

UỶ BAN NHÂN DÂN THÀNH PHỐ HỒ CHÍ MINH

**TRƯỜNG ĐẠI HỌC SÀI GÒN**

**Khoa Công nghệ thông tin**



**Báo cáo đồ án**

**Môn: Lập trình Python**

**Đề tài: Xây dựng ứng dụng game Pacman dùng thư viện Pygame**

Nhóm:

Lê Đức Nam - 3121411135

Tiêu Quang Phú - 3121411165

Nguyễn Văn Ngàn - 3121411142

Phạm Thanh Vương - 3121411229

GVHD: TS. Trịnh Tấn Đạt

**TP Hồ Chí Minh 2024**

**LỜI CẢM ƠN**

Chúng em xin gửi lời cảm ơn chân thành đến Khoa Công nghệ Thông tin, trường Đại học Sài Gòn đã tạo điều kiện thuận lợi để chúng em thực hiện đồ án môn học "Ngôn ngữ lập trình Python". Đây là cơ hội quý báu giúp chúng em củng cố kiến thức và phát triển các kỹ năng cần thiết, làm tiền đề cho những bước tiến xa hơn trong học tập và công việc sau này.

Chúng em cũng xin cảm ơn quý Thầy, Cô trong Khoa đã luôn nhiệt tình giảng dạy và hỗ trợ, trang bị cho chúng em nền tảng kiến thức toàn diện và kỹ lưỡng. Nhờ sự tận tâm của quý Thầy, Cô, chúng em có thêm động lực và tự tin hơn khi đối mặt với những thử thách trong môn học cũng như trong các dự án thực tế.

Đặc biệt, chúng em xin bày tỏ lòng biết ơn sâu sắc đến thầy Trịnh Tấn Đạt, người đã truyền đạt cho chúng em những kiến thức quý báu và hỗ trợ nhiệt tình trong suốt quá trình học tập. Nhờ sự hướng dẫn của thầy, chúng em có thêm nền tảng kiến thức và kỹ năng cần thiết để hoàn thành đồ án này.

Chúng em xin chân thành cảm ơn.

**MỤC LỤC**

[I. MỞ ĐẦU 7](#_Toc183540190)

[1. Giới thiệu đề tài 7](#_Toc183540191)

[2. Lý do chọn đề tài 7](#_Toc183540192)

[3. Mục tiêu – mục đích của đề tài 7](#_Toc183540193)

[II. NỘI DUNG 8](#_Toc183540194)

[1. Giới thiệu, cốt truyện và cách chơi 8](#_Toc183540195)

[2. Yêu cầu kỹ thuật 8](#_Toc183540196)

[3. Cấu trúc chương trình 9](#_Toc183540197)

[3.1. Import thư viện 9](#_Toc183540198)

[3.2. Khởi tạo màn hình game và các biến 10](#_Toc183540199)

[3.2.1. Cài đặt cấu hình 10](#_Toc183540200)

[3.2.2. Khởi tạo các module của pygame 10](#_Toc183540201)

[3.2.3. Thiết lập cửa sổ game và biểu tượng 11](#_Toc183540202)

[3.2.4. Khởi tạo các biến trạng thái game 11](#_Toc183540203)

[3.2.5. Khởi tạo bộ đếm thời gian 11](#_Toc183540204)

[3.2.6. Tải hình ảnh nền cho trò chơi 12](#_Toc183540205)

[3.2.7. Khởi tạo đối tượng điều khiển 13](#_Toc183540206)

[3.2.8. Khởi tạo các đổi tượng trong trò chơi 13](#_Toc183540207)

[3.2.9. Khởi tạo mảng hướng di chuyển 13](#_Toc183540208)

[3.2.10. Khởi tạo bộ đếm thời gian 14](#_Toc183540209)

[3.3. Tạo lớp MovingEntity và StaticEntity (Thực thể di chuyển và thực thể tĩnh) 14](#_Toc183540210)

[3.3.1. Khởi tạo đối tượng MovingEntity 14](#_Toc183540211)

[3.3.2. Phương thức draw và update 15](#_Toc183540212)

[3.3.3. Phương thức kiểm tra va chạm và di chuyển 15](#_Toc183540213)

[3.3.4. Đảm bảo đối tượng luôn nằm trong màn hình 16](#_Toc183540214)

[3.3.5. Reset trạng thái của đối tượng 17](#_Toc183540215)

[3.3.6. Khởi tạo đối tượng StaticEntity 17](#_Toc183540216)

[3.3.7. Phương thức getRect 18](#_Toc183540217)

[3.3.8. Lớp Wall và lớp GhostDoor trong trò chơi 18](#_Toc183540218)

[3.4. Tạo nhân vật Pacman 19](#_Toc183540219)

[3.4.1. Khởi tạo đối tượng Pacman 19](#_Toc183540220)

[3.4.2. Hàm điều khiển Pacman 20](#_Toc183540221)

[3.5. Tạo các con ma 21](#_Toc183540222)

[3.5.1. Khởi tạo lớp GHOST 21](#_Toc183540223)

[3.5.2. Các chế độ hành vi của ma 22](#_Toc183540224)

[3.5.3. Hiển thị trạng thái của ma 23](#_Toc183540225)

[3.5.4. Cập nhật trạng thái của ma 24](#_Toc183540226)

[3.5.5. Khởi tạo và điều khiển các con ma 25](#_Toc183540227)

[3.5.5.1. BlinkyGhost 25](#_Toc183540228)

[3.5.5.2. ClydeGhost 25](#_Toc183540229)

[3.5.5.3. InkyGhost 26](#_Toc183540230)

[3.5.5.4. PinkyGhost 27](#_Toc183540231)

[3.5.6. Quản lý và Điều Khiển Các Trạng Thái của Ma 28](#_Toc183540232)

[3.5.6.1. Lớp GhostState 28](#_Toc183540233)

[3.5.6.2. Lớp ChaseMode (Trạng thái đuổi bắt) 30](#_Toc183540234)

[3.5.6.3. Lớp EatenMode (Trạng thái bị ăn) 30](#_Toc183540235)

[3.5.6.4. Lớp FrightenedMode (Trạng thái sợ hãi) 31](#_Toc183540236)

[3.5.6.5. Lớp HouseMode (Trạng thái ở nhà) 32](#_Toc183540237)

[3.5.6.6. Lớp ScatterMode: Trạng thái phân tán 32](#_Toc183540238)

[3.6. Tạo các viên điểm 33](#_Toc183540239)

[3.6.1. Lớp PacGum 34](#_Toc183540240)

[3.6.2. Lớp SuperPacGum 34](#_Toc183540241)

[3.7. Quản lý va chạm và tính điểm 34](#_Toc183540242)

[3.7.1. Kiểm tra va chạm với thức ăn 35](#_Toc183540243)

[3.7.2. Kiểm tra va chạm với ma quái 35](#_Toc183540244)

[3.7.3. Kiểm tra va chạm với tường 36](#_Toc183540245)

[3.8. Khởi tạo bản đồ và đối tượng trong game 36](#_Toc183540246)

[3.8.1. Mở và đọc tệp level.csv 36](#_Toc183540247)

[3.8.2. Khởi tạo kích thước ô 37](#_Toc183540248)

[3.8.3. Duyệt qua dữ liệu để khởi tạo các đối tượng 37](#_Toc183540249)

[3.8.4. Phân loại các đối tượng vào danh sách 38](#_Toc183540250)

[3.9. Tính năng phát nhạc trong trò chơi 38](#_Toc183540251)

[3.9.1. Hàm playMusic() 39](#_Toc183540252)

[3.9.2. Hàm forcePlayMusic() 40](#_Toc183540253)

[3.10. Hàm main() 40](#_Toc183540254)

[3.10.1. Khởi tạo và Thiết lập 40](#_Toc183540255)

[3.10.2. Vòng lặp chính 41](#_Toc183540256)

[3.10.2.1. Cập nhật và Vẽ Màn Hình 41](#_Toc183540257)

[3.10.2.2. Kiểm tra trạng thái trò chơi 41](#_Toc183540258)

[3.10.2.3. Xử lý sự kiện 42](#_Toc183540259)

[3.10.3. Kết thúc 43](#_Toc183540260)

[III. Thực nghiệm và phân tích kết quả 44](#_Toc183540261)

[1. Kiểm thử các tính năng chính 44](#_Toc183540262)

[2. Phân tích kết quả kiểm thử 48](#_Toc183540263)

[3. Những thách thức và cách khắc phục 48](#_Toc183540264)

[4. Đánh giá hiệu suất và trải nghiệm người dùng 48](#_Toc183540265)

[IV. Kết luận và hướng phát triển 49](#_Toc183540266)

[1. Kết luận 49](#_Toc183540267)

[2. Hướng phát triển 49](#_Toc183540268)

[V. Tài liệu tham khảo 50](#_Toc183540269)

**MỤC HÌNH ẢNH**

[Hình 1: Import thư viện 9](#_Toc183586273)

[Hình 2: Cài đặt cấu hình 10](#_Toc183586274)

[Hình 3: Khởi tạo các module của pygame 10](#_Toc183586275)

[Hình 4: Cửa sổ game và biểu tượng 11](#_Toc183586276)

[Hình 5: Các biến trạng thái game 11](#_Toc183586277)

[Hình 6: Khởi tạo bộ đếm thời gian 11](#_Toc183586278)

[Hình 7: Tạo nền cho trò chơi 12](#_Toc183586279)

[Hình 8: Hình nền của trò chơi 12](#_Toc183586280)

[Hình 9: Khởi tạo đối tượng điều khiển 13](#_Toc183586281)

[Hình 10: Khởi tạo các đổi tượng trong trò chơi 13](#_Toc183586282)

[Hình 11: Khởi tạo mảng hướng di chuyển 13](#_Toc183586283)

[Hình 12: Khởi tạo bộ đếm thời gian 14](#_Toc183586284)

[Hình 13: Khởi tạo đối tượng MovingEntity 14](#_Toc183586285)

[Hình 14: Phương thức draw và update 15](#_Toc183586286)

[Hình 15: Phương thức kiểm tra va chạm và di chuyển 15](#_Toc183586287)

[Hình 16: Đoạn mã đảm bảo thực thể không ra ngoài màn hình 16](#_Toc183586288)

[Hình 17: Hình thực thể ra ngoài nó sẽ xuất hiện lại từ phía đối diện 16](#_Toc183586289)

[Hình 18: Reset trạng thái của đối tượng 17](#_Toc183586290)

[Hình 19: Khởi tạo đối tượng StaticEntity 17](#_Toc183586291)

[Hình 20: Phương thức getRect 18](#_Toc183586292)

[Hình 21: Lớp Wall và lớp GhostDoor 18](#_Toc183586293)

[Hình 22: Khởi tạo đối tượng Pacman 19](#_Toc183586294)

[Hình 23: Hình ảnh của Pacman 19](#_Toc183586295)

[Hình 24: Hàm điều khiển Pacman 20](#_Toc183586296)

[Hình 26: Khởi tạo lớp GHOST 21](#_Toc183586297)

[Hình 27: Khởi tạo các chế độ hành vi của ma 22](#_Toc183586298)

[Hình 28: Hiển thị trạng thái của ma 23](#_Toc183586299)

[Hình 29: Hình ảnh khi ma bị ăn bởi Pacman 23](#_Toc183586300)

[Hình 30: Hình ảnh khi ma trong trạng thái hoảng sợ 23](#_Toc183586301)

[Hình 31: Cập nhật trạng thái của ma 24](#_Toc183586302)

[Hình 32: Khởi tạo và điều khiển BlinkyGhost 25](#_Toc183586303)

[Hình 33: Hình ảnh của BlinkyGhost 25](#_Toc183586304)

[Hình 34: Khởi tạo và điều khiển ClydeGhost 25](#_Toc183586305)

[Hình 35: Hình ảnh của ClydeGhost 26](#_Toc183586306)

[Hình 36: Khởi tạo và điều khiển InkyGhost 26](#_Toc183586307)

[Hình 37: Hình ảnh của InkyGhost 26](#_Toc183586308)

[Hình 38: Khởi tạo và điều khiển PinkhGhost 27](#_Toc183586309)

[Hình 39: Hình ảnh của PinkhGhost 27](#_Toc183586310)

[Hình 40: Khởi tạo trạng thái của ma 29](#_Toc183586311)

[Hình 41: Khởi tạo trạng thái đuổi bắt 30](#_Toc183586312)

[Hình 42: Khởi tạo trạng thái bị ăn 30](#_Toc183586313)

[Hình 43: Khởi tạo trạng thái sợ hãi 31](#_Toc183586314)

[Hình 44: Khởi tạo trạng thái ở nhà 32](#_Toc183586315)

[Hình 45: Khởi tạo trạng thái phân tán 32](#_Toc183586316)

[Hình 46: Hình ảnh có viên điểm PacGum và Super Pac-Gum trong trò chơi 33](#_Toc183586317)

[Hình 47: Khởi tạo viên điểm PacGum 34](#_Toc183586318)

[Hình 48: Khởi tạo viên điểm Super Pac-Gum 34](#_Toc183586319)

[Hình 49: Hàm kiểm tra va chạm với thức ăn 35](#_Toc183586320)

[Hình 50: Hàm kiểm tra va chạm với ma quái 35](#_Toc183586321)

[Hình 51: Hàm kiểm tra va chạm với tường 36](#_Toc183586322)

[Hình 52: hàm mở và đọc tệp level.csv 36](#_Toc183586323)

[Hình 53: Khởi tạo kích thước ô 37](#_Toc183586324)

[Hình 54: Hàm duyệt qua từng dòng và cột trong danh sách level. 37](#_Toc183586325)

[Hình 55: Phân loại các đối tượng vào danh sách 38](#_Toc183586326)

[Hình 56: Hàm phát nhạc nền hoặc các hiệu ứng âm thanh 39](#_Toc183586327)

[Hình 57: Hàm phát nhạc mà không kiểm tra xem có nhạc nào đang phát hay không. 40](#_Toc183586328)

[Hình 58: Hàm khởi tạo và Thiết lập 40](#_Toc183586329)

[Hình 59: Cập nhật và vẽ màn hình 41](#_Toc183586330)

[Hình 60: Kiểm tra trạng thái trò chơi 41](#_Toc183586331)

[Hình 61: Hàm xử lý các sự kiện 42](#_Toc183586332)

[Hình 62: Vòng lặp kết thúc và pygame.quit() 43](#_Toc183586333)

[Hình 63: Giao diện khi bắt đầu trò chơi. 45](#_Toc183586334)

[Hình 64: Giao diện PacMan ăn các viên Pacgum 45](#_Toc183586335)

[Hinh 65: Giao diện PacMan sau khi va chạm Ghost 46](#_Toc183586336)

[Hình 66: Giao diện PacMan ăn được viên Super Pacgum 46](#_Toc183586337)

[Hình 67: Giao diện khi ăn hết các viên Pacgum và chiến thắng 47](#_Toc183586338)

[Hình 68: Giao diện kết thúc trò chơi khi thua hết 3 mạng 47](#_Toc183586339)

# MỞ ĐẦU

## Giới thiệu đề tài

Trò chơi Pacman là một trong những trò chơi điện tử cổ điển nổi tiếng, xoay quanh nhân vật Pacman, người ăn các viên bi trong mê cung và tránh các con ma. Đồ án này nhằm nâng cao kỹ năng lập trình Python và phát triển game bằng thư viện Pygame. Qua việc xây dựng một phiên bản của trò chơi Pacman, người viết mong muốn khám phá các khái niệm lập trình game cơ bản và thực hành thiết kế giao diện, cũng như xử lý tương tác.

## Lý do chọn đề tài

Việc lựa chọn xây dựng trò chơi Pacman trong đồ án này xuất phát từ mong muốn tiếp cận và thực hành những khái niệm lập trình game một cách trực quan và dễ hiểu. Pacman không chỉ là một trò chơi điện tử cổ điển và quen thuộc với nhiều người, mà còn sở hữu lối chơi đơn giản, dễ phát triển cho người mới bắt đầu. Nhờ đó, nhóm có thể tập trung vào việc tìm hiểu cấu trúc của một trò chơi điện tử cơ bản, từ cách điều khiển nhân vật đến cách xây dựng môi trường và quy tắc trò chơi.

Bên cạnh đó, việc phát triển Pacman bằng thư viện Pygame cung cấp cơ hội để nhóm làm quen với một công cụ mạnh mẽ trong phát triển game 2D trên nền tảng Python [1]. Qua dự án này, chúng em mong muốn rèn luyện kỹ năng lập trình, xử lý tình huống và phát triển tư duy logic thông qua việc giải quyết các vấn đề trong quá trình thiết kế và triển khai trò chơi.

## Mục tiêu – mục đích của đề tài

**Mục tiêu:**

* Hiểu rõ về lập trình game cơ bản và cấu trúc của một trò chơi điện tử.
* Thực hành sử dụng thư viện Pygame để phát triển giao diện, xử lý âm thanh và các yếu tố tương tác.
* Tạo ra một trò chơi Pacman hoàn chỉnh với các tính năng di chuyển nhân vật, tương tác với đối thủ, và hệ thống điểm số, qua đó rèn luyện kỹ năng lập trình và giải quyết vấn đề.

**Mục đích:**

* Nắm chắc được được kỹ năng và kiến thức về lập trình.
* Tìm hiều về thư viện Pygame trong ngôn ngữ lập trình Python.
* Cũng cố, áp dụng, nâng cao kiến thức đã được học.
* Nắm bắt được quy trình làm game cơ bản.

# NỘI DUNG

## Giới thiệu, cốt truyện và cách chơi

Pacman là một sinh vật nhỏ bé sống trong một mê cung kỳ bí, luôn khao khát hòa bình và những viên năng lượng để tồn tại. Tuy nhiên, trong mê cung, Pacman luôn bị truy đuổi bởi những con ma đầy ác ý, bao gồm Blinky (ma đỏ), Pinky (ma hồng), Inky (ma xanh), và Clyde (ma cam). Những con ma này được lập trình để săn lùng Pacman và ngăn cản hành trình của Pacman trong mê cung. Với mỗi viên điểm mà Pacman ăn, anh ta trở nên mạnh mẽ hơn, và khi ăn các viên sức mạnh lớn, Pacman có khả năng tấn công lại những con ma trong một khoảng thời gian ngắn, tạm thời loại bỏ mối đe dọa. Mục tiêu của Pacman là ăn hết các viên điểm trong mê cung để giành chiến thắng, đồng thời phải né tránh hoặc tiêu diệt các con ma khi cần.

Trò chơi Pacman là một trò chơi arcade hấp dẫn, nơi người chơi điều khiển nhân vật Pacman di chuyển trong một mê cung. Mục tiêu của trò chơi là ăn tất cả các viên bi năng lượng trong mê cung để ghi điểm, đồng thời tránh bị bắt bởi các con ma.

**Cách chơi cơ bản:**

* **Di chuyển Pacman:** Người chơi điều khiển Pacman di chuyển lên, xuống, trái hoặc phải trong mê cung bằng các phím mũi tên trên bàn phím.
* **Tránh ma:** Trong mê cung có bốn con ma, mỗi con có cách di chuyển riêng. Người chơi cần phải tránh va chạm với chúng để không bị mất mạng. Nếu Pacman va vào bất kỳ con ma nào khi không có trạng thái sức mạnh, người chơi sẽ bị trừ một mạng và phải bắt đầu lại từ điểm đầu.
* **Ăn điểm:** Pacman sẽ ghi điểm mỗi khi ăn viên điểm. Ngoài ra, trong mê cung cũng có những viên sức mạnh (power pellets) cho phép Pacman tạm thời tấn công các con ma để tăng điểm, khiến chúng chạy trốn và trở thành dễ ăn.

Mục tiêu cuối cùng là ăn hết tất cả các viên bi và đạt được số điểm cao nhất trước khi hết mạng. Trò chơi sẽ kết thúc khi Pacman ăn hết các viên điểm trong mê cung hoặc hết mạng.

## Yêu cầu kỹ thuật

Để phát triển trò chơi Pacman, các thư viện và công cụ sau được sử dụng:

* **Python:** Là ngôn ngữ lập trình chính cho dự án, Python cung cấp cú pháp đơn giản và mạnh mẽ, rất phù hợp cho việc phát triển game.
* **Pygame:** Là thư viện hỗ trợ phát triển game trong Python. Pygame cung cấp các chức năng để quản lý đồ họa, âm thanh, và xử lý sự kiện, giúp việc xây dựng giao diện và tương tác trong game trở nên dễ dàng hơn [1], [2].
* **Thư viện hỗ trợ khác:** Có thể sử dụng các thư viện bổ sung như NumPy (để xử lý mảng và tính toán).

Các công cụ này sẽ hỗ trợ người viết trong việc thiết kế và phát triển một trò chơi Pacman hoàn chỉnh, từ lập trình logic game đến việc tạo ra trải nghiệm người dùng thú vị.

## Cấu trúc chương trình

### Import thư viện

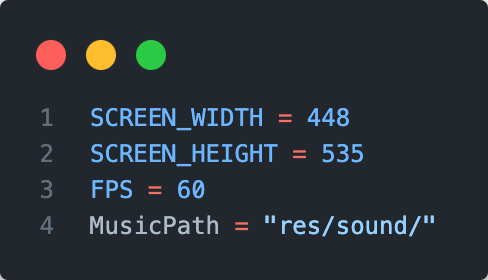


Hình 1: Import thư viện

* **pygame**: Đây là thư viện chính để phát triển game, hỗ trợ vẽ đồ họa, xử lý sự kiện, âm thanh và các sự kiện tương tác (như nhấn phím) một cách trực quan, dễ sử dụng cho các trò chơi 2D [1]. Nhờ cú pháp đơn giản và hỗ trợ đa nền tảng, thư viện này rất thích hợp cho việc phát triển các trò chơi nhỏ và các đồ án học thuật như Pacman.
* **random**: được dùng để tạo ra các giá trị ngẫu nhiên cho vị trí và chuyển động của các đối tượng, đặc biệt là trong việc thay đổi hướng đi của ma khi Pacman chuyển sang trạng thái ăn ma. Điều này giúp tăng tính thử thách và sự đa dạng trong lối chơi.
* **csv**: Thư viện này cho phép đọc và ghi dữ liệu dạng bảng. Trong trò chơi Pacman, thư viện csv được dùng để lưu dữ liệu bản đồ dưới dạng lưới tọa độ. Mỗi ký tự trong tệp CSV đại diện cho một đối tượng (như tường, ma, viên điểm), giúp quá trình thiết lập bản đồ trò chơi dễ dàng hơn và có thể tùy chỉnh mà không cần thay đổi mã nguồn.
* **math**: Cung cấp các hàm toán học nâng cao, ví dụ như math.sqrt(), math.pow(), v.v., giúp tính toán trong các bài toán cần sử dụng toán học.

### Khởi tạo màn hình game và các biến

#### Cài đặt cấu hình



Hình 2: Cài đặt cấu hình

* **SCREEN\_WIDTH, SCREEN\_HEIGHT** lần lượt là kích thước của cửa sổ trò chơi, với chiều rộng là 448-pixel và chiều cao là 535-pixel.
* **FPS (Frames Per Second)**: Đây là số lượng khung hình được hiển thị mỗi giây trong trò chơi. Thiết lập FPS là rất quan trọng để đảm bảo trò chơi chạy mượt mà với tốc độ ổn định.
* Biến **MusicPath** chứa đường dẫn tới thư mục nơi lưu trữ các tệp âm thanh của trò chơi. Đường dẫn này giúp trò chơi dễ dàng truy cập và phát nhạc từ thư mục **"res/sound/"**.

#### Khởi tạo các module của pygame



Hình 3: Khởi tạo các module của pygame

* Hàm **pygame.init()** là hàm khởi tạo chính của thư viện Pygame. Khi gọi hàm này, Pygame sẽ thiết lập tất cả các mô-đun cần thiết để sử dụng các chức năng đồ họa, âm thanh và sự kiện.
* Hàm **pygame.font.init()** khởi tạo mô-đun font của Pygame, cho phép sử dụng các phông chữ trong trò chơi, ví dụ như để hiển thị điểm số, tên trò chơi hoặc các thông báo.
* Hàm **pygame.mixer.init()** khởi tạo mô-đun âm thanh của Pygame, giúp bạn phát nhạc nền, hiệu ứng âm thanh và xử lý âm thanh trong trò chơi.

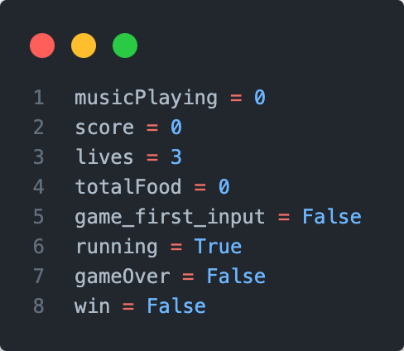
#### Thiết lập cửa sổ game và biểu tượng



Hình 4: Cửa sổ game và biểu tượng

* **pygame.display.set\_icon(icon)** thiết lập biểu tượng cửa sổ game.
* **pygame.display.set\_mode**((SCREEN\_WIDTH, SCREEN\_HEIGHT)) tạo cửa sổ game với kích thước đã định.
* **pygame.display.set\_caption**('Pacman') đặt tiêu đề cửa sổ là "Pacman".

#### Khởi tạo các biến trạng thái game

**

Hình 5: Các biến trạng thái game

* Các biến như **musicPlaying** lưu trạng thái bài nhạc hiện tại, **score, lives, totalFood** lưu trữ điểm số, số mạng và số lượng thức ăn. **game\_first\_input, running, gameOver**, và **win** giúp quản lý trạng thái của game.

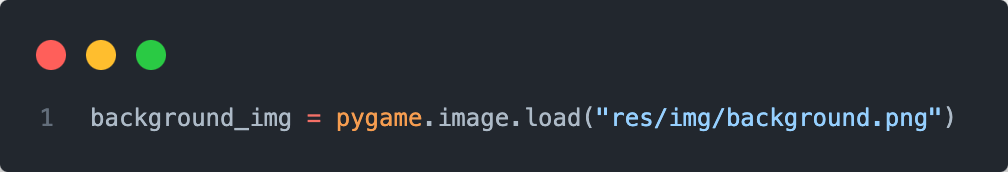
#### Khởi tạo bộ đếm thời gian



Hình 6: Khởi tạo bộ đếm thời gian

Biến **timerStart** sẽ được dùng để quản lý thời gian trong game, chẳng hạn như để tính thời gian còn lại hoặc thời gian cho một sự kiện trong game.

#### Tải hình ảnh nền cho trò chơi



Hình 7: Tạo nền cho trò chơi

Dòng mã này sử dụng **pygame.image.load()** để tải hình ảnh nền từ tệp **background.png**. Biến **background\_img** lưu trữ hình ảnh này. Hình ảnh nền sau đó sẽ được vẽ lên cửa sổ trò chơi trong mỗi vòng lặp để tạo ra môi trường trực quan cho người chơi.

A video game graphics

Description automatically generated

Hình 8: Hình nền của trò chơi

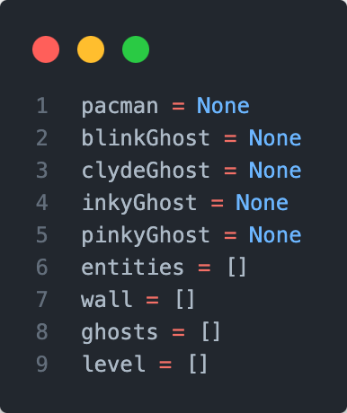
#### Khởi tạo đối tượng điều khiển



Hình 9: Khởi tạo đối tượng điều khiển

Các dòng mã này tạo ra các đối tượng **Key** để theo dõi trạng thái của các phím di chuyển trong trò chơi (lên, xuống, trái, phải). Mỗi đối tượng sẽ xác định khi người chơi nhấn các phím điều khiển, giúp di chuyển nhân vật trong game.

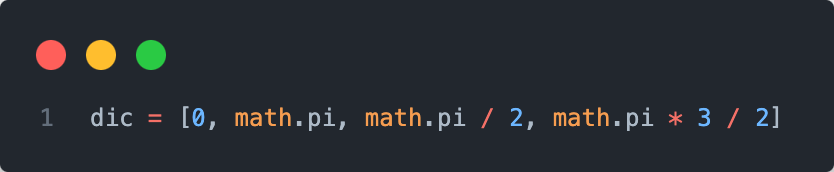
#### Khởi tạo các đổi tượng trong trò chơi



Hình 10: Khởi tạo các đổi tượng trong trò chơi

Khởi tạo các đối tượng cần thiết cho game như Pacman, các ma (Blinky, Clyde, Inky, Pinky), tường, các thực thể và cấp độ.

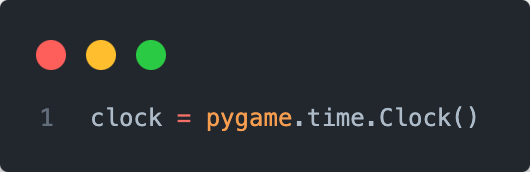
#### Khởi tạo mảng hướng di chuyển



Hình 11: Khởi tạo mảng hướng di chuyển

Biến dic chứa các giá trị hướng di chuyển trong không gian 2D tính theo radian (0 độ, 180 độ, 90 độ, 270 độ).

#### Khởi tạo bộ đếm thời gian



Hình 12: Khởi tạo bộ đếm thời gian

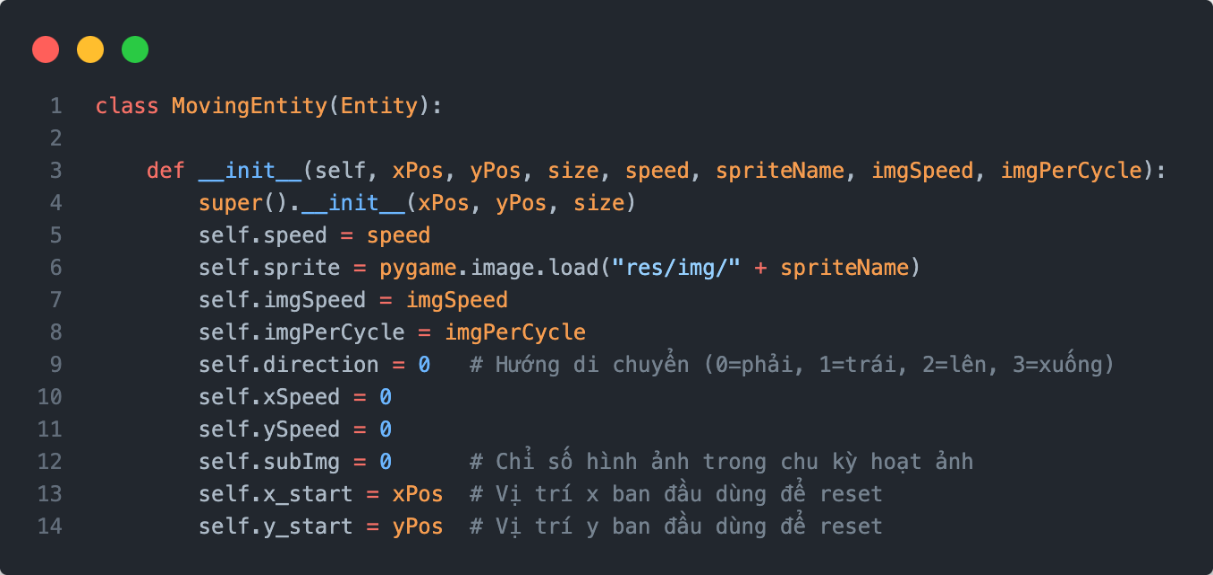
Biến clock sẽ giúp kiểm soát tốc độ khung hình (FPS) trong trò chơi.

### Tạo lớp MovingEntity và StaticEntity (Thực thể di chuyển và thực thể tĩnh)

Lớp MovingEntity kế thừa từ lớp Entity, chịu trách nhiệm cho các hành động di chuyển của thực thể trên màn hình. Các đối tượng kế thừa lớp này sẽ có khả năng di chuyển, cập nhật hình ảnh, kiểm tra va chạm và vẽ lên màn hình.

Lớp StaticEntity là một lớp con của Entity, dành cho các thực thể cố định, không thay đổi vị trí trong trò chơi. Những đối tượng như tường hay vật cản có thể được định nghĩa là StaticEntity để hỗ trợ việc phát hiện va chạm.

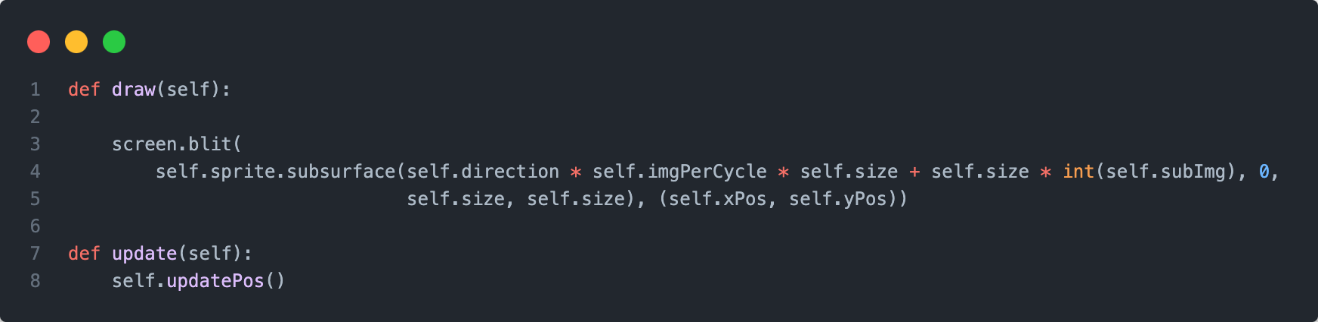
#### Khởi tạo đối tượng MovingEntity



Hình 13: Khởi tạo đối tượng MovingEntity

Phần này khởi tạo các thuộc tính cơ bản của đối tượng **MovingEntity** như vị trí, kích thước, tốc độ, sprite (hình ảnh), và các thuộc tính liên quan đến hoạt ảnh như tốc độ thay đổi hình ảnh và số lượng hình ảnh trong một chu kỳ hoạt ảnh. Các biến **x\_start** và **y\_start** lưu trữ vị trí ban đầu để có thể reset khi cần.

#### Phương thức draw và update



Hình 14: Phương thức draw và update

Lớp MovingEntity có phương thức **draw** để vẽ hình ảnh của đối tượng lên màn hình và phương thức **update** để cập nhật vị trí và trạng thái của đối tượng theo tốc độ di chuyển.

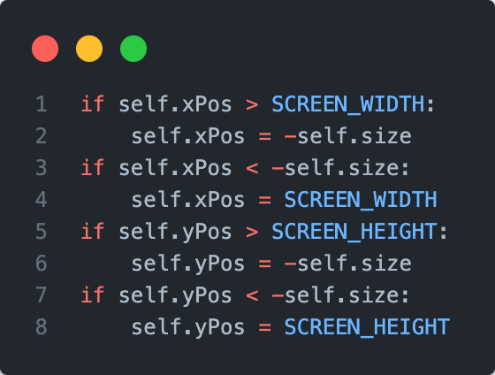
#### Phương thức kiểm tra va chạm và di chuyển



Hình 15: Phương thức kiểm tra va chạm và di chuyển

Phần này kiểm tra và cập nhật vị trí của đối tượng **MovingEntity** khi di chuyển. Các phương thức **onTheGrid** và **onTheGamePlay** dùng để kiểm tra xem thực thể có nằm trong khuôn khổ của trò chơi hay không. Hàm **updatePos** sẽ cập nhật vị trí thực thể và thay đổi hướng đi tùy theo tốc độ di chuyển.

#### Đảm bảo đối tượng luôn nằm trong màn hình



Hình 16: Đoạn mã đảm bảo thực thể không ra ngoài màn hình

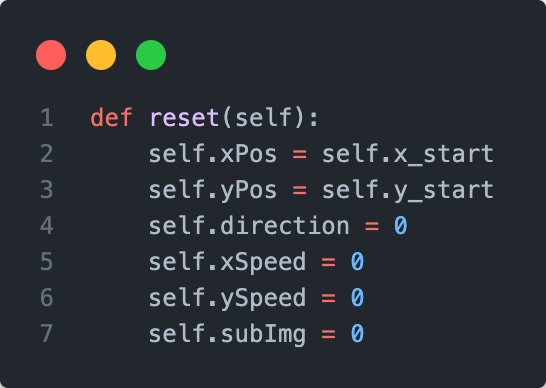
Đoạn mã này đảm bảo rằng thực thể không bao giờ ra ngoài màn hình. Nếu đối tượng vượt ra ngoài màn hình, nó sẽ xuất hiện lại từ phía đối diện.

A video game with colorful ghosts and squares

Description automatically generated

Hình 17: Hình thực thể ra ngoài nó sẽ xuất hiện lại từ phía đối diện

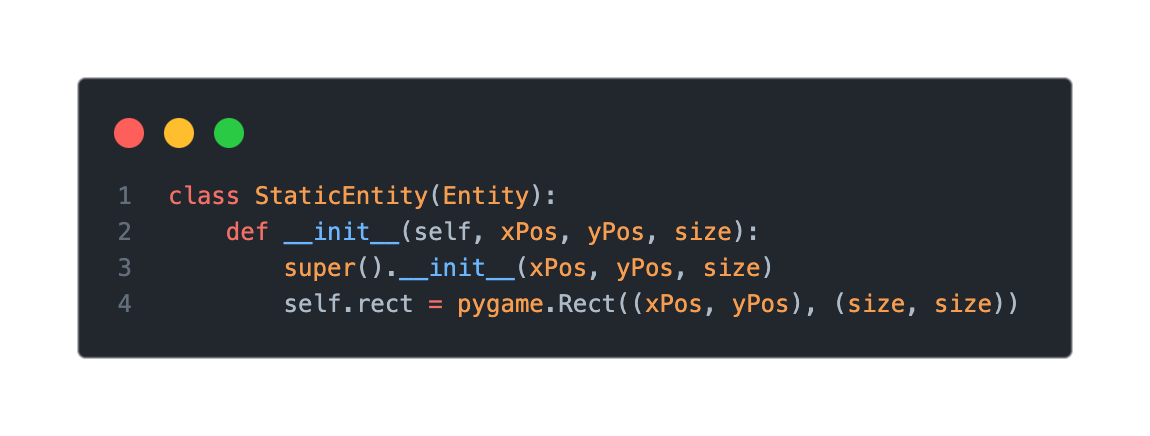
#### Reset trạng thái của đối tượng



Hình 18: Reset trạng thái của đối tượng

Phương thức **reset** sẽ khôi phục lại các thuộc tính ban đầu của thực thể, giúp việc tái khởi động hay đặt lại trạng thái trở nên dễ dàng hơn.

#### Khởi tạo đối tượng StaticEntity



Hình 19: Khởi tạo đối tượng StaticEntity

Phương thức khởi tạo **\_\_init\_\_** nhận vào các tham số **xPos**, **yPos**, và **size**. Sau đó, nó tạo một hình chữ nhật (rect) để đại diện cho vùng chiếm chỗ của thực thể trên màn hình, giúp dễ dàng kiểm tra va chạm với các thực thể khác.

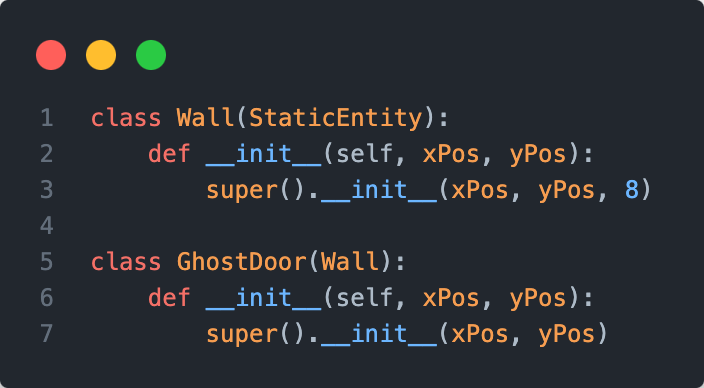
#### Phương thức getRect



Hình 20: Phương thức getRect

Phương thức **getRect** trả về hình chữ nhật (rect) của thực thể, giúp cho các đối tượng khác có thể kiểm tra va chạm với thực thể này. Điều này rất hữu ích cho việc xác định nếu Pacman hoặc các thực thể di chuyển khác đang đụng phải các vật cản.

#### Lớp Wall và lớp GhostDoor trong trò chơi



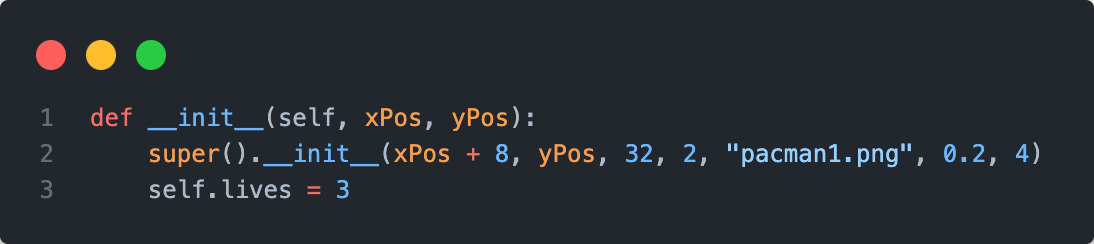
Hình 21: Lớp Wall và lớp GhostDoor

* Lớp Wall là một lớp con của StaticEntity, sử dụng kích thước cố định là 8 cho các đối tượng tường.
* Lớp GhostDoor kế thừa từ lớp Wall, vì vậy nó cũng sẽ có kích thước mặc định là 8, và có thể được mở rộng sau nếu cần các thuộc tính riêng cho cánh cửa của ma.

### Tạo nhân vật Pacman

Lớp Pacman đại diện cho nhân vật Pacman trong trò chơi. Đây là một lớp con của MovingEntity, kế thừa các thuộc tính và phương thức chung cho các thực thể di chuyển và thêm các thuộc tính, phương thức đặc trưng cho Pacman.

#### Khởi tạo đối tượng Pacman



Hình 22: Khởi tạo đối tượng Pacman

Phương thức **\_\_init\_\_** khởi tạo Pacman tại tọa độ (xPos, yPos), với hình ảnh"pacman1.png". Giá trị **lives = 3** xác định số mạng của Pacman cũng như số lượt chơi tối đa của người chơi trong trò chơi.

****

Hình 23: Hình ảnh của Pacman

#### Hàm điều khiển Pacman



Hình 24: Hàm điều khiển Pacman

Phương thức **input** nhận và xử lý thao tác điều khiển từ người chơi. Dựa vào hướng di chuyển (trái, phải, lên, xuống) và kiểm tra va chạm tường (checkWallCollision), phương thức cập nhật tốc độ di chuyển (xSpeed, ySpeed) cho Pacman.

### Tạo các con ma

Trong trò chơi Pacman, các con ma là những nhân vật đối địch với Pacman, mang lại thách thức chính cho người chơi. Mỗi con ma có các chế độ di chuyển khác nhau, tạo nên sự hấp dẫn và khó đoán cho trò chơi.

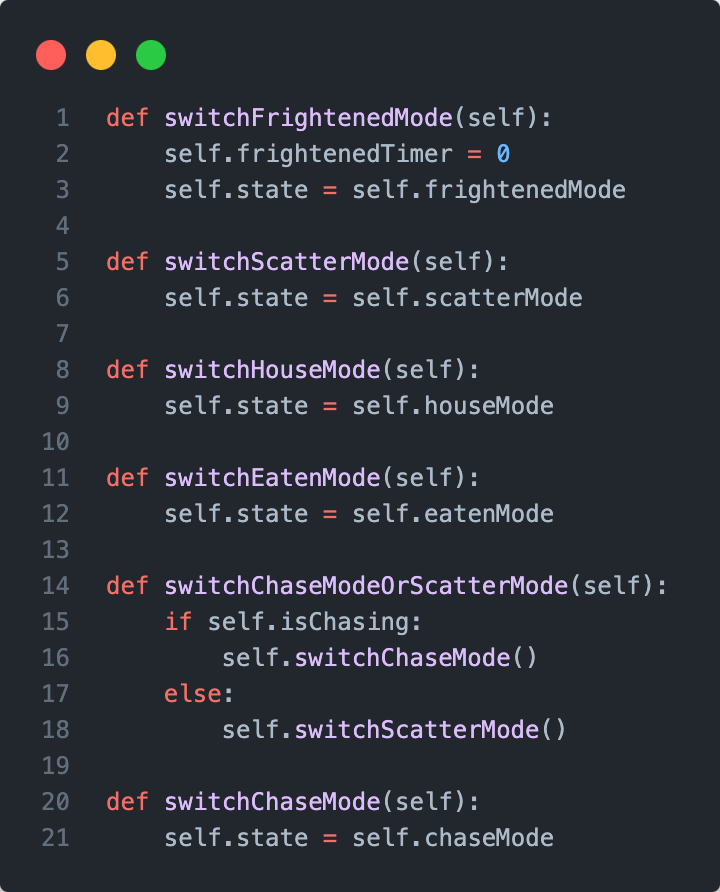
#### Khởi tạo lớp GHOST



Hình 26: Khởi tạo lớp GHOST

* **self.ghost\_eaten\_sprite, self.ghost\_frightened\_1, self.ghost\_frightened\_2:**
  + Đây là các hình ảnh thay đổi để thể hiện các trạng thái khác nhau của ma trong game:
    - **ghost\_eaten\_sprite**: Hình ảnh khi ma bị ăn bởi Pacman.
    - **ghost\_frightened\_1 và ghost\_frightened\_2**: Các hình ảnh khi ma trong trạng thái hoảng sợ (trạng thái bị Pacman đuổi trong một khoảng thời gian).
* **self.state**: Trạng thái hiện tại của ma. Khi một đối tượng Ghost được tạo ra, trạng thái mặc định của nó là houseMode, nghĩa là ma ở trong nhà của chúng (trạng thái bắt đầu). Ma có thể chuyển sang các chế độ khác như đuổi bắt (chaseMode), phân tán (scatterMode), hoảng sợ (frightenedMode), hoặc bị ăn (eatenMode) tùy theo các sự kiện trong game.1
* **self.modeTimer và self.frightenedTimer**:
  + **modeTimer**: Bộ đếm thời gian để theo dõi thời gian của chế độ hiện tại của ma, chẳng hạn như khi ma đang ở trong chế độ phân tán hoặc đuổi bắt.
  + **frightenedTimer**: Bộ đếm thời gian theo dõi thời gian mà ma ở trong trạng thái hoảng sợ (bị Pacman đuổi trong một khoảng thời gian nhất định).

#### Các chế độ hành vi của ma



Hình 27: Khởi tạo các chế độ hành vi của ma

* **switchFrightenedMode(self):** Phương thức này chuyển ma sang trạng thái **hoảng sợ (frightenedMode)**. Khi ma chuyển sang chế độ hoảng sợ, bộ đếm **frightenedTimer** được đặt lại về 0, bắt đầu theo dõi thời gian ma sẽ ở trong trạng thái hoảng sợ.
* **switchScatterMode(self):** Phương thức này chuyển ma sang chế độ **phân tán (scatterMode)**. Trong chế độ phân tán, các con ma di chuyển theo hướng cố định và không đuổi Pacman.
* **switchHouseMode(self):** Phương thức này chuyển ma về chế độ **trong nhà (houseMode)**, tức là trạng thái ban đầu của ma. Khi ma trở lại nhà, nó sẽ chờ đợi trước khi tiếp tục di chuyển.
* **switchEatenMode(self):** Phương thức này chuyển ma sang chế độ **bị ăn (eatenMode)**. Khi Pacman ăn ma, ma sẽ bị tiêu diệt và trở về nhà để tái sinh.
* **switchChaseModeOrScatterMode(self):** Phương thức này kiểm tra trạng thái **isChasing** của ma và chuyển ma sang một trong hai chế độ:
  + Nếu **isChasing** là True, ma sẽ chuyển sang chế độ **đuổi bắt (chaseMode)**, tức là ma sẽ di chuyển để đuổi theo Pacman.
  + Nếu **isChasing** là False, ma sẽ chuyển sang chế độ **phân tán (scatterMode)**, tức là ma sẽ di chuyển theo một hướng cố định.

#### Hiển thị trạng thái của ma



Hình 28: Hiển thị trạng thái của ma

* Khi trạng thái của ma là **frightenedMode** (hoảng sợ):
  + Nếu bộ đếm frightenedTimer nhỏ hơn hoặc bằng 60 giây x 5 (tương đương 5 giây), hoặc nếu giá trị **frightenedTimer % 20 > 10** (một kiểu hiệu ứng nhấp nháy), thì ma sẽ có hình ảnh hoảng sợ 1 (**ghost\_frightened\_1**).
  + Ngược lại, ma sẽ được hiển thị với hình ảnh hoảng sợ 2 (**ghost\_frightened\_2**).
* Khi trạng thái của ma là eatenMode: Ma sẽ được vẽ với hình ảnh ma bị ăn (ghost\_eaten\_sprite), sử dụng hướng di chuyển self.direction để hiển thị đúng sprite.
* Khi ma ở bất kỳ trạng thái nào khác (không phải hoảng sợ hay bị ăn): Ma sẽ được vẽ với hình ảnh ma bình thường (sprite), di chuyển tùy theo hướng và chu kỳ hình ảnh (imgPerCycle). Hình ảnh này được vẽ từ sprite sheet, với các sprite được cắt ra theo các chỉ số xác định.

**ghost\_eaten\_sprite**: Là hình ảnh khi ma bị ăn bởi Pacman.



Hình 29: Hình ảnh khi ma bị ăn bởi Pacman

**ghost\_frightened\_1 và ghost\_frightened\_2**: Các hình ảnh khi ma trong trạng thái hoảng sợ (trạng thái bị Pacman đuổi trong một khoảng thời gian).



Hình 30: Hình ảnh khi ma trong trạng thái hoảng sợ

#### Cập nhật trạng thái của ma

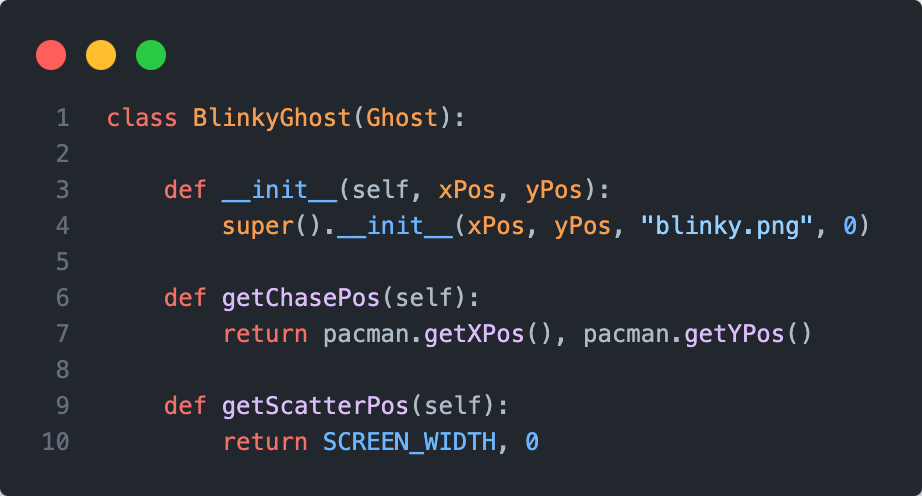


Hình 31: Cập nhật trạng thái của ma

* **self.count:** Bộ đếm này giúp giữ ma ở chế độ mặc định (chờ đợi thời gian bắt đầu) cho đến khi thời gian đạt mức đã định (theo self.time). Khi giá trị này đạt giới hạn, ma sẽ bắt đầu cập nhật trạng thái và hành vi của mình.
* **self.state == self.frightenedMode:**
  + Khi ma trong chế độ hoảng sợ, bộ đếm frightenedTimer sẽ được tăng lên qua từng lần gọi update.
  + Khi frightenedTimer đạt đến giới hạn (7 phút, tính bằng giây), chế độ hoảng sợ sẽ kết thúc, và ma sẽ chuyển sang chế độ khác, thường là chế độ bình thường hoặc chế độ đuổi bắt/phân tán.
* **self.state == self.chaseMode or self.state == self.scatterMode:**
  + Nếu ma đang ở chế độ đuổi bắt (chaseMode) hoặc chế độ phân tán (scatterMode), bộ đếm modeTimer sẽ tiếp tục tăng lên.
  + Khi modeTimer vượt quá giới hạn thời gian (20 giây cho chế độ đuổi bắt, 10 giây cho chế độ phân tán), ma sẽ chuyển trạng thái giữa hai chế độ này (đuổi bắt hoặc phân tán), thay đổi hành vi di chuyển.

#### Khởi tạo và điều khiển các con ma

##### BlinkyGhost



Hình 32: Khởi tạo và điều khiển BlinkyGhost

* BlinkyGhost sẽ luôn đuổi theo Pacman ("ChaseMode").
* Với phương thức getChasePos, BlinkyGhost được thiết kế để tối ưu hóa khả năng bám sát Pacman, điều này tạo ra thử thách lớn hơn cho người chơi và tăng tính hấp dẫn cho trò chơi.
* Phương thức getScatterPos sẽ trả về tọa độ cố định ở góc trên bên phải màn hình, là vị trí mà Blinky sẽ di chuyển tới khi không đuổi theo Pacman (SCREEN\_WIDTH, 0).



Hình 33: Hình ảnh của BlinkyGhost

##### ClydeGhost



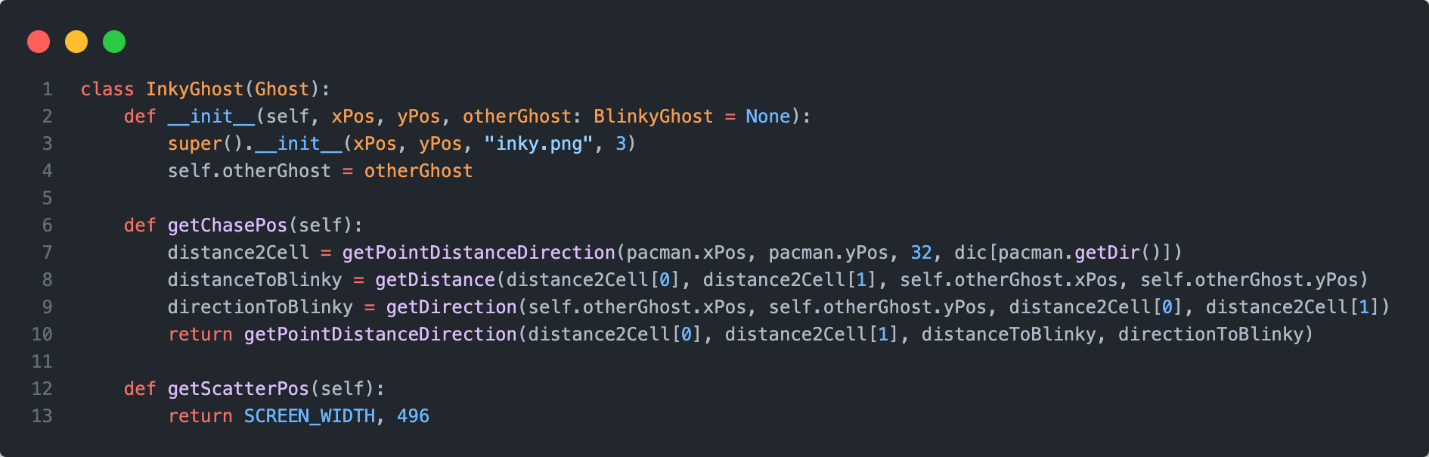
Hình 34: Khởi tạo và điều khiển ClydeGhost

ClydeGhost sẽ có hành vi đặc biệt khác với những chú ma khác. Khi khoảng cách giữa Clyde và Pacman lớn hơn hoặc bằng 256, Clyde sẽ chuyển sang ChaseMode và đuổi theo PacMan, ngược lại khi khoảng cách nhỏ hơn 256, Clyde sẽ quay lại vị trí scatter cố định (0, 496).



Hình 35: Hình ảnh của ClydeGhost

##### InkyGhost



Hình 36: Khởi tạo và điều khiển InkyGhost

* InkyGhost có hành vi phức tạp hơn so với Blinky và Clyde, vì nó không chỉ dựa vào vị trí của Pacman mà còn dựa vào vị trí của BlinkyGhost.
* Phương thức getChasePos tính toán một vị trí mà Inky sẽ đuổi theo, dựa vào vị trí của Pacman và khoảng cách từ Inky đến Blinky.
* Phương thức getScatterPos trả về tọa độ cố định ở góc dưới bên phải màn hình, là vị trí mà Inky sẽ di chuyển tới khi không đuổi theo Pacman (SCREEN\_WIDTH, 496).



Hình 37: Hình ảnh của InkyGhost

##### PinkyGhost



Hình 38: Khởi tạo và điều khiển PinkhGhost

* PinkhGhost có hành vi "Chase", nó sẽ luôn đuổi theo Pacman dựa trên vị trí và hướng di chuyển của Pacman.
* Phương thức getChasePos sẽ trả về tọa độ mà Pinky sẽ đuổi theo, tính toán dựa trên hướng di chuyển của Pacman.
* Phương thức getScatterPos trả về tọa độ cố định ở góc trên bên trái màn hình (0, 0).

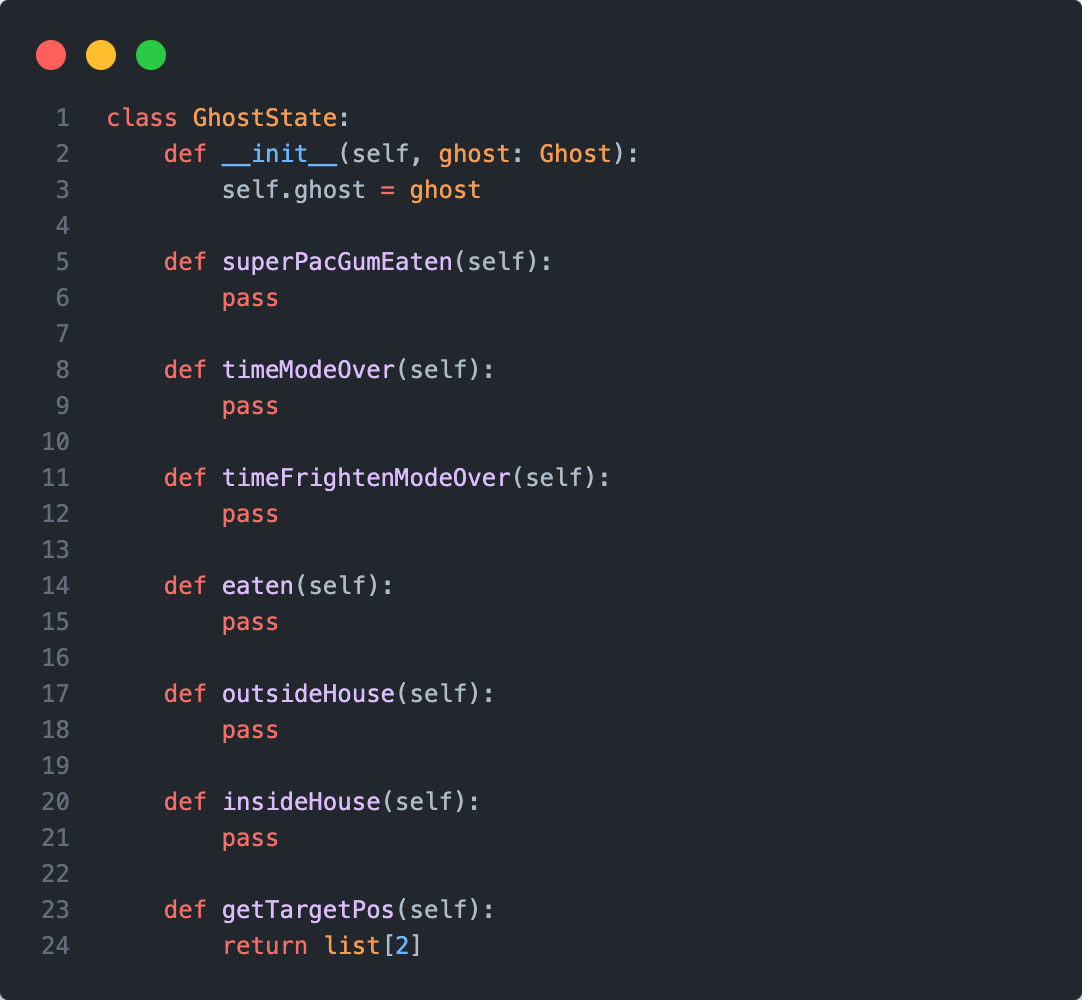


Hình 39: Hình ảnh của PinkhGhost

#### Quản lý và Điều Khiển Các Trạng Thái của Ma

##### Lớp GhostState

Lớp GhostState đại diện cho trạng thái cơ bản của con ma trong trò chơi. Các phương thức của lớp này bao gồm các hành vi và hành động chung, nhưng không thực hiện bất kỳ hành động cụ thể nào, vì chúng sẽ được triển khai trong các lớp con cụ thể.



A screenshot of a computer program

Description automatically generated

Hình 40: Khởi tạo trạng thái của ma

* Phương thức này tính toán hướng di chuyển tiếp theo của con ma dựa trên vị trí hiện tại của nó và mục tiêu (dựa vào getTargetPos).
* Trước khi tính toán hướng, phương thức kiểm tra xem con ma có đang trong trò chơi và trên lưới không.
* Tiếp theo, nó kiểm tra từng hướng (trái, phải, lên, xuống) để xem hướng nào có thể di chuyển mà không bị va chạm với tường.
* Cuối cùng, nó cập nhật tốc độ di chuyển (xSpeed, ySpeed) của con ma, đảm bảo rằng con ma sẽ di chuyển theo hướng tối ưu mà không gặp vật cản.

##### Lớp ChaseMode (Trạng thái đuổi bắt)

A screen shot of a computer program

Description automatically generated

Hình 41: Khởi tạo trạng thái đuổi bắt

Lớp ChaseMode được kế thừa từ lớp cha GhostState và triển khai hành vi của ghost khi nó đang ở trạng thái đuổi theo PacMan:

* superPacGumEaten: Ghosts sẽ chuyển trạng thái sang "sợ hãi" nếu Super Pac-Gum bị ăn.
* timeModeOver: Chuyển ghost sang trạng thái "Scatter" khi hết thời gian đuổi bắt.
* getTargetPos: Lấy vị trí mục tiêu của ghost khi đang đuổi bắt.

##### Lớp EatenMode (Trạng thái bị ăn)



Hình 42: Khởi tạo trạng thái bị ăn

EatenMode quản lý trạng thái khi ghost đã bị Pac-Man ăn:

* insideHouse: Chuyển ghost sang trạng thái "ở trong nhà" khi nó đã trở về.
* getTargetPos: Trả về vị trí của "nhà" mà ghost cần di chuyển về.
* computeNextDir: Tính toán hướng di chuyển của ghost về nhà, bỏ qua các tường trong quá trình quay lại.

##### Lớp FrightenedMode (Trạng thái sợ hãi)

A computer screen shot of a program

Description automatically generated

Hình 43: Khởi tạo trạng thái sợ hãi

FrightenedMode quản lý trạng thái ghost khi nó đang sợ hãi sau khi Pac-Man ăn Super Pac-Gum:

* eaten: Ghosts sẽ chuyển sang trạng thái EatenMode khi nó bị Pac-Man bắt.
* timeFrightenModeOver: Ghosts quay lại ChaseMode hoặc ScatterMode khi hết thời gian hoảng sợ.
* getTargetPos: Trả về vị trí ngẫu nhiên gần Ghosts, phản ánh trạng thái hoảng loạn của ghost.

##### Lớp HouseMode (Trạng thái ở nhà)



Hình 44: Khởi tạo trạng thái ở nhà

HouseMode kiểm soát trạng thái của ghost khi nó đang ở trong nhà (khu vực không thể bị Pac-Man bắt):

* outsideHouse: Chuyển trạng thái Ghosts sang ChaseMode hoặc ScatterMode khi nó ra khỏi nhà.
* getTargetPos: Trả về vị trí của Ghosts trong nhà.
* computeNextDir: Tính toán hướng di chuyển của Ghosts khi nó đang trong nhà.

##### Lớp ScatterMode: Trạng thái phân tán

A screen shot of a computer program

Description automatically generated

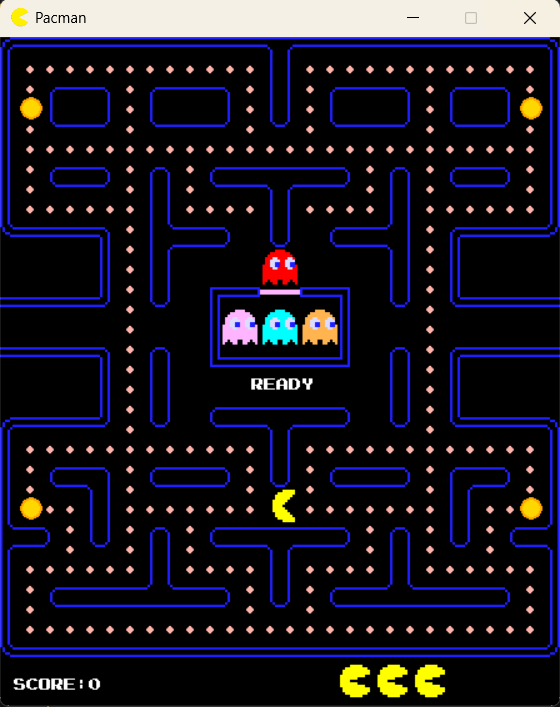
Hình 45: Khởi tạo trạng thái phân tán

Lớp ScatterMode quản lý trạng thái khi ghost không đuổi theo Pac-Man mà đi đến một vị trí cố định trên bản đồ:

* superPacGumEaten: Chuyển ghost sang trạng thái sợ hãi nếu Super Pac-Gum bị ăn.
* timeModeOver: Chuyển ghost sang trạng thái đuổi bắt khi hết thời gian phân tán.
* getTargetPos: Trả về vị trí phân tán của ghost.

### Tạo các viên điểm

Trong trò chơi Pacman, viên điểm (PacGum) và Super Pac-Gum đóng vai trò quan trọng trong việc tăng điểm cho người chơi và tạo ra các tính năng đặc biệt khi Pacman ăn chúng. Đoạn mã dưới đây mô tả cách tạo và vẽ các viên điểm này trên màn hình.



Hình 46: Hình ảnh có viên điểm PacGum và Super Pac-Gum trong trò chơi

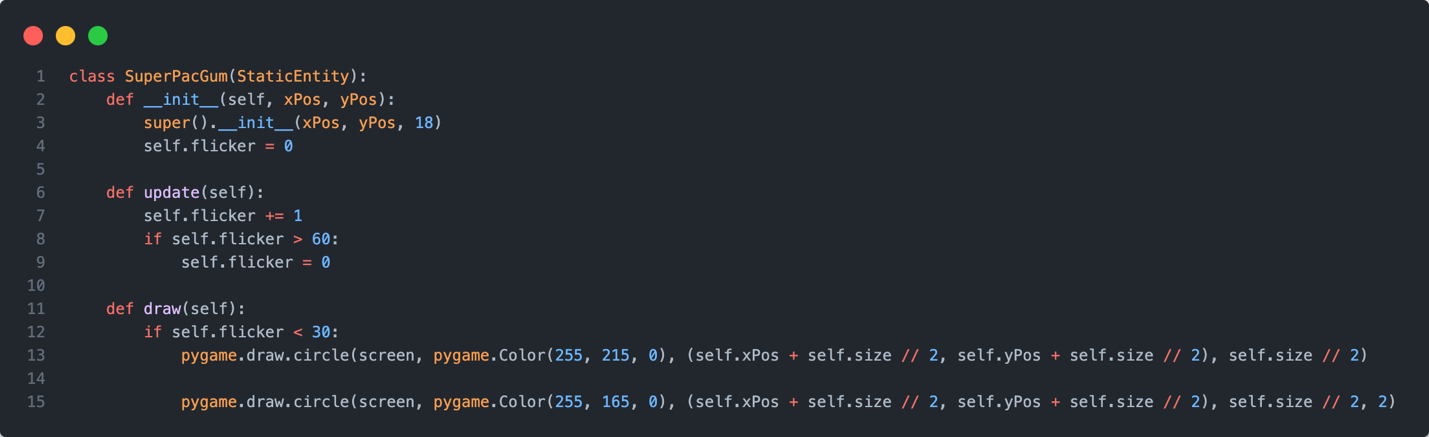
#### Lớp PacGum



Hình 47: Khởi tạo viên điểm PacGum

**Phương thức draw()**: Vẽ viên điểm dưới dạng hình tròn màu hồng nhạt (pygame.Color(255, 183, 174)), đảm bảo viên điểm hiển thị chính xác tại vị trí (xPos, yPos).

#### Lớp SuperPacGum



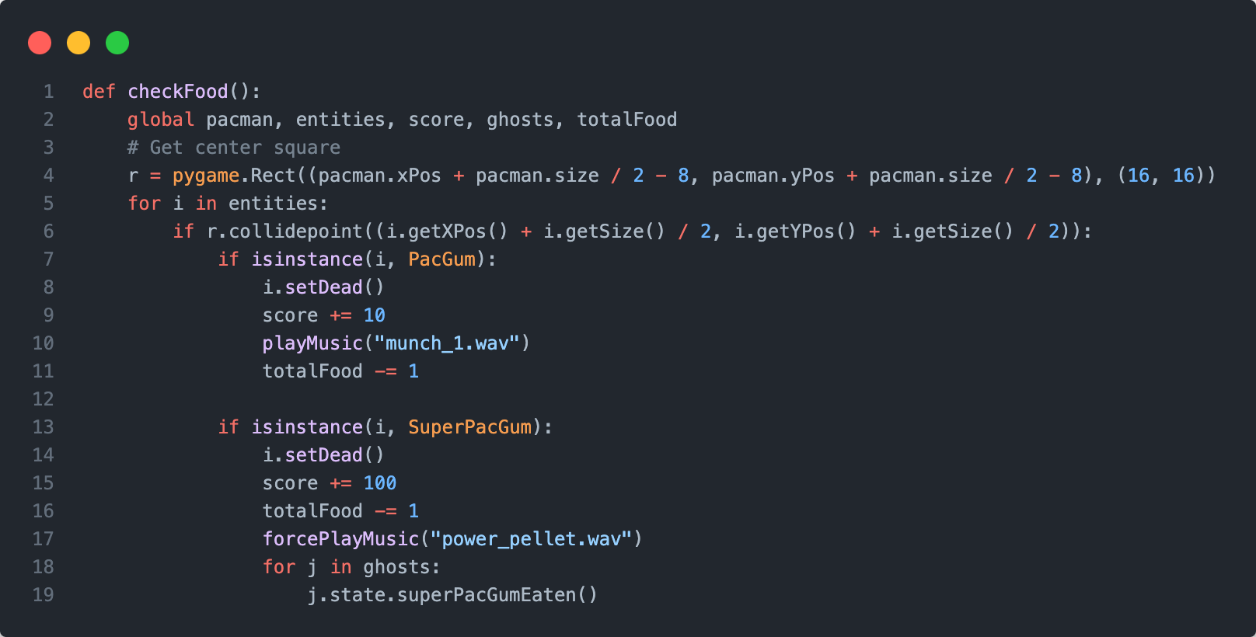
Hình 48: Khởi tạo viên điểm Super Pac-Gum

* **Phương thức update()**: Tạo hiệu ứng nhấp nháy bằng cách thay đổi giá trị flicker mỗi lần cập nhật. Khi giá trị flicker vượt quá 60, nó sẽ được đặt lại, tạo ra chu kỳ nhấp nháy.
* **Phương thức draw()**: Vẽ viên Super Pac-Gum với màu vàng (pygame.Color(255, 215, 0)) và viền cam (pygame.Color(255, 165, 0)) để làm nổi bật nó trên màn hình.

### Quản lý va chạm và tính điểm

Phần này bao gồm các phương thức chịu trách nhiệm xử lý va chạm giữa nhân vật Pacman và các đối tượng trong trò chơi, đồng thời tính toán và cập nhật điểm số dựa trên các va chạm này. Cụ thể, có hai loại va chạm chính được xử lý: va chạm với thức ăn và va chạm với các ma quái (ghosts).

#### Kiểm tra va chạm với thức ăn

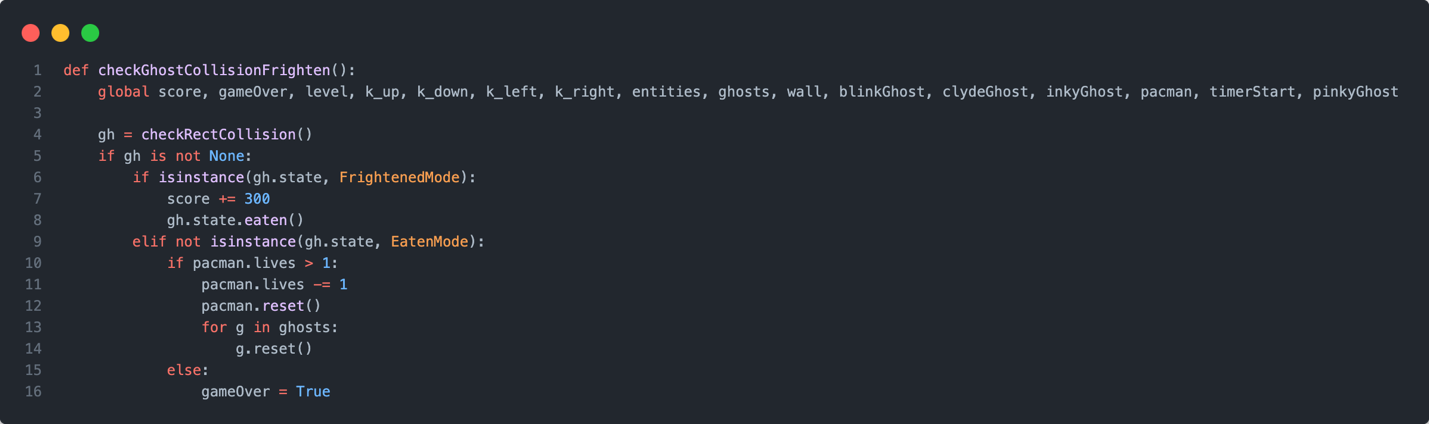


Hình 49: Hàm kiểm tra va chạm với thức ăn

Phương thức **checkFood()** được sử dụng để kiểm tra xem Pacman có ăn phải thức ăn hay không. Nếu Pacman va vào đối tượng là PacGum hoặc SuperPacGum, điểm số sẽ được cập nhật và đối tượng thức ăn sẽ bị loại bỏ khỏi trò chơi [3].

* **PacGum**: Mỗi lần Pacman ăn một viên PacGum, điểm số sẽ cộng thêm 10 điểm.
* **SuperPacGum**: Mỗi lần Pacman ăn một viên SuperPacGum, điểm số sẽ cộng thêm 100 điểm. Đồng thời, các ma quái sẽ bị làm cho hoảng sợ và không thể tấn công Pacman trong một khoảng thời gian.

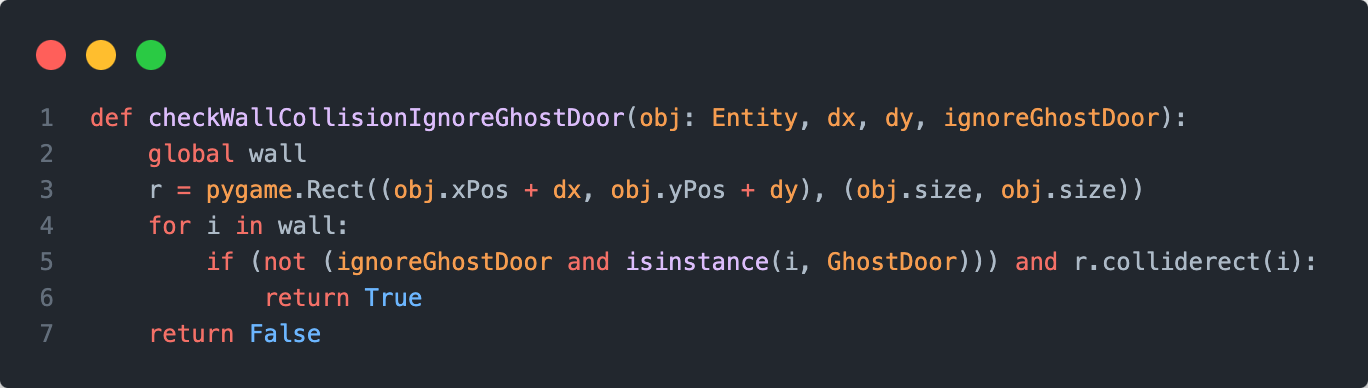
#### Kiểm tra va chạm với ma quái



Hình 50: Hàm kiểm tra va chạm với ma quái

Phương thức **checkGhostCollisionFrighten()** được sử dụng để xử lý va chạm giữa Pacman và các ma quái trong trò chơi. Khi Pacman va vào một ma quái trong chế độ **"FrightenedMode"** (chế độ sợ hãi), điểm số sẽ cộng thêm 300 và ma quái sẽ bị "ăn". Nếu không phải chế độ sợ hãi, và Pacman va phải ma quái, số mạng của Pacman sẽ giảm đi. Trò chơi sẽ kết thúc nếu Pacman không còn mạng.

#### Kiểm tra va chạm với tường



Hình 51: Hàm kiểm tra va chạm với tường

Phương thức **checkWallCollision()** này nhận vào một đối tượng Entity và kiểm tra sự va chạm của tất cả các đối tượng với các bức tường trong trò chơi, khi có va chạm, đối tượng sẽ không thể di chuyển qua bức tường, giúp ngăn chặn việc di chuyển sai lệch trong không gian trò chơi.

### Khởi tạo bản đồ và đối tượng trong game

Trong trò chơi Pacman, việc khởi tạo các đối tượng như Pacman, Ghosts, thức ăn và các tường chắn rất quan trọng để trò chơi có thể hoạt động đúng như mong đợi. **Phương thức init()** sẽ thực hiện việc khởi tạo này thông qua việc đọc dữ liệu từ một tệp CSV, nơi mỗi ký tự trong tệp đại diện cho một loại đối tượng khác nhau. Bản đồ và các đối tượng sẽ được khởi tạo và phân phối vào các danh sách thích hợp để trò chơi có thể sử dụng khi bắt đầu.

#### Mở và đọc tệp level.csv



Hình 52: hàm mở và đọc tệp level.csv

Tệp **level.csv** chứa thông tin về vị trí và loại các đối tượng trên bản đồ trò chơi. Tệp này được mở trong chế độ đọc và dữ liệu được chuyển thành danh sách level bằng cách sử dụng **csv.reader**. Mỗi dòng trong tệp đại diện cho một hàng trong bản đồ, mỗi cột trong dòng tương ứng với một ô trong bản đồ.

#### Khởi tạo kích thước ô



Hình 53: Khởi tạo kích thước ô

Mỗi ô trong bản đồ có kích thước 8x8 pixel. Biến size được đặt là 8 để xác định kích thước của mỗi ô trong trò chơi.

#### Duyệt qua dữ liệu để khởi tạo các đối tượng



Hình 54: Hàm duyệt qua từng dòng và cột trong danh sách level.

Hàm duyệt qua từng dòng và cột trong danh sách level. Tùy vào giá trị tại mỗi ô, nó sẽ khởi tạo các đối tượng tương ứng và thêm chúng vào danh sách entities:

* **PacGum** ("."): Là thức ăn nhỏ mà Pacman có thể ăn.
* **SuperPacGum** ("o"): Là thức ăn đặc biệt giúp Pacman có thể ăn ma.
* **Wall** ("x"): Là các tường chắn mà Pacman và ma không thể vượt qua.
* **GhostDoor** ("-"): Là cửa phòng ma, nơi các ma có thể ra vào.
* **Pacman** ("P"): Là nhân vật chính của trò chơi.
* Các ma **Blinky**, **Inky**, **Pinky**, **Clyde** ("b", "i", "p", "c"): Mỗi ma có vị trí và hành vi riêng biệt.

#### Phân loại các đối tượng vào danh sách



Hình 55: Phân loại các đối tượng vào danh sách

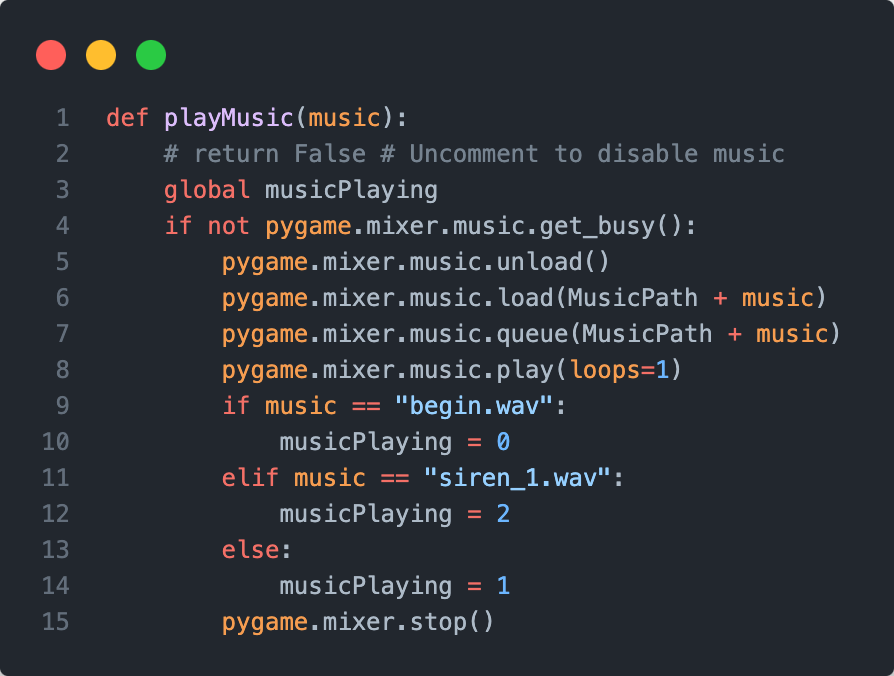
Sau khi các đối tượng được tạo ra, chúng sẽ được phân loại vào các danh sách như wall (tường), ghosts (ma), và entities (tất cả các đối tượng trong trò chơi). Việc phân loại này giúp dễ dàng quản lý và xử lý các đối tượng trong quá trình trò chơi.

### Tính năng phát nhạc trong trò chơi

Trong trò chơi, việc phát nhạc là một yếu tố quan trọng để tạo không khí và giúp người chơi có trải nghiệm thú vị. Hai hàm **playMusic()** và **forcePlayMusic()** được sử dụng để điều khiển việc phát nhạc trong trò chơi, đảm bảo nhạc nền và các hiệu ứng âm thanh được phát đúng cách. Cả hai hàm này đều sử dụng thư viện **pygame.mixer**, một công cụ mạnh mẽ để xử lý âm thanh trong trò chơi.

#### Hàm playMusic()

Hàm **playMusic()** có nhiệm vụ phát nhạc nền hoặc các hiệu ứng âm thanh trong trò chơi, nhưng chỉ khi không có nhạc nào đang được phát.

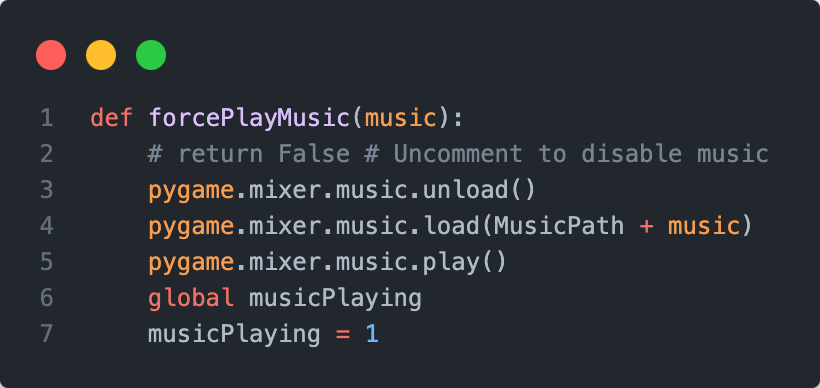


Hình 56: Hàm phát nhạc nền hoặc các hiệu ứng âm thanh

* Hàm bắt đầu bằng việc kiểm tra xem có nhạc nào đang được phát không thông qua **pygame.mixer.music.get\_busy().** Nếu không có nhạc nào đang phát, chương trình sẽ tiếp tục phát nhạc mới.
* Sau khi xác định rằng không có nhạc đang phát, hàm sẽ dừng và tải lại nhạc mới. Hàm **pygame.mixer.music.unload()** được sử dụng để xóa nhạc cũ nếu có, sau đó **pygame.mixer.music.load()** và **pygame.mixer.music.queue()** được sử dụng để tải và xếp hàng phát nhạc mới.
* Tùy thuộc vào tên tệp nhạc, giá trị của biến toàn cục **musicPlaying** sẽ được thay đổi. Ví dụ:
* Nếu nhạc là **begin.wav**, **musicPlaying** sẽ được gán giá trị 0.
* Nếu nhạc là **siren\_1.wav**, **musicPlaying** sẽ được gán giá trị 2.
* Đối với các tệp nhạc khác, **musicPlaying** sẽ được gán giá trị 1.
* Cuối cùng, **pygame.mixer.stop()** được gọi để đảm bảo không có nhạc nào khác đang phát khi phát nhạc mới.

#### Hàm forcePlayMusic()

Hàm forcePlayMusic() được sử dụng để phát nhạc mà không kiểm tra xem có nhạc nào đang phát hay không. Đây là một cách để buộc nhạc được phát ngay lập tức, ngay cả khi có nhạc khác đang chơi.



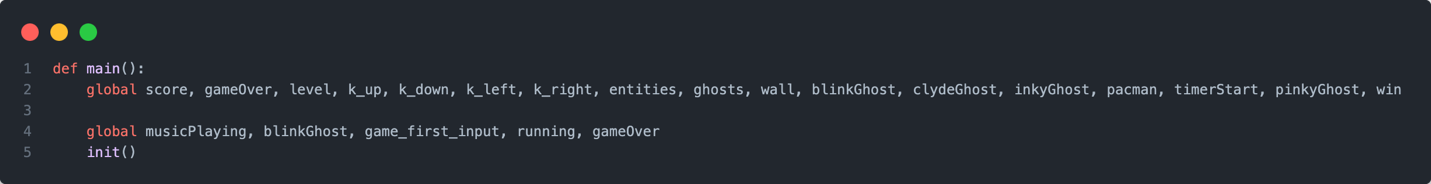
Hình 57: Hàm phát nhạc mà không kiểm tra xem có nhạc nào đang phát hay không.

* Hàm này cũng sử dụng **pygame.mixer.music.unload()** để dừng bất kỳ nhạc nào đang phát và sau đó tải nhạc mới từ tệp được chỉ định.
* Sau khi nhạc được tải, hàm sẽ phát nhạc mà không kiểm tra trạng thái hiện tại của nhạc đang phát. Điều này đảm bảo rằng nhạc được phát ngay lập tức mà không bị gián đoạn.
* Biến toàn cục **musicPlaying** sẽ được gán giá trị 1 khi nhạc mới được phát.

### Hàm main()

Hàm **main()** là trung tâm điều khiển của trò chơi, quản lý vòng lặp chính và các sự kiện trong suốt quá trình chơi. Nó chịu trách nhiệm khởi tạo trò chơi, cập nhật màn hình, xử lý các sự kiện từ người dùng, và thực thi các chức năng khác như điều khiển nhân vật, kiểm tra va chạm và xác định kết quả game (thắng hoặc thua).

#### Khởi tạo và Thiết lập



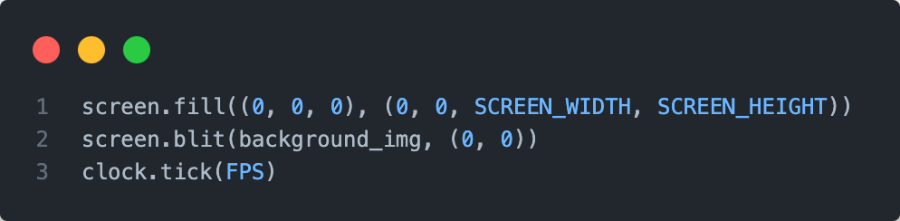
Hình 58: Hàm khởi tạo và Thiết lập

Khi bắt đầu, hàm **main()** sẽ gọi hàm **init()** để khởi tạo tất cả các đối tượng và thiết lập ban đầu như level, các entity (như Pacman, các bóng ma, tường, v.v.). Các giá trị toàn cục như điểm số, tình trạng trò chơi, và trạng thái của các nhân vật cũng được thiết lập lại.

#### Vòng lặp chính

Vòng lặp **while running:** là phần quan trọng của hàm **main()**. Đây là vòng lặp chạy liên tục trong suốt trò chơi, cho đến khi người chơi thoát hoặc kết thúc trò chơi. Mỗi lần lặp, hàm thực hiện các bước sau.

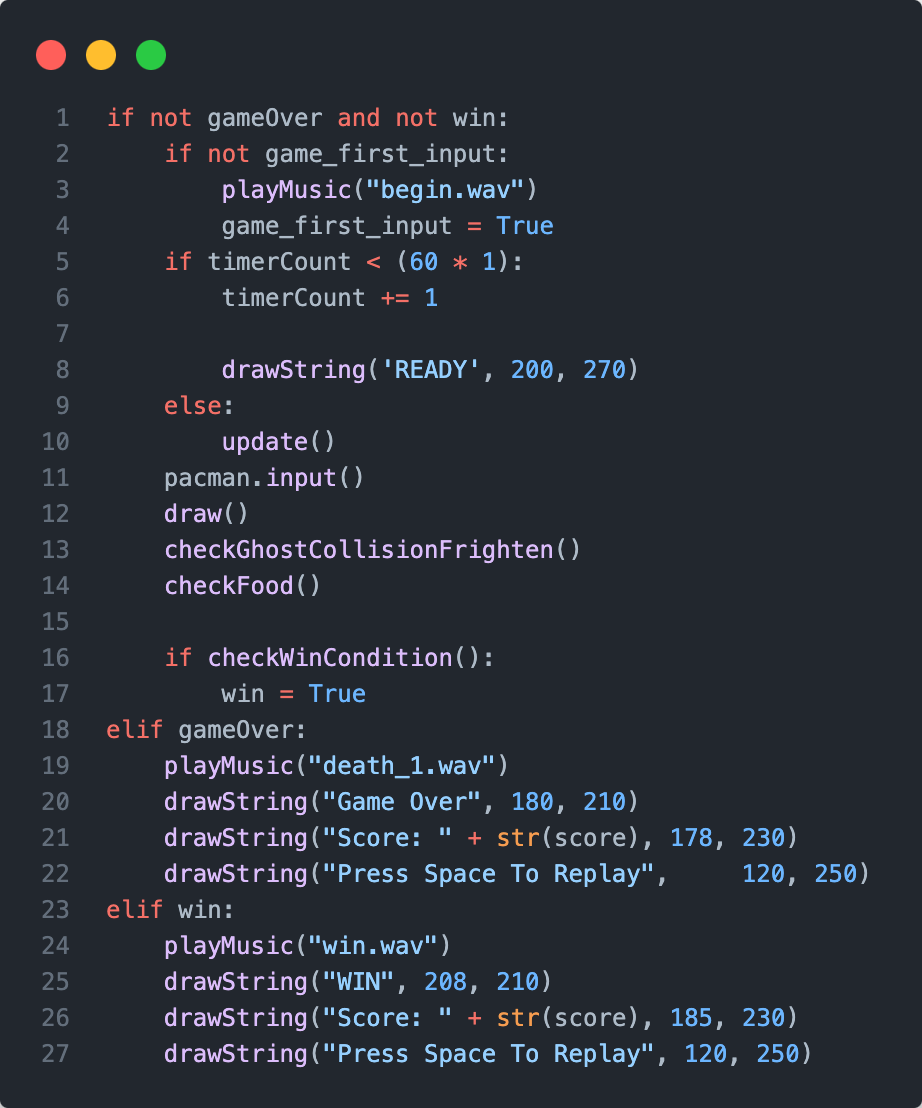
##### Cập nhật và Vẽ Màn Hình



Hình 59: Cập nhật và vẽ màn hình

Trong giai đoạn này, màn hình được xóa và vẽ lại toàn bộ trò chơi, bao gồm các đối tượng như Pacman, ma, và các viên điểm. Việc làm mới màn hình mỗi vòng lặp giúp trò chơi duy trì hình ảnh mới nhất và đảm bảo hiển thị mượt mà.

##### Kiểm tra trạng thái trò chơi

****

Hình 60: Kiểm tra trạng thái trò chơi

Dựa vào trạng thái của trò chơi (đang chơi, game over, hoặc thắng), màn hình sẽ hiển thị thông tin

* Nếu trò chơi chưa kết thúc và chưa thắng: Màn hình hiển thị "READY" trong vòng

1 giây, sau đó bắt đầu cập nhật trò chơi.

* Nếu trò chơi kết thúc (game over): Màn hình hiển thị "Game Over" cùng với điểm số và thông báo nhấn Space để chơi lại.
* Nếu người chơi thắng: Màn hình hiển thị "WIN" và điểm số, cùng với thông báo nhấn Space để chơi lại.

##### Xử lý sự kiện

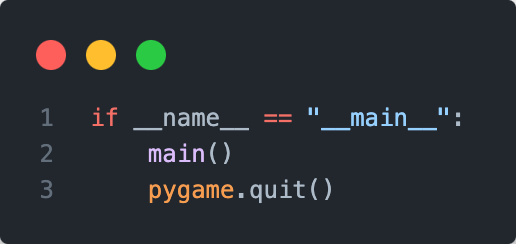


Hình 61: Hàm xử lý các sự kiện

Các sự kiện từ bàn phím (như nhấn và thả phím) được xử lý để di chuyển Pacman và kiểm tra các điều kiện khác:

* Nếu người chơi nhấn phím mũi tên lên, xuống, trái, phải, Pacman sẽ di chuyển theo các hướng tương ứng.
* Nếu người chơi nhấn phím Space, trò chơi sẽ được khởi động lại (gameOver và điểm số được reset, các đối tượng được khởi tạo lại).

#### Kết thúc



Hình 62: Vòng lặp kết thúc và pygame.quit()

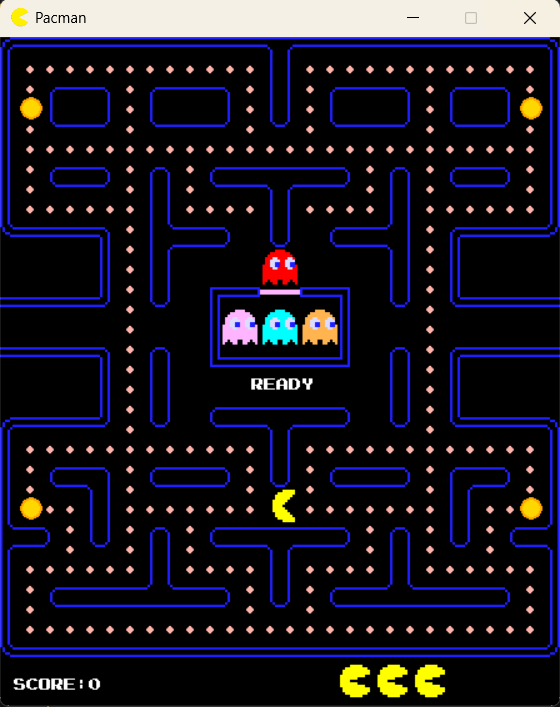
Khi người dùng đóng cửa sổ, vòng lặp kết thúc và **pygame.quit()** được gọi để giải phóng tài nguyên.

# Thực nghiệm và phân tích kết quả

## Kiểm thử các tính năng chính

Quá trình kiểm thử nhằm đảm bảo trò chơi Pacman hoạt động đúng theo yêu cầu thiết kế. Các tính năng chính được kiểm thử bao gồm:

* **Di chuyển của Pacman**:
  + **Mô tả**: Thử nghiệm khả năng điều khiển Pacman di chuyển mượt mà theo các phím mũi tên.
  + **Kết quả**: Pacman phản hồi tốt với các phím điều khiển, di chuyển đều theo hướng bấm và không có hiện tượng giật. Khi Pacman chạm đến biên của mê cung, quá trình di chuyển dừng lại như mong đợi, tránh trường hợp ra ngoài màn hình.
* **Va chạm với ma**:
  + **Mô tả**: Kiểm tra tính năng mất mạng khi Pacman va chạm với ma trong trạng thái bình thường và tính năng “ăn ma” khi Pacman đang ở trạng thái sức mạnh.
  + **Kết quả**: Khi Pacman va chạm với ma ở trạng thái bình thường, trò chơi giảm một mạng như thiết kế. Khi Pacman ăn viên sức mạnh và va vào ma, ma chuyển sang trạng thái bị ăn, và Pacman ghi thêm điểm. Các tính năng hoạt động ổn định.
* **Ăn viên điểm và viên sức mạnh**:
  + **Mô tả**: Thử nghiệm tính năng ghi điểm khi Pacman ăn viên điểm nhỏ và viên sức mạnh lớn. Sau khi ăn hết viên điểm, trò chơi kết thúc với thông báo thắng cuộc.
  + **Kết quả**: Pacman ghi được điểm tương ứng mỗi khi ăn viên điểm hoặc viên sức mạnh, hệ thống điểm số hoạt động chính xác. Trò chơi kết thúc chính xác khi tất cả viên điểm đã bị ăn hết.



Hình 63: Giao diện khi bắt đầu trò chơi.

A video game with a black background

Description automatically generated

Hình 64: Giao diện PacMan ăn các viên Pacgum

A video game with a black background

Description automatically generated

Hinh 65: Giao diện PacMan sau khi va chạm Ghost

PacMan sau khi va chạm Ghost thì mất một mạng, màn hình sẽ hiển thị mất đi một con PacMan ở phía dưới bên phải.

A video game with a square maze

Description automatically generated with medium confidence

Hình 66: Giao diện PacMan ăn được viên Super Pacgum

Khi PacMan ăn được viên Super Pacgum, các con Ghost sẽ biến thành màu xanh và di chuyển chậm lại, và PacMan có thể ăn các con Ghost bằng cách chạm vào chúng. Khi ấy con mắt của những con Ghost sẽ quay trở lại lồng.

A video game with blue lines

Description automatically generated

Hình 67: Giao diện khi ăn hết các viên Pacgum và chiến thắng

A screenshot of a video game

Description automatically generated

Hình 68: Giao diện kết thúc trò chơi khi thua hết 3 mạng

## Phân tích kết quả kiểm thử

Quá trình thử nghiệm cho thấy trò chơi Pacman hoạt động ổn định và đáp ứng được các yêu cầu về tính năng. Từ việc điều khiển nhân vật Pacman đến các tương tác với môi trường, các chức năng đều thực hiện đúng với thiết kế. Việc thử nghiệm giúp đảm bảo tính chính xác trong xử lý va chạm và kiểm soát điểm số, mang lại trải nghiệm liền mạch cho người chơi.

## Những thách thức và cách khắc phục

Trong quá trình triển khai trò chơi, nhóm đã gặp phải một số thách thức đáng kể:

* **Xử lý va chạm chính xác**: Việc phát hiện và xử lý va chạm giữa Pacman với các con ma và viên điểm yêu cầu tính toán chính xác vị trí. Để khắc phục, nhóm đã sử dụng hàm **pygame.Rect.colliderect()** để kiểm tra va chạm hiệu quả hơn, đồng thời tinh chỉnh kích thước và vị trí các đối tượng để va chạm xảy ra đúng thời điểm [2].
* **Điều chỉnh tốc độ và độ mượt mà của trò chơi**: Để Pacman và các con ma di chuyển mượt mà và ổn định, nhóm đã thiết lập tốc độ khung hình (FPS) ở mức 60 FPS. Thử nghiệm cho thấy tốc độ này giúp trò chơi không bị giật và đảm bảo trải nghiệm người dùng tốt hơn.
* **Thiết kế logic của từng con ma**: Mỗi con ma có hành vi khác nhau, đòi hỏi lập trình các thuật toán riêng để chúng di chuyển một cách thông minh và không đoán trước. Để khắc phục, nhóm đã nghiên cứu các chiến thuật di chuyển của từng con ma và triển khai chúng qua các chế độ khác nhau, như đuổi bắt, phân tán và hoảng sợ.

## Đánh giá hiệu suất và trải nghiệm người dùng

Trải nghiệm người dùng cho thấy trò chơi dễ dàng điều khiển và có độ mượt mà cao. Các tính năng như điều khiển Pacman, ăn điểm, và xử lý va chạm diễn ra nhạy và phản hồi nhanh. Nhờ việc tinh chỉnh hiệu suất, trò chơi duy trì được độ ổn định và hạn chế lỗi phát sinh. Trò chơi đã hoàn thành đúng yêu cầu, tạo ra một phiên bản Pacman cơ bản với độ khó vừa phải, giúp người chơi có trải nghiệm giải trí thoải mái.

# Kết luận và hướng phát triển

## Kết luận

Đồ án trò chơi Pacman đã đạt được những kết quả đáng kể trong việc phát triển một trò chơi tương tác, giúp nâng cao kỹ năng lập trình Python và sử dụng thư viện Pygame. Qua quá trình thực hiện, chúng em đã hiểu rõ hơn về các khía cạnh cơ bản của lập trình game, từ việc thiết kế giao diện đến xử lý sự kiện và kiểm tra va chạm. Trò chơi đã hoạt động ổn định, với các tính năng chính như di chuyển Pacman, ăn điểm, và tránh các con ma, mang lại trải nghiệm thú vị cho người chơi.

## Hướng phát triển

* **Thêm cấp độ:** Phát triển nhiều cấp độ khác nhau với độ khó tăng dần, tạo ra sự thách thức và khuyến khích người chơi tiếp tục tham gia [4] [5].
* **Tối ưu AI của các con ma:** Nâng cao trí thông minh nhân tạo của Ghosts để chúng có thể di chuyển thông minh hơn, theo dõi và chặn đường đi của Pacman hiệu quả hơn [5].
* **Thêm hiệu ứng âm thanh:** Tích hợp thêm nhiều hiệu ứng âm thanh để làm phong phú trải nghiệm chơi game, như âm thanh khi ăn điểm, âm thanh khi va chạm với ma, và nhạc nền hấp dẫn.
* **Chế độ chơi đa người:** Phát triển chế độ chơi đa người để người chơi có thể thi đấu với nhau, tạo ra sự tương tác xã hội và cạnh tranh.
* **Nâng cao đồ họa:** Cải thiện đồ họa của trò chơi, có thể thông qua việc sử dụng hình ảnh và hoạt hình chất lượng cao hơn.

Những cải tiến này không chỉ làm cho trò chơi trở nên hấp dẫn hơn mà còn mang đến cơ hội học hỏi và phát triển kỹ năng lập trình cho người phát triển.

# Tài liệu tham khảo

|  |  |
| --- | --- |
| [1] | "Pygame Community," [Online]. Available: https://www.pygame.org/docs/. |
| [2] | W. McGugan, Python Game Programming by Example, Birmingham: UK: Packt Publishing, 2015. |
| [3] | B. Burd, Beginning1 Programming with Python for Dummies, Hoboken: NJ: Packt Publishing, 2017. |
| [4] | S. Sweigart, Invent Your Own Computer Games with Python, 4th ed, Scotts Valley: CA: CreateSpace, 2016. |
| [5] | J. O'Reilly, Artificial Intelligence and Games, Cambridge: UK: Cambridge University Press, 2019. |