TRƯỜNG ĐẠI HỌC SÀI GÒN
KHOA CÔNG NGHỆ THÔNG TIN

PHÁT TRIỂN PHẦN MỀM MÃ NGUỒN MỞ

Phát triển game cờ Gomoku 2 người chơi bằng Pygame và Socket server

GVHD: TỪ LÃNG PHIÊU
SV: NGUYỄN THÀNH LỘC - 3120410292
SV: NGUYỄN HOÀI LỘC - 3120410291

TP. HỒ CHÍ MINH, THÁNG 5/2024

Mục lục

1 GIỚI THIỆU ĐỀ TÀI

Trong thời đại kỹ thuật số ngày nay, trò chơi trực tuyến không chỉ là một hình thức giải trí phổ biến mà còn là một cách tuyệt vời để kết nối và giao lưu giữa mọi người trên toàn thế giới. Với sự phát triển không ngừng của công nghệ mạng, việc tạo ra các trò chơi trực tuyến chất lượng và hấp dẫn đã trở thành một mục tiêu quan trọng của nhiều nhà phát triển phần mềm.

Trong bối cảnh này, việc sử dụng mã nguồn mở đã trở thành một xu hướng phổ biến. Mã nguồn mở không chỉ mang lại sự linh hoạt và tính bảo mật mà còn khuyến khích sự hợp tác và đóng góp từ cộng đồng lập trình viên toàn cầu. Điều này mở ra cơ hội cho việc phát triển các ứng dụng và trò chơi mới mẻ, đồng thời thúc đẩy sự phát triển của ngành công nghiệp phần mềm.

Trong bối cảnh này, đề tài của chúng em là "Phát triển game cờ Gomoku 2 người chơi bằng Pygame và Socket". Trong dự án này, chúng em sẽ kết hợp giữa việc tạo ra một trò chơi cổ điển như Gomoku với việc sử dụng các công nghệ mã nguồn mở như Python và sử dụng Socket server tạo kết nối cho phép họ chơi trò chơi Gomoku với nhau qua mạng LAN. Ngoài ra còn sử dụng tích hợp thuật toán Minimax và Alpha-Beta Pruning để tạo ra một máy đánh cờ. Qua đó, không chỉ khám phá sức mạnh của phần mềm mã nguồn mở mà còn tạo ra một sản phẩm giải trí độc đáo và thú vị.

Qua dự án này, các mục tiêu được chúng em đề ra là:

- Nắm vững kiến thức về lập trình game sử dụng Pygame và xử lý sự kiện.
- Hiểu về cách thiết kế giao diện người dùng cho trò chơi.
- Học cách sử dụng Socket để thiết lập kết nối mạng giữa các máy tính.
- Xây dựng một bot AI để đánh nhau với người chơi.

2 CÁC CÔNG NGHỆ SỬ DỤNG

2.1 Ngôn ngữ lập trình Python

Python là một lựa chọn phù hợp cho việc phát triển ứng dụng này vì có nhiều lợi ích như:

- Có cú pháp đơn giản và dễ đọc, giúp tăng tính đảm bảo và hiểu biết của đội ngũ phát triển.
- Python là một ngôn ngữ đa năng và mạnh mẽ, có thể được sử dụng cho nhiều mục đích khác nhau từ phát triển web, xử lý dữ liệu, đến trí tuệ nhân tạo và học máy.
- Có một cộng đồng lớn và tích cực, cung cấp nhiều thư viện và framework hữu ích giúp việc phát triển ứng dụng trở nên nhanh chóng và dễ dàng hơn.
- Miễn phí và mã nguồn mở, giúp tiết kiệm chi phí phát triển và tạo điều kiện thuận lợi cho sự mở rộng và phát triển của dự án trong tương lai.

2.2 Pygame

Sử dụng Pygame để xây dựng giao diện ứng dụng mang lại nhiều ưu điểm quan trọng.

- Pygame là một thư viện mã nguồn mở cho Python được thiết kế để phát triển các ứng dụng và trò chơi đa phương tiện. Dựa trên SDL (Simple DirectMedia Layer), Pygame cung cấp các công cụ và API cho việc vẽ đồ họa, xử lý sự kiện, âm thanh và vận động, giúp người phát triển dễ dàng tạo ra các trò chơi và ứng dụng có giao diện đồ họa.
- Dễ học và sử dụng: Pygame được thiết kế để đơn giản và dễ hiểu, phù hợp cho cả người mới bắt đầu và những người có kinh nghiệm trong lập trình Python.
- Hỗ trợ đa nền tảng: Pygame hoạt động trên nhiều hệ điều hành khác nhau như Windows, macOS và Linux, cho phép bạn phát triển ứng dụng một cách linh hoạt trên các nền tảng khác nhau.
- Pygame cho phép người phát triển xử lý các sự kiện như nhấn phím, di chuyển chuột và nhấn nút, từ đó tạo ra các tương tác phong phú trong trò chơi.

2.3 Tkinter

Tkinter là một thư viện giao diện người dùng (GUI) được tích hợp sẵn trong Python, cho phép người phát triển tạo ra các ứng dụng có giao diện đồ họa một cách dễ dàng. Tkinter dựa trên toolkit GUI Tk, là một công cụ được sử dụng rộng rãi và phổ biến trong việc phát triển ứng dụng đồ họa trên nhiều nền tảng.Dưới đây là một số điểm nổi bật của Tkinter:

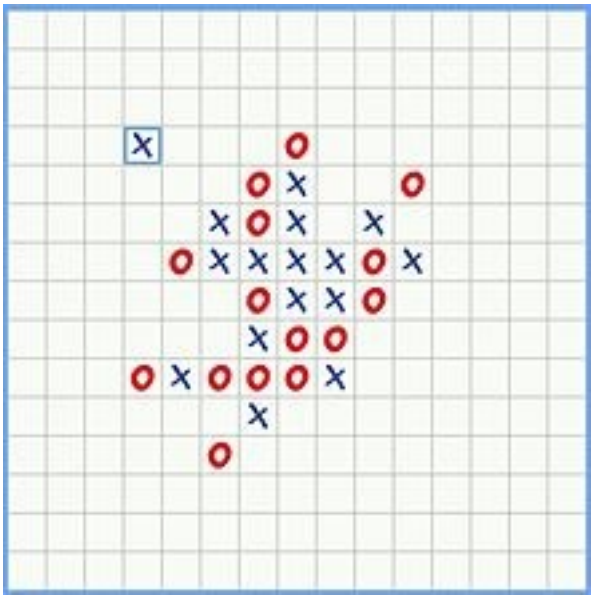
- Tích hợp sẵn trong Python: Tkinter được tích hợp sẵn trong bộ cài đặt Python, do đó không cần cài đặt bổ sung. Điều này làm cho việc sử dụng Tkinter trở nên tiện lợi và dễ dàng cho người phát triển Python.
- Tiết kiệm tài nguyên: Tkinter làm việc hiệu quả và tiết kiệm tài nguyên, không tạo ra nhiều gánh nặng cho hệ thống. Điều này làm cho các ứng dụng sử dụng Tkinter chạy mượt mà và nhanh chóng.
- Mặc dù Tkinter có thể không cung cấp những tính năng đồ họa phức tạp như Pygame, nhưng nó là một công cụ mạnh mẽ và linh hoạt cho việc tạo ra các ứng dụng có giao diện đồ họa đơn giản và trực quan trong Python.

2.4 Socket server

Socket là một giao diện lập trình ứng dụng (API) cung cấp bởi hệ điều hành hoặc các thư viện lập trình để thiết lập và quản lý các kết nối mạng. Trong ngữ cảnh của Python, thư viện Socket cung cấp các công cụ và chức năng để tạo ra các ứng dụng mạng, bao gồm cả việc gửi và nhận dữ liệu qua mạng.

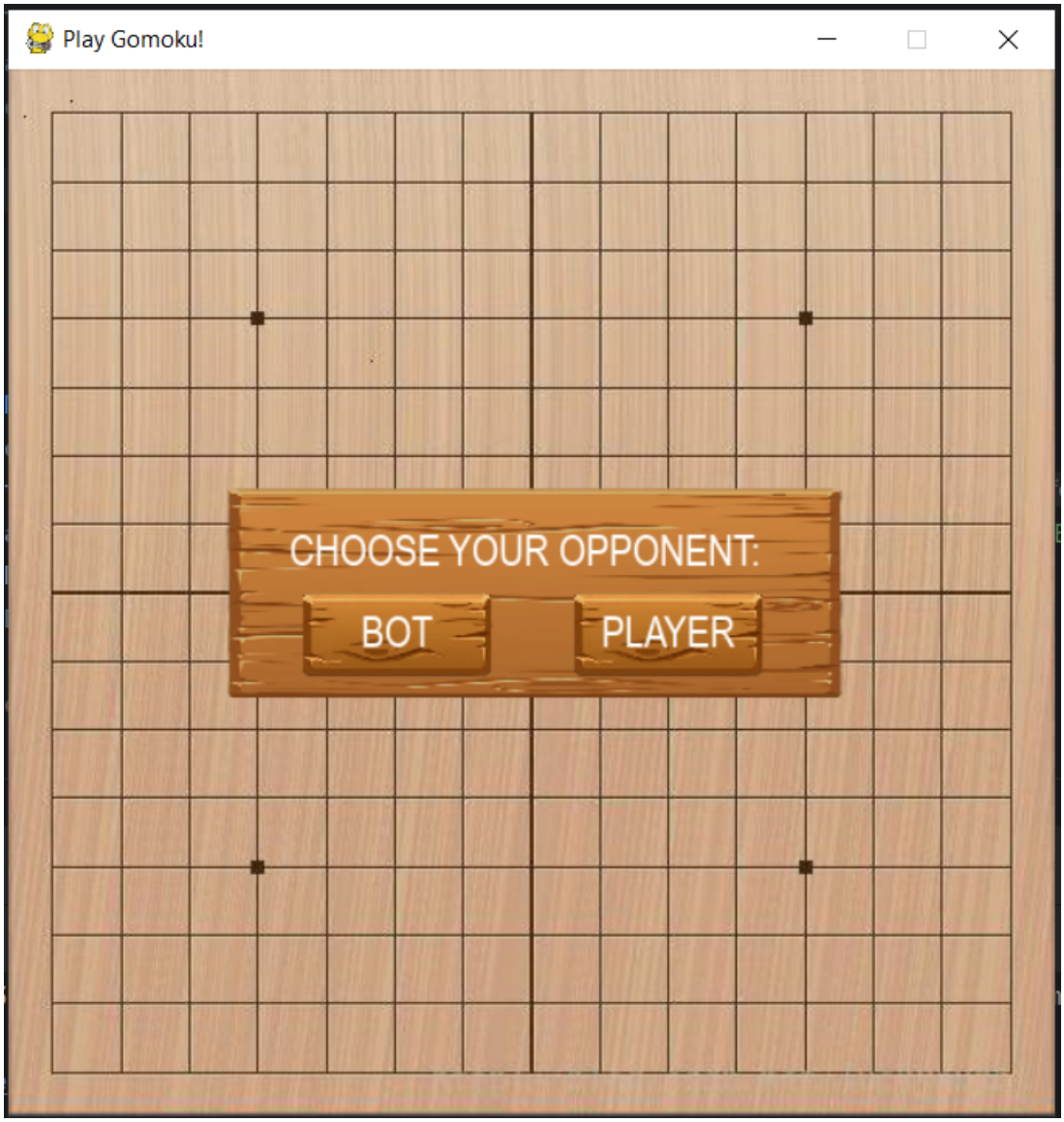
- Tạo và quản lý kết nối: Thư viện Socket cho phép bạn tạo ra các kết nối mạng, bao gồm cả kết nối TCP và UDP. Bằng cách sử dụng các phương thức như socket(), bind(), listen() và accept(), bạn có thể tạo ra và quản lý các kết nối giữa các máy tính.
- Gửi và nhận dữ liệu: Socket cho phép bạn gửi và nhận dữ liệu qua mạng giữa các máy tính. Sử dụng các phương thức như send() và recv(), bạn có thể truyền và nhận dữ liệu qua các kết nối mạng đã thiết lập.
- Phù hợp cho việc tạo ứng dụng mạng: Socket thường được sử dụng trong việc phát triển các ứng dụng mạng như trò chơi trực tuyến, ứng dụng trò chuyện, và ứng dụng chia sẻ tệp. Điều này cho phép các máy tính giao tiếp với nhau và truyền dữ liệu qua mạng một cách hiệu quả.
- Cấp cao hoặc cấp thấp: Socket cung cấp cả các phương thức cấp cao và cấp thấp để tạo ra và quản lý kết nối mạng. Bạn có thể sử dụng các API cấp cao như socketserver trong Python hoặc điều khiển trực tiếp các phương thức như socket() và connect() để kiểm soát cách tạo và quản lý kết nối mạng.

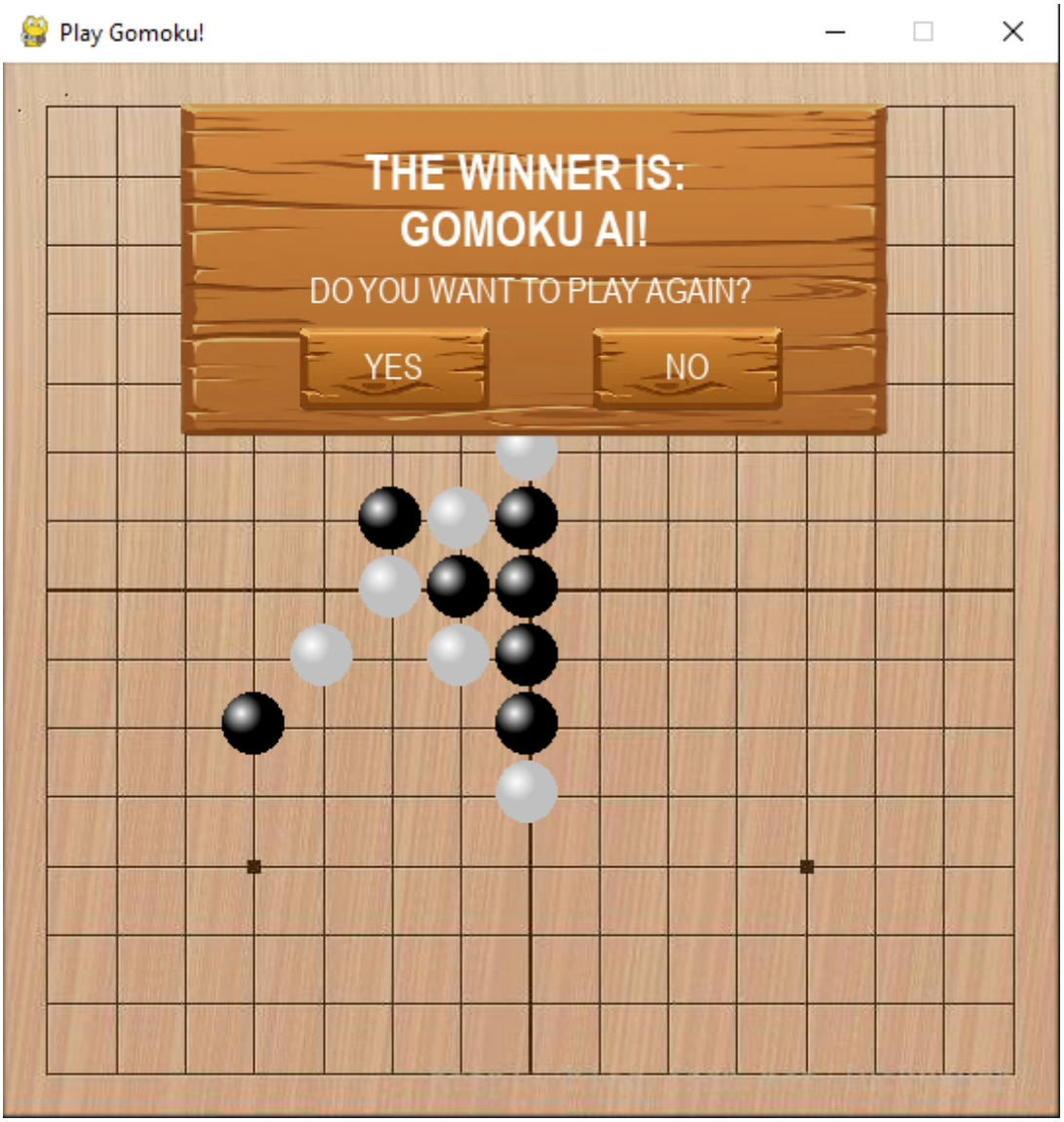
3 GIỚI THIỆU CỜ GOMOKU VÀ CÁC CHỨC NĂNG CHÍNH CỦA GAME CỜ GOMOKU



Giới thiệu về cờ Gomoku
Luật chơi của Gomoku như sau:

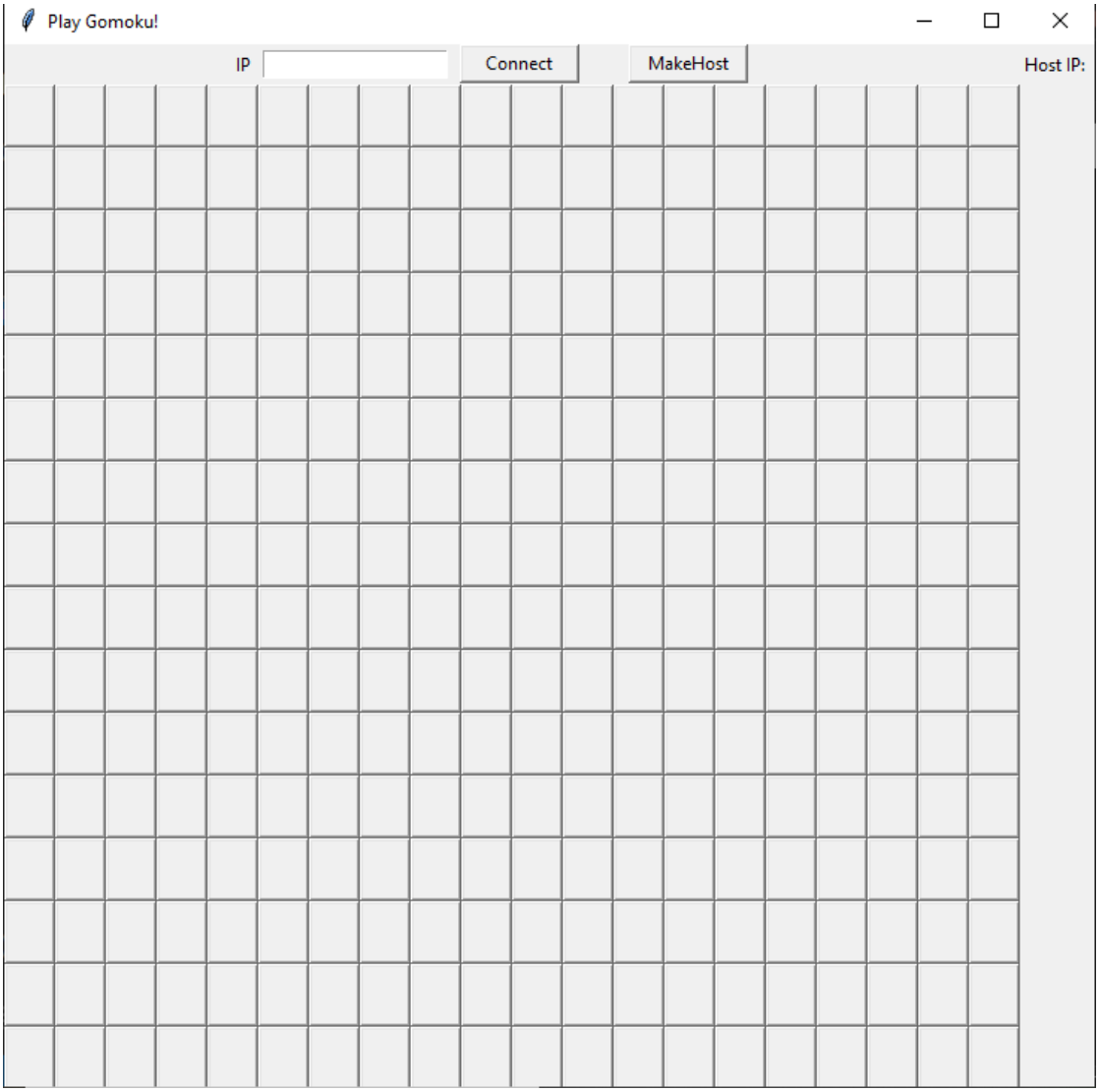
- Bàn cờ 15 x 15, quân đen đi trước.
- Lần lượt tới lượt đánh của 2 người đại diện cho quân trắng, quân đen.
- Người chiến thắng là người đầu tiên tạo ra một hàng 5 quân không gián đoạn theo chiều ngang, chiều dọc, hoặc đường chéo.
- Các nước Overline, lớn hơn hoặc bằng 6 quân thì không có giá trị với cả 2 bên và không bị coi là lỗi.


Play Gomoku!
CHOOSE YOUR OPPONENT:
BOT
PLAYER



Chức năng đánh với máy

- Sử dụng 2 thuật toán Minimax và Alpha-Beta Pruning để xây dựng bot AI.
- Giao diện chơi với máy được thiết kế bằng Pygame.
- Người chơi phải chọn màu quân cờ trước khi đánh với máy.
- Kết hợp nhiều hàm và các kỹ thuật khác để xây dựng nên một máy có khả năng đánh trả lại khi người chơi đánh một nước cờ. Các hàm chính như: hàm đánh giá, hàm alpha-beta Pruning, hàm kiểm tra chiến thắng, ...
- Các nút bấm hay thông báo được định nghĩa bên trong GUI thông qua thư viện Pygame.



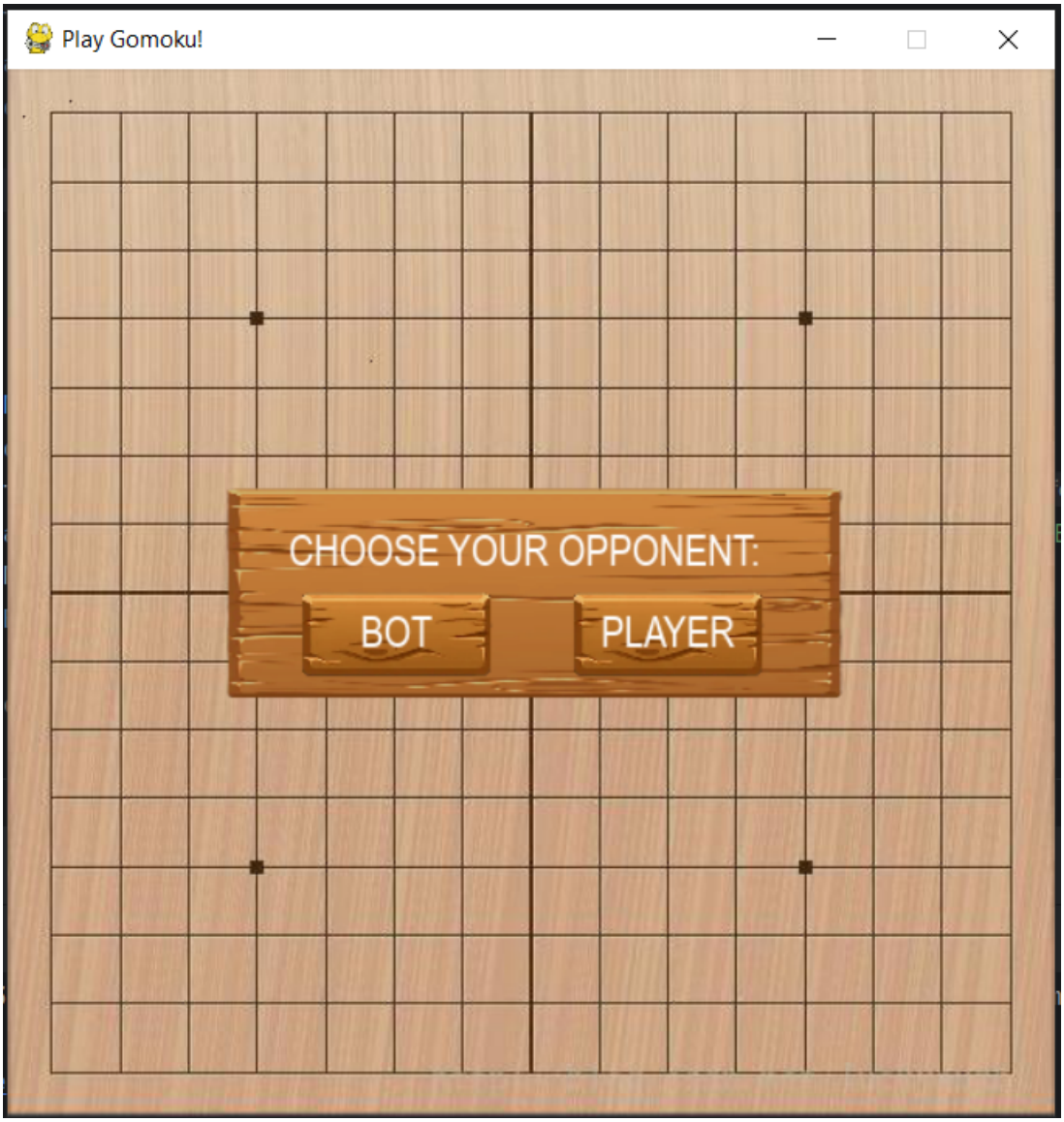
Chức năng đánh với người:

- Giao diện được xây dựng bằng Tkinter
- Sử dụng Socket Server cho phép 2 máy tính khác nhau truy cập vào cùng mạng LAN để đánh cờ với nhau.
- Nút Make Host: mục đích là tạo ra 1 server thông qua mạng LAN đang truy cập.
- Nút Connect: lúc này ở máy tính thứ 2 sẽ đóng vai trò là 1 client cần nhập IP vào input để kết nối tới server đó.
- IP của host được hiện trên label góc phải màn hình.

4 DEMO CHƯƠNG TRÌNH

4.1 Chức năng chọn chế độ chơi

4.1.1 Giao diện



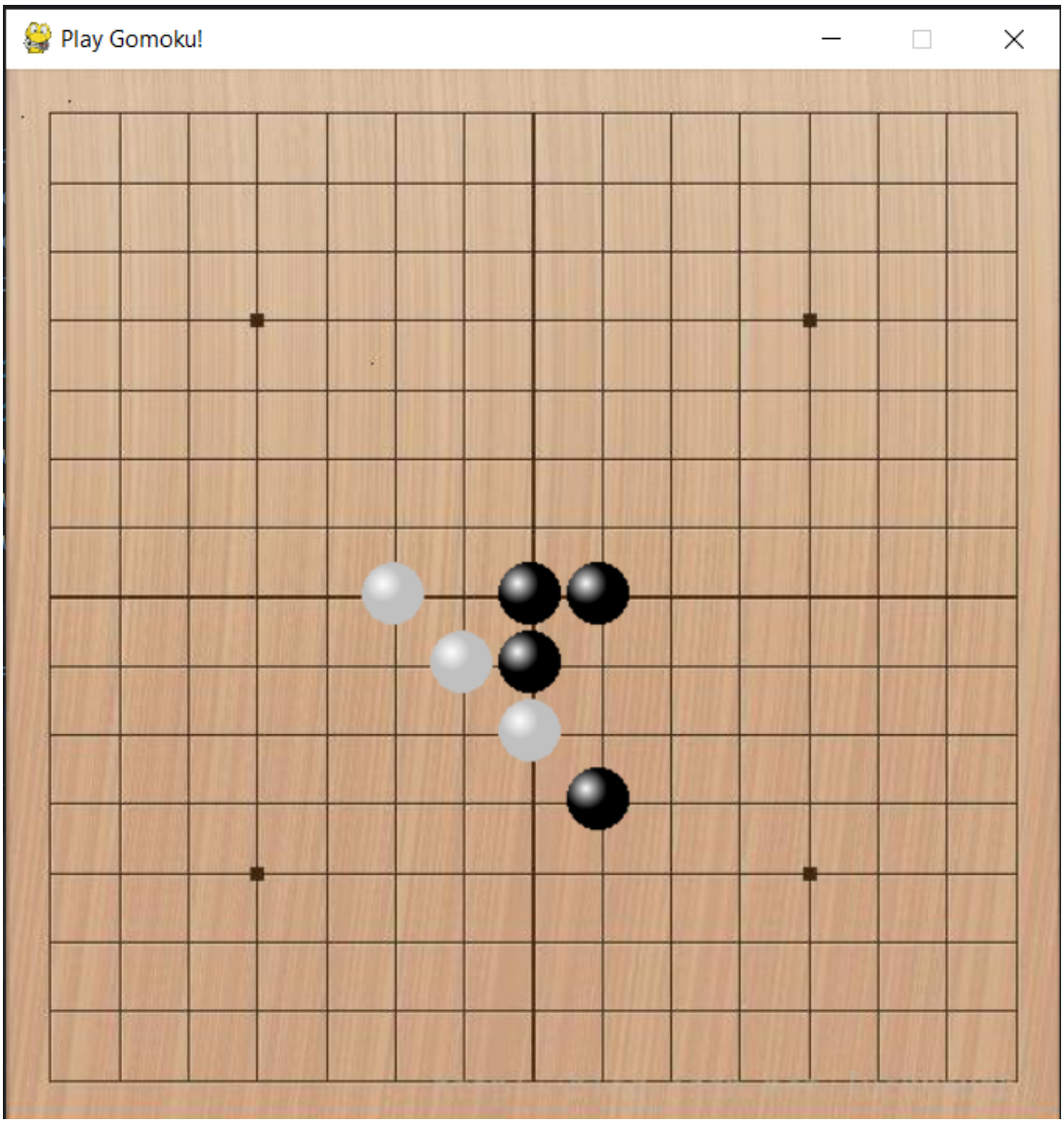
4.1.2 Xử lý

```
1       from gui.interface import *
2       from src.AI import *
3       from gui.button import *
4       from gui.PvPGUI import *
```

```
    import src.utils as utils
    import src.gomoku as gomoku
    import pygame

    def ChoseGameMode():
        game = ChooseMode()
        button_bot = Button(game.buttonSurf, 200, 290, "BOT", 22)
        button_player = Button(game.buttonSurf, 340, 290, "PLAYER", 22)
        game.drawMenu()
        game.drawButtons(button_bot, button_player, game.screen)

        run = True
        while run:
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    run = False
                if event.type == pygame.MOUSEBUTTONDOWN \
                        and pygame.mouse.get_pressed()[0]:
                    mouse_pos = pygame.mouse.get_pos()
                    # Check which color the user has chosen and set the states
                    game.checkOpponentChoice(button_bot, button_player, mouse_pos)
                    run = False
        return game


    def startGame():
        pygame.init()
        game = ChoseGameMode()
        if game.mode == 1:
            playWithBot()
        elif game.mode == 2:
            Ox = 20 #  S    lng      theo  trc  X
            Oy = 20 #  S    lng      theo  trc  Y
            window = Window()
            window.showFrame(Ox, Oy)
            window.mainloop()


    if __name__ == '__main__':
        startGame()
```

4.2 Chức năng Đánh với máy

4.2.1 Giao diện



4.2.2 Xử lý

```
from gui.interface import *
from src.AI import *
from gui.button import *
from gui.PvPGUI import *
import src.utils as utils
import src.gomoku as gomoku
import pygame
```

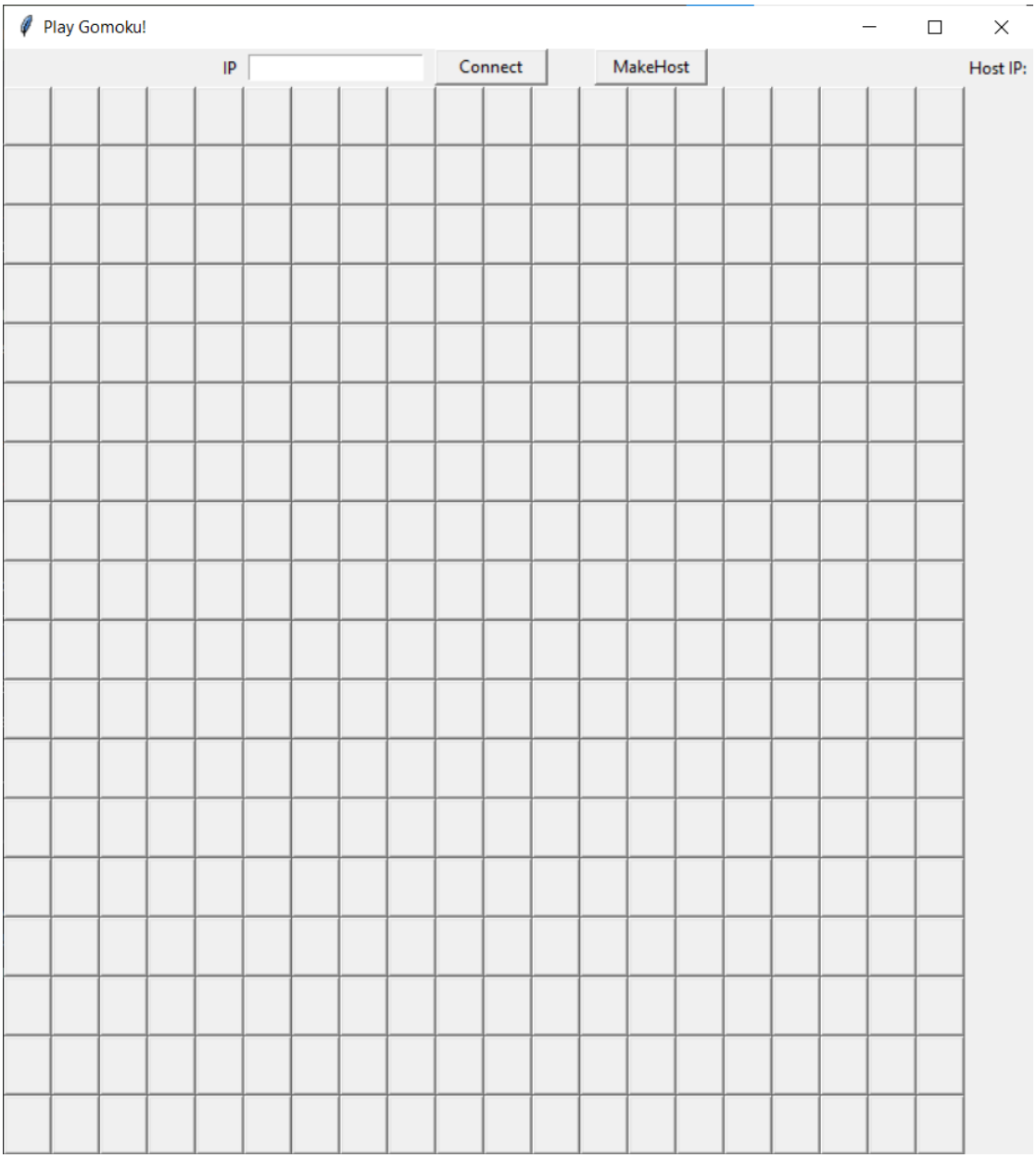
```
8
9
10      def playWithBot():
11          # Initializations
12          ai = GomokuAI()
13          game = GameUI(ai)
14          button_black = Button(game.buttonSurf, 200, 290, "BLACK", 22)
15          button_white = Button(game.buttonSurf, 340, 290, "WHITE", 22)
16
17          # Draw the starting menu
18          game.drawMenu()
19          game.drawButtons(button_black, button_white, game.screen)
20
21          run = True
22          while run:
23              for event in pygame.event.get():
24                  if event.type == pygame.QUIT:
25                      run = False
26                  if event.type == pygame.MOUSEBUTTONDOWN \
27                          and pygame.mouse.get_pressed()[0]:
28                      mouse_pos = pygame.mouse.get_pos()
29                      # Check which color the user has chosen and set the states
30                      game.checkColorChoice(button_black, button_white, mouse_pos)
31                      game.screen.blit(game.board, (0, 0))
32                      pygame.display.update()
33
34                      if game.ai.turn == 1:
35                          game.ai.firstMove()
36                          game.drawPiece('black', game.ai.currentI, game.ai.currentJ)
37                          pygame.display.update()
38                          game.ai.turn *= -1
39
40                      main(game)
41
42                      # When the game ends and there is a winner, draw the result board
43                      if game.ai.checkResult() is not None:
44                          last_screen = game.screen.copy()
45                          game.screen.blit(last_screen, (0, 0))
46                          # endMenu(game, last_screen)
47                          game.drawResult()
48
49                          # Setting for asking the player to restart the game or not
50                          yes_button = Button(game.buttonSurf, 200, 155, "YES", 18)
51                          no_button = Button(game.buttonSurf, 350, 155, "NO", 18)
52                          game.drawButtons(yes_button, no_button, game.screen)
53                          mouse_pos = pygame.mouse.get_pos()
54                          if yes_button.rect.collidepoint(mouse_pos):
55                              # Restart the game
56                              game.screen.blit(game.board, (0, 0))
57                              pygame.display.update()
58                              game.ai.turn = 0
59                              startGame()
60                          if no_button.rect.collidepoint(mouse_pos):
```

```
                        # End the game
                        pygame.quit()
            pygame.display.update()
        endMenu(game, last_screen)


    def endMenu(game, last_screen):
        pygame.init()
        game.screen.blit(last_screen, (0, 0))
        pygame.display.update()
        run = True
        while run:
            for event in pygame.event.get():
                game.drawResult()
                yes_button = Button(game.buttonSurf, 200, 155, "YES", 18)
                no_button = Button(game.buttonSurf, 350, 155, "NO", 18)
                game.drawButtons(yes_button, no_button, game.screen)
                if event.type == pygame.QUIT:
                    run = False
                if event.type == pygame.MOUSEBUTTONDOWN \
                        and pygame.mouse.get_pressed()[0]:
                    mouse_pos = pygame.mouse.get_pos()
                    if yes_button.rect.collidepoint(mouse_pos):
                        print('Selected YES')
                        game.screen.blit(game.board, (0, 0))
                        pygame.display.update()
                        playWithBot()
                    if no_button.rect.collidepoint(mouse_pos):
                        print('Selected NO')
                        run = False
        pygame.quit()


    # Main game play loop #
    def main(game):
        pygame.init()
        end = False
        result = game.ai.checkResult()

        while not end:
            turn = game.ai.turn
            color = game.colorState[turn] # black or white depending on player's
                choice
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    pygame.quit()

                # AI's turn
                if turn == 1:
                    move_i, move_j = gomoku.ai_move(game.ai)
                    # Make the move and update zobrist hash
                    game.ai.setState(move_i, move_j, turn)
                    game.ai.rollingHash ^= game.ai.zobristTable[move_i][move_j][0]
```

```
113                     game.ai.emptyCells -= 1
114
115                     game.drawPiece(color, move_i, move_j)
116                     result = game.ai.checkResult()
117                     # Switch turn
118                     game.ai.turn *= -1
119
120                 # Human's turn
121                 if turn == -1:
122                     if event.type == pygame.MOUSEBUTTONDOWN \
123                             and pygame.mouse.get_pressed()[0]:
124                         # Get human move played
125                         mouse_pos = pygame.mouse.get_pos()
126                         human_move = utils.pos_pixel2map(mouse_pos[0], mouse_pos[1])
127                         move_i = human_move[0]
128                         move_j = human_move[1]
129                         # print(mouse_pos, move_i, move_j)
130
131                         # Check the validity of human's move
132                         if game.ai.isValid(move_i, move_j):
133                             game.ai.boardValue = game.ai.evaluate(move_i, move_j,
                                    game.ai.boardValue, -1, game.ai.nextBound)
134                             game.ai.updateBound(move_i, move_j, game.ai.nextBound)
135                             game.ai.currentI, game.ai.currentJ = move_i, move_j
136                             # Make the move and update zobrist hash
137                             game.ai.setState(move_i, move_j, turn)
138                             game.ai.rollingHash ^=
                                    game.ai.zobristTable[move_i][move_j][1]
139                             game.ai.emptyCells -= 1
140
141                             game.drawPiece(color, move_i, move_j)
142                             result = game.ai.checkResult()
143                             game.ai.turn *= -1
144
145                 if result is not None:
146                     # End game
147                     end = True
```

4.3 Chức năng Đánh với người chơi

4.3.1 Giao diện



4.3.2 Xử lý

```
1       import tkinter as tk
2       from functools import partial
3       import threading
4       import socket
```

```
from tkinter import messagebox

import pygame

Ox = 20
Oy = 20


class Window(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title("Play Gomoku!")
        self.Buts = {}
        self.current_player = "server"
        self.turn_locked = False
        self.ip_label = tk.Label(self, text="Host IP: ")
        self.ip_label.grid(row=0, column=1, padx=0)
        self.Threading_socket = Threading_socket(self, self.ip_label)
        # Initialize pygame window
        pygame.init()
        self.pygame_screen = pygame.display.set_mode((800, 600))
        print(self.Threading_socket.name)

    def showFrame(self, Ox, Oy):
        frame1 = tk.Frame(self)
        frame1.grid(row=0, column=0)
        frame2 = tk.Frame(self)
        frame2.grid(row=1, column=0)

        tk.Label(frame1, text="IP", pady=4).grid(row=0, column=1)
        inputIp = tk.Entry(frame1, width=20) # Khung nhp   a    ch   ip
        inputIp.grid(row=0, column=2, padx=5)
        connectBT = tk.Button(frame1, text="Connect", width=10,
                              command=lambda:
                                  self.Threading_socket.clientAction(inputIp.get()))
        connectBT.grid(row=0, column=3, padx=3)

        makeHostBT = tk.Button(frame1, text="MakeHost", width=10,
                               command=lambda: self.start_server_thread())
        makeHostBT.grid(row=0, column=4, padx=30)
        # Draw pygame board
        for x in range(Ox):
            for y in range(Oy):
                self.Buts[x, y] = tk.Button(frame2, font=('arial', 15, 'bold'),
                    height=1, width=2,
                                            borderwidth=2,
                                                command=partial(self.handleButton,
                                                x=x, y=y))
                self.Buts[x, y].grid(row=x, column=y)

    def start_server_thread(self):
        thread = threading.Thread(target=self.Threading_socket.serverAction,
            args=(self.ip_label,))
```

```
53              thread.start()
54
55          def handleButton(self, x, y):
56              if self.Buts[x, y]['text'] == '':
57                  if not self.turn_locked:
58                      if self.current_player == self.Threading_socket.name:
59                          self.play_turn(x, y, self.Threading_socket.name)
60                          self.turn_locked = True
61                          print("  Lt     nh   : ", self.current_player)
62                      else:
63                          print(self.current_player)
64                          print(self.Threading_socket.name)
65                  else:
66                      print("  Lt     nh         b   kha .")
67              else:
68                  pass
69
70          def play_turn(self, x, y, player):
71              tag = 'O' if player == "server" else 'X'
72              self.Buts[x, y]['text'] = tag
73              self.Threading_socket.sendData("{}|{}|{}|{}".format("hit", x, y, tag))
74              if self.checkWin(x, y, tag, Ox, Oy):
75                  self.notification("Winner: ", player)
76                  self.newGame(Ox, Oy)
77
78          def update_ui(self, x, y, player):
79              self.Buts[x, y]['text'] = player
80              if player == 'O':
81                  self.current_player = "client"
82                  winner = 'server'
83              elif player == 'X':
84                  self.current_player = "server"
85                  winner = 'client'
86              if self.checkWin(x, y, player, Ox, Oy):
87                  self.notification("Winner: ", winner)
88                  self.newGame(Ox, Oy)
89              self.turn_locked = False
90
91          def notification(self, title, msg):
92              messagebox.showinfo(str(title), str(msg))
93
94          def checkWin(self, x, y, XO, Ox, Oy):
95              def checkDirection(dx, dy):
96                  count = 1
97                  i, j = x + dx, y + dy
98
99                  while 0 <= i < Ox and 0 <= j < Oy and self.Buts[i, j]["text"] == XO:
100                     count += 1
101                     i += dx
102                     j += dy
103
104                 i, j = x - dx, y - dy
105                 while 0 <= i < Ox and 0 <= j < Oy and self.Buts[i, j]["text"] == XO:
```

```
                    count += 1
                    i -= dx
                    j -= dy
                return count >= 5

            directions = [(0, 1), (1, 0), (1, 1), (1, -1)] # Hng, ct,    ng     cho
                 phi ,    ng     cho   tri

            for dx, dy in directions:
                if checkDirection(dx, dy):
                    return True
            return False

        def newGame(self, Ox, Oy):
            for x in range(Ox):
                for y in range(Oy):
                    self.Buts[x, y]['text'] = ''


    class Threading_socket():
        def __init__(self, gui, ip_label, conn=None):
            super().__init__()
            self.dataReceive = ""
            self.conn = conn
            self.gui = gui
            self.name = ""
            self.ip_label = ip_label

        def clientAction(self, inputIP):
            self.name = "client"
            print("client connect ..............")
            HOST = inputIP
            PORT = 8000

            self.conn = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            try:
                self.conn.connect((HOST, PORT))
            except ConnectionRefusedError:
                print("Connection refused. Server may not be available.")
                return
            self.gui.notification("    kt    ni    ti ", str(HOST))
            if inputIP: #  Kim  tra  nu  ip_label    c     truyn   vo
                self.ip_label.config(text=f"Host IP: {HOST}")
            t1 = threading.Thread(target=self.client)
            t1.start()

        def serverAction(self, ip_label=None):
            local_ip = get_local_ip()
            self.name = "server"
            HOST = local_ip # Ly    lp     a     ch
            print("Make host.........." + HOST)
            if ip_label:
                ip_label.config(text=f"Host IP: {HOST}")
```

```
158             PORT = 8000
159             s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
160             try:
161                 s.bind((HOST, PORT))
162                 s.listen(2)
163                 self.conn, addr = s.accept()
164                 t2 = threading.Thread(target=self.server, args=(addr, s))
165                 t2.start()
166             except Exception as e:
167                 print("Error:", e)
168                 if self.conn:
169                     self.conn.close()
170                 s.close()
171 
172         def server(self, addr, s):
173             try:
174                 print('Connected by', addr)
175                 while True:
176                     try:
177                         self.dataReceive = self.conn.recv(1024).decode()
178                         if self.dataReceive != "":
179                             action = self.dataReceive.split("|")[0]
180                             turn = self.dataReceive.split("|")[3]
181                             print(action)
182                             if action == "hit" and turn == "X":
183                                 x = int(self.dataReceive.split("|")[1])
184                                 y = int(self.dataReceive.split("|")[2])
185                                 self.gui.update_ui(x, y, turn)
186                         self.dataReceive = ""
187                     except ConnectionResetError:
188                         print("Connection to client reset.")
189                         break
190             finally:
191                 s.close()
192         def client(self):
193             while True:
194                 try:
195                     self.dataReceive = self.conn.recv(1024).decode()
196                     if self.dataReceive != "":
197                         action = self.dataReceive.split("|")[0]
198                         turn = self.dataReceive.split("|")[3]
199                         if action == "hit" and turn == "O":
200                             x = int(self.dataReceive.split("|")[1])
201                             y = int(self.dataReceive.split("|")[2])
202                             self.gui.update_ui(x, y, turn)
203                             print(action, turn, x, y)
204                     self.dataReceive = ""
205                 except ConnectionResetError:
206                     print("Connection to server reset.")
207                     break
208 
209         def sendData(self, data):
210             if self.conn:
```

```
211                 self.conn.sendall(str(data).encode())
212                 print(f"Sent data: {data}")
213             else:
214                 print(" Kt   ni   khng    c     thit   lp .")
215 
216 
217     def get_local_ip():
218         s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
219         try:
220             s.connect(('192.255.255.255', 1))
221             IP = s.getsockname()[0]
222         except:
223             IP = '127.0.0.1'
224         finally:
225             s.close()
226         return IP
```

5 KẾT LUẬN

5.1 Tổng kết công việc đã thực hiện

- Nắm bắt cách thức hoạt động của socket dùng trong mạng LAN
- Xây dựng được một ứng dụng đánh cờ gomuku với người chơi thông qua mạng LAN
- Việc ứng dụng thuật toán Minimax và cải tiến của nó là Alpha-Beta Pruning vào việc xây dựng một bot AI chơi cờ Gomoku là một bước tiến quan trọng trong lĩnh vực trí tuệ nhân tạo và trò chơi. Kết hợp hai phương pháp này giúp bot có thể tính toán các nước đi tiếp theo và đánh giá tình huống hiện tại một cách hiệu quả, từ đó tối ưu hóa quyết định và đảm bảo hiệu suất chơi tốt nhất có thể.

5.2 Các mặt hạn chế

- Tuy bot AI tạo ra còn chưa hoàn hảo nhưng qua quá trình học tập, nghiên cứu thì em học hỏi được nhiều thứ rất hay về lĩnh vực trí tuệ nhân tạo.
- Phương pháp và kiến thức được sử dụng để phát triển đồ án tương đối mới và đa dạng, đặc biệt là Socket server.

5.3 Hướng phát triển tiếp theo trong tương lai

- Tối ưu hóa thuật toán: trong hàm đánh giá (evaluation function) có thể cải tiến tốt hơn nữa. Bởi vì hàm này dựa vào kinh nghiệm và tư duy của người lập trình. Ngoài ra cần tiếp tục nghiên cứu và tối ưu hóa thuật toán Alpha-Beta Pruning để giảm thời gian tính toán mà vẫn đảm bảo chất lượng nước đi.
- Đồng bộ hóa giao diện đánh người với đánh máy nhằm nâng cao trải nghiệm người dùng