHD: Từ Lãng Phiêu
SV: Lại Quang Hải - 3120410150
Email: hainoob2002@gmail.com
SV: Dương Văn Trí - 3120410548
Email: id.vantri911@gmail.com
Nhóm: 1

TP. HỒ CHÍ MINH, THÁNG 5/2024

Mục lục

1 Giới thiệu

"Ultimate Tic Tac Toe" là một trò chơi xoắn não kết hợp giữa Tic Tac Toe truyền thống và chiến lược. Trò chơi này mang lại một trải nghiệm đầy thách thức và sự sáng tạo khi người chơi phải nghĩ cách để chiến thắng trên bàn cờ lớn hơn, gồm 9 bàn cờ nhỏ.

Mỗi bàn cờ nhỏ là một trò chơi Tic Tac Toe thông thường, nhưng thay vì chiến thắng một bàn cờ duy nhất, người chơi cần chiến thắng trên bàn cờ lớn hơn bằng cách chiến thắng các bàn cờ nhỏ. Các nước đi của người chơi xác định bàn cờ nhỏ tiếp theo mà đối thủ phải đánh vào.

Ultimate Tic Tac Toe yêu cầu người chơi phải có chiến thuật cẩn trọng và suy luận để đánh bại đối thủ. Nó không chỉ là một thử thách cho trí não mà còn là một trò chơi để tận hưởng cùng gia đình và bạn bè.

Với cách chơi đơn giản nhưng độ phức tạp cao, Ultimate Tic Tac Toe là lựa chọn lý tưởng cho những ai đam mê trò chơi logic và muốn thử sức mình trong một cuộc thi trí tuệ.

2 Cơ sở lý thuyết

2.1 Ngôn ngữ Python

Python là ngôn ngữ lập trình máy tính bậc cao thường được sử dụng để xây dựng trang web và phần mềm, tự động hóa các tác vụ và tiến hành phân tích dữ liệu. Python là ngôn ngữ có mục đích chung, nghĩa là nó có thể được sử dụng để tạo nhiều chương trình khác nhau và không chuyên biệt cho bất kỳ vấn đề cụ thể nào.

Tính linh hoạt này, cùng với sự thân thiện với người mới bắt đầu, đã khiến nó trở thành một trong những ngôn ngữ lập trình được sử dụng nhiều nhất hiện nay. Một cuộc khảo sát được thực hiện bởi công ty phân tích ngành RedMonk cho thấy rằng đây là ngôn ngữ lập trình phổ biến thứ hai đối với các nhà phát triển vào năm 2021.

- Python được phát triển vào cuối những năm 1980 bởi Guido van Rossum tại Viện Nghiên cứu Quốc gia về Toán học và Khoa học Máy tính ở Hà Lan với tư cách là người kế thừa ngôn ngữ ABC có khả năng xử lý và giao tiếp ngoại lệ.
- Python có nguồn gốc từ các ngôn ngữ lập trình như ABC, Modula 3, small talk, Algol-68.
- Van Rossum đã chọn tên Python cho ngôn ngữ mới từ một chương trình truyền hình, Monty Python's Flying Circus.
- Trang Python là một tệp có phần mở rộng .py chứa có thể là sự kết hợp của Thẻ HTML và tập lệnh Python.
- Vào tháng 12 năm 1989, người sáng tạo đã phát triển trình thông dịch python đầu tiên như một sở thích, và sau đó vào ngày 16 tháng 10 năm 2000, Python 2.0 được phát hành với nhiều tính năng mới.
- Python là mã nguồn mở, có nghĩa là bất kỳ ai cũng có thể tải xuống miễn phí từ trang chủ và sử dụng nó để phát triển các chương trình. Mã nguồn của nó có thể được truy cập và sửa đổi theo yêu cầu trong dự án.
- Python có thể được sử dụng trên nhiều hệ điều hành máy tính khác nhau, chẳng hạn như Windows, macOS, Linux và Unix.

Ứng dụng của Python:

- Python thường được sử dụng để phát triển trang web và phần mềm, tự động hóa tác vụ, phân tích dữ liệu và trực quan hóa dữ liệu. Vì tương đối dễ học, Python đã được nhiều người không phải là lập trình viên như kế toán và nhà khoa học áp dụng cho nhiều công việc hàng ngày, chẳng hạn như tổ chức tài chính.
- Python đã trở thành một yếu tố chính trong khoa học dữ liệu, cho phép các nhà phân tích dữ liệu và các chuyên gia khác sử dụng ngôn ngữ này để thực hiện các phép tính thống kê phức tạp, tạo trực quan hóa dữ liệu, xây dựng thuật toán học máy, thao tác và phân tích dữ liệu cũng như hoàn thành các nhiệm vụ khác liên quan đến dữ liệu.
- Python thậm chí có thể được sử dụng bởi những người mới bắt đầu để tự động hóa các tác vụ đơn giản trên máy tính—chẳng hạn như đổi tên tệp, tìm và tải xuống nội dung trực tuyến hoặc gửi email hoặc văn bản theo khoảng thời gian mong muốn. Trong phát triển phần mềm, Python có thể hỗ trợ các tác vụ như kiểm soát bản dựng, theo dõi lỗi và thử nghiệm. Với Python, các nhà phát triển phần mềm có thể tự động kiểm tra các sản phẩm hoặc tính năng mới. Một số công cụ Python được sử dụng để kiểm thử phần mềm bao gồm Green và Requestium.

Đặc tính của Python: Python đang trở nên phổ biến trong cộng đồng lập trình nhờ có các đặc tính sau:

- Ngôn ngữ thông dịch: Python được xử lý trong thời gian chạy bởi Trình thông dịch Python.
- Ngôn ngữ hướng đối tượng: Nó hỗ trợ các tính năng và kỹ thuật lập trình hướng đối tượng.
- Ngôn ngữ dễ học: Python rất dễ học, đặc biệt là cho người mới bắt đầu.
- Cú pháp đơn giản: Việc hình thành cú pháp Python rất đơn giản và dễ hiểu, điều này cũng làm cho nó trở nên phổ biến.
- Dễ đọc: Mã nguồn Python được xác định rõ ràng và có thể nhìn thấy bằng mắt.
- Di động: Mã Python có thể chạy trên nhiều nền tảng phần cứng có cùng giao diện.
- Có thể cải tiến: Python cung cấp một cấu trúc cải tiến để hỗ trợ các chương trình lớn sau đó là shell-script.
- Có thể mở rộng: Người dùng có thể thêm các mô-đun cấp thấp vào trình thông dịch Python.

2.2 Các thư viện sử dụng

2.2.1 Pygame

Pygame là một thư viện mã nguồn mở được viết bằng Python, chuyên dụng cho việc phát triển trò chơi 2D và các ứng dụng đa phương tiện. Thư viện này cung cấp một bộ công cụ toàn diện cho phép lập trình viên tạo ra các trò chơi với đồ họa, âm thanh và hiệu ứng phong phú, mà không cần kiến thức chuyên sâu về đồ họa máy tính.

Ưu điểm của Pygame:

- Dễ sử dụng: Pygame có cú pháp đơn giản và dễ hiểu, ngay cả đối với những người mới bắt đầu lập trình.
- Miễn phí và mã nguồn mở: Pygame được phát hành theo giấy phép GNU Lesser General Public License, cho phép bạn sử dụng và sửa đổi nó miễn phí cho mục đích cá nhân hoặc thương mại.
- Cộng đồng lớn: Pygame có một cộng đồng người dùng và nhà phát triển lớn, luôn sẵn sàng hỗ trợ và chia sẻ kiến thức.
- Đa nền tảng: Pygame hoạt động trên nhiều hệ điều hành khác nhau, bao gồm Windows, macOS và Linux.

Tính năng chính của Pygame:

- Tạo cửa sổ và màn hình: Pygame cung cấp các hàm để tạo cửa sổ trò chơi với kích thước và độ phân giải tùy chỉnh.
- Vẽ đồ họa: Pygame hỗ trợ vẽ các hình dạng 2D cơ bản như đường thẳng, hình vuông, hình tròn, v.v. và các hình ảnh được tải từ tập tin.
- Xử lý sự kiện: Pygame cho phép bạn theo dõi các sự kiện người dùng như nhấp chuột, di chuyển chuột và nhấn phím.
- Phát âm thanh: Pygame hỗ trợ phát các tệp âm thanh định dạng WAV và MIDI.
- Quản lý thời gian: Pygame cung cấp các hàm để kiểm soát tốc độ khung hình và cập nhật trạng thái trò chơi theo thời gian.

Một số ví dụ về ứng dụng của Pygame:

- Trò chơi 2D: Pygame được sử dụng để phát triển nhiều loại trò chơi 2D khác nhau, từ các trò chơi đơn giản như Cờ Caro đến các trò chơi phức tạp hơn như game bắn súng và game nhập vai.
- Ứng dụng đa phương tiện: Pygame có thể được sử dụng để tạo các ứng dụng đa phương tiện như trình phát nhạc, trình chiếu ảnh và trình tạo hoạt hình.
- Giáo dục: Pygame có thể được sử dụng để tạo các ứng dụng giáo dục tương tác giúp học sinh học tập hiệu quả hơn.
- Nghiên cứu khoa học: Pygame có thể được sử dụng để tạo các mô phỏng và trực quan hóa dữ liệu cho mục đích nghiên cứu khoa học.

2.2.2 Socket

Thư viện Socket trong Python là một công cụ mạnh mẽ cho phép lập trình viên giao tiếp với các socket mạng. Socket là những điểm cuối giao tiếp cho phép dữ liệu được truyền tải giữa các ứng dụng chạy trên các máy khác nhau. **Lợi ích sử dụng thư viện Socket:**

- Phát triển ứng dụng mạng: Socket giúp bạn tạo ra các ứng dụng mạng như client-server, chat, chia sẻ tệp, v.v.
- Tương tác với các thiết bị ngoại vi: Socket cho phép bạn giao tiếp với các thiết bị ngoại vi như webcam, cảm biến, v.v.
- Tự động hóa tác vụ: Socket có thể được sử dụng để tự động hóa các tác vụ liên quan đến mạng, ví dụ như tải xuống tệp, kiểm tra email, v.v.

Tính năng chính của thư viện Socket:

- Tạo socket: Socket được tạo bằng cách sử dụng hàm socket.socket().
- Kết nối socket: Socket được kết nối với nhau bằng cách sử dụng hàm socket.connect() hoặc socket.bind() và socket.listen().
- Gửi và nhận dữ liệu: Dữ liệu được gửi và nhận giữa các socket bằng cách sử dụng các hàm socket.send() và socket.recv().
- Đóng socket: Socket được đóng bằng cách sử dụng hàm socket.close().

2.2.3 Threading

Thư viện Threading là một công cụ mạnh mẽ trong Python giúp bạn viết mã đa luồng (multithreading). Đa luồng cho phép chương trình thực thi nhiều tác vụ song song, nâng cao hiệu suất và khả năng phản hồi của ứng dụng. **Lợi ích sử dụng thư viện Threading:**

- Tăng tốc độ: Đa luồng giúp chia nhỏ các tác vụ phức tạp thành các luồng nhỏ hơn, có thể được thực thi đồng thời trên nhiều CPU, giúp tăng tốc độ xử lý.
- Cải thiện khả năng phản hồi: Khi một luồng bị chặn (ví dụ: chờ đợi phản hồi từ người dùng hoặc dữ liệu từ mạng), các luồng khác vẫn có thể tiếp tục thực thi, giúp ứng dụng luôn phản hồi nhanh với người dùng.
- Giảm thiểu thời gian chờ: Đa luồng giúp che lấp thời gian chờ, ví dụ như khi chờ tải dữ liệu từ mạng, bằng cách cho phép thực thi các tác vụ khác trong khi chờ đợi.

Tính năng chính của thư viện Threading:

- Tạo luồng: Luồng được tạo bằng cách sử dụng lớp Thread và phương thức start().
- Quản lý luồng: Thư viện Threading cung cấp các hàm để khởi động, tạm dừng, tiếp tục và kết thúc luồng.
- Đồng bộ hóa luồng: Đa luồng có thể dẫn đến các vấn đề về đồng bộ hóa nếu nhiều luồng truy cập cùng một tài nguyên cùng lúc. Thư viện Threading cung cấp các cơ chế đồng bộ hóa như khóa (lock), semaphore và event để giải quyết các vấn đề này.

3 Thiết kế ứng dụng

3.1 Yêu cầu chức năng

Giống như Cờ Caro thông thường, Ultimate Tic-Tac-Toe là trò chơi dành cho 2 người chơi (X và O), chơi theo lượt, bắt đầu với người chơi X.
Trò chơi bắt đầu như thế nào?

- Người chơi X được đánh dấu X của mình vào bất kỳ ô trống nào trong tổng số 81 ô trống trên bàn cờ lớn.
- Lượt tiếp theo, đối thủ (O) bị buộc phải đánh dấu vào ô trống trong bàn cờ nhỏ tương ứng với vị trí mà X vừa đánh dấu trên bàn cờ lớn. Ví dụ: nếu X đánh dấu vào ô trống phía trên bên phải của một bàn cờ nhỏ (3x3), thì O phải đánh dấu vào bàn cờ nhỏ nằm ở phía trên bên phải của bàn cờ lớn.
- Người chơi đánh dấu vào bất kỳ ô trống nào trong bàn cờ nhỏ được chọn sẽ xác định bàn cờ nhỏ nào mà người chơi kia phải đánh dấu vào lượt tiếp theo.

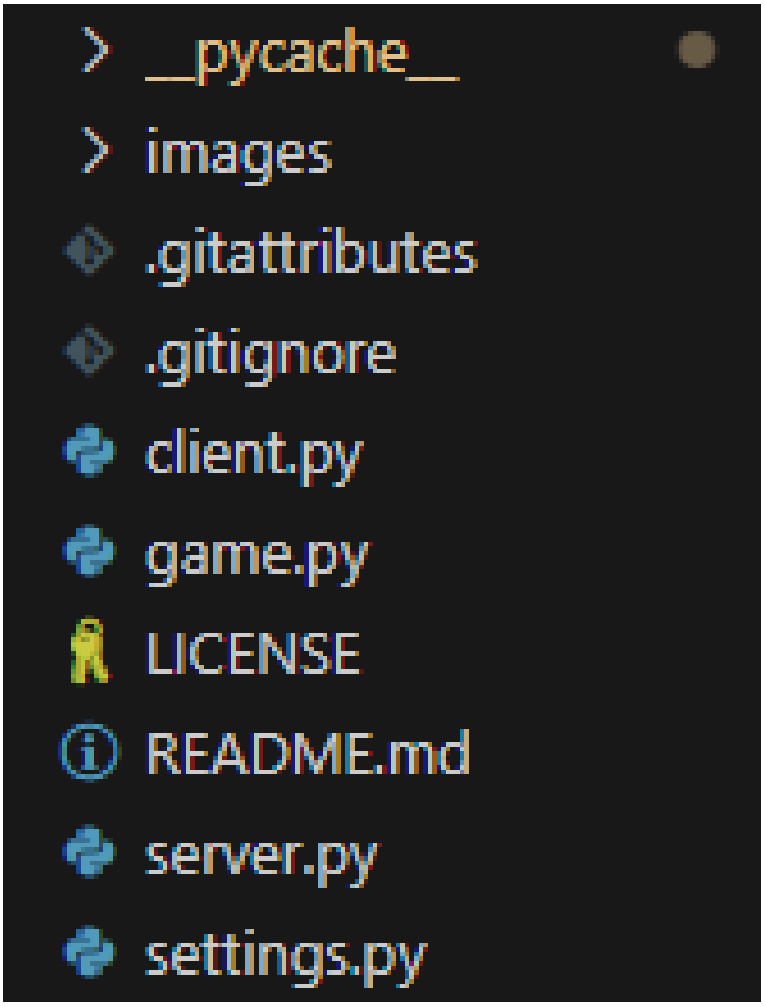
Chiến thắng một ô nhỏ:

- Nếu một nước đi tạo thành một hàng ngang, dọc hoặc chéo gồm 3 ký hiệu giống nhau theo luật chơi của Cờ Caro thông thường trong một bàn cờ nhỏ, thì toàn bộ bàn cờ nhỏ đó được đánh dấu là đã thắng bởi người chơi đó trên bàn cờ lớn.
- Một khi một bàn cờ nhỏ đã được một người chơi thắng hoặc được đánh dấu đầy đủ, thì không thể đánh thêm bất kỳ nước đi nào vào bàn cờ đó nữa.

Nếu bị "gửi" đến một bàn cờ đã thắng hoặc đầy:
Nếu một người chơi bị "gửi" đến một bàn cờ nhỏ đã được thắng hoặc đầy đủ, thì người chơi đó có thể đánh dấu vào bất kỳ bàn cờ nhỏ nào khác trên bàn cờ lớn.

Kết thúc trò chơi:
Trò chơi kết thúc khi một trong hai người chơi thắng trên bàn cờ lớn (bằng cách tạo thành một hàng ngang, dọc hoặc chéo gồm 3 bàn cờ nhỏ đã thắng của mình) hoặc không còn nước đi hợp lệ nào còn lại. Trong trường hợp không còn nước đi hợp lệ, trò chơi được tuyên bố là hòa.

3.2 Cấu trúc mã nguồn



Mã nguồn gồm:

- client.py: Hiển thị giao diện cho client để kết nối vào server và gửi thông tin cho server
- game.py: Test giao diện và cơ chế game
- server.py: Tạo socket, nhận thông tin từ phía client và xử lý cơ chế trò chơi
- settings.py: Lưu trữ biến cái đặt của trò chơi
- README.md: Giới thiệu về dự án, cách cài đặt chương trình.

4 Hiện thực

4.1 Mã nguồn các chức năng

- Tạo giao diện cho trò chơi

```
def drawGrid():
    screen.fill(backgroundColor)
    for i in range(9):
        if i != 0:
            pygame.draw.line(screen,smallLineColor,(i*80,0),(i*80,720),2)
            pygame.draw.line(screen,smallLineColor,(0,i*80),(720,i*80),2)
            if i%3 == 0:
                pygame.draw.line(screen,lineColor,(i*80,0),(i*80,720),4)
                pygame.draw.line(screen,lineColor,(0,i*80),(720,i*80),4)
    pygame.draw.line(screen, lineColor, (720, 0), (720, 720), 4)

    pygame.draw.rect(screen, (0, 255, 0), (SCREEN_WIDTH -
        (SCREEN_WIDTH-SCREEN_HEIGHT) + 100, 150, 180, 50))

    restart_button = pygame.font.Font('freesansbold.ttf', 24).render("Restart
        Game", True, (0, 0, 0))
    screen.blit(restart_button, (SCREEN_WIDTH - (SCREEN_WIDTH-SCREEN_HEIGHT) + 110,
        165))
    title = pygame.font.Font('freesansbold.ttf', 52).render("TIC TAC TOE", True,
        titleColor)
    screen.blit(title, (SCREEN_WIDTH - (SCREEN_WIDTH-SCREEN_HEIGHT) + 20, 10))

    pygame.draw.line(screen, lineColor,(720, 240),(SCREEN_WIDTH,240) , 4)
    chat_box=pygame.font.Font('freesansbold.ttf', 24).render("Chat Box", True,
        titleColor)
    screen.blit(chat_box, (SCREEN_WIDTH - (SCREEN_WIDTH-SCREEN_HEIGHT) + 120, 250))

    WTF=pygame.font.Font('freesansbold.ttf', 24).render("Player: " +
        str(current_player), True, titleColor)
    screen.blit(WTF, (SCREEN_WIDTH - (SCREEN_WIDTH-SCREEN_HEIGHT) + 120, 275))

    input_box=pygame.Rect(750, SCREEN_HEIGHT-50, 300, 40)
    pygame.draw.rect(screen, (0, 0, 0), input_box, 2)
    txt_surface=pygame.font.Font('freesansbold.ttf', 24).render(msg, True,
        titleColor)
    screen.blit(txt_surface, (input_box.x+5, input_box.y+5))

    y_offset = 0
    for message in conversation_messages:
        conversation_text = pygame.font.Font('freesansbold.ttf', 24).render(message,
            True, titleColor)
        screen.blit(conversation_text, (SCREEN_WIDTH - (SCREEN_WIDTH-SCREEN_HEIGHT)
            + 20, 320 + y_offset))
        y_offset += 30

    if(player==1):
```

```
38          centerMessage("X's Turn",XCOLOR)
39      else:
40          centerMessage("O's Turn",OCOLOR)
41
42  def drawMarker():
43      global nextMove,openMove
44      x_pos = 0
45      for i in matrix:
46          y_pos = 0
47          for j in i:
48              if j == 1:
49                  pygame.draw.line(screen,XCOLOR,(x_pos*80+15,y_pos*80+15),
50                  (x_pos*80+65,y_pos*80+65),5)
51                  pygame.draw.line(screen,XCOLOR,(x_pos*80+15,y_pos*80+65),
52                  (x_pos*80+65,y_pos*80+15),5)
53              if j == -1:
54                  pygame.draw.circle(screen,OCOLOR,(x_pos*80+40,y_pos*80+40),30,5)
55              y_pos+=1
56          x_pos+=1
57      x = 0
58      for i in winnerMatrix:
59          y = 0
60          for j in i:
61              if j == 1:
62                  pygame.draw.line(screen,XCOLOR,
63                  (x*240+15,y*240+15),(x*240+225,y*240+225),15)
64                  pygame.draw.line(screen,XCOLOR,
65                  (x*240+225,y*240+15),(x*240+15,y*240+225),15)
66              if j == -1:
67                  pygame.draw.circle(screen,OCOLOR,(x*240+120,y*240+120),100,15)
68              y+=1
69          x+=1
70      if openMove == False:
71          x= nextMove[0]
72          y= nextMove[1]
73          pygame.draw.line(screen,OCOLOR,(x*80,y*80),((x+3)*80,y*80),5)
74          pygame.draw.line(screen,OCOLOR,(x*80,(y+3)*80),((x+3)*80,(y+3)*80),5)
75          pygame.draw.line(screen,OCOLOR,(x*80,y*80),(x*80,(y+3)*80),5)
76          pygame.draw.line(screen,OCOLOR,((x+3)*80,y*80),((x+3)*80,(y+3)*80),5)
77      else:
78          for i in range(4):
79              pygame.draw.line(screen,OCOLOR,(i*3*80,0),(i*3*80,720),5)
80              pygame.draw.line(screen,OCOLOR,(0,i*3*80),(720,i*3*80),5)
```

- Tạo chức năng cho giao diện của client

```
def create_thread():
    t=threading.Thread(target=receive_message)
    t.daemon=True
    t.start()

run = True
while run:

    create_thread()
    drawGrid()
    drawMarker()

    #add event handler
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            run = False
        if event.type == pygame.MOUSEBUTTONDOWN and clicked == False:
            clicked = True

        if event.type== pygame.KEYDOWN:
            if event.key == pygame.K_RETURN:
                print(msg)
                socket.send(str.encode("Player " + str(current_player) + ": " +
                    msg,"utf-8"))
                msg = ""
            elif event.key == pygame.K_BACKSPACE:
                msg = msg[:-1]
            else:
                msg += event.unicode

        if event.type == pygame.MOUSEBUTTONUP and clicked == True:
            clicked = False
            pos = pygame.mouse.get_pos()
            mouse_x = pos[0]
            mouse_y = pos[1]
            if pos >= (0,0) and pos <= (720,720):
                socket.send(str.encode(
                str(current_player)+","+str(mouse_x)+","+str(mouse_y),"utf-8"))

            if mouse_x > SCREEN_WIDTH - (SCREEN_WIDTH-SCREEN_HEIGHT) + 100 and
                mouse_x < SCREEN_WIDTH - (SCREEN_WIDTH-SCREEN_HEIGHT) + 280 and
                mouse_y > 150 and mouse_y < 200:
                print("Send restart message to server")
                socket.send(str.encode("restart","utf-8"))

    pygame.display.update()

pygame.quit()
```

- Xử lý thông tin nhận được từ phía server

```
def receive_message():
    global gameOver
    global matrix
    global winnerMatrix
    global player
    global lastMove
    global openMove
    global nextMove
    global current_player
    global conversation_messages
    while True:
        try:
            data = socket.recv(2048*10).decode('utf-8')
            if data[:3] == "Wel" and current_player == None:
                current_player = int(data[-1])
                current_player = 1 if current_player == 1 else -1
                print("Set Current_player = ", current_player)

            elif data=="Open move: False":
                openMove = False
            elif data=="Open move: True":
                openMove = True

            elif data.count(",")==1:
                matrixRevc = list(map(int, data.split(",")))
                nextMove = matrixRevc

            elif data.count(",")==80:
                matrixRevc = list(map(int, data.split(",")))
                matrix = convert_1d_to_2d(matrixRevc,9)

            elif data.count(",")==8:
                matrixRevc = list(map(int, data.split(",")))
                winnerMatrix = convert_1d_to_2d(matrixRevc,3)

            elif data == "Game has been restarted!":
                matrix = [[0 for _ in range(9)] for _ in range(9)]
                winnerMatrix = [[0 for _ in range(3)] for _ in range(3)]
                player = 1
                gameOver = False
                lastMove = None
                openMove = True
                nextMove = [-1,-1]

            elif data == "It's a tie!":
                centerMessage("It's a tie!")

            elif data == "Player 1 wins!":
                centerMessage("Player 1 wins!")

```

```
            elif data == "Player -1 wins!":
                centerMessage("Player -1 wins!")

            elif data == "1" or data == "-1":
                player = int(data)
                print("Player = ",player)

            else:
                conversation_messages.append(data)
                conversation_messages = conversation_messages[-10:]
                print("Received: ",data)

        except Exception as e:
            print(f"Exception occurred: {e}")
            break
```

- Khởi động server và nhận thông tin từ client

```
def start_server():
    global current_player_count
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind((host, port))
    server.listen(2)
    print("Server started")
    try:
        while True:
            conn, addr = server.accept()
            clients.append(conn)
            print(f"Connection from {addr} has been established!")
            current_player_count += 1

            conn.send(bytes("Welcome! " + str(current_player_count), "utf-8"))

            threading.Thread(target=handle_client, args=(conn, addr)).start()
        #    handle_client(conn, addr)
    except KeyboardInterrupt:
        print("Server shutting down...")
        server.close()

def handle_client(conn, addr):
    global matrix, winnerMatrix, player, gameOver, lastMove, openMove, nextMove
    global current_turn
    global winner
    global conversation_messages

    player = 1
    gameOver = False
    while True:
        data = conn.recv(2048*10).decode("utf-8")
        if not data:
            break
        print(f"Received: {data}")

        if data == "restart":
            matrix = [[0 for _ in range(9)] for _ in range(9)]
            winnerMatrix = [[0 for _ in range(3)] for _ in range(3)]
            player = 1
            current_turn = 1
            gameOver = False
            lastMove = None
            openMove = True
            nextMove = [-1,-1]
            send_msg("Game has been restarted!")

        elif data.count(",") == 2:
            data = data.split(",")

            p = int(data[0])
```

```
            if p != current_turn:
                print("Illegal move from ", p)
                continue

            current_turn *= -1

            x = int(data[1])
            y = int(data[2])

            if gameOver == False:
                if openMove == True:
                    if matrix[x//80][y//80] == 0:
                        matrix[x//80][y//80] = player
                        player *= -1
                        lastMove = [x,y]
                        check_next_move(conn)

                elif openMove == False:
                    if matrix[x//80][y//80] == -2:
                        matrix[x//80][y//80] = player
                        player *= -1
                        lastMove = [x,y]
                        reset_move()
                        check_next_move(conn)

            send_msg(str(player))

            if winner == -2:
                send_msg("It's a tie!")

            if winner == 1:
                send_msg("Player 1 wins!")

            if winner == -1:
                send_msg("Player -1 wins!")

        else:
            conversation_messages.append(data)
            conversation_messages = conversation_messages[-10:]
            send_msg(data)

    conn.close()
```

- Kiểm tra điều kiện của trò chơi

```
def check_columns():
    # print("Checking columns")
    result = 0
    for i in range(9):
        if matrix[i][0] == matrix[i][1] == matrix[i][2]:
            if matrix[i][0] != 0 and matrix[i][0] != 2 and matrix[i][0] != -2:
                result = matrix[i][0]
                add_cell_winner(result,i,0)
        if matrix[i][3] == matrix[i][4] == matrix[i][5]:
            if matrix[i][3] != 0 and matrix[i][3] != 2 and matrix[i][3] != -2:
                result = matrix[i][3]
                add_cell_winner(result,i,3)
        if matrix[i][6] == matrix[i][7] == matrix[i][8]:
            if matrix[i][6] != 0 and matrix[i][6] != 2 and matrix[i][6] != -2:
                result = matrix[i][6]
                add_cell_winner(result,i,6)
    return result

def check_rows():
    # print("Checking rows")
    result = 0
    for i in range(9):
        if matrix[0][i] == matrix[1][i] == matrix[2][i]:
            if matrix[0][i] != 0 and matrix[0][i] != 2 and matrix[0][i] != -2:
                result = matrix[0][i]
                add_cell_winner(result,0,i)
        if matrix[3][i] == matrix[4][i] == matrix[5][i]:
            if matrix[3][i] != 0 and matrix[3][i] != 2 and matrix[3][i] != -2:
                result = matrix[3][i]
                add_cell_winner(result,3,i)
        if matrix[6][i] == matrix[7][i] == matrix[8][i]:
            if matrix[6][i] != 0 and matrix[6][i] != 2 and matrix[6][i] != -2:
                result = matrix[6][i]
                add_cell_winner(result,6,i)
    return result

def check_diagonals():
    # print("Checking diagonals")
    result = 0
    for x in [0,3,6,2,5,8]:
        if x%3==0:
            y=x+1
            z=x+2
        else:
            y=x-1
            z=x-2

        if matrix[x][0] == matrix[y][1] == matrix[z][2]:
            if matrix[x][0] != 0 and matrix[x][0] != 2 and matrix[x][0] != -2:
                result = matrix[x][0]
```

```
51                add_cell_winner(result,x,0)
52            if matrix[x][3] == matrix[y][4] == matrix[z][5]:
53                if matrix[x][3] != 0 and matrix[x][3] != 2 and matrix[x][3] != -2:
54                    result = matrix[x][3]
55                    add_cell_winner(result,x,3)
56            if matrix[x][6] == matrix[y][7] == matrix[z][8]:
57                if matrix[x][6] != 0 and matrix[x][6] != 2 and matrix[x][6] != -2:
58                    result = matrix[x][6]
59                    add_cell_winner(result,x,6)
60        return result
61
62    def add_cell_winner(player,x_pos,y_pos):
63        x = (x_pos//3)
64        y = (y_pos//3)
65        for i in range(3):
66            for j in range(3):
67                if matrix[x*3+i][y*3+j] == 0:
68                    matrix[x*3+i][y*3+j] = 2
69        winnerMatrix[x][y] = player
70        send_winnerMatrix()
71
72    def check_next_move(conn):
73        global lastMove,nextMove,openMove,player,gameOver,winner
74        check_columns()
75        check_rows()
76        check_diagonals()
77        winner = check_winner()
78        if winner == -2:
79            send_msg("Game over!")
80            send_msg("It's a tie!")
81            send_msg("Winner -2")
82            gameOver = True
83        if winner != 0:
84            send_msg("Game over!")
85            send_msg("Winner"+str(winner))
86            gameOver = True
87        else:
88            if lastMove != None:
89                x = lastMove[0]//80
90                y = lastMove[1]//80
91                nextMove[0] = (x-x//3*3)*3
92                nextMove[1] = (y-y//3*3)*3
93                if winnerMatrix[nextMove[0]//3][nextMove[1]//3] == 0:
94                    openMove = False
95                    count = 0
96                    for i in range(3):
97                        for j in range(3):
98                            if matrix[nextMove[0]+i][nextMove[1]+j] == 0:
99                                matrix[nextMove[0]+i][nextMove[1]+j] = -2
100                               count+=1
101                   if count == 0:
102                       nextMove = [-1,-1]
```

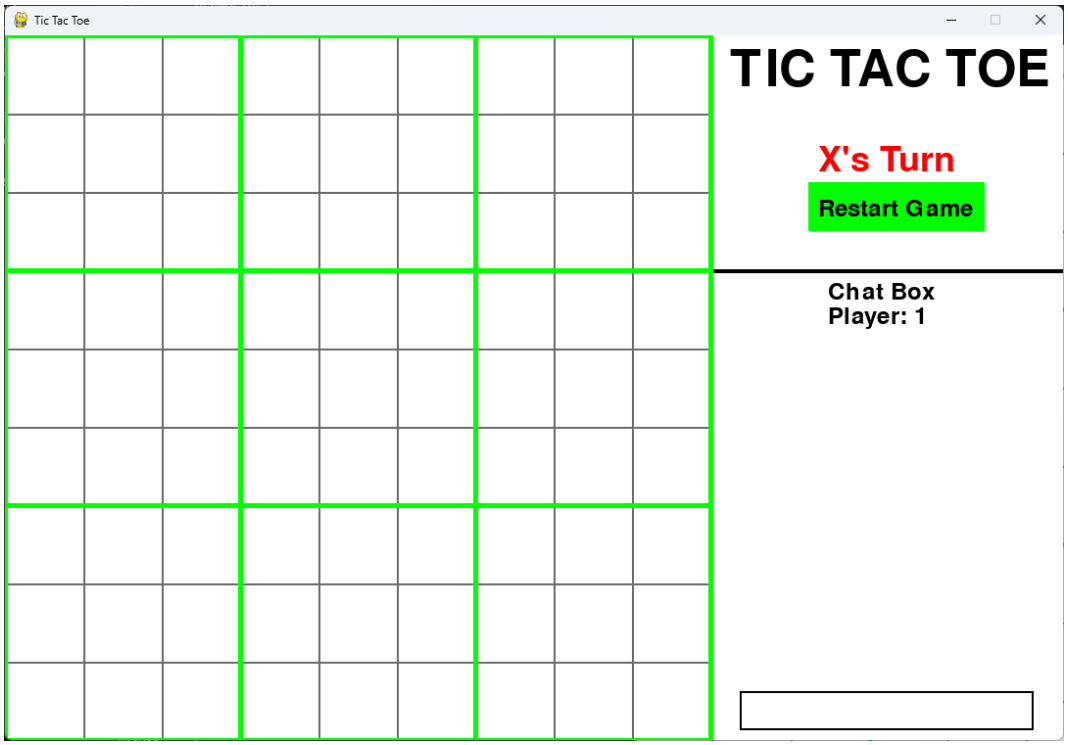
```
                    openMove = True
            else:
                nextMove = [-1,-1]
                openMove = True
        send_msg("Open move: "+str(openMove))
        send_msg(str(nextMove[0])+","+str(nextMove[1]))
        send_matrix()

def check_winner():
    winner = 0
    pos = 0
    for i in winnerMatrix:
        #check columns
        if sum(i) == 3:
            winner = 1
            return winner
        if sum(i) == -3:
            winner = -1
            return winner
        #check rows
        if winnerMatrix[0][pos] == winnerMatrix[1][pos] == winnerMatrix[2][pos]:
            if winnerMatrix[0][pos]!=0:
                winner = winnerMatrix[0][pos]
                return winner
        pos+=1
    #check diagonals
    if winnerMatrix[0][0] == winnerMatrix[1][1] == winnerMatrix[2][2]:
        if winnerMatrix[0][0] != 0:
            winner = winnerMatrix[0][0]
    if winnerMatrix[2][0] == winnerMatrix[1][1] == winnerMatrix[0][2]:
        if winnerMatrix[2][0] != 0:
            winner = winnerMatrix[2][0]
            return winner
    #check tie
    count = 0
    for i in range(3):
        for j in range(3):
            if winnerMatrix[i][j] == 0:
                count += 1
    if count == 0: winner = -2
    return winner

def reset_move():
    if nextMove != [-1,-1]:
        for i in range(3):
            for j in range(3):
                if matrix[nextMove[0]+i][nextMove[1]+j] == -2:
                    matrix[nextMove[0]+i][nextMove[1]+j] = 0
    send_matrix()
```

- Gửi thông tin về phía client

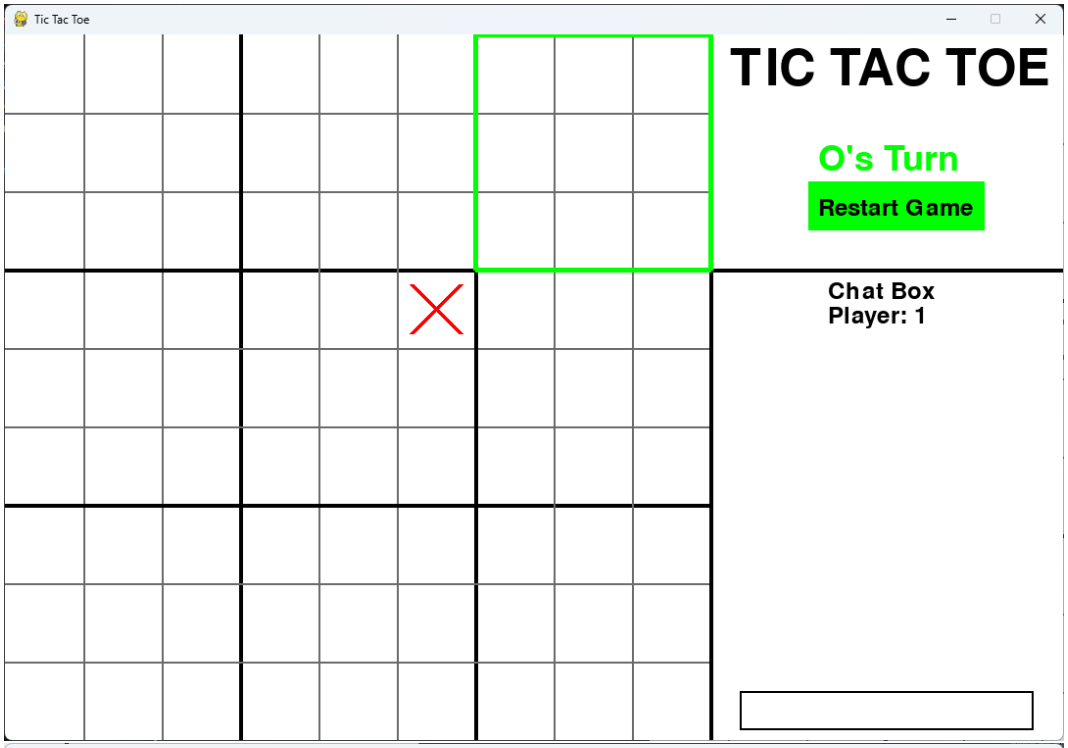
```
def send_msg(message):
    for client in clients:
        client.send(message.encode("utf-8"))

def send_matrix():
    global matrix
    string = ""
    for i in range(9):
        for j in range(9):
            if i== j == 8:
                string += str(matrix[i][j])
            else:
                string += str(matrix[i][j]) + ","
    send_msg(string)

def send_winnerMatrix():
    global winnerMatrix
    string = ""
    for i in range(3):
        for j in range(3):
            if i== j == 2:
                string += str(winnerMatrix[i][j])
            else:
                string += str(winnerMatrix[i][j]) + ","
    send_msg(string)
```

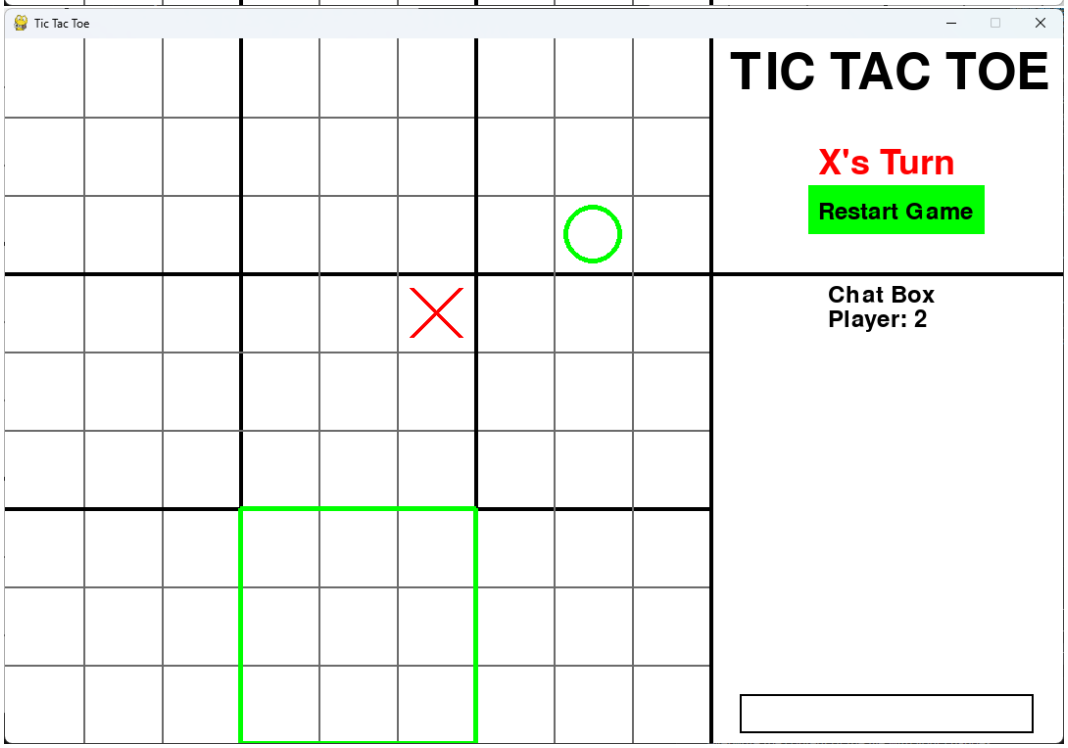
4.2 Giao diện ứng dụng

- Màn hình chính của ứng dụng:

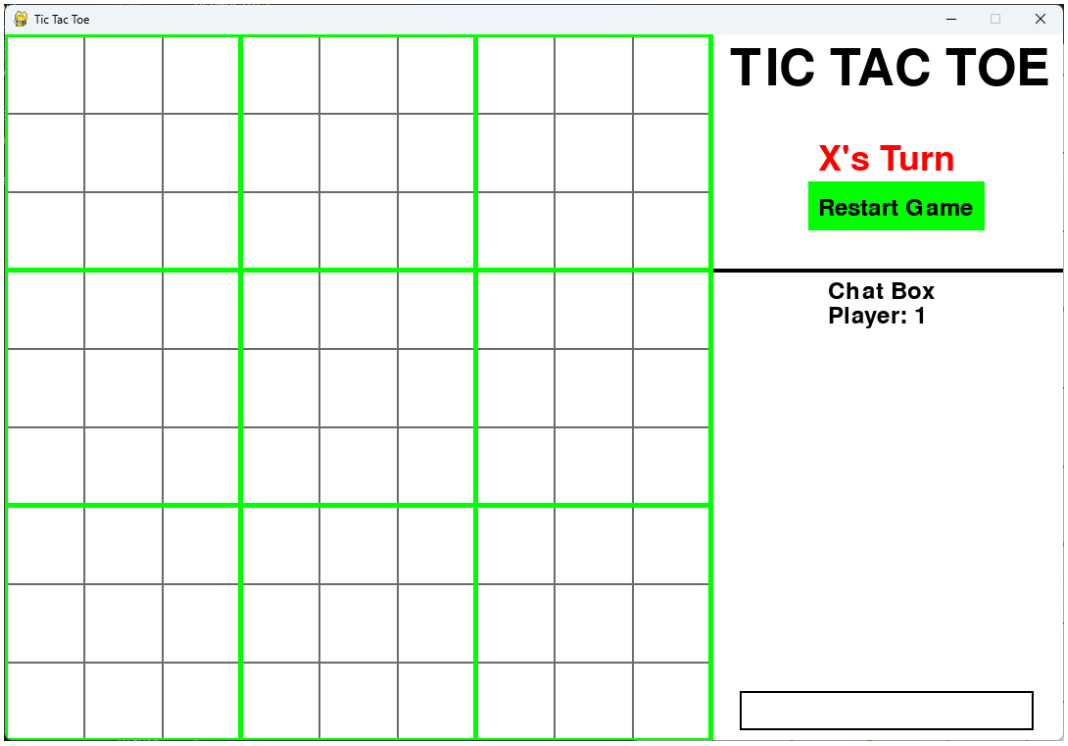


- Màn hình khi người chơi thực hiện bước đi và khóa nước đi tiếp theo

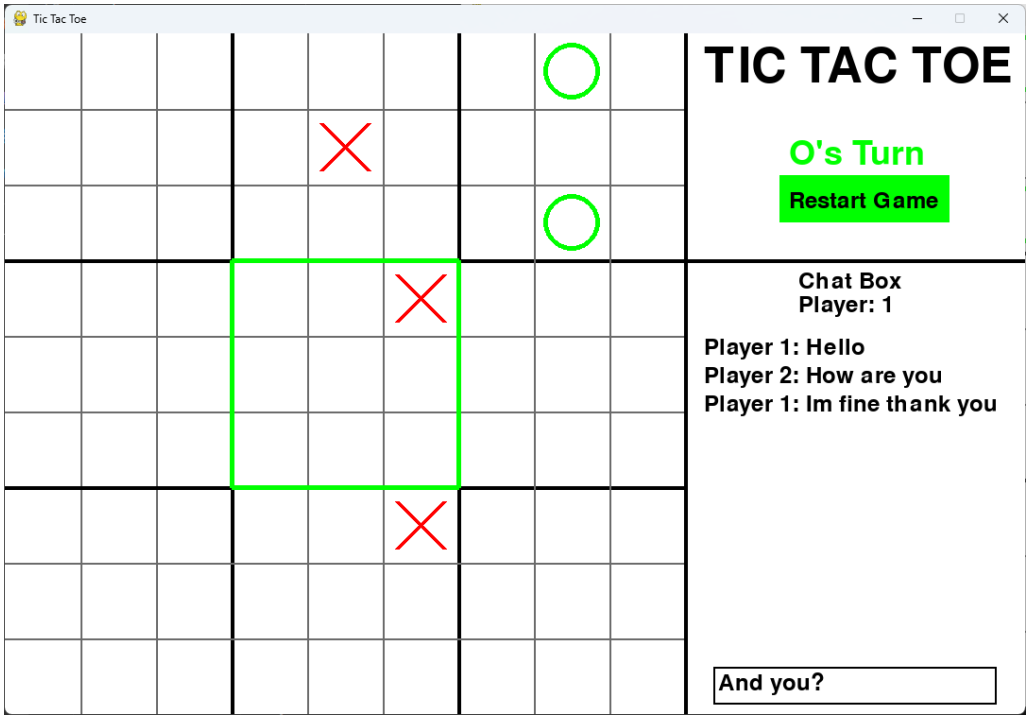




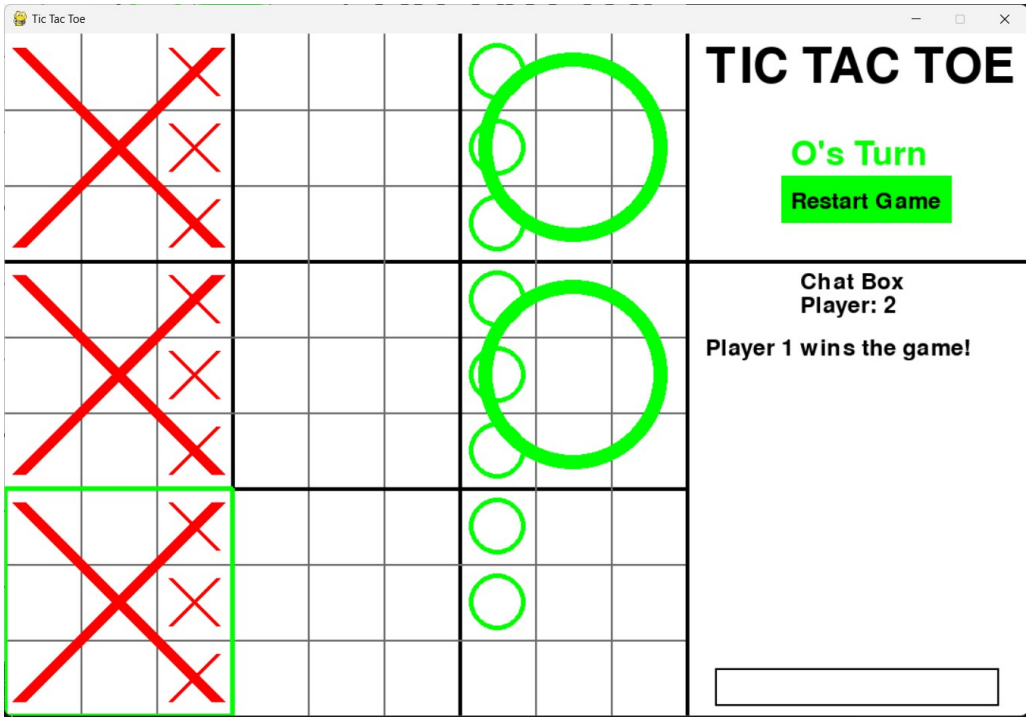
- Khi người dùng bấm nút Restart Game



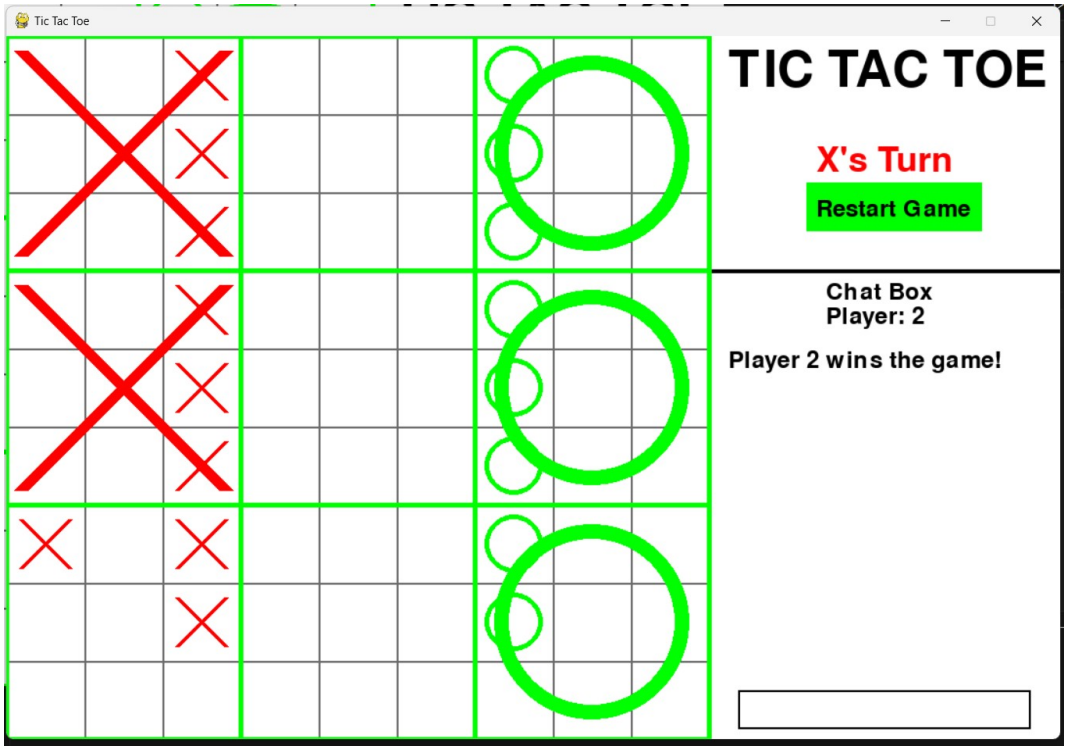
- Khi người dùng sử dụng phần chat box



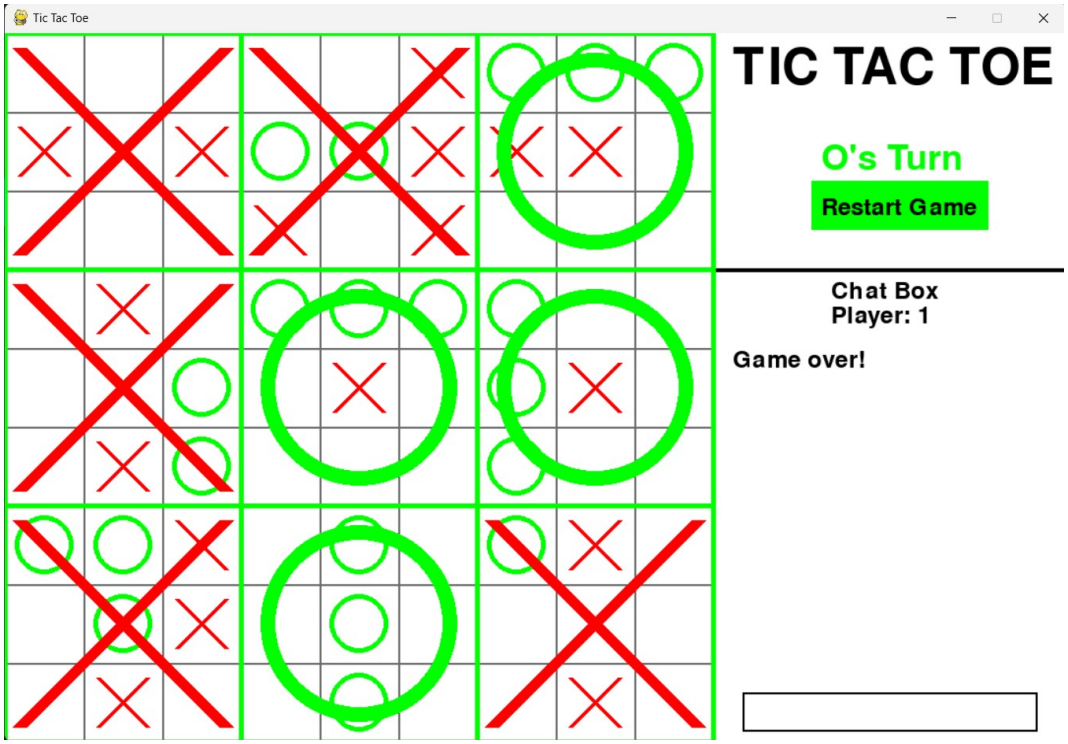
- Khi người chơi 1 chiến thắng



- Khi người chơi 2 chiến thắng



- Khi 2 người chơi hòa nhau



5 Nhiệm vụ và vai trò của thành viên trong nhóm

Lại Quang Hải: Thiết kế cơ chế và giao diện cho game, viết báo cáo Latex.
Dương Văn Trí: Thiết kế socket và threading cho game, Powerpoint, Readme.