ỦY BAN NHÂN DÂN THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC SÀI GÒN

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN MÔN NGÔN NGỮ LẬP TRÌNH PYTHON

Tên đề tài: Xây dựng ứng dụng game Flappy Bird

Sinh viên thực hiện:

Lê Anh Khoa – 3122410184

Giảng viên: **Trịnh Tấn Đạt**

Thành phố Hồ Chí Minh, 10 tháng 5 năm 2024

NHẬN XÉT, ĐÁNH GIÁ CỦA GIẢNG VIÊN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

LỜI CẢM ƠN

Em xin gửi lời cảm ơn chân thành và sự tri ân sâu sắc đối với các thầy cô của trường Đại Học Sài Gòn, đặc biệt là các thầy cô ở khoa Công Nghệ Thông Tin của trường đã tạo điều kiện cho em tiếp cận và tìm hiểu để hoàn thành đồ án môn học lần này. Và em cũng xin chân thành cám ơn thầy Trịnh Tấn Đạt giáo viên giảng dạy đã nhiệt tình hướng dẫn chúng em hoàn thành đồ án lần này.

Trong quá trình nghiên cứu và làm bài báo cáo đồ án, do kiến thức cũng như kinh nghiệm thực tế còn nhiều hạn chế nên bài báo cáo không thể tránh khỏi những thiếu sót, chúng em rất mong nhận được ý kiến đóng góp thầy, cô để em học hỏi được nhiều kĩ năng, kinh nghiệm và sẽ hoàn thành tốt hơn cho những bài báo cáo sắp tới.

Em xin chân thành cảm ơn thầy ạ!

MỤC LỤC

Phần I. MỞ ĐẦU 6

1. Giới thiệu đề tài 6
2. Lý do chọn đề tài 7
3. Mục đích - mục tiêu của để tài 7
PHẦN II. NỘI DUNG 8
1. Đôi nét về game Flappy Bird 8
1.1 Giới thiệu về Flappy Bird 8
1.2 Mục tiêu của game Flappy Bird 8
2. Yêu cầu đối với đồ án 8
3. Tiến hành xây dựng ứng dụng 9
3.1.Cài đặt pygame và các biến cho trò chơi 9
3.2. Tạo while loop.....12
3.3. Tạo hàm.....15
4. Kết luận ..... **Error! Bookmark not defined.**

Phần I. MỞ ĐẦU

1. Giới thiệu đề tài

Tên đề tài:

Xây dựng ứng dụng trò chơi Flabby Pird với thư viện Pygame bằng ngôn ngữ lập trình Python.

2. Lý do chọn đề tài.

Python là một ngôn ngữ lập trình phổ biến, dễ học và mạnh mẽ. Nó được sử dụng rộng rãi trong xây dựng trang web, phần mềm, tự động hóa tác vụ và phân tích dữ liệu. Python có cấu trúc cú pháp đơn giản, thân thiện với người mới bắt đầu, và là lựa chọn phổ biến cho các ứng dụng machine learning và deep learning. Em cũng thấy hứng thú với việc sử dụng thư viện Pygame để xây dựng lại trò chơi Flappy Bird.

3. Mục đích - mục tiêu của để tài.

- Mục đích:

+ Nắm chắc được được kỹ năng và kiến thức về lập trình.

+ Tìm hiểu về thư viện Pygame trong ngôn ngữ lập trình Python.

+ Cũng cố, áp dụng, nâng cao kiến thức đã được học.

+ Nắm bắt được quy trình làm game cơ bản.

- Mục tiêu:

+ Vận dụng được tính chất của lập trình hướng đối tượng.

+ Sử dụng thư viện Pygame vào việc xây dựng game Flappy Bird

PHẦN II. NỘI DUNG

1. Đôi nét về game Flappy Bird

1.1 Giới thiệu về Flappy Bird

Flappy Bird (tạm dịch là Chú chim vỗ cánh) là một trò chơi điện tử trên điện thoại do Nguyễn Hà Đông, một lập trình viên ở Hà Nội, Việt Nam phát triển, và do dotGEARS, một studio phát triển game quy mô nhỏ, hoạt động độc lập có trụ sở tại Việt Nam phát hành vào năm 2013. Trò chơi được trình bày theo phong cách side-scroller (phong cách game với các đối tượng được nhìn thấy ở mặt bên (side-view) và di chuyển từ cạnh trái sang cạnh phải của màn hình), trong đó người chơi điều khiển một chú chim, cố gắng vượt qua các hàng ống màu xanh lá cây mà không chạm vào chúng. Nguyễn Hà Đông tạo ra Flappy Bird trong vòng một vài ngày, sử dụng một nhân vật chú chim mà anh đã thiết kế cho một dự án trò chơi bị hủy bỏ vào năm 2012

1.2 Mục tiêu của game Flappy Bird

Mục tiêu của Flappy Bird là ghi càng nhiều điểm càng tốt bằng cách điều kiến chim xanh vượt qua các ống. Khi vượt qua, người chơi sẽ có thêm điểm, người chơi sử dụng phía mũi tên lên để chơi trò chơi, nếu chạm vào ống hoặc rơi xuống đất hoặc bay quá cao, người chơi phải chơi lại từ đầu

2. Yêu cầu đối với đồ án

Xây dựng 1 ứng dụng giống với Flappy Bird dựa vào những kiến thức đã học về ngôn ngữ lập trình Python.

3. Tiến hành xây dựng ứng dụng

3.1.Cài đặt pygame và các biến cho trò chơi

```
import pygame, sys, random
```

Import pygame, sys

Dòng trên dùng để khai báo các thư viện cần thiết.

```
pygame.mixer.pre_init(frequency=44100, size=-16, channels=2, buffer=512)
pygame.init()
screen= pygame.display.set_mode((432,768))
clock = pygame.time.Clock()
game_font = pygame.font.Font('04B_19.ttf',35)
```

Khởi tạo màn hình game

pygame.mixer.pre_init(frequency=44100, size=-16, channels=2, buffer=512): Dòng này thiết lập các thông số cho hệ thống âm thanh của Pygame. Cụ thể, nó đặt tần số mẫu (frequency) là 44100 Hz, kích thước âm thanh (size) là -16 bit (âm thanh stereo), số kênh (channels) là 2 (âm thanh stereo), và kích thước bộ đệm (buffer) là 512 byte.

pygame.init(): Dòng này khởi tạo Pygame. Bạn cần gọi hàm này trước khi sử dụng bất kỳ chức năng nào của Pygame.

screen= pygame.display.set_mode((432,768)): Dòng này tạo ra một cửa sổ trò chơi với kích thước 432x768 pixel.

clock = pygame.time.Clock(): Dòng này tạo ra một đối tượng đồng hồ (clock) để kiểm soát tốc độ khung hình của trò chơi.

game_font = pygame.font.Font('04B_19.ttf',35): Dòng này tạo ra một đối tượng font chữ (game_font) từ tệp tin font '04B_19.ttf' với kích thước 35 pixel.

```
gravity = 0.25
bird_movement = 0
game_active = True
score = 0
high_score = 0
```

Tạo các biến:

gravity: Biến này đại diện cho trọng lực trong trò chơi. Khi chim không được điều khiển, nó sẽ rơi xuống dưới màn hình. Giá trị của gravity thường được đặt trong khoảng từ 0.2 đến 0.3 để tạo cảm giác tự nhiên cho chuyển động của chim.

bird_movement: Biến này theo dõi chuyển động của con chim. Khi bạn nhấn phím để chim bay lên, giá trị của biến này sẽ tăng lên (ví dụ: bird_movement -= 5 để chim bay lên).

game_active: Biến này kiểm tra trạng thái của trò chơi. Nếu game_active là True, trò chơi đang chạy; nếu là False, trò chơi đã kết thúc (ví dụ: khi chim va chạm với ống hoặc bay quá cao).

score: Biến này lưu trữ điểm số hiện tại của người chơi. Điểm số tăng lên khi chim vượt qua mỗi cặp ống.

high_score: Biến này lưu trữ điểm số cao nhất đã đạt được trong lần chơi trước. Nó được cập nhật khi người chơi đạt được điểm số cao hơn.

```
bg = pygame.image.load('assets/background-night.png').convert()
bg = pygame.transform.scale2x(bg)
```

Dòng mã trên đang tải hình nền. Dòng đầu tiên sử dụng pygame.image.load('assets/background-night.png') để tải hình ảnh từ tệp tin 'background-night.png'. Sau đó, dòng thứ hai sử dụng pygame.transform.scale2x(bg) để tăng kích thước của hình nền lên gấp đôi. Điều này giúp hình nền phù hợp với kích thước màn hình của trò chơi.

```
floor = pygame.image.load('assets/floor.png').convert()
floor = pygame.transform.scale2x(floor)
floor_x_pos = 0
```

Dòng mã trên đang tải hình ảnh sàn. Hình ảnh sàn được lấy từ tệp tin 'floor.png' và sau đó được tăng kích thước lên gấp đôi để phù hợp với kích thước màn hình của trò chơi. Biến floor_x_pos đại diện cho vị trí x của sàn trên màn hình.

```
bird_down = pygame.transform.scale2x(pygame.image.load('assets/yellowbird-downflap.png').convert_alpha())
bird_mid = pygame.transform.scale2x(pygame.image.load('assets/yellowbird-midflap.png').convert_alpha())
bird_up = pygame.transform.scale2x(pygame.image.load('assets/yellowbird-upflap.png').convert_alpha())
bird_list= [bird_down,bird_mid,bird_up] #0 1 2
bird_index = 0
bird = bird_list[bird_index]
bird_rect = bird.get_rect(center = (100,384))
```

bird_down, bird_mid, bird_up: Đây là ba hình ảnh khác nhau của con chim. Mỗi hình ảnh đại diện cho trạng thái của chim khi nó bay xuống, ở giữa và khi nó bay lên. Các hình ảnh này được tải từ các tệp tin 'yellowbird-downflap.png', 'yellowbird-midflap.png' và 'yellowbird-upflap.png'.
bird_list: Biến này lưu trữ danh sách các hình ảnh chim. Trong trường hợp này, danh sách chứa ba hình ảnh của con chim.
bird_index: Biến này đang chỉ đến hình ảnh hiện tại của con chim trong danh sách bird_list. Khi muốn thay đổi hình ảnh của con chim (ví dụ: khi chim bay lên hoặc xuống), có thể thay đổi giá trị của biến này để chọn hình ảnh phù hợp.
Biến bird_rect trong mã đang tạo một hình chữ nhật xung quanh hình ảnh của con chim. Hình chữ nhật này có tâm ở tọa độ (100, 384), tức là con chim sẽ xuất hiện ban đầu tại vị trí đó trên màn hình. Hình chữ nhật này hữu ích cho việc phát hiện va chạm và định vị con chim trong môi trường trò chơi.

```
birdflap = pygame.USEREVENT + 1
pygame.time.set_timer(birdflap,200)
```

Dòng mã trên đang tạo một timer cho con chim trong trò chơi. Được thiết lập để phát sự kiện sau mỗi 200 miligiây (0.2 giây). Khi timer kích hoạt, người chơi có thể thực hiện các hành động liên quan đến con chim, chẳng hạn như thay đổi hình ảnh của nó để tạo hiệu ứng bay lên và xuống.

```
game_over_surface = pygame.transform.scale2x(pygame.image.load('assets/message.png').convert_alpha())
game_over_rect = game_over_surface.get_rect(center=(216,384))
```

Dòng mã trên đang tạo hình ảnh cho màn hình kết thúc trong trò. Hình ảnh này được lấy từ tệp tin 'message.png' và sau đó được tăng kích thước lên gấp đôi để phù hợp với kích thước màn hình của trò chơi.
Biến game_over_rect đại diện cho hình chữ nhật bao quanh hình ảnh, và nó được đặt tại tâm của màn hình (tọa độ (216, 384)).

```
flap_sound = pygame.mixer.Sound('sound/sfx_wing.wav')
hit_sound = pygame.mixer.Sound('sound/sfx_hit.wav')
score_sound = pygame.mixer.Sound('sound/sfx_point.wav')
score_sound_countdown = 100
```

flap_sound: Đây là âm thanh khi con chim bay lên. Nó được tải từ tệp tin 'sfx_wing.wav'.
hit_sound: Đây là âm thanh khi con chim va chạm với ống hoặc màn hình. Nó được tải từ tệp tin 'sfx_hit.wav'.
score_sound: Đây là âm thanh khi bạn đạt được điểm số. Nó được tải từ tệp tin 'sfx_point.wav'.

score_sound_countdown: Biến này đếm ngược để điều chỉnh âm thanh khi người chơi đạt được điểm số. Khi biến này giảm xuống 0, âm thanh sẽ được phát.

3.2 Tạo while loop cho trò chơi

```
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            sys.exit()
```

Vòng lặp chính (while True):
Vòng lặp này sẽ chạy mã trong nó vô hạn (cho đến khi đóng cửa sổ trò chơi).
Nó sẽ xử lý các sự kiện như nhấn phím, va chạm, và di chuyển của con chim.
Xử lý sự kiện (for event in pygame.event.get():):
Vòng lặp này lấy danh sách các sự kiện từ Pygame.
Nếu sự kiện là thoát khỏi cửa sổ (pygame.QUIT), trò chơi sẽ kết thúc.

```
if event.type == pygame.KEYDOWN:
    if event.key == pygame.K_SPACE and game_active:
        bird_movement = 0
        bird_movement =-11
        flap_sound.play()
```

if event.type == pygame.KEYDOWN:: Dòng này kiểm tra xem có phím nào được nhấn không.
if event.key == pygame.K_SPACE and game_active:: Nếu phím được nhấn là phím cách (pygame.K_SPACE) và trò chơi đang hoạt động (không ở trạng thái kết thúc), các hành động sau sẽ xảy ra:
bird_movement được đặt về 0 (để ngừng mọi chuyển động lên hoặc xuống).
Sau đó, bird_movement được đặt về -11, khiến con chim bay lên.
Âm thanh "flap" (flap_sound) được phát, cung cấp phản hồi âm thanh cho người chơi.

```
  if event.key == pygame.K_SPACE and game_active==False:
        game_active = True
        pipe_list.clear()
        bird_rect.center = (100,384)
        bird_movement = 0
        score = 0
```

Nếu sự kiện là nhấn phím cách (pygame.K_SPACE) và trò chơi đã kết thúc (game_active==False), các hành động sau sẽ xảy ra:
Trò chơi sẽ bắt đầu lại (game_active = True).
Danh sách ống sẽ được xóa sạch (pipe_list.clear()).
Vị trí của con chim sẽ được đặt lại tại tâm màn hình (bird_rect.center = (100,384)).
Chuyển động của con chim sẽ được đặt về 0 (bird_movement = 0).
Điểm số cũng sẽ được đặt lại về 0 (score = 0).

```
    if event.type == spawnpipe:
        pipe_list.extend(create_pipe())
    if event.type == birdflap:
        if bird_index < 2:
            bird_index += 1
        else:
            bird_index =0
        bird, bird_rect = bird_animation()
```

if event.type == spawnpipe::
Điều kiện này kiểm tra xem loại sự kiện có liên quan đến việc tạo ra ống nước (tạo ra các cặp ống mới) hay không.
Nếu điều kiện này được thỏa mãn, pipe_list sẽ được mở rộng bằng cách tạo ra một cặp ống mới bằng hàm create_pipe().
if event.type == birdflap::
Điều kiện này kiểm tra xem loại sự kiện có liên quan đến việc chim vỗ cánh (thay đổi khung hình ảnh của nó) hay không.
Nếu điều kiện này được thỏa mãn:
Nếu bird_index nhỏ hơn 2 (nghĩa là chim không ở trong khung hình ảnh cuối cùng), nó sẽ tăng bird_index lên 1 để chuyển sang khung hình ảnh tiếp theo.
Ngược lại, nó sẽ đặt lại bird_index về 0 (để lặp lại khung hình ảnh đầu tiên).
Chim và bird_rect sau đó được cập nhật dựa trên khung hình ảnh mới.

```
if game_active:
    #chim
    bird_movement += gravity
    rotated_bird = rotate_bird(bird)
    bird_rect.centery += bird_movement
    screen.blit(rotated_bird,bird_rect)
    game_active= check_collision(pipe_list)
    #ống
    pipe_list = move_pipe(pipe_list)
    draw_pipe(pipe_list)
    score += 0.01
    score_display('main game')
    score_sound_countdown -= 1
    if score_sound_countdown <= 0:
        score_sound.play()
        score_sound_countdown = 100
```

Chim:
bird_movement được cộng thêm giá trị của trọng lực (gravity).
rotated_bird là hình ảnh chim sau khi được xoay (sử dụng hàm rotate_bird(bird)).
Vị trí dọc của chim (bird_rect.centery) được cập nhật bằng cách thêm bird_movement.
Chim được vẽ lên màn hình bằng screen.blit(rotated_bird, bird_rect).
game_active được kiểm tra bằng hàm check_collision(pipe_list).
Ống:
Danh sách ống (pipe_list) được di chuyển bằng hàm move_pipe(pipe_list).
Ống được vẽ lên màn hình bằng hàm draw_pipe(pipe_list).
Điểm số (score) tăng thêm 0.01.
Hiển thị điểm số lên màn hình bằng hàm score_display('main game').
Đếm ngược để phát âm thanh điểm số (score_sound) giảm xuống (để tránh phát âm thanh quá nhanh).

```
else:
    screen.blit(game_over_surface,game_over_rect)
    high_score = update_score(score,high_score)
    score_display('game_over')
```

Trong trường hợp trò chơi đã kết thúc (else), mã nguồn thực hiện các hành động sau:
Hiển thị màn hình kết thúc trò chơi bằng hình ảnh game_over_surface tại vị trí game_over_rect.
Cập nhật điểm số cao nhất (high_score) bằng cách gọi hàm update_score(score, high_score).
Hiển thị điểm số lên màn hình bằng hàm score_display('game_over').

```
    floor_x_pos -= 1
    draw_floor()
    if floor_x_pos <= -432:
        floor_x_pos =0

    pygame.display.update()
    clock.tick(120)
```

Dòng mã floor_x_pos -= 1 giảm giá trị của floor_x_pos đi 1 đơn vị. Sau đó, hàm draw_floor() được gọi để vẽ lại sàn (hoặc nền) của trò chơi. Nếu floor_x_pos nhỏ hơn hoặc bằng -432, giá trị của floor_x_pos được đặt lại về 0. Cuối cùng, pygame.display.update() cập nhật màn hình và clock.tick(120) giới hạn tốc độ khung hình của trò chơi là 120 khung hình mỗi giây.

3.3 Tạo hàm cho trò chơi

```
def draw_floor():
    screen.blit(floor,(floor_x_pos,650))
    screen.blit(floor,(floor_x_pos+432,650))
```

screen.blit(floor, (floor_x_pos, 650)): Dòng này vẽ hình ảnh floor lên màn hình tại vị trí (floor_x_pos, 650). Biến floor_x_pos xác định vị trí ngang của sàn. Bằng cách cập nhật floor_x_pos, bạn có thể tạo ra ấn tượng về việc sàn di chuyển ngang.
screen.blit(floor, (floor_x_pos + 432, 650)): Tương tự, dòng này vẽ một bản sao khác của hình ảnh floor bên phải hình ảnh đầu tiên. Giá trị 432 đảm bảo rằng hai hình ảnh sàn kết nối mượt mà, tạo hiệu ứng cuộn liên tục.

```
def create_pipe():
    random_pipe_pos = random.choice(pipe_height)
    bottom_pipe = pipe_surface.get_rect(midtop =(500,random_pipe_pos))
    top_pipe = pipe_surface.get_rect(midtop =(500,random_pipe_pos-650))
    return bottom_pipe, top_pipe
```

random_pipe_pos = random.choice(pipe_height): Dòng này chọn ngẫu nhiên một vị trí dọc cho ống nước từ danh sách pipe_height.
bottom_pipe = pipe_surface.get_rect(midtop=(500, random_pipe_pos)): Đoạn mã này tạo một hình chữ nhật (rect) cho ống dưới (bottom pipe) với vị trí trung tâm ở (500, random_pipe_pos).
top_pipe = pipe_surface.get_rect(midtop=(500, random_pipe_pos - 650)): Tương tự, đoạn mã này tạo một hình chữ nhật cho ống trên (top pipe) với vị trí trung tâm ở (500, random_pipe_pos - 650).

```
def move_pipe(pipes):
    for pipe in pipes :
        pipe.centerx -= 5
    return pipes
```

Vòng lặp for pipe in pipes: duyệt qua từng ống trong danh sách pipes.
Dòng pipe.centerx -= 5 giảm giá trị của thuộc tính centerx của ống đi 5 đơn vị. Điều này di chuyển ống sang trái.
Cuối cùng, hàm trả về danh sách ống đã được di chuyển.

```
def draw_pipe(pipes):
    for pipe in pipes:
        if pipe.bottom >= 600 :
            screen.blit(pipe_surface,pipe)
        else:
            flip_pipe = pygame.transform.flip(pipe_surface,False,True)
            screen.blit(flip_pipe,pipe)
```

Vòng lặp for pipe in pipes: duyệt qua từng ống trong danh sách pipes.
Nếu pipe.bottom (đáy của ống) lớn hơn hoặc bằng 600, nghĩa là ống dưới, thì hình ảnh pipe_surface được vẽ lên màn hình tại vị trí của ống.
Ngược lại, nếu pipe.bottom nhỏ hơn 600, nghĩa là ống trên, thì hình ảnh pipe_surface được lật ngược theo chiều dọc và vẽ lên màn hình tại vị trí của ống.

```
def check_collision(pipes):
    for pipe in pipes:
        if bird_rect.colliderect(pipe):
            hit_sound.play()
            return False
    if bird_rect.top <= -75 or bird_rect.bottom >= 650:
            return False
    return True
```

Vòng lặp for pipe in pipes: duyệt qua từng ống trong danh sách pipes.
Dòng if bird_rect.colliderect(pipe): kiểm tra xem hình chữ nhật của chim (bird_rect) có va chạm với hình chữ nhật của ống (pipe) hay không. Nếu có va chạm, âm thanh va chạm (hit_sound) được phát và hàm trả về False.
Dòng if bird_rect.top <= -75 or bird_rect.bottom >= 650: kiểm tra xem chim có bay quá biên trên hoặc dưới màn hình không. Nếu có, hàm cũng trả về False.
Nếu không có va chạm và chim không bay quá biên, hàm trả về True.

```
def rotate_bird(bird1):
    new_bird = pygame.transform.rotozoom(bird1,-bird_movement*3,1)
    return new_bird
```

pygame.transform.rotozoom(bird1, -bird_movement * 3, 1): Dòng này sử dụng hàm rotozoom để xoay hình ảnh bird1 một góc -bird_movement * 3 độ và tỷ lệ zoom là 1. Kết quả là hình ảnh chim sau khi được xoay.

```
def bird_animation():
    new_bird = bird_list[bird_index]
    new_bird_rect = new_bird.get_rect(center = (100,bird_rect.centery))
    return new_bird, new_bird_rect
def score_display(game_state):
```

new_bird = bird_list[bird_index]: Dòng này lấy hình ảnh của con chim từ danh sách bird_list dựa trên chỉ số bird_index. Chỉ số này thay đổi để chọn các khung hình ảnh khác nhau cho chim (ví dụ: khi chim vỗ cánh).
new_bird_rect = new_bird.get_rect(center=(100, bird_rect.centery)): Dòng này tạo một hình chữ nhật (rect) cho con chim với vị trí trung tâm ở (100, bird_rect.centery).

```python
def score_display(game_state):
    if game_state == 'main game':
        score_surface = game_font.render(str(int(score)),True,(255,255,255))
        score_rect = score_surface.get_rect(center = (216,100))
        screen.blit(score_surface,score_rect)
    if game_state == 'game_over':
        score_surface = game_font.render(f'Score: {int(score)}',True,(255,255,255))
        score_rect = score_surface.get_rect(center = (216,100))
        screen.blit(score_surface,score_rect)

        high_score_surface = game_font.render(f'High Score: {int(high_score)}',True,(255,255,255))
        high_score_rect = high_score_surface.get_rect(center = (216,630))
        screen.blit(high_score_surface,high_score_rect)
```

(variable) high_score: int

Trạng thái “main game”:
Nếu game_state là 'main game', dòng mã sau sẽ được thực hiện:
Tạo hình ảnh điểm số (score_surface) từ giá trị điểm số hiện tại (được chuyển thành số nguyên) và màu trắng.
Xác định vị trí của hình ảnh điểm số (score_rect) ở trung tâm màn hình theo trục ngang.
Vẽ hình ảnh điểm số lên màn hình tại vị trí đã xác định.
Trạng thái “game_over”:
Nếu game_state là 'game_over', dòng mã sau sẽ được thực hiện:
Tạo hình ảnh điểm số (score_surface) với chuỗi "Score: " và giá trị điểm số hiện tại (được chuyển thành số nguyên) và màu trắng.
Xác định vị trí của hình ảnh điểm số (score_rect) ở trung tâm màn hình theo trục ngang.
Vẽ hình ảnh điểm số lên màn hình tại vị trí đã xác định.
Tương tự, tạo hình ảnh điểm số cao nhất (high_score_surface) với chuỗi "High Score: " và giá trị điểm số cao nhất (được chuyển thành số nguyên) và màu trắng.

```python
def update_score(score,high_score):
    if score > high_score:
        high_score = score
    return high_score
```

Nếu score (điểm số hiện tại) lớn hơn high_score (điểm số cao nhất), dòng mã high_score = score sẽ được thực hiện. Điều này cập nhật giá trị của high_score thành điểm số hiện tại.
Cuối cùng, hàm trả về giá trị mới của high_score.

4. Kết luận

Sau một thời gian tìm hiểu đồ án, thu thập các kiến thức liên quan và tham khảo cách làm một số nơi trên Internet, em đã hoàn thành đồ án game Flabby Pird. Mặc dù rất cố gắng, nhưng vẫn không tránh khỏi thiếu sót và hạn chế. Em rất mong có được những ý kiến đánh giá, đóng góp của thầy để đồ án thêm hoàn thiện.