ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

SPACE INVADER BẰNG PYGAME

Sinh viên thực hiện:		
STT	Họ tên	MSSV
1	Nguyễn Minh Thiện	19522262
2		

TP. HỒ CHÍ MINH – 5/2022

GIỚI THIỆU

Space invader, một tựa game mà hẳn ai cũng biết đến được ra mắt vào năm 1978 và do Tomohiro Nishikado phát triển. Kể từ khi ra mắt space invader đã trở thành 1 tựa game vô cùng kinh điển và truyền cảm hứng cho nhiều tựa game cổ điển khác như *Defender, Asteroids, Galaxian and Galaga* và vô vàn tựa game thể loại "shoot 'em up" ngoài không gian. Như hầu hết các tựa game được tạo ra trong thập kỉ 80s, 90s space invader được lập trình bằng ngôn ngữ assembly. Tuy vậy để làm lại space invader từ đầu bằng assembly thì khá là khó nhằn vậy nên sử dụng các ngôn ngữ bậc cao phổ biến hiện nay với các thư viện hỗ trợ việc vẽ sprite lên màn hình và xử lí va cham khiến việc làm lại game space invader đơn giản hơn nhiều. Python là 1 trong những ngôn ngữ bậc cao phổ biến hiện nay và để làm game bằng python thì thư viện hỗ trợ phổ biến nhất đó là Pygame. Để hiểu được những tính năng cơ bản của Pygame thì làm lại space invader là vô cùng bổ ích bởi vì trong space invader sẽ có các tính năng cần làm như xử lí va chạm, vẽ sprite lên màn hình, kiểm tra boundary, âm thanh, … Đó là những tính năng cơ bản vô cùng phổ biến và cần có cho hầu hết tất cả mọi game.

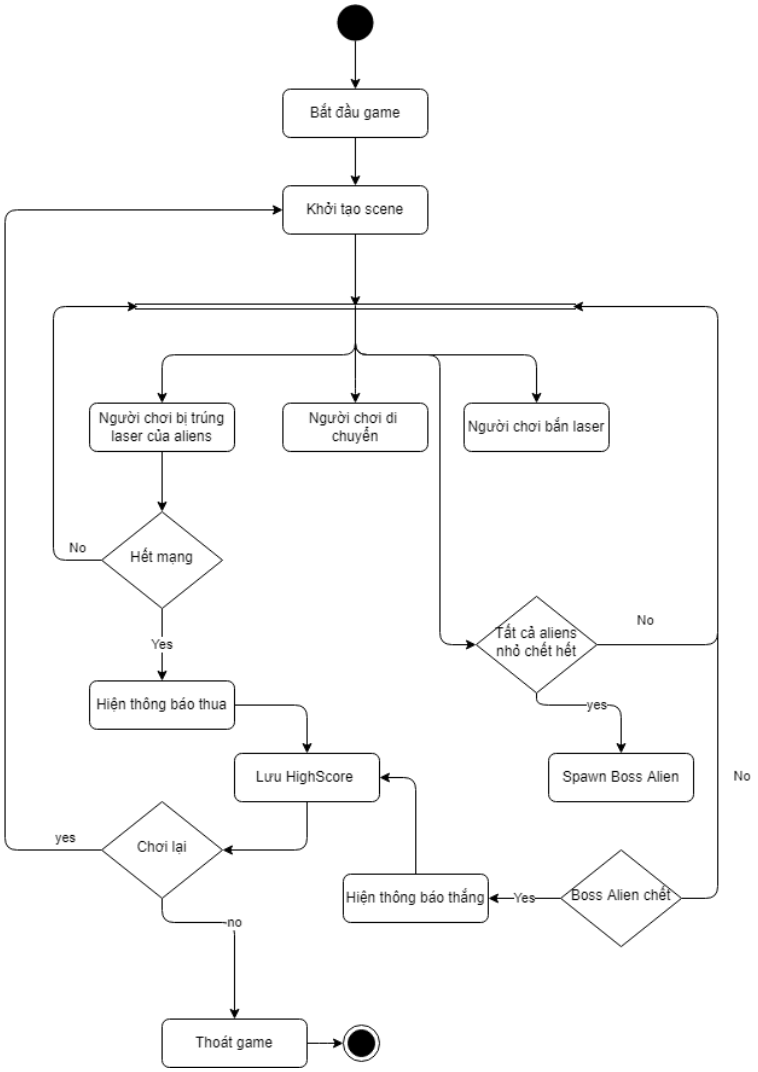
Báo cáo này sẽ tập trung vào ba nội dung chính:

(1) Quy trình áp dụng pygame để xây dựng các thành phần của game

(2) Thiết lập game loop, tính điểm, xử lí va chạm và xử lí âm thanh

(3) Đánh giá kết quả đạt được so với kế hoạch đã lập ra

1. NỘI DUNG



Hình 1. Sơ đồ quy trình hệ thống

1.1. Quy trình áp dụng pygame để xây dựng các thành phần của game

Đầu tiên cần tìm kiếm các assets cần thiết trong game như sprites nhân vật, aliens, boss aliens, extra, laser, laser beam và các hiệu ứng âm thanh, nhạc để đưa vào game

Sau đó cần xây dựng các thành phần lớp chính có trong game:

+ Game
+ Block
+ Player
+ Laser, Laser Beam
+ Alien, Extra
+ Boss Alien

Tất cả các lớp trừ *Game* đều kế thừa từ lớp *Sprite* của pygame để tiện xử lí va chạm và vẽ các sprite của người chơi và alien lên màn hình bằng các biến và phương pháp kế thừa từ *Sprite* như rect, image, draw() ,…

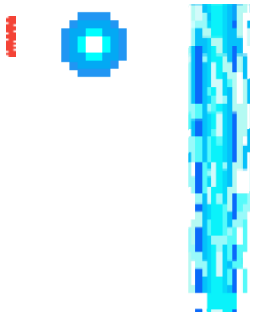
- Lớp *Game* dùng chủ yếu load các assets có trong game và khởi tạo scene cho game và đồng thời cũng xử lí các hành động như pause/unpause game, thắng/thua game, chơi lại, điều khiển *Alien* và triệu hồi *Boss Alien* khi các *Alien* nhỏ chết hết, tăng tốc độ di chuyển qua lại và xuống của các *Alien* nhỏ khi số lượng *Alien* trên màn hình vơi đi đến 1 số lượng nào đó, chạy Soundtrack cho game và đổi Soundtrack khi triệu hồi *Boss Alien*, hiện điểm và mạng của người trên lên góc màn hình, hiện thông báo thắng/thua sau khi hết game sử dụng, vẽ các sprites lên màn hình.
- Lớp *Player* có những phương pháp dùng để di chuyển nhân vật qua lại trái phải như trong game Space Invader cổ điển, Kiểm tra xem nhân vật có đi quá giới hạn màn hình hay không và giữ nhân vật trong giới hạn của màn hình, phương thức để tính tốc độ bắn laser cho người chơi, trừ mạng của người chơi khi bị trúng laser, phương thức để bắn laser và phương thức update() đặt trong vòng lặp của game để thực hiện hành động của nhân vật mỗi frame.

Hình 2. Hình minh họa người chơi

- Lớp *Alien, Extra, Boss Alien* sẽ có nhiều sprite để hiện lên màn hình khác với lớp *Player* vậy nên sẽ có phương thức để đổi sprite của các lớp đó tùy theo thời gian mỗi animation frame hiện trên màn hình. Cả ba lớp này đều có phương thức di chuyển qua lại tuy nhiên chỉ *Alien* và *Boss Alien* mới có thể di chuyển xuống và có tốc độ thay đổi tùy theo phase. Lớp *Extra* sẽ được vẽ và bắt đầu di chuyển trên màn hình sau 1 khoảng thời gian ngắn ngẫu nhiên trong 1 khoảng nhất định, lớp *Alien* được vẽ lên màn hình theo nhiều đội hình đã vẽ sẵn khác nhau được chọn ngẫu nhiên khi thiết lập scene. Chỉ có lớp *Alien* và *Boss Alien* mới có thể bắn lasers ra được ngoài ra lớp *Boss Alien* có thể bắn ra *Laser Beam* và làm nhân vật hết mạng ngay lập tức khi đụng phải. Thời gian có thể bắn những lasers này phụ thuộc vào tốc độ bắn của *Alien* và *Boss Alien* và sử dụng pygame.time.get_ticks() để tính toán thời gian.

Hình 3. Hình minh họa Alien, Boss Alien và Extra

- Lớp *Laser* được sử dụng bởi cả *Player* và *Alien, Boss Alien* để bắn ra. Sẽ có 1 biến để xác định laser bắn ra đi theo hướng nào (trên -> dưới / dưới -> trên). Ngoài ra Laser còn có tốc độ có thể thay đổi được vào lúc khởi tạo.
- *Laser Beam* chỉ được sử dụng bởi Boss Alien để bắn ra laser to hơn hẳn laser thông thường với 1 khoảng thời gian delay trước khi *Laser Beam* bắn ra



Hình 4. Hình minh họa Laser và Laser Beam

- Lớp *Block* kế thừa *Sprite* của pygame dùng để xây dựng các rào chắn hay obstacles để người chơi đứng sau đó để tránh lasers của *Aliens* và *Boss Alien*. Các block được sắp xếp theo 1 đội hình nhất định, mỗi đội hình đó được gọi là obstacle và có nhiều obstacles có thể vẽ lên màn hình và tương tác với laser của *Player* và *Alien*/*Boss Alien*

Hình 5. Hình minh họa 1 obstacle gồm nhiều Block tạo thành

1.2. Thiết lập game loop, tính điểm, xử lí va chạm và xử lí âm thanh

- Để thiết lập game loop chỉ cần khởi tạo lớp *Game* với các thành phần cần thiết của pygame như *font* để hiện chữ và *mixer* cho âm thanh sau đó chạy 1 phương thức trong *Game* có tên "run()" trong 1 vòng lặp while chỉ kết thúc khi thoát game. Trong phương thức run sẽ thực hiện các hành động như vẽ sprites lên màn

hình, di chuyển *Alien, Boss Alien, Extra,* xử lí input của người chơi và xử lí va chạm. Nếu mạng của nhân vật nhỏ hơn hoặc bằng 0 hoặc tất cả aliens chết hết *Game* sẽ chạy hàm pause và đưa lên màn hình kết quả thắng/thua, lưu highscore nếu điểm hiện tại lớn hơn highscore đã lưu và cho người chơi lựa chọn chơi lại khi ấn phím "R". Nếu người chơi chọn chơi lại *Game* sẽ thiết lập lại scene với các thông số mặc định ban đầu khi khởi tạo với đội hình aliens ngẫu nhiên.

- Xử lí va chạm sử dụng hàm spritecollide của pygame để kiểm tra xem size của rect của 1 sprite có nằm đè lên 1 sprite khác không nếu có thì xử lí. Hầu hết các sprite trong game đều có 1 rect size nhất định tùy theo size (width,height) của file của sprite đó
- Âm thanh sẽ được load vào trong quá trình khởi tạo game. Khi người chơi hoặc aliens bắn laser sẽ có 1 âm thanh đi kèm với hành động đó, các laser của *Boss Alien* sẽ có âm thanh khác so với của người chơi và các aliens nhỏ. Sẽ có 2 Soundtrack trong game 1 là mặc định và 2 là nhạc khi *Boss Alien* triệu hồi vào scene. Khi *Alien* và *Boss Alien* bị bắn trúng sẽ có âm thanh đi kèm.
- Tính điểm:
    + Alien: 10 points
    + Boss Alien: 50 points mỗi khi bắn trúng
    + Extra: 100 points

2. KẾT LUẬN

Bằng việc sử dụng thư viện Pygame và các phương pháp lập trình Python nhóm đã hoàn thành hết tất cả mục tiêu đã đặt ra trong việc tái tạo lại game Space Invader trong thời buổi hiện đại với 1 số tính năng mới so với Space Invader cổ điển như đội hình aliens ngẫu nhiên, Boss Alien. Tuy kết quả đặt được đúng với những gì đã đặt ra nhưng sản phẩm có được vẫn thiếu những tính năng cơ bản nên có cho những game hiện đại thời nay như lưu tên người chơi highscore, bảng xếp hạng highscore, menu screen, settings để chỉnh âm thanh, chỉnh mức độ khó của game, lựa chọn formation của aliens mà người chơi muốn chơi, pixel perfect collision detection,… Đây là những tính năng mà trong tương lai nếu có cơ hội quay lại nhóm sẽ phát triển thêm vào. Sau đồ án này nhóm có thêm được kiến thức, trải nghiệm trong việc phát triển game bằng python thông qua sự hỗ trợ của thư viện Pygame và điểm mạnh yếu của Pygame cũng như ngôn ngữ lập trình python

TÀI LIỆU THAM KHẢO

[1] Pygame: https://www.pygame.org/docs/

[2] Lưu và lấy thông tin highscore bằng Json với Pygame: https://www.youtube.com/watch?v=__mZO-53PPM

[3] Xứ lí va chạm trong Pygame: https://www.youtube.com/watch?v=6LF4k-uc75c

[4] Assets mẫu:https://opengameart.org/content/assets-for-a-space-invader-like-game

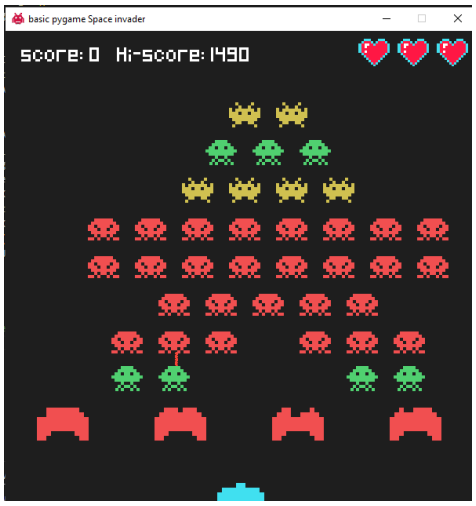
[5] Thêm âm thanh và nhạc nền vào game: https://www.youtube.com/watch?v=pcdB2s2y4Qc

PHỤ LỤC

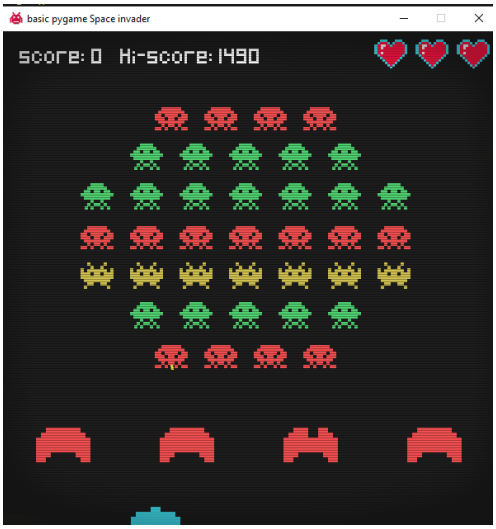
Gameplay demo: https://youtu.be/N17xEoyEkKY

Filter CRT để game nhìn trông đẹp hơn:

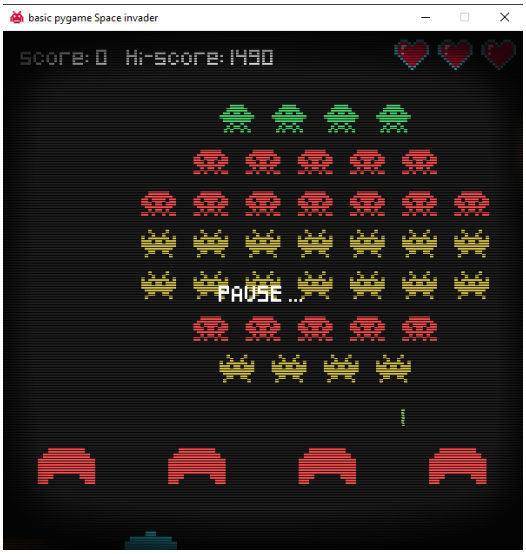
Nếu thông thường không có hiệu ứng thì game sẽ thiếu thẩm mỹ và không bắt mắt đồng thời khi pause game khi không có hiệu ứng CRT sẽ khó thấy được tại không có gì để thể hiện rằng game đang tạm thời ngưng lại. Vì thế thêm hiệu ứng CRT để viền game có thêm khung trông giống TV khiến game trở nên thêm bắt mắt đồng thời cho hiệu ứng nhấp nháy bằng cách thay đổi Opacity của hiệu ứng mỗi frame giúp mô phỏng màn hình TV CRT ngày xưa khiến cho hình ảnh trong game trở nên thú vị hơn hẳn so với mặc định.



Hình 6. Screenshot game khi không có hiệu ứng CRT



Hình 7. Screenshot game với hiệu ứng CRT



Hình 8. Pause game với hiệu ứng CRT

PHỤ LỤC PHÂN CÔNG NHIỆM VỤ

STT	Thành viên	Nhiệm vụ
1	Nguyễn Minh Thiện	Tìm hiểu Pygame, kiếm/vẽ assets cho game, viết báo cáo và lập trình
2		
3		