TRƯỜNG ĐẠI HỌC SÀI GÒN
KHOA CÔNG NGHỆ THÔNG TIN

PHÁT TRIỂN PHẦN MỀM MÃ NGUỒN MỞ

Phát triển game cho 2 người chơi

GVHD: Từ Lãng Phiêu
Nhóm: 15
SV: Đỗ Thị Như Quỳnh - 3121410414
Lê Minh Thuận - 3121410481
Nguyễn Thị Bích Trâm - 3121410517
Gmail:leminhthuannn123@gmail.com

TP. HỒ CHÍ MINH, THÁNG 5/2024

Mục lục

1 Phần giới thiệu **2**
- 1.1 Giới thiệu về game Fatal Siege . . . 2
- 1.2 Các công nghệ sử dụng . . . 3
  - 1.2.1 Pygame . . . 3
  - 1.2.2 Socket . . . 3
  - 1.2.3 Threading . . . 3
  - 1.2.4 Pickle . . . 3

2 Chức năng **5**
- 2.1 Hướng dẫn chơi game Fatal Siege . . . 5
  - 2.1.1 Vào trò chơi . . . 5
  - 2.1.2 Cấu trúc của các file trong hệ thống . . . 6

3 Kết luận **17**
- 3.1 Ưu điểm . . . 17
  - 3.1.1 Thể loại kết hợp: . . . 17
  - 3.1.2 Đa người chơi: . . . 17
  - 3.1.3 Đơn giản nhưng gây nghiện: . . . 17
- 3.2 Nhược điểm . . . 17
  - 3.2.1 Giới hạn về nội dung: . . . 17
  - 3.2.2 Chỉ hỗ trợ 2 người chơi: . . . 17
  - 3.2.3 Hạn chế về tính chiến thuật sâu: . . . 17
- 3.3 Hướng phát triển . . . 18
  - 3.3.1 Tạo tài khoản: . . . 18
  - 3.3.2 Options Menu: . . . 18
  - 3.3.3 Tạo và tìm phòng chơi: . . . 18
  - 3.3.4 Thêm kỹ năng chủ động: . . . 18
  - 3.3.5 Hệ thống âm thanh: . . . 18

4 Cài đặt ứng dụng **19**
- 4.1 Yêu cầu cài đặt . . . 19
- 4.2 Các bước cài đặt . . . 19

5 Phân công công việc **20**

1 Phần giới thiệu

1.1 Giới thiệu về game Fatal Siege

Fatal siege là một game 2 người chơi, thuộc thể loại trò chơi phòng thủ (tower defense) kết hợp chiến thuật thời gian thực (real-time strategy), được phát triển bằng ngôn ngữ Python, sử dụng thư viện Pygame.

- Một ván Fatal siege sẽ bắt đầu khi có đủ 2 người chơi kết nối với nhau.
  - Khi đã có đủ 2 người chơi, trò chơi sẽ bắt đầu.
  - Với mỗi người chơi sẽ hiển thị một màn hình gameplay, nơi đây sẽ có những con lính xuất hiện ngẫu nhiên và liên tục tấn công thành trì của người chơi.
  - Người chơi có nhiệm vụ dùng đại bác để tiêu diệt lính, tránh việc lính chạm vào thành.
  - Mỗi người chơi sẽ có 15HP, mỗi khi lính chạm vào thành và thực hiện tấn công, hệ thống sẽ trừ người chơi 1HP.
  - Một ván đấu sẽ kết thúc khi 1 trong 2 người chơi có số HP trở về 0.
- ***"Fatal Siege"*** là một trò chơi kích thích và đầy thú vị. Trò chơi này đưa người chơi vào một cuộc đối đầu gay cấn giữa hai tường thành, nơi mà chiến thắng chỉ dành cho người có chiến thuật, tính toán và phản xạ nhanh nhạy.
- Lối chơi đơn giản nhưng gây nghiện, "Fatal Siege" yêu cầu người chơi phải sử dụng vũ khí và kỹ năng của mình để tiêu diệt lính địch và bảo vệ tường thành của mình. Với đồ họa sắc nét và hiệu ứng âm thanh sống động, người chơi sẽ được trải nghiệm cảm giác của một cuộc chiến thực sự.
- Với các tính năng đa dạng và cơ chế chơi sáng tạo, "Fatal Siege" hứa hẹn mang lại những giờ phút giải trí đầy thú vị cho bạn bè hoặc gia đình. Hãy chuẩn bị chiến lược của bạn, hãy cẩn thận và hãy trở thành người chiến thắng trong cuộc đối đầu này!

1.2 Các công nghệ sử dụng

1.2.1 Pygame

- **Công dụng:** Pygame là một thư viện Python cung cấp các chức năng để phát triển trò chơi và ứng dụng đa phương tiện.
- **Vai trò:**
  - Vẽ đồ họa và hiển thị các đối tượng như cannon, lính, tường, và viên đạn trên màn hình.
  - Xử lý sự kiện người dùng như nhấn nút, di chuyển chuột, để tương tác với trò chơi.
  - Quản lý các phần tử trò chơi như hình ảnh, âm thanh và font chữ.
  - Thực hiện các hoạt động đồ họa như xoay, phóng to và thu nhỏ hình ảnh.

1.2.2 Socket

- **Công dụng:** Socket là một cơ chế trong Python cho phép truyền dữ liệu qua mạng giữa các máy tính.
- **Vai trò:**
  - Socket được sử dụng để thiết lập và quản lý kết nối mạng giữa máy chủ và các máy khách (clients).
  - Socket cho phép trò chơi gửi và nhận dữ liệu trò chơi giữa máy chủ và các máy khách, bao gồm trạng thái trò chơi, hành động của người chơi và các cập nhật khác.

1.2.3 Threading

- **Công dụng:** Sử dụng threading có thể được áp dụng để xử lý các tác vụ đồng thời mà không làm chậm hoặc gián đoạn quá trình chạy của trò chơi chính.
- **Vai trò:**
  - Khi triển khai chức năng tạo và tìm phòng chơi, việc sử dụng threading có thể hữu ích để xử lý các kết nối mạng.
  - Mỗi kết nối từ người chơi có thể được xử lý trong một luồng riêng biệt để đảm bảo rằng việc kết nối và trao đổi dữ liệu không làm chậm hoặc tắt chương trình chính.

1.2.4 Pickle

- **Công dụng:** Pickle là một module trong Python cho phép chuyển đổi các đối tượng Python sang chuỗi byte và ngược lại, quá trình này được gọi là "serialize" và "deserialize".
- **Vai trò:**
  - Pickle được sử dụng để đóng gói (serialize) trạng thái của trò chơi thành một chuỗi byte trước khi gửi qua mạng.
  - Khi dữ liệu đến từ máy chủ hoặc máy khách, Pickle được sử dụng để giải gói (deserialize) dữ liệu thành các đối tượng Python có thể sử dụng được.

– Việc sử dụng Pickle giúp truyền dữ liệu giữa máy chủ và máy khách dễ dàng hơn bằng cách chuyển đổi các đối tượng Python thành dạng có thể truyền qua mạng và ngược lại.

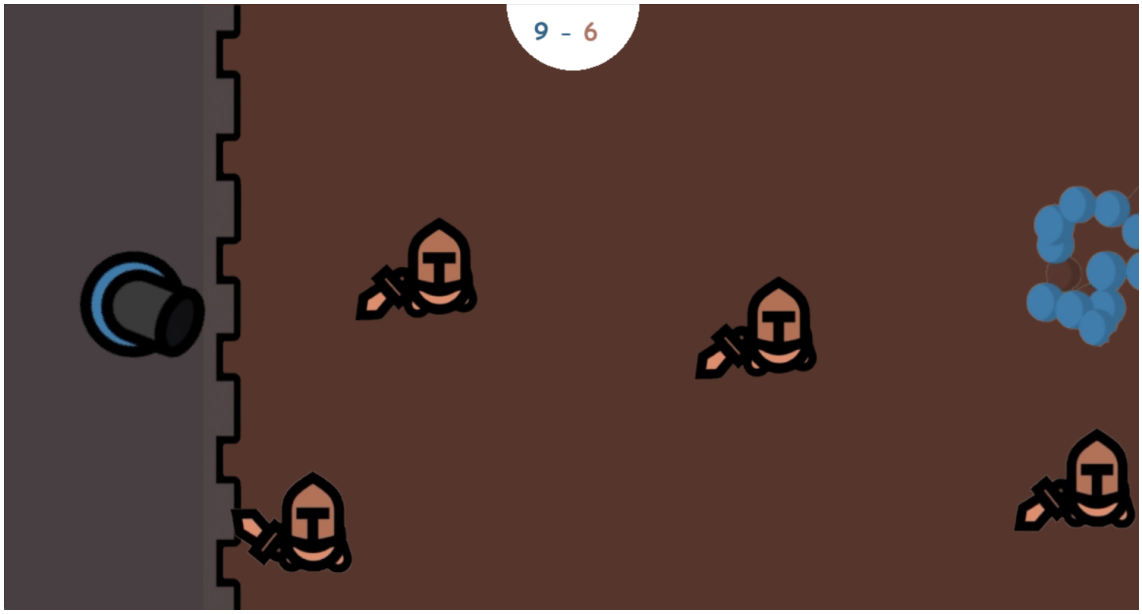
2 Chức năng

2.1 Hướng dẫn chơi game Fatal Siege

2.1.1 Vào trò chơi

Bước 1: Chọn Click to play. Hệ thống hiển thị giao diện chơi game.

Bước 2: Lính sẽ di chuyển từ bên phải màn hình về phía tường thành. Canh góc đại bác về phía lính và ấn phím "Enter" để đại bác bắn đạn tiêu diệt lính. Lưu ý: Ấn càng lâu thì đạn sẽ càng bay xa.



Bước3: Bạn hoặc đối thủ có nhiệm vụ tiêu diệt lính, ai để điểm về 0 thì người đó sẽ thua và người còn lại sẽ thắng.

2.1.2 Cấu trúc của các file trong hệ thống

- **network.py**
  - **__init__**: Khởi tạo thông tin kết nối.

```python
def __init__(self):
    self.client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    self.server = "localhost"
    self.port = 5557
    self.addr = (self.server, self.port)
    self.p = self.connect()
```

- **getP**: Trả về thông tin về player.
- **connect**: Thiết lập kết nối với server.

```python
def connect(self):
    try:
        self.client.connect(self.addr)
        return self.client.recv(2048).decode()
    except Exception as e:
        print(f"Failed to connect to server: {e}")
        return None
```

- **send**: Gửi dữ liệu đến server và nhận dữ liệu từ server.

```python
def send(self, data):
    try:
        self.client.send(str.encode(data))
        reply = self.client.recv(2048 * 2)
        return pickle.loads(reply)
    except socket.error as e:
        print(f"Socket error: {e}")
        return None
```

- **game.py**: Định nghĩa class Game để quản lý trạng thái và logic của trò chơi. Các chức năng chính bao gồm:
  - Quản lý thông tin về trận đấu, bao gồm ID, trạng thái sẵn sàng, và máu của các người chơi.
  - Kiểm tra trạng thái kết nối của trò chơi và xác định người chiến thắng.

```python
class Game:
    def __init__(self, id):
        self.ready = False
        self.id = id
        self.health = [15, 15]

    def get_health(self):
        return self.health

    def check_health(self):
        return any(health == 0 for health in self.health)

    def reset_health(self):
        self.health = [15, 15]

    def current_health(self, player, current_health):
        self.health[player] = current_health

    def connected(self):
        """
        Kiểm tra trạng thái kết nối của trận đấu

        :return: Trạng thái kết nối (True hoặc False)
        """
        return self.ready

    def winner(self):
        if self.health[0] == 0:
            return 1
        elif self.health[1] == 0:
            return 0
```

- **constants.py**: Chứa các hằng số và tài nguyên như màu sắc, font chữ, hình ảnh được sử dụng trong trò chơi.
- **server.py**: Triển khai server để chấp nhận kết nối từ các client và điều phối trò chơi giữa các client. Các chức năng chính bao gồm:
  - Thiết lập server và chờ đợi kết nối từ client.
  - **threaded_client(client_socket, player, gameId)**: Xử lý từng kết nối client trong một luồng riêng biệt.

```
def threaded_client(self, client_socket, player, gameId):
    client_socket.send(str.encode(str(player)))
    while True:
        try:
            data = client_socket.recv(4096).decode()
            if gameId in self.games:
                game = self.games[gameId]

                if not data:
                    break
                else:
                    if data == "reset":
                        game.reset_health()
                    elif data != "get":
                        game.current_health(player, int(data))

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

    print("Lost connection")
    try:
        del self.games[gameId]  # Xóa trận đấu khỏi danh sách khi client
ngắt kết nối
        print("Closing Game", gameId)
    except:
        pass
    self.idCount -= 1
    client_socket.close()
```

– **run()**: Lắng nghe và chấp nhận các kết nối từ client.

```
def run(self):
    while True:
        client_socket, client_address = self.socket.accept()  # Chấp nhận
kết nối từ client mới
        print("Connected to:", client_address)

        self.idCount += 1
        player = 0
        gameId = (self.idCount - 1) // 2
        if self.idCount % 2 == 1:  # Tạo trận đấu mới khi có đủ 2 client
            self.games[gameId] = Game(gameId)
            print("Creating a new game...")
        else:
            self.games[gameId].ready = True  # Đánh dấu trận đấu đã sẵn
sàng
            player = 1

        start_new_thread(self.threaded_client, (client_socket, player,
gameId))  # Bắt đầu một luồng xử lý client mới
```

- **client.py**: Triển khai client để tham gia vào trò chơi, hiển thị giao diện và tương tác với người chơi. Các chức năng chính bao gồm:

- Hiển thị giao diện trò chơi và xử lý tương tác người dùng.
- **Class Gameplay** quản lý vòng lặp trò chơi, sinh lính, di chuyển lính, hiển thị nền, và xử lý logic trò chơi.
  * **__init__(self)** : Khởi tạo cửa sổ trò chơi với kích thước và màu sắc được định nghĩa sẵn. Khởi tạo danh sách lính (**soldiers**), thời gian sinh lính cuối cùng (**last_soldier_time**), khoảng thời gian giữa các lần sinh lính (**spawn_interval**), và các đối tượng tường và pháo.

```
    def __init__(self):
        self.width = c.WIDTH
        self.height = c.HEIGHT
        self.color = c.GAMEPLAY_BG
        self.win = pygame.display.set_mode((self.width, self.height))
        pygame.display.set_caption("Gameplay")
        self.soldiers = []  # Danh sách các lính
        self.last_soldier_time = pygame.time.get_ticks()  # Thời điểm cuối
cùng khi tạo ra lính mới
        self.spawn_interval = 3500  # Khoảng thời gian giữa các lần xuất
hiện lính (ms)
        self.wall = Wall()
        self.cannon = Cannon()
        self.player_health = 15  # Máu của người chơi
```

- **spawn_soldiers(self)** : Sinh ra lính mới dựa trên khoảng thời gian đã trôi qua kể từ lần sinh cuối cùng. Điều chỉnh khoảng thời gian giữa các lần sinh lính và tốc độ của lính theo thời gian.

```
    def spawn_soldiers(self):
        current_time = pygame.time.get_ticks()
        if current_time - self.last_soldier_time >= self.spawn_interval:
            # Tạo ra lính mới
            y = random.randint(150, c.HEIGHT - 150)  # Chọn vị trí y ngẫu
nhiên
            soldier = Soldier(y)
            self.soldiers.append(soldier)
            self.last_soldier_time = current_time
            if self.spawn_interval >= 1500:
                self.spawn_interval -= 180
            if self.spawn_interval <= 2000:
                soldier.speed = 2.5
            elif self.spawn_interval <= 1500:
                soldier.speed = 3
```

- **move_soldiers(self, network)** : Di chuyển lính và cập nhật hoạt ảnh của chúng. Kiểm tra va chạm giữa lính và đạn, và giữa lính và tường. Giảm máu của tường khi lính tấn công tường và gửi cập nhật về máu của tường tới server.

```
    def move_soldiers(self, network):
        current_time = pygame.time.get_ticks()
        for soldier in self.soldiers:
            soldier.draw_and_move(self.win)
            soldier.update_animation(current_time, soldier.is_attack)
            if not soldier.is_alive:
                self.soldiers.remove(soldier)
            for bullet in self.cannon.bullets:
                if bullet.is_explosion:  # Kiểm tra chỉ với các viên đạn đã
nổ
                    soldier.check_collision(bullet)

            # Kiểm tra va chạm giữa lính và tường
            if soldier.rect.colliderect(self.wall.rect_edge):
                # Nếu lính va chạm vào tường lâu hơn 1 giây
                if current_time - soldier.last_wall_hit_time >= 1000:
                    soldier.is_attack = True
                    soldier.last_wall_hit_time = current_time
                    # Giảm máu của người chơi
                    if self.player_health > 0:
                        self.player_health = self.player_health - 1
                        print(self.player_health)
                        network.send(str(self.player_health))
```

  - **display_bg(self)** : Hiển thị nền của trò chơi. Hiển thị trạng thái của tường bao gồm máu của tường.
  - **run(self, network)** : Vòng lặp chính của trò chơi, xử lý sự kiện, sinh lính, hiển thị nền, di chuyển lính và pháo, và gửi thông tin máu của tường tới server.

- **Class Soldier** quản lý thông tin và hành vi của lính.
  - **__init__(self,y)** :Khởi tạo lính với tọa độ y ngẫu nhiên, máu, tốc độ di chuyển, trạng thái sống, và hoạt ảnh.

```
    def __init__(self, y):
        self.speed = 2
        self.is_alive = True
        self.last_wall_hit_time = pygame.time.get_ticks()  # Thời điểm cuối
cùng khi lính va chạm vào tường
        self.images = c.images_solider_red
        self.image_index = 0  # Chỉ số của hình ảnh trong mảng images
        self.image_timer = 0  # Biến đếm thời gian để cập nhật hình ảnh
        self.x = c.WIDTH - self.images[0].get_width()
        self.y = y
        self.images = c.images_solider_red
        self.rect = self.images[self.image_index].get_rect(topleft=(self.x,
self.y))
        self.is_attack = False
```

  - **update_animation(self, current_time, is_attack)** : Cập nhật hoạt ảnh của lính theo thời gian và trạng thái tấn công.

```
    def update_animation(self, current_time, is_attack):
        if not is_attack:
            if current_time - self.image_timer > 200:  # Thay đổi ảnh sau
môi 200ms
                self.image_timer = current_time
                self.image_index = 0 if self.image_index == 1 else 1
        else:
            if current_time - self.image_timer > 500:
                self.image_timer = current_time
                self.image_index = 1 if self.image_index == 2 else 2


while True:
    Gameplay().main()
```

  - **draw_and_move(self, win)** : Vẽ lính lên cửa sổ và di chuyển lính. Kiểm tra trạng thái sống của lính và xóa lính khi máu về 0.

```
   def draw_and_move(self, win):

        if self.x >= 240:
            self.x -= self.speed
        color = c.PLAYER_2_COLOR

        if self.is_alive:
            win.blit(self.images[self.image_index], (self.x, self.y))
            self.rect =
self.images[self.image_index].get_rect(topleft=(self.x, self.y))
```

  - **check_collision(self, bullet)** : Kiểm tra va chạm giữa lính và đạn. Giảm máu của lính khi va chạm với đạn và ẩn đạn sau khi va chạm.

```
    def check_collision(self, bullet):
        if self.rect.colliderect(bullet.rect_explosion):
            self.is_alive = False  # Đặt trạng thái của lính thành không
còn sống
```

- **Class Wall** đại diện cho tường phòng thủ.
  - **__init__(self)** : Khởi tạo tường với vị trí và kích thước xác định, cùng với máu ban đầu.
- **Class Cannon** quản lý pháo và xử lý bắn đạn.
  - **__init__(self)** : Khởi tạo pháo với vị trí và góc độ ban đầu. Khởi tạo danh sách đạn được bắn ra từ pháo.
  - **handle_events(self)** : Xử lý sự kiện điều khiển pháo từ bàn phím, bao gồm xoay pháo và bắn đạn.
  - **draw_and_move(self, win)** : Vẽ pháo lên cửa sổ và di chuyển đạn.

```
def draw_and_move(self, win):

    angle = self.angle
    rotated_cannon = pygame.transform.rotate(self.cannon_image, angle)

    # Tính toán vị trí mới của hình chữ nhật sau khi xoay
    rotated_rect = rotated_cannon.get_rect(center=(self.rect_width / 4,
self.rect_height / 2))

    win.blit(rotated_cannon, (100 - rotated_rect.width / 2 + 45, c.HEIGHT /
2 - rotated_rect.height / 2))

    # Nếu đang bắn và đã trôi qua ít nhất 1 giây từ lần bắn đạn trước đó
    if self.shooting:
        current_time = pygame.time.get_ticks()
        time_since_last_shot = current_time - self.last_shot_time
        if time_since_last_shot <= 1000:
            self.aiming = False
        else:
            self.aiming = True

        if time_since_last_shot >= 1000:
            self.shoot()
            self.last_shot_time = current_time
            self.shooting = False
            self.aiming = False
            self.became_idle_time = current_time

    if not self.aiming and not self.shooting and pygame.time.get_ticks() -
self.became_idle_time >= 300:
        # Tăng góc quay
        if self.clockwise:
            self.angle += 2.5
            if self.angle >= 250:
                self.clockwise = False
        else:
            self.angle -= 2.5
            if self.angle <= 110:
                self.clockwise = True

    for bullet in self.bullets:
        bullet.draw(win)
        bullet.move(win)
        if bullet.explosion_time - bullet.explosion_timer <= 0:
            self.bullets.remove(bullet)
```

– **shoot(self)** : Tạo và thêm đạn vào danh sách đạn mỗi khi pháo bắn.

```
def shoot(self):
    # Tính toán thời gian đã giữ nút Enter
    hold_time = self.start_time_keyup - self.start_time_keydown

    # Tính toán khoảng cách di chuyển của đạn dựa trên thời gian đã giữ nút
Enter
    max_distance = 1000  # Khoảng cách tối đa (px) khi giữ nút Enter 1s
    min_distance = 10  # Khoảng cách tối thiểu (px) khi giữ nút Enter ngay
lập tức
    hold_time = min(hold_time, 1000)  # Giới hạn thời gian giữ nút Enter là
1s
    distance = min_distance + (max_distance - min_distance) * (hold_time /
1000)

    # Tính toán tọa độ xuất phát của viên đạn từ chính giữa cạnh chiều cao
bên phải của hình chữ nhật
    angle = self.angle
    bullet_x = 100 - math.cos(math.radians(angle)) * self.rect_width / 2
    bullet_y = c.HEIGHT / 2 + math.sin(math.radians(angle)) *
self.rect_width / 2

    # Tạo một viên đạn mới với khoảng cách di chuyển tính được
    bullet = Bullet(bullet_x, bullet_y, angle, distance)
    self.bullets.append(bullet)  # Thêm viên đạn vào danh sách
```

- **Class Bullet** quản lý thông tin và hành vi của đạn.
  - **__init__(self, x, y, angle)** : Khởi tạo đạn với vị trí, góc bắn, tốc độ di chuyển, bán kính, và trạng thái hiển thị.
  - **draw_and_move(self, win)** : Vẽ đạn lên cửa sổ và di chuyển đạn. Kiểm tra va chạm với rìa cửa sổ để ẩn đạn khi ra khỏi phạm vi.

```
def draw(self, win):
    color = c.PLAYER_2_COLOR
    if self.is_draw:
        pygame.draw.circle(win, color, (int(self.x), int(self.y)), 40)  #
Vẽ viên đạn là một hình tròn

    if self.is_explosion and self.explosion_time - self.explosion_timer >=
0:
        win.blit(self.img_explosion, (int(self.x - 90), int(self.y - 85)))
        self.explosion_timer += 1
```

```
def move(self, win):
    # Kiểm tra nếu viên đạn đạt đến vị trí cuối cùng
    if abs(self.x - self.end_x) < 10 and abs(self.y - self.end_y) < 10:
        self.is_draw = False
        self.is_explosion = True
        self.rect_explosion =
self.img_explosion.get_rect(topleft=(int(self.x - 90), int(self.y - 85)))
    else:
        # Di chuyển viên đạn theo hướng đã được xác định bởi góc
        self.x -= math.cos(math.radians(self.angle)) * 20
        self.y += math.sin(math.radians(self.angle)) * 20
```

  - **explode(self)** : Xử lý hiệu ứng nổ khi đạn va chạm.

- **assets**: Chứa các tài nguyên như hình ảnh nền, hình ảnh nhân vật, âm thanh, và các tài nguyên khác cần thiết cho trò chơi.

3 Kết luận

3.1 Ưu điểm

3.1.1 Thể loại kết hợp:

"Fatal Siege" kết hợp hai thể loại trò chơi phổ biến là tower defense và real-time strategy, tạo ra một trải nghiệm mới lạ và độc đáo cho người chơi. Điều này giúp tạo ra sự hấp dẫn và đa dạng trong lối chơi.

3.1.2 Đa người chơi:

Khả năng chơi với 2 người chơi là một điểm mạnh, tạo ra cơ hội cho sự tương tác và cạnh tranh giữa người chơi. Điều này có thể tạo ra những trải nghiệm vui vẻ và cạnh tranh đầy kịch tính.

3.1.3 Đơn giản nhưng gây nghiện:

Gameplay của "Fatal Siege" được thiết kế đơn giản nhưng đầy sức hấp dẫn, thu hút người chơi bằng tính cạnh tranh và yêu cầu chiến thuật, phản xạ nhanh nhạy trong thao tác.

3.2 Nhược điểm

3.2.1 Giới hạn về nội dung:

"Fatal Siege" gặp vấn đề về độ dài và sự đa dạng trong nội dung. Sau một thời gian, người chơi có thể cảm thấy nhàm chán với sự lặp lại của gameplay.

3.2.2 Chỉ hỗ trợ 2 người chơi:

Mặc dù việc hỗ trợ 2 người chơi tạo ra cơ hội cho trải nghiệm tương tác, nhưng cũng hạn chế khả năng mở rộng và thu hút người chơi đơn.

3.2.3 Hạn chế về tính chiến thuật sâu:

Gameplay của "Fatal Siege" thiếu sự phức tạp và chiến thuật sâu hơn, giới hạn khả năng của người chơi để phát triển chiến lược phức tạp.

3.3 Hướng phát triển

3.3.1 Tạo tài khoản:

Thêm chức năng đăng ký và đăng nhập vào tài khoản người chơi để lưu trữ thông tin cá nhân và điểm số của họ. Điều này giúp tạo ra tính cạnh tranh và động viên người chơi cố gắng cải thiện kỹ năng của mình.

3.3.2 Options Menu:

Tạo một giao diện Options Menu cho phép người chơi lựa chọn các chế độ chơi, bao gồm chơi một người và chơi hai người trên cùng một máy. Điều này tăng tính linh hoạt và thuận tiện cho người chơi.

3.3.3 Tạo và tìm phòng chơi:

Thêm chức năng tạo và tìm phòng chơi trong trò chơi, cho phép người chơi kết nối và chơi với nhau trực tuyến. Điều này mở ra cơ hội cho các trận đấu nhiều người và tạo ra một cộng đồng chơi game đa dạng.

3.3.4 Thêm kỹ năng chủ động:

Bổ sung các kỹ năng chủ động như mưa đạn, bão tên, hay các kỹ năng khác mà người chơi có thể sử dụng để tấn công hoặc phòng thủ. Điều này làm tăng sự phong phú và chiến lược trong gameplay.

3.3.5 Hệ thống âm thanh:

Thêm hệ thống âm thanh vào trò chơi để tạo ra không khí sống động và hấp dẫn hơn cho người chơi. Âm thanh của việc tấn công, phòng thủ và các sự kiện khác trong trò chơi sẽ tạo ra trải nghiệm chân thực và thú vị.

4 Cài đặt ứng dụng

4.1 Yêu cầu cài đặt

- Python 3.6 trở lên.
- Đã cài đặt thư viện Pygame.
- Giao diện màn hình từ 1200 $\times$ 640 trở lên.
- Có kết nối với internet để có thể chơi game.

4.2 Các bước cài đặt

- **Bước 1:** Click vào đây để tải game.
- **Bước 2:** Nhấn vào nút Code và TẢI XUỐNG ZIP.
- **Bước 3:** Giải nén file zip, mở PyCharm và chạy thư mục đã giải nén.

5 Phân công công việc

Họ và tên	MSSV	Công việc
Đỗ Thị Như Quỳnh	3121410414	Class Cannon (client.py), Bullet (client.py) ; Tìm kiếm tài nguyên hình ảnh, làm powerpoint
Lê Minh Thuận	3121410481	network.py, server.py, Class GamePlay (client.py)
Nguyễn Thị Bích Trâm	3121410517	game.py , Class Soldier (client.py), Class Wall (client.py) ; Viết báo cáo Latex.

Tài liệu

[1] Python Online Game Tutorial "**link: https://youtube.com/playlist?list=PLzMcBGfZo4-kR7Rh-7JCVDN8lm3Utumvqsi=9p9ATeuZxg4xL5q1**"