**TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHIỆP**

**KHOA ĐIỆN TỬ**

**Bộ môn: Công nghệ Thông tin.**


# BÀI TẬP KẾT THÚC MÔN HỌC

MÔN HỌC

LẬP TRÌNH PYTHON

Sinh viên: .HẦU THỊ THANH HUYỀN

Lớp: K58KTP

Giáo viên GIẢNG DẠY: NGUYỄN VĂN HUY

Link GitHub: https://github.com/hauthithanhhuyen/PY.git

Link Youtobe: https://youtu.be/OiAxgkkXyU8?si=ZS-waxj3yYVERQcX

Thái Nguyên- 2025

**TRƯỜNG ĐHKTCN** **CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM**

**KHOA ĐIỆN TỬ** **Độc lập - Tự do - Hạnh phúc**

# BÀI TẬP KẾT THÚC MÔN HỌC

## MÔN HỌC: LẬP TRÌNH PYTHON

## BỘ MÔN : CÔNG NGHỆ THÔNG TIN

*Sinh viên:* *Hầu Thị Thanh Huyền*

*Lớp*: K58KTP *Ngành: KTP*

*Giáo viên hướng dẫn:* *Nguyễn Văn Huy*

*Ngày giao đề:20/05/2025* *Ngày hoàn thành:*

*Tên đề tài :* Xây game Astrocrash với pygame

*Yêu cầu :*

Quay sprite, di chuyển, bắn tên lửa (Missile).

Collisions, di chuyển asteroid.

Phát sound và music.

GIÁO VIÊN HƯỚNG DẪN

*(Ký và ghi rõ họ tên)*

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....
.....
.....
.....
.....

*Thái Nguyên, ngày….tháng…..năm 20....*

GIÁO VIÊN HƯỚNG DẪN

*(Ký ghi rõ họ tên)*

MỤC LỤC

Contents

# LỜI CAM ĐOAN

Tôi xin cam đoan bài tập lơn“Xây game Astrocrash” này là công trình nghiên cứu của riêng tôi. Các số liệu sử dụng trong luận văn là trung thực. Các kết quả nghiên cứu được trình bày trong đồ án chưa từng được công bố tại bất kỳ công trình nào khác.

Tên sinh viên

Hầu thị thanh huyền

# DANH MỤC CÁC BẢNG VÀ HÌNH VẼ, ĐỒ THỊ

# LỜI NÓI ĐẦU

Trong thời đại số hóa hiện nay, lập trình game không chỉ là lĩnh vực giải trí mà còn là môi trường thực tiễn để rèn luyện kỹ năng lập trình, tư duy thuật toán và khả năng xử lý đồ họa thời gian thực. Python là một trong những ngôn ngữ phổ biến, dễ học và đặc biệt mạnh mẽ khi kết hợp với thư viện Pygame để xây dựng các trò chơi 2D đơn giản nhưng sinh động.

Với mục tiêu ứng dụng kiến thức lập trình đã học, đề tài "Xây dựng game Astrocrash (Chapter 12) với Pygame" được lựa chọn. Trong game này, người chơi sẽ điều khiển một tàu vũ trụ di chuyển linh hoạt, bắn hạ các thiên thạch đang bay tới, đồng thời trải nghiệm các hiệu ứng âm thanh sống động. Thông qua đề tài, người học có cơ hội thực hành các khía cạnh quan trọng như: quản lý đối tượng (sprite), phát hiện va chạm, điều khiển bằng bàn phím và xử lý âm thanh.

Em xin chân thành cảm ơn sự quan tâm giúp đỡ của thầy Nguyễn Văn Huy đã giúp đỡ em hoàn thành đề tài này.

# CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

## 1.1. Giới thiệu

Trong thời đại công nghệ số, việc ứng dụng lập trình để xây dựng các trò chơi không chỉ giúp rèn luyện tư duy logic mà còn khơi dậy tính sáng tạo và đam mê công nghệ.

Bài tập lớn này yêu cầu sinh viên phát triển một trò chơi đơn giản có tên Astrocrash, dựa trên thư viện Pygame – một thư viện mạnh mẽ và phổ biến trong phát triển game 2D với Python.

Astrocrash là một trò chơi mô phỏng bối cảnh ngoài không gian, nơi người chơi điều khiển một chiếc tàu vũ trụ với nhiệm vụ tiêu diệt các thiên thạch đang lao xuống. Tàu có thể di chuyển tự do theo bốn hướng và bắn tên lửa để phá hủy kẻ thù. Mỗi thiên thạch bị phá hủy sẽ đem lại điểm số và tạo hiệu ứng nổ đẹp mắt kèm theo âm thanh sống động

## 1.2. Mục tiêu của đề tài

Đề tài này hướng đến mục tiêu xây dựng một trò chơi 2D đơn giản nhưng hấp dẫn mang tên "Astrocrash" bằng thư viện Pygame. Trò chơi sẽ có các tính năng cơ bản sau:

Điều khiển tàu vũ trụ: Người chơi có khả năng điều khiển một tàu vũ trụ di chuyển và xoay trên màn hình thông qua các phím mũi tên.

Bắn tên lửa: Tàu vũ trụ có khả năng bắn ra các tên lửa để tiêu diệt các thiên thạch đang di chuyển.

Thiên thạch di chuyển: Các thiên thạch sẽ xuất hiện ngẫu nhiên và di chuyển trên màn hình, tạo ra thử thách cho người chơi.

Va chạm: Hệ thống sẽ phát hiện các va chạm giữa tàu vũ trụ và thiên thạch, cũng như giữa tên lửa và thiên thạch.

Hiệu ứng nổ: Khi tên lửa bắn trúng thiên thạch, một hiệu ứng nổ đẹp mắt sẽ được hiển thị kèm theo âm thanh.

Âm thanh: Trò chơi sẽ tích hợp nhạc nền và các hiệu ứng âm thanh (tiếng bắn, tiếng nổ) để tăng tính trải nghiệm.

Điểm số: Người chơi sẽ được cộng điểm khi tiêu diệt thành công các thiên thạch.

## 1.3. Tính năng và Thách thức

*Tính năng nổi bật:*

Khả năng điều khiển linh hoạt tàu vũ trụ và bắn tên lửa theo hướng xoay.

Sự đa dạng trong chuyển động của thiên thạch, tạo ra các tình huống bất ngờ.

Phản hồi trực quan và sinh động thông qua hiệu ứng nổ và âm thanh.

Cơ chế tính điểm đơn giản, khuyến khích người chơi tiêu diệt nhiều thiên thạch.

Thách thức:

Xử lý chuyển động và xoay: Đảm bảo tàu vũ trụ di chuyển và xoay mượt mà, phản ứng chính xác vớiInput của người chơi.

Tính toán quỹ đạo tên lửa: Xác định hướng di chuyển của tên lửa dựa trên góc xoay của tàu.

*Phát hiện va chạm*: Xây dựng cơ chế phát hiện va chạm chính xác giữa các đối tượng (tàu - thiên thạch, tên lửa - thiên thạch).

Quản lý và hiển thị hiệu ứng nổ: Đảm bảo hiệu ứng nổ hiển thị đúng vị trí và thời gian.

*Tích hợp âm thanh*: Quản lý việc phát nhạc nền liên tục và các hiệu ứng âm thanh đồng thời một cách hiệu quả.

*Tối ưu hóa hiệu suất*: Đảm bảo trò chơi chạy mượt mà trên các cấu hình máy tính khác nhau.

1.4. Kiến thức vận dụng

Để hoàn thành bài tập lớn này, người thực hiện sẽ vận dụng các kiến thức sau:

Ngôn ngữ lập trình Python: Nền tảng chính để xây dựng trò chơi.

- Thư viện Pygame:

  Pygame.init(): Khởi tạo các module của Pygame.

  Pygame.display: Quản lý cửa sổ hiển thị.

  Pygame.Surface: Tạo và quản lý các bề mặt vẽ.

  Pygame.image: Tải và xử lý hình ảnh.

  Pygame.sprite: Quản lý các đối tượng game (sprites) và xử lý va chạm.

  Pygame.draw: Vẽ các hình cơ bản (trong trường hợp debug hoặc tạo hình đơn giản).

  Pygame.event: Xử lý các sự kiệnInput (bàn phím, chuột).

  Pygame.time: Quản lý thời gian và tốc độ khung hình.

  Pygame.mixer và pygame.mixer.music: Tải và phát âm thanh và nhạc nền.

  Pygame.font: Hiển thị văn bản (điểm số).

Pygame.transform: Xoay và масштабирование hình ảnh.

- Đại số và lượng giác cơ bản: Tính toán hướng di chuyển của tên lửa dựa trên góc xoay (sử dụng hàm sin và cos).

Lập trình hướng đối tượng (OOP): Tổ chức code thành các lớp (ví dụ: Ship, Asteroid, Missile, Explosion) để dễ quản lý và tái sử dụng.

- Cấu trúc dữ liệu: Sử dụng list để lưu trữ các khung hình của hiệu ứng nổ, pygame.sprite.Group để quản lý tập hợp các đối tượng game.

Thuật toán cơ bản: Thuật toán cập nhật vị trí, thuật toán phát hiện va chạm.

# CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

## *2.1.Giới thiệu tổng quan về Pygame*

- Pygame là một bộ thư viện đa nền tảng, mã nguồn mở được thiết kế chuyên biệt để phát triển các trò chơi điện tử bằng ngôn ngữ Python. Ra đời dựa trên thư viện SDL (Simple DirectMedia Layer), Pygame đã phát triển mạnh mẽ với một cộng đồng hỗ trợ lớn và nguồn tài liệu phong phú.
- Ưu điểm nổi bật của Pygame bao gồm tính dễ học, cú pháp trực quan của Python, sự linh hoạt trong việc xây dựng nhiều thể loại game 2D, và một cộng đồng người dùng rộng rãi sẵn sàng chia sẻ và giúp đỡ.

## *2.2. Lập trình hướng đối tượng (Object-Oriented Programming - OOP)*

a) Khái niệm cơ bản:

- Đối tượng (Object): Một thực thể có thuộc tính (attributes) và hành vi (methods). Trong game, tàu, thiên thạch, tên lửa, vụ nổ đều là các đối tượng.
- Lớp (Class): Khuôn mẫu để tạo ra các đối tượng. Ví dụ, lớp Ship định nghĩa các thuộc tính và hành vi chung của tất cả các tàu trong game.
- Thuộc tính (Attributes): Các biến lưu trữ thông tin về đối tượng (ví dụ: vị trí, kích thước, tốc độ).
- Phương thức (Methods): Các hàm định nghĩa hành vi của đối tượng (ví dụ: di chuyển, xoay, bắn).
- Tính đóng gói (Encapsulation): Gói dữ liệu (thuộc tính) và phương thức thao tác trên dữ liệu đó lại với nhau.
- Tính kế thừa (Inheritance): Cho phép một lớp (lớp con) kế thừa các thuộc tính và phương thức của một lớp khác (lớp cha). (Trong game này có thể không sử dụng kế thừa phức tạp, nhưng cần hiểu khái niệm).
- Tính đa hình (Polymorphism): Khả năng các đối tượng thuộc các lớp khác nhau có thể phản ứng với cùng một thông điệp theo những cách khác nhau. (Có thể không thể hiện rõ ràng trong game đơn giản này).
- Ứng dụng của OOP trong game: Giúp tổ chức code một cách logic, dễ quản lý, bảo trì và mở rộng.

*2.3. Các cấu trúc dữ liệu cơ bản*
a) Danh sách (List):

Định nghĩa: Một chuỗi các phần tử có thứ tự, có thể chứa các kiểu dữ liệu khác nhau.

Ứng dụng trong game: Lưu trữ các khung hình của hiệu ứng nổ, quản lý tạm thời các đối tượng (nếu cần).

b) Từ điển (Dictionary - Dict):

Định nghĩa: Một tập hợp các cặp key-value, cho phép truy cập giá trị bằng key.Ứng dụng trong game: Có thể dùng để cấu hình các thuộc tính của đối tượng hoặc lưu trữ các cài đặt game (mặc dù trong game đơn giản này có thể không cần thiết).

c) Tập hợp (Set):

Định nghĩa: Một tập hợp các phần tử duy nhất, không có thứ tự.

Ứng dụng trong game: Có thể dùng để theo dõi các đối tượng duy nhất (mặc dù pygame.sprite.Group thường được ưu tiên hơn).

d) Nhóm Sprite (pygame.sprite.Group):

Định nghĩa: Một container để quản lý nhiều đối tượng pygame.sprite.Sprite.

Ứng dụng trong game: Quản lý tất cả các đối tượng có thể hiển thị và tương tác (tàu, thiên thạch, tên lửa, vụ nổ), cung cấp các phương thức tiện lợi để cập nhật và vẽ tất cả các sprite cùng một lúc, cũng như kiểm tra va chạm.

*2.4 Nội dung chuyên môn sẽ sử dụng trong chương trình*
-Biến đơn (int, float, bool, str): Lưu trữ các giá trị đơn lẻ như điểm số (score), trạng thái game (running, game_over), kích thước màn hình (WIDTH, HEIGHT), đường dẫn file

-Danh sách (list): Được sử dụng để lưu trữ một chuỗi các phần tử có thứ tự. Ví dụ: explosion_anim có thể là một danh sách các hình ảnh cho hiệu ứng nổ (mặc dù trong đoạn mã này, việc load animation có thể nằm ở một file khác).

-Tuple (tuple): Được sử dụng để lưu trữ một chuỗi các phần tử không thay đổi. Ví dụ: kích thước màn hình (WIDTH, HEIGHT), màu sắc (255, 255, 255).

-Đối tượng (object): Là các thể hiện của các lớp. Ví dụ: ship là một đối tượng của lớp Ship, asteroid là đối tượng của lớp Asteroid. Các đối tượng này chứa dữ liệu (thuộc tính) và hành động (phương thức).

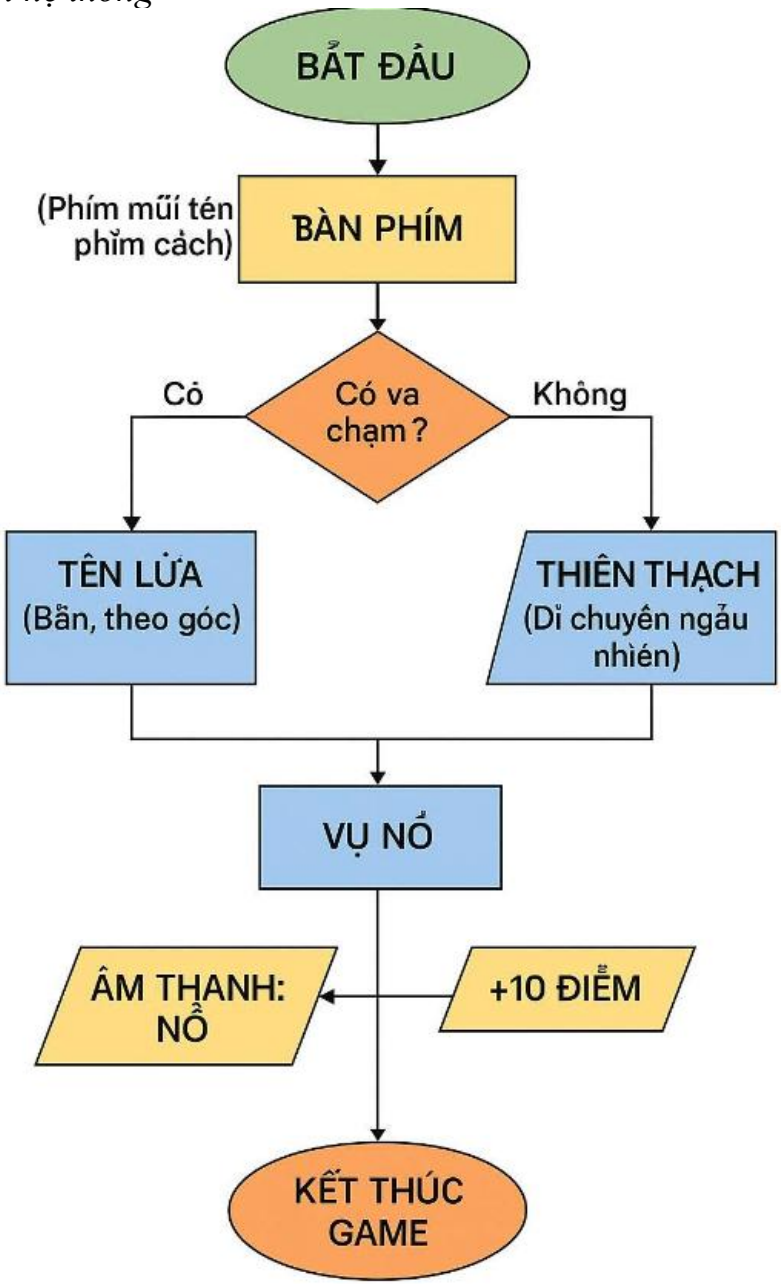
-Nhóm Sprite (pygame.sprite.Group): Một cấu trúc dữ liệu đặc biệt của Pygame để quản lý nhiều đối tượng pygame.sprite.Sprite. Nó cung cấp các phương thức để cập nhật, vẽ và xử lý va chạm hàng loạt các đối tượng. Ví dụ: ship_group, missile_group, asteroid_group, explosion_group.

*Tóm tắt:*

*Các nội dung chuyên môn sử dụng trong game Astrocrash gồm kiến thức về Python, Pygame, lập trình hướng đối tượng, xử lý đồ họa – âm thanh – va chạm, cấu trúc vòng lặp game, sinh ngẫu nhiên và quản lý nhóm đối tượng. Đây là một bài tập thực tế hữu ích giúp sinh viên rèn luyện kỹ năng lập trình, tư duy logic và sáng tạo trong phát triển trò chơi*

# CHƯƠNG 3. THIẾT KẾ XÂY DỰNG CHƯƠNG TRÌNH

## *3.1. Sơ đồ khối hệ thống*




*Hình 3.1.1 Sơ đồ khối hệ thống*

### *3.1.1. Mô tả các module chính trong chương trình*

a)ModuleKhởi tạo game

Khởi tạo Pygame, màn hình (800x600), tải hình nền, âm thanh, phông chữ.

Tạo Ship, 5 Asteroid, các nhóm sprite, đặt score = 0, game_over = False.

b)Module Xử lý sự kiện

Nhận phím (LEFT, RIGHT, UP, DOWN, SPACE) và sự kiện (QUIT, R).

Cập nhật Ship (xoay, di chuyển, bắn), xử lý thoát hoặc khởi động lại.

c)Module Quản lý tên lửa

Tạo Missile khi bắn, cập nhật vị trí, xóa nếu ra ngoài màn hình.

d)Module Quản lý thiên thạch

Tạo Asteroid, cập nhật vị trí (di chuyển xuống), đặt lại nếu ra ngoài.

e)Module Cập nhật trạng thái

Cập nhật Ship, Missile, Asteroid, Explosion.

Kiểm tra va chạm: Missile vs Asteroid (tăng điểm), Ship vs Asteroid (game over).

f)Module Giao diện

Vẽ nền, sprite, điểm số, thông báo "GAME OVER".

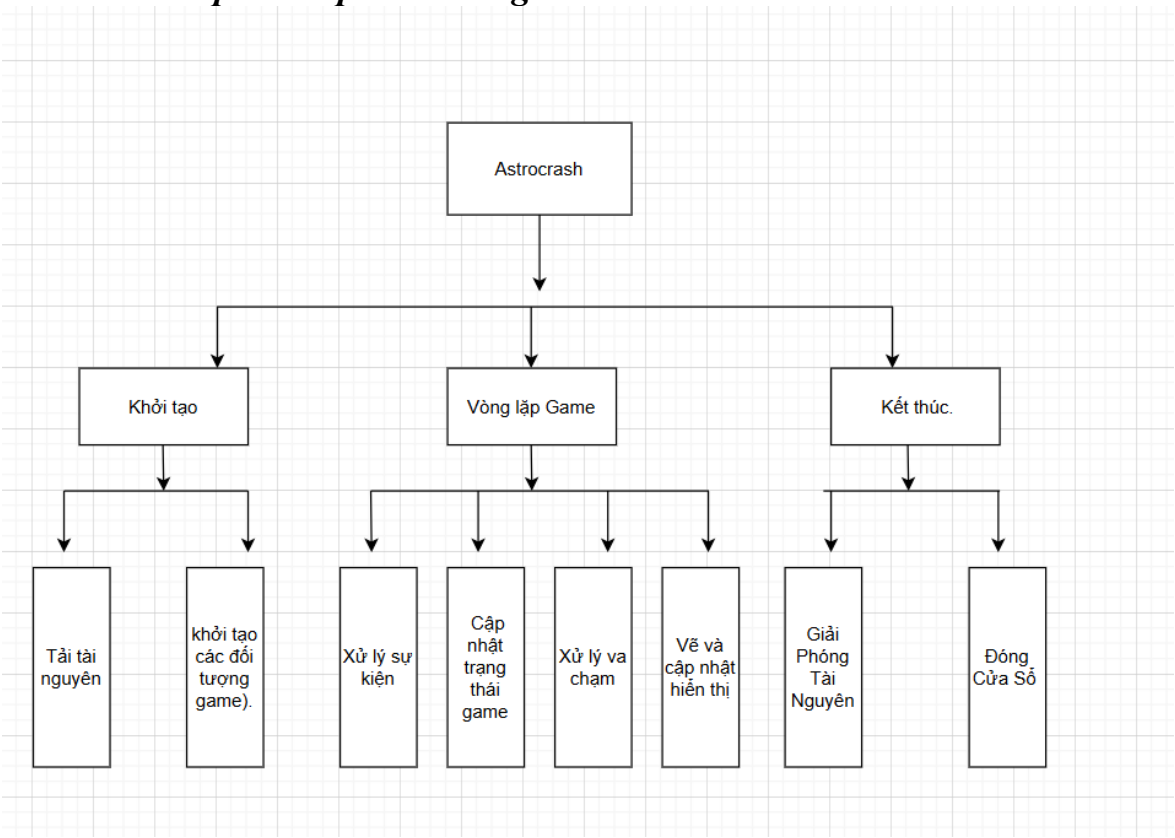
Cập nhật màn hình (pygame.display.flip()).

g)Module Điều khiển chính

Quản lý vòng lặp game, kiểm tra thoát, game_over, khởi động lại.

Điều phối các module khác.

### *3.1.2 Biểu đồ phân cấp chức năng*




*Hình 3.1.2 Biểu đồ phân cấp chức năng của game*

Chức năng

Khởi tạo Pygame:

- Chức năng: Thiết lập môi trường Pygame, bao gồm khởi tạo các module cần thiết (ví dụ: pygame.init()).

Tải Tài Nguyên-

- Tải Hình Nền:Chức năng: Đọc file hình ảnh nền từ thư mục assets và chuẩn bị để hiển thị. Nếu không tìm thấy, tạo một nền mặc định.
- Tải Nhạc Nền:Chức năng: Đọc file âm thanh nhạc nền từ thư mục assets và chuẩn bị để phát.
- Tải Âm Thanh Nổ:Chức năng: Đọc file âm thanh hiệu ứng nổ từ thư mục assets và chuẩn bị để phát khi có va chạm.
- Khởi Tạo Font:Chức năng: Thiết lập font chữ để hiển thị điểm số và các thông báo khác trên màn hình.
- Khởi Tạo Game:

- -Tạo Tàu: Chức năng: Tạo một đối tượng (instance) của lớp Ship, đặt vị trí và các thuộc tính ban đầu cho tàu của người chơi.
- -Tạo Nhóm Sprite: Chức năng: Tạo các nhóm (pygame.sprite.Group) để quản lý các đối tượng game (tàu, tên lửa, thiên thạch, vụ nổ), giúp dễ dàng cập nhật và vẽ chúng.
- Tạo Thiên Thạch Ban Đầu: Chức năng: Tạo một số lượng thiên thạch ban đầu và thêm chúng vào nhóm quản lý thiên thạch (asteroid_group).
- Đặt Điểm Số = 0: Chức năng: Thiết lập điểm số của người chơi về 0 khi bắt đầu game mới.

Vòng lặp Game:

Xử Lý Sự Kiện:

- Lấy Sự Kiện: Chức năng: Thu thập tất cả các sự kiện đã xảy ra kể từ lần kiểm tra trước (ví dụ: phím bấm, di chuyển chuột, đóng cửa sổ).
- Sự Kiện Thoát: Chức năng: Kiểm tra xem có sự kiện yêu cầu thoát game (ví dụ: người dùng nhấn nút đóng cửa sổ) hay không.
- Game Over?: Chức năng: Kiểm tra xem biến game_over có giá trị True hay không, xác định xem game đã kết thúc chưa. C1d. Xử Lý Input (khi không phải Game Over):
- Phím SPACE?: Chức năng: Kiểm tra xem phím cách (thường dùng để bắn) có đang được nhấn hay không.
- Tàu Bắn: Chức năng: Gọi phương thức shoot() của đối tượng tàu để tạo ra một tên lửa mới.
- Cập Nhật Tàu (Phím): Chức năng: Gọi phương thức update() của đối tượng tàu, nhận thông tin về các phím đang được nhấn để di chuyển hoặc xoay tàu. Phím R? (khi Game Over): Chức năng: Kiểm tra xem phím 'R' (thường dùng để chơi lại) có đang được nhấn hay không khi game đã kết thúc.

Xử Lý Va Chạm:

*Tên Lửa - Thiên Thạch:*

- Va Chạm: Kiểm tra va chạm.
- Phát Âm Thanh Nổ: Phát tiếng nổ.
- Tạo Vụ Nổ: Tạo hiệu ứng nổ.

- Xóa Tên Lửa: Loại bỏ tên lửa.
- Xóa Thiên Thạch: Loại bỏ thiên thạch.
- Tạo Thiên Thạch Mới: Tạo thiên thạch mới
- Tăng Điểm Số: Cộng điểm.

*Tàu - Thiên Thạch:*

- Va Chạm: Kiểm tra va chạm.
- Phát Âm Thanh Nổ: Phát tiếng nổ.
- Tạo Vụ Nổ: Tạo hiệu ứng nổ. Đặt Game Over = True: Kết thúc game.

Vẽ

- Xóa Màn Hình: Chức năng: Đổ một màu nền lên toàn bộ màn hình để xóa các hình ảnh cũ từ khung hình trước.
- Vẽ Nền: Chức năng: Vẽ hình ảnh nền lên màn hình. Vẽ Tàu: Chức năng: Vẽ hình ảnh tàu của người chơi lên màn hình tại vị trí hiện tại của nó.
- Vẽ Tên Lửa: Chức năng: Vẽ hình ảnh của tất cả các tên lửa đang bay lên màn hình tại vị trí hiện tại của chúng.
- Vẽ Thiên Thạch: Chức năng: Vẽ hình ảnh của tất cả các thiên thạch lên màn hình tại vị trí hiện tại của chúng.
- Vẽ Vụ Nổ: Chức năng: Vẽ khung hình hiện tại của tất cả các vụ nổ đang diễn ra lên màn hình tại vị trí của chúng.
- Vẽ Điểm Số: Chức năng: Sử dụng font đã khởi tạo để vẽ giá trị của biến score lên một vị trí cố định trên màn hình.
- Vẽ Thông Báo Game Over: Chức năng: Nếu game_over là True, vẽ một thông báo (ví dụ: "GAME OVER - Press R to Restart") lên giữa màn hình.
- Cập Nhật Hiển Thị: Chức năng: Cập nhật toàn bộ màn hình để hiển thị những gì đã được vẽ ở bước trên (pygame.display.flip()).

Kết thúc:

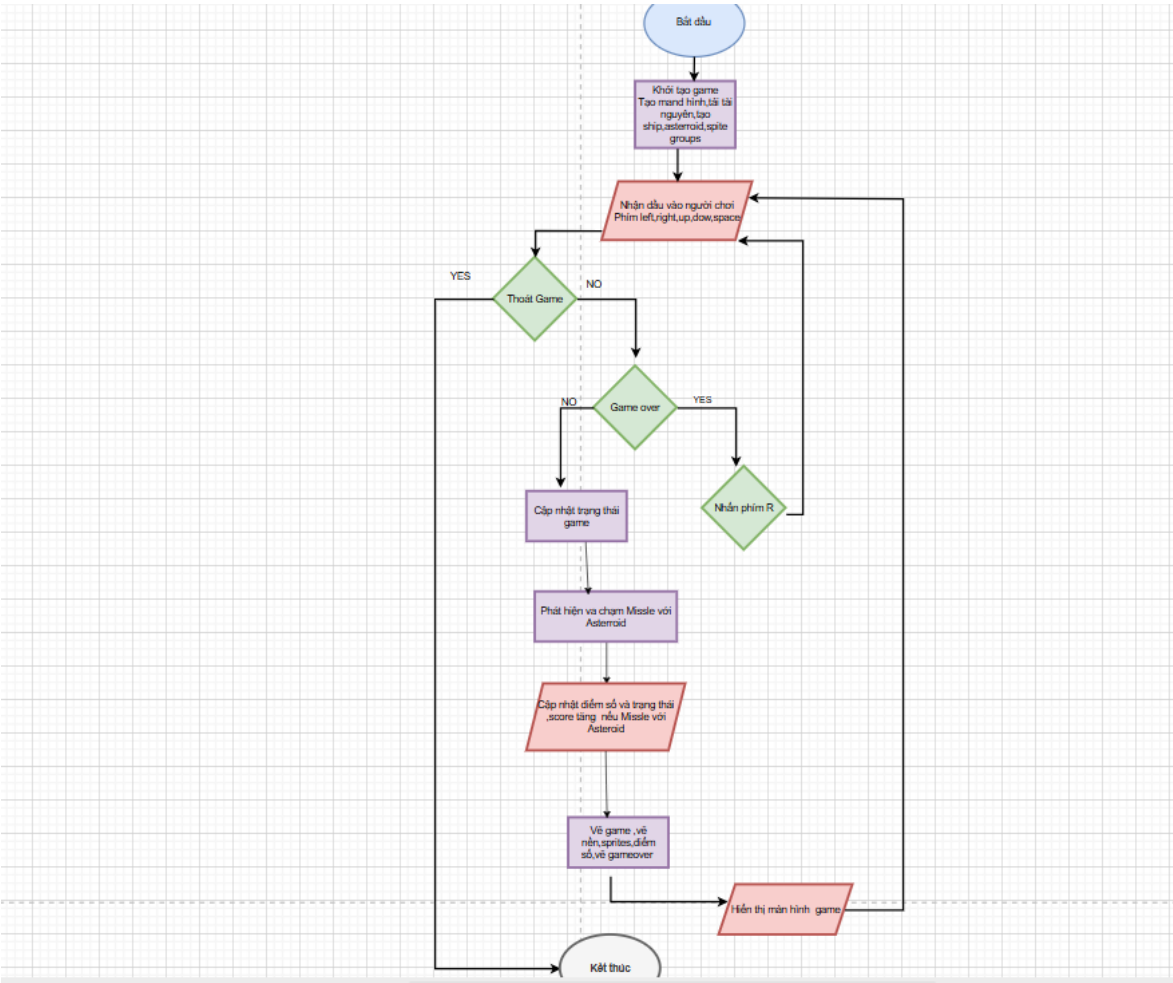
Giải Phóng Tài Nguyên (ngầm khi thoát):

Chức năng: Khi chương trình Python kết thúc, hệ điều hành sẽ tự động giải phóng bộ nhớ và các tài nguyên mà chương trình đã sử dụng.

Đóng Cửa Sổ (ngầm khi thoát):

Chức năng: Khi chương trình Python kết thúc, cửa sổ game sẽ tự động đóng lại.

3.2. Sơ đồ khối các thuật toán chính




*Hình 3.1Hình Sơ đồ khối các thuật toán chính*

### *3.2.1 Quan hệ đầu vào ra giữa các khối, chức năng từng thuật toán*

a) Bắt đầu

Chức năng: Điểm khởi đầu của chương trình, nơi game bắt đầu chạy.

Đầu vào: Không có.

Đầu ra: Chuyển đến Khởi tạo game.

b) Khởi tạo game

Chức năng: Thiết lập trạng thái ban đầu của game.

Tạo màn hình game (800x600) sử dụng Pygame.

Tải tài nguyên: hình ảnh (background.png, ship.png, asteroid.png, explosion.png) và âm thanh (background.wav, boom.wav).

Tạo đối tượng Ship tại vị trí (400, 300).

Tạo 5 đối tượng Asteroid với vị trí và tốc độ ngẫu nhiên.

Tạo các nhóm sprite: ship_group, missile_group, asteroid_group, explosion_group.

Đặt score = 0 và game_over = False.

Đầu vào: Không có.

Đầu ra: Dữ liệu khởi tạo (màn hình, sprite groups, Ship, Asteroids, score) → Chuyển đến Nhận đầu vào người chơi.

c) Nhận đầu vào người chơi

Chức năng: Nhận và xử lý đầu vào từ người chơi, cập nhật trạng thái Ship.

Nhận phím: LEFT (xoay trái), RIGHT (xoay phải), UP (di chuyển lên), DOWN (di chuyển lùi), SPACE (bắn tên lửa).

Nhận sự kiện: QUIT (thoát), R (khởi động lại nếu game over).

Cập nhật Ship: Xoay góc (angle), di chuyển (velocity), giới hạn trong màn hình (rect.clamp_ip).

Đầu vào: Sự kiện phím và hệ thống từ Pygame (pygame.event.get(), pygame.key.get_pressed()).

Đầu ra: Trạng thái cập nhật của Ship, chuyển đến Thoát game?.

d) Thoát game

Chức năng: Kiểm tra xem người chơi có muốn thoát game không (sự kiện QUIT).

Yes (Có): Chuyển đến Kết thúc vì người chơi chọn thoát, game dừng hoàn toàn (pygame.quit()).

No (Không): Chuyển đến Game over tiếp tục kiểm tra trạng thái game.

Đầu vào: Sự kiện QUIT.

Đầu ra: Quyết định thoát (Yes/No) → Kết thúc hoặc Game over

e) Game over

Chức năng: Kiểm tra xem game đã kết thúc chưa (dựa trên game_over).

Yes (Có): Chuyển đến Nhấn phím R vì game kết thúc do va chạm, chờ khởi động lại.

No (Không): Chuyển đến Cập nhật trạng thái thiên thạch, Missile Tiếp tục game, cập nhật các đối tượng.

Đầu vào: Giá trị game_over (True khi tàu va chạm thiên thạch).

Đầu ra: Quyết định tiếp tục hoặc kết thúc → Nhấn phím R? hoặc Cập nhật trạng thái thiên thạch, Missile.

Giải thích Yes/No:

f)Nhấn phím R

Chức năng: Khi game over, kiểm tra xem người chơi có nhấn R để khởi động lại không.

Yes (Có): Khởi động lại game bằng cách gọi reset_game().

No (Không): Khởi động lại game bằng cách gọi reset_game()..

Đầu vào: Sự kiện phím R.

Đầu ra: Quyết định khởi động lại → Khởi tạo game hoặc Nhận đầu vào người chơi.

g) Cập nhật trạng thái thiên thạch, Missile

Chức năng: Cập nhật và xử lý va chạm các đối tượng.

Thiên thạch (Asteroid): Di chuyển xuống (rect.y += speed), đặt lại nếu ra ngoài.

Tên lửa (Missile): Di chuyển (velocity), xóa nếu ra ngoài.

Hiệu ứng nổ (Explosion): Giảm timer, xóa nếu hết.

Kiểm tra va chạm: Missile vs Asteroid (xóa cả hai, tạo Explosion, tạo Asteroid mới).

Đầu vào: Vị trí và trạng thái các sprite.

Đầu ra: Trạng thái cập nhật, thông tin va chạm → Chuyển đến Cập nhật điểm số.

h) Cập nhật điểm số

Chức năng: Cập nhật điểm số dựa trên va chạm.

Tăng score += 10 nếu Missile va chạm Asteroid.

Đặt game_over = True nếu Ship va chạm Asteroid (xử lý ở bước trước).

Đầu vào: Kết quả va chạm từ bước trước.

Đầu ra: Giá trị score cập nhật → Chuyển đến Vẽ game.

i)Vẽ game (Hình chữ nhật - "Vẽ game, vẽ sprites, điểm số, gameover")

Chức năng: Vẽ các thành phần lên màn hình.

Vẽ nền (background).

Vẽ các sprite (ship_group, missile_group, asteroid_group, explosion_group).

Hiển thị score.

Nếu game_over = True, vẽ thông báo "GAME OVER - Press R to Restart".

Đầu vào: Dữ liệu sprite, score, game_over.

Đầu ra: Nội dung vẽ → Chuyển đến Hiển thị màn hình game.

k) Hiển thị màn hình game (Hình chữ nhật - "Hiển thị màn hình game")

Chức năng: Cập nhật màn hình để hiển thị nội dung.

Gọi pygame.display.flip() để hiển thị mọi thứ đã vẽ.

Đầu vào: Nội dung đã vẽ.

Đầu ra: Màn hình được cập nhật → Quay lại Nhận đầu vào người chơi.

Kết thúc

Chức năng: Điểm kết thúc khi người chơi thoát (sự kiện QUIT).

Đầu vào: Quyết định thoát.

Đầu ra: Thoát game (pygame.quit()).

Mối quan hệ đầu vào/đầu ra giữa các khối

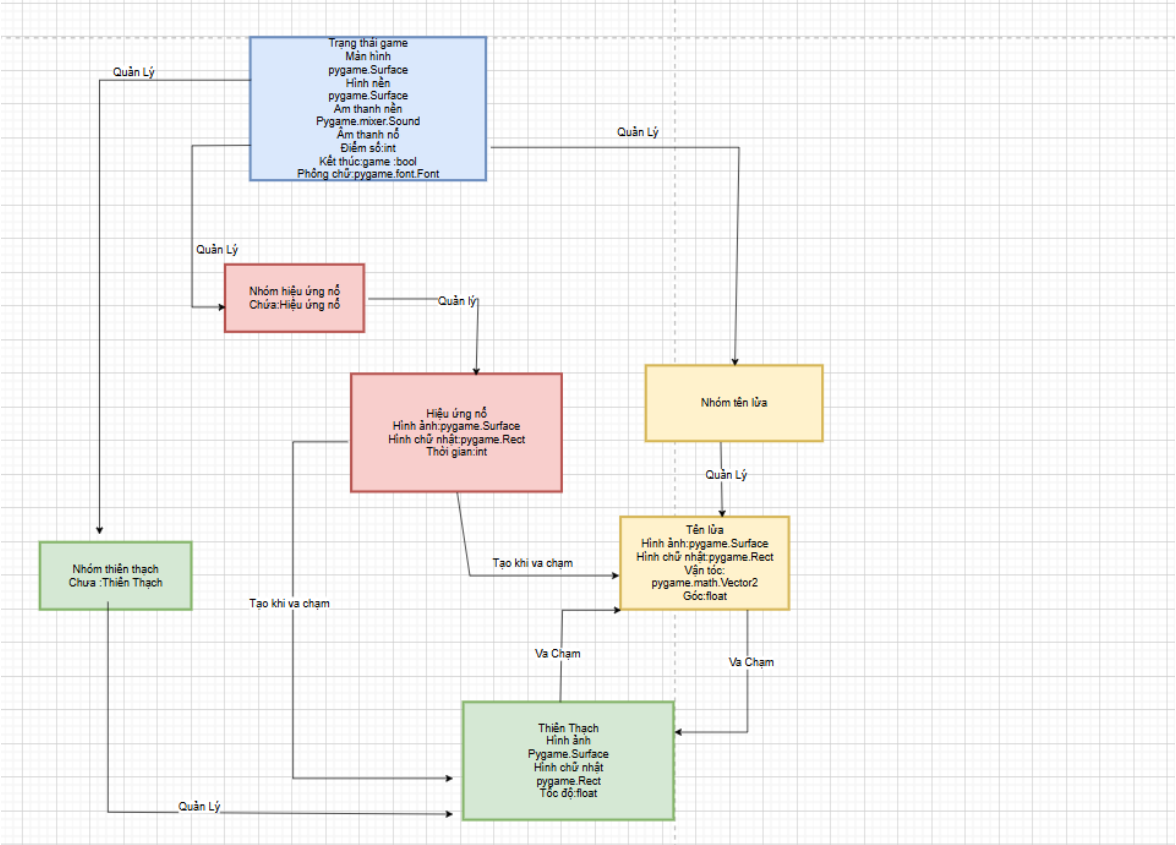
Bắt đầu → Khởi tạo game: Cung cấp trạng thái ban đầu.

Khởi tạo game → Nhận đầu vào người chơi: Truyền màn hình và sprite groups.

Nhận đầu vào người chơi → Thoát game?: Truyền sự kiện phím và trạng thái Ship.

3.3. Cấu trúc dữ liệu

- Mô tả các bảng dữ liệu, các trường thông tin lưu trữ trong dữ liệu




*Hình 3.2 Cấu trúc dữ liệu*

## *3.4. Chương trình*

Trình bày các hàm trong chương trình chính

a)Hàm draw_text

Chức năng: Vẽ văn bản lên màn hình game, dùng để hiển thị điểm số hoặc thông báo như "GAME OVER".

Tham số đầu vào:

Text: Chuỗi văn bản cần vẽ (ví dụ: "Score: 50" hoặc "GAME OVER - Press R to Restart").

Font: Đối tượng phông chữ (ở đây là pygame.font.SysFont(None, 36)), xác định kiểu chữ và kích thước (36).

Color: Màu sắc của văn bản, dạng bộ ba số RGB (ví dụ: (255, 255, 255) là màu trắng, (255, 0, 0) là đỏ).

Surface: Bề mặt để vẽ lên, thường là screen (màn hình game, kiểu pygame.Surface).

- Cách hoạt động:

Dùng font.render(text, True, color) để tạo một hình ảnh văn bản từ chuỗi text, với màu color.

Lấy vị trí của văn bản bằng text_obj.get_rect(topleft=(x, y)) để đặt văn bản ở tọa độ (x, y).

Vẽ văn bản lên surface bằng surface.blit(text_obj, rect).

- Vai trò trong hệ thống:

  Hiển thị điểm số của người chơi ở góc trên bên trái màn hình (tọa độ x=10, y=10).

  Hiển thị thông báo "GAME OVER - Press R to Restart" ở giữa màn hình khi game kết thúc

b) Hàm reset_game

Chức năng: Đặt lại toàn bộ trạng thái game về trạng thái ban đầu, như khi mới bắt đầu chơi.

- Cách hoạt động:

  Tạo một đối tượng Ship mới (tàu của người chơi) bằng ship = Ship().

- Khởi tạo các nhóm sprite:

  Ship_group: Nhóm chứa Ship (chỉ có 1 tàu).

  missile_group: Nhóm chứa các Missile (ban đầu rỗng, sẽ thêm khi bắn).

  Asteroid_group: Nhóm chứa các Asteroid (thêm 5 thiên thạch ban đầu).

  Explosion_group: Nhóm chứa các Explosion (ban đầu rỗng, sẽ thêm khi có va chạm).

  Đặt lại biến score = 0 (điểm số về 0).

  Giá trị trả về: Không trả về giá trị (chỉ cập nhật các biến toàn cục: ship, ship_group, missile_group, asteroid_group, explosion_group, score).

- Vai trò trong hệ thống:

  Được gọi khi game khởi động lần đầu để thiết lập các đối tượng và trạng thái ban đầu.

Được gọi khi người chơi nhấn phím R để khởi động lại game sau khi game_over = True.

*Tóm tắt chương.*

*Chương này mô tả thiết kế game Asteroids sử dụng Pygame, với các module quản lý hiển thị, hình ảnh, âm thanh, sự kiện, và sprite. Quy trình game bao gồm khởi tạo, xử lý input, va chạm, cập nhật trạng thái, vẽ và hiển thị. Cấu trúc dữ liệu lưu thông tin về tàu, thiên thạch, tên lửa, vụ nổ, điểm số và trạng thái game, được quản lý hiệu quả qua các nhóm sprite.*

# CHƯƠNG 4 THỰC NGHIỆM VÀ KẾT LUẬN

## 4.1. Thực nghiệm

1.Khởi tạo Pygame và thiết lập màn hình

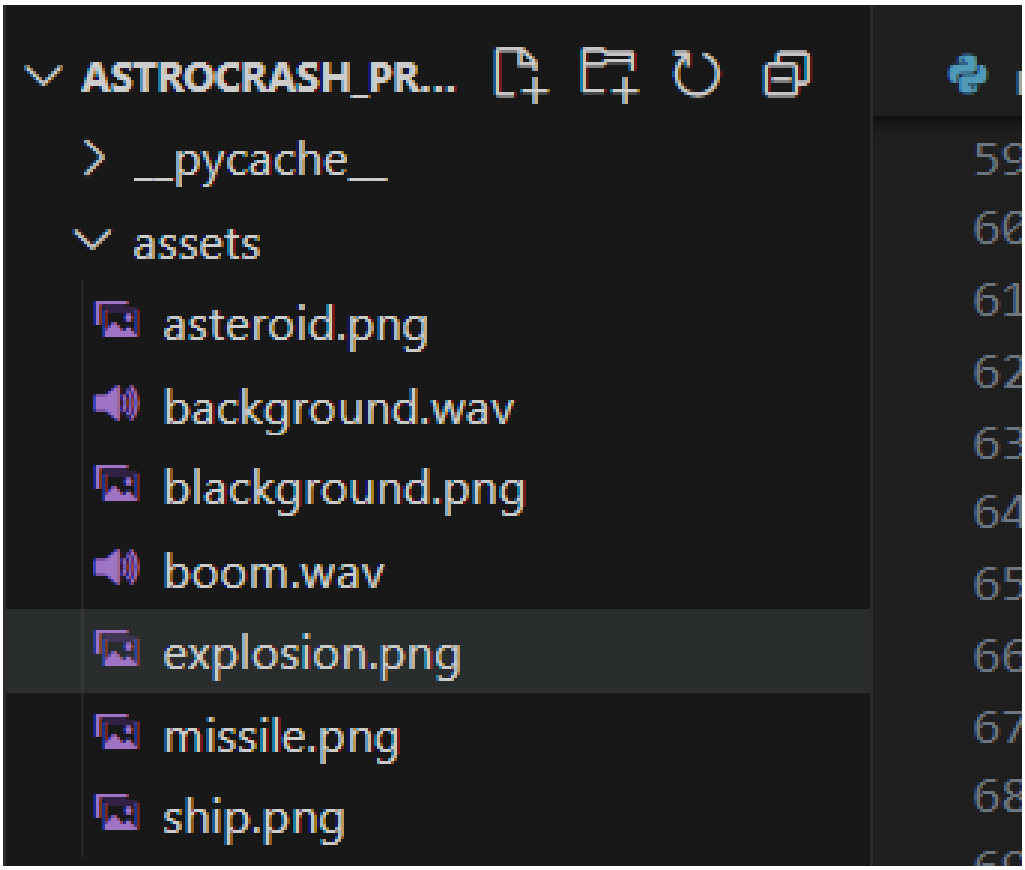
*Mục tiêu: Kiểm tra khởi tạo tài nguyên và đối tượng ban đầu*

```
def reset_game():
    global ship, ship_group, missile_group, asteroid_group, explosion_group, score
    ship = Ship()
    ship_group = pygame.sprite.Group(ship)
    missile_group = pygame.sprite.Group()
    asteroid_group = pygame.sprite.Group()
    explosion_group = pygame.sprite.Group()
    for _ in range(5):
        asteroid_group.add(Asteroid())
    score = 0

reset_game()
running = True
game_over = False
```

*Hình 4.1 Gọi reset_game() để tạo Ship, 5 Asteroid, và các nhóm sprite.*

*Kiểm tra tải hình nền, âm thanh, phông chữ, và biến score, game_over.*




*Hình 4.1.1 Kết quả mô phỏng*

Tính năng*:*

Các đối tượng được tạo đúng: 1 Ship, 5 Asteroid.

Các nhóm sprite khởi tạo đúng: ship_group (1 phần tử), asteroid_group (5 phần tử), missile_group và explosion_group (0 phần tử).

Giả sử file background.png và background.wav không tồn tại: Nền đen với 100 điểm sáng hiển thị, thông báo ">> background.wav không tồn tại." được in ra.

Giả sử file boom.wav tồn tại: boom_sound được tải, âm lượng 0.7.

Phông chữ khởi tạo thành công (pygame.font.SysFont(None, 36)).

score = 0, game_over = False.

*Đánh giá: Đạt yêu cầu, game khởi tạo đúng các đối tượng và tài nguyên ban đầu.*

2. Điều khiển tàu

Mục tiêu: Kiểm tra di chuyển và xoay tàu.

Bước thực hiện:

Nhấn LEFT (xoay trái), RIGHT (xoay phải), UP (di chuyển lên), DOWN (di chuyển xuống).

*Hình 4.1.2 : Kiểm tra di chuyển và xoay tàu*

Tính năng:

Ban đầu angle = 0. Nhấn LEFT: angle = 5. Nhấn LEFT lần nữa: angle = 10.

Nhấn RIGHT: angle = 5. Nhấn RIGHT lần nữa: angle = 0.

Nhấn UP (góc 0): Tàu di chuyển lên, velocity tăng (0, -0.2). Sau 1 khung hình, velocity *= 0.99 → velocity ≈ (0, -0.198).

Nhấn DOWN: Tàu di chuyển xuống, velocity tăng (0, 0.1). Sau 1 khung hình, velocity ≈ (0, 0.099).

Tàu di chuyển đến biên (x=0, y=0): rect.clamp_ip giữ tàu trong màn hình.

Đánh giá: Đạt yêu cầu, tàu phản hồi đúng với phím bấm, không vượt ra ngoài màn hình

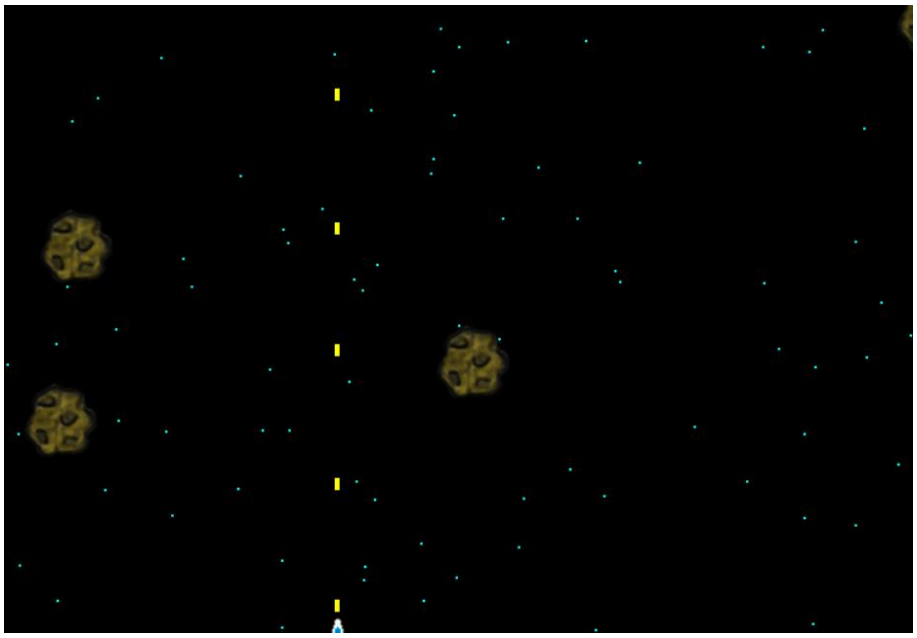
*3.* Bắn tên lửa

Mục tiêu: Kiểm tra bắn và di chuyển tên lửa.

Bước thực hiện:

Nhấn SPACE khi game_over = False.

Theo dõi tên lửa di chuyển và tự hủy.



*Hình 4.1.3 Kiểm tra bắn và di chuyển tên lửa.*

*Tính năng:*

Tàu ở (400, 300), góc angle = 0. Nhấn SPACE: Tên lửa được tạo tại (400, 300 - chiều cao tàu/2), di chuyển lên với vận tốc (0, -10).

Sau 1 khung hình (1/60 giây), vị trí tên lửa: (400, 290). Sau 30 khung hình: (400, 0).

Sau 31 khung hình, rect.bottom < 0 → tên lửa tự hủy (self.kill()).

Nhấn SPACE khi game_over = True: Không tạo tên lửa.

Đánh giá: Đạt yêu cầu, tên lửa được bắn và di chuyển đúng, tự hủy khi ra ngoài.

4. Va chạm

Mục tiêu: Kiểm tra xử lý va chạm.

Mô phỏng Missile va Asteroid.,mô phỏng Ship va Asteroid.


*Hình 4.1.4* Kiểm tra xử lý va chạm.

Tính năng:

Tên lửa (400, 100) va chạm thiên thạch (400, 100),pygame.sprite.groupcollide phát hiện va chạm.

Cả hai bị xóa (True, True).

Âm thanh nổ phát (nếu boom_sound tồn tại).

Tạo Explosion tại (400, 100).

Thêm 1 Asteroid mới,score tăng từ 0 lên 10.

Tàu (400, 300) va chạm thiên thạch (400, 300):pygame.sprite.spritecollideany phát hiện va chạm.

Âm thanh nổ phát:Tạo Explosion tại (400, 300).

game_over = True.

Đánh giá: Đạt yêu cầu, xử lý va chạm đúng, điểm số và trạng thái game cập nhật chính xác.

5. Hiển thị giao diện

Mục tiêu: Kiểm tra vẽ nền, sprite, điểm số, thông báo.


Hình 4.1.5 Giao diện game

Tính năng:

Nền vẽ tại (0, 0).

Các nhóm sprite (ship_group, missile_group,asteroid_group,explosion_group) vẽ đúng vị trí.

Điểm số "Score: 10" hiển thị tại (10, 10), màu trắng.

Khi game_over = True, thông báo "GAME OVER - Press R to Restart" hiển thị tại (250, 300), màu đỏ.

Đánh giá: Đạt yêu cầu, giao diện hiển thị đầy đủ và chính xác.

6. Test Khởi động lại game

Mục tiêu: Kiểm tra xem game có khởi động lại đúng khi nhấn R không.


*Hình 4.1.6 Khi chơi thua nhấn phím R để có thể chơi*

Tính năng:

Game_over = True. Nhấn R:

Reset_game() được gọi, các đối tượng khởi tạo lại.

Game_over = False

*4.3 Kết luận*

Sản phẩm đã làm được những gì?

Game Astrocrash đã hoàn thành các tính năng cơ bản của một game bắn thiên thạch 2D sử dụng Pygame:

Khởi tạo game:Thiết lập môi trường Pygame: Màn hình 800x600, tiêu đề "ASTROCRASH_PROJECT".

Tải tài nguyên: Hình nền (nếu không có thì tạo nền mặc định với các điểm sáng), âm thanh (nhạc nền và âm thanh nổ), phông chữ.Tạo các đối tượng ban đầu: 1 tàu (Ship), 5 thiên thạch (Asteroid), và các nhóm sprite.

Điều khiển và tương tác:Điều khiển tàu: Xoay (LEFT, RIGHT), di chuyển (UP, DOWN) mượt mà, không vượt ra ngoài màn hình.Bắn tên lửa: Nhấn SPACE để bắn, tên lửa di chuyển theo góc của tàu, tự hủy khi ra ngoài.

Xử lý va chạm:

Tên lửa va chạm thiên thạch: Xóa cả hai, tạo hiệu ứng nổ (Explosion), tăng điểm (+10), thêm thiên thạch mới.Tàu va chạm thiên thạch: Tạo hiệu ứng nổ, chuyển sang trạng thái game_over.

Hiển thị giao diện:

Vẽ nền, sprite (tàu, tên lửa, thiên thạch, hiệu ứng nổ) đúng vị trí.

Hiển thị điểm số ("Score: {score}") ở góc trên bên trái.Hiển thị thông báo "GAME OVER - Press R to Restart" khi game kết thúc.

Khởi động lại:Nhấn phím R để khởi động lại game khi game_over = True, đặt lại trạng thái ban đầu.

Hiệu suất và trải nghiệm:Game chạy ổn định ở 60 FPS.Có âm thanh (nếu file tồn tại), tăng trải nghiệm người chơi.

Học được gì

Qua quá trình phát triển và phân tích game Astrocrash, có thể rút ra những bài học sau:

Kỹ năng lập trình với Pygame:Sử dụng Pygame để tạo game 2D: Quản lý màn hình, sprite, âm thanh, và xử lý sự kiện.Quản lý nhóm sprite (pygame.sprite.Group) để xử lý nhiều đối tượng cùng lúc (tàu, tên lửa, thiên thạch). Xử lý va chạm (pygame.sprite.groupcollide, pygame.sprite.spritecollideany) để tạo tương tác giữa các đối tượng.

Tư duy thiết kế game:

Hiểu cách tổ chức mã theo mô-đun: Tách biệt các lớp (Ship, Missile, Asteroid, Explosion) để dễ quản lý.

Quản lý trạng thái game (score, game_over) và luồng điều khiển (vòng lặp game, khởi động lại).

Xử lý lỗi và tài nguyên:

Kỹ năng xử lý lỗi khi tài nguyên (hình ảnh, âm thanh) không tồn tại: Tạo nền mặc định, bỏ qua âm thanh nếu file lỗi.

Quản lý tài nguyên để tránh crash chương trình.

Kiểm thử và phân tích:

Học cách kiểm thử các tính năng (khởi tạo, điều khiển, va chạm, giao diện) để đảm bảo game hoạt động đúng.

Phân tích hiệu suất và trải nghiệm người dùng để tìm điểm cần cải thiện.

Sẽ cải tiến gì?

Dựa trên các bài test và đánh giá trước đó, tôi đề xuất các cải tiến sau để nâng cao chất lượng game:

Thêm tính năng nâng cao:

Thêm cấp độ khó: Tăng số lượng hoặc tốc độ thiên thạch theo thời gian.

Thêm vật phẩm thưởng: Ví dụ, vật phẩm tăng tốc hoặc khiên bảo vệ tàu.

Thêm nhiều loại thiên thạch: Thiên thạch lớn (chia nhỏ khi bắn), thiên thạch nhanh.

Cải thiện giao diện và trải nghiệm:

Thêm màn hình hướng dẫn hoặc menu chính: Hiển thị cách chơi trước khi bắt đầu.

Lưu điểm số cao (highscore): Ghi điểm cao nhất vào file để so sánh.

Thêm hiệu ứng hình ảnh: Ví dụ, hiệu ứng rung màn hình khi va chạm.

Tối ưu hiệu suất:Giới hạn số lượng sprite tối đa (tên lửa, thiên thạch) để tránh giảm FPS khi có quá nhiều đối tượng.Tối ưu tải tài nguyên: Đảm bảo có tài nguyên mặc định nếu file không tồn tại (hình nền, âm thanh).

Cải thiện âm thanh:Thêm nhiều âm thanh: Âm thanh khi bắn tên lửa, âm thanh khi tàu va chạm.Đảm bảo âm thanh luôn có (cung cấp file mặc định nếu người dùng không có).

Khắc phục các hạn chế:Xử lý trường hợp file tài nguyên không tồn tại tốt hơn: Thay vì chỉ in thông báo, nên có tài nguyên dự phòng.Thêm tùy chọn tạm dừng game (pause) để người chơi có thể dừng giữa chừng.