UỶ BAN NHÂN DÂN

THÀNH PHỐ HỒ CHÍ MINH

**TRƯỜNG ĐẠI HỌC SÀI GÒN**

A blue circle with text

Description automatically generated

**BÁO CÁO TIỂU LUẬN**

**NGÔN NGỮ LẬP TRÌNH PYTHON**

**GAME BẮN SÚNG 2D BẰNG THƯ VIỆN PYGAME**

**Thông tin thành viên:**

Nguyễn Cảnh Đức – 3120410139

**Giảng viên:** Nguyễn Trung Tín

**Thành phố Hồ Chí Minh, 12/2024**

**Mục lục**

[Lời mở đầu 4](#_Toc184155053)

[1. Cài đặt và môi trường phát triển 5](#_Toc184155054)

[Cài đặt ngôn ngữ python 5](#_Toc184155055)

[Cài đặt thư viện Pygame 7](#_Toc184155056)

[2. Cơ sở lý thuyết 8](#_Toc184155057)

[Ngôn ngữ lập trình Python 8](#_Toc184155058)

[Đặc điểm nổi bật của Python 8](#_Toc184155059)

[Cú pháp cơ bản của Python 8](#_Toc184155060)

[Kiểu dữ liệu cơ bản 9](#_Toc184155061)

[Hàm 10](#_Toc184155062)

[Xử lý ngoại lệ 11](#_Toc184155063)

[Nhập / Xuất dữ liệu 11](#_Toc184155064)

[Thư viện và module 12](#_Toc184155065)

[Ứng dụng của Python 13](#_Toc184155066)

[Pygame 14](#_Toc184155067)

[Lịch sử phát triển và mục đích ra đời của Pygame. 14](#_Toc184155068)

[Lịch sử phát triển của Pygame 14](#_Toc184155069)

[Mục đích ra đời của Pygame 14](#_Toc184155070)

[Ứng dụng của Pygame 14](#_Toc184155071)

[Các thành phần chính của pygame 16](#_Toc184155072)

[Cửa sổ và bề mặt 16](#_Toc184155073)

[Vẽ đồ họa 18](#_Toc184155074)

[Âm thanh 18](#_Toc184155075)

[Thời gian 18](#_Toc184155076)

[Quản lý sprites 20](#_Toc184155077)

[Phát hiện va chạm 23](#_Toc184155078)

[Đầu vào từ bàn phím và chuột 27](#_Toc184155079)

[Bàn phím 27](#_Toc184155080)

[Chuột 32](#_Toc184155081)

[Hệ thống phông chữ 35](#_Toc184155082)

[Hiệu ứng hình ảnh 38](#_Toc184155083)

[3. Quá trình xây dựng và triển khai 53](#_Toc184155084)

[Giới thiệu 53](#_Toc184155085)

[Tính năng 53](#_Toc184155086)

[Triển khai 53](#_Toc184155087)

[Source code 68](#_Toc184155088)

[Kết luận 69](#_Toc184155089)

[Tài liệu tham khảo 70](#_Toc184155090)

# Lời mở đầu

Trong bối cảnh công nghệ ngày càng phát triển và đa dạng, lĩnh vực phát triển trò chơi điện tử trở thành một ngành nghề đầy tiềm năng và hấp dẫn, không chỉ thu hút các nhà phát triển chuyên nghiệp mà còn cả những người yêu thích lập trình, bao gồm sinh viên ngành công nghệ thông tin. Thế giới trò chơi điện tử không chỉ đơn thuần mang lại giá trị giải trí mà còn là môi trường thử nghiệm, sáng tạo, và ứng dụng những tiến bộ khoa học công nghệ vào thực tế. Đối với sinh viên, quá trình học tập và nghiên cứu về phát triển trò chơi mang lại nhiều lợi ích thiết thực, như cải thiện tư duy logic, kỹ năng lập trình, và khả năng làm việc nhóm.

Pygame là một thư viện mã nguồn mở dựa trên Python, cung cấp một bộ công cụ mạnh mẽ và dễ sử dụng để phát triển các trò chơi 2D. Được thiết kế với mục tiêu giúp người dùng nhanh chóng làm quen và tạo ra sản phẩm, Pygame đã trở thành công cụ phổ biến không chỉ dành cho người mới bắt đầu mà còn cho cả những người đã có kinh nghiệm muốn tạo ra các sản phẩm thử nghiệm. Với các tính năng cơ bản như xử lý đồ họa, âm thanh, và tương tác người chơi, Pygame là lựa chọn tuyệt vời để bắt đầu hành trình khám phá thế giới lập trình game.

Báo cáo này nhằm mục đích tìm hiểu cơ bản về thư viện Pygame, khám phá các tính năng chính và áp dụng vào việc xây dựng các trò chơi đơn giản. Qua đó, nhóm hy vọng sẽ mang đến cho người đọc một cái nhìn tổng quan về cách thức hoạt động của Pygame, cũng như khả năng ứng dụng của nó trong lĩnh vực phát triển trò chơi. Đây cũng là cơ hội để bản thân em rèn luyện kỹ năng lập trình và ứng dụng những kiến thức đã học vào thực tiễn.

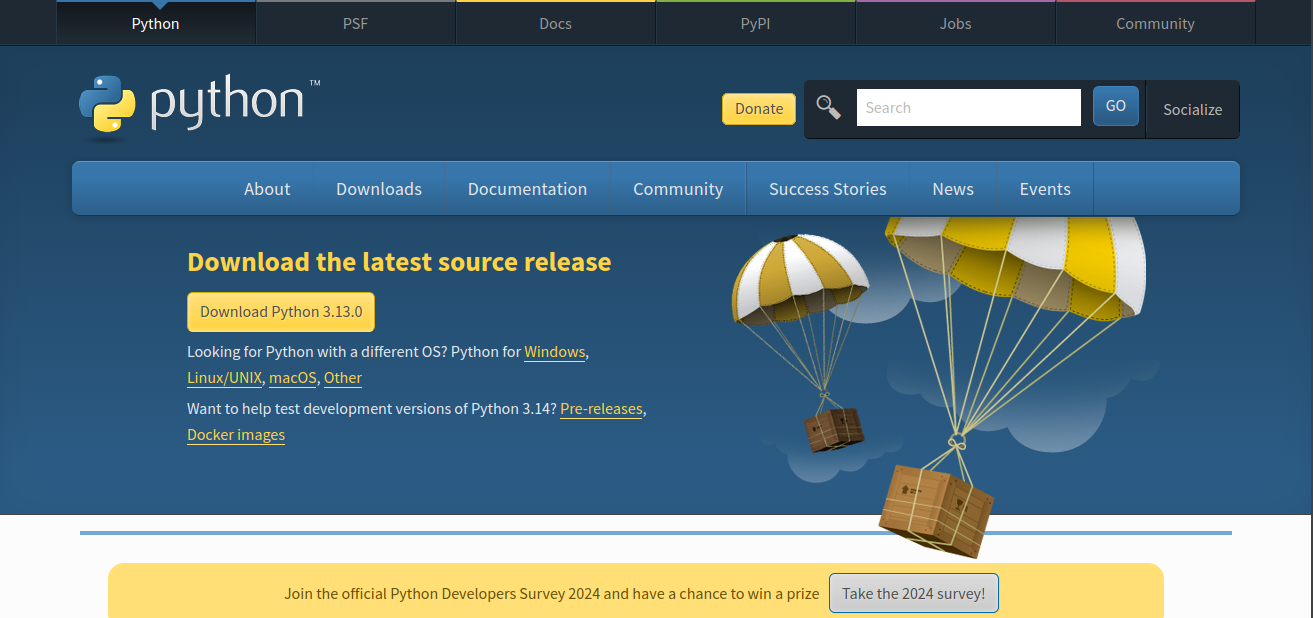
Trong quá trình tìm hiểu và thực hiện báo cáo này, do còn hạn chế về thời gian cũng như kinh nghiệm, nhóm khó tránh khỏi những thiếu sót nhất định. Em kính mong thầy có thể góp ý, chỉ bảo để báo cáo của em được hoàn thiện hơn. Em xin chân thành cảm ơn sự quan tâm và hỗ trợ từ thầy trong quá trình học tập và nghiên cứu.

# Cài đặt và môi trường phát triển

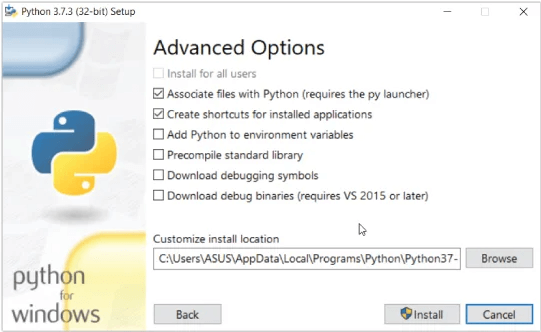
## Cài đặt ngôn ngữ python

**Hệ điều hành Window**

* Truy cập vào trang web chính thức của python: https://www.python.org/downloads/, chọn phiên bản phù hợp và tải xuống



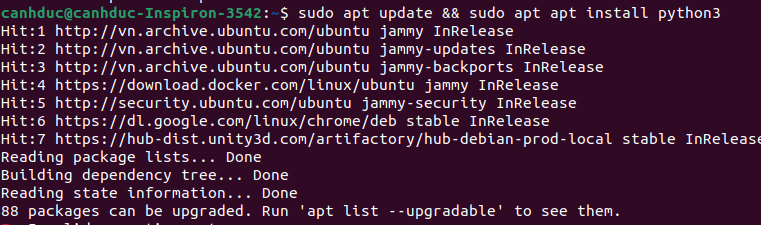
* Sau khi tải thành công, chạy chương trình và làm theo các bước để cài đặt



**Hệ điều hành Linux (Ubuntu)**

* Mở terminal và chạy dòng lệnh sau

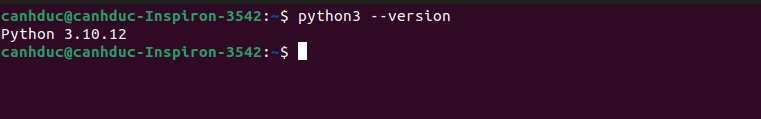
sudo apt update && sudo apt install python3



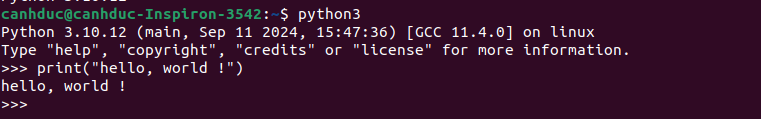
**Kiểm tra phiên bản python**

Mở cửa sổ terminal và chạy dòng lệnh: python3 –version

Nếu màn hình trả ra kết quả số phiên bản của python nghĩa là bạn đã cài đặt thành công

**Chạy thử chương trình đơn giản**

* Mở terminal hoặc Command Prompt.
* Gõ python hoặc python3 để vào Python interpreter.
* Nhập dòng lệnh và nhấn Enter



**1 số IDE và editor tham khảo để lập trình ngôn ngữ python**

* **PyCharm:** <https://www.jetbrains.com/pycharm/download/>
* **VSCode:** <https://code.visualstudio.com/>
* **Jupyter:** <https://jupyter.org/install>
* **Vim:** https://www.vim.org/download.php

## Cài đặt thư viện Pygame

* Mở cửa sổ terminal hoặc command prompt
* Chạy dòng lệnh sau để cài đặt

python3 -m pip install -U pygame --user

* Sau khi chạy dòng lệnh cài đặt, chạy thử 1 vài example của pygame để đảm bảo thư viện đã được cài đặt thành công

python3 -m pygame.examples.aliens



# Cơ sở lý thuyết

## Ngôn ngữ lập trình Python

Python là một ngôn ngữ lập trình bậc cao, linh hoạt và dễ học, được phát triển bởi Guido van Rossum vào cuối những năm 1980 và phát hành lần đầu vào năm 1991. Python đã trở thành một trong những ngôn ngữ lập trình phổ biến nhất trên thế giới nhờ vào cú pháp đơn giản, dễ đọc và khả năng ứng dụng rộng rãi.

### Đặc điểm nổi bật của Python

* Dễ đọc, dễ học: Cú pháp của Python rất trực quan và gần gũi với ngôn ngữ tự nhiên, giúp lập trình viên dễ dàng học và sử dụng.
* Ngôn ngữ thông dịch (interpreted): Python là ngôn ngữ thông dịch, nghĩa là mã nguồn được thực thi trực tiếp mà không cần phải biên dịch thành mã máy.
* Đa nền tảng: Python có thể chạy trên nhiều hệ điều hành khác nhau, như Windows, macOS, và Linux.
* Thư viện phong phú: Python có một hệ sinh thái thư viện rộng lớn, hỗ trợ từ phát triển web (Django, Flask), phân tích dữ liệu (Pandas, NumPy), học máy (TensorFlow, scikit-learn) đến xử lý hình ảnh (OpenCV, PIL) và nhiều lĩnh vực khác.

### Cú pháp cơ bản của Python

* Cách viết lệnh cơ bản

Python sử dụng **dấu xuống dòng** để kết thúc một câu lệnh

Vd:

x = 10

print(x)

Sử dụng dấu gạch chéo ngược (\) để nối câu lệnh dài

Vd:

total = 10 + 20 + 30 + \

40 + 50

* Căn lề

Python sử dụng **khoảng trắng thụt lề** để biểu thị khối lệnh, thay vì dấu ngoặc nhọn {}.

Quy tắc: Mỗi khối lệnh phải thụt lề **đều nhau** (thường là 4 khoảng trắng)

Vd:

if x > 0:

print("x là số dương")

else:

print("x là số không âm")

* Chú thích

**Chú thích một dòng:** Sử dụng ký hiệu #

Vd:

# Đây là một dòng chú thích

print("Hello, Python!") # In ra dòng chữ

Vd:

**Chú thích nhiều dòng:** Sử dụng chuỗi nhiều dòng (triple quotes ''' hoặc """)

"""

Đây là chú thích

nhiều dòng.

"""

* Khai báo và sử dụng biến

Không cần khai báo kiểu dữ liệu trước, Python tự động xác định dựa vào giá trị gán

Vd:

x = 5 # Số nguyên

y = 3.14 # Số thực

z = "Hello" # Chuỗi

* Toán tử cơ bản

**Toán tử số học:** +, -, \*, /, // (chia lấy nguyên), % (lấy dư), \*\* (lũy thừa).

**Toán tử so sánh:** ==, !=, >, <, >=, <=.

**Toán tử logic:** and, or, not.

### Kiểu dữ liệu cơ bản

Python có các kiểu dữ liệu cơ bản sau:

* Kiểu số: int, float, complex, …
* Chuỗi: string
* Danh sách: list
* Bộ: tuple
* Tập hợp: set
* Từ điển: dictionary
* Boolean

### Hàm

**Khái niệm**

Hàm là một khối mã được tổ chức thành một đơn vị để thực hiện một nhiệm vụ cụ thể.

Cách sử dụng: Hàm trong Python được định nghĩa bằng từ khóa def, theo sau là tên hàm và danh sách tham số (nếu có).

Vd:

def cong\_hai\_so(a, b):

return a + b

print(cong\_hai\_so(3, 5)) # Kết quả: 8

**Hàm ẩn danh (Lambda Function):**

Hàm Lambda là hàm không có tên, được định nghĩa bằng từ khóa lambda.

Vd:

cong = lambda x, y: x + y

print(cong(3, 5)) # Kết quả: 8

**Phạm vi biến trong hàm:**

Biến cục bộ (local): Chỉ tồn tại trong hàm.

Biến toàn cục (global): Có thể truy cập ở mọi nơi trong chương trình.

Vd:

x = 10

def thay\_doi():

global x

x = 20

thay\_doi()

print(x) # Kết quả: 20

## 

### Xử lý ngoại lệ

**Khái niệm**

Ngoại lệ (Exception) là lỗi xảy ra trong quá trình thực thi chương trình, khiến chương trình dừng đột ngột nếu không được xử lý.

**Cách sử dụng:**

Python sử dụng khối lệnh try...except để xử lý ngoại lệ.

Vd:

try:

result = 10 / 0

except ZeroDivisionError:

print("Không thể chia cho 0!")

### Nhập / Xuất dữ liệu

**Nhập dữ liệu từ người dùng**

Sử dụng hàm input() để nhận dữ liệu từ bàn phím.

Dữ liệu nhập vào luôn ở dạng chuỗi (str), cần chuyển đổi kiểu dữ liệu nếu cần.

Vd:

name = input("Nhập tên của bạn: ")

age = int(input("Nhập tuổi của bạn: ")) # Chuyển chuỗi thành số nguyên

print(f"Xin chào, {name}. Bạn {age} tuổi.")

**Xuất dữ liệu ra màn hình**

Sử dụng hàm print() để hiển thị dữ liệu.

Có thể truyền nhiều giá trị vào print(), phân cách bởi dấu phẩy.

Tùy chỉnh định dạng bằng chuỗi f-string hoặc phương thức .format().

Vd:

# Xuất thông tin cơ bản

print("Xin chào, Python!")

# Sử dụng f-string

name = "Minh"

age = 20

print(f"Tên: {name}, Tuổi: {age}")

# Sử dụng .format()

print("Tên: {}, Tuổi: {}".format(name, age))

### Thư viện và module

**Khái niệm**

**Thư viện (Library):** Một tập hợp các module được đóng gói sẵn, cung cấp các chức năng cụ thể giúp lập trình viên giải quyết các vấn đề thường gặp.

**Module:** Một tệp Python chứa mã nguồn như các biến, hàm, lớp hoặc thậm chí các module khác.

**Cách sử dụng**

* Sử dụng từ khóa import để nhập một module vào chương trình.
* Có thể nhập toàn bộ module, một phần module, hoặc đổi tên module.

Vd:

# Nhập toàn bộ module

import math

print(math.sqrt(16)) # In ra 4.0

# Nhập một phần module

from math import pi, sin

print(pi) # In ra 3.141592653589793

print(sin(0)) # In ra 0.0

# Đổi tên module

import math as m

print(m.sqrt(25)) # In ra 5.0

**1 số module phổ biến:**

* math: hỗ trợ xử lý các phép toán
* random: hỗ trợ xử lý random
* os: tương tác với hệ điều hành

**Cách sử dụng thư viện:**

Các thư viện bên ngoài không có sẵn trong Python, cần cài đặt thông qua pip (Python Package Index).

Cú pháp:

pip install <tên\_thư\_viện>

**1 số thư viện phổ biến:**

* numpy: Hỗ trợ các phép toán trên mảng (array) và ma trận với hiệu năng cao. Cung cấp các hàm toán học cấp cao.
* pandas: Làm việc với dữ liệu dạng bảng (DataFrame) và chuỗi thời gian (Series)
* matplotlib: Vẽ đồ thị và trực quan hóa dữ liệu
* tensorflow: Xây dựng và huấn luyện mô hình học sâu (deep learning).

**Cách quản lý module và thư viện**

* Sử dụng pip list để liệt kê các thư viện đã cài đặt.
* Sử dụng pip uninstall <tên\_thư\_viện> để gỡ bỏ thư viện.
* Tạo tệp yêu cầu (requirements.txt) để lưu danh sách thư viện cần thiết:

pip freeze > requirements.txt

* Cài đặt lại toàn bộ thư viện từ tệp

pip install -r requirements.txt

### Ứng dụng của Python

* Phát triển web: Sử dụng các framework như Django, Flask.
* Phân tích dữ liệu và học máy: Sử dụng các thư viện như Pandas, NumPy, scikit-learn, TensorFlow.
* Khoa học máy tính và trí tuệ nhân tạo: Python là ngôn ngữ phổ biến trong AI và ML.
* Tự động hóa: Python thường được dùng để viết các script tự động hóa công việc.

## Thư viện Pygame

Pygame là một thư viện mã nguồn mở cho ngôn ngữ lập trình Python, được thiết kế để hỗ trợ phát triển game 2D và các ứng dụng đa phương tiện. Thư viện này cung cấp các công cụ và API (Application Programming Interface) để lập trình viên dễ dàng tạo ra đồ họa, xử lý âm thanh, tương tác người dùng, và điều khiển các yếu tố trong game một cách đơn giản và hiệu quả.

### Lịch sử phát triển và mục đích ra đời của Pygame.

#### Lịch sử phát triển của Pygame

Pygame được ra mắt vào khoảng năm 2000, khi Pete Shinners nhận thấy các công cụ tạo game lúc bấy giờ khá phức tạp và thiếu hỗ trợ cho Python. Vì vậy, anh đã xây dựng Pygame dựa trên SDL để tận dụng khả năng vẽ đồ họa nhanh chóng, phát âm thanh và nhận sự tương tác từ người dùng. Ban đầu, thư viện này được phát triển cho việc tạo game đơn giản, nhưng sau đó đã trở nên phổ biến trong cộng đồng Python nhờ tính dễ học và khả năng ứng dụng đa dạng.

Pygame được cập nhật thường xuyên và đã có nhiều cải tiến trong suốt thời gian phát triển để phù hợp với các phiên bản Python mới hơn cũng như các nền tảng hệ điều hành khác nhau. Đặc biệt, từ khi Python chuyển từ Python 2 sang Python 3, Pygame đã phải trải qua nhiều thay đổi để duy trì khả năng tương thích.

#### Mục đích ra đời của Pygame

Pygame được thiết kế để:

1. **Hỗ trợ phát triển game 2D đơn giản**: Với Pygame, các lập trình viên có thể dễ dàng tạo ra các trò chơi 2D với đồ họa và âm thanh cơ bản, mà không cần phải am hiểu sâu về các công nghệ đồ họa phức tạp.
2. **Giúp người học lập trình**: Pygame không chỉ dành cho phát triển game mà còn là công cụ giáo dục phổ biến, giúp người học Python làm quen với lập trình đồ họa, âm thanh và cách tương tác với người dùng. Rất nhiều khóa học lập trình và giáo trình Python hiện nay sử dụng Pygame như một công cụ để dạy lập trình.
3. **Phát triển ứng dụng tương tác đa phương tiện**: Ngoài game, Pygame cũng được sử dụng để tạo các ứng dụng tương tác đa phương tiện khác như trình diễn đồ họa, mô phỏng, và các chương trình liên quan đến giáo dục hoặc trình diễn nghệ thuật.

Pygame vẫn là một công cụ lý tưởng cho các nhà phát triển độc lập, người mới học lập trình và các dự án thử nghiệm hoặc nhỏ lẻ nhờ vào tính dễ dùng và cộng đồng hỗ trợ mạnh mẽ.

### Ứng dụng của Pygame

Pygame chủ yếu được sử dụng để tạo ra các game và ứng dụng tương tác đơn giản. Dưới đây là một số ứng dụng phổ biến của Pygame:

* **Phát triển game 2D**: Pygame có đầy đủ các công cụ hỗ trợ vẽ đồ họa, phát nhạc và hiệu ứng âm thanh, cũng như xử lý tương tác với người dùng, rất phù hợp cho các game dạng arcade, puzzle, hay platformer.
* **Tạo ứng dụng giáo dục**: Nhờ tính dễ học, Pygame được sử dụng nhiều trong các dự án giáo dục. Giáo viên và sinh viên có thể dùng Pygame để tạo ra các mô phỏng khoa học, toán học hay các công cụ học tập trực quan.
* **Ứng dụng tương tác đa phương tiện**: Ngoài game, Pygame cũng có thể dùng để phát triển các ứng dụng trình diễn nghệ thuật, như vẽ đồ họa động, trình chiếu hình ảnh hoặc video, các chương trình biểu diễn âm nhạc.
* **Dự án thử nghiệm và nguyên mẫu**: Pygame là lựa chọn lý tưởng để nhanh chóng tạo các bản mẫu (prototype) cho game hoặc các ứng dụng tương tác khác nhờ vào tính đơn giản và linh hoạt.

### Các thành phần chính của pygame

#### Cửa sổ và bề mặt

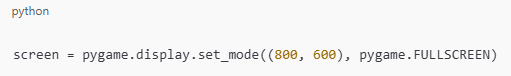
* **Cửa sổ**

Cửa sổ là khu vực hiển thị trò chơi, nơi người chơi sẽ nhìn thấy mọi thứ trong trò chơi.

Để tạo cửa sổ, bạn sử dụng hàm pygame.display.set\_mode() và truyền vào kích thước (chiều rộng và chiều cao) của cửa sổ.



Thiết lập tiêu đề cửa sổ: Sử dụng hàm pygame.display.set\_caption("Tên trò chơi") để đặt tên cho cửa sổ trò chơi.



Điều chỉnh kích thước và chế độ cửa sổ: Bạn có thể tạo các loại cửa sổ khác nhau, ví dụ như chế độ toàn màn hình, bằng cách truyền thêm các cờ vào hàm set\_mode:

* Bề mặt (Surface)

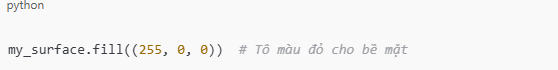
Surface là một trong những khái niệm quan trọng nhất trong Pygame. Nó là đối tượng chứa các pixel mà bạn có thể vẽ lên. Bề mặt này có thể là toàn bộ cửa sổ trò chơi hoặc là các đối tượng đồ họa nhỏ (như nhân vật, vật thể, nền) mà bạn muốn hiển thị.

Ngoài cửa sổ chính (là một Surface), bạn có thể tạo các bề mặt nhỏ khác để vẽ các đối tượng lên đó và sau đó chuyển nó lên màn hình chính.

Sau khi tạo bề mặt, bạn có thể vẽ lên nó bằng các hàm vẽ như pygame.draw hoặc tải hình ảnh lên bề mặt bằng pygame.image.



Sử dụng hàm .fill(color) để tô màu cho bề mặt.

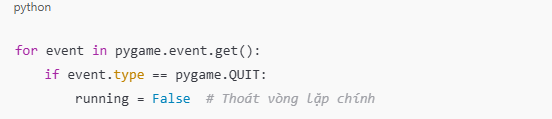


Blitting là hành động vẽ một Surface này lên một Surface khác. Bạn sẽ dùng screen.blit(source\_surface, position) để vẽ bề mặt con lên bề mặt chính.



**Xử lý sự kiện**

* Mục đích: Quản lý và xử lý các sự kiện trong trò chơi, như nhấn phím, di chuột, hoặc nhấp chuột.
* Các hàm phổ biến:
  + pygame.event.get(): Lấy danh sách các sự kiện xảy ra.
  + pygame.event.poll(): Lấy sự kiện đơn lẻ.
* ví dụ:



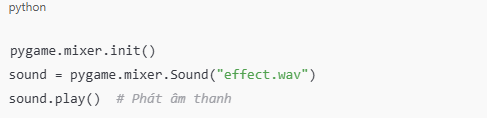
#### Vẽ đồ họa

* Mục đích: Cung cấp các hàm để vẽ các hình cơ bản như đường thẳng, hình tròn, hình chữ nhật trên Surface.
* Các hàm phổ biến:
  + pygame.draw.rect(): Vẽ hình chữ nhật.
  + pygame.draw.circle(): Vẽ hình tròn.
  + pygame.draw.line(): Vẽ đường thẳng.
* Ví dụ:



#### Âm thanh

* Mục đích: Xử lý âm thanh và nhạc nền cho trò chơi.
* Các thành phần chính:
  + pygame.mixer.Sound(): Dùng để chơi các hiệu ứng âm thanh ngắn.
  + pygame.mixer.music: Dùng để phát nhạc nền dài.
* Ví dụ:



#### Thời gian

* **Mục đích:** Quản lý thời gian giúp đảm bảo trò chơi chạy mượt mà ở tốc độ ổn định, điều khiển tốc độ di chuyển của đối tượng, thực hiện các hiệu ứng theo thời gian và quản lý các sự kiện theo thời gian thực trong game.
* **Các hàm phổ biến:**
  + pygame.time.Clock(): Tạo một đối tượng Clock để điều khiển tốc độ khung hình (frame rate).
  + clock.tick(fps): Giới hạn tốc độ khung hình của game, với fps là số khung hình tối đa mỗi giây.
  + pygame.time.get\_ticks(): Lấy tổng thời gian (tính bằng mili giây) kể từ khi Pygame khởi động. Hữu ích cho các tính năng đếm giờ và thời gian trễ.
  + pygame.time.delay(milliseconds): Dừng chương trình trong khoảng thời gian tính bằng mili giây. Thường dùng để tạm dừng một hoạt động ngắn.
* **Ví dụ:**

import pygame

import sys

# Khởi tạo Pygame

pygame.init()

# Thiết lập kích thước cửa sổ

screen = pygame.display.set\_mode((500, 500))

# Tạo Clock để quản lý thời gian

clock = pygame.time.Clock()

# Lấy thời điểm ban đầu

start\_time = pygame.time.get\_ticks()

# Vòng lặp game

while True:

# Xử lý sự kiện

for event in pygame.event.get():

if event.type == pygame.QUIT:

pygame.quit()

sys.exit()

# Cập nhật màn hình và nội dung

screen.fill((0, 0, 0)) # Đổ màu nền đen

# Lấy thời gian hiện tại và tính thời gian trôi qua

current\_time = pygame.time.get\_ticks()

elapsed\_time = current\_time - start\_time

# Cứ mỗi 1000 mili giây (1 giây), thực hiện hành động

if elapsed\_time > 1000:

print("1 giây đã trôi qua")

start\_time = current\_time # Cập nhật lại thời gian bắt đầu

# Cập nhật màn hình

pygame.display.flip()

# Giới hạn tốc độ khung hình ở mức 30 FPS

clock.tick(30)

#### Quản lý sprites

* Mục đích: Quản lý sprites giúp tổ chức và điều khiển các đối tượng trong game, từ nhân vật chính, kẻ thù, đến các vật thể trên màn hình. Sprites cho phép dễ dàng cập nhật, vẽ và xử lý va chạm của các đối tượng trong trò chơi một cách có tổ chức và hiệu quả.
* Các thành phần phổ biến:
  + pygame.sprite.Sprite: Là lớp cơ bản cho tất cả các sprite. Các đối tượng game kế thừa từ lớp này để trở thành một sprite với các thuộc tính như vị trí, hình ảnh và kích thước.
  + pygame.sprite.Group: Là nhóm để chứa nhiều sprite, giúp quản lý và cập nhật chúng dễ dàng hơn. Có nhiều loại nhóm như:
    - pygame.sprite.Group(): Nhóm cơ bản chứa các sprite.
    - pygame.sprite.RenderPlain(): Nhóm có chức năng vẽ và cập nhật sprite.
    - pygame.sprite.GroupSingle(): Nhóm chứa một sprite duy nhất, thường dùng để quản lý các đối tượng duy nhất như nhân vật chính.
  + pygame.sprite.Sprite.add() và pygame.sprite.Sprite.remove(): Thêm hoặc xóa một sprite khỏi nhóm.
  + pygame.sprite.Group.draw(surface): Vẽ tất cả các sprite trong nhóm lên một surface (thường là màn hình chính của game).
  + pygame.sprite.Group.update(): Cập nhật tất cả các sprite trong nhóm, gọi hàm update cho từng sprite trong nhóm.
  + pygame.sprite.spritecollide(sprite, group, dokill): Kiểm tra va chạm giữa một sprite và các sprite trong nhóm group. Nếu dokill=True, các sprite trong nhóm bị va chạm sẽ bị xóa.
* Ví dụ:

import pygame

import sys

# Khởi tạo Pygame

pygame.init()

# Thiết lập kích thước cửa sổ

screen = pygame.display.set\_mode((500, 500))

clock = pygame.time.Clock()

# Tạo một lớp nhân vật chính kế thừa từ pygame.sprite.Sprite

class Player(pygame.sprite.Sprite):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.image = pygame.Surface((50, 50)) # Tạo bề mặt hình vuông cho nhân vật

self.image.fill((0, 128, 255)) # Đổ màu xanh cho nhân vật

self.rect = self.image.get\_rect(center=(250, 250)) # Vị trí ban đầu

def update(self):

keys = pygame.key.get\_pressed()

if keys[pygame.K\_LEFT]: self.rect.x -= 5

if keys[pygame.K\_RIGHT]: self.rect.x += 5

if keys[pygame.K\_UP]: self.rect.y -= 5

if keys[pygame.K\_DOWN]: self.rect.y += 5

# Tạo một lớp kẻ thù kế thừa từ pygame.sprite.Sprite

class Enemy(pygame.sprite.Sprite):

def \_\_init\_\_(self, pos):

super().\_\_init\_\_()

self.image = pygame.Surface((30, 30)) # Tạo bề mặt hình vuông nhỏ hơn cho kẻ thù

self.image.fill((255, 0, 0)) # Đổ màu đỏ cho kẻ thù

self.rect = self.image.get\_rect(center=pos)

# Tạo nhân vật chính và nhóm sprite cho kẻ thù

player = Player()

enemies = pygame.sprite.Group()

# Thêm một số kẻ thù vào nhóm

for i in range(5):

enemy = Enemy((100 \* i + 50, 100))

enemies.add(enemy)

# Tạo nhóm tổng hợp chứa tất cả các sprite

all\_sprites = pygame.sprite.Group(player, \*enemies)

# Vòng lặp game

while True:

# Xử lý sự kiện

for event in pygame.event.get():

if event.type == pygame.QUIT:

pygame.quit()

sys.exit()

# Cập nhật sprite

all\_sprites.update()

# Kiểm tra va chạm giữa player và các kẻ thù

if pygame.sprite.spritecollide(player, enemies, dokill=True):

print("Va chạm! Kẻ thù bị tiêu diệt")

# Vẽ màn hình

screen.fill((255, 255, 255)) # Đổ màu nền trắng

all\_sprites.draw(screen) # Vẽ tất cả sprite lên màn hình

pygame.display.flip()

clock.tick(30) # Giới hạn tốc độ khung hình

#### Phát hiện va chạm

* Mục đích: Phát hiện va chạm giúp kiểm tra sự tương tác giữa các đối tượng, từ đó thực hiện các hành động phản ứng như: tiêu diệt kẻ thù, nhặt vật phẩm, hoặc mất máu khi chạm vào chướng ngại vật.
* Các hàm phổ biến:
  + pygame.sprite.collide\_rect(sprite1, sprite2): Kiểm tra va chạm giữa hai sprite bằng cách so sánh trực tiếp các hình chữ nhật (rect) của chúng. Nếu rect của hai sprite chạm nhau, hàm sẽ trả về True.
  + pygame.sprite.spritecollide(sprite, group, dokill, collided=None): Kiểm tra va chạm giữa một sprite và một nhóm sprite. Trả về danh sách các sprite trong group chạm với sprite. Nếu dokill=True, các sprite va chạm sẽ bị xóa khỏi nhóm.
  + pygame.sprite.groupcollide(group1, group2, dokill1, dokill2, collided=None): Kiểm tra va chạm giữa tất cả các sprite trong group1 và group2. Trả về một từ điển, trong đó mỗi sprite của group1 có một danh sách các sprite của group2 mà nó va chạm. dokill1 và dokill2 cho phép xóa các sprite trong group1 và group2 khi va chạm.
  + pygame.sprite.collide\_circle(sprite1, sprite2): Kiểm tra va chạm giữa hai sprite sử dụng phạm vi hình tròn. Phương pháp này hữu ích khi các đối tượng có hình dạng tròn hoặc gần tròn. Sprite phải có thuộc tính radius để sử dụng hàm này.
* Ví dụ:

import pygame

import sys

# Khởi tạo Pygame

pygame.init()

# Thiết lập kích thước cửa sổ

screen = pygame.display.set\_mode((500, 500))

clock = pygame.time.Clock()

# Lớp nhân vật chính

class Player(pygame.sprite.Sprite):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.image = pygame.Surface((50, 50))

self.image.fill((0, 128, 255))

self.rect = self.image.get\_rect(center=(250, 400))

def update(self):

keys = pygame.key.get\_pressed()

if keys[pygame.K\_LEFT]: self.rect.x -= 5

if keys[pygame.K\_RIGHT]: self.rect.x += 5

if keys[pygame.K\_UP]: self.rect.y -= 5

if keys[pygame.K\_DOWN]: self.rect.y += 5

# Lớp kẻ thù

class Enemy(pygame.sprite.Sprite):

def \_\_init\_\_(self, pos):

super().\_\_init\_\_()

self.image = pygame.Surface((30, 30))

self.image.fill((255, 0, 0))

self.rect = self.image.get\_rect(center=pos)

# Tạo các nhóm sprite

player = Player()

enemies = pygame.sprite.Group()

# Tạo một số kẻ thù và thêm vào nhóm

for i in range(5):

enemy = Enemy((100 \* i + 50, 100))

enemies.add(enemy)

all\_sprites = pygame.sprite.Group(player, \*enemies)

# Vòng lặp game

while True:

# Xử lý sự kiện

for event in pygame.event.get():

if event.type == pygame.QUIT:

pygame.quit()

sys.exit()

# Cập nhật tất cả các sprite

all\_sprites.update()

# Kiểm tra va chạm giữa player và enemies

hits = pygame.sprite.spritecollide(player, enemies, dokill=True)

if hits:

print("Va chạm! Kẻ thù bị tiêu diệt.")

# Vẽ màn hình

screen.fill((255, 255, 255))

all\_sprites.draw(screen)

pygame.display.flip()

clock.tick(30)

#### Đầu vào từ bàn phím và chuột

##### Bàn phím

Để tương tác với bàn phím, ta có thể sử dụng module **pygame.key**

Các thuộc tính và phương thức:

* pygame.key.get\_focused()

Phương thức này trả về True nếu cửa sổ Pygame đang nhận đầu vào từ bàn phím, tức là nó đang ở trạng thái "focused" và sẵn sàng nhận các sự kiện từ bàn phím. Ngược lại, nếu cửa sổ Pygame bị "mất focus" (ví dụ: khi người dùng nhấp vào một cửa sổ khác), phương thức này trả về False.

* pygame.key.get\_pressed()

Phương thức này trả về một danh sách trạng thái của tất cả các phím trên bàn phím. Mỗi vị trí trong danh sách tương ứng với một phím, với giá trị 1 nếu phím đang được nhấn và 0 nếu không. Điều này giúp kiểm tra trạng thái của nhiều phím cùng lúc.

* pygame.key.get\_mods()

Phương thức này trả về một giá trị nguyên thể hiện trạng thái của các phím bổ trợ (modifier keys) hiện đang được giữ, chẳng hạn như Shift, Ctrl, Alt, v.v. Các phím bổ trợ này thường được sử dụng kết hợp với các phím khác.

* pygame.key.set\_mods(modifiers)

Thiết lập tần suất và độ trễ khi một phím được giữ, với delay là độ trễ ban đầu (tính bằng mili giây) và interval là khoảng thời gian lặp lại khi phím được giữ.

* pygame.key.get\_repeat()

Phương thức này trả về cấu hình độ trễ và tần suất lặp lại của phím khi được giữ. Giá trị trả về là một bộ hai phần tử (delay, interval).

* pygame.key.name(key\_id)

Phương thức này nhận vào key\_id (ID của phím) và trả về tên của phím đó dưới dạng chuỗi (string).

* pygame.key.key\_code(key\_name)

Phương thức này nhận vào key\_name (tên của phím dưới dạng chuỗi) và trả về ID của phím đó.

* pygame.key.start\_text\_input()

Bắt đầu xử lý các sự kiện nhập văn bản Unicode, chủ yếu sử dụng trong các ứng dụng yêu cầu nhập liệu ký tự.

* pygame.key.stop\_text\_input()

Kết thúc xử lý các sự kiện nhập văn bản Unicode, dừng nhận đầu vào văn bản.

* pygame.key.set\_text\_input\_rect(rect)

Xác định vị trí của danh sách gợi ý văn bản (candidate list) trong cửa sổ khi nhập liệu. rect là đối tượng hình chữ nhật xác định vị trí và kích thước của danh sách.

Pygame cung cấp module **pygame.event** để quản lý các sự kiện (events) và hàng đợi sự kiện (event queue). Khi người dùng nhấn hoặc thả các phím trên bàn phím, các sự kiện **pygame.KEYDOWN** (khi phím được nhấn) và **pygame.KEYUP** (khi phím được nhả) sẽ được thêm vào hàng đợi sự kiện của Pygame. Cả hai sự kiện này đều có các thuộc tính **key** và **mod**, và riêng sự kiện **pygame.KEYDOWN** có thêm hai thuộc tính bổ sung là **unicode** và **scancode**

* key

Thuộc tính key là một số nguyên đại diện cho mỗi phím trên bàn phím, giúp bạn dễ dàng xác định phím nào đã được nhấn hoặc thả. Mỗi phím trên bàn phím có một mã định danh riêng.

* mod

Thuộc tính mod là một bitmask đại diện cho các phím bổ trợ (modifier keys) như Shift, Ctrl, Alt, v.v., đang được giữ khi sự kiện xảy ra. Giá trị của mod cho phép xác định nhiều phím bổ trợ cùng lúc.

* unicode

Thuộc tính unicode là một chuỗi ký tự đơn đại diện cho ký tự đầy đủ mà người dùng đã nhập. Giá trị này tính đến các phím bổ trợ như Shift và các phím tạo tổ hợp ký tự, giúp xác định chính xác ký tự cuối cùng được nhập vào.

* scancode

Thuộc tính scancode là mã phím đặc trưng cho từng nền tảng, thể hiện mã phím riêng của từng bàn phím (có thể khác nhau tùy theo loại bàn phím). scancode hữu ích trong việc xử lý các phím đặc biệt như phím đa phương tiện.

**Ví dụ minh họa:**

Chương trình vẽ và điều khiển ô vuông di chuyển sử dụng phím mũi tên

**import pygame**

**import sys**

**# Khởi tạo Pygame**

**pygame.init()**

**# Thiết lập màn hình**

**screen\_width, screen\_height = 800, 600**

**screen = pygame.display.set\_mode((screen\_width, screen\_height))**

**pygame.display.set\_caption("Pygame Keyboard Example")**

**# Màu sắc và thông số hình vuông**

**WHITE = (255, 255, 255)**

**RED = (255, 0, 0)**

**square\_size = 50**

**x, y = screen\_width // 2, screen\_height // 2 # Vị trí ban đầu của hình vuông**

**velocity = 5 # Tốc độ di chuyển**

**# Thiết lập lặp lại phím (tốc độ phản hồi khi giữ phím)**

**pygame.key.set\_repeat(50, 50)**

**# Vòng lặp chính của trò chơi**

**running = True**

**while running:**

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

**if event.type == pygame.QUIT:**

**running = False**

**elif event.type == pygame.KEYDOWN:**

**# Tăng tốc nếu nhấn giữ Shift**

**if event.mod & pygame.KMOD\_SHIFT:**

**speed = velocity \* 2**

**else:**

**speed = velocity**

**# Kiểm tra phím mũi tên để di chuyển hình vuông**

**if event.key == pygame.K\_LEFT:**

**x -= speed**

**elif event.key == pygame.K\_RIGHT:**

**x += speed**

**elif event.key == pygame.K\_UP:**

**y -= speed**

**elif event.key == pygame.K\_DOWN:**

**y += speed**

**# Đặt màu nền và vẽ hình vuông**

**screen.fill(WHITE)**

**pygame.draw.rect(screen, RED, (x, y, square\_size, square\_size))**

**# Cập nhật màn hình**

**pygame.display.flip()**

**# Thoát khỏi Pygame**

**pygame.quit()**

**sys.exit()**

### 

##### Chuột

Để làm việc với chuột, ta có thể sử dụng thư viện **pygame.mouse**

Các thuộc tính và phương thức:

* pygame.mouse.get\_pressed()

Trả về một tuple với ba giá trị, tương ứng với ba nút chuột: trái, giữa, và phải. Mỗi giá trị là 1 nếu nút đang được nhấn và 0 nếu không.

* pygame.mouse.get\_pos()

Trả về vị trí hiện tại của con trỏ chuột dưới dạng một tuple (x, y), trong đó x và y là tọa độ trên màn hình.

* pygame.mouse.get\_rel()

Trả về khoảng cách di chuyển của chuột kể từ lần gọi hàm cuối cùng, dưới dạng một tuple (dx, dy) với dx là di chuyển ngang và dy là di chuyển dọc.

* pygame.mouse.set\_pos(pos)

Đặt vị trí của con trỏ chuột tại tọa độ pos, là một tuple (x, y)

* pygame.mouse.set\_visible(visible)

Điều khiển việc hiển thị hoặc ẩn con trỏ chuột. Nếu visible là True, con trỏ sẽ hiển thị; nếu False, con trỏ sẽ bị ẩn.

* pygame.mouse.get\_visible()

Trả về True nếu con trỏ chuột đang hiển thị, False nếu bị ẩn.

* pygame.mouse.get\_focused()

Trả về True nếu cửa sổ Pygame hiện tại đang nhận đầu vào từ chuột, và False nếu không. Điều này xảy ra khi cửa sổ Pygame được "focus" và nhận sự kiện chuột.

* pygame.mouse.set\_cursor(cursor)

Đặt con trỏ chuột thành một con trỏ mới. cursor là một tuple chứa thông tin về con trỏ (vd: ảnh, ký hiệu mũi tên, …).

* pygame.mouse.get\_cursor()

Trả về con trỏ chuột hiện tại đang được sử dụng trong Pygame.

**Ví dụ minh họa:**

Chương trình vẽ hình tròn và cho phép người dùng kéo và di chuyển bằng chuột trái

import pygame

import sys

# Khởi tạo Pygame

pygame.init()

# Thiết lập màn hình

screen\_width, screen\_height = 800, 600

screen = pygame.display.set\_mode((screen\_width, screen\_height))

pygame.display.set\_caption("Mouse Example in Pygame")

# Màu sắc và thông số hình tròn

WHITE = (255, 255, 255)

BLUE = (0, 0, 255)

circle\_radius = 30

circle\_pos = [screen\_width // 2, screen\_height // 2] # Vị trí ban đầu của hình tròn

dragging = False # Biến kiểm tra trạng thái kéo thả

# Vòng lặp chính

running = True

while running:

for event in pygame.event.get():

if event.type == pygame.QUIT:

running = False

elif event.type == pygame.MOUSEBUTTONDOWN:

# Nếu nhấn chuột trái và chuột nằm trong phạm vi hình tròn, kích hoạt kéo

if event.button == 1: # Chuột trái

mouse\_x, mouse\_y = event.pos

distance = ((mouse\_x - circle\_pos[0]) \*\* 2 + (mouse\_y - circle\_pos[1]) \*\* 2) \*\* 0.5

if distance < circle\_radius:

dragging = True

elif event.type == pygame.MOUSEBUTTONUP:

# Ngừng kéo khi nhả chuột trái

if event.button == 1:

dragging = False

elif event.type == pygame.MOUSEMOTION:

# Cập nhật vị trí hình tròn khi đang kéo

if dragging:

circle\_pos = list(event.pos)

# Vẽ màn hình

screen.fill(WHITE)

pygame.draw.circle(screen, BLUE, circle\_pos, circle\_radius)

pygame.display.flip()

# Thoát khỏi Pygame

pygame.quit()

sys.exit()

## 

#### Hệ thống phông chữ

Để load và hiển thị chữ trong pygame, ta có thể sử dụng module **pygame.font**

Các thuộc tính và phương thức:

* pygame.font.init()

Khởi tạo module font trong Pygame. Phải được gọi trước khi có thể tạo và sử dụng các đối tượng font. Cần gọi hàm này một lần trước khi sử dụng bất kỳ chức năng nào của **pygame.font**

* pygame.font.quit()

Hủy khởi tạo module font, giải phóng các tài nguyên liên quan đến font. Thường gọi hàm này khi không cần sử dụng font nữa hoặc trước khi thoát ứng dụng để giải phóng tài nguyên.

* pygame.font.get\_init()

Trả về True nếu module font đã được khởi tạo, ngược lại là False.

* pygame.font.get\_default\_font()

Trả về tên file của font mặc định do Pygame cung cấp.

* pygame.font.get\_sdl\_ttf\_version()

Trả về phiên bản của thư viện SDL\_ttf mà Pygame đang sử dụng dưới dạng tuple (major, minor, patch).

* pygame.font.get\_fonts()

Trả về danh sách tên của tất cả các font chữ có sẵn trên hệ thống.

* pygame.font.match\_font(name: String, bold:Bool, italic: Bool)

Tìm font cụ thể trên hệ thống với tên, và có thể thêm kiểu in đậm (bold) hoặc nghiêng (italic). Trả về tên file của font phù hợp, hoặc None nếu không tìm thấy.

* pygame.font.SysFont(name, size, bold=False, italic=False)

Tạo đối tượng Font từ font hệ thống với tên name, kích thước size, và có thể tùy chọn kiểu bold hoặc italic.

* pygame.font.Font(file, size)

Tạo đối tượng Font từ một file font nhất định (ví dụ: .ttf), với kích thước size.

**Ví dụ minh họa:**

Chương trình viết dòng chữ đơn giản lên chính giữa màn hình

**import pygame**

**import sys**

**# Khởi tạo Pygame**

**pygame.init()**

**# Thiết lập màn hình**

**screen = pygame.display.set\_mode((800, 600))**

**pygame.display.set\_caption("Font Example in Pygame")**

**# Khởi tạo font**

**pygame.font.init()**

**# Sử dụng font hệ thống**

**sys\_font = pygame.font.SysFont('arial', 40, bold=True)**

**# Màu sắc**

**WHITE = (255, 255, 255)**

**BLACK = (0, 0, 0)**

**RED = (255, 0, 0)**

**# Tạo các surface văn bản**

**text\_surface\_sys = sys\_font.render("Hello, world !", True, BLACK)**

**# Tính toán vị trí để căn giữa text trên màn hình**

**text\_rect\_sys = text\_surface\_sys.get\_rect(center=(screen.get\_width() // 2, screen.get\_height() // 2))**

**# Vòng lặp chính**

**running = True**

**while running:**

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

**if event.type == pygame.QUIT:**

**running = False**

**# Vẽ màn hình**

**screen.fill(WHITE)**

**screen.blit(text\_surface\_sys, text\_rect\_sys.topleft) # Hiển thị văn bản hệ thống ở giữa màn hình**

**pygame.display.flip()**

**# Kết thúc Pygame**

**pygame.font.quit()**

**pygame.quit()**

**sys.exit()**

## 

#### Hiệu ứng hình ảnh

Trong Pygame, việc tạo hiệu ứng hình ảnh giúp tăng cường trải nghiệm thị giác cho người dùng. Pygame cung cấp nhiều công cụ và phương pháp để tạo ra các hiệu ứng phổ biến như di chuyển, làm mờ, chuyển cảnh, và xử lý ảnh. Dưới đây là một số thành phần chính thường được sử dụng khi xây dựng hiệu ứng hình ảnh trong Pygame:

* Chuyển động và hoạt ảnh (Animation)

Hiệu ứng di chuyển thường được tạo ra bằng cách thay đổi vị trí của các đối tượng trên màn hình theo thời gian hoặc theo các quy luật vật lý

**Ví dụ minh họa:**

Chương trình tạo hoạt ảnh cho nhân vật khi điều khiển di chuyển trái phải

sprite\_sheet: pygame-animation/DinoSprites-tard.py



import os

import pygame

import sys

# Initialize Pygame

pygame.init()

# Screen settings

WIDTH, HEIGHT = 800, 600

screen = pygame.display.set\_mode((WIDTH, HEIGHT))

pygame.display.set\_caption("Sprite Animation Example")

assets\_folder = os.path.join("pygame-animation")

# Load the sprite sheet

sprite\_sheet = pygame.image.load(os.path.join(assets\_folder, "DinoSprites-tard.png")).convert\_alpha()

# Sprite frame settings

FRAME\_WIDTH = 24

FRAME\_HEIGHT = 24

NUM\_FRAMES = 24

# Extract frames from the sprite sheet for both idle and movement animations

idle\_frame = sprite\_sheet.subsurface((0, 0, FRAME\_WIDTH, FRAME\_HEIGHT)) # Frame 1 (idle)

movement\_frames = [sprite\_sheet.subsurface((i \* FRAME\_WIDTH, 0, FRAME\_WIDTH, FRAME\_HEIGHT)) for i in range(18, 24)]

# Extract each frame from the sprite sheet

frames = []

for i in range(NUM\_FRAMES):

frame = sprite\_sheet.subsurface((i \* FRAME\_WIDTH, 0, FRAME\_WIDTH, FRAME\_HEIGHT))

frames.append(frame)

# Animation settings

frame\_index = 0

frame\_delay = 5 # Delay between frames to control the animation speed

frame\_counter = 0

x\_pos = WIDTH // 2 - FRAME\_WIDTH // 2

y\_pos = HEIGHT // 2 - FRAME\_HEIGHT // 2

speed = 5

moving\_left = False

moving\_right = False

facing\_left = False

# Main loop

running = True

while running:

for event in pygame.event.get():

if event.type == pygame.QUIT:

running = False

elif event.type == pygame.KEYDOWN:

if event.key == pygame.K\_LEFT:

moving\_left = True

facing\_left = True

elif event.key == pygame.K\_RIGHT:

moving\_right = True

facing\_left = False

elif event.type == pygame.KEYUP:

if event.key == pygame.K\_LEFT:

moving\_left = False

elif event.key == pygame.K\_RIGHT:

moving\_right = False

# Clear the screen

screen.fill((255, 255, 255))

# Update the position based on movement flags

if moving\_left:

x\_pos -= speed

if moving\_right:

x\_pos += speed

# Update the animation frame only when moving

if moving\_left or moving\_right:

frame\_counter += 1

if frame\_counter >= frame\_delay:

frame\_counter = 0

frame\_index = (frame\_index + 1) % len(movement\_frames) # Loop through movement frames

current\_frame = movement\_frames[frame\_index]

else:

# Use the idle frame when not moving

frame\_index = 0

current\_frame = idle\_frame

if facing\_left:

current\_frame = pygame.transform.flip(current\_frame, True, False)

# Draw the current frame on the screen

screen.blit(current\_frame, (x\_pos, y\_pos))

# Update the display

pygame.display.flip()

pygame.time.Clock().tick(30)

# Quit Pygame

pygame.quit()

sys.exit()

* Hiệu ứng mờ và làm sáng (Fade In, Fade Out)

Hiệu ứng mờ dần giúp chuyển cảnh hoặc làm nổi bật các phần khác nhau của trò chơi.

**Mờ dần (Fade Out):** Thay đổi giá trị độ trong suốt (alpha) của đối tượng hoặc màn hình từ 255 (hiển thị đầy đủ) về 0 (mờ hoàn toàn).

**Làm sáng (Fade In):** Ngược lại, thay đổi độ trong suốt từ 0 lên 255 để đối tượng hiện dần lên.

**Ví dụ minh họa:**

import pygame

import sys

# Initialize Pygame

pygame.init()

# Screen settings

WIDTH, HEIGHT = 800, 600

screen = pygame.display.set\_mode((WIDTH, HEIGHT))

pygame.display.set\_caption("Fade In and Fade Out Example")

# Colors

BACKGROUND\_COLOR = (50, 150, 200)

FADE\_COLOR = (0, 0, 0) # Black for fade effect

# Fade settings

fade\_alpha = 255 # Starting transparency for fade-out effect

fade\_speed = 5 # Controls how fast the fade occurs

fading\_in = True # Start with fade-in effect

# Create a surface for the fade effect

fade\_surface = pygame.Surface((WIDTH, HEIGHT))

fade\_surface.fill(FADE\_COLOR)

# Main loop

running = True

while running:

for event in pygame.event.get():

if event.type == pygame.QUIT:

running = False

# Clear the screen with background color

screen.fill(BACKGROUND\_COLOR)

# Apply the fade effect

if fading\_in:

fade\_alpha -= fade\_speed # Decrease alpha for fade-in

if fade\_alpha <= 0:

fade\_alpha = 0 # End of fade-in

fading\_in = False # Switch to fade-out after fade-in completes

else:

fade\_alpha += fade\_speed # Increase alpha for fade-out

if fade\_alpha >= 255:

fade\_alpha = 255 # End of fade-out

fading\_in = True # Restart fade-in after fade-out completes

# Apply the fade effect to the overlay surface and draw it

fade\_surface.set\_alpha(fade\_alpha)

screen.blit(fade\_surface, (0, 0))

# Update the display

pygame.display.flip()

pygame.time.Clock().tick(30)

# Quit Pygame

pygame.quit()

sys.exit()

* Xoay và phóng to (Rotation and Scaling)

Pygame cho phép xoay và thay đổi kích thước đối tượng, giúp tạo các hiệu ứng như quay tròn hoặc phóng to, thu nhỏ.

**Xoay (Rotation):** Sử dụng **pygame.transform.rotate** để xoay đối tượng. Góc xoay có thể được thay đổi theo thời gian để tạo hiệu ứng quay liên tục.

**Ví dụ minh họa:**

**import pygame**

**import sys**

**# Initialize Pygame**

**pygame.init()**

**# Screen settings**

**WIDTH, HEIGHT = 800, 600**

**screen = pygame.display.set\_mode((WIDTH, HEIGHT))**

**pygame.display.set\_caption("Rotate and Reverse Animation Example")**

**# Colors**

**BACKGROUND\_COLOR = (50, 150, 200)**

**SQUARE\_COLOR = (255, 100, 100)**

**# Square settings**

**square\_size = 100**

**square\_surface = pygame.Surface((square\_size, square\_size), pygame.SRCALPHA)**

**pygame.draw.rect(square\_surface, SQUARE\_COLOR, (0, 0, square\_size, square\_size))**

**# Rotation settings**

**angle = 0 # Starting angle**

**rotation\_speed = 2 # Degrees to rotate per frame**

**direction = 1 # 1 for clockwise, -1 for counterclockwise**

**# Main loop**

**running = True**

**while running:**

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

**if event.type == pygame.QUIT:**

**running = False**

**# Clear the screen**

**screen.fill(BACKGROUND\_COLOR)**

**# Update the angle for rotation**

**angle += rotation\_speed \* direction**

**# Check if the square has completed a full 360-degree rotation**

**if angle >= 360 or angle <= -360:**

**angle = 0 # Reset angle to zero**

**direction \*= -1 # Reverse the direction**

**# Rotate the square**

**rotated\_square = pygame.transform.rotate(square\_surface, angle)**

**rotated\_rect = rotated\_square.get\_rect(center=(WIDTH // 2, HEIGHT // 2))**

**# Draw the rotated square at the center of the screen**

**screen.blit(rotated\_square, rotated\_rect.topleft)**

**# Update the display**

**pygame.display.flip()**

**pygame.time.Clock().tick(60) # Limit the frame rate to 60 FPS**

**# Quit Pygame**

**pygame.quit()**

**sys.exit()**

**Phóng to và thu nhỏ (Scaling):** Sử dụng **pygame.transform.scale** hoặc **pygame.transform.smoothscale** để thay đổi kích thước của đối tượng.

**Ví dụ minh họa:**

import pygame

import sys

# Initialize Pygame

pygame.init()

# Screen settings

WIDTH, HEIGHT = 800, 600

screen = pygame.display.set\_mode((WIDTH, HEIGHT))

pygame.display.set\_caption("Scale Animation Example")

# Colors

BACKGROUND\_COLOR = (30, 30, 30)

SQUARE\_COLOR = (0, 150, 250)

# Square settings

original\_square\_size = 100 # The original size of the square

max\_scale = 1.5 # Maximum scale factor

min\_scale = 0.5 # Minimum scale factor

scale\_speed = 0.01 # Scale speed per frame

current\_scale = 1 # Starting scale factor

scaling\_up = True # Flag to control the scaling direction

# Main loop

running = True

while running:

for event in pygame.event.get():

if event.type == pygame.QUIT:

running = False

# Clear the screen

screen.fill(BACKGROUND\_COLOR)

# Update scale factor

if scaling\_up:

current\_scale += scale\_speed

if current\_scale >= max\_scale:

scaling\_up = False

else:

current\_scale -= scale\_speed

if current\_scale <= min\_scale:

scaling\_up = True

# Calculate new size based on the scale factor

new\_size = int(original\_square\_size \* current\_scale)

square\_surface = pygame.Surface((new\_size, new\_size), pygame.SRCALPHA)

pygame.draw.rect(square\_surface, SQUARE\_COLOR, (0, 0, new\_size, new\_size))

# Center the square on the screen

rect = square\_surface.get\_rect(center=(WIDTH // 2, HEIGHT // 2))

screen.blit(square\_surface, rect.topleft)

# Update the display

pygame.display.flip()

pygame.time.Clock().tick(60) # Limit the frame rate to 60 FPS

# Quit Pygame

pygame.quit()

sys.exit()

* Làm mờ đối tượng (Blur)

Mặc dù Pygame không có sẵn hàm làm mờ, ta có thể làm mờ một đối tượng bằng cách áp dụng các phép biến đổi thủ công:

**Áp dụng lớp phủ mờ:** Sử dụng ảnh mờ hoặc các lớp mờ với alpha thấp để phủ lên đối tượng cần làm mờ.

**Giảm độ phân giải:** Tạo hiệu ứng làm mờ bằng cách giảm độ phân giải đối tượng và phóng to lại, tuy nhiên phương pháp này sẽ làm đối tượng mất chi tiết.

**Ví dụ minh họa:**

Áp dụng hiệu ứng làm mờ hình ảnh sử dụng thư viện **Pillow**

import os

import pygame

import numpy as np

from PIL import Image, ImageFilter

# Initialize Pygame

pygame.init()

# Set up the display

width, height = 800, 600

screen = pygame.display.set\_mode((width, height))

pygame.display.set\_caption('Blur Effect Example')

# Load the image with Pillow, apply blur, and convert to Pygame surface

def load\_and\_blur\_image(image\_path, blur\_radius):

"""Load an image, apply a blur effect, and return it as a Pygame surface."""

# Open the image using Pillow

pil\_image = Image.open(image\_path)

# Convert image to RGB (if not already in RGB mode)

if pil\_image.mode != 'RGB':

pil\_image = pil\_image.convert('RGB')

# Apply Gaussian blur

blurred\_image = pil\_image.filter(ImageFilter.GaussianBlur(blur\_radius))

# Convert the Pillow image to a format that Pygame can use

return pygame.image.fromstring(blurred\_image.tobytes(), blurred\_image.size, 'RGB')

# Load and blur the image

# Load an image (ensure you have an image file in the same directory)

assets\_folder = os.path.join("pygame-animation")

image\_path = os.path.join(assets\_folder, 'image-blur-test.jpg')

blur\_radius = 15 # Adjust the blur radius as needed

blurred\_image\_surface = load\_and\_blur\_image(image\_path, blur\_radius)

# Main loop

running = True

while running:

for event in pygame.event.get():

if event.type == pygame.QUIT:

running = False

# Clear the screen

screen.fill((255, 255, 255))

# Draw the blurred image

screen.blit(blurred\_image\_surface, (0, 0)) # Position the blurred image

# Update the display

pygame.display.flip()

# Quit Pygame

pygame.quit()

# Quá trình xây dựng và triển khai

## Giới thiệu

Game bắn súng 2D hai người chơi là một trò chơi hành động hấp dẫn, nơi hai người chơi cạnh tranh với nhau trong một môi trường 2D để xem ai có thể tiêu diệt đối thủ đầu tiên. Trong game, mỗi người điều khiển 1 nhân vật sử dụng vũ khí để bắn hạ đối thủ trong khi né tránh đạn.

## Tính năng

* Kết nối 2 người chơi thông qua mạng LAN
* Điều khiển nhân vật di chuyển và bắn đạn
* Tính điểm khi người chơi tiêu diệt đối thủ hoặc bị tiêu diệt
* Hiển thị thông báo khi người chơi thắng hoặc thua

## Triển khai

**Server:**

* Khởi tạo socket, logic nhận và gửi data khi có sự kiện được gửi đến

import socket

from \_thread import \*

import sys

import json

from receive\_handle import receiveHandle

from encode import encode

from decode import decode

server = "localhost"

port = 8080

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

try:

s.bind((server, port))

except socket.error as e:

str(e)

s.listen(2)

print("Waiting for a connection, Server Started")

def threaded\_client(conn: socket, addr, player: int):

try:

conn.send(encode({"action": 3, "data": {"currPlayer": player}}))

while True:

try:

# Receive data from the client

data = conn.recv(2048)

# Check if data is received

if data:

decodeData = decode(data)

receiveHandle(conn, decodeData, player)

else:

# No data received, client closed the connection

break

except json.JSONDecodeError as e:

# Handle JSON decoding errors

print("JSON decoding error:", e)

except socket.error as e:

# Handle socket errors

print("Socket error:", e)

break

finally:

print("Lost connection")

conn.close()

player = 0

while True:

conn, addr = s.accept()

print("Connected to:", addr)

start\_new\_thread(threaded\_client, (conn, addr, player))

player += 1

from enum import Enum

import socket

from encode import encode

class GAME\_ACTION(Enum):

MOVE = 0

FIRE = 1

HITTING = 2

START = 3

SCORE = 4

MOVE\_DATA = [{"player": 0, "x": 20, "y": 20}, {"player": 1, "x": 700, "y": 500}]

BULLET\_DATA = [{"player": 0, "x": 0, "y": 0}, {"player": 1, "x": 0, "y": 0}]

SCORE\_DATA = [0, 0]

def receiveHandle(conn: socket, data: dict, player: int) -> None:

action = data.get("action")

if action == GAME\_ACTION.MOVE.value:

MOVE\_DATA[player] = data["data"]

sendData = encode(

{

"action": GAME\_ACTION.MOVE.value,

"data": MOVE\_DATA[0] if player == 1 else MOVE\_DATA[1],

}

)

conn.sendall(sendData)

if action == GAME\_ACTION.FIRE.value:

BULLET\_DATA[player] = data["data"]

sendData = encode(

{

"action": GAME\_ACTION.FIRE.value,

"data": BULLET\_DATA[0] if player == 1 else BULLET\_DATA[1],

}

)

conn.sendall(sendData)

if action == GAME\_ACTION.SCORE.value:

SCORE\_DATA = data["data"]

sendData = encode({"action": GAME\_ACTION.SCORE.value, "data": SCORE\_DATA})

conn.sendall(sendData)

**Client**

* Đổi tượng **Bullet**

**import math**

**import time**

**import pygame**

**from setting import MAP, CELL\_SIZE**

**WIDTH = len(MAP[0]) \* CELL\_SIZE**

**HEIGHT = len(MAP) \* CELL\_SIZE**

**class Bullet:**

**def \_\_init\_\_(self, screen: pygame.Surface) -> None:**

**self.screen = screen**

**self.bulletX = 0**

**self.bulletY = 0**

**self.isFiring = False**

**self.angel = 0**

**self.startPos = {'x': 0, 'y': 0}**

**self.desPos = [0, 0]**

**self.speed = 6**

**self.fire\_time = None**

**def addFireAction(self, MAP):**

**if self.isFiring:**

**current\_time = time.time()**

**if (current\_time - self.fire\_time) > 5:**

**self.isFiring = False**

**next\_x = self.bulletX + self.speedX**

**next\_y = self.bulletY + self.speedY**

**# Kiểm tra va chạm trái/phải**

**if next\_x - 5 < 0 or next\_x + 5 > WIDTH or MAP[int(self.bulletY / CELL\_SIZE)][int(next\_x / CELL\_SIZE)] == 1:**

**self.speedX = -self.speedX**

**# Kiểm tra va chạm trên/dưới**

**if next\_y - 5 < 0 or next\_y + 5 > HEIGHT or MAP[int(next\_y / CELL\_SIZE)][int(self.bulletX / CELL\_SIZE)] == 1:**

**self.speedY = -self.speedY**

**self.bulletX += self.speedX**

**self.bulletY += self.speedY**

**pygame.draw.circle(self.screen, (255, 0, 0), (int(self.bulletX), int(self.bulletY)), 5)**

**def fire(self, startPos: dict, desPos: tuple):**

**if not self.isFiring:**

**self.bulletX = startPos['x']**

**self.bulletY = startPos['y']**

**self.startPos = startPos**

**self.desPos = desPos**

**self.isFiring = True**

**self.fire\_time = time.time()**

**dx = self.desPos[0] - self.startPos['x']**

**dy = self.desPos[1] - self.startPos['y']**

**distance = math.sqrt(dx\*\*2 + dy\*\*2)**

**self.speedX = self.speed \* (dx / distance)**

**self.speedY = self.speed \* (dy / distance)**

**def getRect(seft):**

**return pygame.Rect(seft.bulletX, seft.bulletY, 5, 5)**

**def update(self, x: int, y: int):**

**self.bulletX = x**

**self.bulletY = y**

**pygame.draw.circle(self.screen, (255,0,0), (self.bulletX, self.bulletY), 5)**

* Đối tượng **Tank**

**import math**

**import time**

**import pygame**

**from network import Network**

**from objects.bullet import Bullet**

**from utils.encode import encode**

**from utils.decode import decode**

**from setting import MAP, CELL\_SIZE**

**class Tank:**

**def \_\_init\_\_(**

**self, screen: pygame.Surface, network: Network, x, y, player, currPlayer**

**):**

**self.screen = screen**

**self.tankX = x**

**self.tankY = y**

**self.speed = 4**

**self.width = 20**

**self.height = 20**

**self.bullet = Bullet(screen)**

**self.player = player**

**self.currPlayer = currPlayer**

**self.network = network**

**def checkCollision(self, new\_x, new\_y, MAP):**

**tank\_rect = pygame.Rect(new\_x, new\_y, self.width + 10, self.height + 10)**

**for y, row in enumerate(MAP):**

**for x, cell in enumerate(row):**

**if cell == 1:**

**wall\_rect = pygame.Rect(**

**x \* CELL\_SIZE, y \* CELL\_SIZE, CELL\_SIZE, CELL\_SIZE**

**)**

**if tank\_rect.colliderect(wall\_rect):**

**return True**

**return False**

**def addMovement(self, MAP):**

**if self.player == self.currPlayer:**

**keys = pygame.key.get\_pressed()**

**new\_x, new\_y = self.tankX, self.tankY**

**if keys[pygame.K\_a] and self.tankX > 0:**

**new\_x -= self.speed**

**if keys[pygame.K\_d] and self.tankX + self.width < self.screen.get\_width():**

**new\_x += self.speed**

**if keys[pygame.K\_w] and self.tankY > 0:**

**new\_y -= self.speed**

**if keys[pygame.K\_s] and self.tankY + self.height < self.screen.get\_height():**

**new\_y += self.speed**

**if not self.checkCollision(new\_x, new\_y, MAP):**

**self.updateMovement(new\_x, new\_y)**

**self.updateMovement(self.tankX, self.tankY)**

**def updateMovement(self, x: int, y: int):**

**self.tankX = x**

**self.tankY = y**

**pygame.draw.circle(**

**self.screen, (255, 0, 0), (self.tankX + 15, self.tankY + 15), 15**

**)**

**def addShooting(self):**

**if self.player == self.currPlayer:**

**self.bullet.addFireAction(MAP)**

**mouse = pygame.mouse.get\_pressed()**

**if mouse[0]:**

**mouse\_pos = pygame.mouse.get\_pos()**

**self.bullet.fire({"x": self.tankX+ self.width // 2, "y": self.tankY+ self.height // 2}, mouse\_pos)**

**def addHitting(self, bullet: Bullet):**

**# tankRect = pygame.Rect(self.tankX, self.tankY, self.width + 5, self.height + 5)**

**# bulletRect = bullet.getRect()**

**# collide = bulletRect.colliderect(tankRect)**

**distance1 = math.sqrt((self.tankX + 15 - bullet.bulletX) \*\* 2 + (self.tankY + 15 - bullet.bulletY) \*\* 2)**

**now = time.time()**

**distance2 = math.sqrt((self.tankX + 15 - self.bullet.bulletX) \*\* 2 + (self.tankY + 15 - self.bullet.bulletY) \*\* 2)**

**isFiredAWhile = False**

**if self.bullet.fire\_time != None:**

**isFiredAWhile = (now - self.bullet.fire\_time) > 1**

**collide = distance1 <= (10 + 5) or (distance2 <= (10 + 5) and isFiredAWhile)**

**return collide**

* Hàm hỗ trợ

Hàm vẽ map lên màn hình

def drawMap(screen):

GRAY = (150, 150, 150)

for y, row in enumerate(MAP):

for x, cell in enumerate(row):

if cell == 1:

pygame.draw.rect(

screen, GRAY, (x \* CELL\_SIZE, y \* CELL\_SIZE, CELL\_SIZE, CELL\_SIZE)

)

Hàm hiển thị điểm lên màn hình:

def drawScrore(screen: pygame.Surface, myScore, otherScore):

text\_color = (255, 255, 255)

text = f"Your score: {myScore} - Opponent score: {otherScore}"

font = pygame.font.Font(None, 25)

text\_surface = font.render(text, True, text\_color)

text\_rect = text\_surface.get\_rect()

text\_rect.center = (screen.get\_width() / 2, 15)

screen.blit(text\_surface, text\_rect)

**Vòng lặp game:**

* Kết nối socket
* Gửi data local cho server
* Nhận data từ server và update
* Lắng nghe sự kiện từ chuột và bàn phím để điều khiển nhân vật di chuyển và bắn đạn
* Kiểm tra xem nhân vật có bị dính đạn hay không, nếu có, cập nhật bảng điểm và reset game
* Nếu 1 trong 2 người chơi ghi được 10 điểm trước thì dừng trò chơi và hiển thị thông báo

def startGame():

pygame.init()

screen\_width = len(MAP[0]) \* CELL\_SIZE

screen\_height = len(MAP) \* CELL\_SIZE

screen = pygame.display.set\_mode((screen\_width, screen\_height))

pygame.display.set\_caption("Shooting game")

clock = pygame.time.Clock()

network = Network()

currPlayer = network.getStartData()["data"]["currPlayer"]

print(f"Your player id {currPlayer}")

tank0 = Tank(screen, network, 60, 60, 0, currPlayer)

tank1 = Tank(

screen, network, screen\_width - 100, screen\_height - 100, 1, currPlayer

)

tank0Score = 0

tank1Score = 0

try:

running = True

while running:

for event in pygame.event.get():

if event.type == pygame.QUIT:

running = False

pygame.quit()

if tank0Score == 10 or tank1Score == 10:

screen.fill("black")

text = ""

if tank0Score == 10 and currPlayer == 0:

text = "You win !!!"

else:

text = "You lose !!!"

text\_color = (255, 255, 255)

font = pygame.font.Font(None, 75)

text\_surface = font.render(text, True, text\_color)

text\_rect = text\_surface.get\_rect()

text\_rect.center = (screen.get\_width() / 2, screen.get\_height() / 2)

screen.blit(text\_surface, text\_rect)

pygame.display.flip()

clock.tick(60)

continue

screen.fill("white")

drawMap(screen)

tank0.addMovement(MAP)

tank0.addShooting()

isTank0Hitting = tank0.addHitting(tank1.bullet)

tank1.addMovement(MAP)

tank1.addShooting()

isTank1Hitting = tank1.addHitting(tank0.bullet)

sendMoveData = {

"action": GAME\_ACTION.MOVE.value,

"data": {

"player": 1 if currPlayer == 1 else 0,

"x": tank1.tankX if currPlayer == 1 else tank0.tankX,

"y": tank1.tankY if currPlayer == 1 else tank0.tankY,

},

}

receiveMoveData = network.send(sendMoveData)

if len(receiveMoveData) > 0:

if receiveMoveData.get("action") == GAME\_ACTION.MOVE.value:

data = receiveMoveData.get("data")

if data["player"] == 0:

tank0.updateMovement(data["x"], data["y"])

else:

tank1.updateMovement(data["x"], data["y"])

sendBulletData = {

"action": GAME\_ACTION.FIRE.value,

"data": {

"player": 1 if currPlayer == 1 else 0,

"x": (

tank1.bullet.bulletX

if currPlayer == 1

else tank0.bullet.bulletX

),

"y": (

tank1.bullet.bulletY

if currPlayer == 1

else tank0.bullet.bulletY

),

},

}

receiveBulletData = network.send(sendBulletData)

if len(receiveBulletData) > 0:

if receiveBulletData.get("action") == GAME\_ACTION.FIRE.value:

data = receiveBulletData.get("data")

if data["player"] == 0:

tank0.bullet.update(data["x"], data["y"])

if data["x"] + data["y"] == 0:

tank0.bullet.isFiring = False

else:

tank1.bullet.update(data["x"], data["y"])

if data["x"] + data["y"] == 0:

tank1.bullet.isFiring = False

sendScoreData = {

"action": GAME\_ACTION.SCORE.value,

"data": [tank0Score, tank1Score],

}

if isTank0Hitting:

sendScoreData["data"][1] += 1

if isTank1Hitting:

sendScoreData["data"][0] += 1

receiveScoreData = network.send(sendScoreData)

if len(receiveScoreData) > 0:

if receiveScoreData.get("action") == GAME\_ACTION.SCORE.value:

data = receiveScoreData.get("data")

if data[0] != tank0Score or data[1] != tank1Score:

pygame.time.wait(1000)

tank0Score = data[0]

tank1Score = data[1]

tank0.updateMovement(60, 60)

tank0.bullet.update(0, 0)

tank0.bullet.isFiring = False

tank1.updateMovement(700, 500)

tank1.bullet.update(0, 0)

tank1.bullet.isFiring = False

drawScrore(

screen,

tank0Score if currPlayer == 0 else tank1Score,

tank1Score if currPlayer == 0 else tank0Score,

)

pygame.display.flip()

clock.tick(60)

# client\_socket.close()

finally:

running = False

# client\_socket.close()

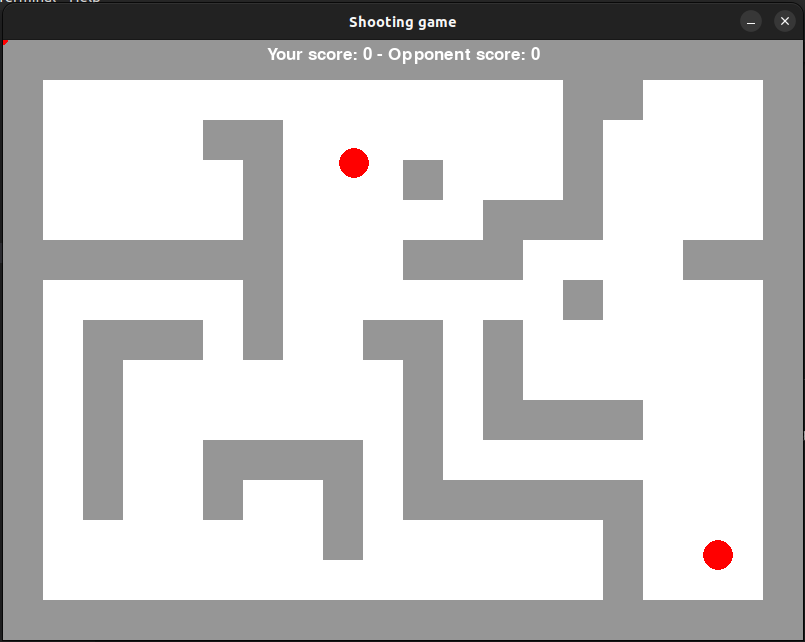
pygame.quit()

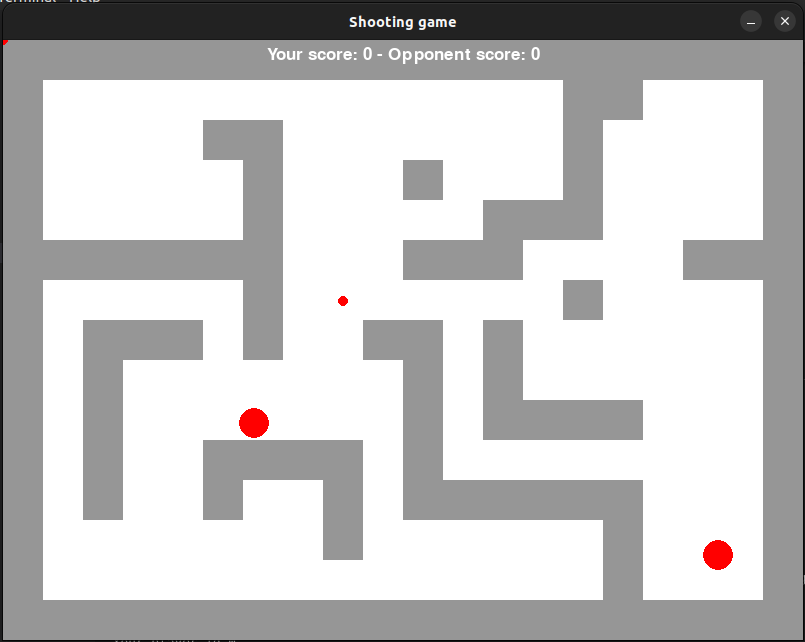
def main():

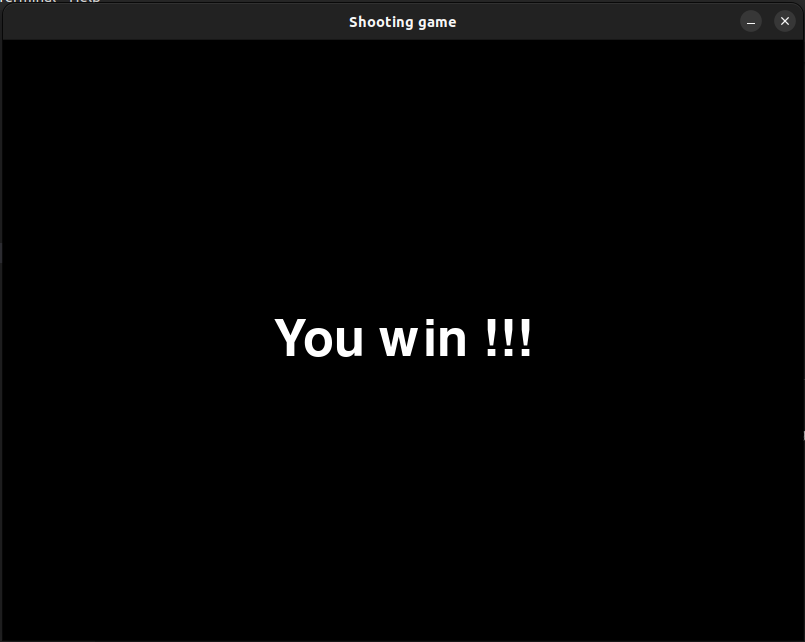
# s = initSocket()

startGame()

main()







## 

## Source code

**Shooting game**: <https://github.com/duccanhole/shooting-game>

# Kết luận

Sau quá trình tìm hiểu, em nhận thấy rằng Pygame là một bước khởi đầu tuyệt vời cho những ai đam mê lập trình game. Nó không chỉ cung cấp kiến thức về lập trình mà còn rèn luyện kỹ năng tư duy phản biện và khả năng sáng tạo. Khi đã nắm vững Pygame, người học có thể tự tin tiến xa hơn trong lĩnh vực phát triển game, khám phá những công nghệ mới và tạo ra những sản phẩm độc đáo, góp phần vào sự phát triển của ngành công nghiệp game đang ngày càng mở rộng. Không chỉ vậy, Pygame còn mang lại nhiều kỹ năng hữu ích trong việc phát triển kỹ năng lập trình.

Về mặt kỹ thuật, Pygame sở hữu những ưu điểm nổi bật như khả năng xử lý đồ họa mượt mà, hỗ trợ đa nền tảng, và cộng đồng phát triển năng động. Điều này giúp các lập trình viên, đặc biệt là những người mới bắt đầu, có thể dễ dàng tiếp cận và phát triển các ứng dụng đồ họa một cách nhanh chóng và hiệu quả.

Những ứng dụng của Pygame không chỉ giới hạn ở game development mà còn mở rộng sang các lĩnh vực như mô phỏng, giáo dục, và các ứng dụng tương tác đa phương tiện. Tính linh hoạt và dễ sử dụng của thư viện đã chứng minh sức mạnh của Python trong việc xây dựng các ứng dụng đồ họa sống động.

Tuy nhiên, bên cạnh những ưu điểm, Pygame cũng có những hạn chế nhất định như hiệu năng kém so với các game engine chuyên nghiệp và một số giới hạn trong việc xử lý đồ họa 3D. Những hạn chế này mở ra những cơ hội nghiên cứu và phát triển cho các lập trình viên trong tương lai.

Sau khi thành thạo việc sử dụng thư viện Pygame, người học có thể cân nhắc tìm hiểu các công nghệ làm game khác như:

* Unity: là một trong những công cụ phát triển game phổ biến nhất hiện nay, được sử dụng để tạo ra các trò chơi 2D và 3D cho nhiều nền tảng khác nhau, bao gồm máy tính, di động, và console. Được phát triển bởi Unity Technologies, Unity cung cấp một môi trường phát triển tích hợp (IDE) thân thiện với người dùng, cho phép lập trình viên và nhà thiết kế dễ dàng xây dựng, kiểm tra và triển khai các trò chơi của mình.
* Godot: là một công cụ phát triển game mã nguồn mở, miễn phí, cho phép người dùng tạo ra các trò chơi 2D và 3D. Được phát triển bởi cộng đồng Godot Engine, nó nổi bật với giao diện thân thiện và khả năng tùy biến cao, phù hợp cho cả lập trình viên mới bắt đầu và những người có kinh nghiệm
* Unreal Engine: là một trong những công cụ phát triển game mạnh mẽ và phổ biến nhất trong ngành công nghiệp game, được phát triển bởi Epic Games. Được biết đến với khả năng tạo ra đồ họa 3D chất lượng cao và hiệu suất vượt trội, Unreal Engine không chỉ được sử dụng để phát triển game mà còn được áp dụng trong nhiều lĩnh vực khác nhau như phim ảnh, kiến trúc, và thực tế ảo (VR).
* Phaser: là một framework phát triển game mã nguồn mở dựa trên JavaScript, được thiết kế đặc biệt cho việc tạo ra các trò chơi 2D trên trình duyệt. Với sự hỗ trợ mạnh mẽ cho các công nghệ web hiện đại, Phaser cho phép lập trình viên dễ dàng xây dựng và triển khai các trò chơi hấp dẫn trên nhiều nền tảng khác nhau, bao gồm máy tính và thiết bị di động.

# Tài liệu tham khảo

Trang tài liệu chính thức của Pygame: <https://www.pygame.org/docs/>

Ebook:

* <https://inventwithpython.com/makinggames.pdf>
* <https://github.com/CleverProgrammer/CS-Books>