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 DỤNG TRONG GAME CỜ CARO

GVHD: Từ Lãng Phiêu
SV: Vũ Ngọc Tú - 3121410552
Nguyễn Hoàng Tuấn - 3121410558
Lý Văn Trưởng - 3121410547

TP. HỒ CHÍ MINH, THÁNG 5/2024

Mục lục

1 PHẦN GIỚI THIỆU

1.1 Lý do chọn đề tài

Game caro có luật chơi đơn giản, dễ hiểu nhưng lại mang lại trải nghiệm chơi game thú vị và hấp dẫn. Người chơi có thể dễ dàng tiếp cận và tham gia vào trò chơi mà không cần nhiều kiến thức hay kỹ năng đặc biệt. Phát triển game caro sẽ giúp bạn thực hành các kỹ năng lập trình như xử lý logic game, xử lý sự kiện, quản lý trạng thái trò chơi, và làm việc với đồ họa. Phát triển một trò chơi caro sẽ giúp bạn hiểu rõ hơn về cách hoạt động của các trò chơi logic và cách thiết kế các thuật toán để kiểm soát trò chơi.

1.2 Đối tượng nghiên cứu

Đối tượng nghiên cứu trong đề tài này bao gồm:

- Ngôn ngữ lập trình Python: một ngôn ngữ lập trình phổ biến với cú pháp rõ ràng và dễ học.
- Game caro nói riêng và các game lập trình bằng Python nói chung: nghiên cứu về cấu trúc, thiết kế và lập trình các game này.

1.3 Phạm vi nghiên cứu

Phạm vi nghiên cứu bao gồm:

- Ngôn ngữ lập trình Python và các tính năng của nó.
- Thư viện Pygame: một thư viện Python mạnh mẽ cho lập trình game.
- Cơ bản về lập trình socket: để xây dựng các chức năng multiplayer cho game caro

1.4 Mục tiêu nghiên cứu

Mục tiêu nghiên cứu bao gồm:

- Hiểu và thành thạo ngôn ngữ lập trình Python.
- Tiếp cận và ứng dụng lập trình socket trong game.
- Xây dựng một game đối kháng cơ bản sử dụng Python và Pygame.

2 PHẦN NỘI DUNG

2.1 LÝ THUYẾT

2.1.1 Tổng quan về Pygame

2.1.1.a Khái niệm

Pygame là một thư viện Python dùng để phát triển các trò chơi điện tử. Nó cung cấp các module cho đồ họa và âm thanh, cho phép các nhà phát triển tạo ra các game phong phú và tương tác một cách dễ dàng.

2.1.1.b Cách cài đặt Pygame trong Python

Để cài đặt Pygame, bạn có thể sử dụng pip:

```
pip install pygame
```

Sau khi cài đặt, bạn có thể bắt đầu lập trình game với Pygame bằng cách import thư viện này vào mã nguồn của bạn.

2.1.2 Giới thiệu về lập trình socket

Lập trình socket là một kỹ thuật lập trình để thiết lập giao tiếp giữa các thiết bị qua mạng. Sockets là điểm cuối của một kết nối hai chiều giữa hai chương trình đang chạy trên mạng.

2.1.2.a Khái niệm về socket

Socket là một giao diện lập trình mạng (API) cho phép các chương trình trên các máy tính khác nhau giao tiếp với nhau. Nó hỗ trợ các giao thức truyền thông như TCP (Transmission Control Protocol) và UDP (User Datagram Protocol).

2.1.2.b Cách hoạt động của socket

Socket hoạt động theo mô hình client-server, nơi một máy chủ (server) lắng nghe các yêu cầu từ các máy khách (client). Dưới đây là các bước cơ bản của quá trình này:

1. Máy chủ tạo một socket và liên kết (bind) nó với một địa chỉ IP và số cổng.
2. Máy chủ chuyển socket sang chế độ lắng nghe (listen) và chờ đợi các yêu cầu từ máy khách.
3. Máy khách tạo một socket và gửi yêu cầu kết nối đến máy chủ.
4. Máy chủ chấp nhận kết nối từ máy khách.
5. Cả máy chủ và máy khách có thể gửi và nhận dữ liệu thông qua các socket đã kết nối.
6. Khi hoàn thành, cả hai phía đóng kết nối.

2.1.2.c Lập trình socket trong Python

Python cung cấp một thư viện tích hợp gọi là ‘socket‘ để thực hiện lập trình socket. Dưới đây là ví dụ cơ bản về cách tạo một máy chủ và máy khách sử dụng socket trong Python.

2.1.2.1 Máy chủ (Server)

```
import socket

# Tạo socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Liên kết socket với địa chỉ và cổng
server_socket.bind(('localhost', 8080))

# Chuyển socket sang chế độ lắng nghe
server_socket.listen(1)
print("Server đang lắng nghe...")

# Chấp nhận kết nối từ máy khách
client_socket, client_address = server_socket.accept()
print(f"Kết nối từ {client_address}")

# Nhận dữ liệu từ máy khách
data = client_socket.recv(1024).decode()
print(f"Dữ liệu nhận được: {data}")

# Gửi dữ liệu phản hồi
client_socket.send("Dữ liệu đã được nhận".encode())

# Đóng kết nối
client_socket.close()
server_socket.close()
```

2.1.2.2 Máy khách (Client)

```
import socket

# Tạo socket
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Kết nối đến máy chủ
client_socket.connect(('localhost', 8080))

# Gửi dữ liệu
client_socket.send("Hello, Server!".encode())

# Nhận phản hồi từ máy chủ
data = client_socket.recv(1024).decode()
print(f"Phản hồi từ Server: {data}")

# Đóng kết nối
client_socket.close()
```

2.1.2.d Ứng dụng của lập trình socket trong game Caro

Trong game Caro, lập trình socket có thể được sử dụng để tạo ra tính năng chơi multiplayer, cho phép hai người chơi ở hai máy tính khác nhau kết nối với nhau thông qua mạng Internet và chơi cùng một trận đấu.

2.1.3 Giới thiệu về game Caro

Game Caro, hay còn gọi là Gomoku, là một trò chơi trí tuệ hai người chơi trên một bàn cờ với các quy tắc đơn giản. Mục tiêu của trò chơi là đạt được một hàng ngang, dọc hoặc chéo gồm ba quân cờ của mình trước đối thủ.

2.1.3.a Lịch sử và sự phát triển

Game Caro có lịch sử lâu đời, có nguồn gốc từ Trung Quốc và sau đó được phổ biến trên toàn thế giới. Trò chơi này đã được chuyển thể vào nhiều nền tảng khác nhau, từ bàn cờ giấy đến game điện tử.

2.1.3.b Các yếu tố cơ bản của game Caro

Game Caro bao gồm các yếu tố sau:

- **Bàn cờ:** Là một lưới ô vuông, thường là 15x15 hoặc 19x19 ô.
- **Quân cờ:** Hai loại quân cờ khác nhau, thường là "X" và "O", mỗi người chơi điền vào một loại.
- **Quy tắc:** Người chơi thay phiên nhau đánh quân cờ của mình vào một ô trống trên bàn cờ, mục tiêu là tạo ra một hàng ngang, dọc hoặc chéo gồm ba quân cờ của mình.

2.1.3.c Các thuật ngữ phổ biến

Trong game Caro, có một số thuật ngữ phổ biến như:

- **Thắng lợi:** Khi một người chơi tạo ra một hàng ngang, dọc hoặc chéo gồm ba quân cờ của mình.
- **Hòa:** Trường hợp cả hai người chơi không thể thắng trong một trận đấu, thường xảy ra khi bàn cờ đã đầy.

2.1.3.d Multiplayer và tương tác mạng

Trong game Caro, chế độ multiplayer cho phép hai người chơi ở hai máy tính khác nhau kết nối với nhau thông qua mạng Internet và chơi cùng một trận đấu. Sử dụng lập trình socket, thông điệp về các nước cờ được truyền giữa hai người chơi để đồng bộ hóa trạng thái của trò chơi.

2.2 THIẾT KẾ ỨNG DỤNG

2.2.1 Tạo server cho game cờ Caro

```python
import socket
import threading
import sys


# Server IP and port
IP = "192.168.1.2"
PORT = 5555

turn = "X"


# Socket type and options
server_socket = socket.socket(family=socket.AF_INET, type=socket.SOCK_STREAM)
server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

server_socket.bind((IP, PORT))
server_socket.listen()

# dictionary of client sockets and their nicknames
clients = {}

# debugging
print(f"Listening for connections on {IP}:{PORT}...")
```

Hình 1: Server cho game Caro

```python

# Sending Messages To All Connected Clients
def broadcast(message, client_socket):
    # Send messages to all clients except to the original sender
    for client in clients.keys():
        if client is not client_socket:
            client.send(message.encode("utf-8"))


# Trong phương thức handle
def handle(client_socket):
    while True:
        message = client_socket.recv(1024).decode("utf-8")
        print(message)
        if "!exit" in message:
            client_socket.close()
            broadcast("{} left!".format(clients[client_socket]), client_socket)
            clients.pop(client_socket)
            print(clients)
            sys.exit()
        elif "WINNER:" in message:  # Kiểm tra nếu là thông điệp chiến thắng
            broadcast(message, client_socket)  # Gửi lại thông điệp này cho tất cả các client
        else:
            broadcast(message, client_socket)

```

Hình 2: Server cho game Caro

```python

# Receiving / Listening Function
def receive():
    global turn
    while True:
        # Accept Connection
        client_socket, address = server_socket.accept()
        print("Connected with {}".format(str(address)))
        client_socket.send(turn.encode('utf-8'))
        nickname = f'player {turn}'
        if turn == "X":
            turn = "O"
        elif turn == "O":
            turn = "X"
        # Add client info to the dictionary
        clients.update({client_socket: nickname})
        # Start Handling Thread For Client
        thread = threading.Thread(target=handle, args=(client_socket,))
        thread.start()


receive()

```

Hình 3: Server cho game Caro

2.2.2 Tạo client cho game cờ Caro

```python
import sys
import socket
import threading
import pygame

# Initialize Pygame
pygame.init()

# Screen dimensions
WIDTH, HEIGHT = 600, 900
GRID_SIZE = 3
CELL_SIZE = WIDTH // GRID_SIZE
FONT_SIZE = 60
CHAT_FONT_SIZE = 20
CHAT_BOX_HEIGHT = 200

# Colors
GREEN = (5, 192, 0)
WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
GRAY = (200, 200, 200)
RED = (255, 0, 0)
BLUE = (0, 0, 255)
CHAT_BG_COLOR = (200, 200, 200)  # Darker background color for the chat box
```

Hình 4: Client của game Caro

```
25
26  # Fonts
27  FONT = pygame.font.SysFont('arial', FONT_SIZE)
28  CHAT_FONT = pygame.font.SysFont('arial', CHAT_FONT_SIZE)
29
30  # Chat assets
31  CHAT_ICON = pygame.image.load("chat_icon.png")
32
33
34  class SocketChat:
        tabnine: test | explain | document | ask
35      def __init__(self):
36          self.IP = "192.168.1.2"
37          self.PORT = 5555
38          self.client_socket = socket.socket(family=socket.AF_INET, type=socket.SOCK_STREAM)
39
        tabnine: test | explain | document | ask
40      def receive(self):
41          message = self.client_socket.recv(1024).decode("utf-8")
42          return message
43
        tabnine: test | explain | document | ask
44      def write(self, msg: str):
45          self.client_socket.send(msg.encode("utf-8"))
46
```

Hình 5: Client của game Caro

```python
class Game:
    tabnine: test | explain | document | ask
    def __init__(self):
        self.screen = pygame.display.set_mode((WIDTH, HEIGHT))
        pygame.display.set_caption("Tic-Tac-Toe with Chat")
        self.chat_object = SocketChat()
        self.chat_object.client_socket.connect((self.chat_object.IP, self.chat_object.PORT))
        self.player = self.chat_object.receive()
        self.turn = "X"
        self.other_player = "O" if self.player == "X" else "X"
        self.board = [["" for _ in range(GRID_SIZE)] for _ in range(GRID_SIZE)]
        self.chat_messages = []
        self.chat_input = []
        self.chat_scroll_y = 0
        self.running = True
        self.show_chat = True  # Default to show the chat box
        self.game_over = False  # Variable to track game over state

        threading.Thread(target=self.handle_incoming_messages).start()
```

Hình 6: Client của game Caro

```python
    tabnine: test | explain | document | ask
    def handle_incoming_messages(self):
        while self.running:
            message = self.chat_object.receive()
            if message.startswith("CHAT:"):
                self.chat_messages.append(message[5:])
            elif message.startswith("WINNER:"):
                winner = message.split(":")[1]
                self.show_winner_message(winner)
                self.reset_game()
            else:
                i, j = map(int, message.split(" "))
                self.board[i][j] = self.turn
                self.toggle_turn()
                self.check_winner()

    tabnine: test | explain | document | ask
    def draw_grid(self):
        for x in range(0, WIDTH, CELL_SIZE):
            pygame.draw.line(self.screen, BLACK, (x, 0), (x, WIDTH))
        for y in range(0, WIDTH, CELL_SIZE):
            pygame.draw.line(self.screen, BLACK, (0, y), (WIDTH, y))

    tabnine: test | explain | document | ask
    def draw_board(self):
        for i in range(GRID_SIZE):
            for j in range(GRID_SIZE):
                if self.board[i][j] != "":
                    text = FONT.render(self.board[i][j], True, RED if self.board[i][j] == "X" else BLUE)
                    self.screen.blit(text, (j * CELL_SIZE + CELL_SIZE//3, i * CELL_SIZE + CELL_SIZE//4))
```

Hình 7: Client của game Caro

```
tabnine: test | explain | document | ask
def draw_chat_box(self):
    chat_bg_rect = pygame.Rect(0, WIDTH, WIDTH, CHAT_BOX_HEIGHT)
    pygame.draw.rect(self.screen, WHITE, chat_bg_rect)

    # Draw chat input background with border
    chat_input_bg_rect = pygame.Rect(WIDTH // 2 - 200, HEIGHT - 90, 400, 40)
    pygame.draw.rect(self.screen, GREEN, chat_input_bg_rect)  # Background

    # Draw chat input text
    chat_input_text = "".join(self.chat_input[-31:])
    chat_input_surf = CHAT_FONT.render(chat_input_text if chat_input_text else "Chat here", True, WHITE)
    input_text_rect = chat_input_surf.get_rect(center=(WIDTH // 2, HEIGHT - 70))
    self.screen.blit(chat_input_surf, input_text_rect)

    # Hiển thị nội dung của ô chat box với viền
    chat_surface = pygame.Surface((WIDTH - 20, CHAT_BOX_HEIGHT - 40))
    chat_surface.fill(WHITE)
    pygame.draw.rect(chat_surface, BLACK, chat_surface.get_rect(), 2)  # Border
    y_offset = CHAT_BOX_HEIGHT - 60 + self.chat_scroll_y
    for message in reversed(self.chat_messages):
        message_surface = CHAT_FONT.render(message, True, BLACK)
        chat_surface.blit(message_surface, (10, y_offset))
        y_offset -= CHAT_FONT_SIZE + 5
    self.screen.blit(chat_surface, (10, HEIGHT - CHAT_BOX_HEIGHT - 60))
```

Hình 8: Client của game Caro

```
def draw_player_info(self):
    player_color = RED if self.player == "X" else BLUE
    player_info_surface = CHAT_FONT.render(f"Player: {self.player} Turn: {self.turn}", True, player_color)
    self.screen.blit(player_info_surface, (10, WIDTH + 5))

tabnine: test | explain | document | ask
def toggle_turn(self):
    self.turn = "O" if self.turn == "X" else "X"

tabnine: test | explain | document | ask
def check_winner(self):
    winning_states = [
        [(0, 0), (0, 1), (0, 2)],
        [(1, 0), (1, 1), (1, 2)],
        [(2, 0), (2, 1), (2, 2)],
        [(0, 0), (1, 0), (2, 0)],
        [(0, 1), (1, 1), (2, 1)],
        [(0, 2), (1, 2), (2, 2)],
        [(0, 0), (1, 1), (2, 2)],
        [(0, 2), (1, 1), (2, 0)],
    ]
    for state in winning_states:
        if self.board[state[0][0]][state[0][1]] != "" and \
            self.board[state[0][0]][state[0][1]] == self.board[state[1][0]][state[1][1]] == self.board[state[2][0]][stat
            winner = self.board[state[0][0]][state[0][1]]
            self.show_winner_message(winner)  # Call new function to display message
            self.chat_object.write(f"WINNER:{winner}")  # Gửi thông điệp chiến thắng
            pygame.time.wait(1000)
            self.reset_game()
            break
```

Hình 9: Client của game Caro

```
tabnine: test | explain | document | ask
    def reset_game(self):
        self.board = [["" for _ in range(GRID_SIZE)] for _ in range(GRID_SIZE)]
        self.game_over = False  # Reset game over state

tabnine: test | explain | document | ask
    def show_winner_message(self, winner):
        # Create a font for the winner message
        winner_font = pygame.font.SysFont('arial', 80)

        # Create the winner message surface
        winner_text = winner_font.render
        winner_text = winner_font.render(f"{winner} Wins!", True, RED if winner == "X" else BLUE)
        winner_text_rect = winner_text.get_rect(center=(WIDTH // 2, HEIGHT // 2))

        # Draw the winner message surface on top of everything else
        self.screen.blit(winner_text, winner_text_rect)
        pygame.display.flip()  # Update the display immediately

        # Wait for some time before resetting the game (optional)
        pygame.time.wait(3000)  # Wait for 3 seconds
```

Hình 10: Client của game Caro

```
tabnine: test | explain | document | ask
def handle_event(self, event):
    if event.type == pygame.QUIT:
        self.running = False
    elif event.type == pygame.MOUSEBUTTONDOWN:
        if not self.game_over:  # Only handle events if the game is not over
            x, y = event.pos
            if y < WIDTH:
                i, j = y // CELL_SIZE, x // CELL_SIZE
                if self.board[i][j] == "" and self.player == self.turn:
                    self.board[i][j] = self.player
                    self.chat_object.write(f"{i} {j}")
                    self.toggle_turn()
                    self.check_winner()
            elif pygame.Rect(0, WIDTH, WIDTH, CHAT_BOX_HEIGHT).collidepoint(event.pos):
                if 455 <= x <= 505 and HEIGHT - 50 <= y <= HEIGHT - 10:
                    self.chat_object.write(f"CHAT:{self.player}: {''.join(self.chat_input)}")
                    self.chat_messages.append(f"Me: {''.join(self.chat_input)}")
                    self.chat_input = []
                    self.show_chat = True  # Show chat box when sending message
                elif 10 <= x <= 50 and HEIGHT - 50 <= y <= HEIGHT - 10:
                    self.show_chat = not self.show_chat  # Toggle chat box display
    elif event.type == pygame.KEYDOWN:
        if not self.game_over:  # Only handle events if the game is not over
            if event.key == pygame.K_RETURN:
                self.chat_object.write(f"CHAT:{self.player}: {''.join(self.chat_input)}")
                self.chat_messages.append(f"Me: {''.join(self.chat_input)}")
                self.chat_input = []
                self.show_chat = True  # Show chat box when sending message
            elif event.key == pygame.K_BACKSPACE:
                if self.chat_input:
                    self.chat_input.pop()
            else:
                self.chat_input.append(event.unicode)
```

Hình 11: Client của game Caro

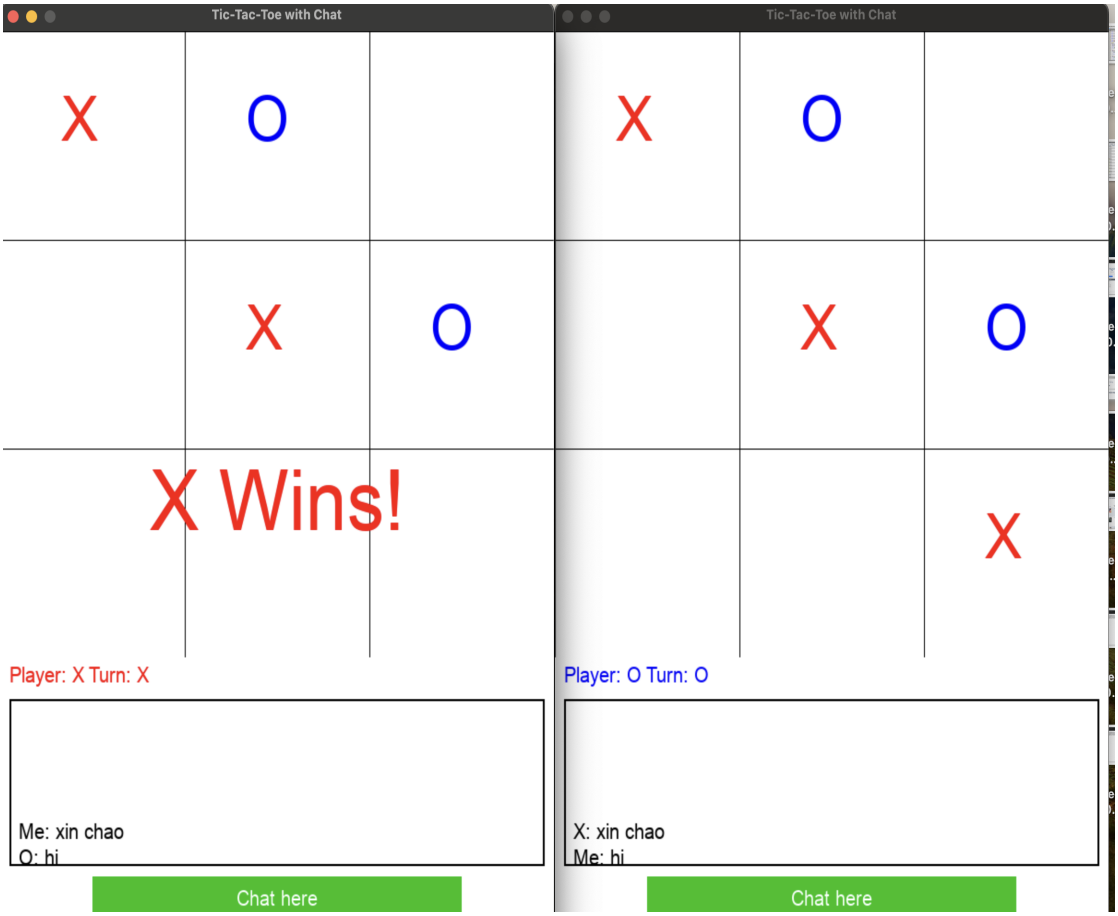
```
204     def run(self):
205         while self.running:
206             self.screen.fill(WHITE)
207             self.draw_grid()
208             self.draw_board()
209             self.draw_chat_box()  # Move chat box drawing to the top
210             self.draw_player_info()
211 
212             for event in pygame.event.get():
213                 self.handle_event(event)
214             pygame.display.flip()
215         pygame.quit()
216         self.chat_object.client_socket.close()
217 
218 
219 if __name__ == "__main__":
220     game = Game()
221     game.run()
222 
```

Hình 12: Client của game Caro

2.2.3 Giao diện game

Hình 13: Giao diện của game Caro

2.2.4 Giao diện game khi thắng cuộc



Hình 14: Giao diện của game Caro khi người chơi thắng

3 PHẦN KẾT LUẬN

Trong tài liệu này, chúng ta đã tìm hiểu về cách áp dụng lập trình socket vào việc phát triển tính năng multiplayer cho game Caro. Từ việc giới thiệu về trò chơi, các yếu tố cơ bản và các thuật ngữ phổ biến, đến việc áp dụng lập trình socket để tạo ra tính năng multiplayer, chúng ta đã có cái nhìn tổng quan về quá trình phát triển một chế độ chơi multiplayer cho game Caro.

3.1 Tóm tắt kết quả đạt được

Thông qua nghiên cứu và thực hành, chúng ta đã đạt được những kết quả sau:

- Hiểu rõ về cách hoạt động của trò chơi Caro và các yếu tố quan trọng của nó.
- Áp dụng thành công lập trình socket để tạo ra tính năng multiplayer cho game Caro.
- Xây dựng được một chế độ chơi multiplayer cho game Caro, cho phép hai người chơi kết nối với nhau qua mạng Internet và chơi cùng một trận đấu.

3.2 Đánh giá và hạn chế

Mặc dù đã đạt được tính năng multiplayer cho game Caro, vẫn còn một số hạn chế cần lưu ý:

- Tính năng multiplayer vẫn còn ở mức cơ bản và có thể được mở rộng thêm để cải thiện trải nghiệm người chơi.
- Chưa có tính năng tương tác phong phú giữa các người chơi, như chat hoặc gửi lời mời chơi.
- Cần kiểm tra và tối ưu hóa tính năng multiplayer để đảm bảo ổn định và mượt mà trong quá trình chơi.

3.3 Hướng nghiên cứu và phát triển trong tương lai

Dựa trên những kết quả và hạn chế đã nêu, có một số hướng nghiên cứu và phát triển trong tương lai:

- Phát triển tính năng tương tác phong phú giữa các người chơi, như chat trong game và gửi lời mời chơi, để tạo ra trải nghiệm giao tiếp tốt hơn.
- Tối ưu hóa tính năng multiplayer để đảm bảo độ ổn định và trải nghiệm chơi mượt mà, đặc biệt là khi kết nối qua mạng Internet không ổn định.
- Nâng cấp giao diện người dùng và đồ họa của game Caro để mang lại trải nghiệm hấp dẫn và thú vị hơn cho người chơi.
- Khám phá các tính năng mới và các chế độ chơi khác nhau, như chế độ giải đấu hoặc chế độ chơi với máy tính, để tăng tính đa dạng và giá trị gia tăng cho game.
- Tiếp tục nghiên cứu và áp dụng các công nghệ mới như trí tuệ nhân tạo (AI) để cải thiện trí tuệ của đối thủ trong game, làm cho trận đấu trở nên thú vị và thách thức hơn.

3.4 Lời cảm ơn

Chúng tôi xin gửi lời cảm ơn đến tất cả những ai đã hỗ trợ và đóng góp vào quá trình nghiên cứu và phát triển của tài liệu này. Đặc biệt, xin cảm ơn các giảng viên, bạn bè và gia đình đã luôn động viên và cung cấp nguồn động viên cho dự án.