TRƯỜNG ĐẠI HỌC SÀI GÒN
KHOA CÔNG NGHỆ THÔNG TIN

PHÁT TRIỂN PHẦN MỀM MÃ NGUỒN MỞ

Xây dựng ứng dụng game "Caro Online"

với thư viện Socket bằng ngôn ngữ lập trình Python

GVHD: Từ Lãng Phiêu
SV: Nguyễn Chí Tài - 3121410433
Lương Ngọc Tâm - 3121410437

TP. HỒ CHÍ MINH, THÁNG 5/2024

NHẬN XÉT, ĐÁNH GIÁ CỦA GIẢNG VIÊN

LỜI CẢM ƠN

Chúng em xin bày tỏ lòng biết ơn và sự tri ân chân thành đến toàn thể thầy cô tại Trường Đại học Sài Gòn, đặc biệt là các thầy cô tại Khoa Công Nghệ Thông Tin, vì đã tạo điều kiện cho chúng em được tiếp cận và nghiên cứu để hoàn thành đồ án môn học này. Đồng thời, chúng em cũng muốn bày tỏ lòng biết ơn đến thầy Từ Lăng Phiêu đã truyền đạt những kiến thức quan trọng làm nền tảng cho em hoàn thành đồ án này.

Trong quá trình nghiên cứu và viết báo cáo đồ án, chúng em nhận thấy rằng còn tồn tại nhiều hạn chế về kiến thức và kinh nghiệm thực tế, dẫn đến sự xuất hiện một số thiếu sót trong bài báo cáo của chúng em. Do đó, chúng em rất mong nhận được ý kiến đóng góp từ các thầy cô để chúng em có thể học hỏi thêm nhiều kỹ năng và kinh nghiệm, từ đó hoàn thiện hơn trong các báo cáo sắp tới.

Một lần nữa, chúng em xin chân thành cảm ơn toàn thể thầy cô đã dành thời gian và tâm huyết để hướng dẫn và giúp đỡ chúng em trong quá trình thực hiện đồ án.

Mục lục

1 MỞ ĐẦU

1.1 Giới thiệu đề tài

Tên đề tài:

Xây dựng ứng dụng trò chơi "Caro Online" với thư viện Socket bằng ngôn ngữ lập trình Python.

1.2 Lý do chọn đề tài.

Việc lựa chọn đề tài xây dựng ứng dụng trò chơi "Caro Online" sử dụng thư viện Socket trong ngôn ngữ lập trình Python được chúng em xem là một sự kết hợp hoàn hảo giữa thú vị của trò chơi và tính ứng dụng của công nghệ hiện đại. Dưới đây là những lý do mà chúng em tin rằng đề tài này đáng được lựa chọn:

Sự phổ biến của trò chơi Caro: Trò chơi Caro là một trong những trò chơi cờ rất phổ biến và được ưa chuộng trên toàn thế giới. Việc xây dựng một phiên bản trực tuyến của trò chơi này không chỉ giúp chúng em thỏa mãn niềm đam mê trong lập trình mà còn mang lại cho người chơi một trải nghiệm giải trí thú vị.

Áp dụng công nghệ Socket: Thư viện Socket trong Python cung cấp các công cụ mạnh mẽ để xây dựng ứng dụng mạng. Việc áp dụng Socket trong việc tạo ra một trò chơi trực tuyến không chỉ giúp chúng em hiểu rõ hơn về cách thức hoạt động của mạng mà còn mở ra cơ hội để nắm bắt các kiến thức và kỹ năng mới trong lập trình mạng.

Khả năng tương tác và kết nối cộng đồng: Bằng cách xây dựng một phiên bản trò chơi Caro trực tuyến, chúng em có thể tạo ra một môi trường tương tác, kết nối giữa các người chơi từ khắp nơi trên thế giới. Điều này không chỉ giúp chúng em rèn luyện kỹ năng lập trình mà còn tạo ra một cơ hội để giao lưu, học hỏi và chia sẻ kiến thức với cộng đồng lập trình viên và người chơi khác.

Thách thức và cơ hội phát triển: Việc xây dựng một ứng dụng trò chơi trực tuyến không phải là một nhiệm vụ dễ dàng, nhưng cũng là một cơ hội tuyệt vời để thử thách và phát triển kỹ năng lập trình của chúng em. Qua quá trình nghiên cứu, thiết kế và triển khai, chúng em có thể đối mặt với nhiều thách thức thú vị và học hỏi được nhiều kỹ năng mới trong quá trình phát triển phần mềm.

Tóm lại, việc lựa chọn đề tài xây dựng ứng dụng trò chơi "Caro Online" với thư viện Socket trong ngôn ngữ lập trình Python không chỉ là một cơ hội để thỏa mãn niềm đam mê trong lập trình mà còn là một hành trình học hỏi và phát triển kỹ năng đầy thú vị và ý nghĩa. Chúng em tin rằng đề tài này sẽ mang lại nhiều giá trị và trải nghiệm mới cho chúng em cũng như cho cộng đồng.

1.3 Mục đích, mục tiêu của đề tài

- Mục đích:
- Nắm chắc được được kỹ năng và kiến thức về lập trình.
- Tìm hiểu về thư viện Socket trong ngôn ngữ lập trình Python.
- Củng cố, áp dụng, nâng cao kiến thức đã được học.
- Nắm bắt được quy trình làm game online cơ bản.
- Mục tiêu:

- Vận dụng được tính chất của lập trình hướng đối tượng.
- Sử dụng thư viện Socket vào việc xây dựng game "Caro Online".

2 NỘI DUNG

2.1 Đôi nét về game “Caro Online”

2.1.1 Giới thiệu về game “Caro Online”

Trò chơi "Caro Online" là một phiên bản trực tuyến của trò chơi Caro, còn được biết đến với tên gọi Gomoku hoặc Five in a Row, là một trò chơi cờ cực kỳ phổ biến và thú vị. Trò chơi này thường được chơi trên một bảng ô vuông với kích thước tiêu chuẩn là 15x15 hoặc 19x19. Hai người chơi luân phiên đánh dấu các ô trên bảng bằng các ký hiệu khác nhau (thường là "X" và "O") với mục tiêu tạo ra một chuỗi gồm năm ký hiệu liên tiếp theo chiều ngang, dọc hoặc chéo trên bảng. Người chơi đầu tiên tạo ra một chuỗi năm quân cờ như vậy sẽ chiến thắng.

Trò chơi Caro Online giúp mang trải nghiệm chơi Caro truyền thống lên một tầm cao mới bằng cách kết nối người chơi từ khắp nơi trên thế giới thông qua Internet. Trong đồ án của chúng em đã phát triển được những tính năng bao gồm:

Tạo host và kết nối: Chơi được online theo mô hình mạng LAN.

Nhắn tin: có thể nhắn tin, trò truyện với nhau thông qua mạng LAN

Cài đặt: Điều chỉnh tiếng âm thanh, hiệu ứng trong trò chơi

2.1.2 Mục tiêu của game “Caro Online”

Mục tiêu của trò chơi "Caro Online" là:

Giải trí và thư giãn: "Caro Online" cung cấp một nền tảng giải trí thú vị cho người chơi, cho phép họ tham gia vào các trận đấu căng thẳng và đầy kích thích với bạn bè hoặc người chơi khác trên toàn thế giới.

Kết nối cộng đồng: Trò chơi "Caro Online" tạo ra một môi trường trực tuyến để các người chơi có thể giao lưu, kết nối và chia sẻ niềm đam mê với nhau. Điều này giúp mở rộng cộng đồng người chơi và tạo ra một không gian thú vị cho sự tương tác xã hội.

Thử thách trí tuệ: Trò chơi "Caro Online" không chỉ là về may mắn mà còn đòi hỏi sự tư duy chiến lược và kỹ năng đánh cờ của người chơi. Việc phải suy nghĩ và dự đoán nước đi của đối thủ là một phần quan trọng của trải nghiệm chơi game này.

Phát triển kỹ năng: Chơi "Caro Online" không chỉ mang lại niềm vui mà còn giúp người chơi phát triển kỹ năng như tư duy logic, quyết định nhanh nhạy và khả năng tương tác xã hội thông qua việc tham gia vào các trận đấu và trò chuyện với đối thủ.

2.2 Yêu cầu đối với đồ án

Xây dựng 1 ứng dụng game “Caro Online” dựa vào những kiến thức đã học về ngôn ngữ lập trình Python. Ứng dụng được thư viện Socket trong ngôn ngữ Python.

2.3 Tiến hành xây dựng ứng dụng

2.3.1 Những thứ cần chuẩn bị.

2.3.1.1 Các hình ảnh của game

Hình 3.1: Logo game

Hình 3.2: Biểu tượng setting

Hình 3.3: Biểu tượng gửi tin nhắn

2.3.1.2 Các file âm thanh của game

2.3.1.3 Các thư viện sử dụng (Tkinter, Socket, Threading, PIL, Pygame)

Tkinter là một thư viện được tích hợp sẵn trong Python, được sử dụng để tạo giao diện người dùng đồ họa (GUI). Nó cung cấp các widget và phương thức để tạo và quản lý các cửa sổ, nút, nhãn, v.v. Trong trò chơi "Caro Online" của chúng em thì Tkinter được sử dụng để tạo giao diện người dùng cho trò chơi Caro.

Ưu điểm: Được tích hợp sẵn trong Python, dễ sử dụng cho các ứng dụng GUI đơn giản, tương thích tốt trên nhiều hệ điều hành.

Nhược điểm: Giao diện không đẹp mắt và linh hoạt như các thư viện GUI phức tạp hơn như PyQt hoặc wxPython.

Socket

Thư viện Socket là một phần của Python cung cấp các giao diện lập trình ứng dụng mạng. Nó cho phép lập trình viên tạo và quản lý các kết nối mạng, truyền gửi dữ liệu qua mạng. Trong trò chơi "Caro Online" của chúng em thì Socket được sử dụng để xây dựng kết nối mạng giữa các client và server trong trò chơi Caro.

Ưu điểm: Cho phép truyền gửi dữ liệu qua mạng một cách linh hoạt, phù hợp cho việc xây dựng các ứng dụng mạng.

Nhược điểm: Yêu cầu hiểu biết sâu về các giao thức mạng, cần quản lý kết nối và xử lý lỗi mạng.

Threading

Thư viện Threading được sử dụng để tạo và quản lý các luồng riêng biệt (threads) trong Python. Trong trò chơi "Caro Online" của chúng em thì Threading được sử dụng để tạo các luồng riêng biệt để xử lý kết nối mạng đến server hoặc client.

Ưu điểm: Cho phép xử lý đồng thời nhiều tác vụ, tăng hiệu suất của ứng dụng khi có nhiều hoạt động cần xử lý cùng một lúc.

Nhược điểm: Cần quản lý các vấn đề liên quan đến đồng bộ hóa và tương tác giữa các luồng.

PIL (Python Imaging Library)

PIL là một thư viện Python để xử lý hình ảnh. Trong trò chơi "Caro Online" của chúng em thì PIL được sử dụng để tải và xử lý các hình ảnh như biểu tượng, logo cho giao diện người dùng.

Ưu điểm: Dễ sử dụng cho các nhiệm vụ xử lý hình ảnh cơ bản, hỗ trợ nhiều định dạng hình ảnh.

Nhược điểm: Thư viện này không được phát triển tích cực, được thay thế bởi Pillow (fork của PIL).

Pygame

Pygame là một thư viện Python để phát triển trò chơi. Nó cung cấp các chức năng để xử lý âm thanh, đồ họa, cũng như các sự kiện trong trò chơi. Trong trò chơi "Caro Online" của chúng em thì Pygame được sử dụng để phát âm thanh khi người dùng thực hiện các hành động trong trò chơi.

Ưu điểm: Mạnh mẽ cho việc phát triển trò chơi, cung cấp nhiều chức năng xử lý âm thanh, đồ họa, sự kiện.

Nhược điểm: Không phù hợp cho các ứng dụng GUI phức tạp, tốn nhiều tài nguyên hơn so với các thư viện GUI khác.

2.3.2 Khai báo thư viện các biến cho trò chơi

- Khai báo các thư viện cần thiết:

```
import tkinter as tk
from functools import partial
import threading
import socket
from tkinter import messagebox
from PIL import Image, ImageTk  # Sử dụng PIL để xử lý hình ảnh
import pygame
```

Những dòng trên dùng để khai báo các thư viện cần thiết:

- tkinter: thư viện tạo giao diện đồ họa.
- functools.partial: giúp tạo các hàm với đối số mặc định.
- threading: hỗ trợ đa luồng, để xử lý socket mà không bị chặn.
- socket: thư viện để làm việc với kết nối mạng.
- PIL (Pillow): thư viện để xử lý hình ảnh
- pygame: thư viện để phát nhạc và âm thanh.
- ***Biến toàn cục***

```
331    if __name__ == "__main__":
332        Ox = 20  # Số lượng ô theo trục X
333        Oy = 20 # Số lượng ô theo trục Y
```

Đặt ở cuối đoạn mã chính, quy định số lượng ô theo trục hoành (X) và trục tung (Y) trong bàn cờ caro.

- ***Biến của lớp Window()***

```
13            self.Buts = {}
```

'self.Buts': dictionary (từ điển) dùng để lưu các nút trên bàn cờ.

```
14            self.memory = []
```

'self.memory': lưu trữ các bước đi để thực hiện chức năng undo.

```
15            self.Threading_socket = Threading_socket(self)
```

'self.Threading_socket': đối tượng của lớp Threading_socket dùng để quản lý kết nối mạng của trò chơi.

```
22            self.effect_volume = 1.0  # Âm lượng hiệu ứng âm thanh
23            self.music_volume = 1.0  # Âm lượng nhạc nền
```

'self.effect_volume': lưu trữ âm lượng âm thanh hiệu ứng.
'selt.music_volume': lưu trữ âm lượng nhạc nền.

```
25            # Tải hình ảnh biểu tượng bánh răng
26            self.settings_icon = ImageTk.PhotoImage(Image.open("assets/setting.png").resize((20, 20)))
27
28            # Tải hình ảnh biểu tượng gửi tin nhắn
29            self.send_icon = ImageTk.PhotoImage(Image.open("assets/send_icon.png").resize((15, 15)))
```

'self.settings_icon': chứa hình ảnh của nút cài đặt.
'self.send_icon': chứa hình ảnh biểu tượng chức năng chat.

```
80            # Tải logo game
81            self.logo = ImageTk.PhotoImage(Image.open("assets/caro_logo2.png").resize((450, 450)))  # Đảm bảo đã có file logo.png
```

```
81            self.logo = ImageTk.PhotoImage(Image.open
```

'self.logo': chứa hình ảnh logo của trò chơi.

```
85            # Thêm khung chat
86            self.chat_display = tk.Text(frame3, height=15, width=50, state=tk.DISABLED, wrap=tk.WORD)
```

'self.chat_display': widget hiển thị khung chat.

```
89            self.chat_entry = tk.Entry(frame3, width=50)  # Đặt độ rộng là 50
```

'self.chat_entry': widget hiển thị khung nhập tin nhắn.

```
253            self.dataReceive = ""
```

- ***Biến của lớp Threading_ socket()***

'self.dataReceive': lưu trữ dữ liệu nhận được từ kết nối mạng.

```
254             self.conn = None
```

'self.conn': lưu trữ đối tượng kết nối socket.

```
255             self.gui = gui
```

'self.gui': tham chiếu tới đối tượng GUI - lớp Window() để tương tác với giao diện.

```
256             self.name = ""
```

'self.name': biến lưu trữ tên của người chơi (client hoặc server).

- *Cài đặt giao diện*

Các thành phần giao diện được cài đặt trong phương thức showFrame()

```
32              frame1 = tk.Frame(self, background="#BDB76B")
33              frame1.grid(row=0, column=0, sticky='nsew')
34
35              frame2 = tk.Frame(self, background="#BDB76B")
36              frame2.grid(row=1, column=0, sticky='nsew')
37
38              frame3 = tk.Frame(self, background="#BDB76B")
39              frame3.grid(row=1, column=1, sticky='nsew')
```

Frame: các khung chứa - 'frame1', 'frame2', 'frame3' được tạo để sắp xếp các thành phần giao diện khác nhau.

```
47          setting_button = tk.Button(frame1, image=self.settings_icon, width=24, height=24, command=self.openSettings)
```

```
47              setting_button = t
```

'setting_button': nút setting để mở giao diện thay đổi âm lượng của hiệu ứng và nhạc nền.

```
51          Undo = tk.Button(frame1, text="Undo", width=10,  # nút quay lại
52                           command=partial(self.Undo, synchronized=True), font=("Cutie Pie", 11, "bold"), foreground="white")
```

```
51              Undo = tk.Button(
```

'Undo': nút undo để xóa các bước vừa đi.

```
63      connectBT = tk.Button(frame1, text="Connect", width=10,
64                            command=lambda: self.Threading_socket.clientAction(inputIp.get()), font=("Cutie Pie", 11, "bold"), foreground="white")
```

```
63          connectBT = tk.Button(
```

'connectBT': nút để kết nối với host thông qua địa chỉ IP.

```
68      makeHostBT = tk.Button(frame1, text="MakeHost", width=10,  # nút tạo host
69                             command=lambda: self.Threading_socket.serverAction(), font=("Cutie Pie", 11, "bold"), foreground="white")
```

```
68          makeHostBT = tk.Button(f
```

'makeHostBT': nút để thiết lập host.

```
73          for x in range(Ox):   # tạo ma trận button Ox * Oy
74              for y in range(Oy):
75                  self.Buts[x, y] = tk.Button(frame2, font=('arial', 15, 'bold'), height=1, width=2,
76                                              borderwidth=2, command=partial(self.handleButton, x=x, y=y))
77                  self.Buts[x, y].grid(row=x, column=y)
78                  self.Buts[x, y].config(background="#fffacd")  # Đặt màu nền cho nút bằng mã màu RGB
```

Khởi tạo bàn cờ caro trong frame2 là một ma trận gồm số dòng là Ox và số cột và Oy.

```
18          # Khởi tạo pygame để phát âm thanh
19          pygame.mixer.init()
20          pygame.mixer.music.load("assets/music_game.mp3")  # Load file nhạc
21          pygame.mixer.music.play(-1)  # Phát nhạc lặp lại vô hạn
```

Âm thanh nền: được khởi tạo và phát lại liên tục bằng pygame.

```
116             # Phát âm thanh khi click vào nút
117             effect_sound = pygame.mixer.Sound("assets/effect_click.wav")
118             effect_sound.set_volume(self.effect_volume)
119             effect_sound.play()
```

Âm thanh hiệu ứng: được phát khi người dùng ấn vào nút trên bàn cờ.

- *Kết nối mạng*

Các phương thức và biến trong Threading_socket() quản lý việc kết nối mạng giữa các người chơi, bao gồm:

```
258     def clientAction(self, inputIP):
```

```
289     def serverAction(self):
```

- 'clientAction' và 'serverAction': Phương thức để khởi tạo kết nối client và server.

```
270     def client(self):
```

```
303     def server(self, addr, s):
```

- 'client' và 'server': Phương thức để xử lý dữ liệu nhận được từ kết nối mạng.

```
327     def sendData(self, data):
```

- 'sendData': Phương thức để gửi dữ liệu qua mạng.

2.3.3 Các phương thức trong class Window()

Lớp Window() trong đoạn mã trên được thiết kế để tạo giao diện đồ họa cho trò chơi cờ caro với nhiều tính năng như phát âm thanh, kết nối mạng và trò chuyện trong trò chơi. Những phương thức này giúp quản lý toàn bộ giao diện và các tính năng của trò chơi cờ caro, bao gồm giao diện người dùng, âm thanh, kết nối mạng, và các thao tác trong trò chơi.

```
9   class Window(tk.Tk):
```

2.3.3.1 *Phương thức __init(self)__*

```
10      def __init__(self):
11          super().__init__()
12          self.title("Caro by Python")
13          self.Buts = {}
14          self.memory = []
15          self.Threading_socket = Threading_socket(self)
16          self.config(background="#BDB76B")
17          print(self.Threading_socket.name)
18          # Khởi tạo pygame để phát âm thanh
19          pygame.mixer.init()
20          pygame.mixer.music.load("assets/music_game.mp3")  # Load file nhạc
21          pygame.mixer.music.play(-1)  # Phát nhạc lặp lại vô hạn
22          self.effect_volume = 1.0  # Âm lượng hiệu ứng âm thanh
23          self.music_volume = 1.0  # Âm lượng nhạc nền
24
25          # Tải hình ảnh biểu tượng bánh răng
26          self.settings_icon = ImageTk.PhotoImage(Image.open("assets/setting.png").resize((20, 20)))
27
28          # Tải hình ảnh biểu tượng gửi tin nhắn
29          self.send_icon = ImageTk.PhotoImage(Image.open("assets/send_icon.png").resize((15, 15)))
30
```

Phương thức khởi tạo của lớp Window, thiết lập giao diện ban đầu và các thành phần của cửa sổ:

- Thiết lập tiêu đề cửa sổ, màu nền và khởi tạo các biến lưu trữ.
- Khởi tạo đối tượng Threading_socket để quản lý kết nối mạng.
- Khởi tạo Pygame để phát nhạc nền và âm thanh.
- Tải các hình ảnh biểu tượng cho nút cài đặt và nút gửi tin nhắn.

2.3.3.2 *Phương thức showFrame(self)*

Thiết lập các khung (frame) và widget trong giao diện chính của trò chơi, bao gồm bảng cờ, các nút chức năng và khung trò chuyện:

```
31      def showFrame(self):
32          frame1 = tk.Frame(self, background="#BDB76B")
33          frame1.grid(row=0, column=0, sticky='nsew')
34
35          frame2 = tk.Frame(self, background="#BDB76B")
36          frame2.grid(row=1, column=0, sticky='nsew')
37
38          frame3 = tk.Frame(self, background="#BDB76B")
39          frame3.grid(row=1, column=1, sticky='nsew')
40
41          self.grid_columnconfigure(0, weight=1)
42          self.grid_rowconfigure(0, weight=0)
43          self.grid_rowconfigure(1, weight=1)
44          self.grid_columnconfigure(1, weight=1)
```

- Tạo và sắp xếp các khung chứa.

```
46    # Nút Setting với hình ảnh biểu tượng bánh răng
47    setting_button = tk.Button(frame1, image=self.settings_icon, width=24, height=24, command=self.openSettings)
48    setting_button.grid(row=0, column=0, padx=10)
49    setting_button.config(background="#006400")  # Đặt màu nền cho nút "Setting"
```

```
51    Undo = tk.Button(frame1, text="Undo", width=10,  # nút quay lại
52                     command=partial(self.Undo, synchronized=True), font=("Cutie Pie", 11, "bold"), foreground="white")
53    Undo.grid(row=0, column=1, padx=10)
54    Undo.config(background="#006400")  # Đặt màu nền cho nút "Undo"
```

```
63  connectBT = tk.Button(frame1, text="Connect", width=10,
64                        command=lambda: self.Threading_socket.clientAction(inputIp.get()), font=("Cutie Pie", 11, "bold"), foreground="white")
65  connectBT.grid(row=0, column=4, padx=3)
66  connectBT.config(background="#006400")  # Đặt màu nền cho nút "Connect"
```

```
68    makeHostBT = tk.Button(frame1, text="MakeHost", width=10,  # nút tạo host
69                           command=lambda: self.Threading_socket.serverAction(), font=("Cutie Pie", 11, "bold"), foreground="white")
70    makeHostBT.grid(row=0, column=5, padx=30)
71    makeHostBT.config(background="#006400")  # Đặt màu nền cho nút "MakeHost"
```

- Tạo và sắp xếp các nút chức năng (Setting, Undo, Connect, MakeHost).

```
73    for x in range(Ox):   # tạo ma trận button Ox * Oy
74        for y in range(Oy):
75            self.Buts[x, y] = tk.Button(frame2, font=('arial', 15, 'bold'), height=1, width=2,
76                                        borderwidth=2, command=partial(self.handleButton, x=x, y=y))
77            self.Buts[x, y].grid(row=x, column=y)
78            self.Buts[x, y].config(background="#fffacd")  # Đặt màu nền cho nút bằng mã màu RGB
```

- Tạo bảng cờ với ma trận các nút.

```
80      # Tải logo game
81      self.logo = ImageTk.PhotoImage(Image.open("assets/caro_logo2.png").resize((450, 450)))
82      logo_label = tk.Label(frame3, image=self.logo, background="#BDB76B")
83      logo_label.grid(row=0, column=0, columnspan=2, pady=10)
```

- Tải và hiển thị logo trò chơi.

```
85      # Thêm khung chat
86      self.chat_display = tk.Text(frame3, height=15, width=50, state=tk.DISABLED, wrap=tk.WORD)
87      self.chat_display.grid(row=1, column=0, padx=5, pady=5, columnspan=2)
88
89      self.chat_entry = tk.Entry(frame3, width=50)  # Đặt độ rộng là 50
90      self.chat_entry.grid(row=2, column=0, padx=0, pady=0)
91      self.chat_entry.bind("<KeyPress>", self.onKeyPress)  # Gán hàm xử lý sự kiện bấm phím
92
93
94      send_button = tk.Button(frame3, image=self.send_icon, width=24, height=24, command=self.sendMessage)
95      send_button.grid(row=2, column=1, padx=0, pady=0)  # Đặt padx và pady thành 5 hoặc một giá trị nhỏ hơn
96      send_button.config(background="#006400")  # Đặt màu nền cho nút "send"
```

- Tạo khung chat và các nút gửi tin nhắn.

2.3.3.3 ***Phương thức onKeyPress(self, event)*** Xử lý sự kiện khi người dùng nhấn phím, cụ thể là phím "Enter" để gửi tin nhắn.

```
98      def onKeyPress(self, event):
99          if event.keysym == "Return":
100             self.sendMessage()
```

2.3.3.4 ***Phương thức sendMessage(self)*** Gửi tin nhắn từ khung nhập liệu tới khung trò chuyện và gửi dữ liệu qua kết nối mạng.

```
102     def sendMessage(self):
103         message = self.chat_entry.get()
104         if message:
105             self.displayMessage("You: " + message)
106             self.Threading_socket.sendData("message|{}".format(message))
107             self.chat_entry.delete(0, tk.END)
108
```

2.3.3.5 ***Phương thức displayMessage(self, message)*** Hiển thị tin nhắn trong khung trò chuyện.

```
109     def displayMessage(self, message):
110         self.chat_display.config(state=tk.NORMAL)
111         self.chat_display.insert(tk.END, message + "\n")
112         self.chat_display.config(state=tk.DISABLED)
113         self.chat_display.yview(tk.END)
```

2.3.3.6 ***Phương thức handleButton(self, x, y)*** Xử lý sự kiện khi người dùng nhấn vào nút trong bảng cờ, cập nhật trạng thái nút và gửi dữ liệu qua kết nối mạng.

```
115     def handleButton(self, x, y):
116         # Phát âm thanh khi click vào nút
117         effect_sound = pygame.mixer.Sound("assets/effect_click.wav")
118         effect_sound.set_volume(self.effect_volume)
119         effect_sound.play()
120         if self.Buts[x, y]['text'] == "": #Kiểm tra ô có ký tự rỗng hay không
121             if self.memory.count([x, y]) == 0:
122                 self.memory.append([x, y])
123             if len(self.memory) % 2 == 1:
124                 self.Buts[x, y]['text'] = 'O'
125                 self.Buts[x, y]['foreground'] = 'blue'  # Đặt màu xanh cho chữ 'O'
126                 self.Threading_socket.sendData("{}|{}|{}|".format("hit", x, y))
127                 if(self.checkWin(x, y, "O")):
128                     self.notification("Winner", "O")
129                     self.newGame()
130             else:
131                 print(self.Threading_socket.name)
132                 self.Buts[x, y]['text'] = 'X'
133                 self.Buts[x, y]['foreground'] = 'red'  # Đặt màu đỏ cho chữ 'X'
134                 self.Threading_socket.sendData("{}|{}|{}|".format("hit", x, y))
135                 if(self.checkWin(x, y, "X")):
136                     self.notification("Winner", "X")
137                     self.newGame()
138
```

- Phát hiệu ứng âm thanh khi người chơi thực hiện các nước đi trên bàn cờ.

```
116         # Phát âm thanh khi click vào nút
117         effect_sound = pygame.mixer.Sound("assets/effect_click.wav")
118         effect_sound.set_volume(self.effect_volume)
119         effect_sound.play()
```

- Xử lý sự kiện khi người chơi thực hiện các nước đi

```
120         if self.Buts[x, y]['text'] == "": #Kiểm tra ô có ký tự rỗng hay không
121             if self.memory.count([x, y]) == 0:
122                 self.memory.append([x, y])
123             if len(self.memory) % 2 == 1:
124                 self.Buts[x, y]['text'] = 'O'
125                 self.Buts[x, y]['foreground'] = 'blue'  # Đặt màu xanh cho chữ 'O
126                 self.Threading_socket.sendData("{}|{}|{}|".format("hit", x, y))
127                 if(self.checkWin(x, y, "O")):
128                     self.notification("Winner", "O")
129                     self.newGame()
130             else:
131                 print(self.Threading_socket.name)
132                 self.Buts[x, y]['text'] = 'X'
133                 self.Buts[x, y]['foreground'] = 'red'  # Đặt màu đỏ cho chữ 'X'
134                 self.Threading_socket.sendData("{}|{}|{}|".format("hit", x, y))
135                 if(self.checkWin(x, y, "X")):
136                     self.notification("Winner", "X")
137                     self.newGame()
```

- Kiểm tra tính hợp lệ của một ô khi người chơi thực hiện nước đi. Ô chỉ có thể được đánh dấu khi rỗng.

```
120         if self.Buts[x, y]['text'] == "": #Kiểm tra ô có ký tự rỗng hay không
121             if self.memory.count([x, y]) == 0:
122                 self.memory.append([x, y])
```

- Xử lý sự kiện và cài đặt cho 'O'.

```
123         if len(self.memory) % 2 == 1:
124             self.Buts[x, y]['text'] = 'O'
125             self.Buts[x, y]['foreground'] = 'blue'  # Đặt màu xanh cho chữ 'O'
126             self.Threading_socket.sendData("{}|{}|{}|".format("hit", x, y))
127             if(self.checkWin(x, y, "O")):
128                 self.notification("Winner", "O")
129                 self.newGame()
```

- Xử lý khi người chơi 'O' thắng

```
127                 if(self.checkWin(x, y, "O")):
128                     self.notification("Winner", "O")
129                     self.newGame()
```

- Xử lý sự kiện và cài đặt cho 'X'.

```
130         else:
131             print(self.Threading_socket.name)
132             self.Buts[x, y]['text'] = 'X'
133             self.Buts[x, y]['foreground'] = 'red'  # Đặt màu đỏ cho chữ 'X'
134             self.Threading_socket.sendData("{}|{}|{}|".format("hit", x, y))
135             if(self.checkWin(x, y, "X")):
136                 self.notification("Winner", "X")
137                 self.newGame()
138
```

- Xử lý khi người chơi 'X' thắng.

```
135             if(self.checkWin(x, y, "X")):
136                 self.notification("Winner", "X")
137                 self.newGame()
```

2.3.3.7 ***Phương thức openSettings(self)*** Mở cửa sổ cài đặt âm lượng cho nhạc nền và hiệu ứng âm thanh.

```
139 def openSettings(self):
140     settings_window = tk.Toplevel(self)
141     settings_window.title("Settings")
142     settings_window.config(background="#BDB76B")
143
144     music_frame = tk.Frame(settings_window, background="#BDB76B")
145     music_frame.pack(pady=10)
146
147     tk.Label(music_frame, text="Music Volume", background="#BDB76B", font=("Arial", 12)).pack(side=tk.LEFT, padx=5)
148     music_volume_slider = tk.Scale(music_frame, from_=0, to=1, resolution=0.1, orient="horizontal", command=self.setMusicVolume,
149                                    troughcolor="#0000FF", sliderrelief='flat')
150     music_volume_slider.set(self.music_volume)
151     music_volume_slider.pack(side=tk.LEFT, padx=5)
152
153     effect_frame = tk.Frame(settings_window, background="#BDB76B")
154     effect_frame.pack(pady=10)
155
156     tk.Label(effect_frame, text="Effect Volume", background="#BDB76B", font=("Arial", 12)).pack(side=tk.LEFT, padx=5)
157     effect_volume_slider = tk.Scale(effect_frame, from_=0, to=1, resolution=0.1, orient="horizontal", command=self.setEffectVolume,
158                                    troughcolor="#0000FF", sliderrelief='flat')  # Màu xanh dương
159     effect_volume_slider.set(self.effect_volume)
160     effect_volume_slider.pack(side=tk.LEFT, padx=5)
161
162     close_button = tk.Button(settings_window, text="Close", command=settings_window.destroy, font=("Cutie Pie", 11, "bold"), foreground="white")
163     close_button.pack(pady=10)
164     close_button.config(background="#ff0000")  # Đặt màu nền cho nút "close"
```

- Mở cửa sổ cài đặt.

```
140         settings_window = tk.Toplevel(self)
141         settings_window.title("Settings")
142         settings_window.config(background="#BDB76B")
```

- Tạo tiêu đề và thanh điều chỉnh âm lượng cho nhạc nền.

```
144     music_frame = tk.Frame(settings_window, background="#BDB76B")
145     music_frame.pack(pady=10)
146
147     tk.Label(music_frame, text="Music Volume", background="#BDB76B", font=("Arial", 12)).pack(side=tk.LEFT, padx=5)
148     music_volume_slider = tk.Scale(music_frame, from_=0, to=1, resolution=0.1, orient="horizontal", command=self.setMusicVolume,
149                                    troughcolor="#0000FF", sliderrelief='flat')
150     music_volume_slider.set(self.music_volume)
151     music_volume_slider.pack(side=tk.LEFT, padx=5)
```

- Tạo tiêu đề và thanh điều chỉnh âm lượng cho hiệu ứng âm thanh.

```
153     effect_frame = tk.Frame(settings_window, background="#BDB76B")
154     effect_frame.pack(pady=10)
155
156     tk.Label(effect_frame, text="Effect Volume", background="#BDB76B", font=("Arial", 12)).pack(side=tk.LEFT, padx=5)
157     effect_volume_slider = tk.Scale(effect_frame, from_=0, to=1, resolution=0.1, orient="horizontal", command=self.setEffectVolume,
158                                     troughcolor="#0000FF", sliderrelief='flat')  # Màu xanh dương
159     effect_volume_slider.set(self.effect_volume)
160     effect_volume_slider.pack(side=tk.LEFT, padx=5)
```

- Tạo nút đóng cửa sổ setting.

```
162     close_button = tk.Button(settings_window, text="Close", command=settings_window.destroy, font=("Cutie Pie", 11, "bold"), foreground="white")
163     close_button.pack(pady=10)
164     close_button.config(background="#ff0000")  # Đặt màu nền cho nút "close"
```

2.3.3.8 *Phương thức setMusicVolume(self, volume)*

Thiết lập âm lượng của nhạc nền.

```
168 def setMusicVolume(self, volume):
169     self.music_volume = float(volume)
170     pygame.mixer.music.set_volume(self.music_volume)
171
```

2.3.3.9 *Phương thức setEffectVolume(self, volume)*

Thiết lập âm lượng của hiệu ứng âm thanh.

```
172 def setEffectVolume(self, volume):
173     self.effect_volume = float(volume)
```

2.3.3.10 *Phương thức Undo(self, synchronized=False)*

Xử lý chức năng Undo, quay lại bước đi trước đó.

```
232     def Undo(self, synchronized):
233         if(len(self.memory) > 0):
234             x = self.memory[len(self.memory) - 1][0]
235             y = self.memory[len(self.memory) - 1][1]
236             # print(x,y)
237             self.Buts[x, y]['text'] = ""
238             self.memory.pop()
239             if synchronized == True:
240                 self.Threading_socket.sendData("{}|".format("Undo"))
241             print(self.memory)
242         else:
243             print("No character")
```

2.3.3.11 ***Phương thức checkWin(self, x, y, player)*** Kiểm tra xem người chơi có thắng không sau khi thực hiện bước đi.

- Kiểm tra theo dòng:

```
179         count = 0
180         i, j = x, y
181         while(j < Ox and self.Buts[i, j]["text"] == XO):
182             count += 1
183             j += 1
184         j = y
185         while(j >= 0 and self.Buts[i, j]["text"] == XO):
186             count += 1
187             j -= 1
188         if count >= 6:
189             return True
```

- Kiểm tra theo cột:

```
190         # check cột
191         count = 0
192         i, j = x, y
193         while(i < Oy and self.Buts[i, j]["text"] == XO):
194             count += 1
195             i += 1
196         i = x
197         while(i >= 0 and self.Buts[i, j]["text"] == XO):
198             count += 1
199             i -= 1
200         if count >= 6:
201             return True
```

- Kiểm tra trên đường chéo phải:

```
202         # check cheo phai
203         count = 0
204         i, j = x, y
205         while(i >= 0 and j < Ox and self.Buts[i, j]["text"] == XO):
206             count += 1
207             i -= 1
208             j += 1
209         i, j = x, y
210         while(i <= Oy and j >= 0 and self.Buts[i, j]["text"] == XO):
211             count += 1
212             i += 1
213             j -= 1
214         if count >= 6:
215             return True
```

- Kiểm tra trên đường chéo trái:

```
216     # check cheo trai
217     count = 0
218     i, j = x, y
219     while(i < Oy and j < Ox and self.Buts[i, j]["text"] == XO):
220         count += 1
221         i += 1
222         j += 1
223     i, j = x, y
224     while(i >= 0 and j >= 0 and self.Buts[i, j]["text"] == XO):
225         count += 1
226         i -= 1
227         j -= 1
228     if count >= 6:
229         return True
230     return False
```

2.3.3.12 ***Phương thức notification(self, title, message)*** Hiển thị thông báo người chơi giành được chiến thắng.

```
175 ∨ def notification(self, title, msg):
176         messagebox.showinfo(str(title), str(msg))
```

2.3.3.13 ***Phương thức newGame(self)*** Phương thức này khởi tạo lại trò chơi và xóa hết các ô đã đi.

```
245     def newGame(self):
246         for x in range(Ox):
247             for y in range(Oy):
248                 self.Buts[x, y]["text"] = ""
```

2.3.4 Các phương thức trong class Threading_socket()

```
250 class Threading_socket():
```

Lớp Threading_socket được thiết kế để quản lý kết nối mạng trong trò chơi cờ caro. Nó cho phép tạo kết nối giữa máy chủ và máy khách, gửi và nhận dữ liệu qua mạng để đồng bộ hóa trạng thái trò chơi giữa hai bên.

2.3.4.1 ***Phương thức __init()__*** Khởi tạo các thuộc tính của đối tượng Threading_socket().

```
251     def __init__(self, gui):
252         super().__init__()
253         self.dataReceive = ""
254         self.conn = None
255         self.gui = gui
256         self.name = ""
```

- 'self.dataReceive': Biến này lưu trữ dữ liệu nhận được từ kết nối mạng.
- 'self.conn': Biến này đại diện cho kết nối mạng.
- 'self.gui': Biến này lưu trữ tham chiếu đến giao diện đồ họa người dùng (GUI), để có thể tương tác với các thành phần GUI từ lớp Threading_socket.
- 'self.name': Biến này lưu trữ tên của kết nối (client hoặc server).

2.3.4.2 ***Phương thức serverAction(self)*** Thiết lập máy chủ, chấp nhận kết nối từ máy khách và khởi động luồng để nhận dữ liệu.

```
289     def serverAction(self):
290         self.name = "server"
291         HOST = socket.gethostbyname(socket.gethostname())  # Lấy địa chỉ
292         print("Make host.........." + HOST)
293         self.gui.notification("Gui IP chp ban", str(HOST))
294         PORT = 8000  # Thiết lập port lắng nghe
295         # cấu hình kết nối
296         s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
297         s.bind((HOST, PORT))  # lắng nghe
298         s.listen(1)  # thiết lập tối ta 1 kết nối đồng thời
299         self.conn, addr = s.accept()  # chấp nhận kết nối và trả về thông số
300         t2 = threading.Thread(target=self.server, args=(addr, s))
301         t2.start()
```

- 'self.name': Gán giá trị "server" cho biến này để xác định loại kết nối.
- 'HOST': Địa chỉ IP của máy chủ (được lấy tự động).
- 'PORT': Cổng lắng nghe của máy chủ.
- 's': Tạo và cấu hình socket.
- 'self.conn' và 'addr': Chấp nhận kết nối từ client.
- 't2': Tạo một luồng để chạy phương thức server.

2.3.4.3 ***Phương thức server(self, addr, s)*** Phương thức này chịu trách nhiệm nhận dữ liệu từ server khi đối tượng đang hoạt động dưới dạng client.

```
303     def server(self, addr, s):
304         try:
305             # in ra thông địa chỉ của client
306             print('Connected by', addr)
307             while True:
308                 # Đọc nội dung client gửi đến
309                 self.dataReceive = self.conn.recv(1024).decode()
310                 if(self.dataReceive != ""):
311                     friend = self.dataReceive.split("|")[0]
312                     action = self.dataReceive.split("|")[1]
313                     print(action)
314                     if(action == "hit" and friend == "client"):
315                         x = int(self.dataReceive.split("|")[2])
316                         y = int(self.dataReceive.split("|")[3])
317                         self.gui.handleButton(x, y)
318                     if(action == "Undo" and friend == "client"):
319                         self.gui.Undo(False)
320                     if(action == "message" and friend == "client"):
321                         message = self.dataReceive.split("|")[2]
322                         self.gui.displayMessage("Friend: " + message)
323                 self.dataReceive = ""
324         finally:
325             s.close()  # đóng socket
```

- Vòng lặp while True: Vòng lặp vô hạn để liên tục lắng nghe và nhận dữ liệu từ server.
- ‘self.dataReceive = self.conn.recv(1024).decode()’:
- ‘self.conn.recv(1024)’: Nhận tối đa 1024 byte dữ liệu từ server.
- ‘.decode()’: Giải mã dữ liệu nhận được từ dạng byte sang dạng chuỗi.
- ‘if self.dataReceive != ""’: Kiểm tra xem dữ liệu nhận được có rỗng hay không.
- Phân tích dữ liệu nhận được:
- ‘friend = self.dataReceive.split("|")[0]’: Tách chuỗi dữ liệu nhận được và lấy phần đầu tiên (người gửi).
- ‘action = self.dataReceive.split("|")[1]’: Lấy phần thứ hai (hành động).
- Xử lý các hành động khác nhau:

- 'if action == "hit" and friend == "server"':': Nếu hành động là "hit" từ server, lấy tọa độ x và y và gọi phương thức handleButton của GUI để cập nhật nước đi.
- 'if action == "Undo" and friend == "server":': Nếu hành động là "Undo" từ server, gọi phương thức Undo của GUI để hoàn tác nước đi mà không đồng bộ hóa lại.
- 'if action == "message" and friend == "server":': Nếu hành động là "message" từ server, lấy tin nhắn và hiển thị nó trong giao diện GUI.
- 'self.dataReceive = ""': Đặt lại biến nhận dữ liệu để sẵn sàng nhận dữ liệu tiếp theo.
- 'finally: s.close()': Đảm bảo socket được đóng lại khi kết thúc phương thức, ngay cả khi có lỗi xảy ra.

2.3.4.4 ***Phương thức clientAction(self, inputIP)*** Thiết lập máy khách, kết nối đến máy chủ và khởi động luồng để nhận dữ liệu.

```
258     def clientAction(self, inputIP):
259         self.name = "client"
260         print("client connect ..............")
261         HOST = inputIP  # Cấu hình địa chỉ server
262         PORT = 8000             # Cấu hình Port sử dụng
263         self.conn = socket.socket(
264             socket.AF_INET, socket.SOCK_STREAM)  # Cấu hình socket
265         self.conn.connect((HOST, PORT))  # tiến hành kết nối đến server
266         self.gui.notification("Đã kết nối tới", str(HOST))
267         t1 = threading.Thread(target=self.client)  # tạo luồng chạy client
268         t1.start()
```

- 'inputIP': Địa chỉ IP của server mà client muốn kết nối tới.
- 'self.name': Gán giá trị "client" cho biến này để xác định loại kết nối.
- 'HOST' và 'PORT': Địa chỉ và cổng của server.
- 'self.conn': Tạo một socket kết nối tới server.
- 'self.gui.notification': Hiển thị thông báo về việc kết nối.
- 't1': Tạo một luồng để chạy phương thức client.

2.3.4.5 ***Phương thức client(self)*** Phương thức này chịu trách nhiệm nhận dữ liệu từ server khi đối tượng đang hoạt động dưới dạng client.

```
270     def client(self):
271         while True:
272             self.dataReceive = self.conn.recv(
273                 1024).decode()  # Đọc dữ liệu server trả về
274             if(self.dataReceive != ""):
275                 friend = self.dataReceive.split("|")[0]
276                 action = self.dataReceive.split("|")[1]
277                 if(action == "hit" and friend == "server"):
278                     #     print(self.dataReceive)
279                     x = int(self.dataReceive.split("|")[2])
280                     y = int(self.dataReceive.split("|")[3])
281                     self.gui.handleButton(x, y)
282                 if(action == "Undo" and friend == "server"):
283                     self.gui.Undo(False)
284                 if(action == "message" and friend == "server"):
285                     message = self.dataReceive.split("|")[2]
286                     self.gui.displayMessage("Friend: " + message)
287             self.dataReceive = ""
```

- Vòng lặp while True: Vòng lặp vô hạn để liên tục lắng nghe và nhận dữ liệu từ server.
- ‘self.dataReceive = self.conn.recv(1024).decode()’:
- ‘self.conn.recv(1024)’: Nhận tối đa 1024 byte dữ liệu từ server.
- ‘.decode()’: Giải mã dữ liệu nhận được từ dạng byte sang dạng chuỗi.
- ‘if self.dataReceive != ""’: Kiểm tra xem dữ liệu nhận được có rỗng hay không.
- Phân tích dữ liệu nhận được:
- ‘friend = self.dataReceive.split("|")[0]’: Tách chuỗi dữ liệu nhận được và lấy phần đầu tiên (người gửi).
- ‘action = self.dataReceive.split("|")[1]’: Lấy phần thứ hai (hành động).
- Xử lý các hành động khác nhau:
- ‘if action == "hit" and friend == "server":’: Nếu hành động là "hit" từ server, lấy tọa độ x và y và gọi phương thức handleButton của GUI để cập nhật nước đi.
- ‘if action == "Undo" and friend == "server":’: Nếu hành động là "Undo" từ server, gọi phương thức Undo của GUI để hoàn tác nước đi mà không đồng bộ hóa lại.

- 'if action == "message" and friend == "server":': Nếu hành động là "message" từ server, lấy tin nhắn và hiển thị nó trong giao diện GUI.
- 'self.dataReceive = ""': Đặt lại biến nhận dữ liệu để sẵn sàng nhận dữ liệu tiếp theo.

2.3.4.6 *Phương thức sendData(self, data)*

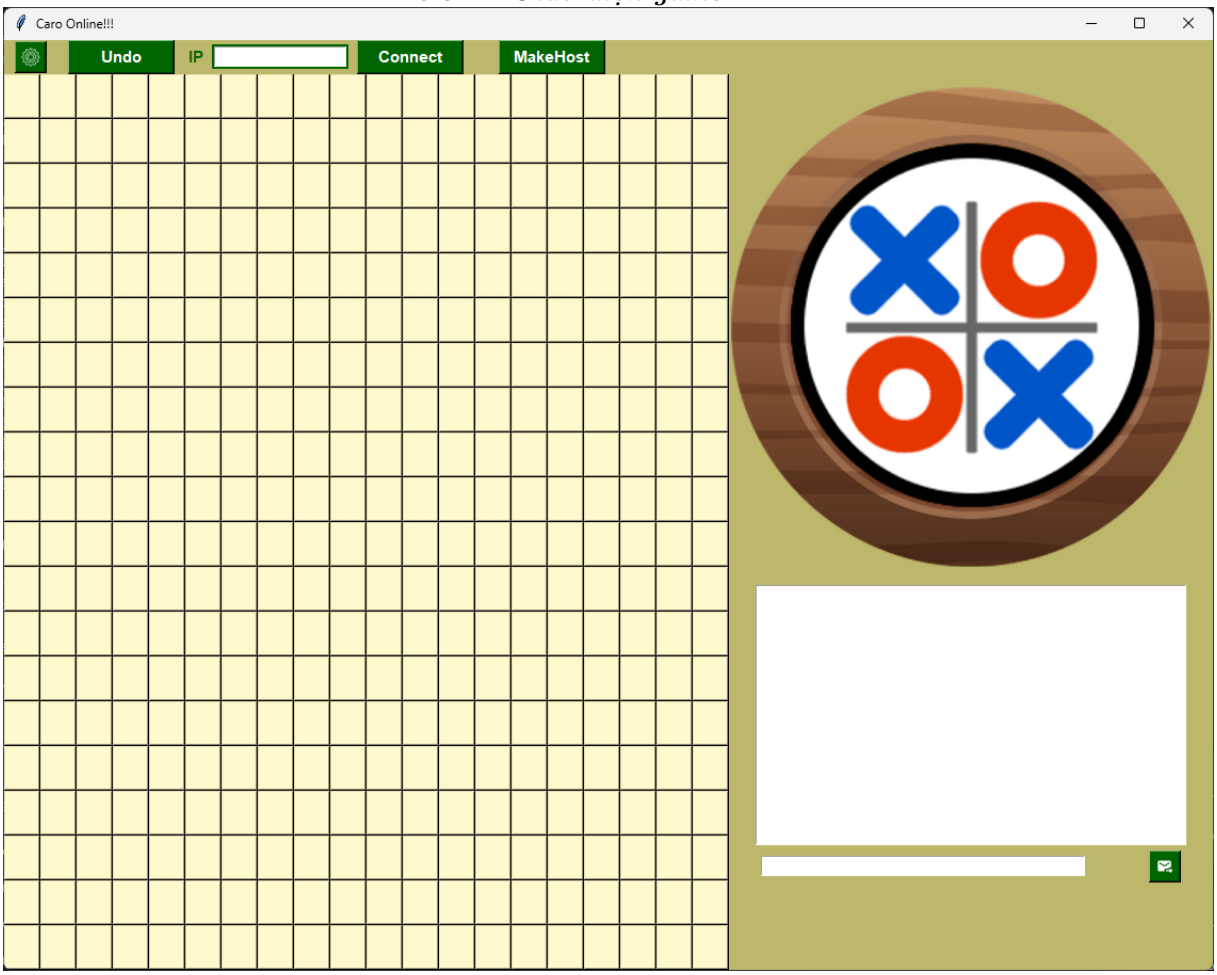
Được dùng để gửi dữ liệu qua mạng.

```
327     def sendData(self, data):
328         # Gửi dữ liệu lên server
329         self.conn.sendall(str("{}|".format(self.name) + data).encode())
```

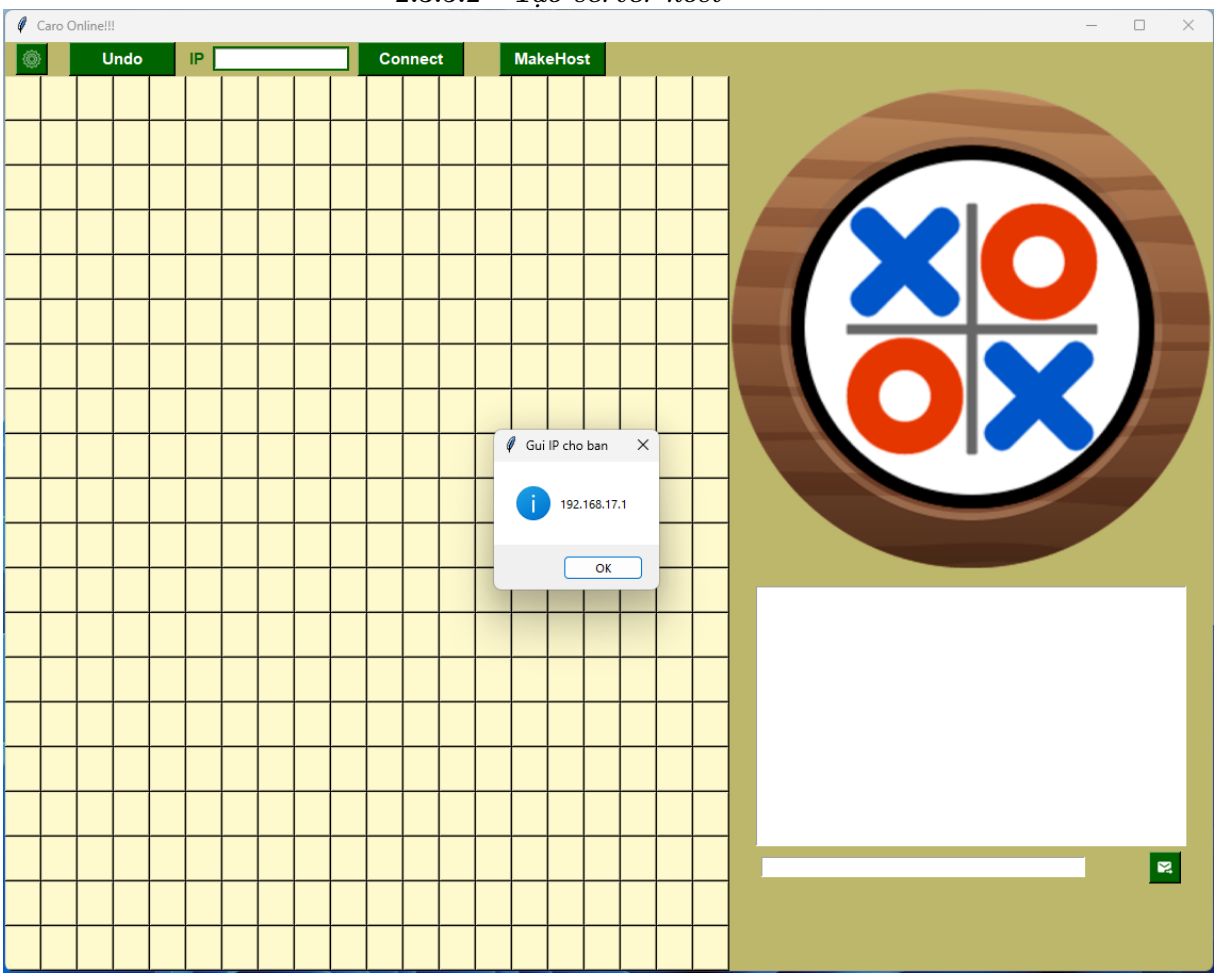
- 'data': Dữ liệu cần gửi.
- 'self.conn.sendall(...)': Gửi toàn bộ dữ liệu qua kết nối mạng.

2.3.5 Demo game

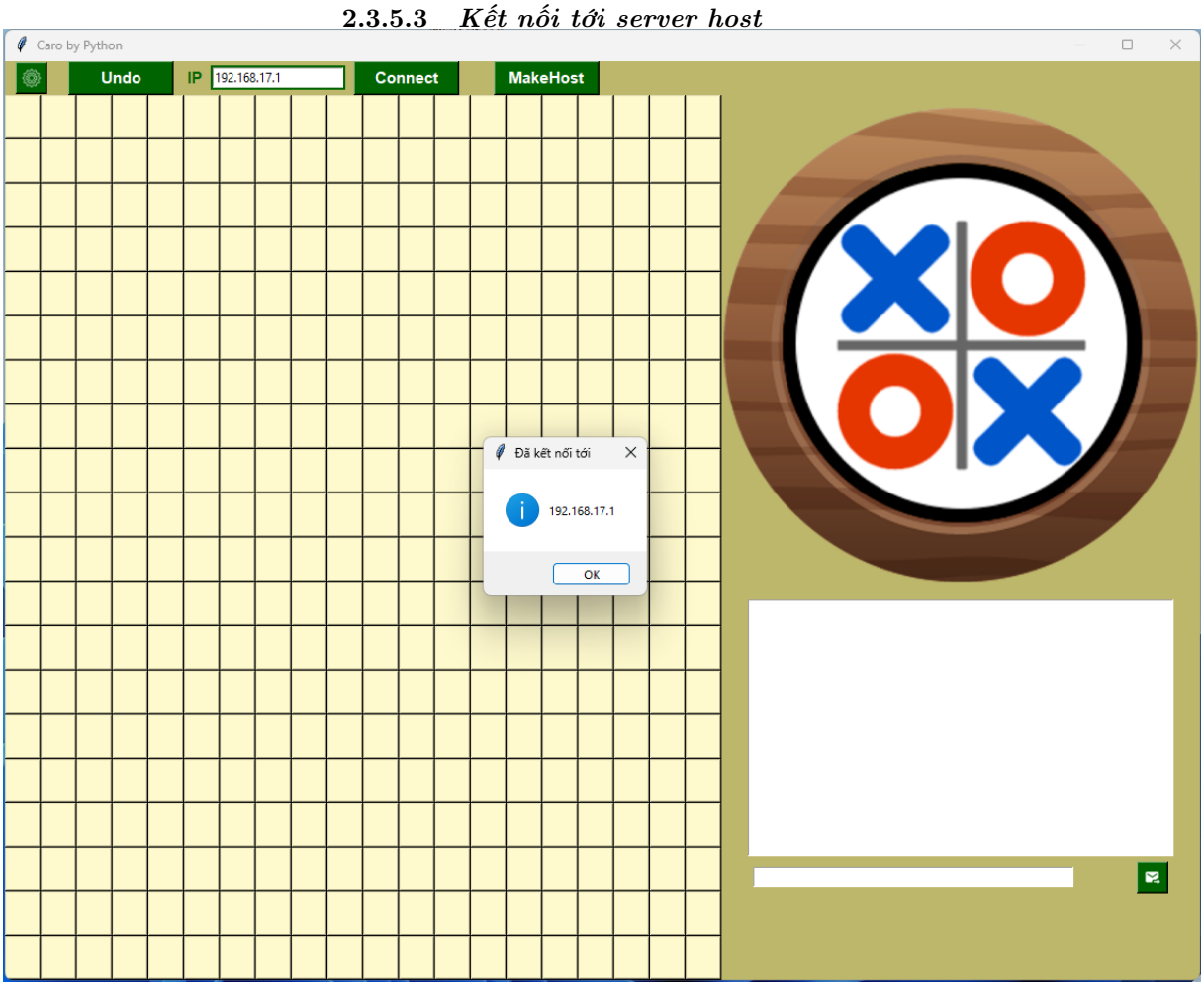
2.3.5.1 *Giao diện game*



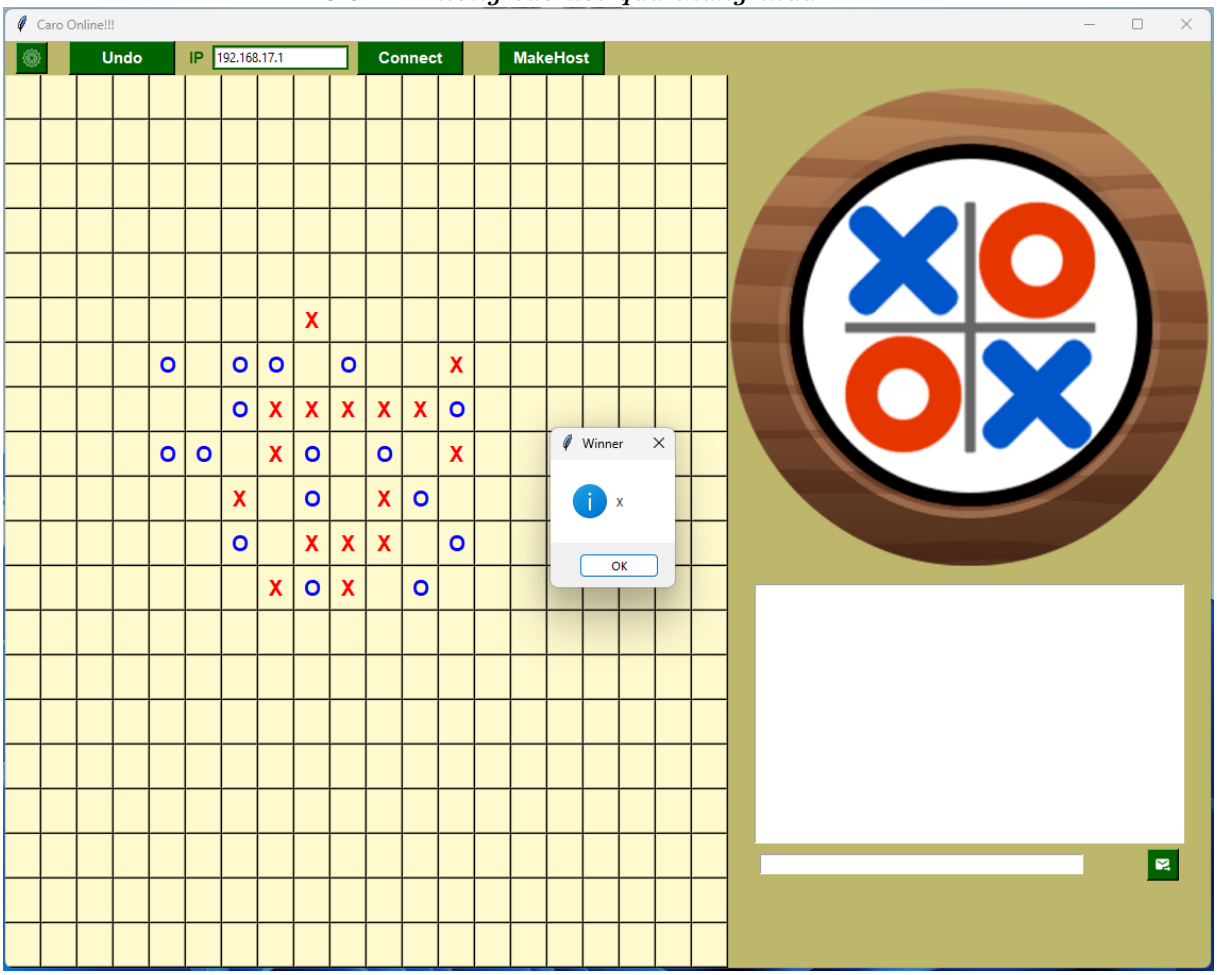
2.3.5.2 *Tạo server host*



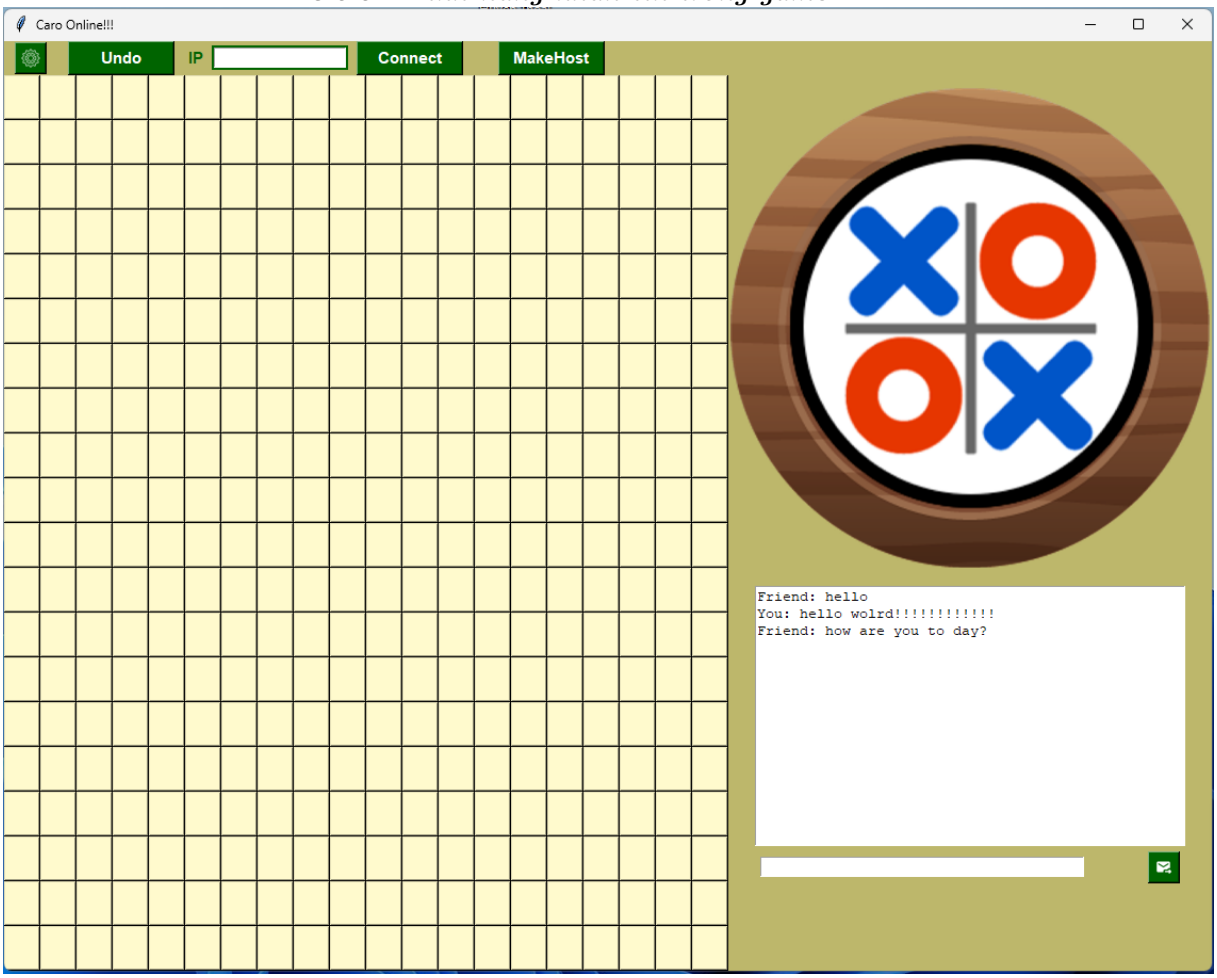
2.3.5.3 *Kết nối tới server host*



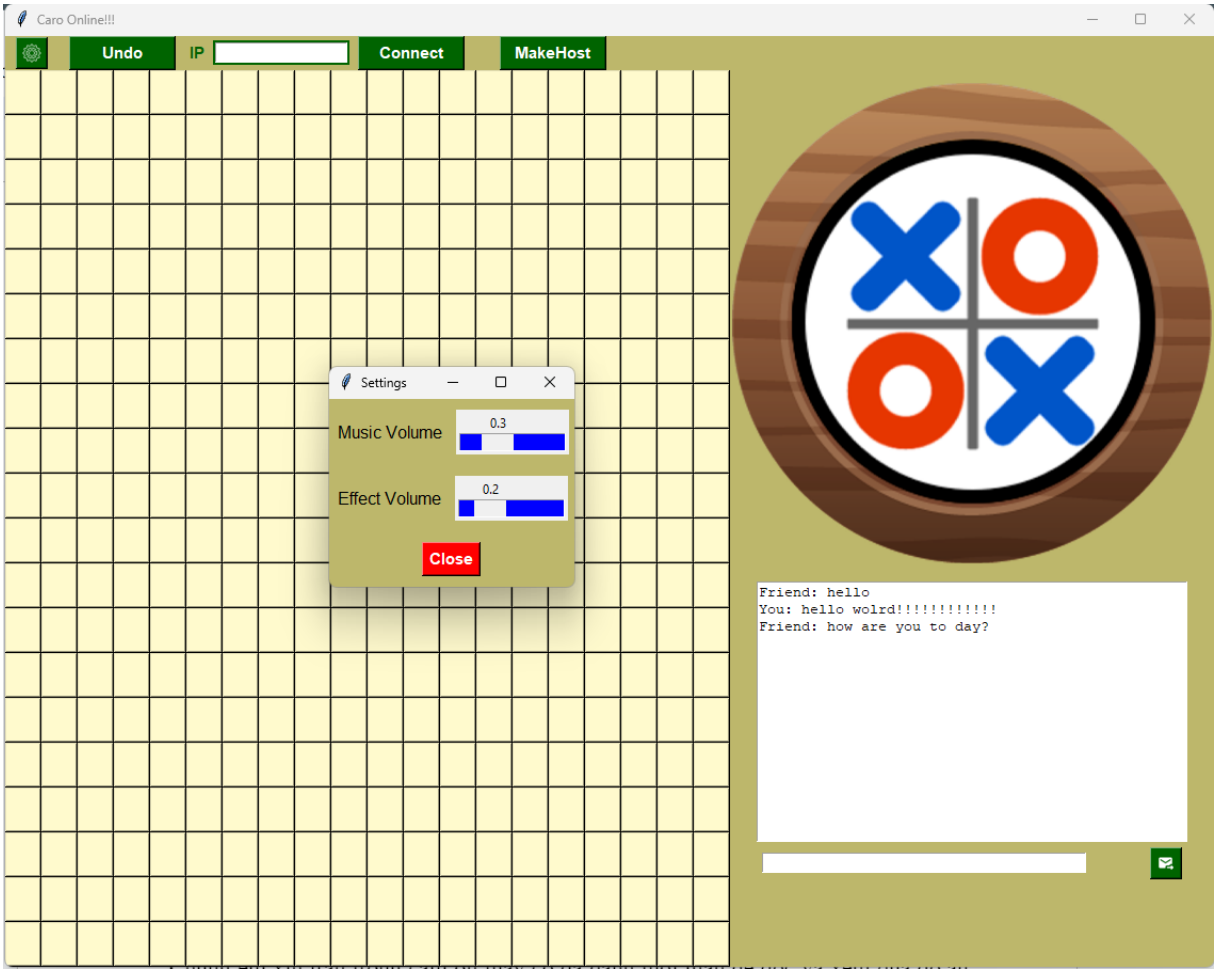
2.3.5.4 *Thông báo kết quả thắng thua*



2.3.5.5 *Tính năng nhắn tin trong game*



2.3.5.6 ***Tính năng điều chỉnh âm thanh nhạc, hiệu ứng trong game*** Tính năng điều chỉnh âm thanh nhạc, hiệu ứng trong game



2.4 Ưu điểm

- Tích hợp nhiều chức năng: Code cung cấp một loạt các chức năng như chơi trò chơi Caro, gửi và nhận tin nhắn giữa client và server, cũng như điều chỉnh âm lượng âm thanh và cài đặt khác.
- Sử dụng giao diện đồ họa (GUI): Sử dụng thư viện tkinter để tạo giao diện người dùng dễ sử dụng và thân thiện.
- Sử dụng luồng (threading): Sử dụng luồng để xử lý các tác vụ mạng, giúp tránh tình trạng chặn luồng chính.
- Tích hợp âm thanh: Code tích hợp âm thanh cho trò chơi, tăng trải nghiệm người dùng.

2.5 Nhược điểm

- Bảo mật: Code không có bất kỳ lớp bảo mật nào. Giao tiếp mạng không được mã hóa hoặc bảo vệ, điều này có thể dẫn đến các lỗ hổng bảo mật.

- Phần mềm thô sơ: Code vẫn còn thiếu một số tính năng như tích hợp cơ chế xác thực, quản lý người dùng và phân quyền.
- Giao diện người dùng đơn giản: Mặc dù có giao diện người dùng, nhưng nó vẫn còn thiếu một số tính năng như thông báo lỗi chi tiết và trực quan hóa tốt hơn.

2.6 Hướng phát triển thêm

- Cải thiện bảo mật: Thêm mã hóa và cơ chế xác thực để bảo vệ dữ liệu mạng.
- Tích hợp đa ngôn ngữ: Cho phép người dùng chọn ngôn ngữ giao diện phù hợp với họ.
- Tối ưu hóa giao diện người dùng: Cải thiện giao diện người dùng với hình ảnh đồ họa hấp dẫn hơn và thông báo lỗi chi tiết hơn.
- Thêm tính năng đa nền tảng: Phát triển ứng dụng để có thể chạy trên nhiều nền tảng như Windows, macOS và Linux.
- Kiểm thử và sửa lỗi: Thực hiện kiểm thử chất lượng phần mềm (QA) để tìm và sửa lỗi, cũng như cải thiện hiệu suất và ổn định của ứng dụng.

2.7 Kết luận

Sau một thời gian tìm hiểu đồ án, thu thập các kiến thức liên quan và tham khảo cách làm một số nơi trên Internet, chúng em đã hoàn thành đồ án game "Caro Online". Mặc dù rất cố gắng, nhưng vẫn không tránh khỏi thiếu sót và hạn chế. Chúng em rất mong có được những ý kiến đánh giá, đóng góp của thầy cô để đồ án thêm hoàn thiện. Chúng em xin trân trọng cảm ơn thầy cô đã dành thời gian để đọc và xem qua đồ án của chúng em.

3 Tài liệu tham khảo:

3.1 Tài liệu trên mạng:

[1] https://codelearn.io/sharing/lap-trinh-socket-voi-tcpip-trong-python
[2] https://viblo.asia/p/lap-trinh-socket-bang-python-jvEla084Zkw
[3] https://www.youtube.com/

3.2 Tài liệu khác:

[1] Slide bài giảng Ngôn ngữ lập trình Python của thầy Trịnh Tấn Đạt.
[2] Slide bài giảng Ngôn ngữ lập trình Python của thầy Nguyễn Trung Tín.
[3] Slide bài giảng Ngôn ngữ lập trình Python của thầy Từ Lãng Phiêu.