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

ĐỀ TÀI: ASTROCRASH – ÂM THANH & CHUYỂN ĐỘNG

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

ĐỀ TÀI: ASTROCRASH – ÂM THANH & CHUYỂN ĐỘNG

SINH VIÊN	:	HẦU NHẬT NINH
LỚP	:	K58KTP
MSSV	:	K225480106077
GIÁO VIÊN GIẢNG DẠY	:	TS. NGUYỄN VĂN HUY
LINK YOUTOBE	:	https://youtu.be/F6Npy8ZWibg

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

Sinh viên: HẦU NHẬT NINH K225480106077

Lớp: K58KTP Ngành: Kỹ thuật máy tính

Bộ môn: Công Nghệ Thông Tin

Giáo viên hướng dẫn: TS. Nguyễn Văn Huy

Ngày giao đề: 20–05–2025 Ngày hoàn thành: 08–06–2025

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

Yêu cầu:

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”.

SINH VIÊN **GIÁO VIÊN HƯỚNG DẪN**

(Ký và ghi rõ họ tên) *(Ký và ghi rõ họ tên)*

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……

Thái Nguyên, ngày….tháng…..năm 20....

GIÁO VIÊN HƯỚNG DẪN

(Ký và ghi rõ họ tên)

MỤC LỤC

LỜI MỞ ĐẦU

Trong thời đại công nghệ phát triển mạnh mẽ hiện nay, lập trình không chỉ là công cụ để giải quyết các bài toán tính toán mà còn được ứng dụng rộng rãi trong nhiều lĩnh vực, trong đó có phát triển trò chơi. Việc xây dựng các trò chơi bằng ngôn ngữ lập trình giúp sinh viên không những củng cố kiến thức lập trình căn bản mà còn rèn luyện tư duy logic, khả năng xử lý sự kiện, đồ họa và âm thanh.

Đồ án "**Astrocrash – Âm thanh & chuyển động**" là một bài tập lớn môn Lập trình Python, nhằm giúp sinh viên tiếp cận lập trình game thông qua thư viện **Pygame**. Trong quá trình thực hiện, em đã học cách điều khiển các đối tượng (sprite), xử lý va chạm, âm thanh, nhạc nền và tạo ra một trò chơi có tính tương tác cao.

Báo cáo này trình bày quá trình xây dựng trò chơi Astrocrash từ khâu thiết kế, lập trình, đến kiểm thử và đánh giá kết quả. Em hy vọng qua bài tập này, không chỉ hiểu rõ hơn về lập trình hướng đối tượng và xử lý đồ họa trong Python mà còn rèn luyện được kỹ năng lập trình thực tế.

Em xin chân thành cảm ơn thầy Nguyễn Văn Huy đã giao đề tài ý nghĩa và tạo điều kiện để em hoàn thành bài tập này.

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

1.1. Nội dung đề tài

Đề tài yêu cầu xây dựng trò chơi **Astrocrash** bằng thư viện **Pygame**, cho phép người chơi điều khiển tàu vũ trụ để bắn phá các thiên thạch trong không gian. Game cần có các tính năng sau:

- Điều khiển tàu quay, di chuyển, tăng tốc.
- Bắn đạn theo hướng tàu bay.
- Di chuyển thiên thạch.
- Hiệu ứng nổ khi va chạm giữa đạn và thiên thạch.
- Phát nhạc nền, hiệu ứng âm thanh khi nổ.
- Tính điểm (+10 điểm mỗi lần bắn trúng).

1.2. Thách thức

- Xử lý góc quay và hướng bay của tàu trong không gian 2D.
- Phát hiện va chạm giữa đạn và thiên thạch.
- Tích hợp âm thanh và nhạc nền.
- Quản lý sprite và tài nguyên hiệu quả.

1.3. Kiến thức áp dụng

- Kỹ thuật lập trình hướng đối tượng trong Python.
- Xử lý đồ họa và sự kiện với thư viện Pygame.
- Quản lý đa luồng thời gian (frame rate, cooldown, thời gian tồn tại).
- Kỹ thuật xử lý va chạm giữa các đối tượng.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

2.1. Giới thiệu về công nghệ và thư viện sử dụng

Trong dự án game Astrocrash, công nghệ chủ đạo được sử dụng là Python cùng với thư viện Pygame — một thư viện mã nguồn mở dùng để phát triển các trò chơi 2D. Pygame hỗ trợ việc xử lý đồ họa, âm thanh, sự kiện bàn phím, chuột và quản lý sprite rất hiệu quả.

Một số tính năng chính của Pygame được sử dụng trong game gồm:

- Hiển thị hình ảnh (sprite và nền).
- Quản lý sự kiện bàn phím.
- Quản lý nhóm sprite (pygame.sprite.Group).
- Phát âm thanh nền và hiệu ứng âm thanh.
- Xử lý va chạm giữa các sprite (collision detection).
- Hiệu ứng rung màn hình (screen shake).

2.2. Các cấu trúc dữ liệu chính và đối tượng trong game

2.2.1. Class Ship (Tàu vũ trụ)

- Đại diện cho tàu vũ trụ do người chơi điều khiển.
- Thuộc lớp kế thừa *pygame.sprite.Sprite*.
- Thuộc tính:
  - *image, rect*: hình ảnh và vị trí tàu.
  - *direction:* vector chỉ hướng di chuyển và bắn.
  - *angle:* góc xoay hình ảnh để thể hiện hướng.
  - *speed:* tốc độ di chuyển.
  - *last_fire:* thời gian lần bắn tên lửa cuối cùng.
- Phương thức:
  - *update():* cập nhật vị trí và hướng theo phím bấm.

- *fire():* tạo tên lửa theo hướng hiện tại.
- *can_fire():* kiểm tra khoảng cách thời gian để bắn tiếp theo.

2.2.2. Class Asteroid (Thiên thạch)

- Đại diện thiên thạch bay xuống màn hình.
- Thuộc lớp *pygame.sprite.Sprite.*
- Thuộc tính:
  - *original_image, image:* ảnh thiên thạch và phiên bản được scale theo kích thước ngẫu nhiên.
  - *rect:* vị trí.
  - *speed_x, base_speed_y:* tốc độ di chuyển theo trục X và Y.
  - *mask:* dùng để xử lý va chạm chính xác.
- Phương thức:
  - *reset():* tái tạo vị trí, kích thước và tốc độ khi thiên thạch rơi ra ngoài màn hình.
  - *update():* di chuyển thiên thạch, tăng tốc theo thời gian.

2.2.3. Class Missile (Tên lửa)

- Đại diện cho đạn được bắn từ tàu.
- Thuộc lớp *pygame.sprite.Sprite.*
- Thuộc tính:
  - *image, rect:* hình ảnh tên lửa và vị trí.
  - *direction:* vector hướng bay.
  - *speed:* tốc độ bay.
- Phương thức:
  - *update():* di chuyển tên lửa theo hướng; nếu ra khỏi màn hình thì tự hủy.

2.2.4. Class Explosion (Hiệu ứng nổ)

- Đại diện cho hiệu ứng nổ khi thiên thạch bị bắn trúng.
- Thuộc lớp *pygame.sprite.Sprite.*
- Thuộc tính:
  - *image, rect:* hình ảnh và vị trí nổ.
  - *spawn_time:* thời điểm tạo để xác định thời gian tồn tại.
- Phương thức:
  - *update():* tự động xóa sprite sau một khoảng thời gian ngắn (0.3 giây).

2.2.5. Tài nguyên đa phương tiện

Tất cả hình ảnh và âm thanh được đặt trong thư mục *assets/:*

- *ship.png:* Hình ảnh tàu vũ trụ.
- *asteroid.png:* Hình ảnh thiên thạch.
- *missile.png:* Hình ảnh tên lửa.
- *explosion.png:* Hình ảnh hiệu ứng nổ.
- *background.png:* Hình nền không gian.
- *boom.wav:* Âm thanh nổ – được phát mỗi khi thiên thạch bị tiêu diệt.
- *music.mp3:* Nhạc nền – đượ phát liên tục trong quá trình chơi.

2.3. Các nhóm sprite (Sprite Groups)

- Sử dụng *pygame.sprite.Group* để quản lý các đối tượng có cùng loại hoặc cùng chức năng.
- Các nhóm trong game gồm:
  - *all_sprites:* chứa tất cả các sprite để cập nhật và vẽ.
  - *missiles:* chứa tên lửa để xử lý va chạm riêng biệt.
  - *asteroids:* chứa các thiên thạch.
  - *explosions:* chứa các hiệu ứng nổ.

2.4. Quản lý và xử lý sự kiện

- Sử dụng vòng lặp *while True* để chạy game liên tục.
- Bắt sự kiện hệ thống (thoát game).
- Bắt phím bấm để điều khiển tàu (phím mũi tên) và bắn tên lửa (phím space).
- Giới hạn tốc độ bắn tên lửa bằng cách kiểm tra khoảng thời gian giữa các lần bắn *(can_fire).*

2.5. Xử lý va chạm (Collision Detection)

- Va chạm tên lửa với thiên thạch:
  - Dùng *pygame.sprite.groupcollide* để kiểm tra trùng nhau.
  - Xóa tên lửa và thiên thạch bị trúng.
  - Tạo hiệu ứng nổ và tăng điểm số.
- Va chạm tàu với thiên thạch:
  - Dùng *pygame.mask* để kiểm tra va chạm chính xác dựa trên pixel.
  - Nếu va chạm xảy ra, dừng nhạc nền, hiển thị thông báo "GAME OVER - Press any key to restart", chờ người chơi nhấn phím bất kỳ để chơi lại.

2.6. Hiệu ứng rung màn hình (Screen Shake)

- Khi xảy ra va chạm tên lửa với thiên thạch, thiết lập biến *shake* = 10.
- Mỗi frame, biến *shake* giảm dần.
- Khi vẽ màn hình và sprite, dịch chuyển vị trí vẽ theo một lượng ngẫu nhiên trong khoảng *[-shake, shake]* để tạo hiệu ứng rung.

2.7. Tăng độ khó theo thời gian

- Tốc độ thiên thạch tăng lên theo thời gian chơi, mỗi 15 giây tăng 20% tốc độ *(speed_multiplier).*
- Số lượng thiên thạch tối đa là 6, game sẽ tự động sinh thêm thiên thạch mới sau mỗi 5 giây nếu chưa đủ số lượng.

2.8. Quản lý điểm số và hiển thị

- Biến *score* giữ điểm số, tăng 10 điểm mỗi khi tên lửa trúng thiên thạch.
- Dùng *pygame.font.SysFont* để tạo font chữ và hiển thị điểm số góc trên trái màn hình.

2.9. Các kiểu dữ liệu và cấu trúc dữ liệu sử dụng

- **List:** Không dùng trực tiếp trong code chính, nhưng các nhóm sprite có thể hiểu như danh sách quản lý nhiều đối tượng.
- **Vector2 (pygame.math.Vector2):** Dùng để biểu diễn hướng và vận tốc trong không gian 2 chiều.
- **Dictionary / Tuple:** Không sử dụng nhiều, các tham số vị trí sử dụng tuple *(x, y)*.

2.10. Tóm tắt luồng hoạt động của game

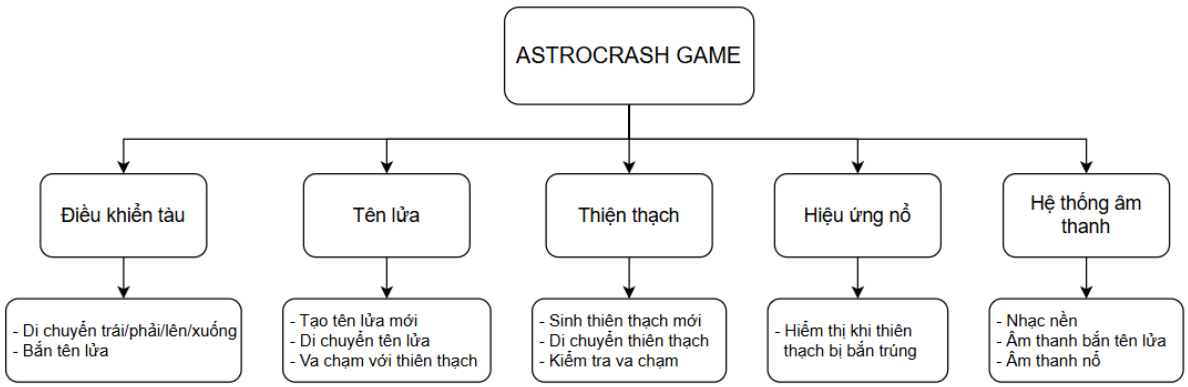
1. Khởi tạo các sprite, nhóm sprite và các biến điều khiển.
2. Vòng lặp game xử lý sự kiện người dùng.
3. Cập nhật vị trí và trạng thái của các sprite.
4. Xử lý va chạm và các hiệu ứng tương ứng.
5. Vẽ nền, sprite, điểm số lên màn hình.
6. Tăng dần độ khó bằng cách tăng tốc độ thiên thạch và sinh thêm thiên thạch.
7. Xử lý kết thúc và chơi lại khi tàu bị va chạm.

CHƯƠNG 3. THIẾT KẾ VÀ XÂY DỰNG CHƯƠNG TRÌNH

3.1. Sơ đồ khối hệ thống

Hệ thống game **Astrocrash** được chia thành các module chính:

- **main.py:** Tập tin khởi chạy chính, xử lý vòng lặp game và tích hợp các thành phần khác.
- **ship.py:** Xử lý điều khiển tàu (người chơi), di chuyển và phát tên lửa.
- **missile.py:** Quản lý tên lửa do tàu bắn ra.
- **asteroid.py:** Quản lý thiên thạch, tạo mới, di chuyển và va chạm.
- **explosion.py:** Hiển thị hiệu ứng nổ khi thiên thạch bị phá hủy.
- Thư mục **assets/:** Chứa hình ảnh (.png) và âm thanh (.wav, .mp3) cho game.



Hình 1. Biểu đồ phân cấp chức năng

3.2. Sơ đồ khối các thuật toán

Các thuật toán chính của hệ thống bao gồm:

1. Vòng lặp chính

- **Input**: Phím điều khiển từ người dùng
- **Output**: Cập nhật vị trí tàu, thiên thạch, tên lửa, điểm số, hiệu ứng
- **Chức năng**:

    - Cập nhật trạng thái game (vật thể, va chạm)
    - Hiển thị hình ảnh
    - Phát âm thanh

2. Điều khiển tàu

- Xử lý phím mũi tên để di chuyển
- Giới hạn tàu trong màn hình

3. Bắn tên lửa

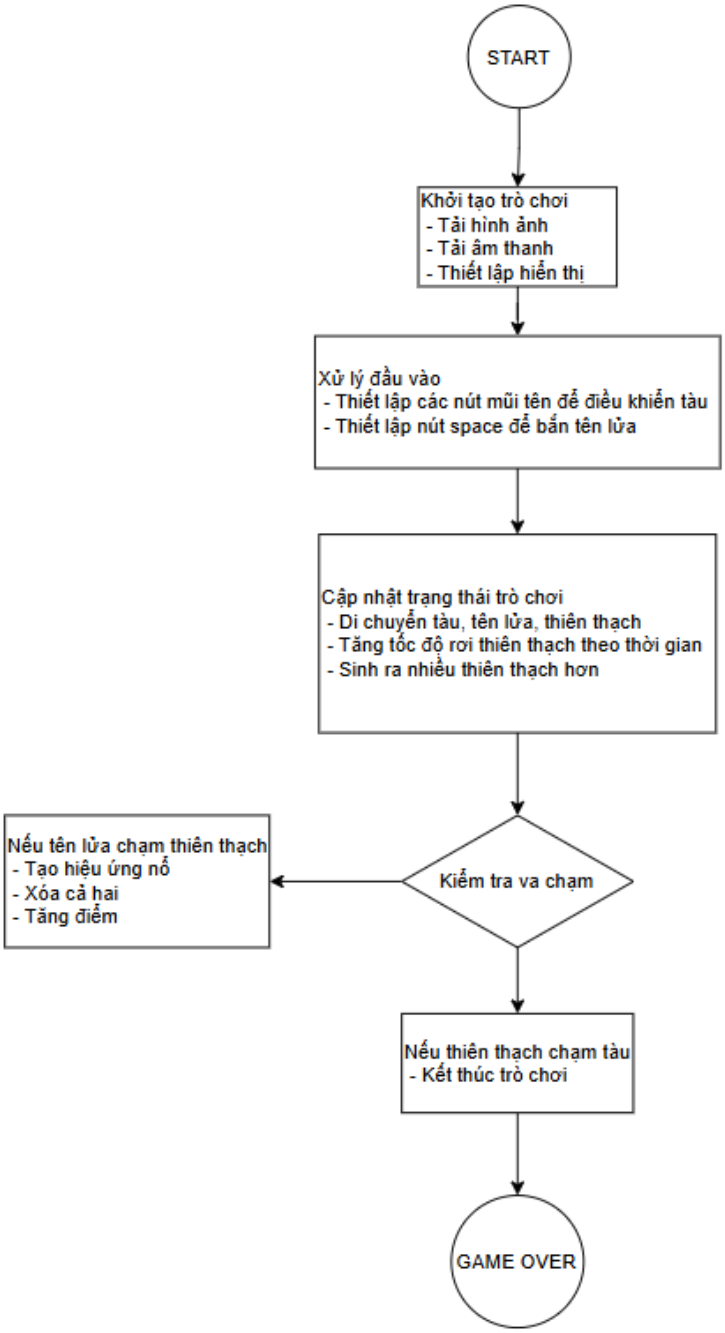
- Khi nhấn Space, tạo tên lửa tại vị trí tàu
- Thêm vào danh sách tên lửa đang hoạt động

4. Tạo và di chuyển thiên thạch

- Theo thời gian, sinh thiên thạch mới
- Di chuyển thiên thạch xuống dưới
- Tăng tốc theo thời gian

5. Kiểm tra va chạm

- Nếu tên lửa chạm thiên thạch:
    - Tạo hiệu ứng nổ
    - Xóa cả hai
    - Tăng điểm
- Nếu thiên thạch chạm tàu:
    - Kết thúc trò chơi



Hình 2. Sơ đồ khối các thuật toán

3.3. Cấu trúc dữ liệu

Không dùng cơ sở dữ liệu, mà chủ yếu dùng danh sách trong Python để quản lý vật thể:

Biến/Tên lớp	Kiểu dữ liệu	Mục đích
Ship	class	Quản lý thông tin tàu
Missile	class	Quản lý tên lửa
Asteroid	class	Quản lý thiên thạch
Explosion	class	Hiệu ứng nổ
missiles	list	Danh sách các tên lửa đang bay
asteroids	list	Danh sách các thiên thạch đang rơi
explosions	list	Danh sách hiệu ứng đang hoạt động
score	int	Điểm của người chơi

Hình 3. Cấu trúc dữ liệu

3.4. Chương trình

Mỗi module chứa các hàm chính như sau:

main.py

- *main():* Khởi động game, xử lý vòng lặp chính
- Gọi và cập nhật các đối tượng trong game
- Phát âm thanh và quản lý điểm

ship.py

- *Ship.update():* Cập nhật vị trí theo phím bấm
- *Ship.draw(screen):* Vẽ tàu lên màn hình
- *Ship.fire():* Tạo tên lửa mới

missile.py

- *Missile.update():* Di chuyển lên trên
- *Missile.draw(screen):* Vẽ tên lửa

asteroid.py

- *Asteroid.update():* Di chuyển xuống
- *Asteroid.draw(screen):* Vẽ thiên thạch

explosion.py

- *Explosion.update():* Cập nhật khung ảnh
- *Explosion.draw(screen):* Vẽ hiệu ứng nổ

CHƯƠNG 4. THỰC NGHIỆM VÀ KẾT LUẬN

4.1. Thực nghiệm

Các tính năng chính của chương trình Astrocrash đã được kiểm thử đầy đủ, bao gồm:

- **Điều khiển tàu vũ trụ**: Người chơi sử dụng các phím mũi tên trái và phải để di chuyển tàu trên màn hình, giúp tránh né các thiên thạch rơi xuống.
- **Bắn tên lửa**: Khi nhấn phím cách (space), tàu sẽ bắn ra tên lửa theo phương thẳng đứng để tiêu diệt thiên thạch.
- **Thiên thạch rơi ngẫu nhiên và tăng tốc dần**: Các thiên thạch xuất hiện ngẫu nhiên ở phía trên màn hình và rơi xuống, với tốc độ tăng dần theo thời gian để tăng mức độ thử thách.
- **Hiệu ứng nổ và âm thanh**: Khi tên lửa bắn trúng thiên thạch, hiệu ứng nổ cùng âm thanh "boom" sẽ được phát ra, tạo cảm giác chân thực.
- **Hiệu ứng rung khi va chạm**: Nếu tàu va chạm với thiên thạch, màn hình sẽ rung nhẹ, kết hợp với âm thanh va chạm để tạo hiệu ứng sống động.
- **Tính điểm và hiển thị điểm số**: Mỗi lần tiêu diệt thiên thạch, người chơi được cộng điểm. Nếu va chạm, điểm sẽ bị trừ. Điểm số luôn được hiển thị ở góc màn hình.
- **Giao diện đồ họa và âm thanh nền**: Hình ảnh tàu, tên lửa, thiên thạch và nền được thiết kế rõ nét, âm thanh nền được phát xuyên suốt trò chơi, không gây trễ hoặc giật.

Hình 4. Giao diện game khi tiêu diệt thiên thạch

Hình 5. Giao diện game khi tàu va chạm với thiện thạch, kết thúc trò chơi

4.2. Kết luận

Kết quả đạt được

- Hoàn thiện một trò chơi bắn thiên thạch 2D bằng Python sử dụng thư viện Pygame.
- Tích hợp các tính năng chính: điều khiển, bắn tên lửa, hiệu ứng nổ, điểm số, âm thanh và hiệu ứng rung.
- Tổ chức mã nguồn thành các module rõ ràng như: main.py, asteroid.py, missile.py, explosion.py, ship.py.

Bài học rút ra

- Nâng cao kỹ năng lập trình hướng đối tượng (OOP) trong Python.
- Làm quen với thư viện Pygame và cách xử lý ảnh, âm thanh, sự kiện bàn phím.
- Hiểu cách tổ chức và quản lý dự án theo cấu trúc module.

Hướng cải tiến

- Thêm màn chơi (level), tăng độ khó theo thời gian hoặc sau mỗi mốc điểm.
- Tích hợp menu chính, bảng xếp hạng điểm cao.
- Tối ưu hiệu suất cho các thiết bị yếu hơn.

TÀI LIỆU THAM KHẢO

1. https://mindx.edu.vn/blog/lap-trinh-game-2d
2. Search Art | OpenGameArt.org
3. Chat GPT
4. Youtobe