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 Khắc Linh

Lớp : K58KTP

MSSV :K225480106037

Giáo viên GIẢNG DẠY : Nguyễn Văn Huy

Link GitHub :

https://github.com/phamlinh0705/Baitap_HTPython.git

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: Phạm Khắc Linh

Lớp : K58KTP *Ngành : Kỹ Thuật Máy Tính*

Giáo viên hướng dẫn: Nguyễn Văn Huy

Ngày giao đề tài: 20/5/2025 *Ngày hoàn thành: 8/6/2025*

Tên đề tài : Xây game Astrocrash với pygame: điều khiển tàu, bắn asteroid, âm thanh.

Mục tiêu đề tài :

Đầu ra – đầu vào :

- 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.

GIÁO VIÊN HƯỚNG DẪN

(Ký và ghi rõ họ tên)

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

……

……

……

……

Xếp loại:…… Điểm :……

Thái Nguyên, ngày….tháng…..năm 2025

GIÁO VIÊN HƯỚNG DẪN

(Ký và ghi rõ họ tên)

LỜI NÓI ĐẦU

Trong thời đại công nghệ số phát triển mạnh mẽ, ngành công nghiệp game không ngừng mở rộng và đóng vai trò quan trọng trong lĩnh vực giải trí cũng như giáo dục. Việc phát triển các trò chơi không chỉ giúp rèn luyện tư duy logic, kỹ năng lập trình mà còn là cơ hội để người học tiếp cận với các công nghệ đồ họa và xử lý âm thanh cơ bản trong lập trình.

Đề tài **"Xây dựng game Astrocrash với Pygame: Điều khiển tàu, bắn asteroid, âm thanh"** được thực hiện với mục tiêu áp dụng kiến thức lập trình Python, đặc biệt là thư viện Pygame, để xây dựng một trò chơi hành động 2D đơn giản. Trong trò chơi này, người chơi sẽ điều khiển một phi thuyền di chuyển trên màn hình, né tránh và tiêu diệt các thiên thạch (asteroid) bằng cách bắn đạn. Bên cạnh yếu tố điều khiển và xử lý va chạm, trò chơi còn được tích hợp các hiệu ứng âm thanh nhằm tăng tính sinh động và hấp dẫn cho người chơi.

Việc thực hiện đề tài không chỉ giúp người học củng cố kiến thức về lập trình hướng đối tượng, xử lý sự kiện, vẽ đồ họa 2D, mà còn rèn luyện khả năng phân tích, thiết kế và triển khai một sản phẩm phần mềm hoàn chỉnh. Đây cũng là tiền đề để phát triển các trò chơi có quy mô lớn hơn trong tương lai.

Em xin chân thành cảm ơn thầy Nguyễn Văn Huy đã hướng dẫn, định hướng và tạo điều kiện thuận lợi để em hoàn thành bài tập này. Em cũng mong nhận được những ý kiến đóng góp quý báu để tiếp tục cải thiện và phát triển sản phẩm trong tương lai.

Sinh viên thực hiện

Phạm Khắc Linh

MỤC LỤC

CHƯƠNG 1 : GIỚI THIỆU ĐỀ TÀI

1.1. Đề tài

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

1.2. Các tính năng chính của trò chơi gồm có

- Điều khiển tàu vũ trụ bằng các phím mũi tên để xoay và di chuyển.
- Bắn tên lửa (missile) bằng phím cách để phá hủy thiên thạch.
- Va chạm giữa tên lửa và thiên thạch sẽ tạo hiệu ứng nổ, phát âm thanh "boom", và cộng +10 điểm cho người chơi.
- Phát nhạc nền trong quá trình chơi.

1.3. Thách thức khi thực hiện

- Xoay và di chuyển sprite theo góc quay của tàu yêu cầu xử lý toán học với lượng giác (sin, cos).
- Quản lý va chạm giữa các đối tượng di chuyển (missile, asteroid) trong môi trường 2D.
- Xử lý hiệu ứng âm thanh và nhạc nền, cần biết cách sử dụng thư viện pygame.mixer.
- Tối ưu hóa hiệu năng, đảm bảo trò chơi chạy mượt mà với nhiều đối tượng cùng xuất hiện.

1.4. Kiến thức và công nghệ áp dụng

- Ngôn ngữ Python và thư viện Pygame để xử lý đồ họa 2D, sự kiện bàn phím, và âm thanh.
- Kiến thức về lập trình hướng đối tượng để xây dựng các lớp như Ship, Asteroid, Missile, Explosion.
- Kiến thức về toán học 2D, đặc biệt là vector, góc quay, vận tốc để điều khiển chuyển động và tính toán va chạm.
- Quản lý vòng lặp game: cập nhật trạng thái game, vẽ màn hình, kiểm tra va chạm, xử lý input.

CHƯƠNG 2 : CƠ SỞ LÝ THUYẾT

2.1. Ngôn ngữ Python

2.1.1. Giới thiệu

Python là một ngôn ngữ lập trình thông dịch (interpreted), hướng đối tượng (object-oriented), và là một ngôn ngữ bậc cao (high-level) ngữ nghĩa động (dynamic semantics). Python hỗ trợ các module và gói (packages), khuyến khích chương trình module hóa và tái sử dụng mã. Trình thông dịch Python và thư viện chuẩn mở rộng có sẵn dưới dạng mã nguồn hoặc dạng nhị phân miễn phí cho tất cả các nền tảng chính và có thể được phân phối tự do.

Các đặc điểm của ngôn ngữ Python:

1. **Python** dễ dàng kết nối với các thành phần khác: Python có thể kết nối với các đối tượng COM, .NET (Ironpython, Python for .net), và CORBA, Java… Python cũng được hỗ trợ bởi Internet Communications Engine (ICE) và nhiều công nghệ kết nối khác. Có thể viết các thư viện trên C/C++ để nhúng vào Python và ngược lại.

2. **Python** là ngôn ngữ có khả năng chạy trên nhiều nền tảng. Python có cho mọi hệ điều hành: Windows, Linux/Unix, OS/2, Mac, Amiga, và những hệ điều hành khác. Thậm chí có cả những phiên bản chạy trên .NET, máy ảo Java, và điện thoại di động (Nokia Series 60). Với cùng một mã nguồn sẽ chạy giống nhau trên mọi nền tảng.

3. **Python** rất đơn giản và dễ học. Python có cộng đồng lập trình rất lớn, hệ thống thư viện chuẩn, và cả các thư viện mã nguồn mở được chia sẻ trên mạng.

4. **Python** là ngôn ngữ mã nguồn mở. Cài đặt Python dùng giấy phép nguồn mở nên được sử dụng và phân tối tự do, ngay cả trong việc thương mại. Giấy phép Python được quản lý bởi Python Software Foundation.

2.1.2. Python làm được những gì

Sau đây là những khả năng của Python, khiến nó đang dần trở thành xu hướng lập trình trên thế giới:

1. Python khi sử dụng trên máy chủ có thể tạo ra các ứng dụng nền web (web application).

2. Python có thể chạy song song cùng các phần mềm khác để dễ phân luồng công việc.

3. Python có thể kết nối dễ dàng đến cơ sở dữ liệu, hay cả việc đọc và ghi file.

4. Với Python, việc xử lý Big Data và các phép toán phức tạp trở nên dễ dàng.

5. Dễ dàng sử dụng Python để tạo ra các sản phẩm demo một cách nhanh chóng. Hơn nữa, các công ty, hoặc lập trình viên chuyên nghiệp luôn ưu tiên sử dụng Python cho việc phát triển những sản phẩm chất lượng.

2.2. Thư viện Pygame

2.2.1. Pygame là gì

Pygame là một thư viện của ngôn ngữ lập trình Python và là một tập hợp các mô-đun Python được thiết kế riêng để lập trình trò chơi. Pygame được viết bởi Pete Shinners thay thế cho chương trình PySDL sau khi quá trình phát triển dự án này bị đình trệ. Chính thức phát hành từ năm 2000, Pygame được phát hành theo phần mền miễn phí GNU Lesser Genaral Public License

Pygame có thể chạy trên nhiều nền tảng và hệ điều hành khác nhau. Với thư viện pygame trong python, các nhà phất triển có thể sử dụng công cụ và chức năng mở rộng để tạo ra các trò chơi nhập vai ấn tượng. Bởi vậy, pygame đang ngày càng phổ biến với nhà phát triển vì tính linh hoạt , dễ sử dụng.

2.2.2. Đặc điểm nổi bật của lập trình Pygame

Dưới đây là một số đặc điểm nổi bật của Pygame:

- Pygame sử dụng Simple DirectMedia Layer (SDL)**,** một thư viện phát triển đa nền tảng, giúp truy cập vào phần cứng máy tính như đồ họa, âm thanh và thiết bị đầu vào.
- Hỗ trợ xây dựng trò chơi trên nhiều nền tảng khác nhau như Windows, macOS, Linux**,** thậm chí cả thiết bị di động.
- Cho phép nhà phát triển quản lý tất cả yếu tố trong quá trình phát triển trò chơi như xuất đồ họa, xử lý sự kiện, hoạt ảnh, hiệu ứng âm thanh, phát lại nhạc.
- Cung cấp nhiều chức năng mở rộng để hỗ trợ lập trình viên tập trung phát triển trò chơi.
- API trực quan và dễ hiểu**,** giúp cả người mới và những nhà phát triển có kinh nghiệm đều dễ dàng sử dụng.
- Có nguồn tài nguyên phong phú, gồm tài liệu hướng dẫn và mã nguồn mở miễn phí.

- Tính đa phương tiện, không chỉ dùng để lập trình trò chơi mà còn có thể ứng dụng trong xử lý hình ảnh, video, mô phỏng và công cụ giáo dục.

2.3. Cấu trúc dữ liệu sử dụng

2.3.1. Lập trình hướng đối tượng (OOP)

Lập trình hướng đối tượng là phương pháp tổ chức mã nguồn bằng cách mô hình hóa các đối tượng trong thế giới thực thành các class**.** Game Astrocrash sử dụng OOP để quản lý các thực thể như tàu vũ trụ, đạn, thiên thạch, và hiệu ứng nổ.

- **Class Ship** : mô tả tàu vũ trụ. Gồm các thuộc tính như vị trí, góc xoay, hình ảnh, vận tốc. Có các phương thức để xoay trái/phải, di chuyển về phía trước, và vẽ tàu lên màn hình.
- **Class Missile** : đại diện cho đạn do tàu bắn ra, có góc bắn, vị trí, vận tốc và phương thức cập nhật, vẽ.
- **Class Asteroid**: mô phỏng các thiên thạch bay ngẫu nhiên trong không gian, có kích thước khác nhau và khả năng bị phá hủy.
- **Class Explosion** : tạo ra hiệu ứng nổ từ các hạt với vận tốc, màu sắc, thời gian sống.
- **Game**: quản lý toàn bộ trò chơi – từ khởi tạo vẽ màn hình, xử lý va chạm, cập nhật đối tượng, giao diện menu..

2.3.2. Toán học 2D trong không gian

Để mô phỏng chuyển động trong không gian hai chiều, chương trình sử dụng kiến thức toán học bao gồm:

- **Góc quay** (angle) được tính theo độ và chuyển thành radian khi dùng trong các hàm lượng giác.
- Sử dụng math.sin() và math.cos() để tính toán hướng bay của tàu và đạn:

```
self.cosine = math.cos(math.radians(self.angle + 90))

self.sine = math.sin(math.radians(self.angle + 90))
```

- Khi tàu xoay, hình ảnh tàu cũng cần xoay tương ứng bằng pygame.transform.rotate():

```
self.rotatedSurf = pygame.transform.rotate(self.img, self.angle)
```

- Các đối tượng di chuyển liên tục trên màn hình theo vector vận tốc được tính từ hướng và tốc độ.

2.3.3 Danh sách (List)

Trong game Astrocrash của tôi có sử dụng một thuật toán List để quản lý các đối tượng trong trò chơi. Dưới đây là danh sách List

- Trong lớp Game đối tượng : `self.missiles (tên lửa)`

Khi người chơi bắn, một đối tượng Missile mới được thêm vào danh sách này. Trong mỗi khung hình, tất cả các tên lửa trong danh sách này được cập nhật vị trí và vẽ. Tên lửa sẽ bị xóa khỏi danh sách khi chúng va chạm với thiên thạch.

- **self.asteroids (**trong lớp Game**)**: Chứa tất cả các đối tượng Asteroid (thiên thạch) hiện đang có mặt trong trò chơi. Khi trò chơi bắt đầu hoặc khi tất cả thiên thạch hiện có đã bị phá hủy, các thiên thạch mới được thêm vào danh sách này. Trong mỗi khung hình, tất cả các thiên thạch trong danh sách này được cập nhật vị trí và vẽ. Thiên thạch bị xóa khỏi danh sách khi chúng bị tên lửa bắn trúng.
- **self.explosions (**trong lớp Game**)**: Chứa tất cả các đối tượng Explosion (vụ nổ) hiện đang hoạt động trên màn hình. Khi một thiên thạch bị phá hủy hoặc tàu của người chơi va chạm, một đối tượng Explosion mới được thêm vào danh sách này. Trong mỗi khung hình, tất cả các vụ nổ trong danh sách này được cập nhật (các hạt của chúng di chuyển và thay đổi) và vẽ. Khi tất cả các hạt của một vụ nổ đã biến mất, vụ nổ đó sẽ bị xóa khỏi danh sách này.
- **self.menu_asteroids** (trong lớp Game)**:** Chứa các đối tượng Asteroid chỉ được sử dụng để trang trí màn hình menu chính và màn hình điểm cao, không tương tác với gameplay chính. Các thiên thạch này được tạo khi khởi tạo trò chơi và chỉ được cập nhật và vẽ khi trò chơi ở trạng thái "menu" hoặc "high_scores".
- **self.particles** (trong lớp Explosion)**:** Chứa tất cả các "hạt" riêng lẻ tạo nên một hiệu ứng vụ nổ. Mỗi phần tử trong danh sách này là một list nhỏ hơn chứa các thuộc tính của một hạt (vị trí tương đối x, y, vận tốc x, y, tuổi thọ hiện tại, tuổi thọ ban đầu, kích thước ban đầu, loại màu sắc). Khi một vụ nổ được tạo, danh sách này được điền đầy các hạt. Trong mỗi khung hình, các hạt trong danh sách này được cập nhật vị trí và vẽ lên màn hình. Khi tuổi thọ của một hạt hết, nó sẽ bị xóa khỏi danh sách này.

2.3.4. Xử lý vị trí và chuyển động

- Trong game Astrocrash của tôi, việc di chuyển các đối tượng như tàu, tên lửa, được thực hiện bằng cách tính vận tốc theo hướng xoay. Sử dụng các hàm lượng giác **math.cos()** và **math.sin()** đẻ chuyển đổi góc quay thành tọa độ vận tốc trên trục X và Y
- Ví dụ: khi tàu xoay một góc nhất định, vận tốc X được tính bằng cos(góc) nhân với lực đẩy, còn vận tốc Y là sin(góc) nhân với lực đẩy.

- Tàu và tên lửa đều có các thành phần vel_x, vel_y được cập nhật mỗi khung hình và cộng vào vị trí x, y để tạo chuyển động mượt mà. Tàu cũng có hiệu ứng trôi (inertia) bằng cách giảm dần vận tốc theo thời gian với hệ số nhỏ hơn 1 (ví dụ: vel_x *= 0.995). Khi đối tượng đi qua ranh giới màn hình, nó sẽ "quấn" lại phía đối diện, tạo hiệu ứng không gian liên tục.

2.3.5. Xử lý va chạm (collision detection)

- Khi xảy ra va chạm giữa bullet vào thiên thạch, hoặc khi Ship va chạm vào thiên thạch

Để kiểm tra va chạm, tôi sử dụng công thức khoảng cách Euclidean giữa hai điểm:

```
distance = math.sqrt((self.ship.x - asteroid.x)**2 + (self.ship.y - asteroid.y)**2)
```

Nếu khoảng cách này nhỏ hơn tổng bán kính của hai đối tượng, va chạm được xác nhận:

```
if distance < asteroid.size + self.ship.size:
#va chạm xảy ra
```

Khi phát hiện va chạm:

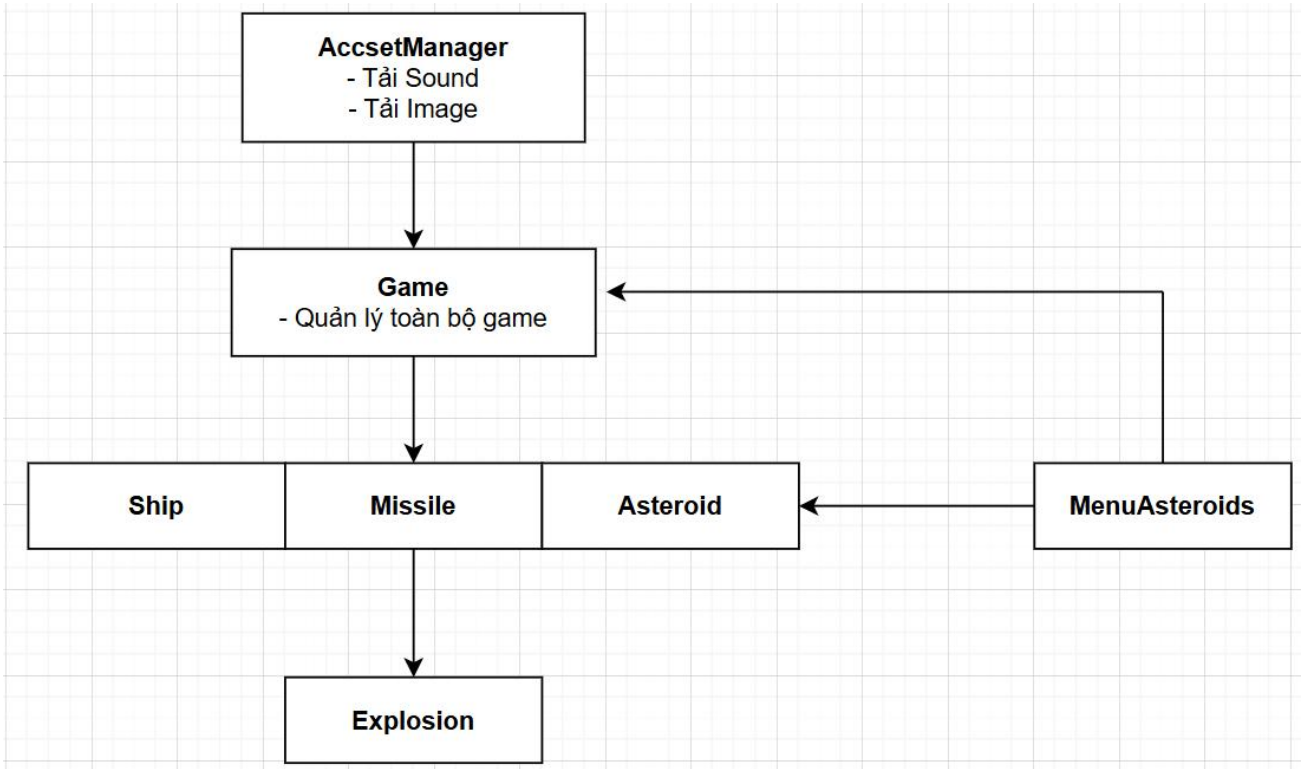
Nếu là tên lửa và thiên thạch: tên lửa bị xóa, thiên thạch phát nổ và sinh điểm.

Nếu là tàu và thiên thạch: giảm mạng, reset vị trí tàu, tạo hiệu ứng nổ.

CHƯƠNG 3 : THIẾT KẾ VÀ XÂY DỰNG CHƯƠNG TRÌNH

3.1. Sơ đồ khối hệ thống

Dưới đây là sơ đồ khối mô tả các module chính trong game :



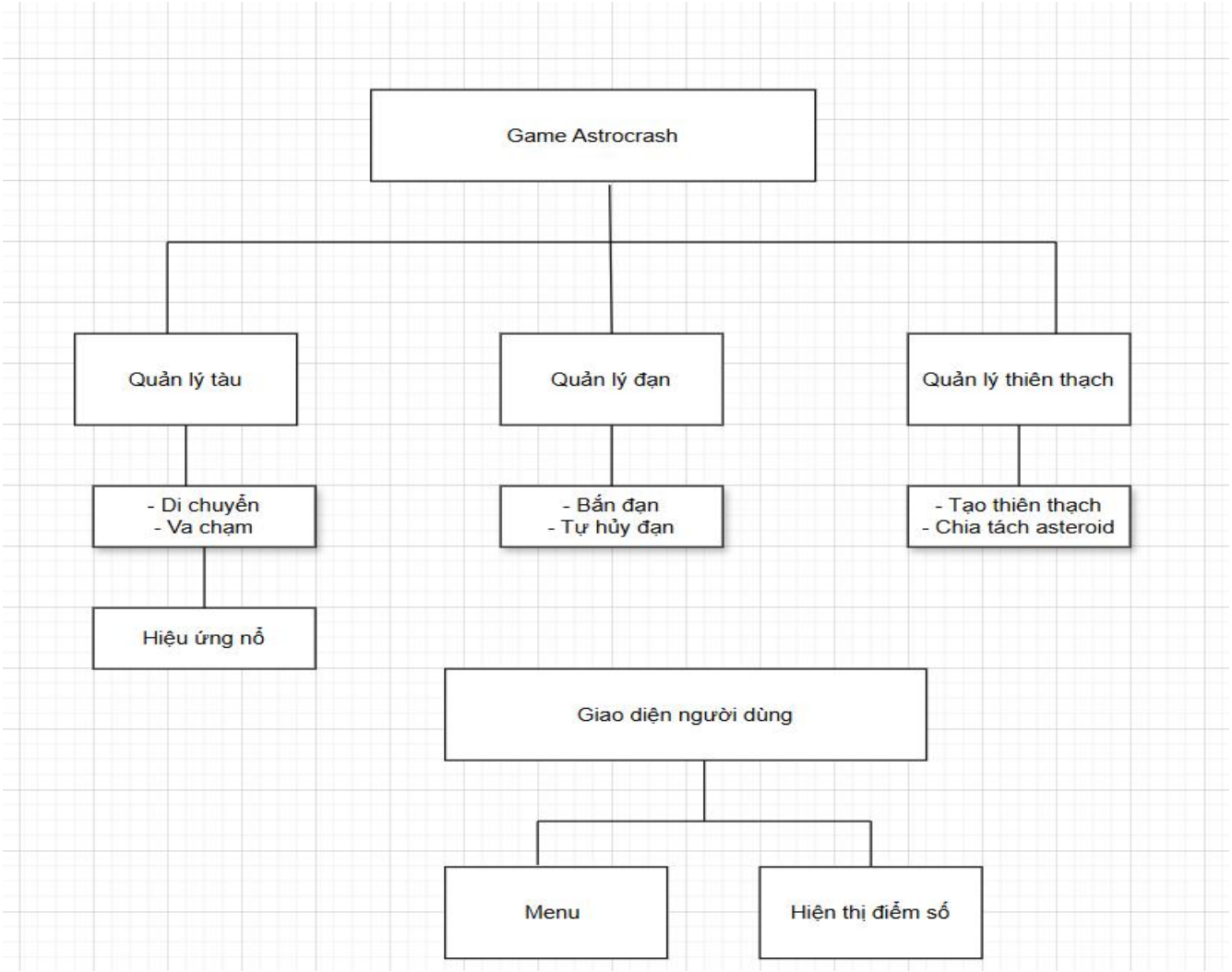
- **AssetManager**: Quản lý hình ảnh và âm thanh
- **Ship**: Điều khiển tàu vũ trụ
- **Missile**: Quản lý tên lửa do tàu bắn ra
- **Asteroid**: Sinh và cập nhật thiên thạch
- **Explosion**: Hiệu ứng nổ khi va chạm
- **Game**: Quản lý trạng thái game, xử lý va chạm, cập nhật đối tượng

Cách chức năng chính của game:

- Menu điều hướng và hiển thị high scores
- Bắt đầu và kết thúc trò chơi
- Bắn tên lửa, ghi điểm, xử lý mạng sống

3.1.1. Biểu đồ phân cấp chức năng

Biểu đồ phân cấp chức năng của game Astrocrash mô tả các chức năng như sau:



- Cấp cao nhất là toàn bộ hệ thống game.
- Các nhánh con chia thành nhóm chức năng chính như quản lý đối tượng, xử lý hiệu ứng và giao diện người dùng.
- Mỗi chức năng lại chia nhỏ ra thành các chức năng con như xử lý va chạm, hiện thị điểm…vv

3.2. Sơ đồ khối các thuật toán chính

❖ Các thuật toán chính có trong game python Astrocrash

1. Điều khiển tàu (Ship.update)
    - Đầu vào : phím điều khiển (trái, phải, lên, xuống)
    - Xử lý :
        - Cập nhật góc quay khi ấn trái – phải
        - Cập nhật vận tốc theo hướng tàu khi nhấn lên
        - Tính toán vận tốc mới bằng cos(angle) và sin(angle)
        - Áp dụng ma sát để giảm vận tốc dần đều
        - Quấn tàu khi ra ngoài màn hình
    - Đầu ra : tọa độ mới của tàu, trạng thái thrust
2. Bắn tên lửa (Missile)
    - Đầu vào : phím Space khi đang chơi game
    - Xử lý :
        - Tính vị trí ban đầu dựa trên đầu mũi tàu
        - Tính vận tốc theo góc tàu
        - Gán tuổi thọ cho tên lửa
    - Đầu ra : đối tượng tên lửa mới trong danh sách missiles
3. Va chạm (check_collisions)
    - Đầu vào : danh sách missiles, asteroids, vị trí ship
    - Xử lý :
        - Tính khoảng cách giữa các đối tượng
            - Nếu va chạm:
                - Xóa tên lửa, tạo exposion, tăng điểm
                - Tách thiên thạch lớn thành 2 thiên thạch nhỏ hơn
                - Nếu tàu bị va chạm, giảm mạng và reset vị trí tàu
    - Đầu ra : cập nhật điểm, giảm mạng và reset vị trí tàu
4. Hiệu ứng nổ (Explosion)
    - Đầu vào : vị trí va chạm

- Xử lý :
    - Tạo danh sách hạt bay ra từ tâm nổ
    - Cập nhật chuyển động, màu sắc, kích thước theo thời gian sống
- Đầu ra : các hạt nổ được vẽ trên màn hình và tự động biến mất

3.3. Cấu trúc dữ liệu

Trong Astrocrash, dữ liệu được tổ chức và lưu trữ dưới dạng danh sách (list) và từ điển (dict).

- **missiles (list)**:
    - Lưu các đối tượng tên lửa do tàu bắn ra.
    - Mỗi phần tử là một instance của lớp Missile chứa vị trí, góc bắn, vận tốc, và tuổi thọ.
- **asteroids (list)**:
    - Lưu các đối tượng thiên thạch xuất hiện trên màn hình.
    - Mỗi thiên thạch có thuộc tính vị trí, vận tốc, kích thước và loại hình ảnh.
- **explosions (list)**:
    - Lưu hiệu ứng nổ khi va chạm xảy ra.
    - Mỗi phần tử là một đối tượng Explosion gồm nhiều hạt (particles) bay ra từ tâm nổ.
- **images (dict)**:
    - Từ điển lưu trữ các hình ảnh trò chơi, ví dụ: tàu (ship), tên lửa (missile), thiên thạch (asteroid1, asteroid2, ...), nền (background).
    - Truy cập thông qua khóa tên: images['ship'], images['background'],...
- **sounds (dict)**:
    - Từ điển lưu âm thanh: bắn (shoot), nổ (boom), tăng tốc (thrust), nhạc nền (background_music).
    - Dùng trong hiệu ứng âm thanh tại các hành động tương ứng.

Cấu trúc này giúp quản lý dễ dàng số lượng lớn đối tượng sinh động trong trò chơi và tối ưu hiệu suất xử lý trong vòng lặp game.

3.4. Chương trình

Các hàm trong chương trình chính. Mỗi lớp có các hàm (method) đảm nhiệm các nhiệm vụ cụ thể:

Lớp Ship

- __init__: Khởi tạo đối tượng tàu, thiết lập vị trí, góc quay, vận tốc, trạng thái thrust.
- reset_position(): Đặt lại vị trí tàu về giữa màn hình, dùng khi mất mạng.
- update(): Cập nhật vị trí tàu, xử lý phím điều khiển, hiệu ứng tăng tốc.
- draw(screen): Vẽ hình ảnh tàu lên màn hình, xử lý xoay và hiệu ứng lửa đẩy.

Lớp Missile

- __init__: Khởi tạo tên lửa với vị trí, góc bắn và vận tốc ban đầu.
- update(): Di chuyển tên lửa mỗi khung hình, giảm tuổi thọ.
- draw(screen): Vẽ tên lửa (ảnh hoặc hình tròn đơn giản).
- is_dead(): Trả về True nếu tên lửa hết tuổi thọ.
- is_off_screen(): Kiểm tra xem tên lửa đã ra khỏi màn hình chưa.

Lớp Asteroid

- __init__: Tạo một thiên thạch với vị trí, kích thước, loại hình ảnh và vận tốc ngẫu nhiên.
- update(): Di chuyển thiên thạch và xoay theo thời gian.
- draw(screen): Vẽ thiên thạch lên màn hình.

Lớp Explosion

- __init__: Khởi tạo hiệu ứng nổ tại vị trí va chạm.
- update(): Cập nhật chuyển động và tuổi thọ của các hạt nổ.
- draw(screen): Vẽ các hạt hiệu ứng.
- is_finished(): Trả về True nếu tất cả các hạt đã biến mất.

Lớp Game

- __init__: Khởi tạo game, tải tài nguyên, tạo tàu, danh sách đạn, thiên thạch, hiệu ứng...

- spawn_asteroids(count): Sinh thiên thạch mới khi vào game hoặc sau mỗi vòng.
- check_collisions(): Kiểm tra va chạm giữa tàu, tên lửa và thiên thạch.
- handle_events(): Xử lý sự kiện bàn phím trong các trạng thái (menu, chơi game, điểm cao).
- start_game(), restart_game(): Bắt đầu hoặc khởi động lại game.
- update(): Cập nhật tất cả đối tượng trong game theo từng frame.
- draw(screen): Vẽ giao diện phù hợp theo trạng thái game.
- draw_menu(screen), draw_high_scores(screen), draw_game(screen): Vẽ từng giao diện cụ thể.
- draw_ui(screen), draw_game_over(screen): Vẽ điểm số, mạng sống, thông báo Game Over.

Nhờ tổ chức chương trình theo các lớp rõ ràng, việc mở rộng hoặc điều chỉnh chức năng được thực hiện dễ dàng và linh hoạt.

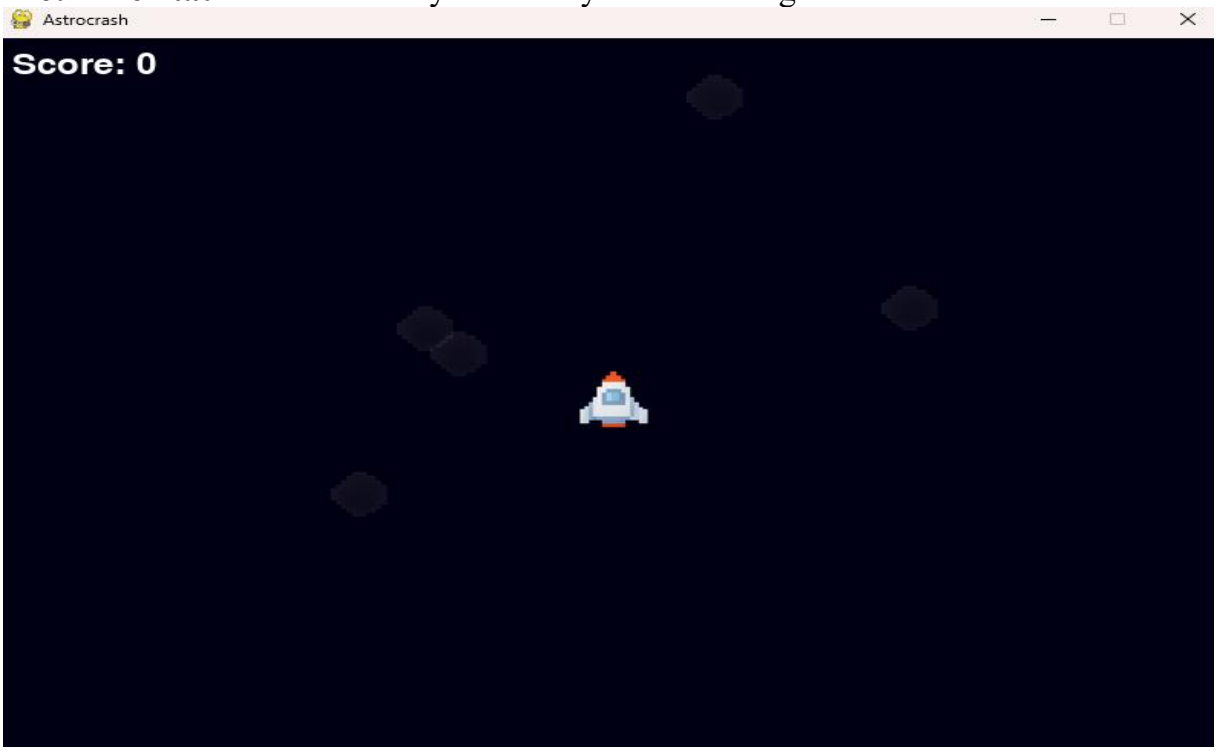
- Ship.update(): Xử lý di chuyển và hiệu ứng thrust
- Missile.update(): Di chuyển và giảm tuổi thọ tên lửa
- Asteroid.update(): Di chuyển và xoay thiên thạch
- Explosion.update(): Cập nhật vị trí hạt nổ
- Game.check_collisions(): Kiểm tra và xử lý va chạm
- Game.handle_events(): Xử lý sự kiện từ bàn phím
- Game.update(), Game.draw(): Cập nhật trạng thái và vẽ khung hình

CHƯƠNG 4: THỰC NGHIỆM VÀ KẾT LUẬN

4.1. Thực nghiệm

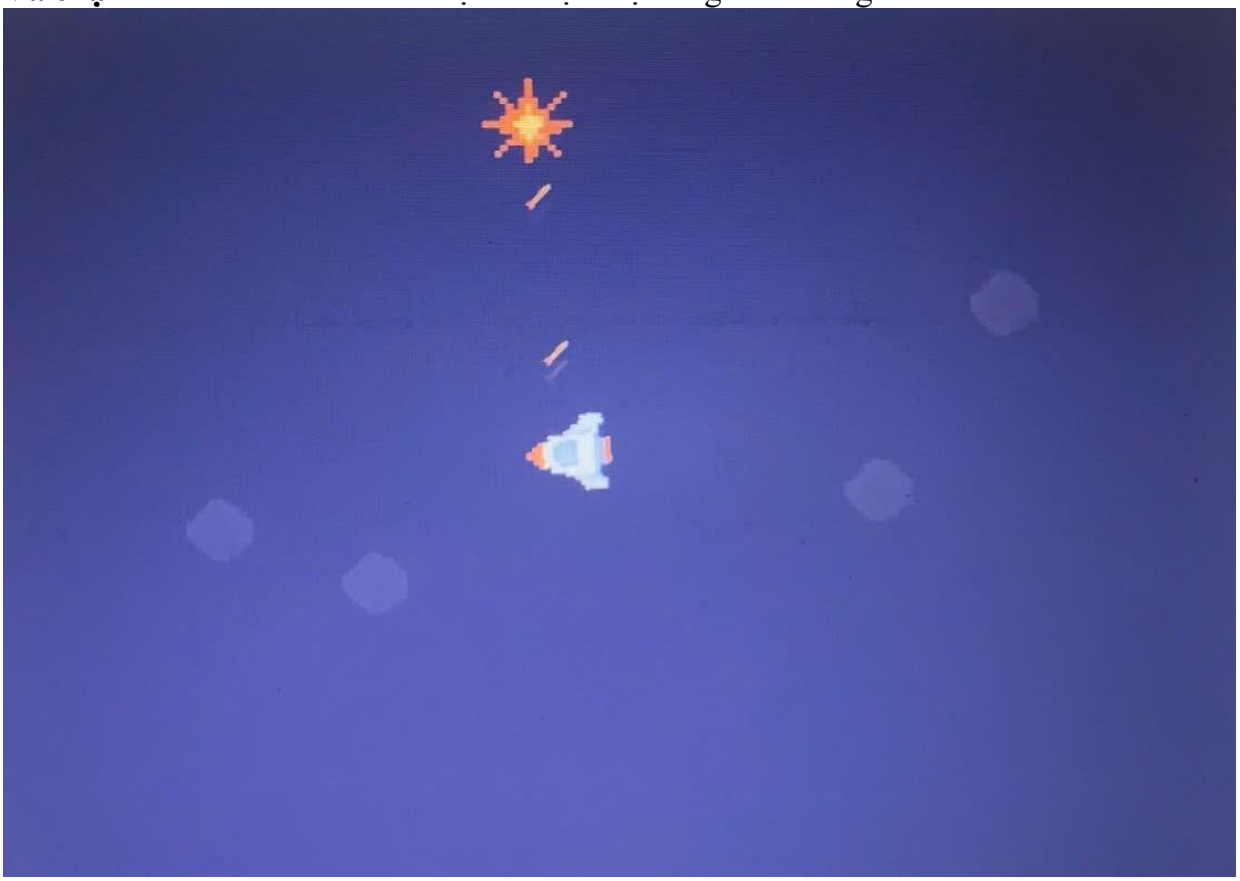
Chương trình đã được chạy thử và kiểm tra các tính năng chính:

- **Điều khiển tàu**: tàu có thể xoay và di chuyển theo hướng mũi tên.



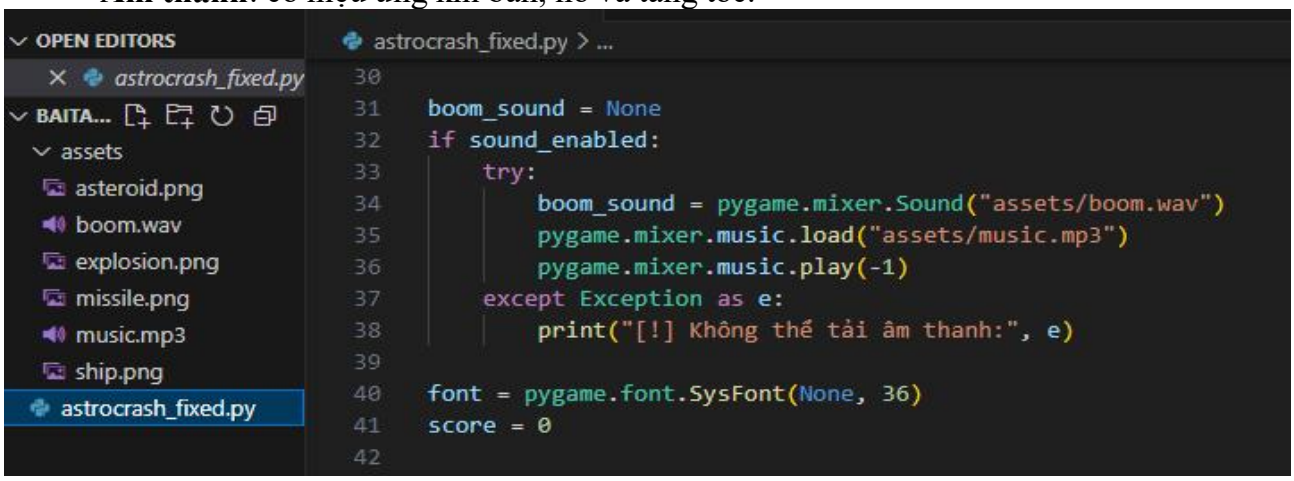
- **Bắn tên lửa**: nhấn phím Space để bắn, tên lửa di chuyển đúng hướng và tự hủy khi hết thời gian.

- **Va chạm**: tên lửa va vào thiên thạch sẽ tạo hiệu ứng nổ và tăng điểm.



- **Thiên thạch**: di chuyển ngẫu nhiên, sinh thêm khi bị phá hủy.

- **Âm thanh**: có hiệu ứng khi bắn, nổ và tăng tốc.



4.2. Kết luận

Sản phẩm đã làm được:

- Game bắn tàu vũ trụ đầy đủ chức năng: điều khiển tàu, bắn đạn, sinh và phá hủy thiên thạch.
- Hiển thị giao diện đồ họa, hiệu ứng hình ảnh sinh động và âm thanh phản hồ
- Có menu chính, màn hình điểm cao, quản lý điểm số và mạng sống.

Bài học rút ra :

- Nắm vững kỹ năng lập trình hướng đối tượng trong Python.
- Biết cách tổ chức và vận hành game loop hiệu quả.
- Biết cách tổ chức và vận hành game loop hiệu quả.
- Áp dụng thư viện Pygame để xử lý đồ họa, âm thanh, và tương tác người dùng.

Hướng phát triển:

- Bổ sung chế độ chơi nhiều người hoặc mạng LAN.
- Cải thiện giao diện và bổ sung hiệu ứng hình ảnh cao cấp hơn (explosion, ánh sáng, v.v.).
- Phát triển hệ thống nâng cấp tàu và vũ khí.
- Phát triển lên app cho các hệ điều hành.

TÀI LIỆU THAM KHẢO

1. CodeLearn. (2025, 05 29). Được truy lục từ PyThon cơ bản: https://codelearn.io/learning/python-co-ban
2. Digiuni. (2021, 8 11). Được truy lục từ 12 Game lập trình bằng Python: https://digiunivietnam.com/12-game-lap-trinh-bang-python-ban-tung-me-met/
3. GaPython. (2023, 11 2). Được truy lục từ Lập trình game bằng python và pygame : https://www.youtube.com/c/GaPython
4. Grantjenks. (2023). Được truy lục từ Free Python Games: https://grantjenks.com/docs/freegames/