TRƯỜNG ĐẠI HỌC SÀI
GÒN KHOA CÔNG NGHỆ
THÔNG TIN

PHÁT TRIỂN PHẦN MỀM MÃ NGUỒN MỞ

Báo cáo môn Phát triển phần mềm mã nguồn mở

Đề tài: SOCKET VÀ ỨNG DỤNGTRONG GAME CỜ CARO

GVHD: Từ Lãng Phiêu

SV: Dương Mãn Quý

Trần Gia Huy

Nguyễn Hoàng Nam

Dương Thanh Tùng

TP. HỒ CHÍ MINH, THÁNG 5/2024

1 PHẦN MỞ ĐẦU

1.1 Lý do chọn đề tài

Cờ Caro, một trò chơi lịch sử, hiện nay đã trở thành một phần không thể thiếu trong cuộc sống học đường. Để đáp ứng nhu cầu giải trí và kết nối của các bạn trẻ, nhiều hãng phần mềm đã phát triển các phiên bản trò chơi Caro trên máy tính, giúp cho việc thưởng thức trò chơi này trở nên tiện lợi và dễ dàng hơn.

Với mong muốn mang niềm vui và sự kết nối qua trò chơi Caro đến với mọi người, nhóm của chúng tôi đã quyết định chọn chương trình phát triển trò chơi Caro làm đề tài nghiên cứu của mình. Chúng tôi hy vọng rằng thông qua dự án này, mọi người sẽ có cơ hội trải nghiệm trò chơi Caro một cách thuận tiện và thú vị hơn, đồng thời tạo ra một cộng đồng chơi game sôi động và gần gũi hơn.

1.2 Đối tượng nghiên cứu

Ngôn ngữ lập trình Python, game cờ caro

1.3 Phạm vi nghiên cứu

Ngôn ngữ lập trình Python, Tkinter, Socket, Threading

1.4 Mục tiêu nghiên cứu

Hiểu rõ hơn về Python, tiếp thu kiến thức làm game

1.5 Phương pháp nghiên cứu

Thu nhập và phân tích những thông tin liên quan đến đề tài để xây dựng ứng dụng. Xác định các yêu cầu để thiết kế ứng dụng

2 PHẦN NỘI DUNG

2.1 CHƯƠNG 1: LÝ THUYẾT

2.1.1 Các công nghệ sử dụng

2.1.1.a Tkinter

- Tkinter là thư viện tiêu chuẩn của Python để tạo giao diện người dùng đồ họa (GUI). Trong game này, tkinter được sử dụng để tạo ra bảng cờ caro, hiển thị các nước đi của người chơi và cập nhật trạng thái trò chơi.

+ Canvas: Dùng để vẽ bảng cờ caro và các quân cờ.

+ Label: Hiển thị các thông báo trạng thái (ví dụ: "Your turn", "Waiting for opponent").

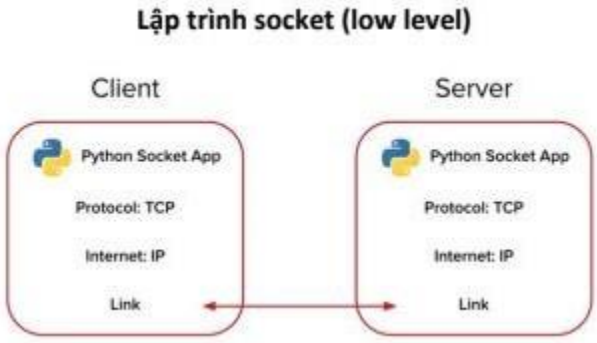
+ Button: Nhận các sự kiện nhấp chuột từ người chơi để thực hiện nước đi

2.1.1.b Threading

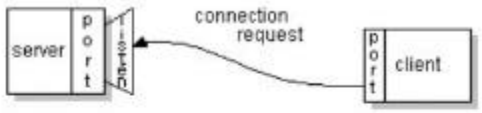
- Thư viện Threading của Python được sử dụng để xử lý các tác vụ song song. Trong trò chơi này, một luồng (thread) riêng biệt được tạo ra để lắng nghe dữ liệu từ server mà không làm gián đoạn giao diện người dùng.

- Receive Data Thread: Một luồng được tạo để nhận dữ liệu từ server liên tục, cập nhật trạng thái trò chơi mà không khóa giao diện.

2.1.1.c Socket

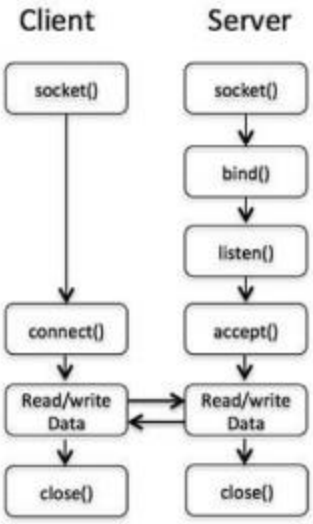


Socket là một điểm cuối (endpoint) của một liên kết giao tiếp 2 chiều giữa 2 chương trình chạy trên hệ thống mạng. Một socket được gắn với một cổng (PORT) để tầng TCP có thể nhận diện fíng dụng nào đã gửi dữ liệu đến.



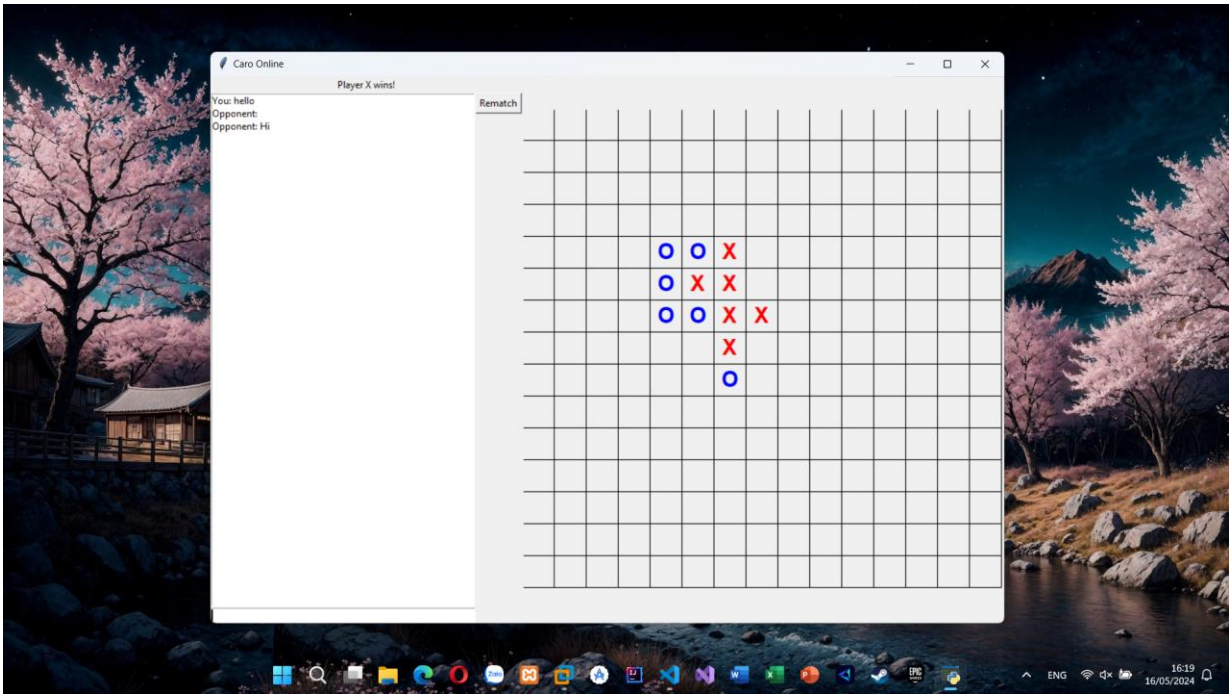
1. Client sẽ mở một kết nối TCP và cố gắng kết nối với server qua một PORT quy định.

2. Nếu kết nối thành công, server chấp nhận kết nối nó sẽ mở ra một PORT và duy trì kết nối này

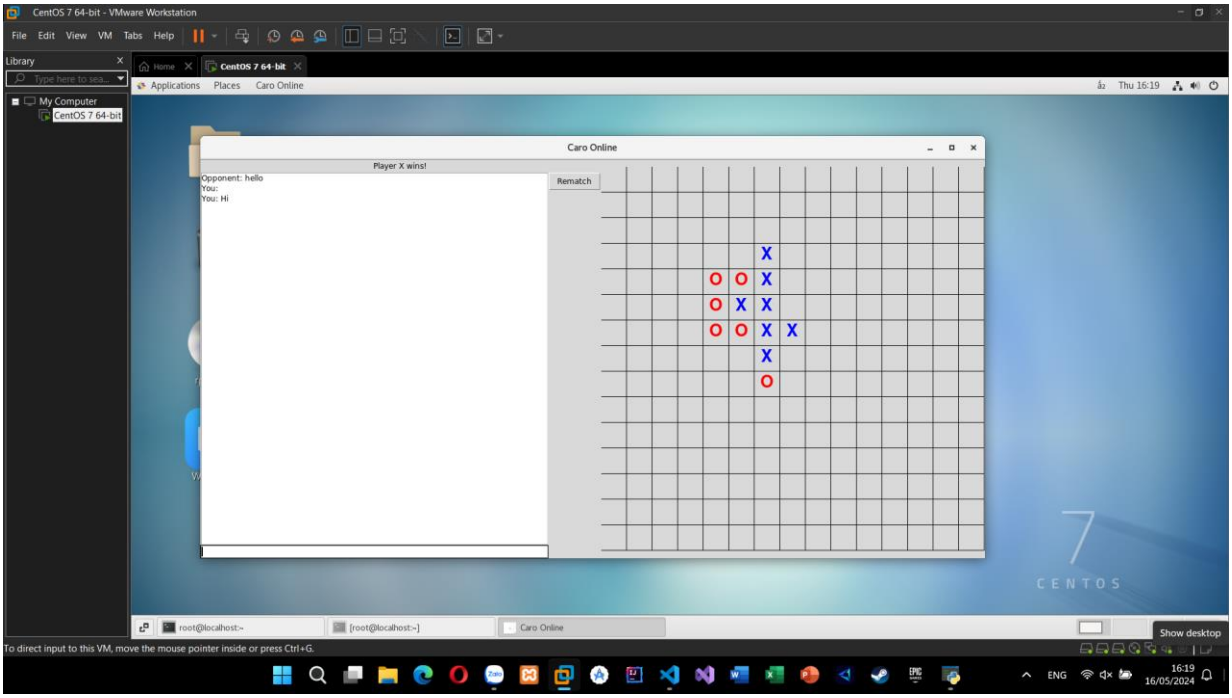


2.1.2 Giới thiệu về trò caro

Cờ caro là 1 trò chơi dân, ban đầu loại cờ này được chơi bằng các quân cờ vây (quân cờ màu trắng và đen) trên một bàn cờ vây. Quân đen đi trước và người chơi lần lượt đặt một quân cờ của họ trên giao điểm còn trống. Người thắng là người đầu tiên có được một chuỗi liên tục gồm 4 quân hàng ngang, hoặc dọc, hoặc chéo không bị chặn đầu nào. Một khi đã đặt xuống, các quân cờ không thể di chuyển hoặc bỏ ra khỏi bàn, do đó loại cờ này có thể chơi bằng giấy bút. Ở Việt Nam, cờ này thường chơi trên giấy tập học sinh (đã có sẵn các ô ca-rô), dùng bút đánh dấu hình tròn (O) và chữ X để đại diện cho 2 quân cờ. Cách chơi cũng tương tự nhưng người thắng là người có được một chuỗi liên tục 5 quân cờ hàng ngang, hoặc dọc, hoặc chéo mà không bị chặn hoặc bị quân khác ngắc đường quân cờ. Dưới đây là hình ảnh minh họa game cờ caro do nhóm phát triển:



Đây là một client được chạy trên hệ điều hành Windows.



Đây là một client khác chạy trên hệ điều hành CentOS.

2.2 CHƯƠNG 2: THIẾT KẾ CHƯƠNG TRÌNH

2.2.1 Tạo Server cho game cờ caro

```python
import socket
import threading

clients = {}
reset_requests = { 'X': False, 'O': False }

both_connected = False
both_connected_lock = threading.Lock()

def handle_client(client_socket, symbol):
    global reset_requests
    while True:
        try:
            data = client_socket.recv(1024).decode('utf-8')
            print(f"[RECEIVED from {symbol}] {data}")
            if data.startswith("MOVE") or data.startswith("WIN"):
                # Forward the data to the other client
                other_symbol = 'O' if symbol == 'X' else 'X'
                clients[other_symbol].send(bytes(data, 'utf-8'))
            elif data.startswith("RESET"):
                reset_requests[symbol] = True
                if all(reset_requests.values()):
                    # Reset the game for both players
                    reset_requests = { 'X': False, 'O': False }
                    for client in clients.values():
                        client.send(bytes("RESET", 'utf-8'))
            elif data.startswith("CHAT"):
                other_symbol = 'O' if symbol == 'X' else 'X'
                clients[other_symbol].send(bytes(data, 'utf-8'))
        except ConnectionResetError:
            print(f"[DISCONNECTED] {symbol} disconnected")
            break

```

```python
def start_server():
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind(("0.0.0.0", 12345))
    server.listen(2)
    print("[STARTED] Server is listening...")

    symbol = 'X'
    while True:
        client_socket, addr = server.accept()
        print(f"[CONNECTED] {addr} connected as {symbol}")
        clients[symbol] = client_socket
        client_socket.send(bytes(f"SYMBOL{symbol}", 'utf-8'))

        # Kiểm tra nếu cả hai người chơi đã kết nối
        if 'X' in clients and 'O' in clients:
            with both_connected_lock:
                both_connected = True  # Cập nhật trạng thái khi cả hai người chơi đã kết nối
            for client in clients.values():
                client.send(bytes("START", 'utf-8'))  # Gửi thông báo "START" đến cả hai client
            print("Both players connected, sending START signal")

        thread = threading.Thread(target=handle_client, args=(client_socket, symbol))
        thread.start()

        symbol = 'O' if symbol == 'X' else 'X'

if __name__ == "__main__":
    start_server()
```

2.2.2 Tạo Client cho game cờ caro

```python
import threading
import tkinter as tk
import socket
# Kích thước bảng caro
BOARD_SIZE = 15
both_connected = False
both_connected_lock = threading.Lock()  # Khóa để đồng bộ truy cập vào both_connected

def create_board(size):
    return [[' ' for _ in range(size)] for _ in range(size)]
```

```
def draw_board():
    for i in range(BOARD_SIZE):
        for j in range(BOARD_SIZE):
            canvas.create_rectangle(j*40, i*40, (j+1)*40, (i+1)*40, outline="black")
    canvas.bind("<Button-1>", lambda event: on_click(event))

def on_click(event):
    global my_turn, game_over, symbol, both_connected
    print(f"both_connected: {both_connected}, my_turn: {my_turn}, game_over: {game_over}")

    with both_connected_lock:
        if not both_connected:  # Kiểm tra nếu cả hai người chơi chưa kết nối
            label.config(text="Waiting for opponent to connect...")
            return

    if not my_turn or game_over:
        return
    col = event.x // 40
    row = event.y // 40
    if board[row][col] == ' ':
        draw_move(row, col, symbol)
        board[row][col] = symbol
        if check_win(board, row, col, symbol):
            label.config(text=f"Player {symbol} wins!")
            send_data(f"WIN{symbol}")
            game_over = True
            reset_button.pack()  # Show reset button
        else:
            send_data(f"MOVE{row:02}{col:02}")
            my_turn = False
            label.config(text="Waiting for opponent...")
```

```
def draw_move(row, col, symbol):
    x = col * 40 + 20
    y = row * 40 + 20
    canvas.create_text(x, y, text=symbol, font=('Helvetica', 20, 'bold'))

def send_data(data):
    try:
        client_socket.send(bytes(data, 'utf-8'))
        print(f"[SENT] {data}")
    except ConnectionResetError:
        print("[ERROR] Connection reset error")
        pass
```

```python
def receive_data():
    global my_turn, game_over, symbol, opponent_symbol, both_connected
    while True:
        try:
            data = client_socket.recv(1024).decode('utf-8')
            print(f"[RECEIVED] {data}")
            if data.startswith("SYMBOL"):
                symbol = data[6]
                opponent_symbol = 'O' if symbol == 'X' else 'X'
                my_turn = (symbol == 'X')
                label.config(text="Your turnnnn" if my_turn else "Waiting for opponent...")
            elif data.startswith("MOVE"):
                row = int(data[4:6])
                col = int(data[6:8])
                board[row][col] = opponent_symbol
                draw_move(row, col, opponent_symbol)
                if check_win(board, row, col, opponent_symbol):
                    label.config(text=f"Player {opponent_symbol} wins!")
                    game_over = True
                    reset_button.pack()  # Show reset button
                else:
                    my_turn = True
                    label.config(text="Your turn")
            elif data.startswith("WIN"):
                label.config(text=f"Player {data[3]} wins!")
                game_over = True
                reset_button.pack()  # Show reset button
            elif data.startswith("RESET"):
                reset_game()
            elif data.startswith("CHAT"):
                chat_message = data[4:]
                chat_listbox.insert(tk.END, f"Opponent: {chat_message}")
```

```python
            elif data.startswith("START"):
                with both_connected_lock:
                    both_connected = True  # Cập nhật trạng thái khi cả hai người chơi đã kết nối
                print("Both players are now connected. Chat enabled and game can start.")
                chat_entry.config(state=tk.NORMAL)  # Bật chức năng chat
                label.config(text="Both players connected. You can start chatting!")
        except ConnectionResetError:
            print("[ERROR] Connection reset error")
            break
```

```
def reset_game():
    global board, my_turn, game_over
    board = create_board(BOARD_SIZE)
    canvas.delete("all")
    draw_board()
    game_over = False
    my_turn = (symbol == 'X')
    label.config(text="Your turn" if my_turn else "Waiting for opponent...")
    reset_button.pack_forget()  # Hide reset button

def on_reset_click():
    send_data("RESET")
```

```
root = tk.Tk()
root.title("Caro Online")

board = create_board(BOARD_SIZE)
my_turn = False
game_over = False
symbol = ''
opponent_symbol = ''

canvas = tk.Canvas(root, width=BOARD_SIZE*40, height=BOARD_SIZE*40)
canvas.pack(side=tk.RIGHT)

draw_board()

label = tk.Label(root, text="Connecting...")
label.pack()

reset_button = tk.Button(root, text="Rematch", command=on_reset_click)
```

2.2.3 Tạo khung chat cho game cờ caro

```
def send_chat_message(event=None):
    message = chat_entry.get()
    chat_listbox.insert(tk.END, f"You: {message}")
    chat_entry.delete(0, tk.END)
    send_data(f"CHAT{message}")
```

```
chat_frame = tk.Frame(root)
chat_frame.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)

chat_listbox = tk.Listbox(chat_frame, height=15, width=50)
chat_listbox.pack(side=tk.TOP, fill=tk.BOTH, expand=True)

chat_entry = tk.Entry(chat_frame, width=50)
chat_entry.pack(side=tk.BOTTOM, fill=tk.X)
chat_entry.bind("<Return>", send_chat_message)
chat_entry.config(state=tk.DISABLED)
```

2.2.4 Tìm người chơi thắng

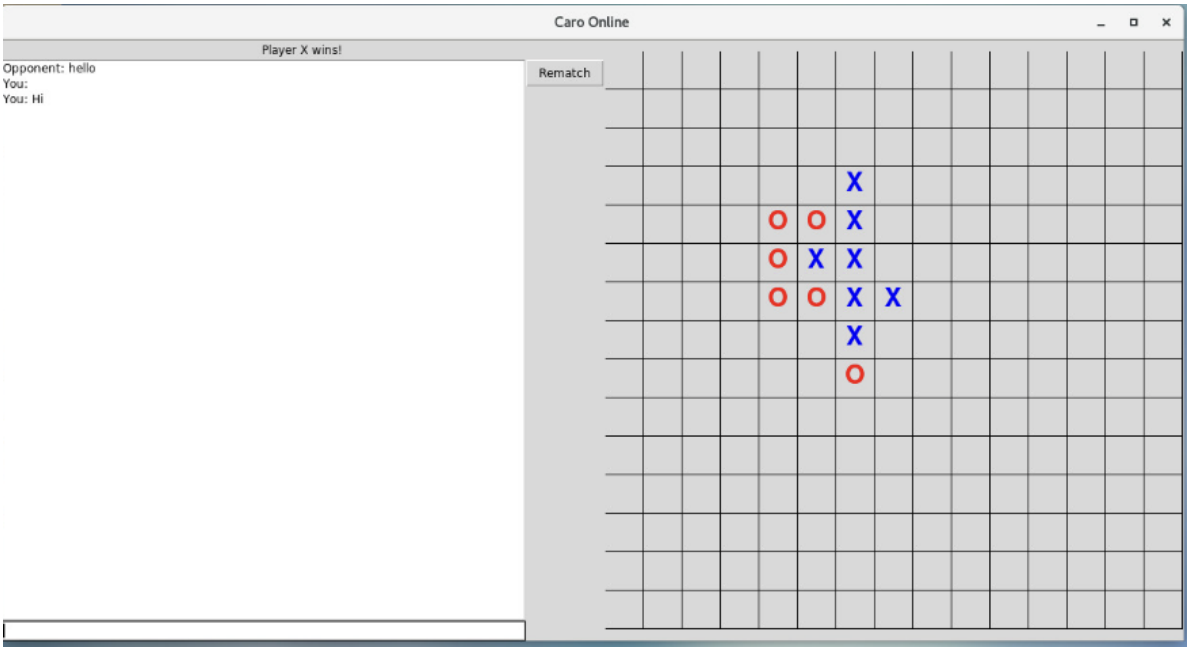
Kiểm tra xem ai đi được 5 ô trước

```python
def check_win(board, row, col, symbol):
    # Kiểm tra hàng
    count = 0
    for c in range(col - 4, col + 5):
        if 0 <= c < BOARD_SIZE and board[row][c] == symbol:
            count += 1
            if count == 5:
                return True
        else:
            count = 0

    # Kiểm tra cột
    count = 0
    for r in range(row - 4, row + 5):
        if 0 <= r < BOARD_SIZE and board[r][col] == symbol:
            count += 1
            if count == 5:
                return True
        else:
            count = 0

    # Kiểm tra đường chéo chính
    count = 0
    for i in range(-4, 5):
        r, c = row + i, col + i
        if 0 <= r < BOARD_SIZE and 0 <= c < BOARD_SIZE and board[r][c] == symbol:
            count += 1
            if count == 5:
                return True
        else:
            count = 0
```

(variable) c: Any

```python
    # Kiểm tra đường chéo phụ
    count = 0
    for i in range(-4, 5):
        r, c = row + i, col - i
        if 0 <= r < BOARD_SIZE and 0 <= c < BOARD_SIZE and board[r][c] == symbol:
            count += 1
            if count == 5:
                return True
        else:
            count = 0

    return False
```

2.2.5 Giao diện



3 KẾT LUẬN

3.1 Ưu điểm

Sử dụng Socket cho phép người chơi tương tác trực tiếp với nhau thông qua mạng, tạo ra trải nghiệm chơi game chân thực và hấp dẫn hơn.

Với việc sử dụng Socket, game Caro có thể hỗ trợ nhiều người chơi cùng tham gia vào trò chơi, tạo ra một môi trường trò chơi đa người chơi trực tuyến.

Viết game Caro use Socket là một cách tốt để rèn luyện kĩ năng lập trình mạng trong Python, bao gồm thiết lập kết nối, truyền dữ liệu và xử lý gói tin.

Socket cho phép tạo ra các tính năng tương tác như trò chuyện trực tiếp giữa người chơi, bổ sung thêm tính năng phong phú và hấp dẫn cho trò chơi.

3.2 Nhược điểm

Do ảnh hưởng của mạng internet và tình trạng mạng của người chơi, có thể xảy ra lỗi mạng hoặc đút trong quá trình truyền dữ liệu, ảnh hưởng đến trải nghiệm trò chơi.

Việc truyền dữ liệu qua mạng sử dụng Socket có thể gặp vấn đề liên quan đến bảo mật. Dữ liệu có thể được đánh cắp hoặc can thiệp nếu không áp dụng các biện pháp bảo mật phù hợp.