TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHIỆP

KHOA ĐIỆN TỬ

BÀI TẬP KẾT THÚC
MÔN HỌC LẬP TRÌNH PYTHON

NGÀNH: KỸ THUẬT MÁY TÍNH

HỆ : ĐẠI HỌC CHÍNH QUY

THÁI NGUYÊN - 2025

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: Phạm Mạnh Quỳnh

Lớp: K58KTP

Giáo viên GIẢNG DẠY: TS. Nguyễn Văn Huy

Link GitHub:

Thái Nguyên – 2025

TRƯỜNG ĐHKTCN
KHOA ĐIỆN TỬ

CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM
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 : Phạm Mạnh Quỳnh

Lớp: *K58KTP* *Ngành: Kỹ Thuật Máy Tính*

Giáo viên hướng dẫn: TS.Nguyễn Văn Huy

Ngày giao đề : 20/5/2025 *Ngày hoàn thành : 10/6/2025*

Tên đề tài : ***Astrocrash – Âm thanh & chuyển động***

Yêu cầu :

Xây game Astrocrash (Chapter 12) với pygame: điều khiển tàu, bắn asteroid, âm thanh.

Đầu vào – đầu ra:

- *Đầu vào: Phím mũi tên quay/move, phím cách bắn.*
- *Đầu ra: Điểm, hiệu ứng nổ, nhạc nền.*

Tính năng yêu cầu:

- *Quay sprite, di chuyển, bắn tên lửa (Missile).*
- *Collisions, di chuyển asteroid.*
- *Phát sound và music.*

Kiểm tra & kết quả mẫu:

- *Bắn trúng asteroid → nổ +10 điểm, có sound "boom".*

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

Lời nói đầu

Trong bối cảnh công nghệ ngày càng phát triển, lĩnh vực phát triển trò chơi điện tử (game) đang trở thành một trong những ngành công nghiệp hấp dẫn, mang tính ứng dụng cao và đầy sáng tạo. Việc tiếp cận và xây dựng các trò chơi 2D đơn giản là một bước đi thiết thực giúp sinh viên công nghệ thông tin rèn luyện kỹ năng lập trình, tư duy logic, và khả năng thiết kế giao diện tương tác với người dùng.

Bài tập lớn với chủ đề **"Astrocrash – Âm thanh & chuyển động"** được thực hiện nhằm mục tiêu áp dụng kiến thức đã học về lập trình Python và thư viện **Pygame** để xây dựng một trò chơi mô phỏng bối cảnh không gian. Trong trò chơi này, người chơi điều khiển một phi thuyền có thể quay, di chuyển và bắn tên lửa để tiêu diệt các thiên thạch. Khi bắn trúng, hệ thống sẽ xử lý va chạm, phát hiệu ứng nổ kèm âm thanh, đồng thời cộng điểm cho người chơi. Trò chơi còn tích hợp nhạc nền để tăng tính hấp dẫn và sinh động.

Đây là một bài tập mang tính ứng dụng cao, giúp sinh viên hình thành tư duy thiết kế phần mềm theo hướng trực quan và thực tế hơn. Đồng thời, bài tập cũng tạo tiền đề cho những dự án game phức tạp hơn trong tương lai, phục vụ cho học tập, nghiên cứu hoặc phát triển sản phẩm cá nhân.

Chương 1. Giới thiệu đề bài

1.1. Giới thiệu đề bài

Đề tài "Astrocrash – Âm thanh & chuyển động" yêu cầu xây dựng một trò chơi bắn thiên thạch bằng thư viện **Pygame**. Trong game, người chơi điều khiển phi thuyền bằng các phím mũi tên để quay, di chuyển và dùng phím cách để bắn tên lửa.

Khi tên lửa bắn trúng thiên thạch, sẽ có hiệu ứng nổ, phát âm thanh "boom", và người chơi được cộng điểm. Trò chơi tích hợp nhạc nền, hiệu ứng hình ảnh và xử lý va chạm giữa các đối tượng như phi thuyền, tên lửa và thiên thạch.

Mục tiêu chính là giúp sinh viên thực hành lập trình hướng đối tượng, xử lý chuyển động, va chạm và âm thanh trong môi trường game 2D.

1.2. Các tính năng chính của chương trình

Trò chơi Astrocrash được xây dựng với các chức năng chính sau:

- ❖ Điều khiển phi thuyền
    - Dùng phím mũi tên để xoay và di chuyển phi thuyền trong không gian.
- ❖ Bắn tên lửa (Missile)
    - Phím cách dùng để bắn tên lửa theo hướng hiện tại của phi thuyền.
- ❖ Di chuyển và tạo thiên thạch (Asteroid)
    - Thiên thạch xuất hiện ngẫu nhiên và di chuyển liên tục trên màn hình.
- ❖ Xử lý va chạm (Collision Detection)
    - Khi tên lửa va chạm với thiên thạch, thiên thạch bị phá hủy, phát nổ và cộng điểm.
- ❖ Hiệu ứng nổ (Explosion)
    - Hình ảnh nổ và âm thanh “boom” được phát khi xảy ra va chạm.
- ❖ Âm thanh và nhạc nền

- Phát nhạc nền liên tục và hiệu ứng âm thanh khi bắn hoặc trúng đích.

❖ Tính điểm

- Mỗi lần phá hủy thiên thạch, người chơi được cộng 10 điểm.

1.3. Kiến thức áp dụng

Trong quá trình xây dựng trò chơi Astrocrash – Âm thanh & chuyển động, các kiến thức chính được vận dụng bao gồm:

❖ Lập trình Python

- Sử dụng ngôn ngữ Python để viết mã nguồn cho trò chơi, bao gồm các cấu trúc điều khiển, hàm, lớp và đối tượng.

❖ Lập trình hướng đối tượng (OOP)

- Thiết kế và triển khai các lớp như Ship (phi thuyền), Asteroid (thiên thạch), Missile (tên lửa), Explosion (hiệu ứng nổ).
- Quản lý trạng thái, thuộc tính và phương thức của các đối tượng trong game.

❖ Thư viện Pygame

- Quản lý cửa sổ game, vẽ hình ảnh (sprites), xử lý sự kiện bàn phím.
- Điều khiển chuyển động, xử lý va chạm giữa các đối tượng.
- Phát âm thanh hiệu ứng và nhạc nền.

❖ Toán học cơ bản cho chuyển động và quay sprite

- Sử dụng lượng giác để tính toán góc quay, hướng di chuyển của phi thuyền và tên lửa.

❖ Xử lý sự kiện và tương tác người dùng

- Bắt và xử lý các phím bấm để điều khiển hành vi nhân vật trong game.

❖ Quản lý tài nguyên game

- Tải và sử dụng hình ảnh, âm thanh, hiệu ứng một cách hiệu quả.

Những kiến thức này giúp tạo ra một trò chơi tương tác có khả năng vận hành mượt mà, đồng thời rèn luyện kỹ năng lập trình và thiết kế phần mềm thực tế.

Chương 2. Cơ sở lý thuyết

2.1. Giới thiệu về lập trình game 2D

Game 2D là loại trò chơi được xây dựng trên mặt phẳng hai chiều, với các đối tượng được hiển thị dưới dạng hình ảnh phẳng (sprites). Các thành phần cơ bản của game 2D bao gồm: nhân vật, bối cảnh, kẻ địch, hiệu ứng, và các yếu tố tương tác như va chạm, điểm số. Lập trình game 2D tập trung vào việc quản lý đồ họa, xử lý sự kiện và logic trò chơi một cách hiệu quả.

2.2. Thư viện Pygame

Pygame là thư viện mã nguồn mở dùng để phát triển game 2D bằng Python, hỗ trợ quản lý đồ họa, âm thanh và xử lý sự kiện. Pygame xây dựng trên SDL (Simple DirectMedia Layer), cho phép tạo cửa sổ game, hiển thị hình ảnh (Surface), quản lý nhóm đối tượng (Sprite), xử lý sự kiện bàn phím, chuột, và phát âm thanh.

Các thành phần chính của Pygame gồm:

- **Surface:** vùng chứa hình ảnh hoặc đồ họa.
- **Sprite:** lớp đại diện cho các đối tượng trong game.
- **Event:** xử lý sự kiện từ bàn phím, chuột.
- **Sound:** phát âm thanh và nhạc nền.

2.3. Lập trình hướng đối tượng trong game

Lập trình hướng đối tượng (OOP) là phương pháp lập trình sử dụng các lớp và đối tượng để tổ chức mã nguồn theo cấu trúc rõ ràng, dễ bảo trì và phát triển. Trong thiết kế game, OOP giúp quản lý các thành phần như phi thuyền, thiên thạch, tên lửa bằng các lớp riêng biệt với các thuộc tính và phương thức cụ thể.

Các tính chất chính của OOP bao gồm:

- **Đóng gói (Encapsulation):** ẩn dữ liệu và phương thức bên trong lớp.
- **Kế thừa (Inheritance):** tạo lớp con từ lớp cha để mở rộng tính năng.
- **Đa hình (Polymorphism):** các đối tượng có thể xử lý các phương thức giống tên khác nhau.

2.4. Xử lý chuyển động và quay sprite

Chuyển động trong game 2D thường được điều khiển bằng tọa độ (x, y) và góc quay. Việc quay sprite sử dụng toán học lượng giác, đặc biệt là các hàm sin và cos, để tính toán hướng và vị trí mới của đối tượng khi quay và di chuyển.

Trong PyGame, các sprite có thể được xoay bằng hàm pygame.transform.rotate() để tạo hiệu ứng quay linh hoạt.

2.5. Xử lý va chạm (Collision Detection)

Va chạm là quá trình kiểm tra sự chạm trán giữa các đối tượng trong game, nhằm phát hiện các tương tác như trúng đạn, chạm vật cản. Các phương pháp phổ biến gồm:

- **Collision Rect (Bounding box):** dùng hình chữ nhật bao quanh sprite để kiểm tra va chạm nhanh.
- **Pixel-perfect collision:** kiểm tra chính xác điểm ảnh để va chạm chi tiết hơn.

Pygame hỗ trợ hàm spritecollide() và colliderect() để dễ dàng xử lý va chạm giữa các sprite.

2.6. Âm thanh và nhạc nền trong game

Âm thanh đóng vai trò quan trọng trong việc tăng trải nghiệm người chơi. Pygame hỗ trợ nhiều định dạng âm thanh như WAV, OGG. Thư viện cung cấp các lớp để phát hiệu ứng âm thanh (Sound) và nhạc nền (Music).

- **Sound:** dùng để phát các hiệu ứng nhỏ như tiếng bắn, nổ.
- **Music:** phát nhạc nền với khả năng lặp lại liên tục.

Các hàm phổ biến gồm pygame.mixer.Sound.play() và pygame.mixer.music.load(), pygame.mixer.music.play().

Chương 3. Thiết kế và xây dựng chương trình

3.1. Sơ đồ khối hệ thống

Hệ thống game **Astrocrash** được thiết kế theo mô hình chia lớp rõ ràng, đảm bảo tính mô-đun và dễ bảo trì. Các khối chức năng chính gồm:

3.1.1. Khối nhập dữ liệu (Input)

- Thiết bị: Bàn phím.
- Chức năng: Nhận tín hiệu từ phím mũi tên (xoay, di chuyển tàu), phím Space (bắn tên lửa), ESC (thoát game).
- Đầu ra: Các lệnh điều khiển chuyển đến khối xử lý.

3.1.2. Khối xử lý chính (Game Engine)

- Các lớp chính: Ship, Missile, Asteroid, Explosion.
- Chức năng:
    - Tính toán di chuyển tàu và asteroid.
    - Xử lý bắn tên lửa.
    - Kiểm tra va chạm và cập nhật điểm.
    - Gọi hiệu ứng nổ khi trúng mục tiêu.
- Đầu vào: Dữ liệu từ khối nhập.
- Đầu ra: Cập nhật trạng thái đối tượng trong game.

3.1.3. Khối hiển thị (Render)

- Chức năng: Vẽ hình ảnh tàu, đạn, asteroid, hiệu ứng nổ, điểm số và các màn hình như menu, game over.
- Thư viện: Sử dụng Pygame để vẽ 2D lên screen.

3.1.4. Khối âm thanh (Audio)

- Chức năng:
    - Phát nhạc nền (music.ogg).
    - Phát âm thanh khi bắn (laser.ogg) và khi trúng địch (explosion.ogg).
- Công cụ: pygame.mixer.

3.1.5. Khối lưu trữ điểm cao (High Score)

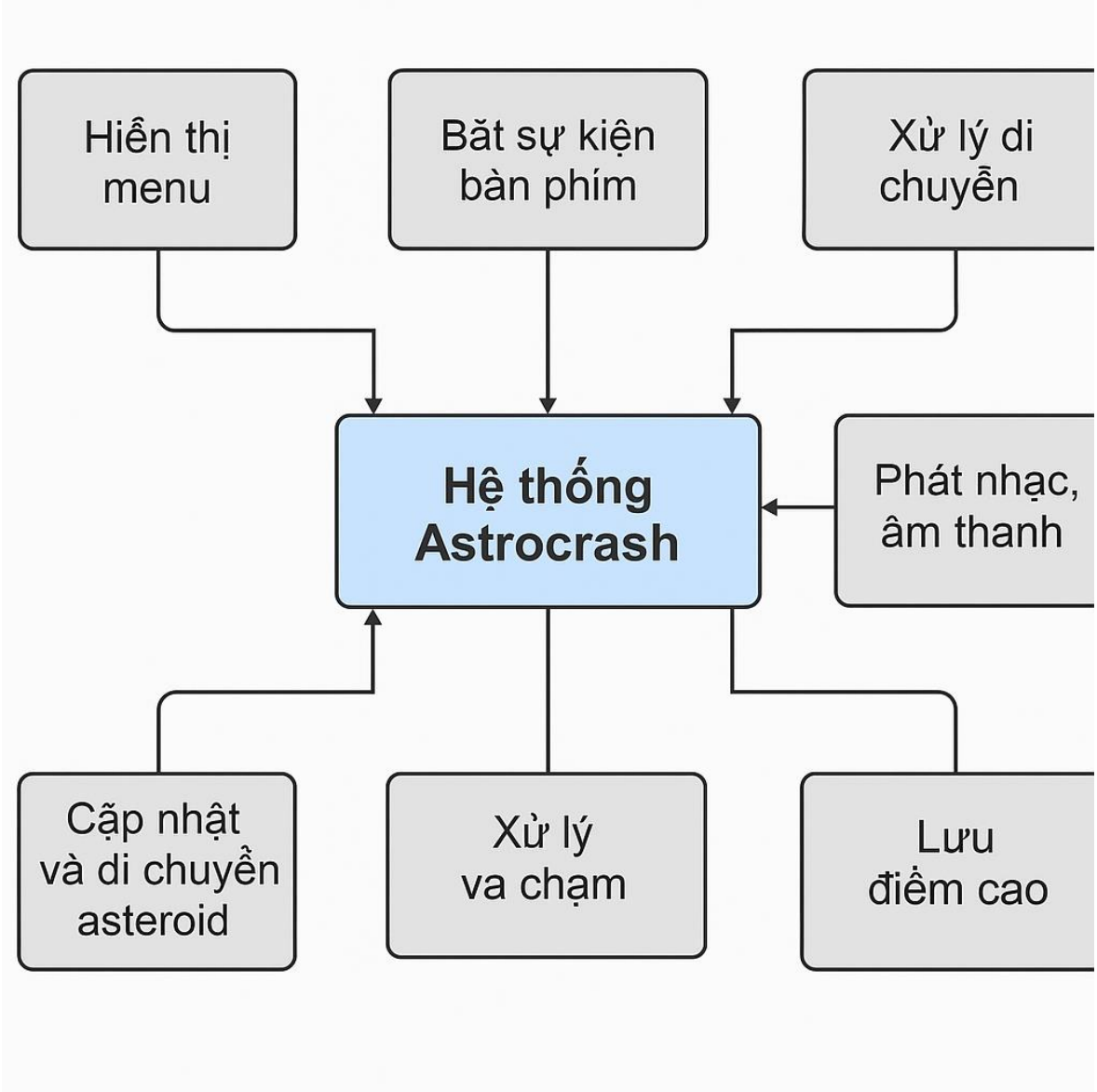
- Chức năng: Đọc và ghi điểm cao vào file highscore.txt sau mỗi lượt chơi.

- Dữ liệu: Ghi bằng định dạng văn bản (.txt).

3.1.6. Giao diện người dùng (UI/Menu)

- Chức năng: Hiển thị menu chính, điểm cao, màn hình kết thúc, và hướng dẫn điều khiển cơ bản.

Dưới đây là biểu đồ chức năng của hệ thống:



Hình 3.1. biểu dồ chức năng hệ thống

3.2. Sơ đồ khối các thuật toán chính

Các thuật toán chính được sử dụng trong chương trình đều xoay quanh thao tác với game. Dưới đây là mô tả đầu vào, xử lý, đầu ra và chức năng từng thuật toán:

3.2.1. Thuật toán điều khiển tàu (Ship Control)

- Chức năng: Điều khiển hướng quay và di chuyển của tàu vũ trụ.
- Đầu vào: Phím mũi tên trái/phải (quay), mũi tên lên (di chuyển).
- Xử lý:
  - Phím trái/phải thay đổi góc quay.
  - Phím lên di chuyển tàu theo hướng hiện tại, sử dụng cos và sin để tính vận tốc.
- Đầu ra: Cập nhật vị trí và góc của tàu trên màn hình.

3.2.2. Thuật toán bắn tên lửa (Fire Missile)

- Chức năng: Tạo và di chuyển tên lửa theo hướng tàu.
- Đầu vào: Phím cách (SPACE).
- Xử lý:
  - Tạo đối tượng tên lửa tại vị trí và hướng của tàu.
  - Di chuyển tên lửa theo hướng đó mỗi khung hình.
- Đầu ra: Tên lửa xuất hiện và bay trên màn hình.

3.2.3 Thuật toán di chuyển thiên thạch (Move Asteroids)

- **Chức năng:** Tạo và cập nhật vị trí thiên thạch rơi xuống.
- **Đầu vào:** Không yêu cầu thao tác người dùng.
- **Xử lý:**
  - Tạo thiên thạch ngẫu nhiên tại cạnh trên màn hình.
  - Di chuyển thiên thạch xuống dưới theo vận tốc xác định.
- **Đầu ra:** Hiển thị thiên thạch di chuyển trên màn hình.

3.2.4 Thuật toán kiểm tra va chạm (Collision Detection)

- **Chức năng:** Phát hiện va chạm giữa tên lửa và thiên thạch, hoặc tàu và thiên thạch.
- **Đầu vào:** Vị trí của các đối tượng trên màn hình.
- **Xử lý:**
  - Sử dụng colliderect() hoặc spritecollide() để phát hiện va chạm.

- **Đầu ra:** Nếu va chạm xảy ra, xóa đối tượng liên quan, kích hoạt hiệu ứng nổ và cập nhật điểm.

3.2.5. Thuật toán hiệu ứng nổ (Explosion Effect)

- **Chức năng:** Tạo hiệu ứng khi có va chạm.
- **Đầu vào:** Tọa độ va chạm.
- **Xử lý:**
  - Phát âm thanh nổ (explosion.ogg).
  - Hiển thị ảnh động nổ (explosion.png).
- **Đầu ra:** Hiệu ứng hình ảnh và âm thanh được phát tại vị trí va chạm.

3.2.6. Thuật toán cập nhật điểm (Score Update)

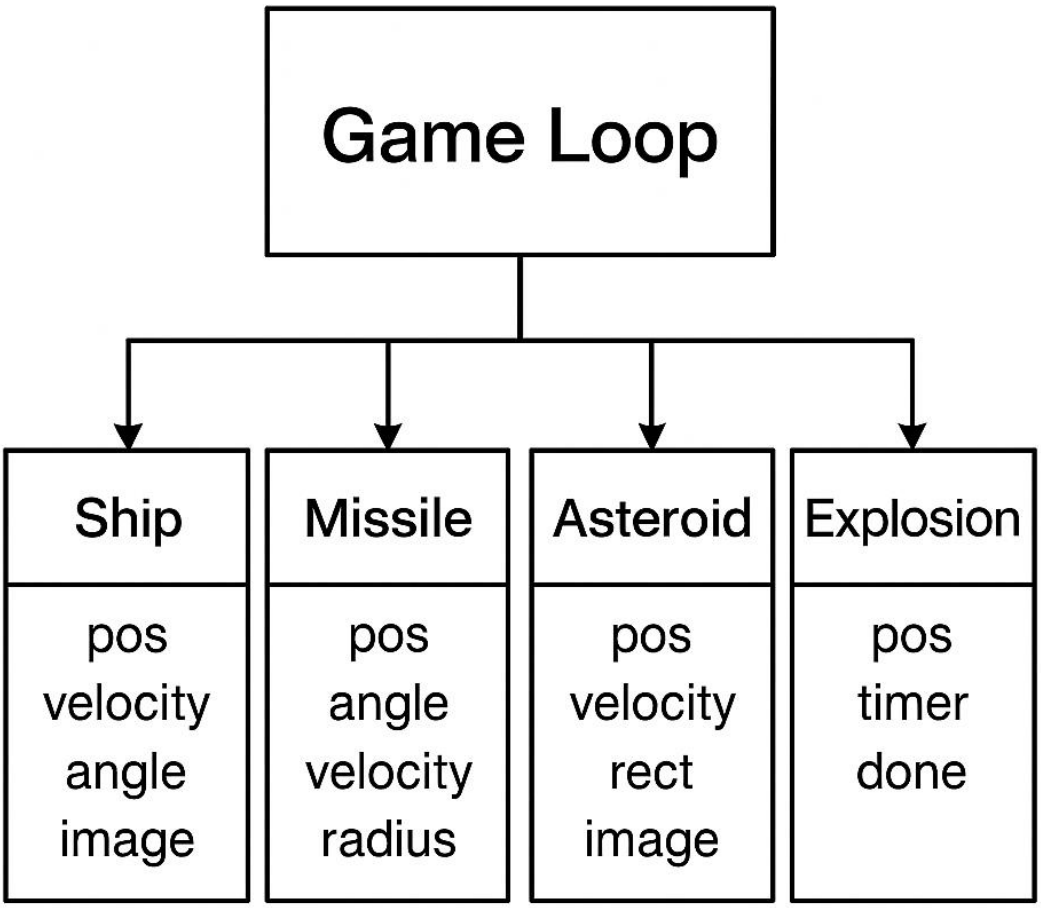
- **Chức năng:** Tăng điểm khi người chơi tiêu diệt thiên thạch.
- **Đầu vào:** Tín hiệu va chạm giữa tên lửa và thiên thạch.
- **Xử lý:** Cộng 10 điểm vào biến điểm (score += 10).
- **Đầu ra:** Giá trị điểm số mới được hiển thị trên màn hình.

3.2.7. Thuật toán cập nhật màn hình (Screen Update)

- **Chức năng:** Hiển thị tất cả đối tượng game sau mỗi lần xử lý.
- **Đầu vào:** Vị trí, trạng thái của tất cả sprite, điểm, hiệu ứng.
- **Xử lý:**
  - Dọn màn hình (screen.fill()), vẽ sprite (blit()), cập nhật điểm, hiệu ứng.
  - Gọi pygame.display.flip() để hiển thị.
- **Đầu ra:** Màn hình hiển thị mới sau mỗi khung hình.

3.3. Cấu trúc dữ liệu

Game Astrocrash sử dụng lập trình hướng đối tượng (OOP), với các lớp (class) chính để quản lý các đối tượng trong game như tàu, tên lửa, quái vật và hiệu ứng nổ. Dữ liệu được lưu trữ chủ yếu dưới dạng thuộc tính của các lớp, cùng với một số danh sách để chứa nhiều đối tượng cùng loại (ví dụ: nhiều asteroid, nhiều missile).



Hình 3.2. sơ đồ cấu trúc dữ liệu astrocrash

Mô tả các cấu trúc chính :

1. Ship

- Kiểu: Class
- Thuộc tính:
  - pos: Vị trí (Vector2)
  - velocity: Vận tốc (Vector2)
  - angle: Góc quay (float)
  - image, rect: Hình ảnh và giới hạn hiển thị

2. Missile

- Kiểu: Class
- Thuộc tính:
  - pos, angle, velocity
  - radius: Bán kính để kiểm tra va chạm

- Lưu trữ trong: Danh sách missiles

3. Asteroid

- Kiểu: Class
- Thuộc tính:
  - pos, velocity, rect, image
- Lưu trữ trong: Danh sách asteroids

4. Explosion

- Kiểu: Class
- Thuộc tính:
  - pos: Vị trí vụ nổ
  - timer: Đếm ngược thời gian nổ
  - done: Đã kết thúc hay chưa
- Lưu trữ trong: Danh sách explosion

📄 Tệp dữ liệu bên ngoài

- highscore.txt: File lưu điểm cao (kiểu int, lưu dạng chuỗi văn bản).
- assets/: Thư mục chứa hình ảnh và âm thanh (.png, .ogg).

3.4. Các hàm trong chương trình

Chương trình được thiết kế theo hướng chia module rõ ràng giữa xử lý dữ liệu và giao diện. Dưới đây là mô tả các hàm:

3.4.1. Hàm chính điều khiển trò chơi

❖ main():

- Chức năng: Vòng lặp chính điều khiển toàn bộ game.
- Đầu vào: Không có.
- Xử lý:
  - Phát nhạc nền.
  - Khởi tạo tàu, thiên thạch, tên lửa, hiệu ứng.
  - Xử lý di chuyển, va chạm, tính điểm.

Đầu ra: Vẽ toàn bộ khung hình, kết thúc game khi ESC hoặc QUIT.

3.4.2. Giao diện người dùng

❖ show_menu()

- Chức năng: Hiển thị menu khởi động.

- Đầu vào: Không.
- Xử lý:
  - Vẽ tiêu đề và lựa chọn.
  - Đợi người dùng nhấn ENTER hoặc ESC.
- Đầu ra: Bắt đầu hoặc thoát game.

❖ show_game_over(score)

- Chức năng: Hiển thị màn hình kết thúc sau khi game over.
- Đầu vào: score – điểm người chơi đạt được.
- Xử lý:
  - Hiển thị thông báo kết thúc và điểm số.
  - Đợi nhấn ENTER để quay lại menu.
- Đầu ra: Trở lại menu chính.

3.4.3. Ghi và đọc điểm cao

❖ load_highscore()

- Chức năng: Đọc điểm cao từ file highscore.txt.
- Đầu vào: Không.
- Xử lý: Mở file, đọc số điểm.
- Đầu ra: Trả về điểm cao đã lưu.

❖ save_highscore(score)

- Chức năng: Ghi điểm cao nếu điểm mới lớn hơn.
- Đầu vào: score – điểm hiện tại.
- Xử lý: So sánh và ghi vào file nếu cần.
- Đầu ra: Cập nhật file điểm cao.

3.4.4. Lớp đối tượng (class)

❖ Ship

- Hàm tạo: Khởi tạo vị trí, vận tốc, ảnh tàu.
- rotate(direction): Quay tàu sang trái/phải.
- move(): Di chuyển tàu theo hướng đang quay.
- draw(screen): Vẽ tàu lên màn hình.

❖ Missile

- Hàm tạo: Tạo tên lửa tại vị trí và góc tàu.
- update(): Di chuyển tên lửa.
- off_screen(): Kiểm tra tên lửa có ra khỏi màn hình không.
- draw(screen): Vẽ tên lửa.
- get_rect(): Lấy vùng va chạm để kiểm tra.

❖ Asteroid

- Hàm tạo: Tạo thiên thạch ngẫu nhiên.
- update(): Di chuyển thiên thạch và vòng lại màn hình.
- draw(screen): Vẽ thiên thạch.

❖ Explosion

- Hàm tạo: Khởi tạo hiệu ứng nổ.
- update(): Giảm thời gian sống.
- draw(screen): Vẽ hình ảnh nổ tại vị trí.

Chương 4. Thực nghiệm và kết luận

4.1. Giới thiệu về Python

Python là một ngôn ngữ lập trình bậc cao, thông dịch, được phát triển bởi Guido van Rossum và phát hành lần đầu tiên vào năm 1991. Python nổi bật nhờ cú pháp rõ ràng, dễ đọc, dễ viết và có cộng đồng phát triển rất lớn. Đây là một trong những ngôn ngữ phổ biến nhất hiện nay trong nhiều lĩnh vực như: phát triển phần mềm, trí tuệ nhân tạo, xử lý dữ liệu, phát triển web, tự động hoá và cả lập trình ứng dụng giao diện người dùng (GUI). Một số đặc điểm chính của Python:

- ❖ Dễ học và dễ đọc: Cú pháp đơn giản, gần giống ngôn ngữ tự nhiên giúp người học nhanh chóng tiếp cận.
- ❖ Hỗ trợ lập trình hướng đối tượng: Cho phép tổ chức mã nguồn theo hướng module, class và tái sử dụng mã.
- ❖ Thư viện phong phú: Có sẵn nhiều thư viện như json, tkinter, os, re, pandas… giúp lập trình viên phát triển nhanh các ứng dụng.
- ❖ Tương thích đa nền tảng: Chạy được trên Windows, Linux, macOS mà không cần chỉnh sửa mã nguồn.

Python được chọn làm ngôn ngữ phát triển cho game Astrocrash nhờ vào các ưu điểm nổi bật sau:

1. Dễ học, dễ đọc:
   Cú pháp Python đơn giản, rõ ràng giúp sinh viên nhanh chóng tiếp cận và triển khai logic game.

2. Thư viện Pygame mạnh mẽ:
   Pygame hỗ trợ đầy đủ các tính năng thiết yếu cho phát triển game 2D như:

   - Vẽ hình ảnh, xử lý âm thanh, nhạc nền.
   - Xử lý va chạm, bàn phím, sprite animation.

3. Phát triển nhanh:
   Python cho phép phát triển game mẫu và nguyên mẫu (prototype) một cách nhanh chóng để kiểm thử ý tưởng.

4. Cộng đồng hỗ trợ lớn:
   Có nhiều tài liệu, ví dụ, hướng dẫn giúp việc học và phát triển game dễ dàng hơn.

5. Dễ tích hợp và triển khai:
   Game viết bằng Python có thể đóng gói bằng PyInstaller để chạy trên Windows, thuận tiện khi nộp bài hoặc chia sẻ.

4.2. Thực nghiệm

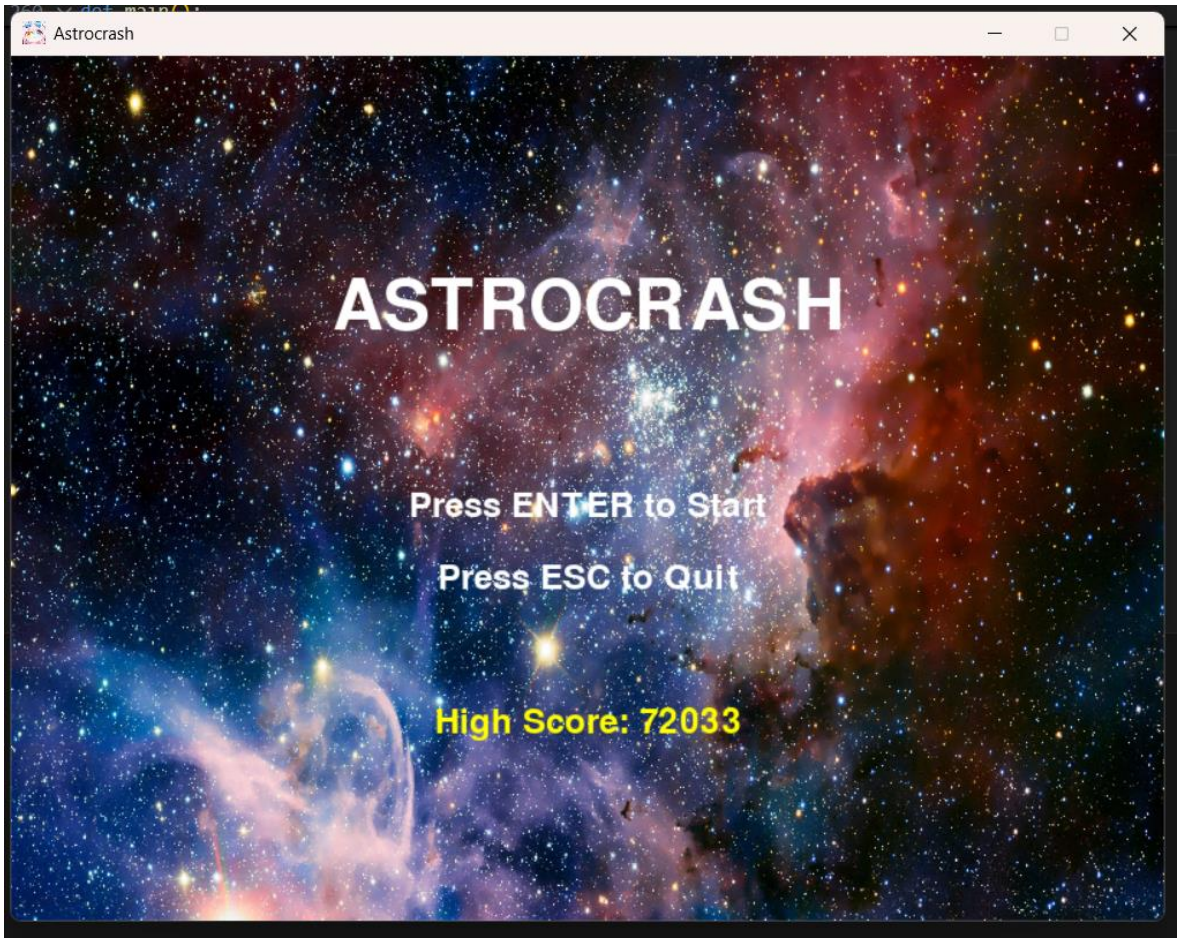
Sau khi hoàn thiện chương trình, nhóm đã tiến hành chạy thử và kiểm tra lần lượt các chức năng chính của ứng dụng quản lý danh bạ. Các kết quả thực nghiệm được ghi lại như sau:

4.2.1. Menu game:

Sau khi chạy chương trình, sẽ có 1 menu game đơn giản hiện ra:

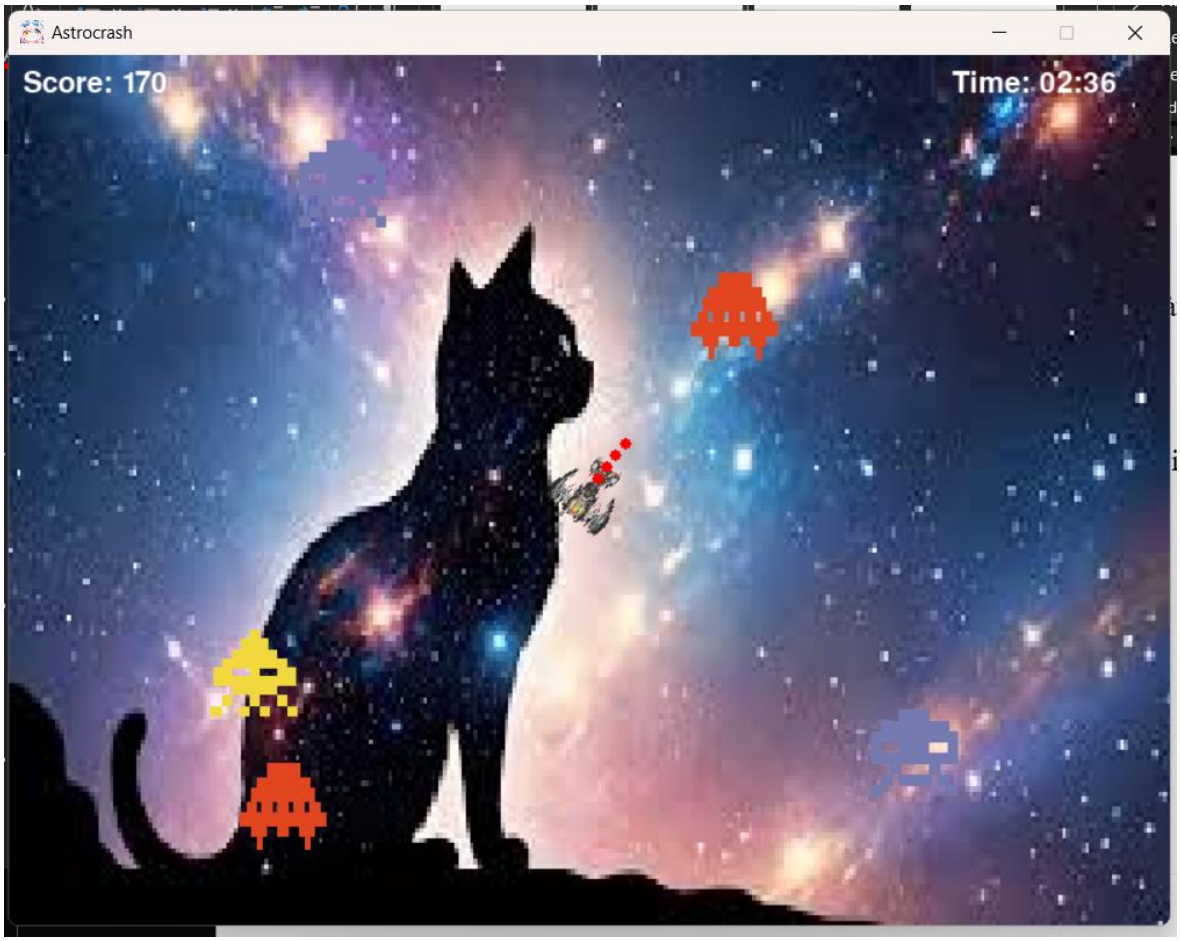
Kết quả mong đợi: Menu hiển thị, không lỗi phông, các tương tác không bị lỗi.

Kết quả thực tế: Đúng như mong đợi, menu đã hiển thị ngay, không có lỗi.



4.2.2. thao tác trong khi chơi

Mô tả: khi thao tác bằng các nút mũi tên hoặc nút Space, tàu sẽ quay theo hướng tương ứng hoặc bắn ra tên lửa.

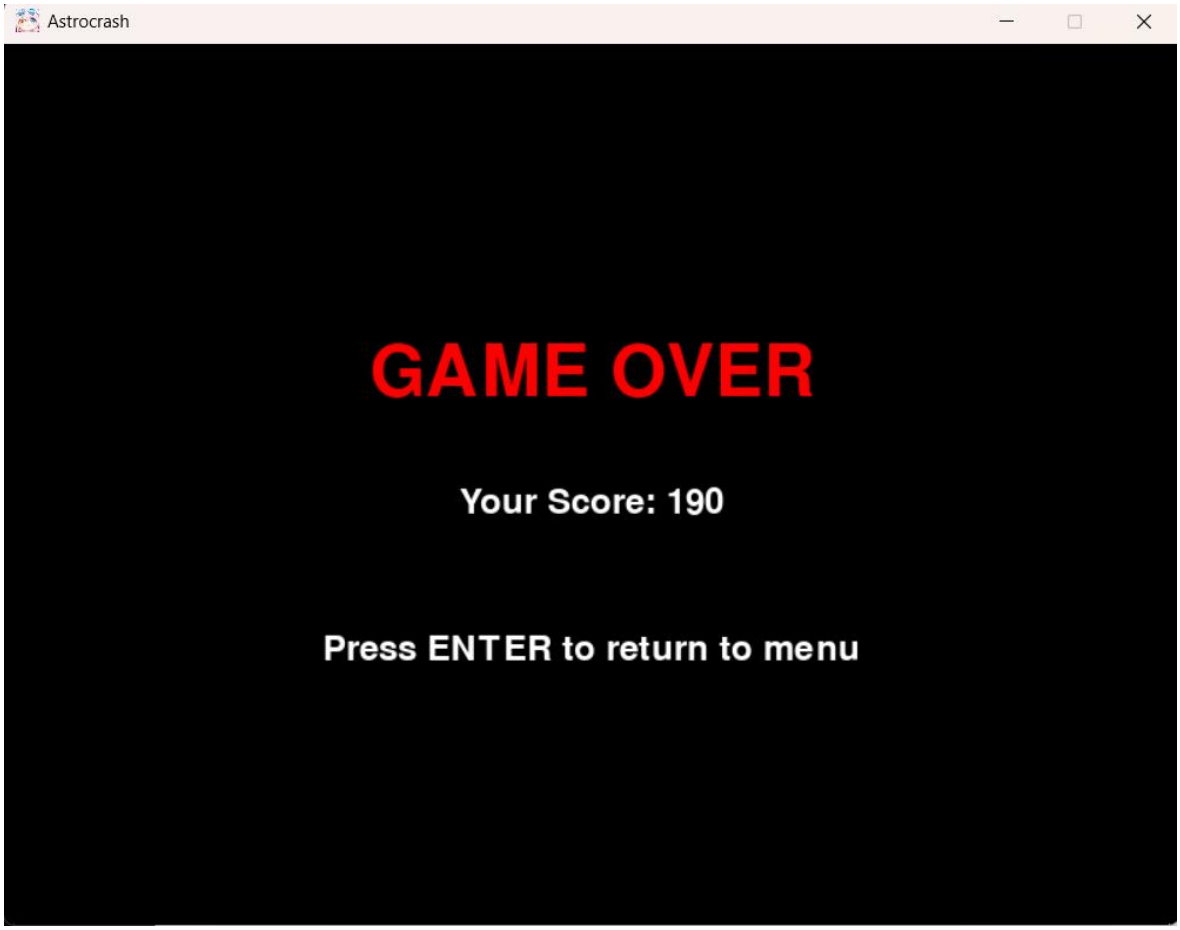


Kết quả mong đợi: tàu thực hiện đúng thao tác, không bị lỗi.

Kết quả thực tế: Đúng như kết quả mong đợi.

4.2.3. Hiển thị kết quả sau khi game kết thúc

Mô tả: Sau khi game kết thúc, màn hình tổng kết hiện ra, thông báo trò chơi kết thúc.



Kết quả mong đợi: màn hình hiển thị không lỗi, hiện rõ ràng, thông báo điểm sau khi chơi.

Kết quả thực tế: Giống như mong đợi.

4.2.4. Mã nguồn

```
import pygame
import math
import random
import os
import sys

pygame.init()

# ======================
# Thiết lập màn hình
# ======================
WIDTH, HEIGHT = 800, 600
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Astrocrash")
clock = pygame.time.Clock()
FPS = 60
```

```
# ======================
# Đường dẫn tới thư mục assets
# ======================
BASE_PATH = os.path.dirname(__file__)
ASSETS_DIR = os.path.join(BASE_PATH, "assets")

def resource_path(relative_path):
    """Hỗ trợ khi đóng gói PyInstaller"""
    try:
        base_path = sys._MEIPASS
    except Exception:
        base_path = BASE_PATH
    return os.path.join(base_path, "assets", relative_path)

# ======================
# Thiết lập icon cửa sổ
# ======================
icon = pygame.image.load(resource_path("icon.png"))
pygame.display.set_icon(icon)

# ======================
# Load âm thanh
# ======================
laser_sound = pygame.mixer.Sound(resource_path("laser.ogg"))          # Tiếng bắn
tên lửa
boom_sound = pygame.mixer.Sound(resource_path("explosion.ogg"))      # Tiếng
nổ khi trúng asteroid
pygame.mixer.music.load(resource_path("music.ogg"))                  # Nhạc nền
pygame.mixer.music.set_volume(0.5)

# ======================
# Load ảnh cho game
# ======================
ship_img =
pygame.image.load(resource_path("Character_Spaceship.png")).convert_alpha()
ship_img = pygame.transform.scale(ship_img, (50, 50))  # Đổi kích thước tàu

asteroid_imgs = []
for i in range(1, 5):
    img = pygame.image.load(resource_path(f"alien_{i}.png")).convert_alpha()
    img = pygame.transform.scale(img, (60, 60))
    asteroid_imgs.append(img)

explosion_img =
pygame.image.load(resource_path("explosion.png")).convert_alpha()
explosion_img = pygame.transform.scale(explosion_img, (60, 60))
```

```
# ======================
# Load ảnh nền
# ======================
bg_menu = pygame.image.load(resource_path("nen_menu.png")).convert()
bg_menu = pygame.transform.scale(bg_menu, (WIDTH, HEIGHT))

bg_game = pygame.image.load(resource_path("nen_choi.png")).convert()
bg_game = pygame.transform.scale(bg_game, (WIDTH, HEIGHT))

# ======================
# Lớp Ship - Tàu người chơi
# ======================
class Ship:
    def __init__(self):
        self.image_orig = ship_img  # Ảnh gốc (chưa xoay)
        self.image = self.image_orig
        self.rect = self.image.get_rect(center=(WIDTH//2, HEIGHT//2))
        self.pos = pygame.Vector2(self.rect.center)  # Vị trí trung tâm
        self.angle = 0
        self.rotation_speed = 3
        self.acceleration = 0.2
        self.velocity = pygame.Vector2(0, 0)  # Vận tốc hiện tại

    def rotate(self, direction):
        self.angle += self.rotation_speed * direction
        self.image = pygame.transform.rotate(self.image_orig, -self.angle)
        self.rect = self.image.get_rect(center=self.rect.center)

    def move(self):
        rad_angle = math.radians(self.angle)
        direction = pygame.Vector2(math.cos(rad_angle), math.sin(rad_angle))
        self.velocity += direction * self.acceleration
        if self.velocity.length() > 5:
            self.velocity.scale_to_length(5)
        self.pos += self.velocity
        self.rect.center = self.pos

        # Giới hạn trong màn hình
        if self.rect.left < 0:
            self.pos.x = self.rect.width // 2
        if self.rect.right > WIDTH:
            self.pos.x = WIDTH - self.rect.width // 2
        if self.rect.top < 0:
            self.pos.y = self.rect.height // 2
        if self.rect.bottom > HEIGHT:
```

```
            self.pos.y = HEIGHT - self.rect.height // 2
        self.rect.center = self.pos

    def draw(self, screen):
        screen.blit(self.image, self.rect)

# ======================
# Lớp Missile - Tên lửa bắn ra
# ======================
class Missile:
    def __init__(self, pos, angle):
        self.pos = pygame.Vector2(pos)
        self.speed = 10
        self.angle = angle
        rad = math.radians(angle)
        self.velocity = pygame.Vector2(math.cos(rad), math.sin(rad)) * self.speed
        self.radius = 4

    def update(self):
        self.pos += self.velocity

    def off_screen(self):
        return self.pos.x < 0 or self.pos.x > WIDTH or self.pos.y < 0 or self.pos.y >
HEIGHT

    def draw(self, screen):
        pygame.draw.circle(screen, (255, 0, 0), (int(self.pos.x), int(self.pos.y)),
self.radius)

    def get_rect(self):
        return pygame.Rect(self.pos.x - self.radius, self.pos.y - self.radius,
self.radius*2, self.radius*2)

# ======================
# Lớp Asteroid - Quái vật
# ======================
class Asteroid:
    def __init__(self):
        self.image = random.choice(asteroid_imgs)
        self.rect = self.image.get_rect()
        self.pos = pygame.Vector2(random.randrange(WIDTH),
random.randrange(HEIGHT))
        self.rect.center = self.pos
        angle = random.uniform(0, 360)
        speed = random.uniform(1, 3)
        rad = math.radians(angle)
```

```
        self.velocity = pygame.Vector2(math.cos(rad), math.sin(rad)) * speed

    def update(self):
        self.pos += self.velocity
        if self.pos.x < 0:
            self.pos.x = WIDTH
        elif self.pos.x > WIDTH:
            self.pos.x = 0
        if self.pos.y < 0:
            self.pos.y = HEIGHT
        elif self.pos.y > HEIGHT:
            self.pos.y = 0
        self.rect.center = self.pos

    def draw(self, screen):
        screen.blit(self.image, self.rect)

# ======================
# Lớp Explosion - Hiệu ứng nổ
# ======================
class Explosion:
    def __init__(self, pos):
        self.pos = pos
        self.timer = 15
        self.done = False

    def update(self):
        self.timer -= 1
        if self.timer <= 0:
            self.done = True

    def draw(self, screen):
        rect = explosion_img.get_rect(center=self.pos)
        screen.blit(explosion_img, rect)

# ======================
# Đọc và ghi điểm cao
# ======================
def load_highscore():
    try:
        with open("highscore.txt", "r") as f:
            return int(f.read())
    except:
        return 0

def save_highscore(score):
```

```
    highscore = load_highscore()
    if score > highscore:
        with open("highscore.txt", "w") as f:
            f.write(str(score))

# ======================
# Hiển thị menu chính
# ======================
def show_menu():
    font_big = pygame.font.SysFont(None, 72)
    font_small = pygame.font.SysFont(None, 36)
    title = font_big.render("ASTROCRASH", True, (255, 255, 255))
    option1 = font_small.render("Press ENTER to Start", True, (255, 255, 255))
    option2 = font_small.render("Press ESC to Quit", True, (255, 255, 255))
    highscore = load_highscore()
    highscore_text = font_small.render(f"High Score: {highscore}", True, (255, 255,
0))

    while True:
        screen.blit(bg_menu, (0, 0))
        screen.blit(title, (WIDTH//2 - title.get_width()//2, 150))
        screen.blit(option1, (WIDTH//2 - option1.get_width()//2, 300))
        screen.blit(option2, (WIDTH//2 - option2.get_width()//2, 350))
        screen.blit(highscore_text, (WIDTH//2 - highscore_text.get_width()//2, 450))

        pygame.display.flip()
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_RETURN:
                    return
                elif event.key == pygame.K_ESCAPE:
                    pygame.quit()
                    sys.exit()
        clock.tick(FPS)

# ======================
# Game Over màn hình
# ======================
def show_game_over(score):
    font_big = pygame.font.SysFont(None, 72)
    font_small = pygame.font.SysFont(None, 36)
    over_text = font_big.render("GAME OVER", True, (255, 0, 0))
    score_text = font_small.render(f"Your Score: {score}", True, (255, 255, 255))
```

```
    prompt = font_small.render("Press ENTER to return to menu", True, (255, 255,
255))

    while True:
        screen.fill((0, 0, 0))
        screen.blit(over_text, (WIDTH//2 - over_text.get_width()//2, 200))
        screen.blit(score_text, (WIDTH//2 - score_text.get_width()//2, 300))
        screen.blit(prompt, (WIDTH//2 - prompt.get_width()//2, 400))

        pygame.display.flip()
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_RETURN:
                    return
        clock.tick(FPS)

# ======================
# Hàm chính chạy game
# ======================
def main():
    pygame.mixer.music.play(-1)

    ship = Ship()
    missiles = []
    asteroids = [Asteroid() for _ in range(5)]
    explosions = []

    score = 0
    running = True

    # ====== Thời gian game ======
    start_ticks = pygame.time.get_ticks()  # thời gian bắt đầu
    game_duration = 3 * 60  # 3 phút = 180 giây

    while running:
        clock.tick(FPS)

        # ====== Tính thời gian còn lại ======
        seconds_passed = (pygame.time.get_ticks() - start_ticks) // 1000
        time_left = max(0, game_duration - seconds_passed)

        if time_left == 0:
            save_highscore(score)
```

```
    running = False

for event in pygame.event.get():
    if event.type == pygame.QUIT:
        save_highscore(score)
        running = False
    if event.type == pygame.KEYDOWN:
        if event.key == pygame.K_ESCAPE:
            save_highscore(score)
            running = False

keys = pygame.key.get_pressed()
if keys[pygame.K_LEFT]:
    ship.rotate(-1)
if keys[pygame.K_RIGHT]:
    ship.rotate(1)
if keys[pygame.K_UP]:
    ship.move()
if keys[pygame.K_SPACE]:
    if len(missiles) < 5:
        laser_sound.play()
        missiles.append(Missile(ship.pos, ship.angle))

for missile in missiles[:]:
    missile.update()
    if missile.off_screen():
        missiles.remove(missile)

for asteroid in asteroids:
    asteroid.update()

for missile in missiles[:]:
    missile_rect = missile.get_rect()
    for asteroid in asteroids[:]:
        if missile_rect.colliderect(asteroid.rect):
            boom_sound.play()
            explosions.append(Explosion(asteroid.pos))
            score += 10
            missiles.remove(missile)
            asteroids.remove(asteroid)
            asteroids.append(Asteroid())
            break

for exp in explosions[:]:
    exp.update()
    if exp.done:
```

```
                explosions.remove(exp)

        screen.blit(bg_game, (0, 0))
        ship.draw(screen)
        for missile in missiles:
            missile.draw(screen)
        for asteroid in asteroids:
            asteroid.draw(screen)
        for exp in explosions:
            exp.draw(screen)

        # ====== Hiển thị điểm và thời gian ======
        font = pygame.font.SysFont(None, 30)
        score_text = font.render(f"Score: {score}", True, (255, 255, 255))
        screen.blit(score_text, (10, 10))

        minutes = time_left // 60
        seconds = time_left % 60
        time_text = font.render(f"Time: {minutes:02}:{seconds:02}", True, (255, 255,
255))
        screen.blit(time_text, (WIDTH - 150, 10))

        pygame.display.flip()

    show_game_over(score)

# ======================
# Chạy game
# ======================
if __name__ == "__main__":
    while True:
        show_menu()
        main()
```

4.3. Kết luận

4.3.1. Những gì sản phẩm làm được

Sản phẩm game **Astrocrash** đã hoàn thiện với các chức năng cơ bản như điều khiển tàu vũ trụ, bắn tên lửa, va chạm với thiên thạch, hiệu ứng nổ, âm thanh, nhạc nền, menu, điểm số và lưu điểm cao. Giao diện trực quan và có giới hạn thời gian chơi.

4.3.2. Những gì đã học được

Qua quá trình phát triển, em đã học được cách sử dụng thư viện **Pygame**, tổ chức code theo hướng đối tượng, xử lý sự kiện, âm thanh – hình ảnh, và cách lưu trữ dữ liệu đơn giản bằng file.

4.3.3. Các cải tiến trong tương lai

- Thêm nhiều màn chơi, độ khó tăng dần
- Bổ sung kỹ năng cho tàu như lá chắn, nâng cấp vũ khí
- Tối ưu hóa hiệu suất và hỗ trợ trên nhiều nền tảng

Tài Liệu Tham Khảo

[1]. Python Official Documentation: https://docs.python.org/

[2]. Stack Overflow – https://stackoverflow.com/

[3]. Stack Overflow – Cộng đồng hỗ trợ lập trình viên: https://stackoverflow.com/

[4]. W3Schools Python Tutorial: https://www.w3schools.com/python/