

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	2
1.1 Lý do chọn đề tài	2
1.2 Đối tượng nghiên cứu	2
1.3 Phạm vi nghiên cứu	2
1.4 Mục tiêu nghiên cứu	2
2 PHẦN NỘI DUNG	3
2.1 LÝ THUYẾT	3
2.1.1 Tổng quan về Pygame	3
2.1.1.a Khái niệm	3
2.1.1.b Cách cài đặt Pygame trong Python	3
2.1.2 Giới thiệu về lập trình socket	3
2.1.2.a Khái niệm về socket	3
2.1.2.b Cách hoạt động của socket	3
2.1.2.c Lập trình socket trong Python	3
2.1.2.d Ứng dụng của lập trình socket trong game Caro	5
2.1.3 Giới thiệu về game Caro	5
2.1.3.a Lịch sử và sự phát triển	5
2.1.3.b Các yếu tố cơ bản của game Caro	5
2.1.3.c Các thuật ngữ phổ biến	5
2.1.3.d Multiplayer và tương tác mạng	5
2.2 THIẾT KẾ ỨNG DỤNG	6
2.2.1 Tạo server cho game cờ Caro	6
2.2.2 Tạo client cho game cờ Caro	8
2.2.3 Giao diện game khi chạy game	13
2.2.4 Giao diện game	13
2.2.5 Giao diện game khi thắng cuộc	15
3 PHẦN KẾT LUẬN	16
3.1 Tóm tắt kết quả đạt được	16
3.2 Dánh giá và hạn chế	16
3.3 Hướng nghiên cứu và phát triển trong tương lai	16
3.4 Lời cảm ơn	17



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 năm 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 năm 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 năm 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

```
1 import socket
2 import threading
3 import sys
4
5
6 # Server IP and port
7 IP = "192.168.1.2"
8 PORT = 5555
9
10 turn = "X"
11
12
13 # Socket type and options
14 server_socket = socket.socket(family=socket.AF_INET, type=socket.SOCK_STREAM)
15 server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
16
17 server_socket.bind((IP, PORT))
18 server_socket.listen()
19
20 # dictionary of client sockets and their nicknames
21 clients = {}
22
23 # debugging
24 print(f"Listening for connections on {IP}:{PORT}...")
```

Hình 1: Server cho game Caro



```
26
27     # Sending Messages To All Connected Clients
28     tabnine: test | explain | document | ask
29     def broadcast(message, client_socket):
30         # Send messages to all clients except to the original sender
31         for client in clients.keys():
32             if client is not client_socket:
33                 client.send(message.encode("utf-8"))
34
35     # Trong phương thức handle
36     tabnine: test | explain | document | ask
37     def handle(client_socket):
38         while True:
39             message = client_socket.recv(1024).decode("utf-8")
40             print(message)
41             if "exit" in message:
42                 client_socket.close()
43                 broadcast("{} left!".format(clients[client_socket]), client_socket)
44                 clients.pop(client_socket)
45                 print(clients)
46                 sys.exit()
47             elif "WINNER:" in message: # Kiểm tra nếu là thông điệp chiến thắng
48                 broadcast(message, client_socket) # Gửi lại thông điệp này cho tất cả các client
49             else:
50                 broadcast(message, client_socket)
```

Hình 2: Server cho game Caro

```
50
51     # Receiving / Listening Function
52     tabnine: test | explain | document | ask
53     def receive():
54         global turn
55         while True:
56             # Accept Connection
57             client_socket, address = server_socket.accept()
58             print("Connected with {}".format(str(address)))
59             client_socket.send(turn.encode('utf-8'))
60             nickname = f'player {turn}'
61             if turn == "X":
62                 turn = "0"
63             elif turn == "0":
64                 turn = "X"
65             # Add client info to the dictionary
66             clients.update({client_socket: nickname})
67             # Start Handling Thread For Client
68             thread = threading.Thread(target=handle, args=(client_socket,))
69             thread.start()
70
71     receive()
```

Hình 3: Server cho game Caro



2.2.2 Tạo client cho game cờ Caro

```
1 import sys
2 import socket
3 import threading
4 import pygame
5
6 # Initialize Pygame
7 pygame.init()
8
9 # Screen dimensions
10 WIDTH, HEIGHT = 500, 900
11 GRID_SIZE = 10
12 CELL_SIZE = WIDTH // GRID_SIZE
13 FONT_SIZE = 60
14 CHAT_FONT_SIZE = 20
15 CHAT_BOX_HEIGHT = 300 # Adjusted height for chat box
16
17 # Colors
18 GREEN = (1, 101, 179)
19 WHITE = (255, 255, 255)
20 BLACK = (0, 0, 0)
21 RED = (255, 0, 0)
22 BLUE = (0, 0, 255)
23
24 # Fonts
25 FONT = pygame.font.SysFont('Comic Sans MS', FONT_SIZE)
26 CHAT_FONT = pygame.font.SysFont('Comic Sans MS', CHAT_FONT_SIZE)
27
```

Hình 4: Client của game Caro



```
28 class SocketChat:  
tabnine: test | explain | document | ask  
29     def __init__(self):  
30         self.IP = "localhost"  
31         self.PORT = 5555  
32         self.client_socket = socket.socket(family=socket.AF_INET, type=socket.SOCK_STREAM)  
33  
34     #receive data from server  
tabnine: test | explain | document | ask  
35     def receive(self):  
36         try:  
37             message = self.client_socket.recv(1024).decode("utf-8")  
38             return message  
39         except OSError as e:  
40             print("Error receiving data:", e)  
41             return None  
42     #send data  
tabnine: test | explain | document | ask  
43     def write(self, msg: str):  
44         self.client_socket.send(msg.encode("utf-8"))  
45
```

Hình 5: Client của game Caro



```
46  class Game:
47      tabnine: test | explain | document | ask
48      def __init__(self, name):
49          self.screen = pygame.display.set_mode((WIDTH, HEIGHT))
50          pygame.display.set_caption("Tic-Tac-Toe with Chat")
51          self.chat_object = SocketChat()
52          self.chat_object.client_socket.connect((self.chat_object.IP, self.chat_object.PORT))
53          self.chat_object.client_socket.write(f"NAME:{name}") # Send the player's name to the server
54          self.player = self.chat_object.receive()
55          self.name = name # Store the player's name
56          self.turn = "X"
57          self.other_player = "O" if self.player == "X" else "X"
58          self.board = ["" for _ in range(GRID_SIZE)] for _ in range(GRID_SIZE)]
59          self.chat_messages = []
60          self.chat_input = []
61          self.chat_scroll_y = 0
62          self.running = True
63          self.show_chat = True
64          self.game_over = False
65          #start new thread to receive messages from the server
66          threading.Thread(target=self.handle_incoming_messages).start()
67
68      tabnine: test | explain | document | ask
69      def handle_incoming_messages(self):
70          while self.running:
71              message = self.chat_object.receive() # receive message from server
72              if message.startswith("CHAT:"):
73                  self.chat_messages.append(message[5:])
74              elif message.startswith("WINNER:"):
75                  winner = message.split(":")[1]
76                  self.show_winner_message(winner)
77                  self.reset_game()
78              else:
79                  i, j = map(int, message.split(" "))
80                  self.board[i][j] = self.turn
81                  self.toggle_turn()
82                  self.check_winner()
```

Hình 6: Client của game Caro

```
82      tabnine: test | explain | document | ask
83      def draw_grid(self):
84          for x in range(0, WIDTH, CELL_SIZE):
85              pygame.draw.line(self.screen, BLACK, (x, 0), (x, WIDTH))
86          for y in range(0, WIDTH, CELL_SIZE):
87              pygame.draw.line(self.screen, BLACK, (0, y), (WIDTH, y))
88
89      tabnine: test | explain | document | ask
90      def draw_board(self):
91          for i in range(GRID_SIZE):
92              for j in range(GRID_SIZE):
93                  if self.board[i][j] != "":
94                      if self.board[i][j] == "X":
95                          text = FONT.render("X", True, RED)
96                      else:
97                          text = FONT.render("O", True, BLUE)
98                      text_rect = text.get_rect(center=(j * CELL_SIZE + CELL_SIZE // 2, i * CELL_SIZE + CELL_SIZE // 2))
99                      self.screen.blit(text, text_rect)
```

Hình 7: Client của game Caro



```
tabnine: test | explain | document | ask
199     def draw_chat_box(self):
200         chat_bg_rect = pygame.Rect(0, WIDTH, WIDTH, CHAT_BOX_HEIGHT)
201         pygame.draw.rect(self.screen, WHITE, chat_bg_rect)
202
203         chat_input_bg_rect = pygame.Rect(WIDTH // 2 - 200, HEIGHT - 90, 400, 40)
204         pygame.draw.rect(self.screen, GREEN, chat_input_bg_rect)
205
206         chat_input_text = "".join(self.chat_input[-31:])
207         chat_input_surf = CHAT_FONT.render(chat_input_text if chat_input_text else "Chat here", True, WHITE)
208         input_text_rect = chat_input_surf.get_rect(center=(WIDTH // 2, HEIGHT - 70))
209         self.screen.blit(chat_input_surf, input_text_rect)
210
211         chat_surface = pygame.Surface((WIDTH - 20, CHAT_BOX_HEIGHT - 40))
212         chat_surface.fill(WHITE)
213         pygame.draw.rect(chat_surface, BLACK, chat_surface.get_rect(), 2)
214         y_offset = CHAT_BOX_HEIGHT - 70 + self.chat_scroll_y
215         for message in reversed(self.chat_messages):
216             message_surface = CHAT_FONT.render(message, True, BLACK)
217             chat_surface.blit(message_surface, (10, y_offset))
218             y_offset -= CHAT_FONT_SIZE + 5
219         self.screen.blit(chat_surface, (10, HEIGHT - CHAT_BOX_HEIGHT - 60))
220
221 tabnine: test | explain | document | ask
222     def draw_player_info(self):
223         player_color = RED if self.player == "X" else BLUE
224         player_info_surface = CHAT_FONT.render(f"Player: {self.player} ({self.name}) Turn: {self.turn}", True, player_color)
225         self.screen.blit(player_info_surface, (10, WIDTH + 5))
226
227         pygame.draw.rect(self.screen, GREEN, (WIDTH - 110, WIDTH + 5, 100, 35))
228         restart_text = CHAT_FONT.render("Restart", True, WHITE)
229         self.screen.blit(restart_text, (WIDTH - 100, WIDTH + 8))
230
```

Hình 8: Client của game Caro

```
tabnine: test | explain | document | ask
130     def toggle_turn(self):
131         self.turn = "0" if self.turn == "X" else "X"
132
133     def check_winner(self):
134         winning_states = []
135         # Horizontal winning states
136         for i in range(GRID_SIZE):
137             for j in range(GRID_SIZE - 4):
138                 winning_states.append([(i, j + k) for k in range(5)])
139
140         # Vertical winning states
141         for i in range(GRID_SIZE - 4):
142             for j in range(GRID_SIZE):
143                 winning_states.append([(i + k, j) for k in range(5)])
144
145         # Diagonal winning states (top-left to bottom-right)
146         for i in range(GRID_SIZE - 4):
147             for j in range(4, GRID_SIZE):
148                 winning_states.append([(i + k, j + k) for k in range(5)])
149
150         # Diagonal winning states (top-right to bottom-left)
151         for i in range(GRID_SIZE - 4):
152             for j in range(4, GRID_SIZE):
153                 winning_states.append([(i + k, j - k) for k in range(5)])
154
155         for state in winning_states:
156             symbols = [self.board[i][j] for i, j in state]
157             if symbols[0] != "" and all(symbol == symbols[0] for symbol in symbols):
158                 winner = symbols[0]
159                 self.show_winner_message(winner)
160                 self.chat_object.write(f"WINNER:{winner}")
161                 pygame.time.wait(1000)
162                 self.reset_game()
163                 break
```

Hình 9: Client của game Caro

```
tabnine: test | explain | document | ask
165
166     def reset_game(self):
167         self.board = [ [" " for _ in range(GRID_SIZE)] for _ in range(GRID_SIZE)]
168         self.game_over = False
169
170     tabnine: test | explain | document | ask
171     def show_winner_message(self, winner):
172         winner_font = pygame.font.SysFont('Comic Sans MS', 80)
173         winner_text = winner_font.render(f'{winner} Wins!', True, RED if winner == "X" else BLUE)
174         winner_text_rect = winner_text.get_rect(center=(WIDTH // 2, HEIGHT // 2))
175         self.screen.blit(winner_text, winner_text_rect)
176         pygame.display.flip()
177         pygame.time.wait(3000)
178
179     tabnine: test | explain | document | ask
180     def handle_event(self, event):
181         if event.type == pygame.QUIT:
182             self.running = False
183         elif event.type == pygame.MOUSEBUTTONDOWN:
184             if not self.game_over:
185                 x, y = event.pos
186                 if WIDTH - 110 <= x <= WIDTH - 10 and WIDTH + 5 <= y <= WIDTH + 45:
187                     self.reset_game() # Thiết lập lại trò chơi khi nhấn nút chơi lại
188                 if y < WIDTH:
189                     i, j = y // CELL_SIZE, x // CELL_SIZE
190                     if self.board[i][j] == "" and self.player == self.turn:
191                         self.board[i][j] = self.player
192                         self.chat_object.write(f'{i} {j}')
193                         self.toggle_turn()
194                         self.check_winner()
195
196             elif pygame.Rect(0, WIDTH, WIDTH, CHAT_BOX_HEIGHT).collidepoint(event.pos):
197                 if 10 <= x <= WIDTH - 10 and HEIGHT - 90 <= y <= HEIGHT - 50:
198                     self.show_chat = True
199                 elif 455 <= x <= 505 and HEIGHT - 50 <= y <= HEIGHT - 10:
200                     self.chat_object.write(f'CHAT:{self.name}: {"'.join(self.chat_input)}')
201                     self.chat_messages.append(f'Me: {"'.join(self.chat_input)}')
202                     self.chat_input = []
203                     self.show_chat = True
```

Hình 10: Client của game Caro

```
199         self.show_chat = True
200     elif event.type == pygame.KEYDOWN:
201         if not self.game_over and self.show_chat:
202             if event.key == pygame.K_RETURN:
203                 self.chat_object.write(f"CHAT:{self.name}: {''.join(self.chat_input)}")
204                 self.chat_messages.append(f"Me: {''.join(self.chat_input)}")
205                 self.chat_input = []
206             elif event.key == pygame.K_BACKSPACE:
207                 if self.chat_input:
208                     self.chat_input.pop()
209             else:
210                 self.chat_input.append(event.unicode)
211
212
213     tabnine: test | explain | document | ask
214     def run(self):
215         while self.running:
216             self.screen.fill(WHITE)
217             self.draw_grid()
218             self.draw_board()
219             self.draw_chat_box()
220             self.draw_player_info()
221
222             for event in pygame.event.get():
223                 self.handle_event(event)
224             pygame.display.flip()
225
226         pygame.quit()
227         self.chat_object.client_socket.close()
```

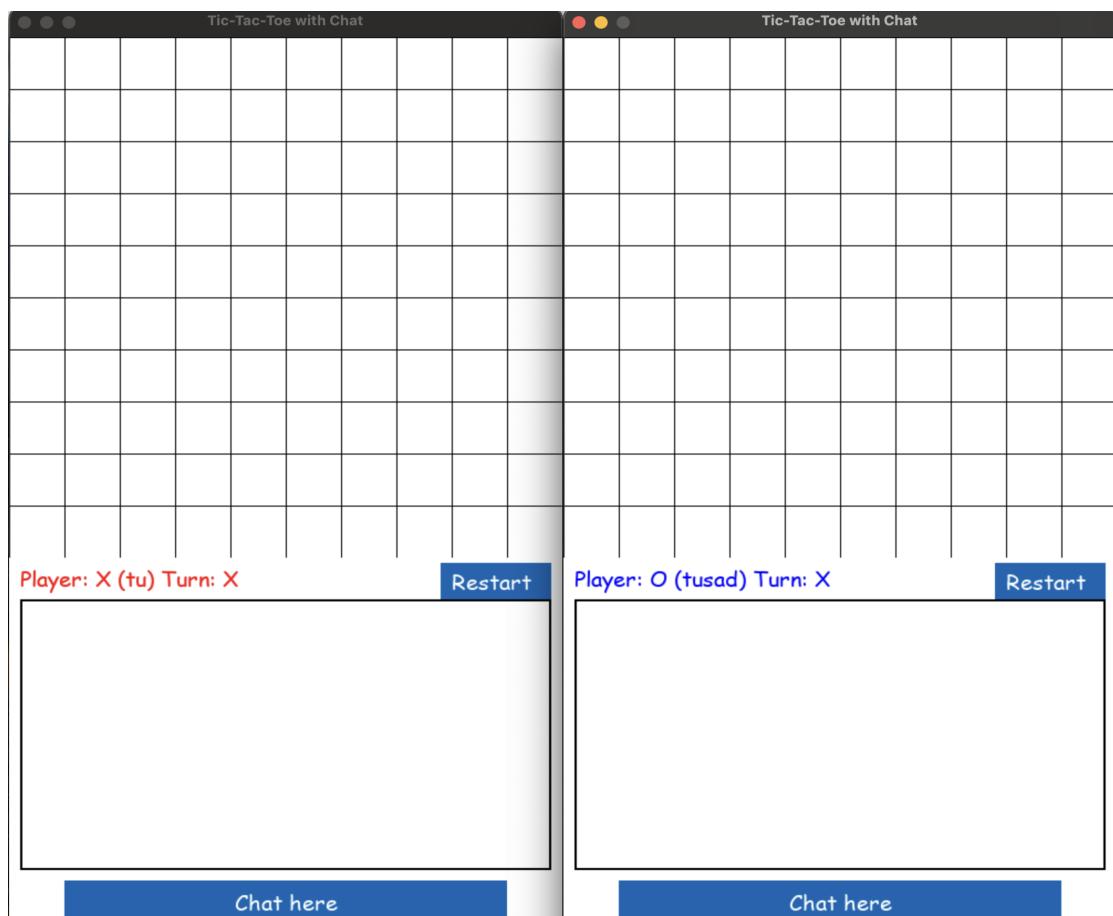
Hình 11: Client của game Caro

2.2.3 Giao diện game khi chạy game



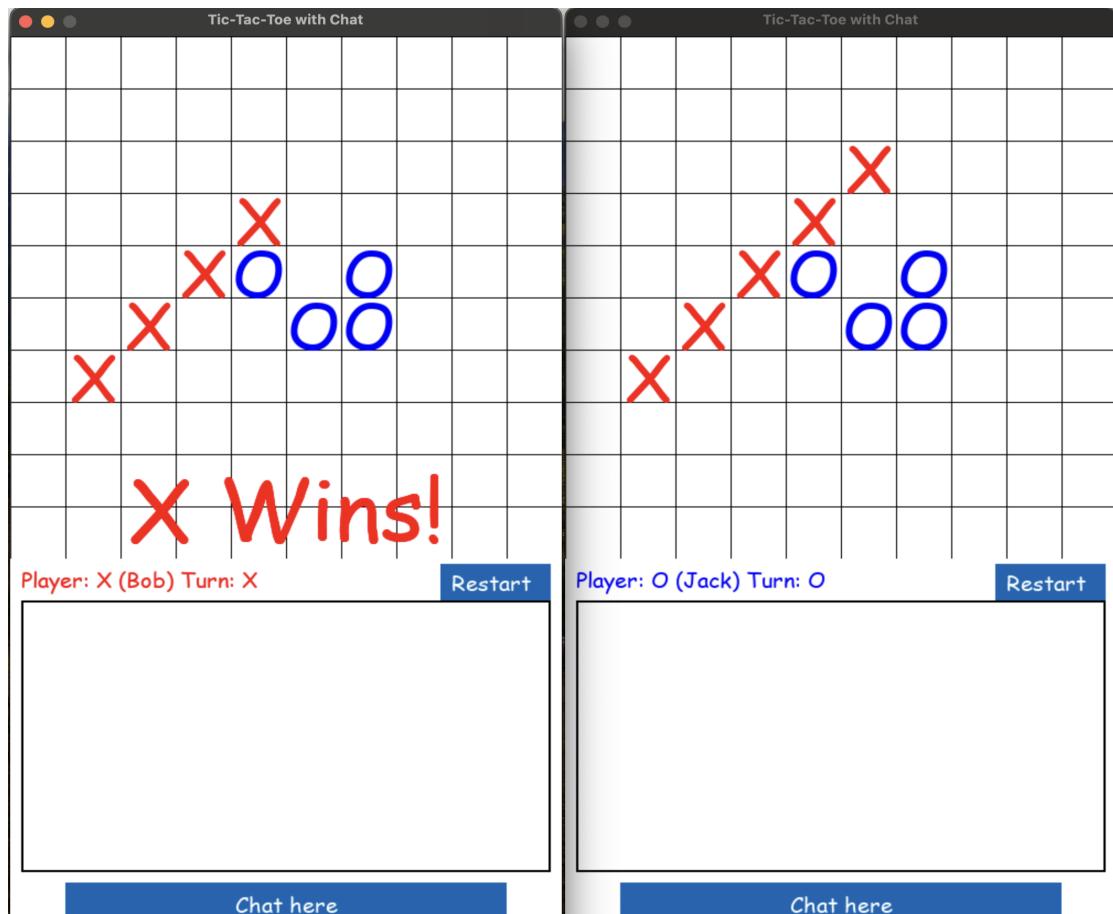
Hình 12: Giao diện của game Caro

2.2.4 Giao diện game



Hình 13: Giao diện của game Caro

2.2.5 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

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 Dánh 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.