TRƯỜNG ĐẠI HỌC SÀI GÒN
KHOA CÔNG NGHỆ THÔNG TIN

PHÁT TRIỂN PHẦN MỀM MÃ NGUỒN MỞ

Ứng dụng game

Rock Paper Scissors trên PYTHON

GVHD: Từ Lãng Phiêu
Nhóm 29
SV: Huỳnh Minh Quân - 312410433
Thiều Anh Sang - 3120410449
Huỳnh Đức Tâm - 3120410459
Nguyễn Bảo Tân - 3120410464

TP. HỒ CHÍ MINH, THÁNG 5/2024

Mục lục

1 Phần giới thiệu **2**
1.1 Giới thiệu game Rock Paper Scissors 2
1.2 Lợi ích của lập trình game với Python 2

2 Cơ sở lý thuyết **4**
2.1 Phần mềm mã nguồn mở 4
2.2 Python 4
2.3 IDE Pycharm 5
2.4 Thư viện hỗ trợ 5

3 Xây dựng game Rock Paper Scissor **7**
3.1 Mô tả giao diện ứng dụng game 7
3.2 Code chức năng cho game 8
3.3 Cài đặt và thiết lập môi trường 16

4 Xây dựng game đối kháng (Street Fighter) **17**
4.1 Ý tưởng thực hiện 17
4.2 Giao diện của trò chơi 17
4.3 Code chức năng cho game 19

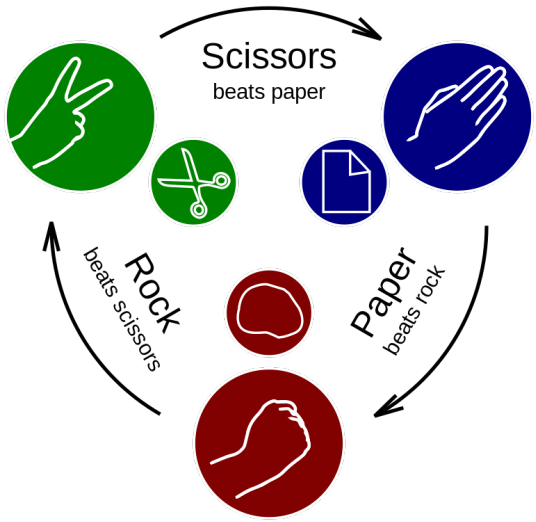
1 Phần giới thiệu

1.1 Giới thiệu game Rock Paper Scissors

"Rock Paper Scissors" là một ứng dụng game giải trí đơn giản nhưng vô cùng thú vị, dựa trên trò chơi truyền thống cùng tên hay được gọi là Oẳn Tù Tì, hoàn toàn chiến thắng dựa trên sự may mắn. Trong trò chơi này, người chơi chọn một trong ba lựa chọn: "Rock" (đá), "Paper" (giấy), hoặc "Scissors" (kéo), và cố gắng chiến thắng bằng cách chọn lựa tốt hơn so với lựa chọn của đối thủ.

Luật chơi đá giấy kéo (Rock Paper Scissors) rất đơn giản:

- Cần có 2 người tham gia trò chơi
- Mỗi người chơi bí mật chọn một trong ba cử chỉ: Đá, Giấy hoặc Kéo.
- Đá thắng Kéo, Kéo thắng Giấy, Giấy thắng Đá.
- Nếu hai người chơi chọn cùng một cử chỉ, sẽ là hòa và chơi lại.



Hình 1: Hình minh họa rock paper scissors

Mục tiêu xây dựng game

Ứng dụng game được phát triển nhằm mục đích nghiên cứu về hệ điều hành mã nguồn mở và cách sử dụng socket trong ngôn ngữ python. Bằng việc xây dựng game Rock Paper Scissors - một trò chơi quen thuộc và đơn giản nhằm minh họa để hiểu rõ hơn môn học, mặc khác giúp người dùng dễ làm quen, tạo trải nghiệm thú vị khi tương tác với những người chơi khác.

1.2 Lợi ích của lập trình game với Python

Game được phát triển bằng Python mang lại nhiều lợi ích khác nhau, không chỉ cho những người tham gia phát triển mà còn cho người chơi. Dưới đây là một số lợi ích chính của việc phát triển game bằng Python:

- Dễ học và sử dụng: Python là một ngôn ngữ lập trình dễ học và sử dụng, đặc biệt là cho người mới bắt đầu. Cú pháp đơn giản và ngắn gọn của Python làm cho việc phát triển game trở nên dễ dàng hơn.

- Phong phú về thư viện và framework: Python có nhiều thư viện và framework mạnh mẽ như Pygame, Panda3D, Pyglet, và Kivy, giúp nhà phát triển tạo ra các trò chơi đa dạng từ game 2D đơn giản đến game 3D phức tạp.
- Đa nền tảng: Game được phát triển bằng Python có thể chạy trên nhiều nền tảng khác nhau bao gồm Windows, macOS và Linux, giúp mở rộng đối tượng người chơi tiềm năng.
- Cộng đồng lớn và hỗ trợ mạnh mẽ: Python có một cộng đồng phát triển lớn và mạnh mẽ, nơi bạn có thể tìm kiếm thông tin, hướng dẫn và hỗ trợ từ các nhà phát triển khác trên toàn thế giới.
- Tích hợp dễ dàng với công nghệ khác: Python có khả năng tích hợp tốt với các công nghệ khác như AI (Trí tuệ nhân tạo), Machine Learning (Học máy), và các API (Giao diện lập trình ứng dụng), mở ra nhiều cơ hội mở rộng cho trò chơi.
- Tăng cường kỹ năng lập trình: Phát triển game bằng Python không chỉ giúp bạn tạo ra sản phẩm giải trí mà còn giúp bạn cải thiện kỹ năng lập trình của mình, từ việc xử lý dữ liệu đến thiết kế giao diện người dùng.

2 Cơ sở lý thuyết

2.1 Phần mềm mã nguồn mở

Phần mềm mã nguồn mở là phần mềm với mã nguồn được công bố, với một giấy phép sử dụng đi kèm, cho phép bất cứ ai cũng có thể sử dụng, nghiên cứu, thay đổi cải tiến phần mềm và phân phối phần mềm ở dạng chưa thay đổi hoặc đã thay đổi

Một số lợi ích của việc phát triển phần mềm mã nguồn mở:

- Sử dụng miễn phí

  Điều đầu tiên mà có lẽ ai khi biết về mã nguồn mở có thể sử dụng cho cả mục đích cá nhân lẫn thương mại mà không phải chi trả chi phí bản quyền. Điều này khiến mọi người có thể tùy biến được hầu hết các chức năng của mã nguồn mà không bị giới hạn hay trả thêm chi phí như những phần mềm mã nguồn độc quyền khác.

- Khả năng bảo mật cao

  Mặc dù là mã nguồn miễn phí nhưng hầu hết các mã nguồn mở đều có khả năng bảo mật tuyệt vời. Lý do vì mã nguồn mở ngay từ đầu được xây dựng và đóng góp bởi cộng đồng, trong đó có vô số những nhà lập trình thiên tài. Nghĩa là khi có bất kỳ vấn đề nào xảy ra, chúng ngay lập tức được sửa đổi, khắc phục... điều này thể hiện rõ nhất tinh thần cộng đồng của mã nguồn mở.

- Can thiệp sâu vào vấn đề quản trị và điều chỉnh

  Khi làm việc trên mã nguồn mở bạn có thể toàn quyền truy cập, quản trị cũng như điều chỉnh cấu trúc để mã nguồn có thể phù hợp với nhu cầu của bản thân. Điều này cũng kích thích một số nhà lập trình từ việc sử dụng mã nguồn mở này để tạo ra nhiều phần mềm hữu ích hơn cho cộng đồng.

- Tính ổn định của mã nguồn mở

  Điều này được thể hiện căn bản nhất là vô số những website sử dụng mã nguồn mở đến hiện tại vẫn duy trì được khả năng ổn định trong vận hành liên tục mà không mắc phải bất kỳ vấn đề nào. Nguyên nhân chính có lẽ xuất phát từ việc mã nguồn mở được xây dựng dựa trên nguyên tắc tối ưu cho cộng đồng ai cũng có thể sử dụng nên về bản chất sẽ duy trì được tính ổn định trong vận hành hơn so với một số mã nguồn đóng.

2.2 Python

Python là một ngôn ngữ lập trình được sử dụng rộng rãi trong các ứng dụng web, phát triển phần mềm, khoa học dữ liệu và máy học (ML). Các nhà phát triển sử dụng Python vì nó hiệu quả, dễ học và có thể chạy trên nhiều nền tảng khác nhau. Phần mềm Python được tải xuống miễn phí, tích hợp tốt với tất cả các loại hệ thống và tăng tốc độ phát triển.

Những lợi ích của Python bao gồm:

- Các nhà phát triển có thể dễ dàng đọc và hiểu một chương trình Python vì ngôn ngữ này có cú pháp cơ bản giống tiếng Anh.
- Python giúp cải thiện năng suất làm việc của các nhà phát triển vì so với những ngôn ngữ khác, họ có thể sử dụng ít dòng mã hơn để viết một chương trình Python.
- Python có một thư viện tiêu chuẩn lớn, chứa nhiều dòng mã có thể tái sử dụng cho hầu hết mọi tác vụ. Nhờ đó, các nhà phát triển sẽ không cần phải viết mã từ đầu.

- Các nhà phát triển có thể dễ dàng sử dụng Python với các ngôn ngữ lập trình phổ biến khác như Java, C và C++.
- Cộng đồng Python tích cực hoạt động bao gồm hàng triệu nhà phát triển nhiệt tình hỗ trợ trên toàn thế giới. Nếu gặp phải vấn đề, bạn sẽ có thể nhận được sự hỗ trợ nhanh chóng từ cộng đồng.
- Trên Internet có rất nhiều tài nguyên hữu ích nếu bạn muốn học Python. Ví dụ: bạn có thể dễ dàng tìm thấy video, chỉ dẫn, tài liệu và hướng dẫn dành cho nhà phát triển.
- Python có thể được sử dụng trên nhiều hệ điều hành máy tính khác nhau, chẳng hạn như Windows, macOS, Linux và Unix.

2.3 IDE Pycharm

PyCharm là một Môi trường phát triển tích hợp (IDE) được phát triển bởi JetBrains, chuyên dụng cho lập trình Python. Đây là công cụ được đánh giá cao và sử dụng rộng rãi bởi các lập trình viên Python ở mọi trình độ, từ mới bắt đầu đến chuyên nghiệp. Điểm nổi bật của PyCharm:

- Hỗ trợ mã hóa thông minh: PyCharm cung cấp nhiều tính năng hỗ trợ mã hóa thông minh như tự động hoàn thành code, kiểm tra lỗi cú pháp, refactor code, v.v., giúp lập trình viên viết code nhanh chóng, chính xác và hiệu quả hơn.
- Điều hướng dự án dễ dàng: PyCharm giúp bạn dễ dàng điều hướng trong dự án Python của mình với các tính năng như tìm kiếm nhanh, xem cấu trúc thư mục, chuyển đổi nhanh giữa các tệp tin, v.v.
- Giải thích và gỡ lỗi hiệu quả: PyCharm cung cấp các công cụ mạnh mẽ để giải thích và gỡ lỗi code Python, giúp bạn nhanh chóng xác định và sửa lỗi trong chương trình của mình.
- Hỗ trợ phát triển web: PyCharm hỗ trợ phát triển web Python với các framework phổ biến như Django và Flask.
- Tích hợp nhiều công cụ: PyCharm tích hợp nhiều công cụ khác nhau như Git, SVN, Docker, v.v., giúp bạn quản lý dự án và phát triển hiệu quả hơn.
- Giao diện tùy chỉnh: PyCharm có giao diện người dùng trực quan và có thể tùy chỉnh cao, giúp bạn tạo môi trường làm việc phù hợp với sở thích của mình.

2.4 Thư viện hỗ trợ

Thư viện Pygame

Pygame là một thư viện mã nguồn mở cho phép lập trình game và đa phương tiện trên Python. Thư viện này cung cấp các công cụ cho việc tạo các ứng dụng đa phương tiện, chẳng hạn như các trò chơi, âm nhạc, đồ họa và các ứng dụng tương tự.

Ưu điểm của Pygame:

- Dễ học: Pygame có cú pháp đơn giản và dễ hiểu, cho phép người dùng mới bắt đầu nhanh chóng học được cách sử dụng thư viện.
- Đa nền tảng: Pygame có thể hoạt động trên nhiều nền tảng khác nhau, bao gồm Windows, Mac OS X, Linux và nhiều hệ điều hành khác.

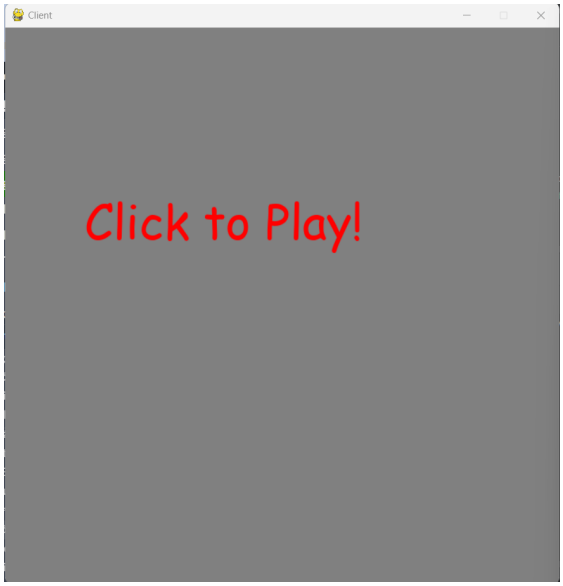
- Đa dạng: Pygame có nhiều tính năng và hỗ trợ đầy đủ cho các yêu cầu phức tạp trong lập trình game và đa phương tiện.
- Cộng đồng lớn: Pygame có một cộng đồng lớn với nhiều tài liệu hướng dẫn và các ví dụ, giúp người dùng tìm hiểu và giải quyết các vấn đề.

Nhược điểm của Pygame:

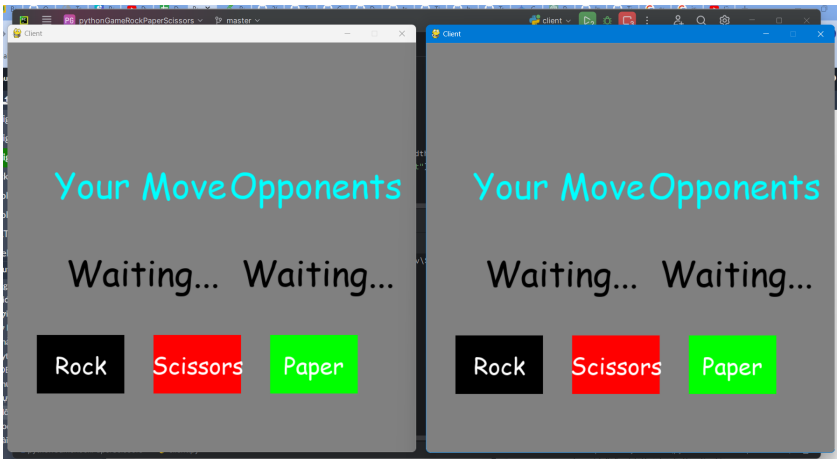
- Khả năng tối ưu hóa không tốt: Pygame không được tối ưu hóa tốt cho các ứng dụng có đồ họa phức tạp, do đó, nếu không được xử lý tốt, có thể dẫn đến giảm hiệu suất.
- Hạn chế đồ họa: Pygame không hỗ trợ các tính năng đồ họa cao cấp như DirectX hay OpenGL, vì vậy không thể sử dụng để tạo ra các trò chơi và ứng dụng đồ họa cao cấp.
- Không được bảo trì thường xuyên: Pygame không được bảo trì và cập nhật thường xuyên, do đó, có thể không hoạt động tốt với các phiên bản Python mới nhất.
- Không hỗ trợ 3D: Pygame không hỗ trợ lập trình 3D, do đó không phù hợp để tạo ra các trò chơi và ứng dụng có đồ họa 3D phức tạp.

3 Xây dựng game Rock Paper Scissor

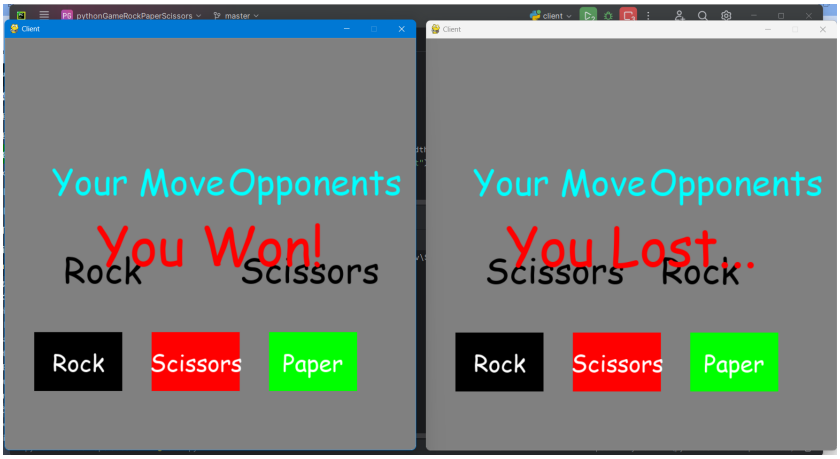
3.1 Mô tả giao diện ứng dụng game



Hình 2: Giao diện chương trình khi khởi chạy



Hình 3: Giao diện người chơi



Hình 4: Kết quả sau khi chơi

3.2 Code chức năng cho game

Socket trong python

Socket là một giao diện lập trình ứng dụng (API) mạng được sử dụng để truyền và nhận dữ liệu giữa hai chương trình chạy trên các máy tính khác nhau. Socket được ví như một đầu nối ảo cho phép hai chương trình giao tiếp với nhau thông qua mạng.

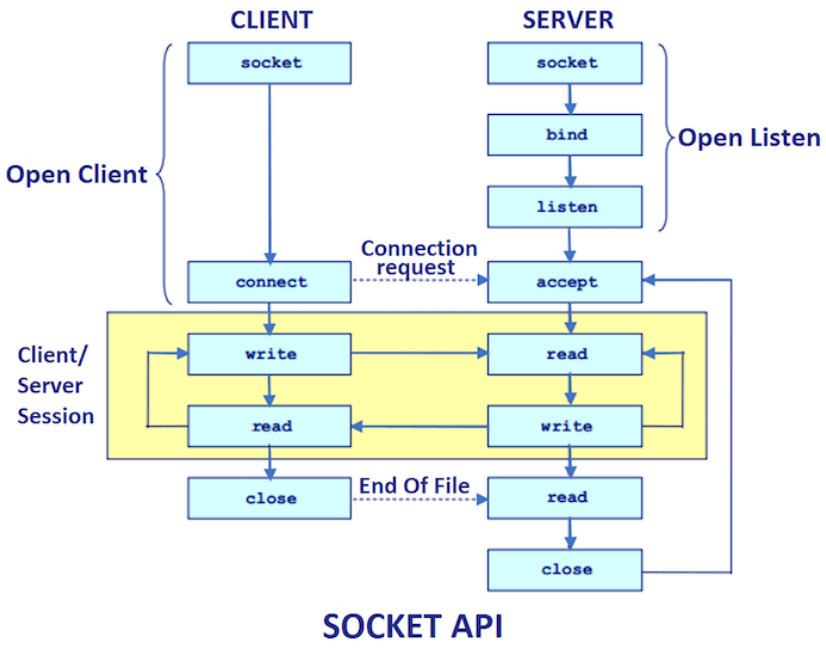
Mô-đun socket trong Python: Python cung cấp mô-đun socket để hỗ trợ lập trình socket. Mô-đun này cung cấp các hàm và lớp để tạo socket, kết nối với máy chủ, gửi và nhận dữ liệu,...

Có hai loại socket chính:

- Socket Stream (TCP): Loại socket này cung cấp kết nối đáng tin cậy giữa hai chương trình. Dữ liệu được gửi theo thứ tự và không bị mất.
- Socket Datagram (UDP): Loại socket này cung cấp kết nối không đáng tin cậy giữa hai chương trình. Dữ liệu có thể bị mất hoặc đến không đúng thứ tự.

Các bước sử dụng socket trong Python:

```
import socket %Nhap mo-dun socket

%Tao socket
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

%Ket noi voi may chu
sock.connect(('localhost', 8000))

%Gui du lieu
sock.sendall("Hello, world!")

%Nhan du lieu
data = sock.recv(1024)
print(data.decode('utf-8'))

```



Hình 5: Sơ đồ của socket

```
%Dong socket
sock.close()
```

Ứng dụng của socket:
Socket được sử dụng trong nhiều ứng dụng mạng khác nhau như:

- Web: Các trình duyệt web sử dụng socket để kết nối với các máy chủ web và tải xuống trang web.
- Email: Các máy khách email sử dụng socket để gửi và nhận email.
- Chat: Các ứng dụng chat sử dụng socket để cho phép người dùng trò chuyện với nhau.
- Chơi game: Các trò chơi trực tuyến sử dụng socket để cho phép người chơi chơi game với nhau.

server.py

```
import socket
from _thread import *
import pickle
from game import Game

server = "192.168.1.4"
port = 5555

```

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

try:
    s.bind((server, port))
except socket.error as e:
    str(e)

s.listen(2)
print("Waiting for a connection, Server Started")

connected = set()
games = {}
idCount = 0


def threaded_client(conn, p, gameId):
    global idCount
    conn.send(str.encode(str(p)))

    reply = ""
    while True:
        try:
            data = conn.recv(4096).decode()

            if gameId in games:
                game = games[gameId]

                if not data:
                    break
                else:
                    if data == "reset":
                        game.resetWent()
                    elif data != "get":
                        game.play(p, data)

                    conn.sendall(pickle.dumps(game))
            else:
                break
        except:
            break

    print("Lost connection")
    try:
        del games[gameId]
        print("Closing Game", gameId)
    except:
        pass
    idCount -= 1
    conn.close()
```

```
61  while True:
62      conn, addr = s.accept()
63      print("Connected to:", addr)
64
65      idCount += 1
66      p = 0
67      gameId = (idCount - 1)//2
68      if idCount % 2 == 1:
69          games[gameId] = Game(gameId)
70          print("Creating a new game...")
71      else:
72          games[gameId].ready = True
73          p = 1
74
75
76      start_new_thread(threaded_client, (conn, p, gameId))
```

network.py

```
1   import socket
2   import pickle
3
4
5   class Network:
6       def __init__(self):
7           self.client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
8           self.server = "192.168.1.4"
9           self.port = 5555
10          self.addr = (self.server, self.port)
11          self.p = self.connect()
12
13      def getP(self):
14          return self.p
15
16      def connect(self):
17          try:
18              self.client.connect(self.addr)
19              return self.client.recv(2048).decode()
20          except:
21              pass
22
23      def send(self, data):
24          try:
25              self.client.send(str.encode(data))
26              return pickle.loads(self.client.recv(2048*2))
27          except socket.error as e:
28              print(e)
```

game.py

```
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

def get_player_move(self, p):
    """
    :param p: [0,1]
    :return: Move
    """
    return self.moves[p]

def play(self, player, move):
    self.moves[player] = move
    if player == 0:
        self.p1Went = True
    else:
        self.p2Went = True

def connected(self):
    return self.ready

def bothWent(self):
    return self.p1Went and self.p2Went

def winner(self):

    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

    return winner

```

```
52    def resetWent(self):
53        self.p1Went = False
54        self.p2Went = False
```

client.py

```
1     import pygame
2     from network import Network
3     import pickle
4     pygame.font.init()
5
6     width = 700
7     height = 700
8     win = pygame.display.set_mode((width, height))
9     pygame.display.set_caption("Client")
10
11
12    class Button:
13        def __init__(self, text, x, y, color):
14            self.text = text
15            self.x = x
16            self.y = y
17            self.color = color
18            self.width = 150
19            self.height = 100
20
21        def draw(self, win):
22            pygame.draw.rect(win, self.color, (self.x, self.y, self.width,
                  self.height))
23            font = pygame.font.SysFont("comicsans", 40)
24            text = font.render(self.text, 1, (255,255,255))
25            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)))
26
27        def click(self, pos):
28            x1 = pos[0]
29            y1 = pos[1]
30            if self.x <= x1 <= self.x + self.width and self.y <= y1 <= self.y +
                  self.height:
31                return True
32            else:
33                return False
34
35
36    def redrawWindow(win, game, p):
37        win.fill((128,128,128))
38
39        if not(game.connected()):
40            font = pygame.font.SysFont("comicsans", 80)
41            text = font.render("Waiting for Player...", 1, (255,0,0), True)
```

```
            win.blit(text, (width/2 - text.get_width()/2, height/2 -
                text.get_height()/2))
        else:
            font = pygame.font.SysFont("comicsans", 60)
            text = font.render("Your Move", 1, (0, 255,255))
            win.blit(text, (80, 200))

            text = font.render("Opponents", 1, (0, 255, 255))
            win.blit(text, (380, 200))

            move1 = game.get_player_move(0)
            move2 = game.get_player_move(1)
            if game.bothWent():
                text1 = font.render(move1, 1, (0,0,0))
                text2 = font.render(move2, 1, (0, 0, 0))
            else:
                if game.p1Went and p == 0:
                    text1 = font.render(move1, 1, (0,0,0))
                elif game.p1Went:
                    text1 = font.render("Locked In", 1, (0, 0, 0))
                else:
                    text1 = font.render("Waiting...", 1, (0, 0, 0))

                if game.p2Went and p == 1:
                    text2 = font.render(move2, 1, (0,0,0))
                elif game.p2Went:
                    text2 = font.render("Locked In", 1, (0, 0, 0))
                else:
                    text2 = font.render("Waiting...", 1, (0, 0, 0))

            if p == 1:
                win.blit(text2, (100, 350))
                win.blit(text1, (400, 350))
            else:
                win.blit(text1, (100, 350))
                win.blit(text2, (400, 350))

            for btn in btns:
                btn.draw(win)

        pygame.display.update()


    btns = [Button("Rock", 50, 500, (0,0,0)), Button("Scissors", 250, 500,
        (255,0,0)), Button("Paper", 450, 500, (0,255,0))]
    def main():
        run = True
        clock = pygame.time.Clock()
        n = Network()
        player = int(n.getP())
        print("You are player", player)

```

```
92        while run:
93            clock.tick(60)
94            try:
95                game = n.send("get")
96            except:
97                run = False
98                print("Couldn't get game")
99                break
100
101           if game.bothWent():
102               redrawWindow(win, game, player)
103               pygame.time.delay(500)
104               try:
105                   game = n.send("reset")
106               except:
107                   run = False
108                   print("Couldn't get game")
109                   break
110
111               font = pygame.font.SysFont("comicsans", 90)
112               if (game.winner() == 1 and player == 1) or (game.winner() == 0 and
                      player == 0):
113                   text = font.render("You Won!", 1, (255,0,0))
114               elif game.winner() == -1:
115                   text = font.render("Tie Game!", 1, (255,0,0))
116               else:
117                   text = font.render("You Lost...", 1, (255, 0, 0))
118
119               win.blit(text, (width/2 - text.get_width()/2, height/2 -
                      text.get_height()/2))
120               pygame.display.update()
121               pygame.time.delay(2000)
122
123           for event in pygame.event.get():
124               if event.type == pygame.QUIT:
125                   run = False
126                   pygame.quit()
127
128               if event.type == pygame.MOUSEBUTTONDOWN:
129                   pos = pygame.mouse.get_pos()
130                   for btn in btns:
131                       if btn.click(pos) and game.connected():
132                           if player == 0:
133                               if not game.p1Went:
134                                   n.send(btn.text)
135                           else:
136                               if not game.p2Went:
137                                   n.send(btn.text)
138
139           redrawWindow(win, game, player)
140
141   def menu_screen():
```

```
        run = True
        clock = pygame.time.Clock()

        while run:
            clock.tick(60)
            win.fill((128, 128, 128))
            font = pygame.font.SysFont("comicsans", 60)
            text = font.render("Click to Play!", 1, (255,0,0))
            win.blit(text, (100,200))
            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()

    while True:
        menu_screen()
```

3.3 Cài đặt và thiết lập môi trường

- Python version: python 3.9.0
- OS: Ubuntu 22.04, Windown 11 22H2, macOS 14.4.1

Tải ứng dụng

```
git clone https://github.com/thieuanhsang/pythonGameRockPaperScissors.git
```

Download package pygame

```
pip install pygame
```

Chỉnh lại cấu hình server trong file server.py và network.py

```
ipconfig
```

Hình 6: Xem ip của host

```
server = "192.168.1.4"
port = 5555
```

4 Xây dựng game đối kháng (Street Fighter)

4.1 Ý tưởng thực hiện

Trò chơi điện tử đối kháng là một thể loại trò chơi điện tử hay còn gọi là trò chơi đánh nhau (tiếng Anh fighting game). Trò chơi điện tử đối kháng là các trò chơi mà trong đó người chơi điều khiển một nhân vật tham gia một cuộc đấu tay đôi với một nhân vật khác trên một màn hình có giới hạn. Các nhân vật thường sẽ có khả năng ngang nhau. Người chơi sẽ phải dùng các nút bấm để tung ra các đòn đánh cơ bản, hay các đòn đặc biệt như vật, khóa đòn, phản công hay các "tuyệt chiêu". Việc bấm các nút nhanh và hợp lý sẽ giúp tung ra các đòn đánh liên hoàn ("combo"). Lấy ý tưởng từ các tựa game đối kháng nổi bật như "Street Fighter, Mortal Combat,

Hình 7: Minh họa game đối kháng

Tekken..." nhóm chúng em quyết định lựa chọn thực hiện xây dựng một tựa game bằng ngôn ngữ lập trình Python với thư viện PyGame.

4.2 Giao diện của trò chơi

Cách điều khiển di chuyển

Nhân vật 1: Sử dụng tổ hợp phím D và A để nhân vật có thể Tiến và Lùi, W để nhảy

Nhân vật 2: Sử dụng tổ hợp phím mũi tên trái và phải để nhân vật có thể Tiền và Lùi, mũi tên lên để nhảy

Cách tấn công

Nhân vật 1: Sử dụng phím R và T để nhân vật có thể tấn công đối phương.

Nhân vật 2: Sử dụng phím 1 và 2 để nhân vật có thể tấn công đối phương.

Mục tiêu

Mục tiêu của trò chơi là đánh bại đối thủ bằng cách giảm thanh máu của họ đến 0 và giành được điểm số cao hơn trong mỗi trận đấu.

Nguyên tắc chơi:

Lựa chọn nhân vật:

Hình 8: Giao diện của game đối kháng (Street Fighter)

- -Trước khi trận đấu bắt đầu, sẽ có 3 giây để người chơi lựa chọn một trong hai nhân vật có sẵn.
- -Hai nhân vật là Võ Sĩ và Pháp Sư, mỗi nhân vật có các đặc điểm riêng, bao gồm tấn công, phòng thủ và tốc độ di chuyển.

Trận đấu:

- -Trận đấu diễn ra trong một môi trường đối kháng không giới hạn.
- -Hai người chơi sẽ đối đầu với nhau trong một trận đấu không có hạn chế về thời gian cho đến khi một trong hai người chơi hết máu.

Điều khiển nhân vật:

- -Người chơi sử dụng bàn phím để điều khiển nhân vật của mình.
- Mỗi phím trên bàn phím được gán cho một hành động cụ thể như tấn công, né tránh, phòng thủ và sử dụng kỹ năng đặc biệt cho từng nhân vật.

Hạ gục đối thủ:

- -Mục tiêu của mỗi người chơi là giảm máu của đối thủ đến 0 bằng cách sử dụng kỹ năng và chiến thuật.
- -Người chơi có thể tấn công trực tiếp hoặc sử dụng kỹ năng để gây sát thương.

Xác định người chiến thắng:

- -Người chơi nào hạ gục đối thủ đầu tiên sẽ được coi là người chiến thắng của trận đấu và giành được 1 điểm.
- -Nếu cả hai người chơi hết máu cùng một lúc, trận đấu được coi là hòa và không ai được tính điểm.

4.3 Code chức năng cho game

fighter.py

```
  import pygame

class Fighter():
 def __init__(self, player, x, y, flip, data, sprite_sheet, animation_steps,
     sound):
   self.player = player
   self.size = data[0]
   self.image_scale = data[1]
   self.offset = data[2]
   self.flip = flip
   self.animation_list = self.load_images(sprite_sheet, animation_steps)
   self.action = 0#0:idle #1:run #2:jump #3:attack1 #4: attack2 #5:hit #6:death
   self.frame_index = 0
   self.image = self.animation_list[self.action][self.frame_index]
   self.update_time = pygame.time.get_ticks()
   self.rect = pygame.Rect((x, y, 80, 180))
   self.vel_y = 0
   self.running = False
   self.jump = False
   self.attacking = False
   self.attack_type = 0
   self.attack_cooldown = 0
   self.attack_sound = sound
   self.hit = False
   self.health = 100
   self.alive = True


 def load_images(self, sprite_sheet, animation_steps):
   #extract images from spritesheet
   animation_list = []
   for y, animation in enumerate(animation_steps):
     temp_img_list = []
     for x in range(animation):
       temp_img = sprite_sheet.subsurface(x * self.size, y * self.size, self.size,
           self.size)
       temp_img_list.append(pygame.transform.scale(temp_img, (self.size *
           self.image_scale, self.size * self.image_scale)))
     animation_list.append(temp_img_list)
   return animation_list


 def move(self, screen_width, screen_height, surface, target, round_over):
   SPEED = 10
   GRAVITY = 2
   dx = 0
   dy = 0
   self.running = False
```

```
    self.attack_type = 0

    #get keypresses
    key = pygame.key.get_pressed()

    #can only perform other actions if not currently attacking
    if self.attacking == False and self.alive == True and round_over == False:
      #check player 1 controls
      if self.player == 1:
        #movement
        if key[pygame.K_a]:
          dx = -SPEED
          self.running = True
        if key[pygame.K_d]:
          dx = SPEED
          self.running = True
        #jump
        if key[pygame.K_w] and self.jump == False:
          self.vel_y = -30
          self.jump = True
        #attack
        if key[pygame.K_r] or key[pygame.K_t]:
          self.attack(target)
          #determine which attack type was used
          if key[pygame.K_r]:
            self.attack_type = 1
          if key[pygame.K_t]:
            self.attack_type = 2


      #check player 2 controls
      if self.player == 2:
        #movement
        if key[pygame.K_LEFT]:
          dx = -SPEED
          self.running = True
        if key[pygame.K_RIGHT]:
          dx = SPEED
          self.running = True
        #jump
        if key[pygame.K_UP] and self.jump == False:
          self.vel_y = -30
          self.jump = True
        #attack
        if key[pygame.K_KP1] or key[pygame.K_KP2]:
          self.attack(target)
          #determine which attack type was used
          if key[pygame.K_KP1]:
            self.attack_type = 1
          if key[pygame.K_KP2]:
            self.attack_type = 2

```

```
98
99    #apply gravity
100   self.vel_y += GRAVITY
101   dy += self.vel_y
102
103   #ensure player stays on screen
104   if self.rect.left + dx < 0:
105     dx = -self.rect.left
106   if self.rect.right + dx > screen_width:
107     dx = screen_width - self.rect.right
108   if self.rect.bottom + dy > screen_height - 110:
109     self.vel_y = 0
110     self.jump = False
111     dy = screen_height - 110 - self.rect.bottom
112
113   #ensure players face each other
114   if target.rect.centerx > self.rect.centerx:
115     self.flip = False
116   else:
117     self.flip = True
118
119   #apply attack cooldown
120   if self.attack_cooldown > 0:
121     self.attack_cooldown -= 1
122
123   #update player position
124   self.rect.x += dx
125   self.rect.y += dy
126
127
128 #handle animation updates
129 def update(self):
130   #check what action the player is performing
131   if self.health <= 0:
132     self.health = 0
133     self.alive = False
134     self.update_action(6)#6:death
135   elif self.hit == True:
136     self.update_action(5)#5:hit
137   elif self.attacking == True:
138     if self.attack_type == 1:
139       self.update_action(3)#3:attack1
140     elif self.attack_type == 2:
141       self.update_action(4)#4:attack2
142   elif self.jump == True:
143     self.update_action(2)#2:jump
144   elif self.running == True:
145     self.update_action(1)#1:run
146   else:
147     self.update_action(0)#0:idle
148
149   animation_cooldown = 50
```

```
    #update image
    self.image = self.animation_list[self.action][self.frame_index]
    #check if enough time has passed since the last update
    if pygame.time.get_ticks() - self.update_time > animation_cooldown:
      self.frame_index += 1
      self.update_time = pygame.time.get_ticks()
    #check if the animation has finished
    if self.frame_index >= len(self.animation_list[self.action]):
      #if the player is dead then end the animation
      if self.alive == False:
        self.frame_index = len(self.animation_list[self.action]) - 1
      else:
        self.frame_index = 0
        #check if an attack was executed
        if self.action == 3 or self.action == 4:
          self.attacking = False
          self.attack_cooldown = 20
        #check if damage was taken
        if self.action == 5:
          self.hit = False
          #if the player was in the middle of an attack, then the attack is stopped
          self.attacking = False
          self.attack_cooldown = 20


  def attack(self, target):
    if self.attack_cooldown == 0:
      #execute attack
      self.attacking = True
      self.attack_sound.play()
      attacking_rect = pygame.Rect(self.rect.centerx - (2 * self.rect.width *
          self.flip), self.rect.y, 2 * self.rect.width, self.rect.height)
      if attacking_rect.colliderect(target.rect):
        target.health -= 10
        target.hit = True


  def update_action(self, new_action):
    #check if the new action is different to the previous one
    if new_action != self.action:
      self.action = new_action
      #update the animation settings
      self.frame_index = 0
      self.update_time = pygame.time.get_ticks()

  def draw(self, surface):
    img = pygame.transform.flip(self.image, self.flip, False)
    surface.blit(img, (self.rect.x - (self.offset[0] * self.image_scale),
        self.rect.y - (self.offset[1] * self.image_scale)))
```

main.py

```
    import pygame
from pygame import mixer
from fighter import Fighter

mixer.init()
pygame.init()

#create game window
SCREEN_WIDTH = 1000
SCREEN_HEIGHT = 600

screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
pygame.display.set_caption("Game Doi Khang")

#set framerate
clock = pygame.time.Clock()
FPS = 60

#define colours
RED = (255, 0, 0)
YELLOW = (255, 255, 0)
WHITE = (255, 255, 255)

#define game variables
intro_count = 3
last_count_update = pygame.time.get_ticks()
score = [0, 0]#player scores. [P1, P2]
round_over = False
ROUND_OVER_COOLDOWN = 2000

#define fighter variables
WARRIOR_SIZE = 162
WARRIOR_SCALE = 4
WARRIOR_OFFSET = [72, 56]
WARRIOR_DATA = [WARRIOR_SIZE, WARRIOR_SCALE, WARRIOR_OFFSET]
WIZARD_SIZE = 250
WIZARD_SCALE = 3
WIZARD_OFFSET = [112, 107]
WIZARD_DATA = [WIZARD_SIZE, WIZARD_SCALE, WIZARD_OFFSET]

#load music and sounds
pygame.mixer.music.load("assets/audio/music.mp3")
pygame.mixer.music.set_volume(0.5)
pygame.mixer.music.play(-1, 0.0, 5000)
sword_fx = pygame.mixer.Sound("assets/audio/sword.wav")
sword_fx.set_volume(0.5)
magic_fx = pygame.mixer.Sound("assets/audio/magic.wav")
magic_fx.set_volume(0.75)

#load background image
```

```
bg_image = pygame.image.load("assets/images/background/background.jpg").convert_alpha()

#load spritesheets
warrior_sheet = pygame.image.load("assets/images/warrior/Sprites/warrior.png").convert_alpha()
wizard_sheet = pygame.image.load("assets/images/wizard/Sprites/wizard.png").convert_alpha()

#load vicory image
victory_img = pygame.image.load("assets/images/icons/victory.png").convert_alpha()

#define number of steps in each animation
WARRIOR_ANIMATION_STEPS = [10, 8, 1, 7, 7, 3, 7]
WIZARD_ANIMATION_STEPS = [8, 8, 1, 8, 8, 3, 7]

#define font
count_font = pygame.font.Font("assets/fonts/turok.ttf", 80)
score_font = pygame.font.Font("assets/fonts/turok.ttf", 30)

#function for drawing text
def draw_text(text, font, text_col, x, y):
  img = font.render(text, True, text_col)
  screen.blit(img, (x, y))

#function for drawing background
def draw_bg():
  scaled_bg = pygame.transform.scale(bg_image, (SCREEN_WIDTH, SCREEN_HEIGHT))
  screen.blit(scaled_bg, (0, 0))

#function for drawing fighter health bars
def draw_health_bar(health, x, y):
  ratio = health / 100
  pygame.draw.rect(screen, WHITE, (x - 2, y - 2, 404, 34))
  pygame.draw.rect(screen, RED, (x, y, 400, 30))
  pygame.draw.rect(screen, YELLOW, (x, y, 400 * ratio, 30))


#create two instances of fighters
fighter_1 = Fighter(1, 200, 310, False, WARRIOR_DATA, warrior_sheet, WARRIOR_ANIMATION_STEPS, sword_fx)
fighter_2 = Fighter(2, 700, 310, True, WIZARD_DATA, wizard_sheet, WIZARD_ANIMATION_STEPS, magic_fx)

#game loop
run = True
while run:

  clock.tick(FPS)

  #draw background
  draw_bg()
```

```
98
99  #show player stats
100 draw_health_bar(fighter_1.health, 20, 20)
101 draw_health_bar(fighter_2.health, 580, 20)
102 draw_text("P1: " + str(score[0]), score_font, RED, 20, 60)
103 draw_text("P2: " + str(score[1]), score_font, RED, 580, 60)
104
105 #update countdown
106 if intro_count <= 0:
107   #move fighters
108   fighter_1.move(SCREEN_WIDTH, SCREEN_HEIGHT, screen, fighter_2, round_over)
109   fighter_2.move(SCREEN_WIDTH, SCREEN_HEIGHT, screen, fighter_1, round_over)
110 else:
111   #display count timer
112   draw_text(str(intro_count), count_font, RED, SCREEN_WIDTH / 2, SCREEN_HEIGHT /
        3)
113   #update count timer
114   if (pygame.time.get_ticks() - last_count_update) >= 1000:
115     intro_count -= 1
116     last_count_update = pygame.time.get_ticks()
117
118 #update fighters
119 fighter_1.update()
120 fighter_2.update()
121
122 #draw fighters
123 fighter_1.draw(screen)
124 fighter_2.draw(screen)
125
126 #check for player defeat
127 if round_over == False:
128   if fighter_1.alive == False:
129     score[1] += 1
130     round_over = True
131     round_over_time = pygame.time.get_ticks()
132   elif fighter_2.alive == False:
133     score[0] += 1
134     round_over = True
135     round_over_time = pygame.time.get_ticks()
136 else:
137   #display victory image
138   screen.blit(victory_img, (360, 150))
139   if pygame.time.get_ticks() - round_over_time > ROUND_OVER_COOLDOWN:
140     round_over = False
141     intro_count = 3
142     fighter_1 = Fighter(1, 200, 310, False, WARRIOR_DATA, warrior_sheet,
          WARRIOR_ANIMATION_STEPS, sword_fx)
143     fighter_2 = Fighter(2, 700, 310, True, WIZARD_DATA, wizard_sheet,
          WIZARD_ANIMATION_STEPS, magic_fx)
144
145 #event handler
146 # Who get 3 point first is winner
```

```
147  if score[0] == 3 or score[1] == 3:
148    run = False
149
150  # Close btn
151  for event in pygame.event.get():
152    if event.type == pygame.QUIT:
153      run = False
154
155  #update display
156  pygame.display.update()
157
158 #exit pygame
159 pygame.quit()
```

Tài liệu

[1] Python documentation "**link: https://docs.python.org/3/**", *Python3*, lần truy cập cuối: 8/05/2024.

[2] Pygame documentation "**link: https://www.pygame.org/docs/**", *PyGame*, lần truy cập cuối: 8/05/2024.

[3] Python Online Game "**link: https://www.youtube.com/playlist?list=PLzMcBGfZo4-kR7Rh-7JCVDN8lm3Utumvq**", *Tech With Tim*, lần truy cập cuối: 8/05/2024.