

TRƯỜNG ĐẠI HỌC SÀI GÒN
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO MÔN HỌC
PHÁT TRIỂN ỨNG DỤNG PHẦN MỀM
MÃ NGUỒN MỞ
ĐỀ TÀI
GAME ĐỐI KHÁNG

GV hướng dẫn: TÙ LÃNG PHIÊU

Nhóm sinh viên thực hiện:

<i>Họ và tên</i>	<i>MSSV</i>
Trần Minh Quân	3120410439
Lê Hồng Việt	3120410609

Liên hệ nhóm:

Gmail 1: rjck024vipro@gmai.com

Gmail 2: vietlehong22@gmail.com

Tp, Hồ Chí Minh, 2024

Mục lục

1 LỜI MỞ ĐẦU	2
2 Giới thiệu về đề tài	3
2.1 Lý do chọn đề tài	3
2.2 Mô tả đề tài	3
2.3 Đối tượng và phạm vi nghiên cứu	3
3 Cơ sở lý thuyết	4
3.1 Các thư viện sử dụng trong đề tài, ưu điểm và nhược điểm các thư viện	4
3.1.1 Thư viện Pygame:	4
3.1.2 Thư viện Socket:	5
3.1.3 Thư viện Threading:	5
4 Thiết kế ứng dụng	7
4.1 Mô tả thiết kế, ý nghĩa các bảng dữ liệu và các trường trong bảng (nếu có). Trình bày cấu trúc mã nguồn, mô hình ứng dụng, các tính năng được xây dựng, flowchart,...	7
4.2 Class main	7
4.3 Class fighter	9
4.4 Class server	12
4.5 Class client	14
4.6 Class menuControl	14
4.7 Class menuOnline	15
4.8 Class menuOffline	18
5 Hiện thực	22
5.1 Các tính năng được thiết kế và hình ảnh	22
6 Cách thức cài đặt ứng dụng, môi trường chạy ứng dụng,	25
6.1 Cài đặt Python trên Widown hoặc Linux	25
6.2 Cài đặt Pycharm	28
6.3 Opent project game vào Pycharm	31
7 Nhiệm vụ, vai trò của từng thành viên trong nhóm.	33



1 LỜI MỞ ĐẦU

Hiện nay, ngành công nghệ thông tin đã trở thành một phần không thể thiếu trong cuộc sống, công nghệ thông tin đã và đang đóng vai trò quan trọng trong các ngành khoa học, kỹ thuật, kinh tế, xã hội tại các nước trên thế giới. Nó mang lại giá trị về kinh tế, giáo dục và đặc biệt là mang tính giải trí cao.

Sự ra đời của Internet, game online xuất hiện và nhanh chóng gây được sức hút to lớn. Hiện nay game online cũng đang tiếp tục phát triển mạnh mẽ. Năm bắt được xu thế của người dùng, nhóm chúng em đã lên ý tưởng và xây dựng ứng dụng trò chơi giải trí “Game đối kháng cổ điển” dựa trên ngôn ngữ lập trình python



2 Giới thiệu về đề tài

2.1 Lý do chọn đề tài

Trò chơi đối kháng là một đề tài phù hợp để áp dụng các kiến thức và kỹ năng lập trình. Vừa là cơ hội học hỏi về cách thiết kế game, đồ họa và âm nhạc. Biết cách để người chơi tương tác với nhau qua kết nối mạng.

2.2 Mô tả đề tài

Trong trò chơi đối kháng, hai người chơi sẽ sử dụng các phím trên bàn phím để điều khiển nhân vật của mình di chuyển và chiến đấu với nhau cho đến khi 1 trong 2 nhân vật bị hạ gục thì ván đấu đó sẽ kết thúc và bắt đầu một ván đấu mới.

2.3 Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu: trò chơi đối kháng.

Phạm vi nghiên cứu:

- Cơ chế trò chơi
- Đồ họa trò chơi
- Âm thanh trong chơi



3 Cơ sở lý thuyết

3.1 Các thư viện sử dụng trong đề tài, ưu điểm và nhược điểm các thư viện

3.1.1 Thư viện Pygame:

Ưu điểm:

1. Đơn giản và dễ học: Pygame cung cấp một cách tiếp cận dễ dàng cho việc phát triển trò chơi và ứng dụng đa phương tiện trên nền tảng Python.
2. Đa nền tảng: Pygame hoạt động trên nhiều hệ điều hành như Windows, Mac OS X, Linux, iOS và Android, tạo điều kiện thuận lợi cho việc phát triển ứng dụng đa nền tảng.
3. Hỗ trợ đa phương tiện: Pygame cung cấp các công cụ để xử lý âm thanh, đồ họa và đầu vào người dùng, giúp người dùng tạo ra các ứng dụng đa phương tiện phức tạp.
4. Cộng đồng mạnh mẽ: Có một cộng đồng nhiệt tình và sẵn sàng chia sẻ kiến thức, ví dụ như hướng dẫn, mã nguồn mở và phần mềm miễn phí.

Nhược điểm:

1. Hiệu suất hạn chế: Đối với các ứng dụng đa phương tiện phức tạp, Pygame có thể đạt đến giới hạn hiệu suất do sử dụng ngôn ngữ Python không tối ưu cho xử lý đồ họa và âm thanh.
2. Thiếu tính linh hoạt: So với các thư viện đồ họa và game engine mạnh mẽ khác, Pygame có thể thiếu các tính năng linh hoạt và công cụ phân tích mạnh mẽ hơn.
3. Hạn chế cho các trò chơi lớn quy mô: Pygame, chỉ với thư viện cơ bản, có thể gặp hạn chế đối với việc phát triển các trò chơi lớn quy mô với yêu cầu cao về hiệu suất và tính năng.
4. Còn tiềm ẩn một số lỗi và rủi ro an ninh: Do sự phát triển không ngừng, Pygame vẫn có thể chứa đựng một số lỗi và rủi ro an ninh mà cần được cập nhật và vá sau này.



3.1.2 Thư viện Socket:

Ưu điểm:

1. Đơn giản và dễ sử dụng: Thư viện socket trong Python cung cấp một cách đơn giản và dễ dàng để tạo ra và quản lý các kết nối mạng.
2. Độ linh hoạt: Socket cho phép tùy chỉnh cấu hình kết nối mạng theo nhu cầu cụ thể, bao gồm cả việc thiết lập các giao thức truyền dữ liệu khác nhau như TCP và UDP.
3. Hỗ trợ đa nhiệm: Thư viện socket trong Python cho phép xử lý nhiều kết nối mạng cùng một lúc, điều này rất hữu ích trong việc xây dựng ứng dụng mạng đa luồng.
4. Sự tương thích: Thư viện socket là một phần của thư viện chuẩn của Python, điều này đảm bảo tính tương thích cao với hệ thống và các phiên bản Python khác nhau.

Nhược điểm:

1. Phức tạp khi xử lý lỗi: Việc xử lý các tình huống lỗi và ngoại lệ khi sử dụng thư viện socket có thể phức tạp, đặc biệt khi xử lý kết nối mạng không ổn định.
2. Cấp thấp: Đôi khi, việc làm việc với thư viện socket yêu cầu sự hiểu biết về cấp thấp của giao thức mạng, điều này có thể làm tăng độ phức tạp trong quá trình phát triển ứng dụng.
3. Thiếu tính bảo mật mặc định: Khi sử dụng socket, người lập trình phải tự quản lý và thực hiện các biện pháp bảo mật mạng như mã hóa dữ liệu và xác thực người dùng, vì socket không có tính năng bảo mật mặc định.
4. Khả năng xảy ra lỗi không dự đoán được: Việc làm việc với mạng và kết nối có thể dẫn đến các tình huống lỗi không dự đoán được, đặc biệt khi xử lý kết nối mạng qua Internet.

3.1.3 Thư viện Threading:

Ưu điểm:

1. Đa luồng (Multithreading): Cho phép thực hiện nhiều công việc cùng một lúc, tận dụng tối đa tài nguyên hệ thống và tăng hiệu suất xử lý.



2. Tăng tốc độ xử lý: Giúp giảm thời gian xử lý tác vụ bằng cách chia nhỏ chúng thành các luồng riêng biệt.

3. Phản hồi người dùng tốt: Hiệu suất và phản hồi người dùng được cải thiện khi có thể thực hiện các tác vụ đa luồng trong khi vẫn có khả năng tương tác với giao diện người dùng.

Nhược điểm:

1. Gặp vấn đề với GIL: Global Interpreter Lock (GIL) hạn chế việc thực thi đa luồng song song ở mức độ mã nguồn Python, làm giảm hiệu suất thực sự của đa luồng.

2. Rủi ro về đồng bộ hóa: Việc quản lý sự đồng bộ hóa giữa các luồng có thể gây ra các vấn đề như deadlock và race conditions.

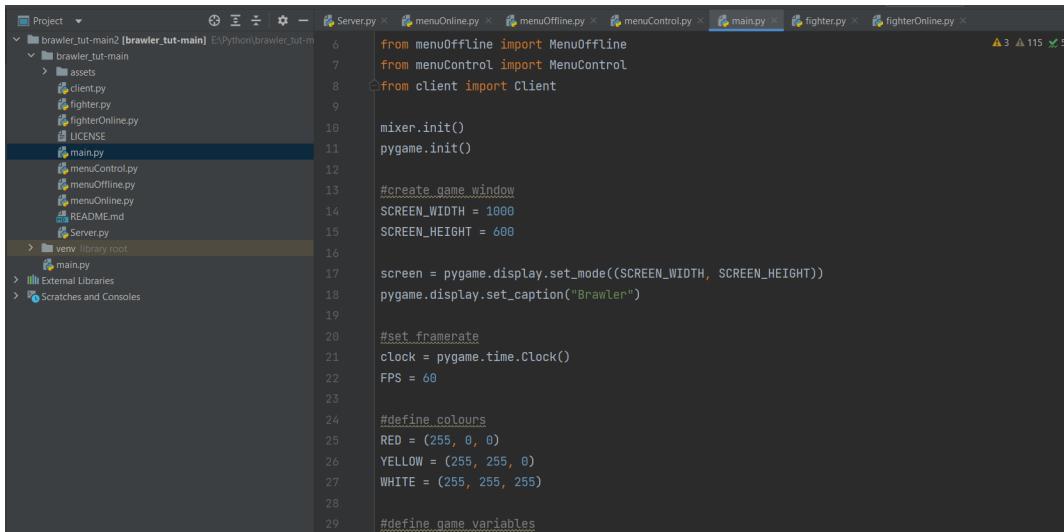
3. Tính ổn định: Cần phải hết sức cẩn trọng trong việc quản lý luồng để tránh gây ra lỗi và khó debug khi có nhiều luồng đồng thời chạy.

4 Thiết kế ứng dụng

4.1 Mô tả thiết kế, ý nghĩa các bảng dữ liệu và các trường trong bảng (nếu có). Trình bày cấu trúc mã nguồn, mô hình ứng dụng, các tính năng được xây dựng, flowchart,...

4.2 Class main

- Class **mail** là class chính



The screenshot shows the PyCharm IDE interface. On the left is the project tree for 'brawler_tut-main2' containing files like Server.py, menuOnline.py, menuOffline.py, menuControl.py, main.py (selected), fighter.py, fighterOnline.py, client.py, assets, LICENSE, README.md, and Server.py. The main editor window displays the content of main.py:

```
6  from menuOffline import MenuOffline
7  from menuControl import MenuControl
8  from client import Client
9
10 mixer.init()
11 pygame.init()
12
13 #create game window
14 SCREEN_WIDTH = 1000
15 SCREEN_HEIGHT = 600
16
17 screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
18 pygame.display.set_caption("Brawler")
19
20 #set framerate
21 clock = pygame.time.Clock()
22 FPS = 60
23
24 #define colours
25 RED = (255, 0, 0)
26 YELLOW = (255, 255, 0)
27 WHITE = (255, 255, 255)
28
29 #define game variables
```

- Khởi tạo menu online/offline dựa vào lựa chọn của người chơi

```
#create menu
menuControl = MenuControl()
status = menuControl.createMenu()

#create menu offline
if status == 1:
    client = Client()
    menuOnline = MenuOnline()
    ft, ft2, map = menuOnline.createMenu(client)
else:
    client = 1
    menuOffline = MenuOffline()
    ft, ft2, map = menuOffline.createMenu()
```

- Tạo các đối tượng dựa vào lựa chọn của người chơi về nhân vật

```
#create two instances of fighters
if status == 1:
    if ft == 1:
        fighter_1 = FighterOnline(1, 200, 310, False, WARRIOR_DATA, warrior_sheet, WARRIOR_ANIMATION_STEPS,
    elif ft == 2:
        fighter_1 = FighterOnline(1, 200, 310, False, WIZARD_DATA, wizard_sheet, WIZARD_ANIMATION_STEPS,
    if ft2 == 1:
        fighter_2 = FighterOnline(2, 700, 310, False, WARRIOR_DATA, warrior_sheet, WARRIOR_ANIMATION_STEPS,
    elif ft2 == 2:
        fighter_2 = FighterOnline(2, 700, 310, False, WIZARD_DATA, wizard_sheet, WIZARD_ANIMATION_STEPS,
else:
    if ft == 1:
        fighter_1 = Fighter(1, 200, 310, False, WARRIOR_DATA, warrior_sheet, WARRIOR_ANIMATION_STEPS, sword_fx
    elif ft == 2:
        fighter_1 = Fighter(1, 200, 310, False, WIZARD_DATA, wizard_sheet, WIZARD_ANIMATION_STEPS, magic_fx
    if ft2 == 1:
        fighter_2 = Fighter(2, 700, 310, False, WARRIOR_DATA, warrior_sheet, WARRIOR_ANIMATION_STEPS, sword_fx
    elif ft2 == 2:
        fighter_2 = Fighter(2, 700, 310, False, WIZARD_DATA, wizard_sheet, WIZARD_ANIMATION_STEPS, magic_fx

#name loop
```

- Xử lý kết thúc vòng đấu, hiển thị thông báo, và chuẩn bị vòng đấu mới
- Xử lý sự kiện thoát game

4.3 Class fighter

- Class **fighter** tạo ra các đối tượng nhân vật:
- Hàm tải hình ảnh và các bước animation trả về danh sách hình ảnh

```
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)  
            temp_img_list.append(pygame.transform.scale(temp_img, (self.size, self.size)))  
        animation_list.append(temp_img_list)  
    return animation_list
```

- Hàm xử lý di chuyển nhân vật dựa trên các sự kiện từ bàn phím, giới hạn tầm di chuyển



```
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:
        #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
```

- Cập nhật trạng thái nhân vật, tấn công, xử lý sự kiện nhân vật bị tấn công hoặc chết

```
def update(self):
    #check what action the player is performing
    if self.health <= 0:
        self.health = 0
        self.alive = False
        self.update_action(6)#6:death
    elif self.hit == True:
        self.update_action(5)#5:hit
    elif self.attacking == True:
        if self.attack_type == 1:
            self.update_action(3)#3:attack1
        elif self.attack_type == 2:
            self.update_action(4)#4:attack2
    elif self.jump == True:
        self.update_action(2)#2:jump
    elif self.running == True:
        self.update_action(1)#1:run
    else:
        self.update_action(0)#0:idle

    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
```

- Hàm xử lý nhân vật tấn công, giảm máu, xác định các va chạm giữa nhân vật

```
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 / 3),
                                      self.rect.centery,
                                      self.rect.width / 3, 10)
        if attacking_rect.colliderect(target.rect):
            target.health -= 10
            target.hit = True
```

- Hàm cập nhật hành động mới của nhân vật và hàm vẽ nhân vật lên màn hình tương ứng với hành động và trạng thái hiện tại của nhân vật

```
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)))
```

4.4 Class server

- Hàm **handle-client(conn, addr)** xử lý kết nối từ client:



```
def handle_client(conn, addr):
    print(f"[NEW CONNECTION] {addr} connected.")

    while True:
        try:
            msg = conn.recv(30).decode(FORMAT)
            print(f"[{addr}] {msg}")
            if msg[12:12+len(DISCONNECT_MESSAGE)] == DISCONNECT_MESSAGE:
                clients.remove(conn)
                print(f"[{addr}]----end----")
                break

            for remote_client in clients:
                if remote_client != conn:
                    remote_client.send(msg.encode(FORMAT))

        except:
            clients.remove(conn)
            print(f"[{addr}]----end----")
            break
```

Nhận đầu vào là đối tượng (conn) và địa chỉ của client(addr)

Lắng nghe các sự kiện ở client và gửi các sự kiện, tin nhắn tới các client khác

Lắng nghe sự kiện ngắn kết nối từ client

- Hàm **start()** lắng nghe kết nối từ client và xử lý gọi hàm handle-client().

```
def start():
    server.listen()
    print(f"[LISTENING] Server is listening on {SERVER}")
    while True:
        conn, addr = server.accept()
        print(f"[ACTIVE CONNECTIONS] {threading.active_count() - 1}")
        clients.append(conn)
        thread = threading.Thread(target=handle_client, args=(conn, addr))
        thread.start()

    print("[STARTING] Server is starting...")
    start()
```



Lắng nghe kết nối đến máy chủ

Chấp nhận kết nối từ client và tạo 1 luồng xử lý riêng biệt.

4.5 Class client

Kết nối đến server thông qua id của server và cổng port

```
class Client():
    def __init__(self):
        self.PORT = 5050
        self.FORMAT = 'utf-8'
        self.DISCONNECT_MESSAGE = '!DISCONNECT'
        self.SERVER = "192.168.56.1"
        self.ADDR = (self.SERVER, self.PORT)
        self.clientsocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.clientsocket.connect(self.ADDR)
        self.clientsocket.settimeout(0.002)
        self.connected = True

    def send(self, msg):
        message = msg.encode(self.FORMAT)
        self.clientsocket.send(message)
```

4.6 Class menuControl

Class MenuControl này hỗ trợ việc tạo và hiển thị menu khởi động của trò chơi, cũng như xử lý người chơi lựa chọn giữa chơi online và offline.

init(self): Hàm khởi tạo cho đối tượng MenuControl, thiết lập một số thông tin cơ bản như kích thước màn hình, FPS, trạng thái chạy, font chữ, ảnh nền menu, ảnh biểu tượng online và offline.

draw-menu(self): Hàm vẽ menu trên màn hình bằng cách sử dụng hình ảnh nền của menu.

```
def draw_menu(self):
    scaled_menu = pygame.transform.scale(self.menu_image, (self.SCREEN_W, self.SCREEN_H))
    self.screen.blit(scaled_menu, (0, 0))
```



createMenu(self): Hàm này chứa vòng lặp để vẽ menu và xử lý sự kiện từ người dùng.

```
def createMenu(self):
    while self.run:

        #draw menu
        self.draw_menu()

        milliseconds = pygame.time.get_ticks()
        if milliseconds // 1000 % 2 == 0:
            text_surface = self.text_font.render("Press Enter To Start", True, (255, 255, 255)) # Màu trắng
        else:
            text_surface = self.text_font.render("", True, (255, 255, 255)) # Màu trắng
        self.screen.blit(text_surface, (self.SCREEN_W - self.SCREEN_W * 0.8, self.SCREEN_H - self.SCREEN_H * 0.15))

        #event handler
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                break
            elif event.type == pygame.KEYDOWN:
                if event.key == pygame.K_RETURN:
                    self.run = False
                elif event.key == pygame.K_LEFT and self.status > 1:
                    self.status = self.status - 1
                elif event.key == pygame.K_RIGHT and self.status < 2:
                    self.status = self.status + 1
```

- Trong vòng lặp, menu được vẽ trên màn hình và thông báo "Press Enter To Start".

- Bắt sự kiện người dùng chọn chơi chế độ nào online hoặc offline và nút Enter

4.7 Class menuOnline

Tạo ra menu ở chế độ chơi online khi người dùng chọn

init(self): Phương thức khởi tạo của class MenuOnline thiết lập các thông số ban đầu cho menu online, bao gồm kích thước màn hình, lựa chọn nhân vật và bản đồ, hình ảnh, âm thanh và các thông số khác.



```
def __init__(self):
    self.SCREEN_W = 1000
    self.SCREEN_H = 600
    self.ft = "a"
    self.ft2 = "a"
    self.ready1 = "Choosing"
    self.ready2 = "Choosing"
    self.map = 1
    self.FPS = 60
    self.run = True
    self.intro_count = 3
    self.text_font = pygame.font.Font("assets/fonts/emulogic.ttf", 30)
    self.screen = pygame.display.set_mode((self.SCREEN_W, self.SCREEN_H))
    self.menu_image = pygame.image.load("assets/images/background/menuOffline.jpg").convert_alpha()
    self.map1_image = pygame.image.load("assets/images/background/desert.jfif").convert_alpha()
    self.map2_image = pygame.image.load("assets/images/background/football.jfif").convert_alpha()
    self.map3_image = pygame.image.load("assets/images/background/hill.jfif").convert_alpha()
    self.vs_image = pygame.image.load("assets/images/icons/vs.png").convert_alpha()
```

draw-menu(self): Hàm vẽ menu trên màn hình, bao gồm nền, hình ảnh người chơi, hình ảnh bản đồ

```
def draw_menu(self):
    scaled_menu = pygame.transform.scale(self.menu_image, (self.SCREEN_W, self.SCREEN_H))
    self.screen.blit(scaled_menu, (0, 0))
    scaled_vs = pygame.transform.scale(self.vs_image, (self.SCREEN_W/4, self.SCREEN_H/4))
    self.screen.blit(scaled_vs, (350, 250))
    if self.map == 1:
        scaled_map1 = pygame.transform.scale(self.map1_image, (self.SCREEN_W/4, self.SCREEN_H/4))
        self.screen.blit(scaled_map1, (370, 50))
    elif self.map == 2:
        scaled_map2 = pygame.transform.scale(self.map2_image, (self.SCREEN_W/4, self.SCREEN_H/4))
        self.screen.blit(scaled_map2, (370, 50))
    elif self.map == 3:
        scaled_map3 = pygame.transform.scale(self.map3_image, (self.SCREEN_W/4, self.SCREEN_H/4))
        self.screen.blit(scaled_map3, (370, 50))
```

draw-fighter(self, ft, ft2): Hàm vẽ các nhân vật (fighters) dựa trên lựa chọn của người chơi.



```
def draw_fighter(self, ft, ft2):
    if ft == "a":
        self.fighter_1.update()
        self.fighter_1.draw(self.screen)
    elif ft == "b":
        self.fighter_2.update()
        self.fighter_2.draw(self.screen)
    if ft2 == "a":
        self.fighter_a.update()
        self.fighter_a.draw(self.screen)
    elif ft2 == "b":
        self.fighter_b.update()
        self.fighter_b.draw(self.screen)
```

createMenu(self, client): Hàm này chứa vòng lặp để tạo menu cho chế độ online, và xử lý các sự kiện từ người chơi.



```
def createMenu(self, client):
    while self.run:

        #draw menu
        self.draw_menu()

        milliseconds = pygame.time.get_ticks()
        if milliseconds // 1000 % 2 == 0:
            text_surface = self.text_font.render("Press Enter To Ready", True, (255, 255, 255)) # Màu trắng
        else:
            text_surface = self.text_font.render("", True, (255, 255, 255)) # Màu trắng
        self.screen.blit(text_surface, (self.SCREEN_W - self.SCREEN_W * 0.8, self.SCREEN_H - self.SCREEN_H * 0.1))

        if milliseconds // 100 % 2 == 0:
            text_ready = self.text_font.render(self.ready1, True, (255, 255, 255)) # Màu trắng
        elif self.ready1 == "Choosing":
            text_ready = self.text_font.render("", True, (255, 255, 255)) # Màu trắng
        self.screen.blit(text_ready, (self.SCREEN_W - self.SCREEN_W * 0.9, self.SCREEN_H - self.SCREEN_H * 0.2))

        if milliseconds // 100 % 2 == 0:
            text_ready2 = self.text_font.render(self.ready2, True, (255, 255, 255)) # Màu trắng
        elif self.ready2 == "Choosing":
            text_ready2 = self.text_font.render("", True, (255, 255, 255)) # Màu trắng
        self.screen.blit(text_ready2, (self.SCREEN_W - self.SCREEN_W * 0.3, self.SCREEN_H - self.SCREEN_H * 0.2))
```

4.8 Class menuOffline

Tạo ra menu ở chế độ chơi offline khi người dùng chọn

init(self): Phương thức khởi tạo của class MenuOnline thiết lập các thông số ban đầu cho menu online, bao gồm kích thước màn hình, lựa chọn nhân vật và bản đồ, hình ảnh, âm thanh và các thông số khác.

```
def __init__(self):
    self.SCREEN_W = 1000
    self.SCREEN_H = 600
    self.ft = "a"
    self.ft2 = "a"
    self.ready1 = "Choosing"
    self.ready2 = "Choosing"
    self.map = 1
    self.FPS = 60
    self.run = True
    self.intro_count = 3
    self.text_font = pygame.font.Font("assets/fonts/emuLogic.ttf", 30)
    self.screen = pygame.display.set_mode((self.SCREEN_W, self.SCREEN_H))
    self.menu_image = pygame.image.load("assets/images/background/menuOffline.jpg").convert_alpha()
    self.map1_image = pygame.image.load("assets/images/background/desert.jfif").convert_alpha()
    self.map2_image = pygame.image.load("assets/images/background/football.jfif").convert_alpha()
    self.map3_image = pygame.image.load("assets/images/background/hill.jfif").convert_alpha()
    self.vs_image = pygame.image.load("assets/images/icons/vs.png").convert_alpha()
```

draw-menu(self): Hàm vẽ menu trên màn hình, bao gồm nền, hình ảnh



người chơi, hình ảnh bản đồ

```
def draw_menu(self):
    scaled_menu = pygame.transform.scale(self.menu_image, (self.SCREEN_W, self.SCREEN_H))
    self.screen.blit(scaled_menu, (0, 0))
    scaled_vs = pygame.transform.scale(self.vs_image, (self.SCREEN_W/4, self.SCREEN_H/4))
    self.screen.blit(scaled_vs, (350, 250))
    if self.map == 1:
        scaled_map1 = pygame.transform.scale(self.map1_image, (self.SCREEN_W/4, self.SCREEN_H/4))
        self.screen.blit(scaled_map1, (370, 50))
    elif self.map == 2:
        scaled_map2 = pygame.transform.scale(self.map2_image, (self.SCREEN_W/4, self.SCREEN_H/4))
        self.screen.blit(scaled_map2, (370, 50))
    elif self.map == 3:
        scaled_map3 = pygame.transform.scale(self.map3_image, (self.SCREEN_W/4, self.SCREEN_H/4))
        self.screen.blit(scaled_map3, (370, 50))
```

draw-fighter(self, ft, ft2): Hàm vẽ các nhân vật (fighters) dựa trên lựa chọn của người chơi.



```
def draw_fighter(self, ft, ft2):
    if ft == 1:
        self.fighter_1.update()
        self.fighter_1.draw(self.screen)
    elif ft == 2:
        self.fighter_2.update()
        self.fighter_2.draw(self.screen)
    elif ft == 3:
        self.fighter_3.update()
        self.fighter_3.draw(self.screen)
    if ft2 == 1:
        self.fighter_a.update()
        self.fighter_a.draw(self.screen)
    elif ft2 == 2:
        self.fighter_b.update()
        self.fighter_b.draw(self.screen)
    elif ft2 == 3:
        self.fighter_c.update()
        self.fighter_c.draw(self.screen)
```

createMenu(self, client): Hàm này chứa vòng lặp để tạo menu cho chế độ online, và xử lý các sự kiện từ người chơi.



```
def createMenu(self, client):
    while self.run:

        #draw menu
        self.draw_menu()

        milliseconds = pygame.time.get_ticks()
        if milliseconds // 1000 % 2 == 0:
            text_surface = self.text_font.render("Press Enter To Ready", True, (255, 255, 255)) # Màu trắng
        else:
            text_surface = self.text_font.render("", True, (255, 255, 255)) # Màu trắng
        self.screen.blit(text_surface, (self.SCREEN_W - self.SCREEN_W * 0.8, self.SCREEN_H - self.SCREEN_H * 0.1))

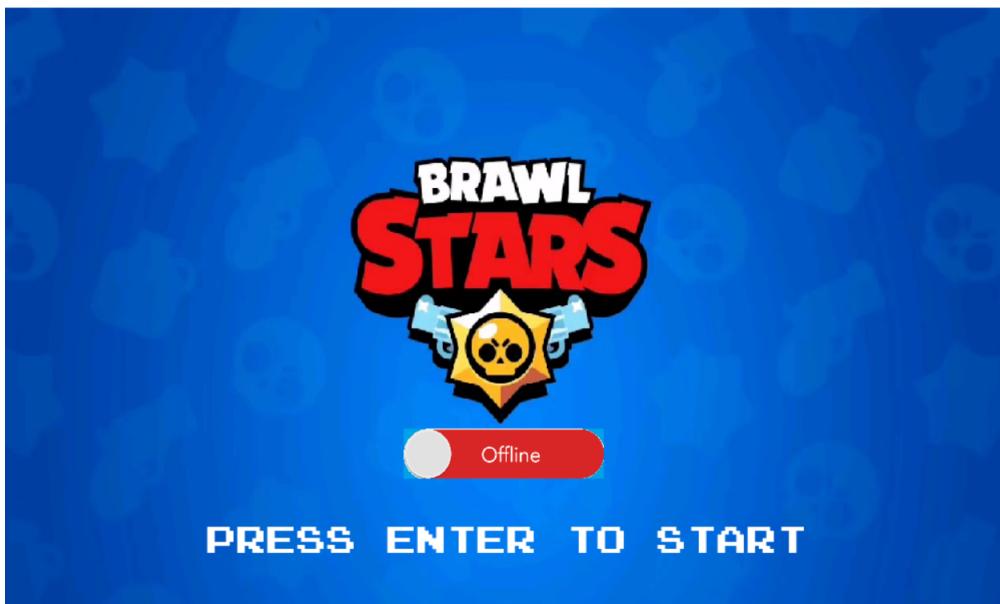
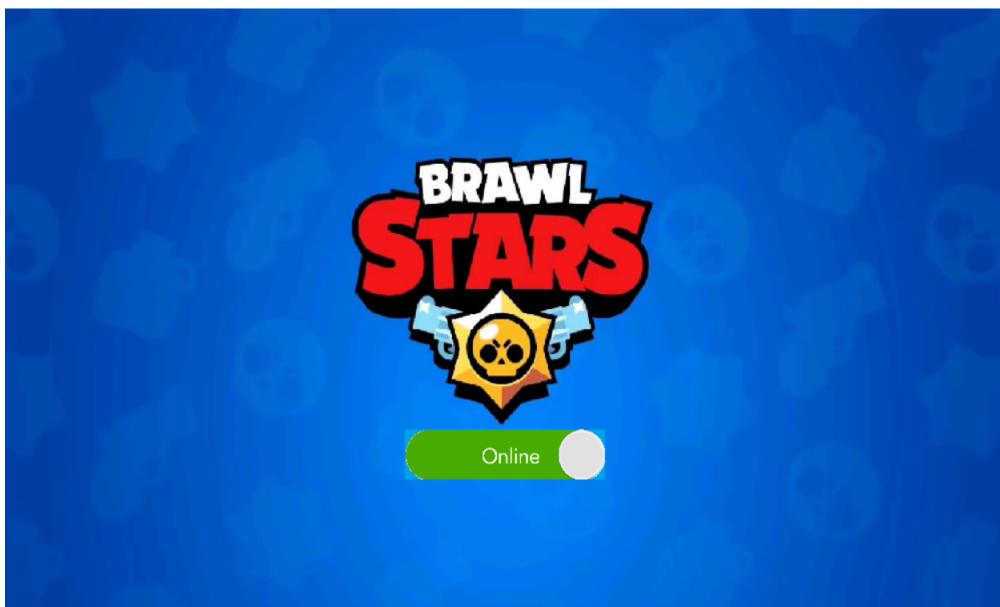
        if milliseconds // 100 % 2 == 0:
            text_ready = self.text_font.render(self.ready1, True, (255, 255, 255)) # Màu trắng
        elif self.ready1 == "Choosing":
            text_ready = self.text_font.render("", True, (255, 255, 255)) # Màu trắng
        self.screen.blit(text_ready, (self.SCREEN_W - self.SCREEN_W * 0.9, self.SCREEN_H - self.SCREEN_H * 0.2))

        if milliseconds // 100 % 2 == 0:
            text_ready2 = self.text_font.render(self.ready2, True, (255, 255, 255)) # Màu trắng
        elif self.ready2 == "Choosing":
            text_ready2 = self.text_font.render("", True, (255, 255, 255)) # Màu trắng
        self.screen.blit(text_ready2, (self.SCREEN_W - self.SCREEN_W * 0.3, self.SCREEN_H - self.SCREEN_H * 0.2))
```

5 Hiện thực

5.1 Các tính năng được thiết kế và hình ảnh

Bước 1: Chọn chế độ chơi Online hoặc Offline: Sử dụng phím mũi tên trái hoặc phải để chọn chế độ chơi game



Bước 2: Nếu chọn chế độ chơi Online phải chạy khởi động server trước

khi chơi. Nếu chọn chế độ Offline thì qua bước 3

```
Project: brawler_tut-main2 [brawler_tut-main] E:\Python\brawler_tut-main\Server.py
12     server.bind(ADDR)
13
14     def handle_client(conn, addr):
15         print(f"[NEW CONNECTION] {addr} connected.")
16
17         while True:
18             try:
19                 msg = conn.recv(30).decode(FORMAT)
20                 print(f"[{addr}] {msg}")
21                 if msg[12:12+len(DISCONNECT_MESSAGE)] == DISCONNECT_MESSAGE:
22                     clients.remove(conn)
23                     print(f"[{addr}] -----end-----")
24                     break
25
26
Run: Server
E:\Python\brawler_tut-main2\venv\Scripts\python.exe E:/Python/brawler_tut-main2/brawler_tut-main/Server.py
[STARTING] Server is starting...
[LISTENING] Server is listening on 192.168.56.1
```

Bước 3: Màn hình chọn nhân vật và background:

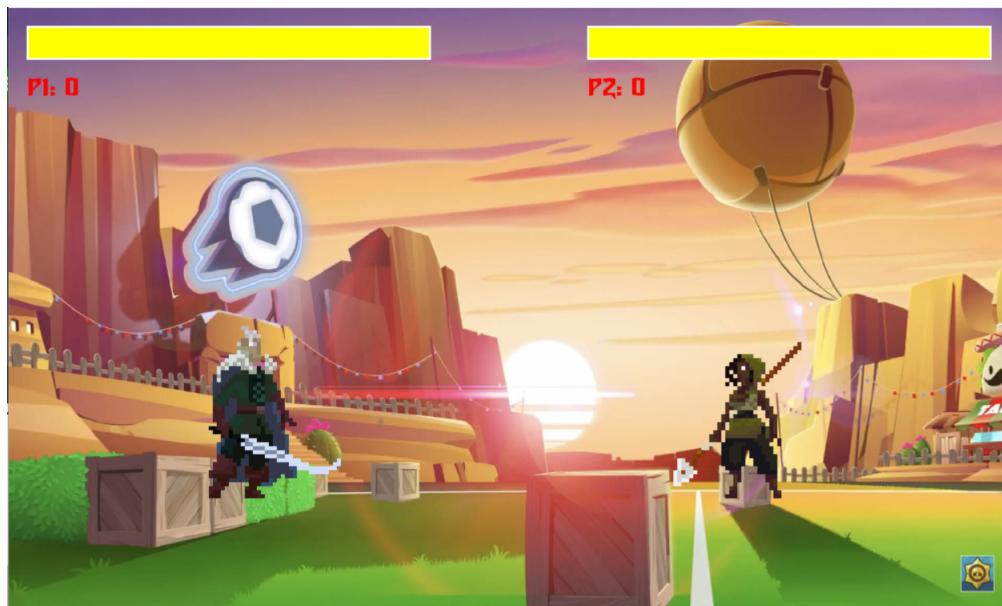
Chọn background: Sử dụng phím cách "space" để chọn màn hình



Chọn nhân vật: - Người chơi 1: Sử dụng phím A hoặc D để di chuyển và chọn nhân vật 1
- Người chơi 2: Sử dụng phím mũi tên trái phải để chọn nhân vật 2



Bước 5: Sau khi chọn xong nhân vật và background ấn Enter để bắt đầu trò chơi





6 Cách thức cài đặt ứng dụng, môi trường chạy ứng dụng,

6.1 Cài đặt Python trên Widown hoặc Linux

Bước 1: Vào trang chủ python chọn phiên bản cần và tải xuống trình cài đặt

- <https://www.python.org/downloads/>

The screenshot shows the Python official website's main navigation bar with links for Python, PSF, Docs, PyPI, Jobs, and Community. Below the navigation is the Python logo and a search bar with a 'GO' button. A 'Donate' button is also visible. The main content area has tabs for About, Downloads, Documentation, Community, Success Stories, News, and Events. The 'Downloads' tab is currently selected. At the bottom of the page, there is a breadcrumb navigation: Python >> Downloads >> Windows.

Python Releases for Windows

- [Latest Python 3 Release - Python 3.11.2](#)

Stable Releases

- [Python 3.10.10 - Feb. 8, 2023](#)

Note that Python 3.10.10 cannot be used on Windows 7 or earlier.

- Download [Windows embeddable package \(32-bit\)](#)
- Download [Windows embeddable package \(64-bit\)](#)
- Download [Windows help file](#)
- Download [Windows installer \(32-bit\)](#)
- Download [Windows installer \(64-bit\)](#)

Pre-releases

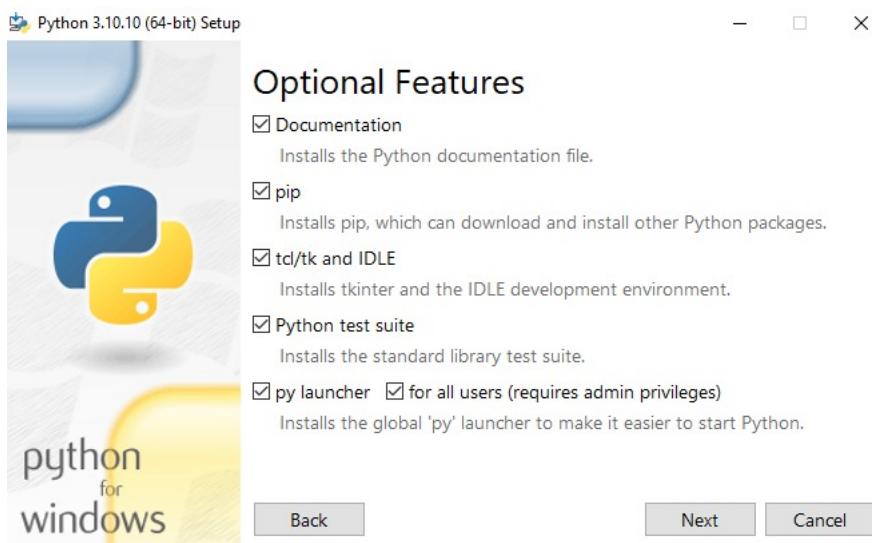
- [Python 3.12.0a5 - Feb. 7, 2023](#)

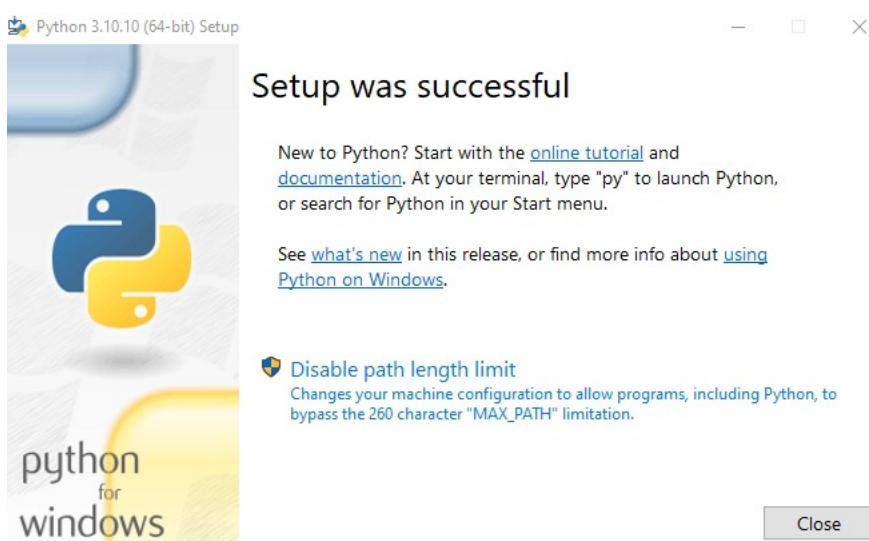
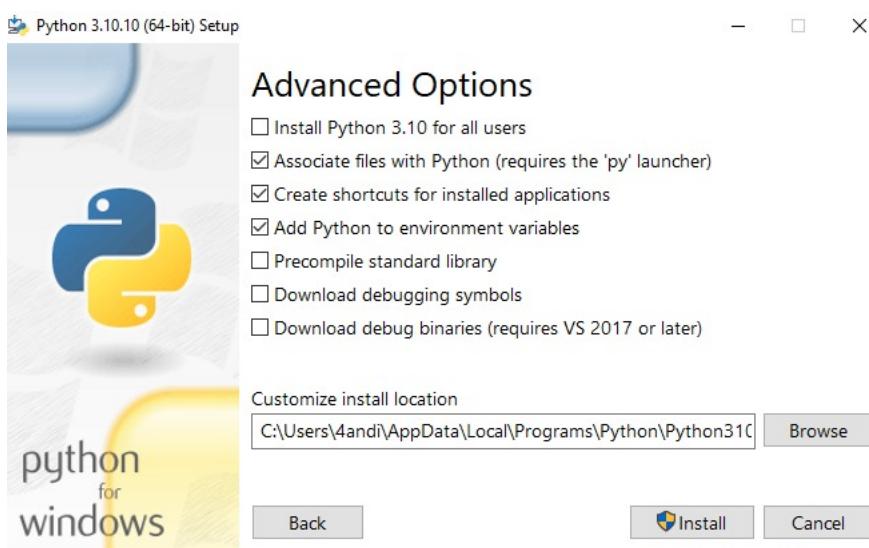
- Download [Windows embeddable package \(32-bit\)](#)
- Download [Windows embeddable package \(64-bit\)](#)
- Download [Windows embeddable package \(ARM64\)](#)
- Download [Windows installer \(32-bit\)](#)
- Download [Windows installer \(64-bit\)](#)
- Download [Windows installer \(ARM64\)](#)

- [Python 3.12.0a4 - Jan. 10, 2023](#)

- Download [Windows embeddable package \(32-bit\)](#)

Bước 2: Chạy trình cài đặt thực thi





Bước 3: Thêm Python vào Biến môi trường

Bước 1: Di tới **Start** và nhập advanced system settings vào thanh tìm kiếm.

Bước 2: Nhập vào **View advanced system settings**.

Bước 3: In the System Properties dialog box, nhập vào **Advanced** và sau đó nhập vào **Environment Variables**.

Bước 4: Tùy thuộc vào cài đặt của bạn:



- Nếu bạn đã chọn **Install for all users** trong khi cài đặt, hãy chọn **Path from the System Variables** và nhấp **Edit**.

- Nếu bạn không chọn **Settings for all users** trong khi cài đặt, hãy chọn **Path from the System Variables** và nhấp **Edit**.

Bước 5: Bấm vào **New** và nhập đường dẫn thư mục Python, sau đó bấm **OK** cho đến khi tất cả các hộp thoại được đóng lại.

Bước 4: Kiểm tra Python đã cài đặt chưa

Đi tới **Start** và nhập **cmd** vào thanh tìm kiếm. Bấm vào **cmd**.

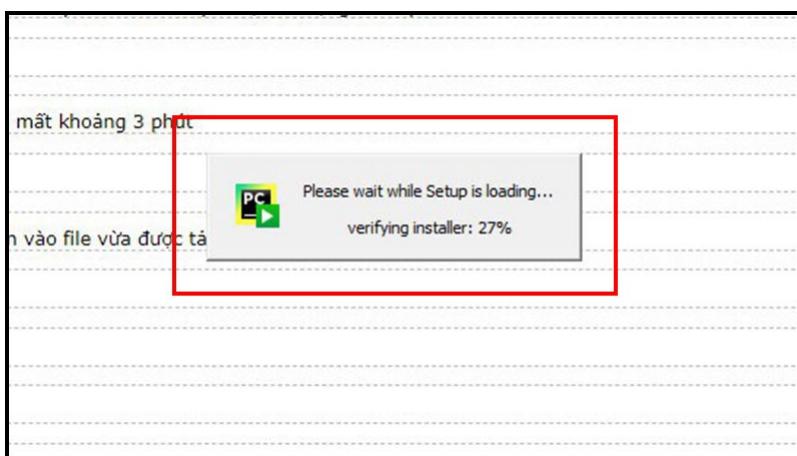
Bạn cũng có thể kiểm tra phiên bản Python bằng cách mở ứng dụng IDLE. Vào **Start** và nhập **python** vào thanh tìm kiếm rồi nhấn vào ứng dụng IDLE, ví dụ IDLE (Python 3.10 64-bit).

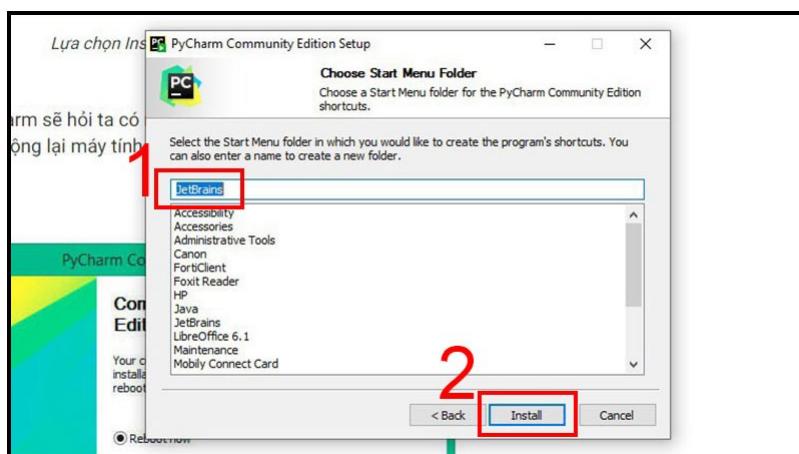
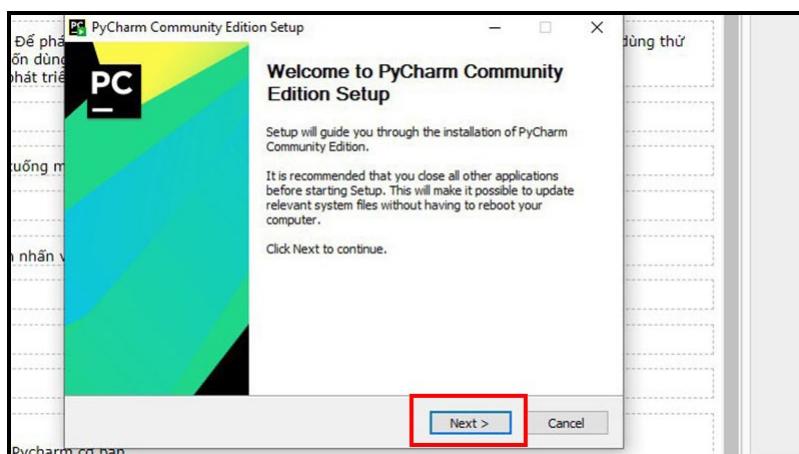
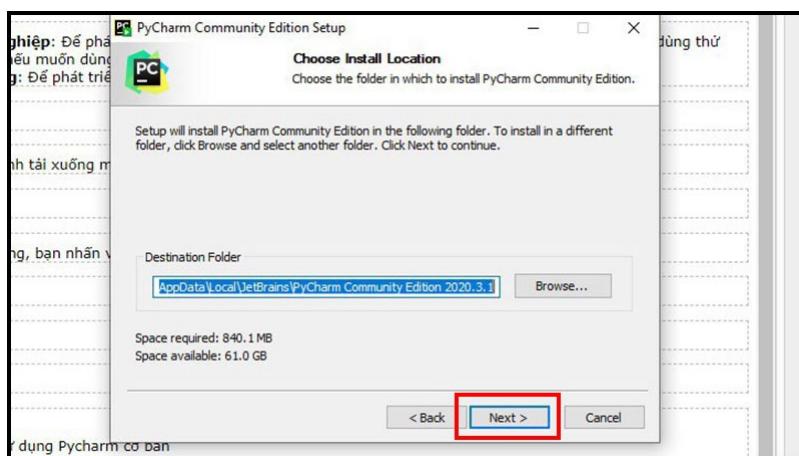
The screenshot shows the IDLE Shell 3.10.10 window. The title bar says "IDLE Shell 3.10.10". The menu bar includes File, Edit, Shell, Debug, Options, Window, Help. The main window displays the Python 3.10.10 welcome message: "Python 3.10.10 (tags/v3.10.10:aad5f6a, Feb 7 2023, 17:20:36) [MSC v.1929 64 bit (AMD64)] on win32 Type "help", "copyright", "credits" or "license()" for more information." Below the message, there is a red text ">>>" indicating the Python prompt.

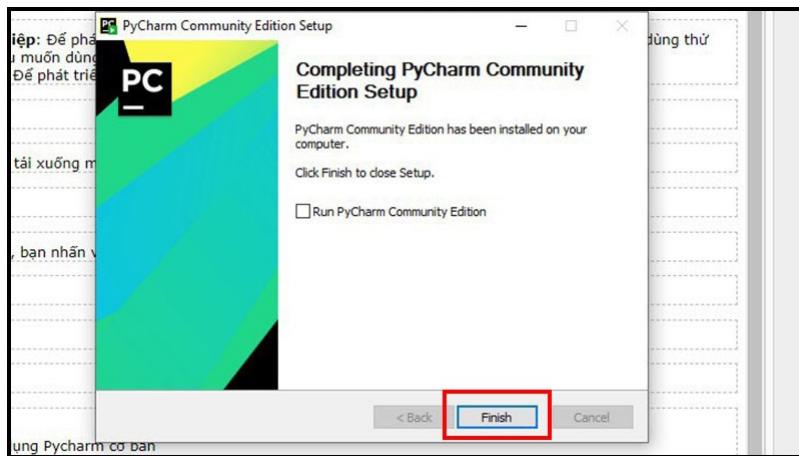
6.2 Cài đặt Pycharm

Làm theo các bước như hình:

- <https://www.jetbrains.com/pycharm/>



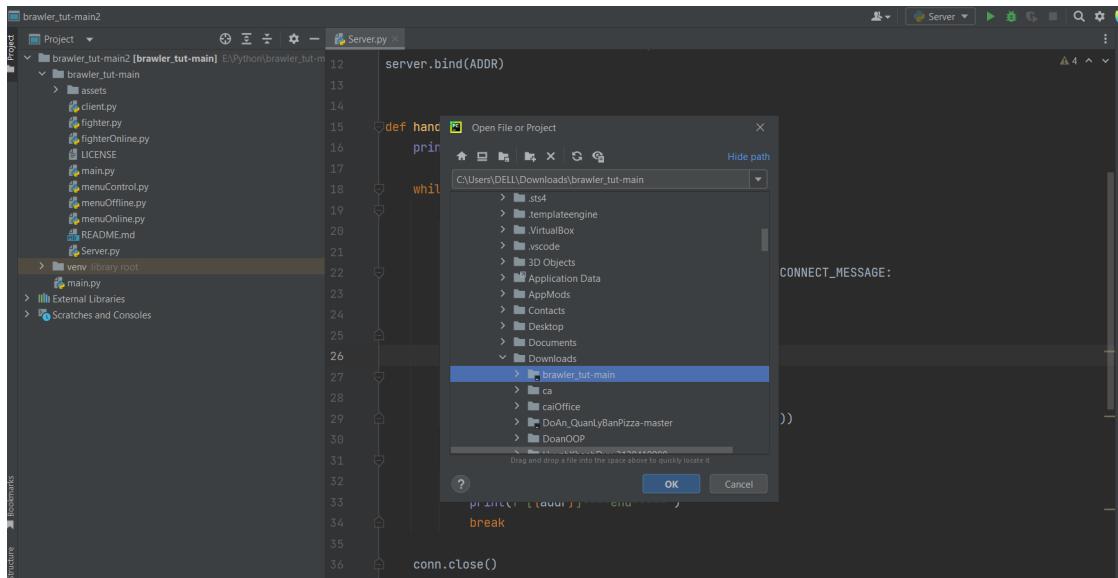




6.3 Open project game vào Pycharm

Bước 1: Tải thư viện pygame vào terminal gõ lệnh **pip install pygame**

Bước 2: Mở project trong pycharm và import các thư viện vào



Bước 3: Run file main và bắt đầu chơi game



```
from client import Client
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("Brawlers Online")

#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.
```

The screenshot shows the PyCharm IDE interface. On the left is the Project tool window displaying the 'brawler_tut-main2' project structure. The right side shows two tabs: 'Server.py' and 'main.py'. The 'main.py' tab is active, showing Python code for initializing pygame, creating a window, setting a framerate, defining colors, and initializing game variables. A context menu is open over the code, with the 'Run main (1)' option highlighted. Other options in the menu include Paste, Copy / Paste Special, Column Selection Mode, Find Usages, Refactor, Folding, Go To, Generate, Run 'main (1)', Debug 'main (1)', Modify Run Configuration..., Open In, Local History, Execute Line in Python Console, Run File in Python Console, and Compare with Clipboard.



7 Nhiệm vụ, vai trò của từng thành viên trong nhóm.

Trần Minh Quân:

- Làm menu, chế độ chơi game online, offline
- Xử lý các sự kiện từ người dùng khi chọn các chế độ như giao diện, nhân vật
 - Tìm kiếm hình ảnh giao diện, tìm kiếm thêm nhân vật

Lê Hồng Việt:

- Làm socket cho game
- Viết báo cáo



Tài liệu

- [1] Socket “link: <https://www.youtube.com/watch?v=erKZiVBV9Sot=70s>”
- [2] Source “link: <https://www.youtube.com/watch?v=s5bd9KMSSW4t=500s>”