TRƯỜNG ĐẠI HỌC SÀI GÒN
KHOA CÔNG NGHỆ THÔNG TIN

PHÁT TRIỂN PHẦN MỀM MÃ NGUỒN MỞ

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

GVHD: Từ Lãng 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

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:

```
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:

```
import pygame
import os
import time
import random
from pygame import mixer
import sys
import socket
import threading
import json
from threading import Lock
```

- **Khởi tạo màn hình game**

```
# initialize pygame, mixer and font
pygame.init()
pygame.font.init()
mixer.init()

WIDTH, HEIGHT = 700, 670
WIN = pygame.display.set_mode((WIDTH, HEIGHT))
```

```
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**

```
# Load images
RED_SPACE_SHIP = pygame.image.load(os.path.join("assets",
    "pixel_ship_red.png"))
GREEN_SPACE_SHIP = pygame.image.load(os.path.join("assets",
    "pixel_ship_green.png"))
BLUE_SPACE_SHIP = pygame.image.load(os.path.join("assets",
    "pixel_ship_blue.png"))
BLOOD_IMAGE = pygame.image.load(os.path.join("assets", "blood.png"))
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.

```
# Player player
YELLOW_SPACE_SHIP = pygame.image.load(os.path.join("assets",
    "pixel_ship.png"))

# Lasers
RED_LASER = pygame.image.load(os.path.join("assets", "pixel_laser.png"))
GREEN_LASER = pygame.image.load(os.path.join("assets", "pixel_laser.png"))
BLUE_LASER = pygame.image.load(os.path.join("assets", "pixel_laser.png"))
YELLOW_LASER = pygame.image.load(os.path.join("assets", "bullet.png"))
DOUBLE_LASER = pygame.image.load(os.path.join("assets",
    "double_bullet.png"))
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.

```
# Background
BG = pygame.transform.scale(pygame.image.load(os.path.join("assets",
    "background.png")), (WIDTH, HEIGHT))

# Background menu
BG_MENU = pygame.transform.scale(pygame.image.load(os.path.join("assets",
    "background_menu.jpg")), (WIDTH, HEIGHT))

```

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

Chèn các background trong lúc chơi trò chơi.

- **Chèn các âm thanh**

```
# load sound filed
shoot_sound = mixer.Sound(os.path.join("assets", "bullet_sound.wav"))
shoot_sound_enemy = mixer.Sound(os.path.join("assets",
    "bullet_sound_enemy.wav"))
get_score = mixer.Sound(os.path.join("assets", "get_score.wav"))
gameOver_sound = mixer.Sound(os.path.join("assets", "gameOver_sound.wav"))
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

```
class Laser:
    def __init__(self, x, y, img):
        self.x = x
        self.y = y
        self.img = img
        self.mask = pygame.mask.from_surface(self.img)

    def draw(self, window):
        window.blit(self.img, (self.x, self.y))

    def move(self, vel):
        self.y += vel

    def off_screen(self, height):
        return not(self.y <= height and self.y >= 0)

    def collision(self, obj):
        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

```
class Ship:
    COOLDOWN = 20

    def __init__(self, x, y, health=100):
        self.x = x
        self.y = y
        self.health = health
        self.ship_img = None
        self.laser_img = None
        self.lasers = []
        self.cool_down_counter = 0

    def draw(self, window):
        window.blit(self.ship_img, (self.x, self.y))
        for laser in self.lasers:
            laser.draw(window)

    def move_lasers(self, vel, obj):
        self.cooldown()
        for laser in self.lasers:
            laser.move(vel)
            if laser.off_screen(HEIGHT):
                self.lasers.remove(laser)
            elif laser.collision(obj):
                obj.health -= 10
                self.lasers.remove(laser)

    def cooldown(self):
        if self.cool_down_counter >= self.COOLDOWN:
            self.cool_down_counter = 0
        elif self.cool_down_counter > 0:
            self.cool_down_counter += 1

    def shoot(self):
        if self.cool_down_counter == 0:
            shoot_sound.play()
            laser = Laser(self.x + 20, self.y, self.laser_img)
            self.lasers.append(laser)
            self.cool_down_counter = 1

```

```
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 thoả 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 toạ độ x,y, thuộc tính img được gán bằng hình ảnh của thanh máu, một mask chứa surface của img vừa khơi 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 toạ 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 bổ 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 toạ 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 khơi 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 toạ 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

```
class Player(Ship):
    def __init__(self, x, y, health=100):
        super().__init__(x, y, health)
        self.ship_img = YELLOW_SPACE_SHIP
        self.laser_img = YELLOW_LASER
        self.mask = pygame.mask.from_surface(self.ship_img)
        self.max_health = health
        self.score = 0
        self.max_score = self.score + 30
        self.power_level = 1

    def spawn_blood_icon(self, blood_icons):
        if random.randint(1, 200) == 1:
            blood_icon = BloodIcon(random.randint(50, WIDTH - 50), -100)
            blood_icons.append(blood_icon)

    def spawn_energy_icon(self, energy_icons):
        while self.score == self.max_score:
            energy_icon = Energy(random.randint(50, WIDTH - 50), -100)
            energy_icons.append(energy_icon)
            self.max_score += 30

    def power_up(self, power_level):
        if power_level == 1:
            self.power_level = 1
        elif power_level == 2:
            self.power_level = 2
        elif power_level == 3:
            self.power_level = 3
        else:
            self.power_level = 1 # Neu gia tri khong hop le thi mac dinh la cap
                do 1

    def move_lasers(self, vel, objs, blood_icons, energy_icons):
        self.spawn_blood_icon(blood_icons)
        self.spawn_energy_icon(energy_icons)
        self.cooldown()
        for laser in self.lasers:
            laser.move(vel)
            if laser.off_screen(HEIGHT):
```

```
                self.lasers.remove(laser)
            else:
                for obj in objs:
                    if laser.collision(obj):
                        if self.power_level == 1:
                            self.laser_img = YELLOW_LASER
                            self.power_level = 1
                            for player_laser in self.lasers:
                                if collide(obj, player_laser):
                                    obj.health -= 20
                        elif self.power_level == 2:
                            self.laser_img = DOUBLE_LASER
                            self.power_level = 2
                            for player_laser in self.lasers:
                                if collide(obj, player_laser):
                                    obj.health -= 30
                        elif self.power_level >= 3:
                            self.laser_img = SUPER_LASER
                            self.power_level = 3
                            for player_laser in self.lasers:
                                if collide(obj, player_laser):
                                    obj.health -= 50
                        else:
                            self.power_level = 1
                        if obj.health <= 0:
                            objs.remove(obj)
                            self.score += 10
                            get_score.play()
                        if laser in self.lasers:
                            self.lasers.remove(laser)

    def draw(self, window):
        super().draw(window)
        self.healthbar(window)

    def healthbar(self, window):
        pygame.draw.rect(window, (255,0,0), (self.x, self.y +
            self.ship_img.get_height() + 10, self.ship_img.get_width(), 7))
        pygame.draw.rect(window, (0,255,0), (self.x, self.y +
            self.ship_img.get_height() + 10, self.ship_img.get_width() *
            (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 khơi 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

```
class Enemy(Ship):
    COLOR_MAP = {
                "red": (RED_SPACE_SHIP, RED_LASER),
                "green": (GREEN_SPACE_SHIP, GREEN_LASER),
                "blue": (BLUE_SPACE_SHIP, BLUE_LASER)
                }

    """
        Initializes an enemy ship object.

        Args:
        x (int): The x-coordinate of the enemy ship.
        y (int): The y-coordinate of the enemy ship.
        color (str): The color of the enemy ship.
        health (int): The health of the enemy ship.
    """
    def __init__(self, x, y, color, health=100):
        super().__init__(x, y, health)
        self.ship_img, self.laser_img = self.COLOR_MAP[color]
        self.mask = pygame.mask.from_surface(self.ship_img)
        self.cool_down_counter = 0

    # Moves the enemy ship down by the specified velocity.
    def move(self, vel):
        self.y += vel

```

```
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 +
                self.ship_img.get_height() + 10, bar_length, bar_height))
54          pygame.draw.rect(screen, (0, 255, 0), (self.x, self.y +
                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, gắn 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

```
class Server:
    def __init__(self, max_connections_callback=None):
        self.host = socket.gethostbyname(socket.gethostname())
        self.port = 5500
        self.max_connections = 2
        self.current_connections = 0
        self.max_connections_reached = False

        self.server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.server_socket.bind((self.host, self.port))
        self.server_socket.listen()

        print(f"Server dang lang nghe tren {self.host}:{self.port}")

        self.clients = []
        self.max_connections_callback = max_connections_callback

        self.accept_connections()

    def accept_connections(self):
        while True:
            if self.current_connections < self.max_connections:
                try:
                    client_socket, client_address = self.server_socket.accept()
                    print(f"Ket noi tu {client_address} duoc chap nhan")

                    self.clients.append(client_socket)
                    self.current_connections += 1

                    threading.Thread(target=self.handle_client,
                        args=[client_socket]).start()

                    # Kiem tra neu da du ket noi, gui thong bao den tat ca client
                    if self.current_connections == self.max_connections:
                        print("Da du ket noi, gui thong bao den tat ca client")
                        for client in self.clients:
                            try:
                                client.send("ready_to_start".encode())
                            except Exception as e:
                                print(f"Loi khi gui thong bao den client: {e}")
                except Exception as e:
                    print(f"Loi khi chap nhan ket noi: {e}")

```

```
            elif not self.max_connections_reached:
                print("Da dat den so luong ket noi toi da, tu choi ket noi moi.")
                self.max_connections_reached = True
                if self.max_connections_callback:
                    self.max_connections_callback() # Goi ham callback

    def handle_client(self, client_socket):
        try:
            while True:
                message = client_socket.recv(1024).decode()

                if not message:
                    break

                print(f"Nhan duoc tu {client_socket.getpeername()}: {message}")

                for client in self.clients:
                    if client is not client_socket and isinstance(client,
                        socket.socket):
                        try:
                            client.send(message.encode())
                        except Exception as e:
                            print(f"Loi khi gui thong diep den client: {e}")

        except ConnectionResetError:
            print(f"Client {client_socket.getpeername()} da dong ket noi mot
                cach bat ngo.")
        except Exception as e:
            print(f"Loi khi xu ly client {client_socket.getpeername()}: {e}")
        finally:
            client_socket.close() # Dong ket noi voi client
            self.clients.remove(client_socket)
            self.current_connections -= 1


def max_connections_reached_callback():
    # Thong bao cho main khi may chu dat den so luong ket noi toi da
    print("May chu da dat den so luong ket noi toi da.")

# Tao mot bien global de luu tru thong tin may chu
server_instance =
    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

```
def main_menu():
    title_font = pygame.font.SysFont("Arial", 50)
    run = True
    while run:
        WIN.blit(BG, (0,0))

        MENU_MOUSE_POS=pygame.mouse.get_pos()
        #game header text
        MENU_TEXT=get_font(45).render("SPACE SHOOTER",True,"#b68f40")
        MENU_RECT=MENU_TEXT.get_rect(center=(340,50))

        #buttons in main menu
        PLAY_BUTTON = Button(image=pygame.image.load(os.path.join("assets",
            "Play_Rect.png")), pos=(340, 250),
                             text_input="PLAY", font=get_font(45),
                                 base_color="#d7fcd4", hovering_color="Green")
        HELP_BUTTON = Button(image=pygame.image.load(os.path.join("assets",
            "Help_Rect.png")), pos=(340, 400),
                             text_input="HELP", font=get_font(45),
                                 base_color="#d7fcd4", hovering_color="Green")
        QUIT_BUTTON = Button(image=pygame.image.load(os.path.join("assets",
            "Quit_Rect.png")), pos=(340, 550),
                             text_input="QUIT", font=get_font(45),
                                 base_color="#d7fcd4", hovering_color="Green")

        WIN.blit(MENU_TEXT, MENU_RECT)

        for button in [PLAY_BUTTON, HELP_BUTTON, QUIT_BUTTON]:
            button.changeColor(MENU_MOUSE_POS)
            button.update(WIN)

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                run = False
                # pygame.quit()
                # sys.exit()
```

```
            if event.type == pygame.MOUSEBUTTONDOWN:
                if PLAY_BUTTON.checkForInput(MENU_MOUSE_POS):
                    play()
                if HELP_BUTTON.checkForInput(MENU_MOUSE_POS):
                    help()
                if QUIT_BUTTON.checkForInput(MENU_MOUSE_POS):
                    pygame.quit()
                    sys.exit()

        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

```
#mode help in main menu
def help():
    while True:
        HELP_MOUSE_POS=pygame.mouse.get_pos()
        WIN.fill("black")
        str='''Players must destroy all enemies on their flight path to pass
            different levels. Players use a keyboard with keys W, A, D, S to
            move the plane, use SPACE to shoot bullets. Players can also use
            helpful items such as bombs, missiles, or energy to help them
            destroy opponents faster'''
        blit_text(WIN,str,(20,100),get_font(25))
        # WIN.blit(HELP_TEXT,HELP_RECT)

        HELP_BACK = Button(image=None, pos=(340, 500), text_input="BACK",
            font=get_font(40), base_color="White", hovering_color="Green")

        HELP_BACK.changeColor(HELP_MOUSE_POS)
        HELP_BACK.update(WIN)

        for event in pygame.event.get():
```

```
            if event.type==pygame.QUIT:
                pygame.quit()
                sys.exit()
            if event.type==pygame.MOUSEBUTTONDOWN:
                if HELP_BACK.checkForInput(HELP_MOUSE_POS):
                    main_menu()

        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

```
# mode play in main menu
connected = threading.Event()
ready_to_start = threading.Event()
client_socket = None

def reset():
    reset_game_state()
    if client_socket is not None:
        try:
```

```
            client_socket.close()
            print("Socket closed successfully.")
        except Exception as e:
            print(f"Error closing socket: {e}")

def connect_to_server():
    global client_socket
    try:
        client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        host = socket.gethostbyname(socket.gethostname())
        port = 5500
        client_socket.connect((host, port))
        connected.set()
        print("Connected to server successfully.")
    except Exception as e:
        print(f"Loi khi ket noi toi may chu: {e}")
        client_socket = None

def ready_from_server():
    global client_socket
    while True:
        if ready_to_start.is_set():
            break
        try:
            if client_socket is not None:
                if connected.is_set():
                    # Nhan thong diep tu may chu
                    message = client_socket.recv(1024).decode()
                    if message == "ready_to_start":
                        ready_to_start.set()
                else:
                    # Dung mot thoi gian ngan de cho socket co co hoi ket noi
                    time.sleep(0.1)
        except ConnectionResetError as e:
            # print(f"Ket noi da bi dong boi may chu: {e}")
            break
        except Exception as e:
            # print(f"Loi khi nhan thong diep tu may chu: {e}")
            break

def reset_game_state():
    global connected, ready_to_start
    connected.clear()
    ready_to_start.clear() # Dat lai trang thai "ready_to_start"

def play():
    global connected, ready_to_start, client_thread

    # Reset game state
    reset_game_state()

    waiting_text = "Waiting for other players..."
```

```
62        start_time = time.time()
63        display_start_message = False
64        connect = False
65
66        # Create Pygame screen
67        screen = pygame.display.set_mode((WIDTH, HEIGHT))
68
69        running = True
70        while running:
71            for event in pygame.event.get():
72                if event.type == pygame.QUIT:
73                    pygame.quit()
74                    sys.exit()
75                elif event.type == pygame.KEYDOWN:
76                    if event.key == pygame.K_ESCAPE:
77                        if connect:
78                            try:
79                                client_socket.close()
80                            except Exception as e:
81                                print(f"Error closing socket: {e}")
82                        running = False
83                    elif event.key == pygame.K_p:
84                        if not connect:
85                            connect_to_server()
86                            connected.wait(1) # Wait until connected, with a timeout
                                 of 5 seconds
87                            if not connected.is_set():
88                                screen.fill((0, 0, 0))
89                                font = pygame.font.Font(None, 36)
90                                text = font.render("Failed to connect to the server.",
                                     True, (255, 255, 255))
91                                text_rect = text.get_rect(center=(WIDTH // 2, HEIGHT
                                     // 2))
92                                screen.blit(text, text_rect)
93                                pygame.display.flip()
94
95                                # Wait for 2 seconds to show the message before
                                     returning
96                                wait_time = 2000 # 2000 milliseconds = 2 seconds
97                                start_wait = pygame.time.get_ticks()
98                                while pygame.time.get_ticks() - start_wait < wait_time:
99                                    for event in pygame.event.get():
100                                       if event.type == pygame.QUIT:
101                                           pygame.quit()
102                                           sys.exit()
103                                       elif event.type == pygame.KEYDOWN:
104                                           if event.key == pygame.K_ESCAPE:
105                                               return
106                               return # Return if unable to connect to the server
107                           client_thread = threading.Thread(target=ready_from_server)
108                           client_thread.start()
109                       connect = True
```

```
110
111         if not connect:
112             screen.fill((0, 0, 0))
113             font = pygame.font.Font(None, 36)
114             text = font.render("Press 'P' to play", True, (255, 255, 255))
115             text_rect = text.get_rect(center=(WIDTH // 2, HEIGHT // 2))
116             screen.blit(text, text_rect)
117         else:
118             screen.fill((0, 0, 0))
119             font = pygame.font.Font(None, 36)
120             text = font.render(waiting_text, True, (255, 255, 255))
121             text_rect = text.get_rect(center=(WIDTH // 2, HEIGHT // 2))
122             screen.blit(text, text_rect)
123
124             if ready_to_start.is_set() and not display_start_message:
125                 display_start_message = True
126                 waiting_text = "Both players are ready, let's start the game..."
127
128         pygame.display.flip()
129
130         if display_start_message:
131             elapsed_time = time.time() - start_time
132             if elapsed_time >= 1:
133                 running = False
134                 main()
135
136     # Reset game state if the loop breaks
137     reset_game_state()
138
139
140 # Khoi tao luong cho client
141 client_thread = threading.Thread(target=ready_from_server)
142 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

```
#Chat
class ChatButton:
    def __init__(self, image, pos):
        self.image = image
        self.pos = pos

    def draw(self, win):
        win.blit(self.image, self.pos)

    def is_clicked(self, pos):
        x, y = pos
        button_rect = self.image.get_rect(topleft=self.pos)
        return button_rect.collidepoint(x, y)

# Tao mot doi tuong lock
chat_log_lock = threading.Lock()

# Trong ham send_message:
def send_message(client_socket, input_text):
    try:
        client_socket.sendall(input_text.encode('utf-8'))
        # Them tin nhan vao chat_log
        with chat_log_lock:
            chat_log.append("You: "+input_text)
    except Exception as e:
        print("Error sending message:", e)

# Trong ham receive_messages:
def receive_messages(client_socket, chat_log):
    try:
        while True:
            message = client_socket.recv(1024).decode('utf-8')
            if message:
                with chat_log_lock:
                    # Them tin nhan tu client vao chat log
                    chat_log.append(message)
            else:
                break
    except Exception as e:
        print("Error receiving message:", e)
    finally:
        client_socket.close()
```

```
43
44  chat_log = [] # Danh sach de luu tin nhan trong cuoc tro chuyen
45
46  def chat_box():
47      run_chat = True
48      chat_font = pygame.font.SysFont("Arial", 14)
49      input_font = pygame.font.SysFont("Arial", 14)
50
51      current_client_address = client_socket.getpeername()
52
53      input_text = "" # Bien de luu noi dung cua o nhap lieu
54
55      # Kich thuoc cua chat menu
56      chat_menu_width = WIDTH // 2
57      chat_menu_height = HEIGHT // 2
58
59      # Vi tri cua chat menu
60      chat_menu_x = 10
61      chat_menu_y = 150
62
63      # Vi tri cua nut tat
64      close_button_x = chat_menu_x + chat_menu_width - 30
65      close_button_y = chat_menu_y + 10
66
67      while run_chat:
68          for event in pygame.event.get():
69              if event.type == pygame.QUIT:
70                  pygame.quit() # Thoat khoi tro choi khi cua so bi dong
71                  sys.exit()
72              if event.type == pygame.KEYDOWN:
73                  if event.key == pygame.K_RETURN: # Xu ly khi nguoi dung nhan Enter
74                      if input_text: # Dam bao rang o nhap lieu khong trong
75                          send_message(client_socket, input_text) # Gui tin nhan den
                                may chu
76                          input_text = "" # Dat lai noi dung cua o nhap lieu
77                  elif event.key == pygame.K_BACKSPACE: # Xu ly khi nguoi dung nhan
                        phim Backspace
78                      input_text = input_text[:-1] # Xoa ky tu cuoi cung trong o
                            nhap lieu
79                  else:
80                      # Them ky tu vao o nhap lieu
81                      input_text += event.unicode if event.unicode.isprintable()
                            else ""
82
83              elif event.type == pygame.MOUSEBUTTONDOWN: # Xu ly khi nguoi dung
                    nhan chuot
84                  mouse_pos = pygame.mouse.get_pos()
85                  # Kiem tra neu nguoi dung nhan vao nut tat
86                  if close_button_x <= mouse_pos[0] <= close_button_x + 20 and \
87                     close_button_y <= mouse_pos[1] <= close_button_y + 20:
88                      run_chat = False # Dong cua so chat menu
89
```

```
90            # Ve nen trong suot cho chat menu
91            chat_menu_surface = pygame.Surface((chat_menu_width, chat_menu_height),
                  pygame.SRCALPHA)
92            pygame.draw.rect(chat_menu_surface, (0, 0, 0, 100), (0, 0,
                  chat_menu_width, chat_menu_height), border_radius=10)
93            WIN.blit(chat_menu_surface, (chat_menu_x, chat_menu_y))
94
95            # Ve giao dien chat
96            y_offset = chat_menu_y + 10
97            for message in chat_log:
98                # Kiem tra xem tin nhan co den tu client khac khong
99                if message.startswith("You:") and message[5:] !=
                      current_client_address:
100                   message = "You: " + message[5:]
101               else:
102                   message = "Other: " + message
103
104               text_surface = chat_font.render(message, True, (255, 255, 255)) #
                      Render tin nhan
105               WIN.blit(text_surface, (chat_menu_x + 10, y_offset)) # Ve tin nhan
                      len man hinh
106               y_offset += text_surface.get_height() + 5 # Tang y_offset de ve tin
                      nhan tiep theo
107
108           # Ve o nhap lieu
109           pygame.draw.rect(WIN, (255, 255, 255), (chat_menu_x + 5, chat_menu_y +
                  chat_menu_height - 35, chat_menu_width - 10, 30), 2) # Ve khung o
                  nhap lieu
110           input_surface = input_font.render(input_text, True, (255, 255, 255)) #
                  Render noi dung o nhap lieu
111           WIN.blit(input_surface, (chat_menu_x + 10, chat_menu_y +
                  chat_menu_height - 30)) # Ve noi dung o nhap lieu len man hinh
112
113           # Ve nut tat
114           pygame.draw.rect(WIN, (255, 0, 0), (close_button_x, close_button_y, 20,
                  20))
115           close_text = input_font.render("X", True, (255, 255, 255))
116           WIN.blit(close_text, (close_button_x + 4, close_button_y))
117
118           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

```
def main():
    global run
    run = True
    level = 0
    enemies = []
    explosions = []
    player_vel = 6
    laser_vel = 7
    player = Player(300, 550)
    dif = 1 # health bar of enemise reduce rely on this variable
    blood_icons = []
    energy_icons = []
    bg_running = 0

    main_font = pygame.font.SysFont("Arial", 30)
    lost_font = pygame.font.SysFont("Arial", 40)
    win_font = pygame.font.SysFont("Arial", 40)

    # get the level from user
    FPS = 50
    wave_length = 5
    enemy_vel = 1
    lives = 6
    dif = 4
    bg_running = 0.50
    clock = pygame.time.Clock()

    global lost, win, status_sent

    lost = False

    win = False

    status_sent = False

```

```
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",
            "Chat_Rect.png")) # Load hinh anh cua nut chat
40      chat_button = ChatButton(CHAT_BUTTON_IMAGE, CHAT_BUTTON_POS)
41
42      # Khoi tao thread de nhan tin nhan
43      receive_thread = threading.Thread(target=receive_messages,
            args=(client_socket, chat_log))
44      receive_thread.daemon = True # Dat thread nhan tin nhan thanh daemon de no
            tu dong khi chuong trinh chinh ket thuc
45      receive_thread.start()
46
47      def redraw_window():
48          WIN.blit(BG, (0,0))
49          bg.update(bg_running) # running background
50          # draw text
51          lives_label = main_font.render(f"Lives: {lives}", 1, (255,255,255))
52          level_label = main_font.render(f"Level: {level}", 1, (255,255,255))
53          score_label = main_font.render(f"Score: {player.score}", 1,
                (255,255,255))
54
55          chat_button.draw(WIN)
56
57          WIN.blit(lives_label, (10, 10))
58          WIN.blit(level_label, (WIDTH - level_label.get_width() - 10, 10))
59          WIN.blit(score_label, (WIDTH/2 - score_label.get_width()/2, 10))
60
61          for enemy in enemies:
62              enemy.draw(WIN)
63
64          for laser in player.lasers:
65              laser.draw(WIN)
66
67          for blood_icon in blood_icons:
68              blood_icon.draw(WIN)
69
70          for energy_icon in energy_icons:
71              energy_icon.draw(WIN)
72
73          player.draw(WIN)
74
75          for enemy in enemies:
76              enemy.draw(WIN)
77              enemy.healthbar(WIN) # draw health bar for each enemy
78
79          if lost:
80              lost_label = lost_font.render("Game Over!", 1, (255,255,255))
81              if lost_label:
82                  gameOver_sound.play()
83              WIN.blit(lost_label, (WIDTH/2 - lost_label.get_width()/2, 350))
```

```
84
85          if win:
86              win_label = win_font.render("You Win!", 1, (255,255,255))
87              if win_label:
88                  gameWin_sound.play()
89              WIN.blit(win_label, (WIDTH/2 - win_label.get_width()/2, 350))
90
91          pygame.display.update()
92
93      def delay_run_false():
94          global run
95          run = False
96
97      def send_game_status(status):
98          message = json.dumps({'status': status})
99          client_socket.sendall(message.encode('utf-8'))
100
101     def receive_game_status():
102         while True:
103             try:
104                 data = client_socket.recv(1024).decode('utf-8')
105                 if data:
106                     status = json.loads(data).get('status')
107                     if status == 'lost':
108                         global win
109                         win = True
110                     elif status == 'win':
111                         global lost
112                         lost = True
113             except Exception as e:
114                 print(f"Error receiving game status: {e}")
115                 break
116
117     status_thread = threading.Thread(target=receive_game_status)
118     status_thread.daemon = True
119     status_thread.start()
120
121     while run:
122         clock.tick(FPS)
123         redraw_window()
124
125         if lives <= 0 or player.health <= 0:
126             lost = True
127
128         if player.score == 500:
129             win = True
130
131         if (win or lost) and not status_sent:
132             if win:
133                 send_game_status('win') # Gui thong tin chien thang toi server
134             if lost:
135                 send_game_status('lost') # Gui thong tin thua cuoc toi server
```

```
136
137             threading.Timer(2, delay_run_false).start() # Tri hoan viec dat run
                    thanh False
138             threading.Timer(2, reset).start() # Tri hoan viec reset
139             status_sent = True # Dat co de ngan chan viec gui nhieu lan
140
141
142         if len(enemies) == 0:
143             level += 1
144             wave_length += 5
145             for i in range(wave_length):
146                 enemy = Enemy(random.randrange(50, WIDTH-100),
                        random.randrange(-1500, -100), random.choice(["red", "blue",
                        "green"]))
147                 enemies.append(enemy)
148
149         # spawn new BloodIcon objects randomly at the top of the screen
150         if random.randint(0, 1300) < 1:
151             blood_icon = BloodIcon(random.randint(50, WIDTH-50), 0)
152             blood_icons.append(blood_icon)
153
154         # move and draw the BloodIcon objects
155         for blood_icon in blood_icons:
156             blood_icon.move()
157             blood_icon.draw(WIN)
158             if blood_icon.collision(player):
159                 player.health += 20
160                 if player.health > player.max_health:
161                     player.health = player.max_health
162                 blood_icons.remove(blood_icon)
163
164         # spawn new EnergyIcon objects randomly at the top of the screen
165         while player.score == player.max_score:
166             energy_icon = Energy(random.randint(50, WIDTH - 50), 1, -100)
167             energy_icons.append(energy_icon)
168             player.max_score += 30
169
170         # move and draw the EnergyIcon objects
171         for energy_icon in energy_icons:
172             energy_icon.move()
173             energy_icon.draw(WIN)
174             if energy_icon.collision(player):
175                 player.power_level += 1
176                 # Kiem tra thoi gian ton tai cua power-up, neu het hieu luc thi
                        huy no
177                 if time.time() - energy_icon.start_time >= POWER_UP_DURATION:
178                     player.power_level = 1
179                 energy_icons.remove(energy_icon)
180             elif time.time() - energy_icon.start_time >= 5:
181                 energy_icons.remove(energy_icon)
182
183         for event in pygame.event.get():
```

```
            if event.type == pygame.QUIT:
                quit()
            elif event.type == pygame.KEYDOWN:
                if event.key == pygame.K_ESCAPE:
                    client_socket.close()
                    return
            elif event.type == pygame.MOUSEBUTTONDOWN:
                mouse_pos = pygame.mouse.get_pos()
                if chat_button.is_clicked(mouse_pos):
                    chat_box() # Hien thi giao dien chat khi nut chat duoc nhan
            elif event.type == pygame.KEYDOWN:
                if event.key == pygame.K_m:
                    chat_box() # Hien thi giao dien chat khi nhan phim "M"

        keys = pygame.key.get_pressed()
        if keys[pygame.K_a] and player.x - player_vel > 0: # left
            player.x -= player_vel
        if keys[pygame.K_d] and player.x + player_vel + player.get_width() <
             WIDTH: # right
            player.x += player_vel
        if keys[pygame.K_w] and player.y - player_vel > 0: # up
            player.y -= player_vel
        if keys[pygame.K_s] and player.y + player_vel + player.get_height() + 15
             < HEIGHT: # down
            player.y += player_vel
        if keys[pygame.K_LEFT] and player.x - player_vel > 0: # left
            player.x -= player_vel
        if keys[pygame.K_RIGHT] and player.x + player_vel + player.get_width() <
             WIDTH: # right
            player.x += player_vel
        if keys[pygame.K_UP] and player.y - player_vel > 0: # up
            player.y -= player_vel
        if keys[pygame.K_DOWN] and player.y + player_vel + player.get_height() +
             15 < HEIGHT: # down
            player.y += player_vel
        if keys[pygame.K_SPACE]:
            player.shoot()

        for enemy in enemies[:]:
            enemy.move(enemy_vel)
            enemy.move_lasers(laser_vel, player)

            if random.randrange(0, 2*FPS) == 1:
                enemy.shoot(player)

            # collisions of enemy and player
            if collide(enemy, player):
                player.health -= 10
                enemies.remove(enemy)
            elif enemy.y + enemy.get_height() > HEIGHT:
                lives -= 1
                enemies.remove(enemy)
```

```
232
233             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.

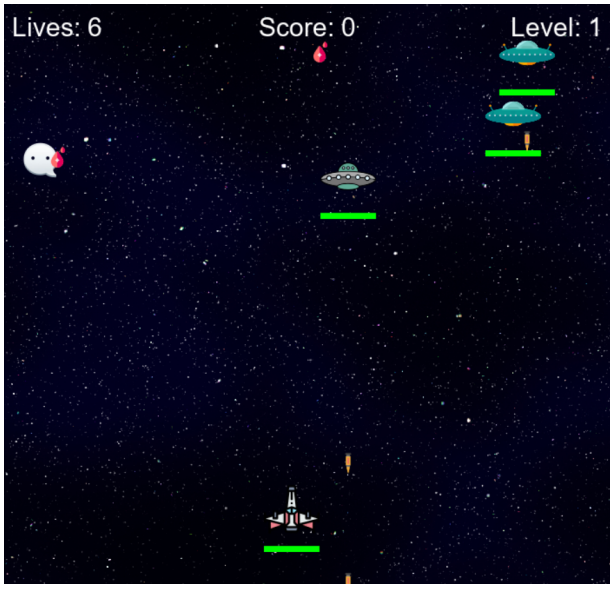
Hàm `receive_game_status()` lắng nghe thông điệp từ máy chủ để xác định trạng thái của trò chơi (thắng hoặc thua).

Trong hàm main(), có các lời gọi để bắt đầu các luồng như receive_thread và status_thread, để lắng nghe tin nhắn và trạng thái của trò chơi từ máy chủ một cách song song với việc chạy trò chơi chính.

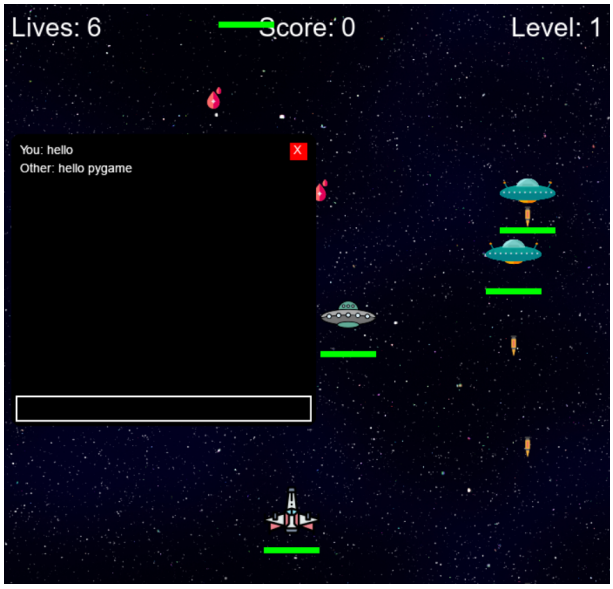
2.8 Hình ảnh về trò chơi

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 đồ án, thu thập các kiến thức liên quan và tham khảo cách làm một số nơi trên Internet, nhóm chúng em đã hoàn thành đồ 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 đồ án thêm hoàn thiện.