**Khai báo thư viện**

import pygame, sys, random

from pygame import mixer

Khai báo các thư viện cần thiết để cài đặt

* Thư viện pygame để cài đặt game
* Thư viện sys để thực hiện các tác vụ liên quan đến hệ thống như thoát app...
* Thư viện random để các ống sinh ra ngẫu nhiên
* Thư viện mixer để dùng các đoạn âm thanh trong game

**Khởi tạo màn hình game**

#Cửa sổ game

screen = pygame.display.set\_mode((346, 614))

#Tiêu đề game

pygame.display.set\_caption('Flappy Bird')

#clock

clock = pygame.time.Clock()

**Khởi tạo các âm thanh dùng trong game**

#Các âm thanh game

mixer.init()

wing = pygame.mixer.Sound("flappy-bird-assets-master/audio/wing.wav" )

point = pygame.mixer.Sound("flappy-bird-assets-master/audio/point.wav")

die = pygame.mixer.Sound("flappy-bird-assets-master/audio/die.wav")

hit = pygame.mixer.Sound("flappy-bird-assets-master/audio/hit.wav")

move\_pointer = pygame.mixer.Sound("flappy-bird-assets-master/audio/coin.wav")

**Khởi tạo các biến của game**

#Các biến trạng thái của game

game\_play = True

choose\_skin = False

show\_start\_screen = True

show\_choose\_skin\_screen = False

is\_day = True

time\_counter = 0

start\_time = pygame.time.get\_ticks()

#trọng lực

p = 0.15

#Điểm của game

score = 0   #Khởi tạo điểm của game

high\_score = 0  #Khởi tạo điểm cao nhất

game\_font = pygame.font.Font(r'flappy-bird-assets-master\04B\_19\_\_.TTF', 40)

* game\_play: Thể hiện game đang được chơi
* choose\_skin: Thể hiện đã chọn skin hay chưa
* show\_start\_screen: Có đang hiện màn hình bắt đầu không
* show\_choose\_skin\_screen: Có đang hiện màn hình chọn skin không
* is\_day: Có đang là ban ngày hay không (dùng để thay đổi background)
* time\_counter: Biến đếm thời gian
* start\_game: Bắt đầu đếm từ khi mở game
* p: Trọng lực của game, giá trị mặc định 0.15
* score: Điểm của người chơi
* high\_score: Điểm cao nhất
* game\_font: Phông chữ của game

**Khởi tạo icon và background ban đầu cho game**

#Icon game

icon = pygame.image.load(r"flappy-bird-assets-master\favicon.ico").convert()

pygame.display.set\_icon(icon)

#Background game

background = pygame.image.load(r"flappy-bird-assets-master\sprites\background-day.png").convert()

#Chỉnh background size

background = pygame.transform.scale(background, (346, 614))

**Khởi tạo sàn của game**

#Floor game

floor = pygame.image.load(r"flappy-bird-assets-master\sprites\base.png").convert()

#Chỉnh floor size lớn hơn

floor = pygame.transform.scale(floor, (346, 134))

floor\_x = 0 #Biến x của floor

#Hàm vẽ floor

def draw\_floor():

    global floor\_x

    floor\_x -= 1    #Biến x của floor giảm dần -> chạy sang trái

    screen.blit(floor, (floor\_x, 614 - 134))

    screen.blit(floor, (floor\_x + 346, 614 - 134))  #Thêm 1 floor nối tiếp cái ban đầu

    if floor\_x == -346:     #Nếu floor đầu chạy hết (x của floor đầu ban đầu bằng 0) thì reset floor\_x

        floor\_x = 0

Giải thích hàm draw\_floor():

* global floor\_x: Biến này lưu trữ vị trí hiện tại của floor trên trục hoành (theo đơn vị pixel).
* floor\_x -= 1: Biến floor\_x sẽ giảm đi 1 đơn vị pixel mỗi khi hàm được gọi, tức là floor sẽ di chuyển sang trái.
* screen.blit(floor, (floor\_x, 614 - 134)): Dòng code này sẽ vẽ floor lên màn hình game ở vị trí (floor\_x, 614 - 134). 614 là chiều cao của màn hình game và 134 là chiều cao của hình ảnh floor. Vị trí này sẽ được cập nhật mỗi khi hàm được gọi.
* screen.blit(floor, (floor\_x + 346, 614 - 134)): Dòng code này sẽ vẽ thêm một floor nữa nối tiếp floor ban đầu để tạo ra hiệu ứng di chuyển vô hạn. Vị trí của floor này được tính bằng cách cộng thêm 346 vào giá trị hiện tại của floor\_x, vì chiều dài của floor là 346 pixel.
* if floor\_x == -346:: Nếu floor đã di chuyển sang trái đến vị trí -346 pixel, nghĩa là floor ban đầu đã di chuyển hết ra khỏi màn hình, thì biến floor\_x sẽ được đặt lại về 0. Điều này sẽ giúp tạo ra hiệu ứng di chuyển vô hạn của floor trong game.

**Khởi tạo chim của game:**

#Chim của game

bird\_mid = pygame.image.load(r"flappy-bird-assets-master\sprites\bluebird-midflap.png").convert\_alpha()

bird\_up = pygame.image.load(r"flappy-bird-assets-master\sprites\bluebird-upflap.png").convert\_alpha()

bird\_down = pygame.image.load(r"flappy-bird-assets-master\sprites\bluebird-downflap.png").convert\_alpha()

bird\_mid = pygame.transform.scale\_by(bird\_mid, 1.2)

bird\_up = pygame.transform.scale\_by(bird\_up, 1.2)

bird\_down = pygame.transform.scale\_by(bird\_down, 1.2)

bird\_list = [bird\_down, bird\_mid, bird\_down]

bird\_index = 0

bird = bird\_list[bird\_index]

bird\_rect = bird.get\_rect(center = (50, (614 - 134)/2))

bird\_y = 0  #Biến y của chim

Tạo ra 3 trạng thái của chim khi đập cánh là bird\_up, bird\_mid, bird\_down. Tạo 1 bird\_list chứa cả 3 trạng thái, biến bird\_index để truy cập đến các trạng thái, ban đầu sẽ là 0. Cuối cùng là tạo rect của bird và 1 biến bird\_y để truy cập đến tọa độ trục y của bird.

**Hàm tạo hiệu ứng xoay chim**

def rotate\_bird(bird):

    new\_bird = pygame.transform.rotozoom(bird, bird\_y \* 3, 1)

    return new\_bird

Giải thích hàm rotate\_bird():

* bird: Tham số đầu vào của hàm, là một hình ảnh của bird.
* pygame.transform.rotozoom(): Phương thức này được cung cấp bởi thư viện Pygame và có tác dụng xoay và thay đổi kích thước hình ảnh. Hàm nhận đầu vào là 3 tham số:
* bird: Hình ảnh bird ban đầu.
* bird\_y \* 3: Góc quay của bird, được tính bằng cách nhân giá trị hiện tại của biến bird\_y với 3. Điều này giúp tạo ra hiệu ứng bird càng nhấp càng xoay nhiều hơn.
* 1: Tỉ lệ thay đổi kích thước của hình ảnh bird sau khi xoay. Trong trường hợp này, tỉ lệ là 1, nghĩa là không có thay đổi kích thước.
* new\_bird: Biến new\_bird là hình ảnh bird sau khi được xoay và thay đổi kích thước.
* return new\_bird: Hàm sẽ trả về biến new\_bird, là hình ảnh bird sau khi được xoay và thay đổi kích thước.
* Tổng quát, hàm rotate\_bird() được sử dụng để tạo ra hiệu ứng xoay hình ảnh bird trong game

**Tạo hiệu ứng đập cánh:**

#Taọ timer cho chim

bird\_flap = pygame.USEREVENT + 1

pygame.time.set\_timer(bird\_flap, 200)

def bird\_animation():

    new\_bird = bird\_list[bird\_index]

    new\_bird\_rect = new\_bird.get\_rect(center = (50, bird\_rect.centery))

    return new\_bird, new\_bird\_rect

Tạo 1 timer cho chim đập cánh sau mỗi 200ms

Giải thích hàm bird\_animation():

* bird\_list: Là một danh sách chứa các hình ảnh của bird. Mỗi phần tử trong danh sách là một hình ảnh bird ở trạng thái khác nhau để tạo ra hiệu ứng đập cánh.
* bird\_index: Là biến lưu trữ chỉ số của hình ảnh bird hiện tại trong danh sách bird\_list. Biến này sẽ được tăng giá trị lên mỗi lần hàm bird\_animation() được gọi để thay đổi hình ảnh bird.
* new\_bird: Biến này lưu trữ hình ảnh bird mới sau khi thay đổi hình ảnh. Giá trị của biến này được lấy từ danh sách bird\_list thông qua chỉ số bird\_index.
* new\_bird\_rect: Biến này là một rect được tạo ra từ new\_bird bằng cách sử dụng phương thức get\_rect() của hình ảnh. Biến này được đặt ở vị trí giữa màn hình (tọa độ x = 50) và giữ nguyên tọa độ y của hình ảnh bird hiện tại (được lấy từ biến bird\_rect.centery).
* return new\_bird, new\_bird\_rect: Hàm sẽ trả về hai biến new\_bird và new\_bird\_rect. Biến new\_bird là hình ảnh bird mới sau khi thay đổi, biến new\_bird\_rect là rect được tạo ra từ hình ảnh bird mới này để vẽ lên màn hình.
* Tổng quát, hàm bird\_animation() được sử dụng để thay đổi hình ảnh của bird trong game để tạo ra hiệu ứng đập cánh.

**Khởi tạo màn hình chọn skin**

#Khởi tạo các đối tượng màn hình chọn skin:

red\_bird = pygame.image.load(r"flappy-bird-assets-master\sprites\redbird-midflap.png").convert\_alpha()

red\_bird = pygame.transform.scale\_by(red\_bird, 1.2)

redbird\_rect = red\_bird.get\_rect(center = (screen.get\_rect().centerx, screen.get\_rect().centery))

blue\_bird = pygame.image.load(r"flappy-bird-assets-master\sprites\bluebird-midflap.png").convert\_alpha()

blue\_bird = pygame.transform.scale\_by(blue\_bird, 1.2)

bluebird\_rect = blue\_bird.get\_rect(center = (redbird\_rect.centerx - 100, screen.get\_rect().centery))

yellow\_bird = pygame.image.load(r"flappy-bird-assets-master\sprites\yellowbird-midflap.png").convert\_alpha()

yellow\_bird = pygame.transform.scale\_by(yellow\_bird, 1.2)

yellowbird\_rect = yellow\_bird.get\_rect(center = (redbird\_rect.centerx + 100, screen.get\_rect().centery))

pointer = pygame.image.load(r"flappy-bird-assets-master\sprites\pointer.png").convert\_alpha()

pointer = pygame.transform.scale(pointer, (40,40))

pointer\_rect = pointer.get\_rect(center = (screen.get\_rect().centerx + 5, screen.get\_rect().centery + 40))

Khởi tạo màn hình chọn skin gồm các đối tượng red\_bird, blue\_bird, yellow\_bird và pointer

**Hàm vẽ màn hình chọn skin**

def draw\_choose\_skin\_screen():

    screen.blit(red\_bird, redbird\_rect)

    screen.blit(blue\_bird, bluebird\_rect)

    screen.blit(yellow\_bird, yellowbird\_rect)

    screen.blit(pointer, pointer\_rect)

    text\_surface = game\_font.render('Choose skin', True, (255, 255, 255))

    text\_rect = text\_surface.get\_rect(center = ((screen.get\_rect().centerx), 200))

    screen.blit(text\_surface, text\_rect)

Vẽ các đối tượng của màn hình chọn skin lên màn hình

**Khởi tạo đối tượng ống**

pipe\_surface = pygame.image.load(r"flappy-bird-assets-master\sprites\pipe-green.png")

pipe\_surface = pygame.transform.scale\_by(pipe\_surface, 1.2)

pipe\_list = []

pipe\_height = [200, 250, 300, 350, 400]

Khởi tạo hình ảnh của ống, khai báo pipe\_list lưu các ống và pipe\_height lưu chiều cao có thể của các ống.

**Hàm tạo ống**

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 - 550))

    return bottom\_pipe, top\_pipe

Random chiều dài của ống từ pipe\_height, sau đó tạo rect của ống trên và ống dưới rồi trả về.

**Hàm di chuyển ống**

def move\_pipe(pipes):

    for pipe in pipes:

        pipe.centerx -= 2

    return pipes

Các ống sẽ di chuyển sang trái 2 pixel mỗi khi hàm này được gọi.

**Hàm vẽ ống**

#Hàm vẽ ống

def draw\_pipe(pipes):

    for pipe in pipes:

        if pipe.bottom >= 614 - 134:

            screen.blit(pipe\_surface, pipe)

        else:

            flip\_pipe = pygame.transform.flip(pipe\_surface, False, True)

            screen.blit(flip\_pipe, pipe)

Vẽ ống vào màn hình. Nếu là ống dưới thì sẽ dùng hàm flip để xoay ảnh ngược lại

**Tạo timer cho ống**

#Tạo timer

spawn\_pipe = pygame.USEREVENT

pygame.time.set\_timer(spawn\_pipe, 1000)

Timer này để hiện thêm ống sau mỗi 1 giây.

**Khởi tạo màn hình bắt đầu và màn hình kết thúc game**

#Màn hình kết thúc game

screen\_over = pygame.image.load(r'flappy-bird-assets-master\sprites\gameover.png')

screen\_over = pygame.transform.scale\_by(screen\_over, 1.2)

screen\_over\_rect = screen\_over.get\_rect(center = (screen.get\_rect().center))

#Màn hình bắt đầu game

screen\_start = pygame.image.load(r'flappy-bird-assets-master\sprites\message.png')

screen\_start = pygame.transform.scale\_by(screen\_start, 1.2)

screen\_start\_rect = screen\_start.get\_rect(center = (screen.get\_rect().center))

**Hàm hiển thị điểm**

#Hàm hiện điểm

def score\_view(game\_state):

    if game\_state:

        score\_surface = game\_font.render(f'Score: {int(score)}', True, (255, 255, 255))

        score\_rect = score\_surface.get\_rect(center = ((screen.get\_rect().centerx), 100))

        screen.blit(score\_surface, score\_rect)

    else:

        score\_surface = game\_font.render(f'Score: {int(score)}', True, (255, 255, 255))

        score\_rect = score\_surface.get\_rect(center = ((screen.get\_rect().centerx), 100))

        screen.blit(score\_surface, score\_rect)

        #Điểm cao nhất

        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 = ((screen.get\_rect().centerx), 200 ))

        screen.blit(high\_score\_surface, high\_score\_rect)

Giải thích hàm score\_view(): Truyền trạng thái chơi game vào hàm, nếu game đang chơi thì chỉ hiện điểm của người chơi, nếu game đã kết thúc thì hiển thị thêm điểm cao nhất

**Hàm đọc, ghi điểm cao nhất vào file text**

#Đọc điểm cao nhất từ file text

def read\_high\_score(file\_name):

    f = open(file\_name, 'r')

    global high\_score

    high\_score = float(f.read())

    f.close()

#Ghi điểm từ file text

def write\_high\_score(file\_name):

    f = open(file\_name, 'w')

    global high\_score

    f.write(str(high\_score))

    f.close()

2 hàm này sẽ lần lượt đọc điểm cao nhất từ file text lúc mới vào game và lưu điểm cao nhất vào file text mỗi khi người chơi vượt điểm cao nhất.

**Hàm xử lý va chạm**

#Hàm xử lý va chạm

def check\_collision(pipes):

    if (bird\_rect.bottom >= 614 - 134) or (bird\_rect.top <= 0):

        pygame.mixer.Sound.play(hit)

        return False

    for pipe in pipes:

        if bird\_rect.colliderect(pipe):

            pygame.mixer.Sound.play(hit)

            return False

    return True

Hàm này sẽ trả về true nếu chim va chạm với ống hoặc sàn, hoặc trên cùng màn hình, cùng lúc đó chạy âm thanh bị đụng trúng.

**Vòng lặp while xử lý game**

while True:

    read\_high\_score('high\_score.txt')

Khi mới vào vòng lặp đầu tiên sẽ đọc điểm từ file high\_score.txt.

**Vòng for xử lý các event**

for event in pygame.event.get():    #Event thoát game

        if event.type == pygame.QUIT:

            pygame.quit()

            sys.exit()

Thoát game khi người dùng tắt

if event.type == spawn\_pipe and choose\_skin and game\_play:    #Cho ống xuất hiện

            pipe\_list.extend(create\_pipe())

Nếu đã chọn skin và đang chơi thì dùng timer spawn\_pipe đã tạo, mỗi 1 giây sẽ tạo thêm ống vào pipe\_list.

if event.type == bird\_flap and game\_play:   #Chim đập cánh

            if bird\_index < 2:

                bird\_index += 1

            else:

                bird\_index = 0

            bird, bird\_rect = bird\_animation()

Nếu game đang chơi thì dùng timer bird\_flap đã tạo, lần lượt thay đổi bird index và dùng hàm bird\_animation() để tạo hiệu ứng chim đập cánh sau mỗi 200ms.

**Các event liên quan đến phím**

if event.type == pygame.KEYDOWN:

            if event.key == pygame.K\_SPACE and show\_start\_screen:

                screen\_start\_rect.centerx = 1000

                screen\_start\_rect.centery = 1000

                show\_start\_screen = False

Nếu đang hiện màn hình bắt đầu game mà người dùng nhấn phím cách thì màn hình bắt đầu sẽ biến mất, set giá trị cho biến show\_start\_screen về False.

if event.key == pygame.K\_SPACE and game\_play and choose\_skin:

                bird\_y = -5

                pygame.mixer.Sound.play(wing)

Nếu đã chọn skin xong và đang chơi mà nhấn phím cách thì tọa độ y của chim sẽ giảm đi, từ đó tạo hiệu ứng chim bay lên, cùng lúc đó chạy âm thanh đập cánh.

if event.key == pygame.K\_SPACE and game\_play == False:  #Cho phép chơi lại

                game\_play = True

                pipe\_list.clear()

                bird\_y = 0

                bird\_rect.center = (50, (614 - 134)/2)

                score = 0

Nếu game đã kết thúc mà nhấn phím cách sẽ cho phép chơi lại. Biến game\_play sẽ thành True, list các ống sẽ được clear, điểm sẽ về lại 0 và chim sẽ về lại vị trí ban đầu.

if event.key == pygame.K\_RIGHT and choose\_skin == False and show\_start\_screen == False and show\_choose\_skin\_screen:

                pygame.mixer.Sound.play(move\_pointer)

                if pointer\_rect.centerx != yellowbird\_rect.centerx + 5:

                    if pointer\_rect.centerx == redbird\_rect.centerx + 5:

                        pointer\_rect.centerx = yellowbird\_rect.centerx + 5

                    else: pointer\_rect.centerx = redbird\_rect.centerx + 5

Nếu đang ở màn hình chọn skin và chưa chọn skin mà nhấn phím phải thì pointer sẽ chỉ đến chú chim tiếp theo, cùng lúc đó chạy âm thanh di chuyển con trỏ.

 if event.key == pygame.K\_LEFT and choose\_skin == False and show\_start\_screen == False and show\_choose\_skin\_screen:

                pygame.mixer.Sound.play(move\_pointer)

                if pointer\_rect.centerx != bluebird\_rect.centerx + 5:

                    if pointer\_rect.centerx == redbird\_rect.centerx + 5:

                        pointer\_rect.centerx = bluebird\_rect.centerx + 5

                    else: pointer\_rect.centerx = redbird\_rect.centerx + 5

Tương tự như trên, di chuyển con trỏ về bên trái

if event.key == pygame.K\_SPACE and show\_choose\_skin\_screen:

                if pointer\_rect.centerx == bluebird\_rect.centerx + 5:

                    bird\_mid = pygame.image.load(r"flappy-bird-assets-master\sprites\bluebird-midflap.png").convert\_alpha()

                    bird\_up = pygame.image.load(r"flappy-bird-assets-master\sprites\bluebird-upflap.png").convert\_alpha()

                    bird\_down = pygame.image.load(r"flappy-bird-assets-master\sprites\bluebird-downflap.png").convert\_alpha()

                    choose\_skin = True

                    show\_choose\_skin\_screen = False

                if pointer\_rect.centerx == redbird\_rect.centerx + 5:

                    bird\_mid = pygame.image.load(r"flappy-bird-assets-master\sprites\redbird-midflap.png").convert\_alpha()

                    bird\_up = pygame.image.load(r"flappy-bird-assets-master\sprites\redbird-upflap.png").convert\_alpha()

                    bird\_down = pygame.image.load(r"flappy-bird-assets-master\sprites\redbird-downflap.png").convert\_alpha()

                    choose\_skin = True

                    show\_choose\_skin\_screen = False

                if pointer\_rect.centerx == yellowbird\_rect.centerx + 5:

                    bird\_mid = pygame.image.load(r"flappy-bird-assets-master\sprites\yellowbird-midflap.png").convert\_alpha()

                    bird\_up = pygame.image.load(r"flappy-bird-assets-master\sprites\yellowbird-upflap.png").convert\_alpha()

                    bird\_down = pygame.image.load(r"flappy-bird-assets-master\sprites\yellowbird-downflap.png").convert\_alpha()

                    choose\_skin = True

                    show\_choose\_skin\_screen = False

Nếu đang ở màn hình chọn skin mà nhấn phím cách thì sẽ dựa vào vị trí pointer mà thay đổi hình ảnh chim tương ứng, biến choose\_skin sẽ thành True và show\_choose\_skin\_screen sẽ về False.

**Các xử lý bên ngoài vòng for**

 # Cập nhật thời gian

    time\_counter += pygame.time.get\_ticks() - start\_time

    start\_time = pygame.time.get\_ticks()

Biến time\_counter sẽ tăng dần ứng với thời gian bắt đầu game

# Thay đổi background nếu đã đủ thời gian

    if time\_counter >= 20000:

        if is\_day:

            background = pygame.image.load(r"flappy-bird-assets-master\sprites\background-night.png").convert()

        else:

            background = pygame.image.load(r"flappy-bird-assets-master\sprites\background-day.png").convert()

        background = pygame.transform.scale(background, (346, 614))

        is\_day = False

        time\_counter = 0

Nếu đã đủ 20 giây, background sẽ thay đổi hình ảnh ban ngày hoặc ban đêm dựa vào biến is\_day, sau đó cập nhật biến is\_day và set lại time\_counter tiếp tục đếm 20 giây tiếp theo.

#Thêm background vào game

    screen.blit(background, (0, 0))

    #Thêm sàn

    draw\_floor()

    screen.blit(screen\_start, screen\_start\_rect)

Thêm background và floor vào màn hình

if choose\_skin == False and show\_start\_screen == False:

        draw\_choose\_skin\_screen()

        show\_choose\_skin\_screen = True

Nếu chưa chọn skin và màn hình chọn skin chưa hiển thị thì hiển thị màn hình chọn skin, biến show\_choose\_skin\_screen sẽ thành True.

if game\_play and show\_start\_screen == False and choose\_skin:

Các xử lý khi game đang chơi và đã chọn skin xong:

#Thêm chim vào game:

        rotated\_bird = rotate\_bird(bird)

        screen.blit(rotated\_bird, bird\_rect)

        bird\_y += p #y của chim tăng theo p -> bị rớt xuống

        bird\_rect.centery += bird\_y

        #Ống

        pipe\_list = move\_pipe(pipe\_list)

        draw\_pipe(pipe\_list)\

        #sàn

        draw\_floor()

#Thêm score vào màn hình

        score\_view(game\_play)

Vẽ chim, ống và sàn, điểm vào game. Phải vẽ lại sàn sau khi thêm ống để ống sẽ không đè hình ảnh của sàn.

#Xử lý ghi điểm

        bird\_mid\_pos = bird\_rect.x + bird\_rect.width / 2

        for pipe in pipe\_list:

            pipe\_mid\_pos = pipe.x + pipe.width / 2

            if pipe\_mid\_pos <=  bird\_mid\_pos <=  pipe\_mid\_pos:

                score += 0.5

                pygame.mixer.Sound.play(point)

Xử lý mỗi khi chim bay qua 1 ống sẽ ghi được 1 điểm

#Cập nhật điểm cao nhất

        if score > high\_score:

            high\_score = score

            write\_high\_score('high\_score.txt')

Mỗi khi điểm cao hơn điểm cao nhất thì sẽ cập nhật lại điểm cao nhất và ghi điểm cao nhất vào file high\_score.txt

game\_play = check\_collision(pipe\_list)

Biến game\_play sẽ cập nhật lại nếu chim bị va chạm

elif game\_play == False and show\_start\_screen == False and choose\_skin:

        screen.blit(screen\_over, screen\_over\_rect)

        score\_view(game\_play)

        screen\_start\_rect.centerx = 1000

        screen\_start\_rect.centery = 1000

Nếu game kết thúc thì sẽ vẽ màn hình kết thúc game và hiện điểm của người chơi.

pygame.display.update()

    #clock 120

    clock.tick(120)

Cuối cùng update display của pygame và đặt clock là 120.