

TRƯỜNG ĐẠI HỌC SÀI GÒN
KHOA CÔNG NGHỆ THÔNG TIN



PHÁT TRIỂN PHẦN MỀM MÃ NGUỒN MỞ

Đề án phát triển ứng dụng game bắn máy bay
dùng được cho 2 người chơi

GVHD: Từ Lăng Phiêu
SV: Trần Hồng Quang - 3120560079
Bùi Lê Bích Nhung - 3121410011
Võ Duy Luân - 3121410307
Trần Khôi Nguyên - 3121410353

TP. HỒ CHÍ MINH, THÁNG 5/2024

Mục lục

1	Phần giới thiệu	2
1.1	Giới thiệu về pygame	2
1.2	Giới thiệu về socket	2
1.3	Lý do chọn đề tài	3
1.4	Mục đích và mục tiêu của đề tài	3
2	Quá trình xây dựng trò chơi	4
2.1	Chức năng cơ bản của trò chơi	4
2.2	Cài đặt pygame và các biến cho trò chơi	4
2.3	Xây dựng các thành phần chính của trò chơi	6
2.3.1	Xây dựng lớp Laser	6
2.3.2	Xây dựng lớp Ship	7
2.3.3	Xây dựng lớp BloodIcon	8
2.3.4	Xây dựng lớp Energy	9
2.3.5	Xây dựng lớp Player	10
2.3.6	Xây dựng lớp Enemy	12
2.4	Xây dựng server	14
2.5	Xây dựng menu trò chơi	16
2.5.1	Xây dựng hàm help	17
2.5.2	Xây dựng hàm play	18
2.6	Xây dựng Chat	21
2.7	Xây dựng hàm chạy trò chơi	25
2.8	Hình ảnh về trò chơi	29
3	Tổng kết	33
3.1	Ưu điểm	33
3.2	Nhược điểm	33
3.3	Kết luận	33



1 Phần giới thiệu

1.1 Giới thiệu về pygame

Pygame là một thư viện Python mạnh mẽ được sử dụng để phát triển các ứng dụng trò chơi và đa phương tiện. Với **Pygame**, bạn có thể tạo ra các trò chơi 2D và các ứng dụng đa phương tiện động dễ dàng, nhờ vào các tính năng cơ bản như xử lý sự kiện, đồ họa, âm thanh và đa luồng.

Dưới đây là một số điểm nổi bật về thư viện **Pygame**:

1. **Dễ học và sử dụng:** **Pygame** được thiết kế để dễ dàng tiếp cận, đặc biệt là cho người mới bắt đầu với lập trình trò chơi.
2. **Đa nền tảng:** **Pygame** hoạt động trên nhiều hệ điều hành phổ biến như Windows, macOS và Linux, giúp bạn tạo ra ứng dụng một cách linh hoạt.
3. **Xử lý sự kiện và đồ họa:** **Pygame** cung cấp các công cụ mạnh mẽ để xử lý sự kiện như phím và chuột, cũng như vẽ đồ họa và xử lý hình ảnh.
4. **Âm thanh:** **Pygame** hỗ trợ phát lại âm thanh và âm nhạc, cho phép bạn tạo ra các trải nghiệm đa phương tiện phong phú cho người chơi.
5. **Đa luồng:** **Pygame** cho phép bạn làm việc với nhiều luồng đồng thời, giúp tối ưu hóa hiệu suất và trải nghiệm người dùng.
6. **Cộng đồng lớn:** **Pygame** có một cộng đồng sôi nổi và nhiều tài liệu, ví dụ, mã nguồn và hướng dẫn sẵn sàng để bạn học và phát triển trò chơi của mình.

Với những tính năng này, **Pygame** là một công cụ mạnh mẽ cho việc phát triển trò chơi và ứng dụng đa phương tiện trong Python.

1.2 Giới thiệu về socket

Python cung cấp một thư viện mạnh mẽ gọi là **socket** cho việc lập trình mạng. Thư viện này cho phép bạn tạo và quản lý các kết nối mạng, gửi và nhận dữ liệu qua mạng, và thực hiện các thao tác mạng cơ bản.

Dưới đây là một số điểm nổi bật về thư viện **socket** trong Python:

1. **Đa nền tảng:** Thư viện **socket** hoạt động trên nhiều hệ điều hành khác nhau, bao gồm Windows, macOS và Linux. Điều này cho phép bạn viết mã một cách linh hoạt và chạy ứng dụng mạng trên nhiều nền tảng.
2. **Tích hợp với giao thức TCP/IP và UDP:** **socket** hỗ trợ cả giao thức TCP/IP (Transmission Control Protocol/Internet Protocol) và UDP (User Datagram Protocol), hai giao thức mạng phổ biến nhất. Điều này cho phép bạn tạo ra cả kết nối đáng tin cậy (TCP) và kết nối không đáng tin cậy (UDP) tùy thuộc vào yêu cầu của ứng dụng.
3. **Tạo và quản lý kết nối:** **socket** cho phép bạn tạo ra các đối tượng socket để thiết lập kết nối mạng với các máy chủ hoặc các thiết bị khác trên mạng. Bạn có thể kiểm soát và quản lý các kết nối này để gửi và nhận dữ liệu.



4. **Gửi và nhận dữ liệu:** Với `socket`, bạn có thể gửi dữ liệu từ một máy chủ đến một client hoặc ngược lại. Bạn cũng có thể nhận dữ liệu từ các kết nối mạng và xử lý nó theo cách bạn muốn.
5. **Thiết lập máy chủ và client:** Bằng cách sử dụng thư viện `socket`, bạn có thể viết cả máy chủ và client trong một ứng dụng mạng. Điều này cho phép bạn tạo ra các ứng dụng mạng hoàn chỉnh có khả năng giao tiếp với nhau.

Thư viện `socket` trong Python là một công cụ mạnh mẽ cho việc phát triển ứng dụng mạng đơn giản và phức tạp. Bằng cách sử dụng các tính năng của nó, bạn có thể tạo ra các ứng dụng mạng đa dạng và linh hoạt.

1.3 Lý do chọn đề tài

Hiện nay, ngành Công nghệ thông tin đã có những bước phát triển nhanh chóng, được ứng dụng trong mỗi lĩnh vực của đời sống xã hội. Công nghệ thông tin góp phần đẩy mạnh công cuộc công nghiệp hoá, hiện đại hoá và quá trình phát triển của đất nước. Việc ứng dụng những thành quả của khoa học công nghệ vào đời sống, vào công việc là một việc thiết yếu. Ứng dụng của công nghệ thông tin kết hợp với truyền thông hoá không chỉ giúp cho các hoạt động của các công ty, tổ chức được rõ ràng, minh bạch, mà còn góp phần thay đổi tư duy của con người, giúp con người năng động hơn, kết nối với nhau mọi lúc mọi nơi làm tăng năng suất và hiệu quả của công việc.

Python là một trong những ngôn ngữ lập trình được sử dụng phổ biến nhất hiện nay. Nó rất thích hợp cho những người mới bắt đầu học lập trình vì sự đơn giản về cú pháp và cách viết. Python được dùng để tạo ra các website, ứng dụng để giải quyết các vấn đề về số, xử lý văn bản, tạo ra trò chơi, phát triển dựa trên mô hình machine learning và deep learning.

Trong quá trình tìm hiểu chúng em thấy rất hứng thú với các ứng dụng game được cài đặt và lập trình bằng ngôn ngữ lập trình Python bằng thư viện Pygame. Pygame là một bộ công cụ tiện ích, là một phần của ngôn ngữ lập trình Python, có rất nhiều tựa game huyền thoại đời đầu đã được tạo nên từ ngôn ngữ Python, nên chúng em đã quyết định sử dụng thư viện Pygame của Python để xây dựng game Bắn máy bay để thấy rõ được khả năng mạnh mẽ của nó. Đồng thời sử dụng socket để có thể phát triển thành game dành cho 2 người chơi

1.4 Mục đích và mục tiêu của đề tài

1. Mục đích

- Nắm chắc được kĩ năng và kiến thức về lập trình.
- Tìm hiểu về thư viện Pygame và socket trong ngôn ngữ lập trình Python.
- Củng cố, áp dụng, nâng cao kiến thức đã được học.
- Nắm bắt được quy trình làm game cơ bản.

2. Mục tiêu

- Vận dụng được tính chất của lập trình hướng đối tượng.
- Sử dụng thư viện Pygame để lập trình game Bắn máy bay.
- Sử dụng socket để thiết lập kết nối giữa các máy tính trên mạng



2 Quá trình xây dựng trò chơi

2.1 Chức năng cơ bản của trò chơi

Bắn máy bay (tiếng anh là Space Shooter) là một trò chơi điện tử phổ biến, trong đó người chơi sẽ điều khiển một chiếc máy bay và tiêu diệt các mục tiêu bằng cách bắn đạn hoặc tên lửa.

Trong game, người chơi sẽ phải vượt qua các chướng ngại vật và tránh các đạn được bắn ra từ các mục tiêu khác, đồng thời sẽ so sánh điểm với đối thủ để tìm ra người chiến thắng. Game bắn máy bay còn cho phép người chơi thu thập các vật phẩm như hồi phục máu hay tăng cường sức mạnh cho máy bay của mình, giúp người chơi dễ dàng tiêu diệt các mục tiêu khó khăn hơn.

Nếu người chơi bị các mục tiêu tấn công dẫn đến mất toàn bộ thanh máu hoặc số lần sống sót của máy bay trở về 0 hoặc đối thủ đạt tới số điểm nhất định thì bạn sẽ thua.

2.2 Cài đặt pygame và các biến cho trò chơi

- **Cài đặt pygame**

Để tạo được một chương trình game, trước hết ta phải cài đặt gói **pygame** bằng cách truy cập vào Command Prompt và nhập lệnh:

```
1 pip install pygame
```

- **Import các thư viện cần thiết**

Sau khi cài đặt xong, ta khai báo các thư viện cần thiết cho chương trình:

```
1 import pygame
2 import os
3 import time
4 import random
5 from pygame import mixer
6 import sys
7 import socket
8 import threading
9 import json
10 from threading import Lock
```

- **Khởi tạo màn hình game**

```
1 # initialize pygame, mixer and font
2 pygame.init()
3 pygame.font.init()
4 mixer.init()
5
6 WIDTH, HEIGHT = 700, 670
7 WIN = pygame.display.set_mode((WIDTH, HEIGHT))
```

```
8 pygame.display.set_caption("Space Shooter")
```

Tạo 2 biến WIDTH và HEIGHT lần lượt là chiều dài và chiều rộng của cửa sổ game.

Để tạo ra cửa sổ game, ta dùng `pygame.display.set_mode()` và truyền vào 1 tuple chứa chiều dài và chiều rộng của cửa sổ game, sau đó dùng biến WIN để lưu lại.

Ta gán tiêu đề cho cửa sổ game là “Space Shooter” thông qua hàm `set_caption()`:

- Chèn các hình ảnh

```
1 # Load images
2 RED_SPACE_SHIP = pygame.image.load(os.path.join("assets",
3     "pixel_ship_red.png"))
4 GREEN_SPACE_SHIP = pygame.image.load(os.path.join("assets",
5     "pixel_ship_green.png"))
6 BLUE_SPACE_SHIP = pygame.image.load(os.path.join("assets",
7     "pixel_ship_blue.png"))
8 BLOOD_IMAGE = pygame.image.load(os.path.join("assets", "blood.png"))
9 ENERGY_IMAGE = pygame.image.load(os.path.join("assets", "energy.png"))
```

Chèn các mục tiêu xuất hiện trong game như là máy bay địch hoặc vật phẩm hồi máu hay tăng sức mạnh đạn.

```
1 # Player player
2 YELLOW_SPACE_SHIP = pygame.image.load(os.path.join("assets",
3     "pixel_ship.png"))
4
5 # Lasers
6 RED_LASER = pygame.image.load(os.path.join("assets", "pixel_laser.png"))
7 GREEN_LASER = pygame.image.load(os.path.join("assets", "pixel_laser.png"))
8 BLUE_LASER = pygame.image.load(os.path.join("assets", "pixel_laser.png"))
9 YELLOW_LASER = pygame.image.load(os.path.join("assets", "bullet.png"))
10 DOUBLE_LASER = pygame.image.load(os.path.join("assets",
11     "double_bullet.png"))
12 SUPER_LASER = pygame.image.load(os.path.join("assets", "super_bullet.png"))
```

Chèn hình ảnh của người chơi và các loại đạn được các máy bay bắn ra.

```
1 # Background
2 BG = pygame.transform.scale(pygame.image.load(os.path.join("assets",
3     "background.png")), (WIDTH, HEIGHT))
4
5 # Background menu
6 BG_MENU = pygame.transform.scale(pygame.image.load(os.path.join("assets",
7     "background_menu.jpg")), (WIDTH, HEIGHT))
```

```
7 # them bien background
8 bg = Background(WIN)
```

Chèn các background trong lúc chơi trò chơi.

- Chèn các âm thanh

```
1 # load sound filed
2 shoot_sound = mixer.Sound(os.path.join("assets", "bullet_sound.wav"))
3 shoot_sound_enemy = mixer.Sound(os.path.join("assets",
4 "bullet_sound_enemy.wav"))
5 get_score = mixer.Sound(os.path.join("assets", "get_score.wav"))
6 gameOver_sound = mixer.Sound(os.path.join("assets", "gameOver_sound.wav"))
7 gameWin_sound = mixer.Sound(os.path.join("assets", "get_score.wav"))
```

Chèn âm thanh vào game như âm thanh bắn đạn của người chơi, âm thanh bắn đạn của kẻ thù, âm thanh khi ghi điểm, âm thanh khi game kết thúc và âm thanh khi chiến thắng.

2.3 Xây dựng các thành phần chính của trò chơi

2.3.1 Xây dựng lớp Laser

```
1 class Laser:
2     def __init__(self, x, y, img):
3         self.x = x
4         self.y = y
5         self.img = img
6         self.mask = pygame.mask.from_surface(self.img)
7
8     def draw(self, window):
9         window.blit(self.img, (self.x, self.y))
10
11    def move(self, vel):
12        self.y += vel
13
14    def off_screen(self, height):
15        return not(self.y <= height and self.y >= 0)
16
17    def collision(self, obj):
18        return collide(self, obj)
```

Đây là lớp tạo các viên đạn khi các máy bay bắn ra. Hàm `init()` dùng để khởi tạo các thuộc tính cho laser, bao gồm tọa độ x, y, thuộc tính hình ảnh `img` và khởi tạo một `mask` để đưa `img` lên `surface`.

Hàm `draw()` dùng để vẽ `img` lên `window` bằng cách sử dụng phương thức `blit()` với các tham số là `surface` muốn chèn và một tuple chứa vị trí muốn chèn `img` lên cửa sổ.

Hàm `move()` với tham số là `vel` dùng để dịch chuyển vị trí của laser theo chiều dọc ở mỗi lần lặp game.

Hàm `off_screen()` với tham số là `height` dùng để kiểm tra sự dịch chuyển của laser có nằm trong phạm vi của màn hình chơi hay không.

Hàm `collision()` với tham số là một `obj` để kiểm tra va chạm khi một viên đạn bắn vào một mục tiêu.

2.3.2 Xây dựng lớp Ship

```
1 class Ship:
2     COOLDOWN = 20
3
4     def __init__(self, x, y, health=100):
5         self.x = x
6         self.y = y
7         self.health = health
8         self.ship_img = None
9         self.laser_img = None
10        self.lasers = []
11        self.cool_down_counter = 0
12
13    def draw(self, window):
14        window.blit(self.ship_img, (self.x, self.y))
15        for laser in self.lasers:
16            laser.draw(window)
17
18    def move_lasers(self, vel, obj):
19        self.cooldown()
20        for laser in self.lasers:
21            laser.move(vel)
22            if laser.off_screen(HEIGHT):
23                self.lasers.remove(laser)
24            elif laser.collision(obj):
25                obj.health -= 10
26                self.lasers.remove(laser)
27
28    def cooldown(self):
29        if self.cool_down_counter >= self.COOLDOWN:
30            self.cool_down_counter = 0
31        elif self.cool_down_counter > 0:
32            self.cool_down_counter += 1
33
34    def shoot(self):
35        if self.cool_down_counter == 0:
36            shoot_sound.play()
37            laser = Laser(self.x + 20, self.y, self.laser_img)
38            self.lasers.append(laser)
39            self.cool_down_counter = 1
40
```



```
41     def get_width(self):  
42         return self.ship_img.get_width()  
43  
44     def get_height(self):  
45         return self.ship_img.get_height()
```

Đây là lớp tạo ra các máy bay trong game.

Hàm `init()` dùng để khởi tạo các thuộc tính cho Ship, bao gồm toạ độ x,y, health để chỉ thanh máu, `ship_img`, `laser_img` lưu các img của ship và laser, một list lasers để chứa các laser được vẽ lên màn hình và một biến counter để giảm độ trễ của laser khi bắn.

Hàm `draw()` với tham số window sẽ vẽ các `ship_img` lên màn hình bằng phương thức `blit()`, sau đó với mỗi laser trong list lasers sẽ tiến hành vẽ lên màn hình để khi thực hiện bắn đạn thì hình ảnh đạn được xuất hiện.

Hàm `move_laser()` cho phép laser được chuyển động, đầu tiên ta giảm độ trễ của laser, sau đó duyệt các phần tử trong list laser và gọi hàm `move()` để di chuyển laser. Nếu một laser đã ra khỏi màn hình thì loại bỏ laser đó khỏi list lasers, ngược lại nếu một laser bắn trúng một mục tiêu, thực hiện giảm thanh máu xuống 10 đơn vị và cũng loại bỏ laser đó khỏi list lasers.

Hàm `cooldown()` giảm độ trễ giữa các lần bắn đạn. Nếu biến counter lớn hơn COOLDOWN, biến counter được đặt lại về 0, cho phép người chơi bắn viên đạn tiếp theo. Ngược lại nếu biến counter lớn hơn 0, biến counter tăng lên 1, người chơi không thể bắn được đạn cho đến khi thỏa điều kiện trước đó.

Hàm `shoot()` cho phép máy bay được bắn đạn nếu biến counter có giá trị 0, khi đó các công việc được thực hiện là phát âm thanh khi bắn đạn, khởi tạo một laser có toạ độ và img đi kèm, thêm laser đó vào list lasers để vẽ ra màn hình và biến counter được đặt giá trị là 1.

Hàm `get_width()` và `get_height()` dùng để lấy giá trị chiều dài và chiều rộng của img.

2.3.3 Xây dựng lớp BloodIcon

```
1     class BloodIcon:  
2         def __init__(self, x, y, speed = 1):  
3             self.x = x  
4             self.y = y  
5             self.img = BLOOD_IMAGE  
6             self.mask = pygame.mask.from_surface(self.img)  
7             self.speed = speed  
8  
9         def move(self):  
10             self.y += self.speed  
11  
12         def draw(self, window):  
13             window.blit(self.img, (self.x, self.y))  
14
```

```
15     def collision(self, obj):  
16         return collide(self, obj)
```

Đây là lớp tạo ra các vật phẩm giúp hồi máu khi người chơi bị trúng đạn từ kẻ địch.

Hàm `init()` dùng để khởi tạo các thuộc tính cho `BloodIcon`, bao gồm tọa độ `x,y`, thuộc tính `img` được gán bằng hình ảnh của thanh máu, một `mask` chứa `surface` của `img` vừa khởi tạo và thuộc tính `speed` để gán tốc độ di chuyển cho `BloodIcon`.

Hàm `move()` có tác dụng làm di chuyển `BloodIcon` bằng cách ở mỗi vòng lặp game gán tọa độ `y` bằng giá trị của `speed` cộng 1 đơn vị cho đến khi `BloodIcon` rời khỏi màn hình chơi.

Hàm `draw()` với tham số `window` sẽ vẽ `img` lên màn hình bằng phương thức `blit()`, từ đó tạo ra các `BloodIcon` xuất hiện trên màn hình chơi.

Hàm `collision()` với tham số là 1 `obj` để kiểm tra va chạm khi máy bay của người chơi chạm vào `BloodIcon`, giúp người chơi hồi phục được một lượng máu nhất định trong quá trình chơi.

2.3.4 Xây dựng lớp Energy

```
1     class Energy:  
2         def __init__(self, x, y, power_level, speed=1):  
3             self.x = x  
4             self.y = y  
5             self.vel = speed  
6             self.img = ENERGY_IMAGE  
7             self.power_level = power_level  
8             self.mask = pygame.mask.from_surface(self.img)  
9             self.start_time = time.time()  
10  
11         def move(self):  
12             self.y += self.vel  
13  
14         def draw(self, window):  
15             window.blit(self.img, (self.x, self.y))  
16  
17         def collision(self, obj):  
18             return collide(self, obj)
```

Đây là lớp tạo ra các hỗ trợ giúp người chơi bắn được nhiều đạn cùng lúc, giúp tăng sức mạnh cho đạn khi bắn ra, tiêu diệt kẻ địch nhanh chóng hơn.

Hàm `init()` dùng để khởi tạo các thuộc tính cho `Energy`, bao gồm tọa độ `x,y`, thuộc tính `vel` để gán tốc độ di chuyển cho `Energy`, thuộc tính `img` được gán bằng hình ảnh của viên năng lượng, thuộc tính `power_level` để gán cấp độ sức mạnh của máy bay, một `mask` chứa `surface` của `img` vừa khởi tạo và thuộc tính `start-time` để gán thời gian xuất hiện của viên năng lượng.

Hàm `move()` có tác dụng di chuyển Energy bằng cách ở mỗi vòng lặp game gán tọa độ y bằng giá trị của `vel` cộng 1 đơn vị cho đến khi Energy rời khỏi màn hình chơi.

Hàm `draw()` với tham số `window` sẽ vẽ `ship_img` lên màn hình bằng phương thức `blit()`, từ đó tạo ra các Energy xuất hiện trên màn hình chơi.

Hàm `collision()` với tham số là 1 obj để kiểm tra va chạm khi máy bay của người chơi chạm vào Energy, giúp người chơi tăng cấp độ sức mạnh của máy bay trong quá trình chơi.

2.3.5 Xây dựng lớp Player

```
1 class Player(Ship):
2     def __init__(self, x, y, health=100):
3         super().__init__(x, y, health)
4         self.ship_img = YELLOW_SPACE_SHIP
5         self.laser_img = YELLOW_LASER
6         self.mask = pygame.mask.from_surface(self.ship_img)
7         self.max_health = health
8         self.score = 0
9         self.max_score = self.score + 30
10        self.power_level = 1
11
12    def spawn_blood_icon(self, blood_icons):
13        if random.randint(1, 200) == 1:
14            blood_icon = BloodIcon(random.randint(50, WIDTH - 50), -100)
15            blood_icons.append(blood_icon)
16
17    def spawn_energy_icon(self, energy_icons):
18        while self.score == self.max_score:
19            energy_icon = Energy(random.randint(50, WIDTH - 50), -100)
20            energy_icons.append(energy_icon)
21            self.max_score += 30
22
23    def power_up(self, power_level):
24        if power_level == 1:
25            self.power_level = 1
26        elif power_level == 2:
27            self.power_level = 2
28        elif power_level == 3:
29            self.power_level = 3
30        else:
31            self.power_level = 1 # Neu gia tri khong hop le thi mac dinh la cap
32                               do 1
33
34    def move_lasers(self, vel, objs, blood_icons, energy_icons):
35        self.spawn_blood_icon(blood_icons)
36        self.spawn_energy_icon(energy_icons)
37        self.cooldown()
38        for laser in self.lasers:
39            laser.move(vel)
40            if laser.off_screen(HEIGHT):
```

```
40         self.lasers.remove(laser)
41     else:
42         for obj in objs:
43             if laser.collide(obj):
44                 if self.power_level == 1:
45                     self.laser_img = YELLOW_LASER
46                     self.power_level = 1
47                     for player_laser in self.lasers:
48                         if collide(obj, player_laser):
49                             obj.health -= 20
50                 elif self.power_level == 2:
51                     self.laser_img = DOUBLE_LASER
52                     self.power_level = 2
53                     for player_laser in self.lasers:
54                         if collide(obj, player_laser):
55                             obj.health -= 30
56                 elif self.power_level >= 3:
57                     self.laser_img = SUPER_LASER
58                     self.power_level = 3
59                     for player_laser in self.lasers:
60                         if collide(obj, player_laser):
61                             obj.health -= 50
62             else:
63                 self.power_level = 1
64             if obj.health <= 0:
65                 objs.remove(obj)
66                 self.score += 10
67                 get_score.play()
68             if laser in self.lasers:
69                 self.lasers.remove(laser)
70
71     def draw(self, window):
72         super().draw(window)
73         self.healthbar(window)
74
75     def healthbar(self, window):
76         pygame.draw.rect(window, (255,0,0), (self.x, self.y +
77             self.ship_img.get_height() + 10, self.ship_img.get_width(), 7))
78         pygame.draw.rect(window, (0,255,0), (self.x, self.y +
79             self.ship_img.get_height() + 10, self.ship_img.get_width() *
80             (self.health/self.max_health), 7))
```

Đây là lớp tạo ra máy bay cho người chơi điều khiển.

Hàm `init()` dùng để khởi tạo các thuộc tính cho Player, bao gồm việc kế thừa hàm Ship đã được xây dựng để tạo ra Ship, thuộc tính `ship_img` và `laser_ship` gán bằng hình ảnh của máy bay và laser, một mask chứa surface của img vừa khởi tạo, thuộc tính `max_health` gán bằng số lượng máu mặc định của máy bay, thuộc tính `score` gán bằng số điểm ban đầu, thuộc tính `max_score` gán bằng số điểm ban đầu cộng thêm 30 và thuộc tính `power_level` gán bằng sức mạnh ban đầu của máy bay là 1.

Hàm `spawn_blood_icon()` dùng để tạo random viên máu xuất hiện ở vị trí ngẫu nhiên

trên màn hình. Nếu số random từ 1-200 là 1 thì sẽ tạo 1 BloodIcon ở vị trí random gán vào blood_icon, sau đó thêm blood_icon vào blood_icons.

Hàm `spawn_energy_icon()` dùng để tạo random viên năng lượng xuất hiện ở vị trí ngẫu nhiên trên màn hình. Nếu score bằng max_score thì sẽ tạo 1 Energy ở vị trí random gán vào energy_icon, sau đó thêm energy_icon vào energy_icons và max_score cộng thêm 30.

Hàm `power_up()` dùng để gán cấp độ sức mạnh hiện có của máy bay.

Hàm `move_lasers()` được dùng để tạo laser của máy bay. Nếu power_level của máy bay là 1 thì sẽ gán laser_img là YELLOW_LASER và sẽ làm máu của mục tiêu giảm 20 nếu va chạm. Nếu power_level của máy bay là 2 thì sẽ gán laser_img là DOUBLE_LASER và sẽ làm máu của mục tiêu giảm 30. Nếu power_level của máy bay là 3 thì sẽ gán laser_img là SUPER_LASER và sẽ làm máu của mục tiêu giảm 40. Nếu power_level của máy bay là 4 thì sẽ gán laser_img là SUPER_LASER và sẽ làm máu của mục tiêu giảm 50. Nếu máu của mục tiêu obj.health <=0 thì remove(obj) khỏi objs và tăng điểm của người chơi lên 10.

Hàm `draw()` với tham số window sẽ vẽ img ship của Player lên màn hình cùng với thanh máu tương ứng ở hàm `healthbar()`, thanh máu gồm 2 thanh là một thanh màu xanh và một thanh màu đỏ, thanh màu xanh báo hiệu máu đầy còn màu đỏ là lượng máu giảm đi, nếu màu đỏ toàn bộ thanh thì kết thúc trò chơi.

2.3.6 Xây dựng lớp Enemy

```
1 class Enemy(Ship):
2     COLOR_MAP = {
3         "red": (RED_SPACE_SHIP, RED_LASER),
4         "green": (GREEN_SPACE_SHIP, GREEN_LASER),
5         "blue": (BLUE_SPACE_SHIP, BLUE_LASER)
6     }
7
8     """
9     Initializes an enemy ship object.
10
11     Args:
12     x (int): The x-coordinate of the enemy ship.
13     y (int): The y-coordinate of the enemy ship.
14     color (str): The color of the enemy ship.
15     health (int): The health of the enemy ship.
16     """
17     def __init__(self, x, y, color, health=100):
18         super().__init__(x, y, health)
19         self.ship_img, self.laser_img = self.COLOR_MAP[color]
20         self.mask = pygame.mask.from_surface(self.ship_img)
21         self.cool_down_counter = 0
22
23     # Moves the enemy ship down by the specified velocity.
24     def move(self, vel):
25         self.y += vel
26
```

```
27     def kill(self):
28         self.visible = False
29         self.health = 0
30
31     def shoot(self, player):
32         if self.cool_down_counter == 0:
33             laser = Laser(self.x + 20, self.y + 20, self.laser_img)
34             self.lasers.append(laser)
35             self.cool_down_counter = 1
36             for player_laser in player.lasers:
37                 if collide(self, player_laser):
38                     self.health -= 20
39                     player.lasers.remove(player_laser)
40                     if self.health <= 0:
41                         self.kill()
42                     else:
43                         self.healthbar(WIN)
44
45     def off_screen(self, height):
46         return self.y > height or self.y < 0
47
48     # Draws a health bar on the screen to represent the enemy's health.
49     def healthbar(self, screen):
50         bar_length = self.ship_img.get_width()
51         bar_height = 7
52         health_bar = (self.health / 100) * bar_length
53         pygame.draw.rect(screen, (255, 0, 0), (self.x, self.y +
54             self.ship_img.get_height() + 10, bar_length, bar_height))
55         pygame.draw.rect(screen, (0, 255, 0), (self.x, self.y +
56             self.ship_img.get_height() + 10, health_bar, bar_height))
```

Đây là lớp tạo các kẻ địch trong trò chơi, Player sẽ phải đối đầu với chúng nếu muốn vượt qua các level khác trong trò chơi.

Đầu tiên ta tạo một dict COLOR_MAP chứa các kẻ địch gồm 3 loại red, green và blue, gán giá trị cho chúng là các tuple chứa img của ship và laser tương ứng.

Hàm `init()` dùng để khởi tạo các thuộc tính cho Enemy, bao gồm việc kế thừa hàm Ship đã được xây dựng để tạo ra lớp Ship, các img của ship và laser được lấy ra từ dict COLOR_MAP, một mask chứa surface của img vừa khởi tạo và một biến đếm để giảm độ trễ khi bắn đạn.

Hàm `move()` có tác dụng di chuyển Enemy bằng cách ở mỗi vòng lặp game gán toạ độ y bằng giá trị của vel cộng 1 đơn vị cho đến khi Enemy rời khỏi màn hình chơi.

Hàm `kill()` có tác dụng khi Player giết được Enemy thì cho Enemy ẩn khỏi màn hình chơi và cho thanh máu của chúng trở về 0.

Hàm `shoot()` với tham số là player dùng để xét sự kiện khi player bắn vào enemy, nếu đạn của player va chạm với enemy thì giảm máu của enemy xuống 20 ở mỗi lần trúng, khi mà máu của enemy trở về 0 thì gọi hàm kill để loại enemy ra khỏi màn hình chơi, ngược lại thì giảm thanh máu của enemy xuống tương ứng với lượng máu đã mất.

Hàm `off_screen()` dùng để xét 1 đối tượng có đang ở trong phạm vi trò chơi hay không.

Hàm `healthbar()` có tác dụng xây dựng thanh máu cho mỗi enemy xuất hiện, cũng có 2 thanh màu xanh và màu đỏ giống như player.

2.4 Xây dựng server

```
1 class Server:
2     def __init__(self, max_connections_callback=None):
3         self.host = socket.gethostname(socket.gethostname())
4         self.port = 5500
5         self.max_connections = 2
6         self.current_connections = 0
7         self.max_connections_reached = False
8
9         self.server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
10        self.server_socket.bind((self.host, self.port))
11        self.server_socket.listen()
12
13        print(f"Server đang lang nghe tren {self.host}:{self.port}")
14
15        self.clients = []
16        self.max_connections_callback = max_connections_callback
17
18        self.accept_connections()
19
20    def accept_connections(self):
21        while True:
22            if self.current_connections < self.max_connections:
23                try:
24                    client_socket, client_address = self.server_socket.accept()
25                    print(f"Ket noi tu {client_address} duoc chap nhan")
26
27                    self.clients.append(client_socket)
28                    self.current_connections += 1
29
30                    threading.Thread(target=self.handle_client,
31                                    args=[client_socket]).start()
32
33                    # Kiem tra neu da du ket noi, gui thong bao den tat ca client
34                    if self.current_connections == self.max_connections:
35                        print("Da du ket noi, gui thong bao den tat ca client")
36                        for client in self.clients:
37                            try:
38                                client.send("ready_to_start".encode())
39                            except Exception as e:
40                                str(e)
41                    except Exception as e:
42                        str(e)
```

```
43         elif not self.max_connections_reached:
44             print("Đã đạt đến số lượng kết nối tối đa, từ chối kết nối mới.")
45             self.max_connections_reached = True
46             if self.max_connections_callback:
47                 self.max_connections_callback() # Gọi hàm callback
48
49     def handle_client(self, client_socket):
50         try:
51             while True:
52                 message = client_socket.recv(1024).decode()
53
54                 if not message:
55                     break
56
57                 print(f"Nhan được từ {client_socket.getpeername()}: {message}")
58
59                 for client in self.clients:
60                     if client is not client_socket and isinstance(client,
61                             socket.socket):
62                         try:
63                             client.send(message.encode())
64                         except Exception as e:
65                             str(e)
66
67             except ConnectionResetError:
68                 print(f"Client {client_socket.getpeername()} đã đóng kết nối một
69                     cách bất ngờ.")
70             except Exception as e:
71                 str(e)
72             finally:
73                 client_socket.close() # Đóng kết nối với client
74                 self.clients.remove(client_socket)
75                 self.current_connections -= 1
76
77     def max_connections_reached_callback():
78         # Thông báo cho main khi máy chủ đạt đến số lượng kết nối tối đa
79         print("Máy chủ đã đạt đến số lượng kết nối tối đa.")
80
81     # Tạo một biến global để lưu trữ thông tin máy chủ
82     server_instance =
83         Server(max_connections_callback=max_connections_reached_callback)
```

Hàm `init(self, max_connections_callback=None)` khởi tạo các thuộc tính của máy chủ như host, cổng, số kết nối tối đa được phép, số kết nối hiện tại, v.v. Nó tạo một đối tượng socket (`server_socket`), gắn nó với host và cổng, và bắt đầu lắng nghe các kết nối đến. Nếu có một `max_connections_callback` được cung cấp, nó sẽ được lưu trữ để sử dụng sau này. Cuối cùng, nó gọi phương thức `accept_connections` để bắt đầu chấp nhận các kết nối đến.

Hàm `accept_connections(self)` liên tục chấp nhận các kết nối đến trong một vòng lặp cho đến khi số lượng kết nối tối đa được đạt. Khi một kết nối mới được chấp nhận, nó in ra một thông báo chỉ ra việc chấp nhận kết nối, thêm socket của client vào danh sách `clients`, và bắt

đầu một luồng mới để xử lý giao tiếp với client. Nếu số lượng kết nối tối đa được đạt, nó gửi một thông điệp "ready_to_start" cho tất cả client nếu chúng tồn tại.

Hàm `handle_client(self, client_socket)` xử lý giao tiếp với một client duy nhất. Nó liên tục nhận các thông điệp từ client và phát sóng chúng cho tất cả các client khác đang kết nối. Nếu một client ngắt kết nối một cách đột ngột, nó loại bỏ client khỏi danh sách các client và giảm số lượng kết nối hiện tại.

Hàm `max_connections_reached_callback()` được gọi khi số lượng kết nối tối đa được đạt. Nó đơn giản chỉ in ra một thông báo chỉ ra rằng máy chủ đã đạt đến giới hạn kết nối tối đa của mình.

2.5 Xây dựng menu trò chơi

```
1 def main_menu():
2     title_font = pygame.font.SysFont("Arial", 50)
3     run = True
4     while run:
5         WIN.blit(BG, (0,0))
6
7         MENU_MOUSE_POS=pygame.mouse.get_pos()
8         #game header text
9         MENU_TEXT=get_font(45).render("SPACE SHOOTER",True,"#b68f40")
10        MENU_RECT=MENU_TEXT.get_rect(center=(340,50))
11
12        #buttons in main menu
13        PLAY_BUTTON = Button(image=pygame.image.load(os.path.join("assets",
14        "Play_Rect.png")), pos=(340, 250),
15        text_input="PLAY", font=get_font(45),
16        base_color="#d7fcd4", hovering_color="Green")
17        HELP_BUTTON = Button(image=pygame.image.load(os.path.join("assets",
18        "Help_Rect.png")), pos=(340, 400),
19        text_input="HELP", font=get_font(45),
20        base_color="#d7fcd4", hovering_color="Green")
21        QUIT_BUTTON = Button(image=pygame.image.load(os.path.join("assets",
22        "Quit_Rect.png")), pos=(340, 550),
23        text_input="QUIT", font=get_font(45),
24        base_color="#d7fcd4", hovering_color="Green")
25
26        WIN.blit(MENU_TEXT, MENU_RECT)
27
28        for button in [PLAY_BUTTON, HELP_BUTTON, QUIT_BUTTON]:
29            button.changeColor(MENU_MOUSE_POS)
30            button.update(WIN)
31
32        for event in pygame.event.get():
33            if event.type == pygame.QUIT:
34                run = False
35                # pygame.quit()
36                # sys.exit()
```

```
31         if event.type == pygame.MOUSEBUTTONDOWN:
32             if PLAY_BUTTON.checkForInput(MENU_MOUSE_POS):
33                 play()
34             if HELP_BUTTON.checkForInput(MENU_MOUSE_POS):
35                 help()
36             if QUIT_BUTTON.checkForInput(MENU_MOUSE_POS):
37                 pygame.quit()
38                 sys.exit()
39
40     pygame.display.update()
```

Đây là một hàm để hiển thị menu chính của trò chơi.

Trong vòng lặp while run, nó hiển thị hình nền của menu và các nút chức năng, bao gồm PLAY_BUTTON (nút chơi), HELP_BUTTON (nút trợ giúp) và QUIT_BUTTON (nút thoát). Nó cũng theo dõi vị trí của chuột để thay đổi màu sắc của các nút khi chuột di chuyển qua chúng.

Nếu người chơi nhấp vào nút “PLAY”, hàm sẽ gọi hàm play() để bắt đầu trò chơi. Nếu người chơi nhấp vào nút “HELP”, hàm sẽ gọi hàm help() để hiển thị hướng dẫn chơi. Nếu người chơi nhấp vào nút “QUIT”, hàm sẽ thoát khỏi trò chơi.

Hàm cũng kiểm tra các sự kiện trong vòng lặp, bao gồm sự kiện nhấn nút chuột. Nếu người chơi nhấp vào nút PLAY, HELP hoặc QUIT, chương trình sẽ thoát khỏi vòng lặp và thực hiện hành động tương ứng.

Cuối cùng, hàm sử dụng hàm pygame.display.update() để cập nhật màn hình trò chơi.

2.5.1 Xây dựng hàm help

```
1  #mode help in main menu
2  def help():
3      while True:
4          HELP_MOUSE_POS=pygame.mouse.get_pos()
5          WIN.fill("black")
6          str='''Players must destroy all enemies on their flight path to pass
7              different levels. Players use a keyboard with keys W, A, D, S to
8              move the plane, use SPACE to shoot bullets. Players can also use
9              helpful items such as bombs, missiles, or energy to help them
10             destroy opponents faster'''
11         blit_text(WIN,str,(20,100),get_font(25))
12         # WIN.blit(HELP_TEXT,HELP_RECT)
13
14         HELP_BACK = Button(image=None, pos=(340, 500), text_input="BACK",
15                             font=get_font(40), base_color="White", hovering_color="Green")
16
17         HELP_BACK.changeColor(HELP_MOUSE_POS)
18         HELP_BACK.update(WIN)
19
20     for event in pygame.event.get():
```

```
16         if event.type==pygame.QUIT:
17             pygame.quit()
18             sys.exit()
19         if event.type==pygame.MOUSEBUTTONDOWN:
20             if HELP_BACK.checkForInput(HELP_MOUSE_POS):
21                 main_menu()
22
23     pygame.display.update()
```

Đây là hàm hiển thị hộp thoại HELP cho trò chơi, đoạn mã sẽ hiển thị một đoạn văn bản giới thiệu về trò chơi và các phím điều khiển.

Vòng lặp while True sẽ liên tục chạy để hiển thị hộp thoại trợ giúp cho đến khi người dùng nhấn vào nút "BACK" hoặc đóng cửa sổ.

Đầu tiên, đoạn mã sử dụng hàm `pygame.mouse.get_pos()` để lấy vị trí hiện tại của chuột và lưu trữ nó vào biến `HELP_MOUSE_POS`.

Sau đó, đoạn mã sử dụng hàm `WIN.fill("black")` để xóa màn hình và chuẩn bị vẽ lại các yếu tố mới.

Tạo biến str để lưu trữ đoạn văn bản hướng dẫn trò chơi.

Đoạn văn bản được lưu trữ trong biến str sẽ được hiển thị bằng cách sử dụng hàm `blit_text()`, truyền vào vị trí (40, 100), font chữ và kích thước phù hợp.

Nút "BACK" được tạo bằng lớp Button và được cập nhật trạng thái và hiển thị bằng hàm `HELP_BACK.update(WIN)`.

Trong vòng lặp for hàm sử dụng câu lệnh điều kiện để kiểm tra xem loại sự kiện được kích hoạt là gì: +Nếu sự kiện là "QUIT", hàm sẽ gọi hàm `pygame.quit()` để dừng pygame và sau đó sẽ gọi `sys.exit()` để kết thúc chương trình. +Nếu loại sự kiện là "MOUSEBUTTONDOWN" (bấm chuột và người dùng đã nhấp vào nút "BACK", đoạn mã gọi hàm `main_menu()` để quay trở lại menu chính.

Cuối cùng, hàm sử dụng hàm `pygame.display.update()` để cập nhật màn hình trò chơi.

2.5.2 Xây dựng hàm play

```
1     # mode play in main menu
2     connected = threading.Event()
3     ready_to_start = threading.Event()
4     client_socket = None
5
6     def reset():
7         reset_game_state()
8         if client_socket is not None:
9             try:
```



```
10         client_socket.close()
11         print("Socket closed successfully.")
12     except Exception as e:
13         print(f"Error closing socket: {e}")
14
15 def connect_to_server():
16     global client_socket
17     try:
18         client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
19         host = socket.gethostbyname(socket.gethostname())
20         # host = "192.168.187.206"
21         port = 5500
22         client_socket.connect((host, port))
23         connected.set()
24         print("Connected to server successfully.")
25     except Exception as e:
26         print(f"Loi khi ket noi toi may chu: {e}")
27         client_socket = None
28
29 def ready_from_server():
30     global client_socket
31     while True:
32         if ready_to_start.is_set():
33             break
34         try:
35             if client_socket is not None:
36                 if connected.is_set():
37                     # Nhan thong diep tu may chu
38                     message = client_socket.recv(1024).decode()
39                     if message == "ready_to_start":
40                         ready_to_start.set()
41                 else:
42                     # Dung mot thoi gian ngan de cho socket co co hoi ket noi
43                     time.sleep(0.1)
44             except ConnectionResetError as e:
45                 # print(f"Ket noi da bi dong boi may chu: {e}")
46                 break
47             except Exception as e:
48                 # print(f"Loi khi nhan thong diep tu may chu: {e}")
49                 break
50
51 def reset_game_state():
52     global connected, ready_to_start
53     connected.clear()
54     ready_to_start.clear() # Dat lai trang thai "ready_to_start"
55
56 def play():
57     global connected, ready_to_start, client_thread
58
59     # Reset game state
60     reset_game_state()
61
```

```
62     waiting_text = "Waiting for other players..."
63     start_time = time.time()
64     display_start_message = False
65     connect = False
66
67     # Create Pygame screen
68     screen = pygame.display.set_mode((WIDTH, HEIGHT))
69
70     running = True
71     while running:
72         for event in pygame.event.get():
73             if event.type == pygame.QUIT:
74                 pygame.quit()
75                 sys.exit()
76             elif event.type == pygame.KEYDOWN:
77                 if event.key == pygame.K_ESCAPE:
78                     if connect:
79                         try:
80                             client_socket.close()
81                         except Exception as e:
82                             print(f"Error closing socket: {e}")
83                     running = False
84                 elif event.key == pygame.K_p:
85                     if not connect:
86                         connect_to_server()
87                         connected.wait() # Wait until connected
88                     if connected.is_set():
89                         client_thread =
90                             threading.Thread(target=ready_from_server)
91                         client_thread.start()
92                         connect = True
93
94                     if not connect:
95                         screen.fill((0, 0, 0))
96                         font = pygame.font.Font(None, 36)
97                         text = font.render("Press 'P' to play", True, (255, 255, 255))
98                         text_rect = text.get_rect(center=(WIDTH // 2, HEIGHT // 2))
99                         screen.blit(text, text_rect)
100                     else:
101                         screen.fill((0, 0, 0))
102                         font = pygame.font.Font(None, 36)
103                         text = font.render(waiting_text, True, (255, 255, 255))
104                         text_rect = text.get_rect(center=(WIDTH // 2, HEIGHT // 2))
105                         screen.blit(text, text_rect)
106
107                     if ready_to_start.is_set() and not display_start_message:
108                         display_start_message = True
109                         waiting_text = "Both players are ready, let's start the game..."
110
111                 pygame.display.flip()
112
113             if display_start_message:
```

```
113         elapsed_time = time.time() - start_time
114         if elapsed_time >= 1:
115             running = False
116             # Khởi tạo thread để nhận tin nhắn
117             global chat_log
118             chat_log = []
119             receive_thread = threading.Thread(target=receive_messages)
120             receive_thread.daemon = True # Đặt thread nhận tin nhắn thành
121                 daemon để nó tự động khi chương trình chính kết thúc
122             receive_thread.start()
123             main()
124
125         # Reset game state if the loop breaks
126         reset_game_state()
127
128     # Khởi tạo luồng cho client
129     client_thread = threading.Thread(target=ready_from_server)
130     client_thread.start()
```

Biến `connected` và `ready_to_start` là các biến kiểu Event từ module `threading`, được sử dụng để đồng bộ hóa việc kết nối và sự sẵn sàng của các máy client. Khi một sự kiện xảy ra, các thread có thể chờ hoặc tiếp tục thực thi.

Hàm `reset()` được sử dụng để thiết lập lại trạng thái của trò chơi. Nó đóng kết nối socket nếu nó đang mở và in ra một thông báo nếu có lỗi xảy ra.

Hàm `connect_to_server()` thực hiện việc kết nối máy client với máy chủ thông qua một socket. Nếu kết nối thành công, biến `connected` được đặt.

Hàm `ready_from_server()` chạy trong một vòng lặp vô hạn để kiểm tra sự sẵn sàng từ máy chủ. Nếu máy chủ gửi tin nhắn `"ready_to_start"`, biến `ready_to_start` được đặt.

Hàm `reset_game_state()` được sử dụng để đặt lại trạng thái của trò chơi bằng cách xóa các biến `connected` và `ready_to_start`.

Hàm `play()` là hàm chính để bắt đầu trò chơi. Nó xử lý việc kết nối với máy chủ, đợi sự sẵn sàng từ máy chủ, và hiển thị thông điệp `"waiting_text"` trong khi đợi. Khi cả hai máy client đều sẵn sàng, trò chơi bắt đầu. Điều này được xác định bằng biến `display_start_message`. Khi trò chơi kết thúc, trạng thái của trò chơi được đặt lại bằng hàm `reset_game_state()`.

Biến `client_thread` là một luồng được tạo ra để lắng nghe các tin nhắn từ máy chủ bằng cách gọi hàm `ready_from_server()`.

2.6 Xây dựng Chat

```
1     #Chat
2     class ChatButton:
```



```
3     def __init__(self, image, pos):
4         self.image = image
5         self.pos = pos
6
7     def draw(self, win):
8         win.blit(self.image, self.pos)
9
10    def is_clicked(self, pos):
11        x, y = pos
12        button_rect = self.image.get_rect(topleft=self.pos)
13        return button_rect.collidepoint(x, y)
14
15    # Tao mot doi tuong lock
16    chat_log_lock = threading.Lock()
17
18    # Trong ham send_message:
19    def send_message(input_text):
20        try:
21            client_socket.sendall(input_text.encode('utf-8'))
22            # Them tin nhan vao chat_log
23            with chat_log_lock:
24                chat_log.append("You: "+input_text)
25        except Exception as e:
26            print("Error sending message:", e)
27
28    # Trong ham receive_messages:
29    def receive_messages():
30        try:
31            while True:
32                message = client_socket.recv(1024).decode('utf-8')
33                if message:
34                    with chat_log_lock:
35                        # Them tin nhan tu client vao chat log
36                        chat_log.append(message)
37                        global win, lost
38                        if message == "status:lost":
39                            win = True
40                        elif message == "status:win":
41                            lost = True
42                    else:
43                        break
44        except Exception as e:
45            str(e)
46
47    chat_log = [] # Danh sach de luu tin nhan trong cuoc tro chuyen
48
49    def chat_box():
50        run_chat = True
51        chat_font = pygame.font.SysFont("Arial", 14)
52        input_font = pygame.font.SysFont("Arial", 14)
53
54        current_client_address = client_socket.getpeername()
```

```
55
56     input_text = "" # Biến để lưu nội dung của ô nhập liệu
57
58     # Kích thước của chat menu
59     chat_menu_width = WIDTH // 2
60     chat_menu_height = HEIGHT // 2
61
62     # Vị trí của chat menu
63     chat_menu_x = 10
64     chat_menu_y = 150
65
66     # Vị trí của nút tắt
67     close_button_x = chat_menu_x + chat_menu_width - 30
68     close_button_y = chat_menu_y + 10
69
70     while run_chat:
71         for event in pygame.event.get():
72             if event.type == pygame.QUIT:
73                 pygame.quit() # Thoát khỏi trò chơi khi của sổ bị đóng
74                 sys.exit()
75             if event.type == pygame.KEYDOWN:
76                 if event.key == pygame.K_RETURN: # Xử lý khi người dùng nhấn Enter
77                     if input_text: # Đảm bảo rằng ô nhập liệu không trống
78                         send_message(input_text) # Gửi tin nhắn đến máy chủ
79                         input_text = "" # Đặt lại nội dung của ô nhập liệu
80                     elif event.key == pygame.K_BACKSPACE: # Xử lý khi người dùng nhấn
81                         # phím Backspace
82                         input_text = input_text[:-1] # Xóa ký tự cuối cùng trong ô
83                         # nhập liệu
84                     else:
85                         # Thêm ký tự vào ô nhập liệu
86                         input_text += event.unicode if event.unicode.isprintable()
87                         else ""
88
89                 elif event.type == pygame.MOUSEBUTTONDOWN: # Xử lý khi người dùng
90                     # nhấn chuột
91                     mouse_pos = pygame.mouse.get_pos()
92                     # Kiểm tra nếu người dùng nhấn vào nút tắt
93                     if close_button_x <= mouse_pos[0] <= close_button_x + 20 and \
94                         close_button_y <= mouse_pos[1] <= close_button_y + 20:
95                         run_chat = False # Đóng cửa sổ chat menu
96
97             # Vẽ nền trong suốt cho chat menu
98             chat_menu_surface = pygame.Surface((chat_menu_width, chat_menu_height),
99                 pygame.SRCALPHA)
100             pygame.draw.rect(chat_menu_surface, (0, 0, 0, 100), (0, 0,
101                 chat_menu_width, chat_menu_height), border_radius=10)
102             WIN.blit(chat_menu_surface, (chat_menu_x, chat_menu_y))
103
104             # Vẽ giao diện chat
105             y_offset = chat_menu_y + 10
106             for message in chat_log:
```



```
101         # Kiểm tra xem tin nhắn có đến từ client khác không
102         if message.startswith("You:") and message[5:] !=
            current_client_address:
103             message = "You: " + message[5:]
104         else:
105             message = "Other: " + message
106
107         text_surface = chat_font.render(message, True, (255, 255, 255)) #
            Render tin nhắn
108         WIN.blit(text_surface, (chat_menu_x + 10, y_offset)) # Vẽ tin nhắn
            lên màn hình
109         y_offset += text_surface.get_height() + 5 # Tăng y_offset để vẽ tin
            nhắn tiếp theo
110
111         # Vẽ ô nhập liệu
112         pygame.draw.rect(WIN, (255, 255, 255), (chat_menu_x + 5, chat_menu_y +
            chat_menu_height - 35, chat_menu_width - 10, 30), 2) # Vẽ khung ô
            nhập liệu
113         input_surface = input_font.render(input_text, True, (255, 255, 255)) #
            Render nội dung ô nhập liệu
114         WIN.blit(input_surface, (chat_menu_x + 10, chat_menu_y +
            chat_menu_height - 30)) # Vẽ nội dung ô nhập liệu lên màn hình
115
116         # Vẽ nút tắt
117         pygame.draw.rect(WIN, (255, 0, 0), (close_button_x, close_button_y, 20,
            20))
118         close_text = input_font.render("X", True, (255, 255, 255))
119         WIN.blit(close_text, (close_button_x + 4, close_button_y))
120
121         pygame.display.update()
```

Lớp ChatButton đại diện cho một nút trong giao diện chat. Các đối tượng ChatButton được tạo ra với một hình ảnh và vị trí. Phương thức draw được sử dụng để vẽ nút trên cửa sổ Pygame. Phương thức is_clicked kiểm tra xem vị trí chuột có nằm trong vùng của nút không.

Biến chat_log_lock là một đối tượng khóa (Lock) trong module threading, được sử dụng để đồng bộ hóa truy cập vào danh sách chat_log giữa các luồng.

Hàm send_message(client_socket, input_text) được sử dụng để gửi tin nhắn từ client lên server thông qua client_socket. Tin nhắn được mã hóa thành chuỗi UTF-8 trước khi gửi. Sau đó, tin nhắn được thêm vào chat_log với sự bảo vệ của chat_log_lock.

Hàm receive_messages(client_socket, chat_log) chạy trong một vòng lặp vô hạn để nhận tin nhắn từ server thông qua client_socket. Tin nhắn được giải mã từ dạng UTF-8 và sau đó được thêm vào chat_log với sự bảo vệ của chat_log_lock.

Biến chat_log là một danh sách để lưu trữ các tin nhắn trong cuộc trò chuyện.

Hàm chat_box() tạo ra giao diện chat trong Pygame. Nó chạy một vòng lặp vô hạn để vẽ giao diện và xử lý sự kiện từ người dùng. Giao diện bao gồm các tin nhắn được hiển thị trong một ô chat, một ô nhập liệu cho người dùng nhập tin nhắn, và một nút "đóng" để đóng cửa sổ

chat. Tin nhắn mới từ chat_log được hiển thị và tin nhắn từ người dùng được gửi thông qua hàm send_message().

2.7 Xây dựng hàm chạy trò chơi

```
1  def main():
2      global run
3      run = True
4      level = 0
5      enemies = []
6      explosions = []
7      player_vel = 6
8      laser_vel = 7
9      player = Player(300, 550)
10     dif = 1 # health bar of enemise reduce rely on this variable
11     blood_icons = []
12     energy_icons = []
13     bg_running = 0
14
15     main_font = pygame.font.SysFont("Arial", 30)
16     lost_font = pygame.font.SysFont("Arial", 40)
17     win_font = pygame.font.SysFont("Arial", 40)
18
19     # get the level from user
20     FPS = 50
21     wave_length = 5
22     enemy_vel = 1
23     lives = 6
24     dif = 4
25     bg_running = 0.50
26     clock = pygame.time.Clock()
27
28     global lost, win, status_sent
29
30     lost = False
31
32     win = False
33
34     status_sent = False
35
36
37     # Ve nut chat chi khi nguoi dung da nhan "PLAY"
38     CHAT_BUTTON_POS = (10, 150) # Vi tri cua nut chat
39     CHAT_BUTTON_IMAGE = pygame.image.load(os.path.join("assets",
40     "Chat_Rect.png")) # Load hinh anh cua nut chat
41     chat_button = ChatButton(CHAT_BUTTON_IMAGE, CHAT_BUTTON_POS)
42
43     def redraw_window():
44         WIN.blit(BG, (0,0))
45         bg.update(bg_running) # running background
46         # draw text
```



```
46     lives_label = main_font.render(f"Lives: {lives}", 1, (255,255,255))
47     level_label = main_font.render(f"Level: {level}", 1, (255,255,255))
48     score_label = main_font.render(f"Score: {player.score}", 1,
49                                     (255,255,255))
50
51     chat_button.draw(WIN)
52
53     WIN.blit(lives_label, (10, 10))
54     WIN.blit(level_label, (WIDTH - level_label.get_width() - 10, 10))
55     WIN.blit(score_label, (WIDTH/2 - score_label.get_width()/2, 10))
56
57     for enemy in enemies:
58         enemy.draw(WIN)
59
60     for laser in player.lasers:
61         laser.draw(WIN)
62
63     for blood_icon in blood_icons:
64         blood_icon.draw(WIN)
65
66     for energy_icon in energy_icons:
67         energy_icon.draw(WIN)
68
69     player.draw(WIN)
70
71     for enemy in enemies:
72         enemy.draw(WIN)
73         enemy.healthbar(WIN) # draw health bar for each enemy
74
75     if lost:
76         lost_label = lost_font.render("Game Over!", 1, (255,255,255))
77         if lost_label:
78             gameOver_sound.play()
79             WIN.blit(lost_label, (WIDTH/2 - lost_label.get_width()/2, 350))
80
81     if win:
82         win_label = win_font.render("You Win!", 1, (255,255,255))
83         if win_label:
84             gameWin_sound.play()
85             WIN.blit(win_label, (WIDTH/2 - win_label.get_width()/2, 350))
86
87     pygame.display.update()
88
89     def delay_run_false():
90         global run
91         run = False
92
93     def send_game_status(status):
94         message = "status:"+status
95         client_socket.sendall(message.encode('utf-8'))
96
97     while run:
```



```
197         clock.tick(FPS)
198         redraw_window()
199
200         if lives <= 0 or player.health <= 0:
201             lost = True
202
203         if player.score == 20:
204             win = True
205
206         if (win or lost) and not status_sent:
207             if win:
208                 send_game_status('win') # Gui thông tin chiến thắng tới server
209             if lost:
210                 send_game_status('lost') # Gui thông tin thua cuộc tới server
211
212             threading.Timer(2, delay_run_false).start() # Trì hoãn việc đặt run
                thành False sau 2 giây
213             threading.Timer(2, reset).start() # Trì hoãn việc reset sau 2 giây
214             status_sent = True # Dát có để ngăn chặn việc gửi nhiều lần
215
216
217         if len(enemies) == 0:
218             level += 1
219             wave_length += 5
220             for i in range(wave_length):
221                 enemy = Enemy(random.randrange(50, WIDTH-100),
222                               random.randrange(-1500, -100), random.choice(["red", "blue",
223                                     "green"]))
224                 enemies.append(enemy)
225
226         # spawn new BloodIcon objects randomly at the top of the screen
227         if random.randint(0, 1300) < 1:
228             blood_icon = BloodIcon(random.randint(50, WIDTH-50), 0)
229             blood_icons.append(blood_icon)
230
231         # move and draw the BloodIcon objects
232         for blood_icon in blood_icons:
233             blood_icon.move()
234             blood_icon.draw(WIN)
235             if blood_icon.collision(player):
236                 player.health += 20
237                 if player.health > player.max_health:
238                     player.health = player.max_health
239                 blood_icons.remove(blood_icon)
240
241         # spawn new EnergyIcon objects randomly at the top of the screen
242         while player.score == player.max_score:
243             energy_icon = Energy(random.randint(50, WIDTH - 50), 1, -100)
244             energy_icons.append(energy_icon)
245             player.max_score += 30
246
247         # move and draw the EnergyIcon objects
```

```
146     for energy_icon in energy_icons:
147         energy_icon.move()
148         energy_icon.draw(WIN)
149         if energy_icon.collission(player):
150             player.power_level += 1
151             # Kiểm tra thời gian tồn tại của power-up, nếu hết hiệu lực thì
               huy nó
152             if time.time() - energy_icon.start_time >= POWER_UP_DURATION:
153                 player.power_level = 1
154                 energy_icons.remove(energy_icon)
155             elif time.time() - energy_icon.start_time >= 5:
156                 energy_icons.remove(energy_icon)
157
158     for event in pygame.event.get():
159         if event.type == pygame.QUIT:
160             quit()
161         elif event.type == pygame.KEYDOWN:
162             if event.key == pygame.K_ESCAPE:
163                 client_socket.close()
164                 return
165             elif event.type == pygame.MOUSEBUTTONDOWN:
166                 mouse_pos = pygame.mouse.get_pos()
167                 if chat_button.is_clicked(mouse_pos):
168                     chat_box() # Hiện thị giao diện chat khi nút chat được nhấn
169             elif event.type == pygame.KEYDOWN:
170                 if event.key == pygame.K_m:
171                     chat_box() # Hiện thị giao diện chat khi nhấn phím "M"
172
173
174     keys = pygame.key.get_pressed()
175     if keys[pygame.K_a] and player.x - player_vel > 0: # left
176         player.x -= player_vel
177     if keys[pygame.K_d] and player.x + player_vel + player.get_width() <
           WIDTH: # right
178         player.x += player_vel
179     if keys[pygame.K_w] and player.y - player_vel > 0: # up
180         player.y -= player_vel
181     if keys[pygame.K_s] and player.y + player_vel + player.get_height() + 15
           < HEIGHT: # down
182         player.y += player_vel
183     if keys[pygame.K_LEFT] and player.x - player_vel > 0: # left
184         player.x -= player_vel
185     if keys[pygame.K_RIGHT] and player.x + player_vel + player.get_width() <
           WIDTH: # right
186         player.x += player_vel
187     if keys[pygame.K_UP] and player.y - player_vel > 0: # up
188         player.y -= player_vel
189     if keys[pygame.K_DOWN] and player.y + player_vel + player.get_height() +
           15 < HEIGHT: # down
190         player.y += player_vel
191     if keys[pygame.K_SPACE]:
192         player.shoot()
```

```
193
194     for enemy in enemies[:]:
195         enemy.move(enemy_vel)
196         enemy.move_lasers(laser_vel, player)
197
198         if random.randrange(0, 2*FPS) == 1:
199             enemy.shoot(player)
200
201         # collisions of enemy and player
202         if collide(enemy, player):
203             player.health -= 10
204             enemies.remove(enemy)
205         elif enemy.y + enemy.get_height() > HEIGHT:
206             lives -= 1
207             enemies.remove(enemy)
208
209     player.move_lasers(-laser_vel, enemies, blood_icons, dif)
```

Hàm `main()` là hàm chính của trò chơi. Nó quản lý toàn bộ logic và hiển thị của trò chơi.

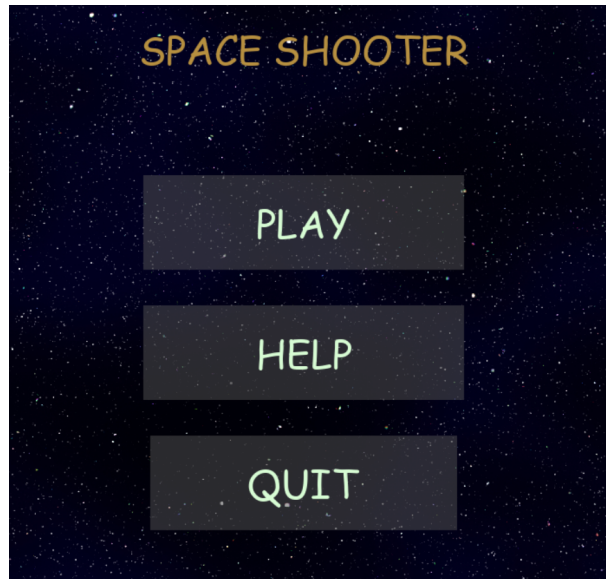
Trong hàm này, các biến được khởi tạo, giao diện chơi được vẽ, các sự kiện từ người chơi được xử lý và trạng thái của trò chơi được cập nhật.

Hàm `redraw_window()` vẽ lại toàn bộ giao diện trò chơi sau mỗi lần cập nhật. Nó vẽ người chơi, kẻ thù, các biểu tượng máu và năng lượng, và các thông số như số mạng, điểm số và cấp độ hiện tại.

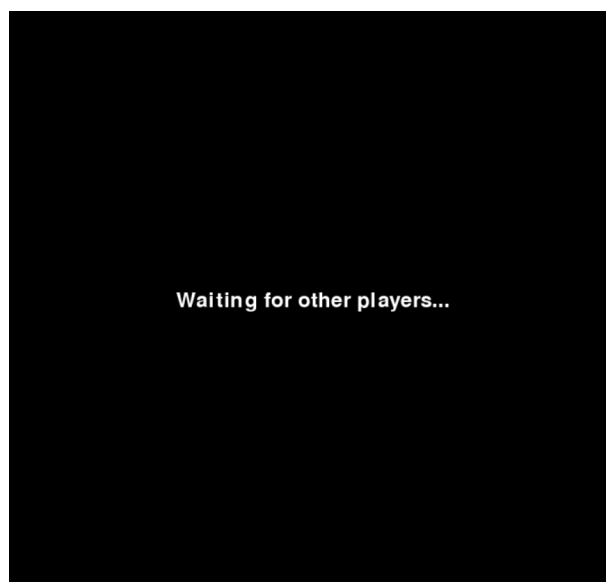
Hàm `delay_run_false()` được sử dụng để đặt biến `run` thành `False` sau một khoảng thời gian, thông báo rằng trò chơi đã kết thúc và cần thoát khỏi vòng lặp chính.

Hàm `send_game_status(status)` gửi trạng thái của trò chơi (thắng hoặc thua) đến máy chủ thông qua socket.

2.8 Hình ảnh về trò chơi



Màn hình menu



Màn hình chờ trước trận đấu



Màn hình khi vào trò chơi



Màn hình chat



Màn hình khi thua trò chơi



3 Tổng kết

3.1 Ưu điểm

- Trò chơi đáp ứng được các yêu cầu cơ bản của một tựa game bắn máy bay giải trí đơn giản dành cho 2 người chơi
- Có chức năng chat để giao tiếp giữa 2 người chơi
- Giao diện dễ nhìn và thân thiện với người chơi

3.2 Nhược điểm

- Đồ họa vẫn còn khá đơn giản và không bắt mắt để có thể thu hút nhiều người chơi
- Phần chat còn khá sơ sài, không đầy đủ chức năng so với các phần mềm chuyên về chat. Đồng thời khi chat thì trò chơi pause
- Cần phải khởi tạo server trước khi chạy trò chơi, chỉ có thể chơi trong mạng cục bộ và cần nhập địa chỉ ip để kết nối tới server

3.3 Kết luận

Sau một thời gian tìm hiểu đề án, thu thập các kiến thức liên quan và tham khảo cách làm một số nơi trên Internet, nhóm chúng em đã hoàn thành đề án game Bắn máy bay dành cho 2 người chơi. Mặc dù rất cố gắng, nhưng vẫn không tránh khỏi thiếu sót và hạn chế. Chúng em rất mong có được những ý kiến đánh giá, đóng góp của thầy để đề án thêm hoàn thiện.