**ỦY BAN NHÂN DÂN THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC SÀI GÒN**

**NHÓM 08**

**XÂY DỰNG GAME PIXEL WAR**

**SỬ DỤNG THƯ VIỆN PYGAME**

**BÁO CÁO ĐỒ ÁN MÔN HỌC**

**NGÔN NGỮ LẬP TRÌNH PYTHON**

NGÀNH: CÔNG NGHỆ THÔNG TIN

**Thành phố Hồ Chí Minh, tháng 4 năm 2024**

**ỦY BAN NHÂN DÂN THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC SÀI GÒN**

**NHÓM 08**

**XÂY DỰNG GAME PIXEL WAR**

**SỬ DỤNG THƯ VIỆN PYGAME**

**BÁO CÁO ĐỒ ÁN MÔN HỌC**

**NGÔN NGỮ LẬP TRÌNH PYTHON**

ngành: CÔNG NGHỆ THÔNG TIN

**Giảng viên phụ trách**

**TS. TRỊNH TẤN ĐẠT**

**Thành phố Hồ Chí Minh, tháng 4 năm 2024**

# **Lời cam đoan**

Em tên là Võ Đăng Quang đại diện nhóm 08, em xin cam đoan rằng đồ án “*Xây dựng game Pixel War sử dụng thư viện Pygame*” là công trình nghiên cứu của nhóm dưới sự hướng dẫn của TS. Trịnh Tấn Đạt.

Mọi trích dẫn sử dụng trong báo cáo đều được ghi rõ nguồn tài liệu tham khảo theo đúng quy định.

Em xin hoàn toàn chịu trách nhiệm và chịu mọi hình thức kỷ luật theo quy định nếu có bất kì hành vi vi phạm, gian trá nào.

**Thành phố Hồ Chí Minh, tháng 4 năm 2024**

**Nhóm 08**

# **Lời cảm ơn**

Trước hết, nhóm 08 chúng em xin giành lời cảm ơn đến quý thầy cô Trường Đại học Sài Gòn – khoa Công nghệ thông tin đã truyền đạt cho em những kiến thức vô cùng quý báu và bổ ích trong suốt quá trình nghiên cứu và học tập tại trường.

Tiếp đó, nhóm chúng em xin gửi lời cảm ơn chân thành và sâu sắc nhất đến cô TS. Trịnh Tấn Đạt, người trực tiếp hướng dẫn và tận tình chỉ bảo cho tới khi nhóm hoàn thành đồ án.

Cuối cùng, cảm ơn những người bạn trong nhóm đã luôn hỗ trợ những lúc cần thiết, giúp nhau hoàn thành đồ án này.

# **Mục lục**

[**Lời cam đoan** i](#_Toc163657263)

[**Lời cảm ơn** ii](#_Toc163657264)

[**Mục lục** iii](#_Toc163657265)

[**Danh mục hình ảnh** v](#_Toc163657266)

[**Lời mở đầu** 1](#_Toc163657267)

[**Chương 1. SƠ LƯỢC VỀ ĐỀ TÀI** 2](#_Toc163657268)

[**1.1.** **Tổng quan Python** 2](#_Toc163657269)

[**1.1.1.** **Lịch sử phát triển** 2](#_Toc163657270)

[**1.1.2.** **So sánh Python với các ngôn ngữ lập trình khác** 2](#_Toc163657271)

[**1.2.** **Giới thiệu về thư viện Pygame** 4](#_Toc163657272)

[**1.2.1.** **Tính năng của Pygame:** 4](#_Toc163657273)

[**1.2.2.** **Ưu điểm của Pygame:** 4](#_Toc163657274)

[**Chương 2. Phân tích và thiết kế** 5](#_Toc163657275)

[**2.1.** **Use case Diagram** 5](#_Toc163657276)

[**2.2.** **Đặc tả Use case** 5](#_Toc163657277)

[**2.3.** **Sequence Diagram** 5](#_Toc163657278)

[**2.4.** **Các thư viện cơ bản được sử dụng trong trò chơi** 5](#_Toc163657279)

[**2.5.** **Cấu trúc của trò chơi** 5](#_Toc163657280)

[**Chương 3. XÂY DỰNG TRÒ CHƠI VÀ DEMO** 6](#_Toc163657281)

[**3.1.** **Xây dựng trò chơi** 6](#_Toc163657282)

[**3.1.1.** **Thiết lập khung nhìn** 6](#_Toc163657283)

[**3.1.2.** **Thiết lập môi trường** 7](#_Toc163657284)

[**3.1.3.** **Xây dựng background** 8](#_Toc163657285)

[**3.1.4.** **Khởi tạo hitbox** 8](#_Toc163657286)

[**3.1.5.** **Khởi tạo camera di chuyển theo nhân vật và background** 10](#_Toc163657287)

[**3.1.6.** **Xây dựng nhân vật** 16](#_Toc163657288)

[**3.1.7.** **Hệ thống vũ khí và phép thuật** 16](#_Toc163657289)

[**3.1.8.** **Hệ thống nâng cấp nhân vật** 16](#_Toc163657290)

[**3.1.9.** **Xây dựng quái** 16](#_Toc163657291)

[**3.2.** **Demo** 16](#_Toc163657292)

[**3.2.1.** **Giao diện chính** 16](#_Toc163657293)

[**3.2.2.** **Di chuyển** 17](#_Toc163657294)

[**3.2.3.** **Tấn công** 17](#_Toc163657295)

[**3.3.** **Link source code** 17](#_Toc163657296)

[**KẾT QUẢ ĐẠT ĐƯỢC VÀ HƯỚNG PHÁT TRIỂN** 18](#_Toc163657297)

[**Kết quả đạt được** 18](#_Toc163657298)

[**Hướng phát triển** 18](#_Toc163657299)

[**TÀI LIỆU THAM KHẢO** 19](#_Toc163657300)

# **Danh mục hình ảnh**

[Hình 0.1. Nhân viên tư vấn sản phẩm cho khách hàng 1](#_Toc152421214)

[Hình 1.1. Shopee khuyến nghị sản phẩm dựa trên lịch sử tìm kiếm 3](#_Toc152421898)

[Hình 1.2. Các thành phần của hệ thống khuyến nghị 4](#_Toc152421899)

[Hình 1.3. Mô hình hệ thống khuyến nghị 7](#_Toc152421900)

[Hình 1.4. Kiến trúc tổng quan của tiếp cận nội dung 8](#_Toc152421901)

[Hình 1.5. Ma trận tiếp cận lọc cộng tác 11](#_Toc152421902)

[Hình 1.6. Mô hình tiếp cận lọc cộng tác dựa trên bộ nhớ 12](#_Toc152421903)

[Hình 1.7. Mô hình lọc lai ghép 14](#_Toc152421904)

[Hình 1.8. Hệ thống nhà hàng Entree Chicago 17](#_Toc152421905)

[Hình 1.9. Mô hình hệ thống Libra 18](#_Toc152421906)

[Hình 2.1. Mô hình cho kỹ thuật phân rã ma trận 25](#_Toc152422287)

[Hình 2.2. Sản phẩm tìm kiếm 29](#_Toc152422288)

[Hình 2.3. Hệ thống gợi ý sản phẩm 29](#_Toc152422289)

[Hình 2.4. Mô hình xử lý ngữ cảnh 31](#_Toc152422290)

[Hình 2.5. Sau khi xử lý ngữ cảnh đầu vào 32](#_Toc152422291)

[Hình 3.1. Buổi thảo luận của ông Zhong Nan 36](#_Toc152867359)

[Hình 3.2. Thanh tìm kiếm của Shopee 37](#_Toc152867360)

[Hình 3.3. Hệ thống gợi ý sản phẩm khi chưa có thông tin người dùng 38](#_Toc152867361)

[Hình 3.4. Thanh tìm kiếm của Shopee sau khi có thông tin người dùng 38](#_Toc152867362)

[Hình 3.5. Shopee khuyến nghị sản phẩm sau khi có thông tin người dùng 38](#_Toc152867363)

[Hình 3.6. Chọn ốp lưng 39](#_Toc152867364)

[Hình 3.7. Hệ thống đề xuất sản phẩm 40](#_Toc152867365)

[Hình 3.8. Chương trình khuyến mãi của Lazada 41](#_Toc152867366)

[Hình 3.9. Thanh tìm kiếm của Lazada 42](#_Toc152867367)

[Hình 3.10. Hệ thống khuyến nghị khi chưa có thông tin người dùng 42](#_Toc152867368)

[Hình 3.11. Hệ thống khuyến nghị khi có thông tin người dùng 43](#_Toc152867369)

[Hình 3.12. Tìm kiếm sản phẩm 43](#_Toc152867370)

[Hình 3.13. Kết quả tìm kiếm 43](#_Toc152867371)

[Hình 3.14. Chọn xem sản phẩm 44](#_Toc152867372)

[Hình 3.15. Các sản phẩm được đề xuất sau hai kĩ thuật 44](#_Toc152867373)

[Hình 3.16. Doanh thu của Amazon từ năm 1997 đến 2021 45](#_Toc152867374)

[Hình 3.17. Amazon đề xuất cho người dùng các sản phẩm phổ biến 46](#_Toc152867375)

[Hình 3.18. Cung cấp thông tin cá nhân sau khi tạo tài khoản 47](#_Toc152867376)

[Hình 3.19. Amazon khảo sát sở thích người dùng 47](#_Toc152867377)

[Hình 3.20. Hệ thống đề xuất sản phẩm sau khi khảo sát sở thích 48](#_Toc152867378)

[Hình 3.21. Tìm kiếm sản phẩm 48](#_Toc152867379)

[Hình 3.22. Sản phẩm giống từ khoá 49](#_Toc152867380)

[Hình 3.23. Sản phẩm đề xuất gần giống từ khoá 49](#_Toc152867381)

[Hình 3.24. Hệ thống so sánh sản phẩm 50](#_Toc152867382)

[Hình 3.25. Danh mục mới được hệ thống tạo riêng cho khách hàng 50](#_Toc152867383)

# **Lời mở đầu**

Python là một ngôn ngữ có cấu trúc rõ ràng, thuận tiện cho người mới bắt đầu học lập trình. Đây là một ngôn ngữ lập trình đơn giản nhưng lại rất hiệu quả. Bên cạnh đó, Python có tính hướng đối tượng cao và được sử dụng rất nhiều cho các bài toán xử lý với dữ liệu lớn. Đây cũng là một ngôn ngữ tuyệt vời để tạo những nguyên mẫu (bản chạy thử – prototype). Ví dụ, bạn có thể sử dụng Pygame (thư viện viết game) để tạo nguyên mẫu game trước. Nếu thích nguyên mẫu đó có thể dùng C/C++ hay Java để viết game thật sự.

Trong bối cảnh công nghệ ngày càng phát triển như hiện nay thì nhiều thể loại game được nhiều ra đời ngày càng nhiều và thu hút được nhiều bạn trẻ. Có thể nói game đã và đang trở thành một nhu cầu giải trí thiết yếu trong cuộc sống của con người hiện nay. Một trong những thể loại game kinh điển mà khi nhắc tới không còn xa lạ gì với chúng ta, Role-Playing Game (RPG). Đây là thể loại game phổ biến, nơi người chơi điều khiển một nhân vật trong thế giới ảo, trải nghiệm các cuộc phiêu lưu, chiến đấu với kẻ thù và hoàn thành nhiệm vụ. RPG thu hút người chơi bởi sự đa dạng, khả năng sáng tạo và tính giải trí cao.

Sử dụng các kiến thức học được, nhóm chúng em chọn “Xây dựng game Pixel War sử dụng thư viện Pygame” để làm đồ án cho môn học này. Quá trình nghiên cứu nhóm em sẽ trình bày sơ lược về ngôn ngữ lập trình Python cũng như cấu trúc của nó. Kết hợp việc quan sát và thực hành để tạo ra một phiên bản RPG trực quan và đơn giản nhất cho đồ án của mình.

# **Chương 1. SƠ LƯỢC VỀ ĐỀ TÀI**

## **Tổng quan Python**

### **Lịch sử phát triển**

Vào đầu những năm 1980, ABC là một ngôn ngữ lập trình bậc cao, được phát triển bởi Leo Geurts và Lambert Meertens tại Đại học Amsterdam. ABC được thiết kế để trở thành một ngôn ngữ đơn giản, dễ học và dễ sử dụng, phù hợp cho việc giảng dạy lập trình. Đặc điểm của ABC là cú pháp đơn giản, dễ đọc, kiểu dữ liệu động, hỗ trợ lập trình cấu trúc và lập trình hướng đối tượng. ABC được sử dụng chủ yếu cho việc giảng dạy lập trình. ABC cũng có thể được sử dụng để phát triển các ứng dụng đơn giản.

Với mục tiêu đơn giản hoá hơn các cú pháp và tăng cường khả năng xử lý, Guido Van Rossum bắt đầu phát triển Python. Công cuộc thiết kế bắt đầu vào cuối những năm 1980 và vào tháng 2 năm 1991, Python được phát hành lần đầu tiên. Sau hơn 30 năm phát triển và cải tiến, Python đã trở thành một trong những ngôn ngữ lập trình phổ biến nhất hiện nay.

Python được sử dụng rộng rãi trong nhiều lĩnh vực khác nhau, bao gồm: Phát triển web, Khoa học dữ liệu, Học máy, Tự động hóa, Phát triển phần mềm,… Cộng đồng lớn và hoạt động tích cực cũng là một trong những lí do làm nên sự thành công của Python. Trong hơn 30 năm phát triển, Python có một thư viện tiêu chuẩn phong phú với nhiều chức năng sẵn có và được cấp nhật thường xuyên các tính năng mới như: numpy, scipy, matplotlib , sklearn,…

### **So sánh Python với các ngôn ngữ lập trình khác**

Python, Java và C++ là ba ngôn ngữ lập trình phổ biến nhất hiện nay. Mỗi ngôn ngữ đều có những ưu điểm và nhược điểm riêng, phù hợp với những mục đích sử dụng khác nhau.

Dưới đây là bảng so sánh chi tiết giữa Python, Java và C++:

|  |  |  |  |
| --- | --- | --- | --- |
| Tiêu chí | Python | Java | C++ |
| Cú pháp | Dễ đọc, dễ học | Dễ đọc, dễ học | Khó đọc, khó học |
| Kiểu dữ liệu | Động | Tĩnh | Tĩnh |
| Quản lý bộ nhớ | Tự động | Tự động | Thủ công |
| Hiệu suất | Chậm | Nhanh | Nhanh |
| Khả năng mở rộng | Cao | Cao | Thấp |
| Thư viện tiêu chuẩn | Phong phú | Phong phú | Ít phong phú |
| Ứng dụng | Phát triển web, khoa học dữ liệu, học máy, tự động hóa | Phát triển web, ứng dụng di động, phần mềm doanh nghiệp | Hệ thống nhúng, trò chơi, phần mềm đồ họa |

Qua bản so sánh trên, ta có thể kết luận được rằng:

* Python là lựa chọn tốt nhất dành cho người mới học lập trình vì Python có cú pháp đơn giản và dễ học, giúp bạn dễ dàng tiếp cận với lập trình.
* Nếu muốn phát triển web, Java hoặc Python mới là sự lựa chọn tốt. Java và Python đều có các framework web mạnh mẽ và cộng đồng hỗ trợ lớn.
* Nếu muốn nghiên cứu khoa học dữ liệu hoặc training AI, thì Python là sự lựa chọn tối ưu vì cú pháp và các thư viện hỗ trợ rất tốt cho mảng này.

## **Giới thiệu về thư viện Pygame**

Pygame là một thư viện mã nguồn mở của Python được sử dụng để phát triển game. Nó cung cấp bộ công cụ cần thiết để tạo ra các thành phần cơ bản đến nâng cao. Kể cả khi là người mới cũng có thể dễ dàng sử dụng thư viện này.

### **Tính năng của Pygame:**

* Đồ họa: Pygame cung cấp các chức năng để vẽ hình ảnh, sprite và các hiệu ứng đồ họa khác.
* Âm thanh: Pygame cung cấp các chức năng để phát nhạc và âm thanh.
* Đầu vào: Pygame cung cấp các chức năng để xử lý các sự kiện đầu vào như nhấp chuột, di chuyển chuột và nhấn phím.
* Khả năng tương thích: Pygame có thể được sử dụng trên nhiều hệ điều hành khác nhau, bao gồm Windows, Mac và Linux.

### **Ưu điểm của Pygame:**

* Dễ học và sử dụng: Pygame có cú pháp đơn giản và dễ hiểu, giúp cho người mới bắt đầu dễ dàng tiếp cận.
* Linh hoạt: Pygame có thể được sử dụng để tạo ra nhiều loại trò chơi khác nhau, từ các trò chơi đơn giản đến các trò chơi phức tạp.
* Mã nguồn mở: Pygame là mã nguồn mở, nghĩa là nó miễn phí và có thể được sử dụng bởi bất kỳ ai.
* Cộng đồng lớn: Pygame có một cộng đồng lớn và hoạt động tích cực, luôn sẵn sàng hỗ trợ lẫn nhau.

# **Chương 2. Phân tích và thiết kế**

## **Use case Diagram**

## **Đặc tả Use case**

## **Sequence Diagram**

## **Các thư viện cơ bản được sử dụng trong trò chơi**

## **Cấu trúc của trò chơi**

# **Chương 3. XÂY DỰNG TRÒ CHƠI VÀ DEMO**

## **Xây dựng trò chơi**

### **Thiết lập khung nhìn**

Để sử dụng trò chơi ta cần khai báo 2 thư viện: pygame và sys ở file main.py.

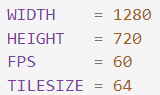


* pygame: Thư viện đồ hoạ và âm thanh cho game.
* sys: Thư viện builtin của Python, nó chứa các thông tin liên quan đến chính chương trình python interpreter bạn đang chạy.

Tiếp theo tạo file settings.py lưu tất cả các cấu hình của game vào. Sau đó import settings vào trong main.



Trong file settings.py ta có các cài đặt sau:



* WIDTH và HEIGHT: là thông số xác định kích thước cửa sổ trò chơi. Ví dụ, chiều rộng của cửa sổ là 1280 pixel và chiều cao là 720 pixel.
* FPS: là thông số thể hiện số lượng hình ảnh được hiển thị trên màn hình trò chơi mỗi giây, FPS càng cao thì game hiển thị càng mượt. Trong ví dụ này, trò chơi sẽ được vẽ lại 60 lần mỗi giây.
* TILESIZE: là kích thước của nền trò chơi

### **Thiết lập môi trường**

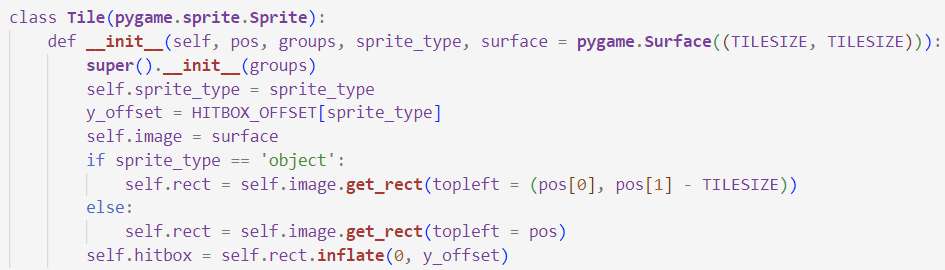


* pygame.init(): dòng này khởi tạo thư viện Pygame, thư viện cần thiết để tạo trò chơi trong Python. Hàm này phải được gọi trước khi sử dụng bất kỳ chức năng nào khác của Pygame.
* pygame.display.set\_mode((WIDTH, HEIGHT)): kích thước cửa sổ của game sẽ được xác định bởi WIDTH và HEIGHT (1280 pixel x 720 pixel).
* pygame.display.set\_caption('PixelWar Project'): Dòng này đặt tiêu đề cho cửa sổ trò chơi hiển thị ở đầu khung cửa sổ. Tiêu đề trong trường hợp này được đặt thành "PixelWar Project".
* self.clock = pygame.time.Clock(): Dòng này tạo một đối tượng đồng hồ bằng cách sử dụng mô-đun pygame.time. Đồng hồ này được sử dụng để quản lý tốc độ khung hình của trò chơi, đảm bảo trải nghiệm hoạt hình mượt mà.
* self.level = Level(): Tạo một lớp Level để quản lý các vấn đề liên quan đến cấp độ trong game và gán nó cho biến self.level trong lớp hiện tại.
* Các dòng này tiếp theo được sử dụng để thiết lập nhạc nền cho trò chơi:
* main\_sound = pygame.mixer.Sound('../audio/main.mp3'): Dòng này tải tệp âm thanh (main.mp3) nằm trong thư mục con tên "audio" liên quan đến tập lệnh Python.
* main\_sound.set\_volume(0.5): Dòng này đặt âm lượng của tệp âm thanh thành 50% âm lượng của máy.
* main\_sound.play(loops = -1): Dòng này để đặt tệp âm thanh lặp lại vô hạn (loops = -1).

### **Xây dựng background**

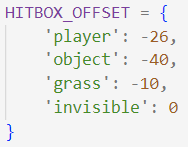
### **Khởi tạo hitbox**

Tạo 1 file tên tile.py để chứa các thiết lập liên quan đến hitbox trong game. Mỗi đối tượng (nhân vật, quái, cây cỏ, tường,…) là 1 ô vuông (tile) trong game. Để các đối tượng có thể tương tác được với nhau lớp Tile sẽ được sẽ sử dụng để quản lý chúng.



* def \_\_init\_\_(self, pos, groups, sprite\_type, surface = pygame.Surface((TILESIZE, TILESIZE))): Đây là hàm khởi tạo của lớp Tile, trong đó:
* pos: Tham số này là một tuple chứa vị trí (x, y) của ô tile trên màn hình.
* groups: Tham số này là một hoặc nhiều nhóm sprite mà ô tile sẽ được thêm vào.
* sprite\_type: Chuỗi ký tự xác định loại của ô tile (ví dụ: 'ground', 'wall', 'object').
* surface (tham số tuỳ chọn): Tham số này để tạo ô tile cho cửa sổ trò chơi lấy kích thước của nền trò chơi làm mặc định.
* super().\_\_init\_\_(groups): Hàm khởi tạo của lớp cha pygame.sprite.Sprite, truyền tham số groups để thêm ô tile vào các nhóm sprite đã cho.

Trở lại với file settings.py ta tiếp tục thêm HITBOX\_OFFSET để thêm vị trí của hitbox cho các loại ô tile khác nhau:



Về phía file tile.py:

* y\_offset = HITBOX\_OFFSET[sprite\_type]: Đây là lấy giá trị lấy từ settings.py là HITBOX\_OFFSET với key là sprite\_type gán cho biến y\_offset.
* self.image = surface: Thuộc tính này lưu trữ hình ảnh đại diện cho ô tile.

Cách các hitbox hoạt động như sau:

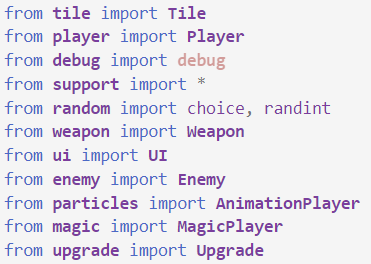
* Trường hợp sprite\_type là 'object':
* self.rect = self.image.get\_rect(topleft = (pos[0], pos[1] - TILESIZE)): Tạo một hình chữ nhật bao quanh image (được lưu trong self.rect) và đặt vị trí trên cùng bên trái (topleft) của hình chữ nhật tại tọa độ (pos[0], pos[1] - TILESIZE).
* Trường hợp khác (sprite\_type không phải 'object'):
* self.rect = self.image.get\_rect(topleft = pos): Tạo một hình chữ nhật bao quanh image và đặt vị trí trên cùng bên trái của hình chữ nhật tại tọa độ là pos.

Kết thúc vòng lặp

* self.hitbox = self.rect.inflate(0, y\_offset): Tạo hitbox dựa trên self.rect và nới rộng vùng này theo trục Y (y\_offset) để tính toán hitbox chính xác hơn. Giá trị y\_offset được lấy từ HITBOX\_OFFSET tùy theo loại ô tile đã khai báo ở settings.py.

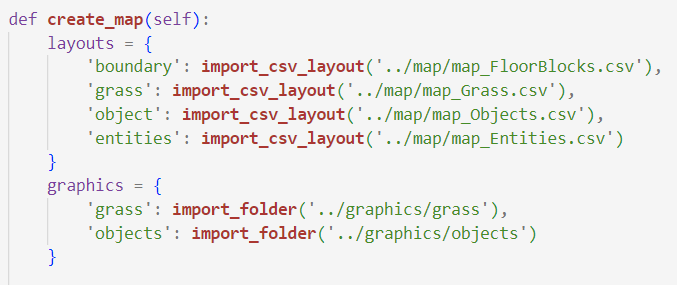
### **Khởi tạo camera di chuyển theo nhân vật và background**

Tạo 1 file level.py để chứa tất cả các vấn đề liên quan đến cấp độ trong trò chơi. Trong file này ta cần các thư viện cũng như các file sau để trò chơi được vận hành một cách trơn tru nhất.



* tile: ta cần file tile.py đã tạo trước đó để sử dụng hitbox.
* player: tạo 1 file player.py để chứa các thiết lập liên quan đến nhân vật.
* debug: tạo 1 file debug.py để hiện các thông báo của trò chơi trên màn hình máy.
* support: tạo 1 file support.py để quản lý các file csv và hình ảnh từ các thư mục khác nhau.
* random: sử dụng thư viện random để chọn ngẫu nhiên các object (vật thể) như cỏ (grass).
* ui: tạo 1 file ui.py để quản lý các giao diện trong trò chơi.
* enemy: tạo 1 file enemy.py để chứa các thiết lập liên quan tới quái.
* particles: tạo 1 file particles.py để chứa các thiết lập liên quan đến chuyển động của nhân vật.
* magic: tạo 1 file magic.py để chứa các thiết lập phép thuật của nhân vật.
* upgrade: tạo 1 file upgrade.py để chứa các thiết lập của hệ thống nâng cấp nhân vật.

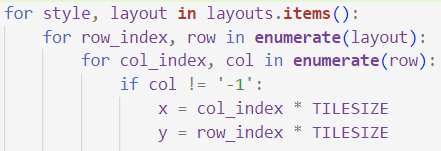
Khởi tạo hàm create\_map để tạo ra bố cục và sắp xếp các yếu tố khác nhau trong trò chơi.



Hàm bắt đầu bằng việc xác định: layouts và graphics.

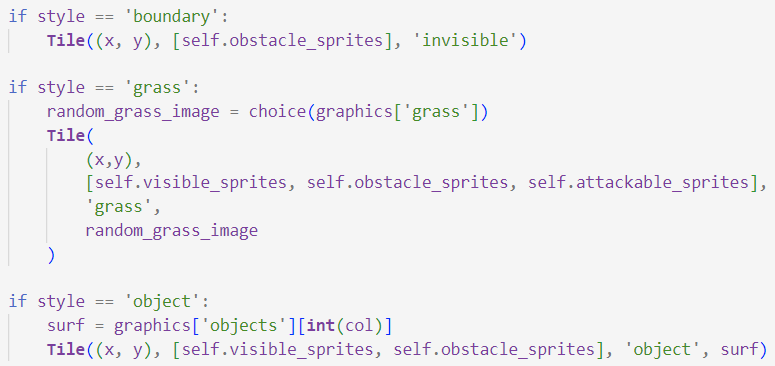
* layouts: là khóa đại diện cho các kiểu ô khác nhau như "boundary", "grass", "object" và "entities". Giá trị là kết quả của việc gọi hàm import\_csv\_layout , hàm này đọc dữ liệu từ các tệp CSV tương ứng.
* graphics: là khoá để lưu trữ hình ảnh của các loại cỏ (grass) và vật thể (objects) đã được lưu trước đó.

Tạo vòng lặp qua các file csv đã đọc ở trên:



* row\_index: Chỉ mục của hàng hiện tại đang được xử lý.
* row: Danh sách thực tế đại diện cho hàng ô hiện tại.
* col\_index: Chỉ mục của cột hiện tại trong hàng.
* col: Giá trị tại vị trí ô hiện tại trong bố cục.
* if col != '-1': Bỏ qua việc xử lý các ô trống được đánh dấu bằng '-1' trong csv.

Sau khi đã kiểm tra và xử lý file csv, tiếp theo ta tạo ô tile dựa trên style. Tùy thuộc vào style, các ô tile khác nhau sẽ được tạo:



* boundary: Nếu style là "boundary", một ô tile được tạo với đồ họa trống nhưng được đánh dấu là chướng ngại vật (invisible hitbox).
* grass: Nếu style là "grass", một ô tile được tạo với hình ảnh cỏ ngẫu nhiên được hiển thị trên cửa sổ.
* object: Nếu style là "object", một ô tile được tạo với bề mặt hình ảnh tương ứng được lấy từ graphics['objects'] sử dụng giá trị từ file csv và chỉ mục là [int(col)].

Tiếp theo ta sẽ hiển thị vị trí thực thể gồm nhân vật và quái vật. Trong trò chơi này khi được bắt đầu, nhân vật và quái sẽ đứng ở một vị trí cố định:



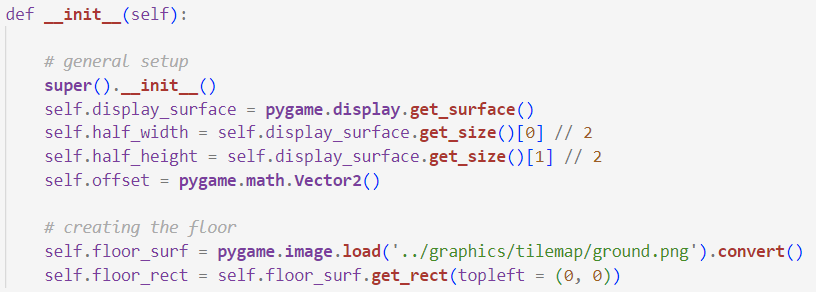
Nếu style là "entities" thì sẽ tiến hành kiểm tra trong file csv giá trị ô hiện tại (col).

* Nhân vật col mặc định sẽ là "394", một ô tile dành cho nhân vật được tạo bao gồm các thuộc tính sau:
* Vị trí cố định.
* Lớp sprite (hiển thị, chướng ngại vật)
* Chức năng tấn công/phá hủy vật thể/sử dung phép thuật.
* Quái vật: có các col như sau: "390" là bamboo, "391" là spirit, "392" là raccoon và "393" là squid. Khi kiểm tra đúng col thì ô tile cho quái được tạo với các thuộc tính sau:
* Loại quái vật dựa trên giá trị col.
* Vị trí dựa trên chỉ mục.
* Lớp sprite hiển thị, có thể tấn công và chướng ngại vật.
* Chức năng gây sát thương cho nhân vật, kích hoạt hiệu ứng chết và thưởng điểm kinh nghiệm cho nhân vật.

Sau khi tạo xong nền cho nhân vật và quái có thể tương tác với các vật thể xung quanh thì trò chơi đã hoàn thành được nửa chặng đường. Bây giờ ta cần xây dựng một hệ thống camera di chuyển theo nhân vật nếu không thì ta không thể điều khiển nhân vật nếu nó đi khỏi tầm mắt của màn hình. Để làm được điều đó ta khởi tạo hàm YsortCameraGroup.



Trong đó khởi tạo phương thức \_\_init\_\_ để định nghĩa các đối tượng sẽ được sử dụng.



* super().\_\_init\_\_(): Dòng này gọi hàm khởi tạo của lớp cha (pygame.sprite.Group), đảm bảo khởi tạo đúng để quản lý các sprite.
* self.display\_surface = pygame.display.get\_surface(): Dòng này lấy bề mặt hiển thị chính nơi trò chơi được hiển thị.
* self.half\_width = self.display\_surface.get\_size()[0] // 2: Dòng này tính toán một nửa chiều rộng của bề mặt hiển thị, có thể được sử dụng để căn giữa camera.
* self.half\_height = self.display\_surface.get\_size()[1] // 2: Tương tự như trên, dòng này tính toán một nửa chiều cao của bề mặt hiển thị.
* self.offset = pygame.math.Vector2(): Dòng này tạo một đối tượng pygame.math.Vector2 để lưu trữ độ lệch camera.
* self.floor\_surf = pygame.image.load('../graphics/tilemap/ground.png').convert(): Dòng này tải ảnh nền và chuyển đổi để hiển thị nhanh hơn.
* self.floor\_rect = self.floor\_surf.get\_rect(topleft = (0, 0)): Dòng này lấy một hình chữ nhật biểu thị vị trí và kích thước của ảnh nền.

Sau khi có các định nghĩa ở trên ta khởi tạo phương thức custom\_draw để tính toán độ sai lệch camera trên trục X và Y sau đó mới tiến hành vẽ hình ảnh của nhân vật lên bề mặt đã được tính toán trước:



* self.offset.x = player.rect.centerx - self.half\_width: Dòng này tính toán độ lệch camera trên trục X bằng cách trừ một nửa chiều rộng hiển thị khỏi vị trí X trung tâm của nhân vật. Điều này làm cho camera tập trung vào nhân vật.
* self.offset.y = player.rect.centery - self.half\_height: Tương tự như trên, dòng này tính toán độ lệch camera trên trục Y dựa trên vị trí Y trung tâm của nhân vật.
* floor\_offset\_pos = self.floor\_rect.topleft - self.offset: Dòng này tính toán vị trí của ảnh nền trên màn hình bằng cách trừ độ lệch camera khỏi góc trên bên trái ban đầu của nó.
* self.display\_surface.blit(self.floor\_surf, floor\_offset\_pos): Dòng này sao chép (vẽ) ảnh nền lên bề mặt hiển thị chính tại vị trí được tính toán.
* for sprite in sorted(self.sprites(), key = lambda sprite: sprite.rect.centery): Dòng này lặp qua các sprite trong nhóm, nhưng trước tiên, chúng được sắp xếp dựa trên vị trí Y trung tâm của chúng (sử dụng hàm lambda làm khóa).
* offset\_pos = sprite.rect.topleft - self.offset: Tương tự như nền, dòng này tính toán vị trí trên màn hình của mỗi sprite bằng cách trừ độ lệch camera khỏi vị trí ban đầu của nó.
* self.display\_surface.blit(sprite.image, offset\_pos): Dòng này sao chép từng sprite lên bề mặt hiển thị chính tại vị trí được tính toán tương ứng. Việc sắp xếp này đảm bảo rằng các sprite ở xa hơn (trung tâm Y cao hơn) được vẽ phía sau các sprite gần hơn.

Cuối cùng, tạo phương thức enemy\_update để xử lý hành vi của quái với nhân vật



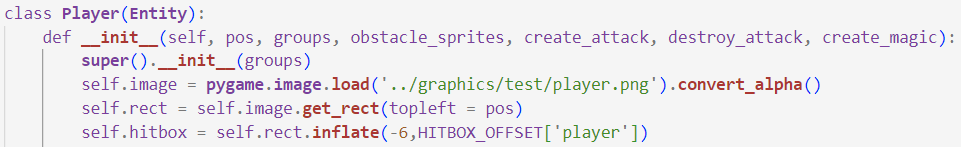
* enemy\_sprites = [sprite for sprite in self.sprites() if hasattr(sprite, 'sprite\_type') and sprite.sprite\_type == 'enemy']: Dòng này tạo một danh sách chỉ chứa các sprite quái. Nó lọc dựa trên việc sprite có thuộc tính sprite\_type được đặt thành "enemy" hay không.
* for enemy in enemy\_sprites: Vòng lặp này lặp qua các sprite của quái.
* enemy.enemy\_update(player): Dòng này gọi phương thức enemy\_update có thể tồn tại trên lớp sprite kẻ thù. Phương thức này có thể xử lý logic của quái và cập nhật hành vi của chúng dựa trên vị trí hoặc hành động của nhân vật.

### **Xây dựng nhân vật**

![A pixelated cartoon character

Description automatically generated](data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAEAAAABACAYAAACqaXHeAAABDUlEQVR42u3bvRGDMAyGYYZIxxqpM0DSs0fmSJGlWIAqO6SDBZzaKqLTybYAv777KoJ/nkKHwRkGGo3WdUspuXIZxxQZ7/wBAAAA34LX+RkaLwgAAABgK2qf1z1LNICcj7VoAgBA7wDyB9t3zWIFkPfLtAbQ1gMAAAD8v0Fm7wDa+AAAAEBdgOgHIQAAAMAGUHozVPsl6O53gwAAcHAADeRoAO63wgAAAEDW4fK4ZpETktdLRxsPAAAAaAvQenMkxwMAAADqAlhBWif8hAgAAJwcwLrg922qmtIgAAAAgO8gpHww0aIt0Nqf9yAlAAAA4AOILoIAAABAWQDvx1Kt6Gn9WYui909VAADQOcAP5d2OzHHxhKoAAAAASUVORK5CYII=)

Đây là nhân vật chính của trò chơi. Nhân vật được thiết có bốn kiểu di chuyển: Lên, Xuống, Trái, Phải. Để cho nhân vật có thể chuyển động cũng như tương tác với mọi thứ xung quanh, ta tạo file player.py để chứa các thiết lập liên quan đến nhân vật.

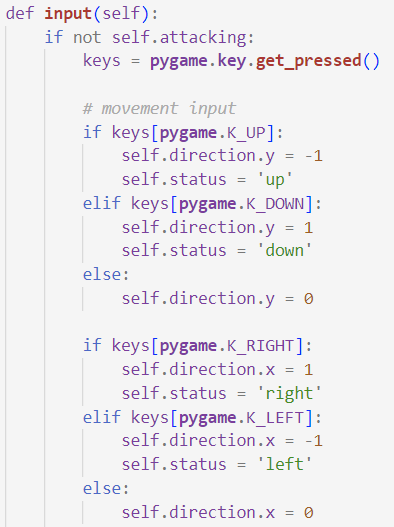


Trong file ta khởi tạo hàm Player và vẽ cho nhân vật ô tile để hitbox của nhân vật có thể tương tác với các thực thể trên bản đồ. Tiếp đến ta thêm các hành động di chuyển cho nhân vật.



* character\_path: Biến này lưu trữ đường dẫn đến thư mục chứa các hình ảnh của nhân vật trong thư mục graphics/player.
* Khối lệnh self.animations = {...} chứa các khoá thể hiện hướng di chuyển như: ’up', 'down', 'left', 'right' hoặc trạng thái kết hợp với hướng di chuyển như: 'right\_idle', 'left\_idle'.
* for animation in self.animations.keys(): vòng lặp để lặp qua từng khóa trong animations. Bên trong vòng lặp:
* full\_path = character\_path + animation: Biến full\_path được tạo ra để lưu trữ đường dẫn đầy đủ đến thư mục chứa các hình ảnh hoạt hình cho một hướng di chuyển/trạng thái cụ thể. Ví dụ, nếu animation là 'up', thì full\_path sẽ bằng '../graphics/player/up/'.
* self.animations[animation] = import\_folder(full\_path): Dòng lệnh này chỉ ra rằng hàm import\_folder có trách nhiệm đọc các hình ảnh từ thư mục được cung cấp (full\_path) và trả về một danh sách chứa các hình ảnh đó. Danh sách này sau đó được gán vào khóa tương ứng trong animations.

Sau khi nhân vật đã được hiển thị trên cửa sổ, phần code tiếp theo ta sẽ tiến hành xử lý các thao tác nhập từ bàn phím của người chơi để điều khiển nhân vật.



* if not self.attacking: Lệnh này sẽ kiểm tra xem nhân vật có đang tấn công hay không, nếu không, các lệnh bên trong lệnh if sẽ được thực thi.
* keys = pygame.key.get\_pressed(): Dòng lệnh này lấy trạng thái của tất cả các phím trên bàn phím. Kết quả trả về là một danh sách keys, trong đó:
* Mỗi phần tử trong danh sách tương ứng với một phím trên bàn phím.
* Giá trị của phần tử là True nếu phím đó đang được bấm, False nếu không.

### **Hệ thống vũ khí và phép thuật**

### **Hệ thống nâng cấp nhân vật**

### **Xây dựng quái**

## **Demo**

### **Giao diện chính**

### **Di chuyển**

### **Tấn công**

## **Link source code**

<https://github.com/QuangVo11311/PixelWar-Project/tree/main/code>

# **KẾT QUẢ ĐẠT ĐƯỢC VÀ HƯỚNG PHÁT TRIỂN**

## **Kết quả đạt được**

Đồ án đã tiến hành khảo sát các kiến thức cần biết về Python và khái niệm cơ bản thế nào là một game thể loại RPG. Trong đồ án này, nhóm đã so sánh Python và các ngôn ngữ lập trình khác để thấy được vì sao người mới học lập trình nên chọn Python. Song song đó nhóm cũng đã nghiên cứu thư viện Pygame để đưa ra các ưu điểm cũng như nhược điểm của thư viện này. Cuối cùng nhóm em đã tổng hợp lại các kiến thức và lập trình ra game Pixel War.

Trong quá trình thực hiện đồ án, nhóm đã cố gắng tham khảo các tài liệu liên quan bằng tiếng Việt lẫn tiếng Anh. Tuy nhiên do thời gian và trình độ có hạn nên không tránh khỏi những hạn chế và thiếu sót nhất định. Do vậy em thật sự mong muốn nhận được những góp ý cả về kiến thức chuyên môn lẫn cách trình bày.

## **Hướng phát triển**

Việc lập trình game bằng Python sử dụng thư viện Pygame đã cho nhóm thấy một hướng phát triển mới cho từng người. Trong tương lai, nhóm muốn tự mình tối ưu và cập nhật thêm một số chức năng cho game Pixel War như: thêm các màn chơi tiếp theo, thêm nhân vật mới hay nâng cấp hệ thống quái trong game.

# **TÀI LIỆU THAM KHẢO**

|  |  |
| --- | --- |
| [1] | Allen B. Downey, *“Think Python: How to think like a computer scientist”*, Green Tea Press, 2nd edition, 2015. |
| [2] | Mark Lutz, *“Learning Python”*, O'Reilly Media, Inc., 5th edition, June 2013. |
| [3] | Swaroop, “A Bite of Python”, Time document create [2004] [Online]. Available: <https://python.swaroopch.com/> [07/03/2024]. |
| [5] | Lê Quang Nhật và Bùi Tấn Lâm, “*Xây dựng trợ lý ảo bằng Python*”, Trường Đại học Công nghệ thông tin và Truyền thông Việt – Hàn, Báo cáo đồ án, 1/2020. |
| [4] | Võ Duy Tuấn, *“Python cơ bản…Rất là CƠ BẢN”*, ebook, 2021. |