TRƯỜNG ĐẠI HỌC SÀI GÒN
KHOA CÔNG NGHỆ THÔNG TIN

PHÁT TRIỂN PHẦN MỀM MÃ NGUỒN MỞ

Xây dựng ứng dụng game

Kéo, búa, bao nhiêu người chơi trên PYTHON

Nhóm 15

GVHD: Từ Lãng Phiêu
Thành viên nhóm: Lê Quỳnh Thiên Hương - 3120560036
Bùi Thị Yến Nhi - 3120560069
Nhâm Gia Phát - 3120560071
Email liên hệ: thienhuong6935@gmail.com

TP. HỒ CHÍ MINH, THÁNG 2/2024

Mục lục

1 PHẦN GIỚI THIỆU

Dự án này hướng đến việc xây dựng một ứng dụng game Kéo, Búa, Bao nhiều người chơi bằng Python. Game sẽ bao gồm hai người chơi trực tuyến đối đầu với nhau theo quy tắc truyền thống của trò chơi Kéo, Búa, Bao. Sử dụng thư viện Pygame để tạo giao diện người dùng và một máy chủ mạng để quản lý kết nối và tương tác giữa các người chơi. Việc hoàn thành dự án này sẽ cung cấp một cái nhìn sâu sắc về cách xây dựng một ứng dụng trò chơi từ đầu đến cuối, bao gồm thiết kế giao diện người dùng, lập trình logic trò chơi, và quản lý kết nối mạng.

2 CƠ SỞ LÝ THUYẾT

2.1 Thư viện PyGame

Pygame là một thư viện phổ biến trong Python được sử dụng để phát triển các trò chơi và ứng dụng đa phương tiện. Nó cung cấp các chức năng cần thiết để xử lý đồ họa, âm thanh, và các sự kiện đầu vào từ bàn phím, chuột, và các thiết bị khác.

2.1.1 Giới thiệu về Pygame

Pygame được xây dựng trên SDL (Simple DirectMedia Layer), một thư viện cấp thấp bằng C cung cấp quyền truy cập vào các thiết bị đa phương tiện như đồ họa, âm thanh, và các thiết bị đầu vào. Pygame cung cấp một giao diện đơn giản để truy cập các tính năng mạnh mẽ này, giúp lập trình viên dễ dàng phát triển các trò chơi và ứng dụng đồ họa.

2.1.2 Cài đặt Pygame

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

```
pip install pygame
```

2.1.3 Các thành phần chính Pygame

- **Surface**: Đây là đối tượng chính dùng để vẽ. Màn hình hiển thị chính là một Surface, và bạn có thể tạo nhiều Surface khác để quản lý các yếu tố đồ họa khác nhau.
- **Rect**: Đối tượng Rect (Rectangle) được sử dụng để quản lý vị trí và kích thước của các yếu tố trong trò chơi, cũng như để xử lý va chạm giữa chúng.
- **Event**: Hệ thống sự kiện của Pygame cho phép bạn xử lý các sự kiện từ bàn phím, chuột, và các thiết bị khác
- **Sprite**: Pygame cung cấp một module sprite để quản lý các đối tượng di chuyển và tương tác trong trò chơi một cách dễ dàng hơn.

2.1.4 Các bước cơ bản để tạo mộ trò chơi với Pygame

2.1.4.a Khởi tạo Pygame:

```
import pygame
  pygame.init()
```

2.1.4.b Tạo cửa sổ hiển thị:

```
screen = pygame.display.set_mode((width, height))
```

2.1.4.c Tạo vòng lặp chính của trò chơi:

```
    running = True
      while running:
          for event in pygame.event.get():
              if event.type == pygame.QUIT:
                  running = False
          # Update game state
          # Draw everything
      pygame.display.flip()
      pygame.quit()
```

2.1.4.d Xử lý sự kiện:

```
    for event in pygame.event.get():
          if event.type == pygame.KEYDOWN:
              if event.key == pygame.K_LEFT:
              # Move to left
          elif event.type == pygame.MOUSEBUTTONDOWN:
              #Move to right
```

2.1.4.e Vẽ các yếu tố lên màn hình:

```
    screen.fill((0, 0, 0)) # remove screen by black
      pygame.draw.rect(screen, (255, 0, 0), (x, y, width, height)) # Draw red
          rectangle
      pygame.display.flip() # screen update
```

2.1.5 Quán lý Sprite

Pygame cung cấp mộ tmodule sprite để quản lý các đối tượng trong trò chơi:

```
    import pygame
      pygame.init()

      class Player(pygame.sprite.Sprite):
          def __init__(self):
              super().__init__()
              self.image = pygame.Surface((50, 50))
              self.image.fill((0, 255, 0))
              self.rect = self.image.get_rect()
              self.rect.center = (width // 2, height // 2)

          def update(self):
              # Update player position
```

```
            pass

    all_sprites = pygame.sprite.Group()
    player = Player()
    all_sprites.add(player)

    running = True
    while running:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                running = False

        all_sprites.update()
        screen.fill((0, 0, 0))
        all_sprites.draw(screen)
        pygame.display.flip()

    pygame.quit()
```

2.1.6 Xử lý âm thanh

Pygame cũng hỗ trợ âm thanh và nhạc nền:

```
pygame.mixer.init()
  pygame.mixer.music.load('background.mp3')
  pygame.mixer.music.play(-1) # play loop background music

  sound = pygame.mixer.Sound('effect.wav')
  sound.play() # play effect music
```

Pygame là một thư viện mạnh mẽ và linh hoạt cho việc phát triển trò chơi và ứng dụng đa phương tiện bằng Python. Nó cung cấp tất cả các công cụ cần thiết để tạo ra các trò chơi từ đơn giản đến phức tạp, từ quản lý đồ họa, âm thanh đến xử lý sự kiện. Bằng cách hiểu và sử dụng các thành phần và phương pháp của Pygame, bạn có thể tạo ra những trò chơi thú vị và ứng dụng hấp dẫn.

2.2 Socket trong Python

Socket là giao diện lập trình ứng dụng mạng được dùng để truyền và nhận dữ liệu trên internet. Giữa hai chương trình chạy trên mạng cần có một liên kết giao tiếp hai chiều (two-way communication) để kết nối 2 process trò chuyện với nhau. Điểm cuối (endpoint) của liên kết này được gọi là socket. Socket cho phép giao tiếp trong 1 tiến trình, giữa những tiến trình trên cùng 1 máy hoặc giữa nhiều máy với nhau. Socket được chia chủ yếu thành 2 loại:

- Socket (dựa trên giao thức TCP): Việc truyền dữ liệu chỉ được thực hiện giữa 2 quá trình đã thiết lập kết nối. Steam socket đảm bảo dữ liệu truyền đi đáng tin cậy nhờ có cơ chế chống tắc nghẽn và cơ chế quản lý luồng lưu thông trên mạng.
- Socket (dựa trên giao thức UDP): Việc truyền dữ liệu không cần có thiết lập kết nối giữa 2 quá trình. Trái ngược với TCP, truyền dữ liệu theo giao thức UDP kém tin cậy, có thể sai trình tự và bị lặp lại. Tuy nhiên cơ chế của Datagram đơn giản hơn nên có tốc độ nhanh, thường được ứng dụng trong các ứng dụng chat hoặc game online.

2.2.1 Mô hình Lập trình Socket Python

2.2.1.a Mô tả mô hình Socket:

- Trước tiên chúng ta sẽ tạo ra một máy chủ bằng cách mở một socket – socket() để tạo ổ cắm socket cho Server. Đây là quá trình Hệ điều hành phân bổ tài nguyên, chuẩn bị kết nối. Bạn cần chỉ định tên hoặc số hiệu port cho socket để Client biết đến ổ cắm của Server.
- Sau đó chúng ta liên kết máy chủ với host hoặc một máy và một port – bind().
- Tiếp theo server sẽ bắt đầu lắng nghe các kết nối từ Client đưa đến trên port đó– listen().
- Một yêu cầu kết nối được gửi từ client tới server – connect().Server chấp nhận yêu cầu của client, kết nối từ đó được thiết lập – accept()
- Bây giờ cả hai đều có thể đã có thể gửi và nhận tin – read() / write() tương tự dùng lệnh read/write để đọc ghi trên tập tin. Socket dựa vào số mô tả (socket descriptor) để xác định cần đọc ghi cho hàm read/write.
- Và cuối cùng khi hoàn thành chúng có thể đóng kết nối – close()

2.2.1.b Cách sử dụng mô hình socket

Module socket trong Python sẽ giúp chúng ta thực hiện các kết nối client server để giao tiếp giữa các máy với nhau và để có thể sử dụng được nó thì ta cần thục hiện khởi tạo socket:

- Đầu tiên chúng ta cần phải import module socket vào chương trình với cú pháp sau:

```
1    import socket
```

- Tiếp theo chúng ta sẽ khởi tao đối tượng socket, cụ thể ở đây chúng ta sẽ tạo một socket stream (TCP) như sau:

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

Trong đó, tham số đầu tiên là Address Family: kiểu thiết lập kết nối và Python hỗ trợ 3 dạng: AF_INET: Ipv4, AF_INET6: Ipv6, AF_UNIX. Tham số thứ hai là Socket Type: cách thiết lập giao thức SOCK_STREAM: TCP và SOCK_DGRAM: UDP

Một vài phương thức hay được sử dụng trong đối tượng socket:

- bind(address, port): Phương thức này được dùng để lắng nghe đến địa chỉ address và port
  Ví dụ: s.bind((HOST, PORT)) Đăng ký tên cho socket, ràng buộc địa chỉ vào socket:
- listen(backlog): Phương thức này thiết lập mở kết nối trên server, với tham số truyền vào là số kết nối được phép (nhỏ nhất là 0 và lớn nhất là do cấu hình của server)
  Ví dụ: s.listen(2) Cho socket đang lắng nghe tới tối đa 2 kết nối
- connect(address): Phương thức này thiết lập một kết nối từ client đến server.
- accept(): Phương thức này thiết lập chấp nhận một kết nối, và nó sẽ trả về một tuple gồm 2 thông số (conn, address) để chúng ta có thể gửi ngược về client.
  Ví dụ: client, addr = s.accept() Khi một client gõ cửa, server chấp nhận kết nối và 1 socket mới được tạo ra. Client và server bây giờ đã có thể truyền và nhận dữ liệu với nhau.
- recv(bufsize, flag): Phương thức này dùng để nhận dữ liệu qua giao thức TCP.
  Ví dụ data = client.recv(1024)
- decode("utf8"): Phân tích gói dữ liệu vừa nhận.
  Ví dụ: strdata = data.decode("utf8")
- send(byte,flag): Phương thức này gửi dữ liệu thông qua giao thức TCP.
- recvfrom(bufsize, flag): Phương thức này dùng để nhận dữ liệu qua giao thức UCP.
- s.close(): Phương thức này dùng để đóng một kết nối

2.2.2 Ví dụ lập trình socket

Xây dựng một demo nhỏ về nhận và gửi dữ liệu giữa client với server

2.2.2.a Tạo server

```
import socket

HOST = 'localhost' #Thiet lap ia chi address
PORT = 8000 # Thiet lap post lang nghe
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM) # cau hinh ket noi
s.bind((HOST, PORT)) # lang nghe
s.listen(1) # thiet lap toi ta 1 ket noi ong thoi
conn, addr = s.accept() # chap nhan ket noi v tra ve thong so
with conn:
    try:
        # in ra thong ia chi cua client
```

```
            print('Connected by', addr)
            while True:
                #  oc  noi dung client gui en
                data = conn.recv(1024)
                # In ra Noi dung
                print(data)
                # Va gui  ni  dung ve may khach
                conn.sendall(b'Hello client')
                if not data: # neu khong con data thi dung oc
                    break
        finally:
            s.close() # ong socket
```

2.2.2.b Tạo client

```
import socket

HOST = 'localhost'  # Cau hinh address server
PORT = 8000            # Cau hinh Port su dung
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM) # Cau hinh socket
s.connect((HOST, PORT)) # tien hanh ket noi en server
s.sendall(b'Hello server!') # Gui du lieu len server
data = s.recv(1024) # oc du lieu server tra ve
print('Server Respone: ', repr(data))
```

3 THIẾT KẾ ỨNG DỤNG

3.1 Hoạt động của ứng dụng

Về cơ bản, đây là một trò chơi Kéo, Búa, Bao rất đơn giản. Hai người chơi sẽ được kết nối với nhau. Sau đó, họ có thể lựa chọn bước đi của mình "kéo", "búa", hoặc "bao". Sau khi cả hai người chơi đều đã đưa ra lựa chọn, trò chơi sẽ thông báo người chiến thắng, và vòng chơi mới sẽ được tiếp tục.
Về phương diện kỹ thuật, trò chơi được xây dựng bằng thư viện PyGame của ngôn ngữ lập trình Python, phương thức kết nối những người chơi với nhau là thông qua Socket. Server (máy chủ) sẽ là nơi liên tục gửi và nhận các thông điệp từ các người dùng. Các thông điệp này có thể là các bước đi, yêu cầu làm mới màn chơi, yêu cầu ngắt kết nối... Phần Client (máy khách) sẽ là các người chơi của chúng ta, là nơi mà ta sử dụng thư viện PyGame để lập trình trò chơi, cũng như là hiển thị giao diện trò chơi cùng với các xử lý logic. Lớp Network sẽ là cầu nối giữa Client và Server. Cuối cùng, lớp Game sẽ là lớp giúp ta xử lý về logic của trò chơi.

3.2 Cấu trúc mã nguồn

3.2.1 server.py

Mã nguồn của file này triển khai một máy chủ (server) cho trò chơi Kéo, Búa, Bao sử dụng Socket để thiết lập kết nối mạng giữa máy chủ và các máy khách. Dưới đây là phân tích cấu trúc của nó:

- Import các thư viện: Mã mở đầu bằng việc import các thư viện cần thiết như socket, _thread, pickle, và Game từ module game.
- Thiết lập thông tin máy chủ: Địa chỉ IP của máy chủ và cổng kết nối được xác định trong phần này.
- Tạo socket và liên kết: Máy chủ tạo một socket và liên kết nó với địa chỉ và cổng đã chỉ định. Nếu có lỗi, nó sẽ in ra thông báo lỗi.
- Lắng nghe kết nối đến máy chủ: Máy chủ bắt đầu lắng nghe các kết nối đến từ các máy khách.
- Biến toàn cục và vòng lặp chính: Có một số biến toàn cục được khởi tạo, bao gồm connected (một set lưu trữ các kết nối đã thiết lập), games (trò chơi đang diễn ra), và idCount (đếm số lượng kết nối đã được thiết lập).
- Hàm threaded_client(): Đây là hàm được chạy trên mỗi luồng mới để xử lý kết nối từ một máy khách cụ thể. Nó nhận các tham số là conn (kết nối socket), p (số nguyên đại diện cho người chơi), và gameId (định danh trò chơi). Hàm này xử lý việc gửi và nhận dữ liệu từ máy khách và cập nhật trạng thái của trò chơi.
- Vòng lặp vô hạn cho việc chấp nhận kết nối mới: Trong vòng lặp vô hạn này, máy chủ chấp nhận kết nối từ các máy khách mới. Mỗi khi có kết nối mới được chấp nhận, một luồng mới được tạo để xử lý kết nối đó.

Thông qua cấu trúc này, máy chủ có khả năng xử lý đồng thời nhiều kết nối từ các máy khách, cho phép chúng tham gia vào trò chơi và tương tác với nhau thông qua giao thức mạng.

3.2.2 network.py

Mã nguồn của file này triển khai một lớp Network để tạo và quản lý kết nối mạng giữa máy khách và máy chủ trong trò chơi của chúng ta. Dưới đây là phân tích cấu trúc của nó:

- Import các thư viện: Mã mở đầu bằng việc import các thư viện cần thiết như socket và pickle.
- Lớp Network: Định nghĩa lớp Network để quản lý kết nối mạng giữa máy khách và máy chủ.
- Hàm khởi tạo (__init__): Tạo một socket và thiết lập địa chỉ và cổng kết nối. Hàm này cũng gọi hàm connect() để thiết lập kết nối và nhận thông tin về vai trò của người chơi (người chơi 0 hoặc 1).
- Hàm getP(): Trả về số thứ tự của người chơi.
- Hàm connect(): Thực hiện kết nối đến máy chủ thông qua socket. Nếu kết nối thành công, nó nhận thông tin về vai trò của người chơi từ máy chủ.
- Hàm send(): Gửi dữ liệu đến máy chủ và nhận phản hồi từ nó. Dữ liệu được mã hóa bằng cách sử dụng pickle.

Thông qua lớp Network, máy khách có thể gửi dữ liệu đến máy chủ và nhận phản hồi từ nó, cho phép trò chơi diễn ra qua mạng.

3.2.3 client.py

Mã nguồn của file này triển khai một ứng dụng client cho trò chơi của chúng ta sử dụng thư viện Pygame để tạo giao diện người dùng đồ họa. Dưới đây là phân tích cấu trúc của nó:

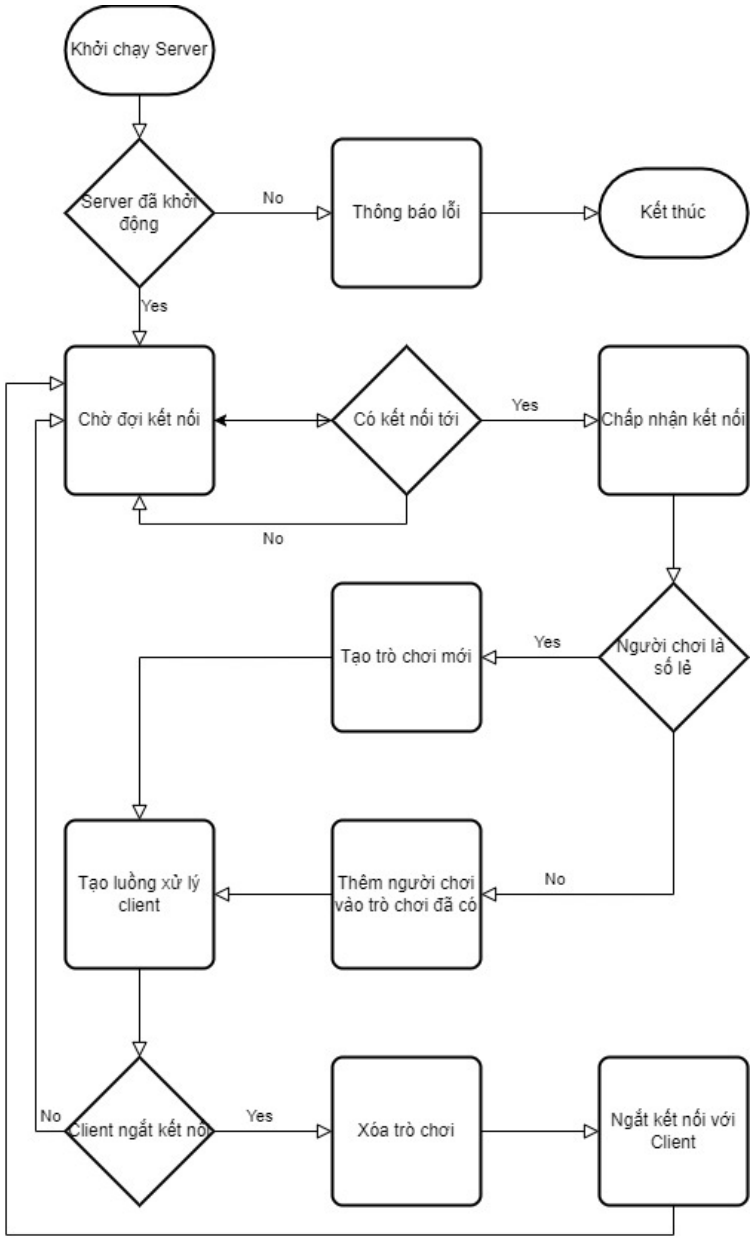
- Import các thư viện: Mã mở đầu bằng việc import các thư viện cần thiết như pygame, Network từ module network, và pickle.
- Khởi tạo cửa sổ và biến: Cài đặt kích thước cửa sổ và tên của cửa sổ Pygame. Khởi tạo một danh sách các nút (Button) để người dùng có thể chọn lựa các nước đi.
- Lớp Button: Định nghĩa một lớp để tạo nút với văn bản, vị trí và màu sắc được chỉ định.
- Hàm redrawWindow(): Vẽ lại cửa sổ với trạng thái hiện tại của trò chơi và các nút. Hiển thị thông tin như trạng thái của người chơi, nước đi của họ và của đối thủ.
- Hàm main: Hàm chính điều khiển luồng chính của trò chơi. Nó liên tục gửi và nhận dữ liệu từ server qua mạng để cập nhật trạng thái của trò chơi và hiển thị nó lên cửa sổ Pygame.
- Hàm menu_screen(): Hàm này hiển thị màn hình menu để bắt đầu trò chơi. Người chơi chỉ cần nhấp chuột để bắt đầu trò chơi.
- Vòng lặp chính: Trò chơi được bắt đầu bằng cách gọi hàm menu_screen(). Sau khi người chơi kết thúc màn hình menu, hàm main() được gọi để bắt đầu trò chơi. Sau khi trò chơi kết thúc, màn hình menu được hiển thị lại để cho phép người chơi bắt đầu trò chơi mới.

3.2.4 game.py

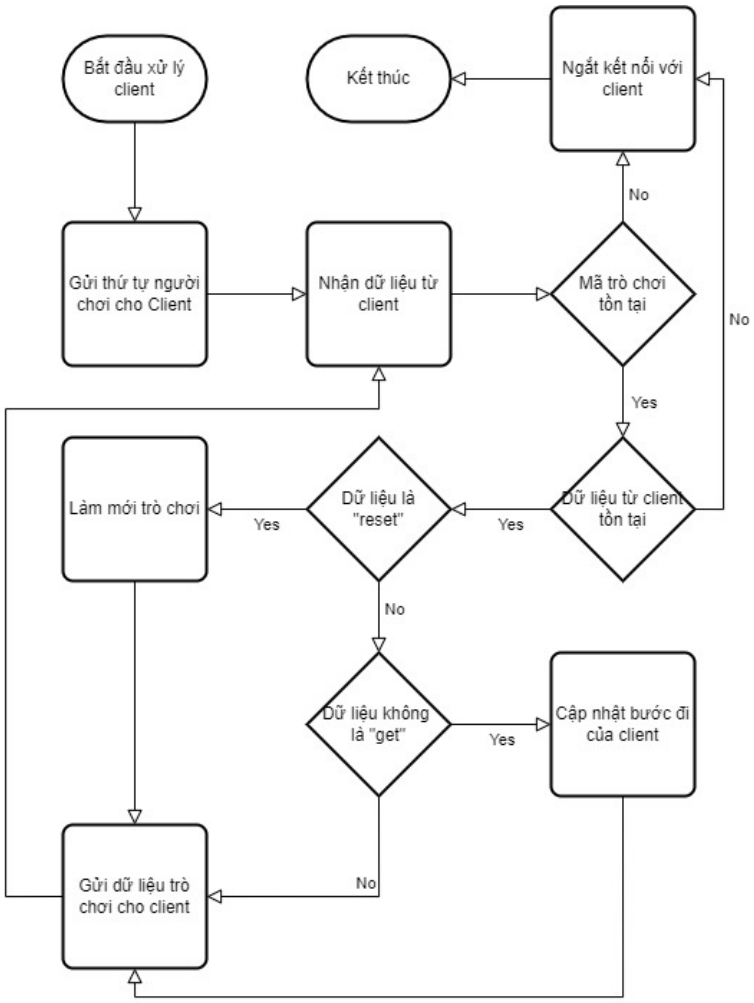
- Hàm khởi tạo (__init__): Hàm này được gọi khi một đối tượng Game mới được tạo. Nó thiết lập các thuộc tính ban đầu của trò chơi như p1Went, p2Went, ready, id, moves, wins, và ties.
- Hàm get_player_move(): Hàm này nhận một tham số p là số nguyên (0 hoặc 1) đại diện cho người chơi và trả về nước đi của người chơi đó.
- Hàm play(): Hàm này ghi lại nước đi của một người chơi vào danh sách nước đi. Nó cũng đánh dấu rằng người chơi đã thực hiện nước đi.
- Hàm connected(): Hàm này trả về trạng thái kết nối của trò chơi.
- Hàm bothWent(): Hàm này kiểm tra xem cả hai người chơi đã thực hiện nước đi chưa.
- Hàm winner(): Hàm này xác định người chiến thắng của ván đấu dựa trên nước đi của cả hai người chơi. Nó trả về 0 nếu người chơi 1 thắng, 1 nếu người chơi 2 thắng và -1 nếu hòa.
- Hàm resetWent(): Hàm này đặt lại trạng thái của p1Went và p2Went về False, để chuẩn bị cho một ván mới.

3.3 Flowchart

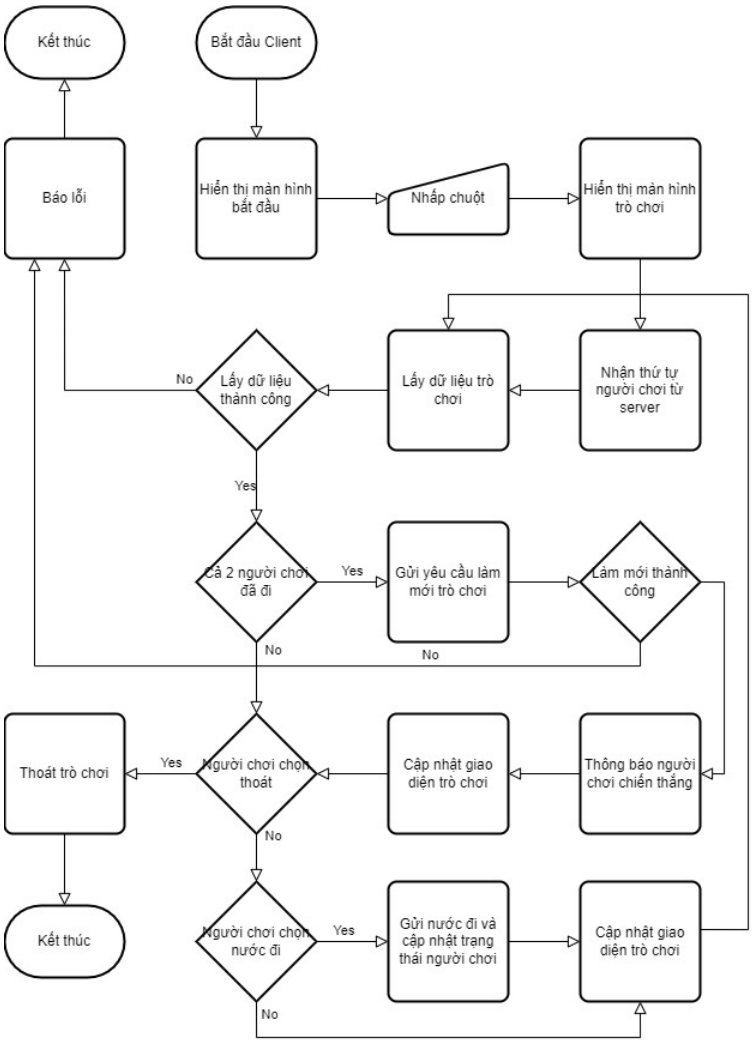
3.3.1 Hoạt động của Server



3.3.2 Hoạt động của phần xử lý client



3.3.3 Hoạt động của Client



4 ỨNG DỤNG ĐÃ XÂY DỰNG

4.1 Mã nguồn

4.1.1 network.py

Tệp network.py chứa mã nguồn tạo một lớp Network để quản lý việc kết nối từ client đến server trong trò chơi mạng "Kéo, Búa, Bao" (Rock, Paper, Scissors). Lớp Network cung cấp các phương thức để kết nối tới server, gửi và nhận dữ liệu một cách dễ dàng. Nó quản lý socket của client và sử dụng pickle để truyền các object phức tạp qua mạng. Các phương thức chính bao gồm:

- connect: Thiết lập kết nối với server và nhận số thứ tự của người chơi.
- send: Gửi dữ liệu tới server và nhận lại trạng thái trò chơi từ server.
- getP: Trả về số thứ tự của người chơi. Lớp này được thiết kế để tích hợp vào ứng dụng trò chơi "Kéo, Búa, Bao" (Rock, Paper, Scissors), nơi mà client cần liên tục gửi và nhận thông tin từ server để đồng bộ hóa trạng thái trò chơi.

```
import socket
import pickle


class Network:
    def __init__(self):
        #Tao mot socket moi cho client TCP/IP.
        self.client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.server = "192.168.1.8" #Dia chi IP cua server
        self.port = 5555 #Cong ma server lang nghe
        self.addr = (self.server, self.port) #chua dia chi IP va cong dung de ket
            noi toi server
        self.p = self.connect() #Bien luu tru so thu tu cua nguoi choi duoc nhan tu
            server sau khi ket noi

    #Phuong Thuc getP tra ve so thu tu cua nguoi choi da duoc nhan tu server
    def getP(self):
        return self.p

    def connect(self):
        try:
            self.client.connect(self.addr) #Ket noi toi server su dung dia chi self
                addr
            #Tra ve so thu tu cua nguoi choi nhan duoc tu server
            return self.client.recv(2048).decode() #Nhan du lieu tu server so thu tu
                cua nguoi choi voi kich thuoc toi da 2048 byte va giai ma decode no
                thanh chuoi
        #Neu xay ra loi trong qua trinh ket noi bo qua va khong lam gi trong except
        except:
            pass

    def send(self, data):
        try:
```

```
29            #Gui du lieu den server sau khi ma hoa encode thanh chuoi
30            self.client.send(str.encode(data))
31            #Nhan lai du lieu tu server voi kich thuoc toi da 4096 byte 2048 2 va
                  giai tuan tu hoa deserialize no bang pickle
32            #Tra ve object da duoc giai tuan tu hoa tu server
33            return pickle.loads(self.client.recv(2048*2))
34        #Neu xay ra loi trong qua trinh gui hoac nhan du lieu in ra loi
35        except socket.error as e:
36            print(e)
```

4.1.2 server.py

Tệp server.py chứa mã nguồn tạo ra một máy chủ (server) để quản lý trò chơi "Kéo, Búa, Bao" (Rock, Paper, Scissors) cho hai người chơi, xử lý các kết nối client và duy trì trạng thái trò chơi giữa các client. Mỗi khi một client kết nối, server sẽ xác định xem có cần tạo một trò chơi mới hay không và bắt đầu một thread mới để xử lý client đó. Các trạng thái của trò chơi được truyền qua mạng bằng cách sử dụng 'pickle'.
Mã nguồn này thiết lập một server TCP cho trò chơi hai người chơi, sử dụng các thư viện _thread, pickle, socket, và một lớp Game từ module game.

```
1 import socket
2 from _thread import *
3 import pickle #Thu vien de tuan tu hoa serialize va giai tuan tu hoa deserialize
      cac object Python giup chung co the truyen qua mang.
4 from game import Game
5
6 server = "192.168.1.8"
7 port = 5555
8
9 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
10
11 try:
12     s.bind((server, port)) #Gan dia chi IP va cong vao socket Neu co loi bat loi va
           luu tru thong bao loi
13 except socket.error as e:
14     str(e)
15
16 s.listen() #Dat socket vao che do lang nghe cac ket noi den
17 print("Waiting for a connection, Server Started")
18
19 connected = set()
20 games = {}
21 idCount = 0 #Bien dem so luong ket noi dung de xac dinh ID tro choi
22
23 def threaded_client(conn, p, gameId):
24     global idCount
25     #Gui so thu tu cua nguoi choi p den client sau khi ket noi
26     conn.send(str.encode(str(p)))
27
28     reply = ""
```

```
    #Vong lap de lien tuc nhan du lieu tu client
    while True:
        try:
            #Nhan du lieu tu client va giai ma no thanh chuoi
            data = conn.recv(4096).decode()
            #Kiem tra xem tro choi co ton tai khong
            if gameId in games:
                #Lay doi tuong tro choi tuong ung
                game = games[gameId]

                if not data:
                    break
                else:
                    #Neu nhan duoc lenh reset thi goi ham resetWent cua tro choi
                    if data == "reset":
                        game.resetWent()
                    #Neu du lieu khong phai lenh get thi choi tro choi voi du lieu
                        nhan duoc
                    elif data != "get":
                        game.play(p, data)
                    #Gui trang thai tro choi da duoc tuan tu hoa den client
                    conn.sendall(pickle.dumps(game))
            else:
                break
        except:
            break

    print("Lost connection")
    #Xoa tro choi khi ket noi bi mat
    try:
        del games[gameId]
        print("Closing Game", gameId)
    except:
        pass
    #Giam so luong ket noi
    idCount -= 1
    #Dong ket noi
    conn.close()

while True:
    conn, addr = s.accept()
    print("Connected to:", addr)
    #Tang so luong ket noi
    idCount += 1
    #Bien de xac dinh so thu tu cua nguoi choi 0 hoac 1
    p = 0
    gameId = (idCount - 1)//2
    #Neu so luong ket noi la le tao mot tro choi moi
    if idCount % 2 == 1:
        #Tao doi tuong tro choi moi voi ID
        games[gameId] = Game(gameId)
        print("Creating a new game...")
```

```
    else:
        #Dat trang thai san sang cho tro choi
        games[gameId].ready = True
        p = 1

    #Tao mot thread moi de xu ly client
    start_new_thread(threaded_client, (conn, p, gameId))
```

4.1.3 client.py

Tệp client.py chứa mã nguồn của một trò chơi "Kéo, Búa, Bao" (Rock, Paper, Scissors) sử dụng thư viện Pygame để tạo giao diện người dùng. Mã nguồn này kết nối với một máy chủ để chơi game trực tuyến.

```
import pygame
from network import Network
import pickle
pygame.font.init()

width = 700
height = 700
#Xac dinh kich thuoc cua so tro choi
win = pygame.display.set_mode((width, height))
pygame.display.set_caption("Client")


class Button:
    def __init__(self, text, x, y, color):
        self.text = text
        self.x = x
        self.y = y
        self.color = color
        self.width = 150
        self.height = 100

    def draw(self, win):
        pygame.draw.rect(win, self.color, (self.x, self.y, self.width, self.height))
        font = pygame.font.SysFont("comicsans", 40)
        text = font.render(self.text, 1, (255,255,255))
        win.blit(text, (self.x + round(self.width/2) - round(text.get_width()/2),
            self.y + round(self.height/2) - round(text.get_height()/2)))

    #Kiem tra xem vi tri chuot (pos) co nam trong khu vuc nut khong.
    def click(self, pos):
        x1 = pos[0]
        y1 = pos[1]
        if self.x <= x1 <= self.x + self.width and self.y <= y1 <= self.y +
            self.height:
            return True
        else:
```

```
35            return False
36
37#Ve cua so tro choi
38def redrawWindow(win, game, p):
39#Set background thanh mau trang
40    win.fill((255, 255, 255))
41
42    if not(game.connected()):
43        #Neu game khong duoc ket noi, hien thi "Waiting for Player..."
44        font = pygame.font.SysFont("comicsans", 60)
45        text = font.render("Waiting for Player...", 1, (140, 120, 204), True)
46        win.blit(text, (width/2 - text.get_width()/2, height/2 -
              text.get_height()/2))
47    else:
48        #Neu game duoc ket noi, hien thi cac thong bao "Your Move" va "Opponents"
49        font = pygame.font.SysFont("comicsans", 60)
50        text = font.render("Your Move", 1, (140, 120, 204))
51        win.blit(text, (80, 200))
52
53        text = font.render("Opponents", 1, (140, 120, 204))
54        win.blit(text, (380, 200))
55
56        #Lay thao tac cua 2 nguoi choi
57        move1 = game.get_player_move(0)
58        move2 = game.get_player_move(1)
59        if game.bothWent():
60            #Neu ca hai deu da di, hien thi nuoc di cua ho
61            text1 = font.render(move1, 1, (0,0,0))
62            text2 = font.render(move2, 1, (0, 0, 0))
63        else:
64        #Nguoc lai, neu ca hai chua move, display trang thai "Waiting..." hoac
              "Locked In" tuy thuoc vao tinh trang cua nguoi choi
65            if game.p1Went and p == 0:
66                text1 = font.render(move1, 1, (0,0,0))
67            elif game.p1Went:
68                text1 = font.render("Locked In", 1, (0, 0, 0))
69            else:
70                text1 = font.render("Waiting...", 1, (0, 0, 0))
71
72            if game.p2Went and p == 1:
73                text2 = font.render(move2, 1, (0,0,0))
74            elif game.p2Went:
75                text2 = font.render("Locked In", 1, (0, 0, 0))
76            else:
77                text2 = font.render("Waiting...", 1, (0, 0, 0))
78        #Hien thi nuoc di o vi tri phu hop dua tren so thu tu (p) cua nguoi choi
79        if p == 1:
80            win.blit(text2, (100, 350))
81            win.blit(text1, (400, 350))
82        else:
83            win.blit(text1, (100, 350))
84            win.blit(text2, (400, 350))
```

```
85          #Ve cac button len cua so
86          for btn in btns:
87              btn.draw(win)
88      #Cap nhat man hinh de hien thi cac thay doi
89      pygame.display.update()
90
91  #Tao 3 button cho cac lua chon "Rock", "Scissors" va "Paper"
92  btns = [Button("Rock", 50, 500, (146, 146, 214)), Button("Scissors", 250, 500,
        (146, 146, 214)), Button("Paper", 450, 500, (146, 146, 214))]
93
94  def main():
95      run = True #Bien dieu khien vong lap chinh cua tro choi
96      clock = pygame.time.Clock() #Tao doi tuong clock kiem soat toc do khung hinh
            cua Pygame
97      n = Network()
98      player = int(n.getP()) #Lay so thu tu cua nguoi choi tu server
99      print("You are player", player)
100
101     while run:
102         clock.tick(60)
103         try:
104             game = n.send("get") #Gui yeu cau 'get' toi server de lay thong tin ve
                    trang thai tro choi
105         except:
106             run = False
107             print("Couldn't get game")
108             break
109
110         if game.bothWent():
111             redrawWindow(win, game, player)
112             pygame.time.delay(500)
113             try:
114                 game = n.send("reset") #Gui yeu cau cau 'reset' toi server de dat
                        lai trang thai tro choi
115             except:
116                 run = False
117                 print("Couldn't get game")
118                 break
119             #Hien thi ket qua tro choi
120             font = pygame.font.SysFont("comicsans", 80)
121             if (game.winner() == 1 and player == 1) or (game.winner() == 0 and
                    player == 0):
122                 text = font.render("You Won!", 1, (0,255,0))
123             elif game.winner() == -1:
124                 text = font.render("Tie Game!", 1, (255,255,0))
125             else:
126                 text = font.render("You Lost...", 1, (255, 0, 0))
127
128             win.blit(text, (width/2 - text.get_width()/2, height/2 -
                    text.get_height()/2))
129             pygame.display.update()
130             pygame.time.delay(2000)
```

```
        #Lap qua tat ca su kien da xay ra
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                run = False
                pygame.quit()

            if event.type == pygame.MOUSEBUTTONDOWN:
                pos = pygame.mouse.get_pos() #Lay vi tri con tro chuot
                for btn in btns:
                    #Kiem tra vi tri nhap chuot co nam trong pham vi cua nut hay
                        khong v tro choi da san sang chua
                    if btn.click(pos) and game.connected():
                        if player == 0:
                            if not game.p1Went:
                                n.send(btn.text)
                        else:
                            if not game.p2Went:
                                n.send(btn.text)

        redrawWindow(win, game, player)
#Hien thi man hinh menu cho nguoi choi nhap chuot de bat dau tro choi. Khi nguoi
     choi nhap chuot, ham 'main' duoc goi de bat dat tro choi
def menu_screen():
    run = True
    clock = pygame.time.Clock()

    while run:
        clock.tick(60)
        win.fill((255, 255, 255))
        font = pygame.font.SysFont("comicsans", 60)
        text = font.render("Click to Play!", 1, (140, 120, 204))
        win.blit(text, (200,250))
        pygame.display.update()

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                run = False
            if event.type == pygame.MOUSEBUTTONDOWN:
                run = False

    main()
#Vong lap de hien thi man hinh menu lien tuc
while True:
    menu_screen()
```

4.1.4 game.py

Tệp 'game.py' chứa định nghĩa của lớp 'Game'. Lớp Game này quản lý trạng thái và logic của trò chơi "Kéo, Búa, Bao" giữa hai người chơi. Nó theo dõi các nước đi của người chơi, xác định người thắng cuộc, và quản lý các trạng thái khác của trò chơi như kết nối và sẵn sàng.

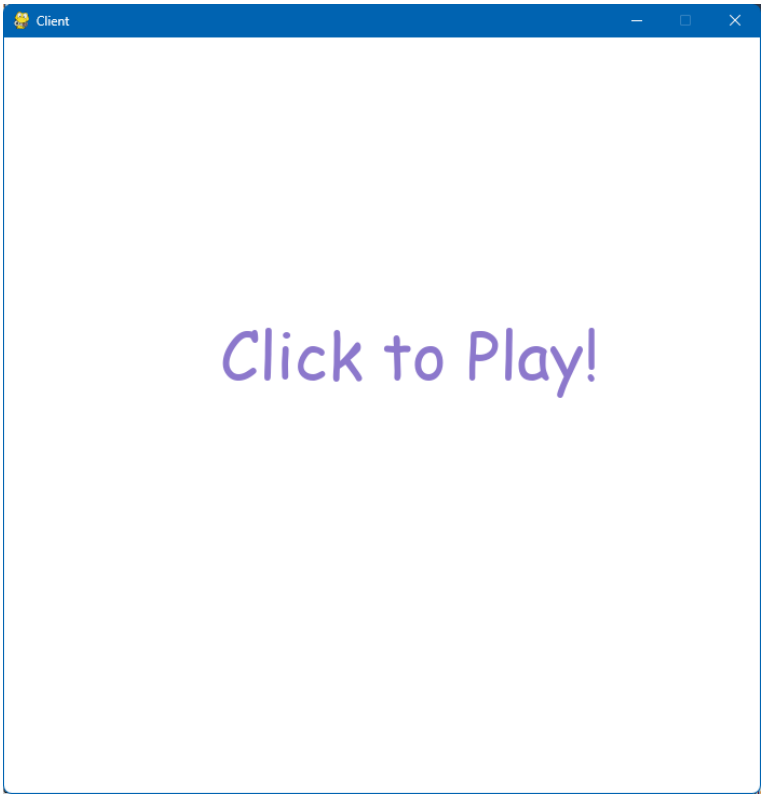
```
class Game:
    def __init__(self, id):
        self.p1Went = False
        self.p2Went = False
        self.ready = False
        self.id = id
        self.moves = [None, None]
        self.wins = [0,0]
        self.ties = 0
    #return nuoc di cua nguoi choi duoc chi dinh
    def get_player_move(self, p):
        """
        :param p: [0,1]
        :return: Move
        """
        return self.moves[p]
    #Cap nhat nuoc di cua nguoi choi (0 hoac 1) voi gia tri 'move'
    def play(self, player, move):
        self.moves[player] = move
        if player == 0:
            self.p1Went = True
        else:
            self.p2Went = True
    #Tra ve gia tri cua 'self.ready' de kiem tra xem tro choi da san sang chua
    def connected(self):
        return self.ready
    #Kiem tra xem ca hai nguoi choi da thuc hien nuoc di cua ho chua
    def bothWent(self):
        return self.p1Went and self.p2Went
    #Ham xac dinh nguoi thang
    def winner(self):
        #Nuoc di duoc chuyen thanh chu hoa va lay ky tu dau tien, de dam bao rang no
            la 'R;, 'P' hoac 'S'
        p1 = self.moves[0].upper()[0]
        p2 = self.moves[1].upper()[0]

        winner = -1
        if p1 == "R" and p2 == "S":
            winner = 0
        elif p1 == "S" and p2 == "R":
            winner = 1
        elif p1 == "P" and p2 == "R":
            winner = 0
        elif p1 == "R" and p2 == "P":
            winner = 1
        elif p1 == "S" and p2 == "P":
            winner = 0
        elif p1 == "P" and p2 == "S":
            winner = 1
        # Tra ve '0' neu nguoi choi 1 thang, '1' neu nguoi choi 2 thang va '-1' neu
            hoa
```

```
50          return winner
51      #Dat lai 2 bien co thanh 'False' de chuan bi cho luot choi tiep theo
52      def resetWent(self):
53          self.p1Went = False
54          self.p2Went = False
```

4.2 Hình ảnh giao diện

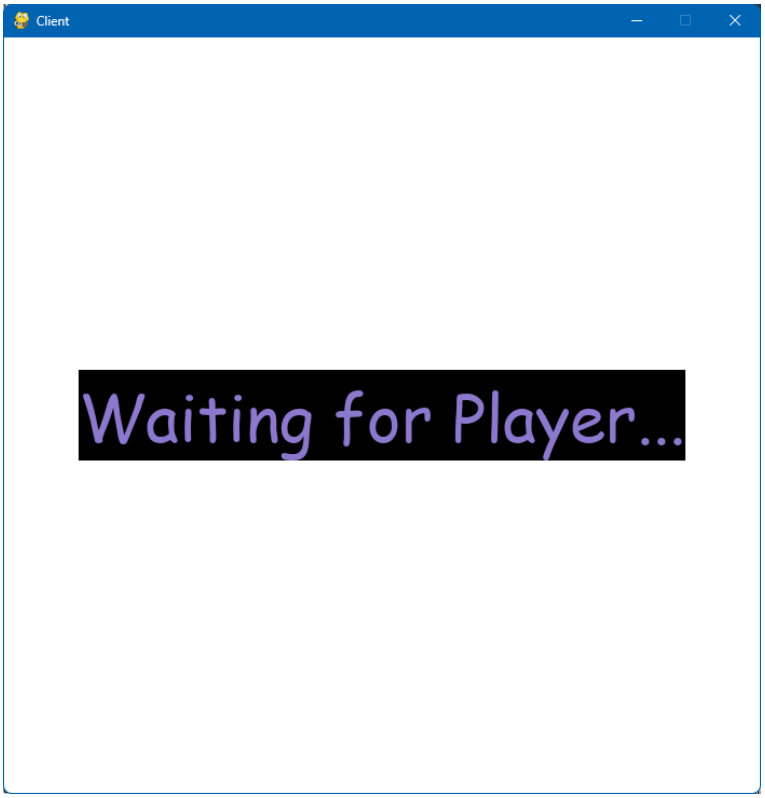
4.2.1 Giao diện chờ kết nối vào game

Sau khi chạy lệnh 'python client.py' trên command line, giao diện chờ kết nối vào game với nội dung "Click to Play!" sẽ được hiển thị để chờ thao tác click từ người dùng/



4.2.2 Giao diện chờ thao tác kết nối vào game từ người chơi còn lại

Sau khi người chơi click vào màn hình giao diện chờ kết nối, màn hình sẽ thay đổi thành "Waiting for Player" để chờ kết nối từ người chơi còn lại.



4.2.3 Giao diện game chính

Sau khi cả hai người chơi đều click vào giao diện chờ kết nối, giao diện game chính được hiển thì, giao diện chia màn hình ra thành 2 phần, phần bên trái thể hiện nước đi của người chơi hiện tại với nội dung "Your Move", phần bên phải sẽ hiển thị nước đi của đối phương với nội dung "Opponents". Bên dưới hiển thị 3 button tương ứng với lựa chọn nước đi của người chơi.

4.2.4 Giao diện chờ lượt chơi từ người chơi còn lại

Người chơi sau khi chọn 1 trong 3 nước đi, màn hình sẽ cập nhật nước đi cho người chơi hiện tại và chờ nước đi từ đối thủ của họ.

Khi đối thủ đã chọn nước đi, bên Opponents sẽ cập nhật thành "Locked In" và chờ nước đi từ người chơi hiện tại.

4.2.5 Giao diện thắng

Giao diện hiển thị kết quả người chơi đã thắng với nội dung "You Won!"

4.2.6 Giao diện hòa

Giao diện hiển thị kết quả hòa với nội dung "Tie Game!"

4.2.7 Giao diện thua

Giao diện hiển thị kết quả người chơi đã thua với nội dung "You Lost..."


Client
Your Move
Opponents
You Lost...
Scissors
Rock
Rock
Scissors
Paper

5 CÁCH THỨC CÀI ĐẶT ỨNG DỤNG

5.1 Cài đặt

Đầu tiền, ta cần phải cài đặt thư viện PyGame bằng lệnh:

```
pip install pygame
```

Sau đó, ta sẽ clone dự án từ link github: https://github.com/giaphat-nham/Rock-Paper-and-Scissors-Multiplayer-Game, bằng lệnh sau:

```
git clone <Đường dẫn đến repository>
```

Vậy là ta đã xong phần cài đặt.

5.2 Cấu hình

Tiếp theo, ta cần phải cấu hình mã nguồn cho phù hợp với thiết bị của chúng ta. Cụ thể hơn là cấu hình địa chỉ IP của server. Ta mở file `server.py` sau đó ghi địa chỉ IP của máy chủ mà ta chọn làm host tại dòng `server = ...` Ta cũng sẽ ghi địa chỉ IP của máy chủ vào file `network.py` tại dòng `self.server = ...`
Vậy là bây giờ chúng ta có thể tiến hành chạy trò chơi.

5.3 Chạy trò chơi

Để chạy trò chơi, đầu tiên tại máy làm máy chủ, ta sẽ sử dụng lệnh:
`python server.py`
Lúc này máy chủ đã được khởi động thành công. Với mỗi người chơi muốn tham gia vào trò chơi, họ sẽ sử dụng lệnh:
`python client.py`
Sau khi gõ lệnh, cửa sổ trò chơi sẽ hiển thị trên màn hình của người chơi, và hai người chơi có thể chơi với nhau.

6 PHẦN CÔNG CÔNG VIỆC GIỮA CÁC THÀNH VIÊN

Tên thành viên	**Công việc**
Lê Quỳnh Thiên Hương	Viết báo cáo phần Cơ sở lý thuyết Pygame; Mã nguồn client.py, game.py; Mô tả hình ảnh giao diện
Bùi Thị Yến Nhi	Viết báo cáo phần Cơ sở lý thuyết Socket trong Python; Mã nguồn server.py và network.py
Nhâm Gia Phát	Xây dựng code ứng dụng, Viết báo cáo phần Thiết kế ứng dụng

Bảng 1: Bảng phân công công việc

Tài liệu

[1] freeCodeCamp.org(27/03/2019). "**link: https://t.ly/A4go-**", *Python Online Multiplayer Game Development Tutorial*, lần truy cập cuối: 05/05/2024.

[2] Geeksforgeeks(12/03/2024). "**link: https://www.geeksforgeeks.org/pygame-tutorial/**", *PyGame Tutorial*, lần truy cập cuối: 05/05/2024.

[3] W3Schools. "**link: https://www.w3schools.com/python**", *Python Tutorial*, lần truy cập cuối: 05/05/2024.