TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHIỆP

KHOA ĐIỆN TỬ

Bài tập lớn lập trình Python

NGÀNH : KỸ THUẬT MÁY TÍNH

HỆ : ĐẠI HỌC CHÍNH QUY

ĐỀ TÀI: PIZZA PANIC VỚI PYGAME

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

Sinh viên: Hoàng Thị Quyến

Lớp : K58KTP.01

Giáo viên giảng dạy: Ts. Nguyễn Văn Huy

Link GitHub: hoangquyencode/BTCM_Python

Video: https://www.youtube.com/watch?v=uaM3QYh4KsQ

Thái Nguyên – 2025

TRƯỜNG ĐHKTCN
KHOA ĐIỆN TỬ

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
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: Hoàng Thị Quyến

Lớp: K58KTP

Ngành: Kĩ thuật máy tính

Giáo viên hướng dẫn: Ts.Nguyễn Văn Huy

Ngày giao đề: *20/5/ 2025*

Ngày hoàn thành: 10/6/2025

Tên đề tài : Pizza Panic với pygame

Yêu cầu : Triển khai game Pizza Panic (Chapter 11) dùng pygame/ livewires: điều khiển cái chảo hứng pizza rơi.

GIÁO VIÊN HƯỚNG DẪN

(Ký và ghi rõ họ tên)

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

Thái Nguyên, ngày….tháng…..năm 20....

GIÁO VIÊN HƯỚNG DẪN

(Ký ghi rõ họ tên)

MỤC LỤC

Chương 1. Giới thiệu đầu bài

1.1 Đề bài được giao

Bài 7. Pizza Panic với pygame

Đầu bài: *"Triển khai game Pizza Panic (Chapter 11) dùng pygame/ livewires: điều khiển cái chảo hứng pizza rơi."*

Đầu vào – đầu ra:

- Đầu vào: Chuột di chuyển pan.
- Đầu ra: Sprite pizza rơi, điểm số, game over khi mất quá 3 pizza.

Tính năng yêu cầu:

- Sử dụng Sprite, games.screen, hình nền.
- Quản lý collision giữa pizza và pan.
- Tăng tốc độ theo thời gian.

Kiểm tra & kết quả mẫu:

- Hứng 5 pizza → +5 điểm.
- Bỏ 3 lần → "Game Over".

Các bước triển khai:

1. Khởi tạo màn hình, background.
2. Class Pan và Pizza kế thừa Sprite.
3. Trong vòng lặp, update vị trí, kiểm tra va chạm.
4. Hiển thị điểm.

1.2 Tính năng của chương trình

Khởi tạo Pygame và âm thanh

- Khởi tạo thư viện pygame và bộ trộn âm thanh pygame.mixer.
- Tải và sử dụng ba âm thanh:
    - catch.wav: phát khi bắt được pizza.
    - miss.wav: phát khi bỏ lỡ pizza.
    - gameover.wav: phát khi kết thúc trò chơi.

Nhập tên người chơi

- Trước khi bắt đầu trò chơi, chương trình yêu cầu người dùng nhập tên qua lệnh input().

- Tên người chơi được dùng để ghi lại kết quả sau mỗi lượt chơi.

Cấu hình cửa sổ game

- Kích thước cửa sổ: 640 x 480 pixels.
- Tiêu đề cửa sổ: "Pizza Panic".

Tạo hình ảnh đối tượng game

- Chảo (Pan): hình elip màu đỏ, di chuyển theo chuột.
- Pizza: hình tròn màu vàng, với các điểm nâu tượng trưng cho pepperoni được vẽ ngẫu nhiên.

Lớp đối tượng game

- Pan: đại diện cho chảo, di chuyển theo trục ngang của chuột và bị giới hạn trong phạm vi màn hình.
- Pizza: rơi từ trên xuống, nếu không bắt được sẽ tăng biến đếm So_pizza_roi, phát âm miss, và bị loại khỏi màn hình.

Quản lý sprite

- all_sprites: nhóm chứa toàn bộ sprite (chảo và pizza).
- pizza_group: nhóm riêng chứa các pizza, dùng để kiểm tra va chạm với chảo.

Tạo pizza mới định kỳ

- Cứ sau 50 frame (~0.8 giây ở 60 FPS), một pizza mới được tạo và thêm vào nhóm sprite.

Kiểm tra va chạm

- Sử dụng pygame.sprite.spritecollide() để phát hiện va chạm giữa pizza và chảo.
- Nếu bắt được pizza:
    - Tăng điểm (Diemso).
    - Phát âm thanh catch.
    - Xóa pizza khỏi nhóm.

Tăng độ khó

- Tốc độ rơi của pizza (fall_speed) tăng dần theo điểm số.
- Cứ mỗi 5 điểm, tốc độ tăng thêm 1 đơn vị.

Hiển thị điểm số và số lần bỏ lỡ

- Điểm số và số pizza bị rơi được hiển thị ở góc trái phía trên màn hình.

Kết thúc game

- Khi số pizza bỏ lỡ đạt 3:
  - Phát âm thanh gameover.wav.
  - Hiển thị dòng chữ "GAME OVER" màu đỏ ở giữa màn hình.

Lưu kết quả vào file JSON

- Kết quả chơi được lưu vào file DLgame.json bao gồm:
  - Tên người chơi.
  - Điểm số đạt được.
  - Số pizza bị rơi.
  - Tổng số pizza xuất hiện.
  - Thời gian bắt đầu và kết thúc trò chơi.
- Nếu file đã tồn tại, chương trình sẽ đọc dữ liệu cũ và nối thêm lượt chơi mới vào danh sách.

Giới hạn tốc độ khung hình (FPS)

- Sử dụng pygame.time.Clock() để giới hạn trò chơi ở 60 FPS.
- Điều này giúp trò chơi chạy mượt, ổn định, không bị quá nhanh trên các máy cấu hình cao.

1.3 Thách thức

• Điều khiển chính xác chảo theo chuột: Cần xử lý vị trí của con trỏ chuột, đảm bảo chảo không ra khỏi giới hạn màn hình.

• Tạo đối tượng pizza ngẫu nhiên: Tạo pizza rơi liên tục và không gây lag, quản lý danh sách pizza hiệu quả.

• Xử lý va chạm chính xác: Kiểm tra va chạm giữa chảo (đối tượng có hình elip) và pizza (hình tròn) bằng cách dùng tính năng sprite collision của pygame.

• Tăng độ khó hợp lý: Cần tăng tốc độ rơi pizza dần dần sao cho game không quá dễ hoặc quá khó.

• Quản lý âm thanh đồng bộ: Phát âm thanh đúng lúc bắt pizza, bỏ lỡ hoặc game over mà không bị trễ hoặc bị lặp lại gây khó chịu.

• Hiển thị thông tin game rõ ràng: Điểm số, số pizza bị bỏ lỡ, và thông báo game over phải rõ ràng dễ nhìn.

• Quản lý vòng lặp game mượt mà: Giữ cho game chạy ở tốc độ 60 FPS ổn định.

1.4 Kiến thức cần có

Python và Pygame cơ bản:

- Khởi tạo màn hình, xử lý sự kiện.
- Vẽ hình (hình elip, hình tròn).
- Sử dụng sprite và sprite groups để quản lý đối tượng game.

Lập trình hướng đối tượng (OOP):

- Tạo lớp Pan và Pizza kế thừa pygame.sprite.Sprite.
- Sử dụng phương thức update() để cập nhật trạng thái đối tượng.

Xử lý va chạm:

- Dùng pygame.sprite.spritecollide() để kiểm tra va chạm giữa chảo và pizza.

Quản lý âm thanh:

- Tải và phát âm thanh bằng pygame.mixer.Sound().

Quản lý biến và điều kiện game:

- Biến điểm (score), biến số pizza bỏ lỡ (missed).
- Điều kiện dừng game khi bỏ lỡ 3 pizza.

Điều khiển thời gian và tốc độ:

- Dùng pygame.time.Clock() để giữ tốc độ khung hình.
- Tạo timer để sinh pizza định kỳ.

Hiển thị chữ:

- Sử dụng font của Pygame để vẽ điểm số và thông báo.

Đọc/ghi file JSON

- Dùng json.load() để đọc file kết quả cũ (nếu có).
- Dùng json.dump() để ghi kết quả game mới.
- Cần kiến thức về:
    - Thao tác với file (with open(...))
    - Xử lý dữ liệu dạng dict/list trong Python.

Xử lý chuỗi và nhập liệu từ người dùng

- Lấy tên người chơi bằng input().

- Gán tên người chơi vào dict lưu kết quả.
- Kỹ năng về xử lý chuỗi, nhập xuất cơ bản trong Python.

Tăng độ khó theo điểm số

- Điều chỉnh biến tốc độ rơi fall_speed theo điểm.
- Cần hiểu cách kết hợp điều kiện và biến trong quá trình cập nhật logic game.

Chương 2 Cơ sở lý thuyết

Trong chương trình "Pizza Panic", nhiều kiến thức chuyên môn trong lập trình Python và thư viện Pygame được áp dụng để xây dựng một trò chơi tương tác. Dưới đây là các nội dung lý thuyết và kỹ thuật chính được sử dụng.

2.1 Danh sách (List)

Khái niệm: Danh sách (list) trong Python là một cấu trúc dữ liệu cho phép lưu trữ nhiều giá trị trong cùng một biến.

Ứng dụng trong chương trình: Danh sách được dùng để lưu trữ các đối tượng pizza đang rơi. Mỗi khi một pizza mới được tạo ra, nó sẽ được thêm vào danh sách pizzas.

Ví dụ: pizzas = []

Lợi ích: Giúp quản lý nhiều đối tượng cùng lúc, dễ dàng cập nhật và loại bỏ pizza bị bắt hoặc rơi ra ngoài.

2.2 Lập trình hướng đối tượng (OOP)

Khái niệm: Lập trình hướng đối tượng là cách tổ chức chương trình bằng cách mô hình hóa các thành phần thành lớp (class) và đối tượng (object).

Đối tượng (Object) là những thực thể tồn tại có hành vi.

Ví dụ: Đối tượng là một xe ô tô có tên hãng, màu sắc, loại nguyên liệu, hành vi đi, dừng, đỗ, nổ máy...

2.3 Lớp (Class)

Khái niệm: Class là một kiểu dữ liệu đặc biệt do người dùng định nghĩa, tập hợp nhiều thuộc tính đặc trưng cho mọi đối tượng được tạo ra từ lớp đó.

Ứng dụng trong chương trình: Hai lớp chính được định nghĩa là

Pan: đại diện cho cái chảo.

Pizza: đại diện cho các pizza rơi.

Lợi ích: Giúp chia nhỏ chương trình, dễ bảo trì và mở rộng.

2.4 Sprite và Sprite Group trong Pygame

Khái niệm:Sprite là đối tượng đồ họa có thể di chuyển và tương tác. Sprite Group là tập hợp nhiều sprite giúp quản lý hiệu quả.

Ứng dụng:

- Quản lý tất cả các đối tượng trong game bằng all_sprites.

- Dễ dàng gọi .update() và .draw() cho tất cả các đối tượng.
- Kiểm tra va chạm giữa chảo và pizza bằng spritecollide().

Ví dụ:

```
hits = pygame.sprite.spritecollide(pan, pizza_group, True)
```

2.5 Câu lệnh điều kiện (if/else)

Khái niệm: Câu lệnh điều kiện cho phép kiểm tra các tình huống và quyết định luồng đi của chương trình.

Ứng dụng:

- Kiểm tra nếu bắt được pizza, nếu rơi quá 3 pizza thì kết thúc game.
- Tăng tốc độ rơi pizza khi điểm tăng.

2.6 Vòng lập (Loop)

Khái niệm: Vòng lặp giúp lặp lại khối lệnh nhiều lần liên tục.

Ứng dụng trong game:

- Vòng lặp chính while running: chạy suốt quá trình game hoạt động.
- Vòng lặp phụ for event in pygame.event.get(): xử lý các sự kiện như thoát game.

2.7 Hàm (function)

Khái niệm: Hàm là một khối mã thực hiện một chức năng cụ thể.

Ứng dụng:

- Dùng để vẽ chữ lên màn hình qua hàm draw_text():python
  def draw_text(surface, text, x, y, color=(0, 0, 0)):

Lợi ích: Giúp mã ngắn gọn, dễ đọc và tái sử dụng.

2.8 Xử lý âm thanh với pygame.mixer

Khái niệm: pygame.mixer là module trong pygame dùng để xử lý âm thanh.

Ứng dụng:

Phát âm thanh khi bắt pizza, bỏ lỡ hoặc kết thúc game.

Ví dụ:

```
catch_sound = pygame.mixer.Sound("catch.wav")
catch_sound.play()
```

2.10. Điều khiển thời gian và tốc độ game

Khái niệm:

pygame.time.Clock() được dùng để điều khiển tốc độ khung hình (FPS).

Ứng dụng:

- Giúp game chạy mượt, ổn định ở 60 FPS.
- Điều khiển thời điểm tạo pizza (theo biến spawn_timer).

2.11 Đọc/ghi file JSON

Khái niệm:

JSON (JavaScript Object Notation) là một định dạng trao đổi dữ liệu siêu nhẹ và dễ đọc, phù hợp cho cả con người lẫn máy móc. Trong thế giới lập trình và khoa học dữ liệu, JSON được ưa chuộng để truyền tải dữ liệu giữa máy chủ và ứng dụng web, cũng như lưu trữ dữ liệu một cách hiệu quả. Python, ngôn ngữ lập trình hot hit nhất hiện nay, có khả năng xử lý JSON cực kỳ dễ dàng và hiệu quả. Trong bài viết này, chúng ta sẽ cùng khám phá cách sử dụng Python để làm việc với JSON, từ các thao tác cơ bản đến lưu trữ dữ liệu.

Ứng dụng:

- Ghi lại điểm số người chơi sau mỗi lượt chơi.

Chương 3. Thiết kế và xây dựng chương trình

3.1. Sơ đồ khối hệ thống

3.1.1 Các module chính trong chương trình

* Giao diện khởi tạo

- Khởi tạo pygame, âm thanh, cửa sổ game.
- Thiết lập kích thước màn hình, FPS, font chữ.
- Tạo các hình ảnh như chảo và pizza bằng Surface.
- Bao gồm: pygame.init(), pygame.mixer.init()
  screen, clock, font, WIDTH, HEIGHT

* Quản lý đối tượng (Object Management)

- Định nghĩa các lớp Pan và Pizza.
- Cập nhật vị trí, trạng thái của các đối tượng.
- Sử dụng Sprite để tổ chức đối tượng dễ dàng hơn.
- Bao gồm: Lớp Pan, Lớp Pizza
  all_sprites, pizza_group

* Vòng lặp chính (Main Game Loop)

- Xử lý sự kiện (như thoát game).
- Cập nhật trò chơi (đối tượng, điểm số, tốc độ rơi).
- Vẽ mọi thứ lên màn hình.
- Kiểm tra va chạm và tình trạng kết thúc game.
- Bao gồm: while running:

  for event in pygame.event.get():

  all_sprites.update(), all_sprites.draw()

* Phát sinh và xử lý pizza

- Tạo pizza mới định kỳ (dựa trên biến spawn_timer).
- Kiểm tra xem pizza bị bỏ lỡ hay bắt được.
- Phát âm thanh tương ứng với từng trạng thái.

* Hiển thị và âm thanh

- Hiển thị điểm, số pizza bị bỏ lỡ.
- Phát các âm thanh như bắt được pizza, rơi pizza, thua.

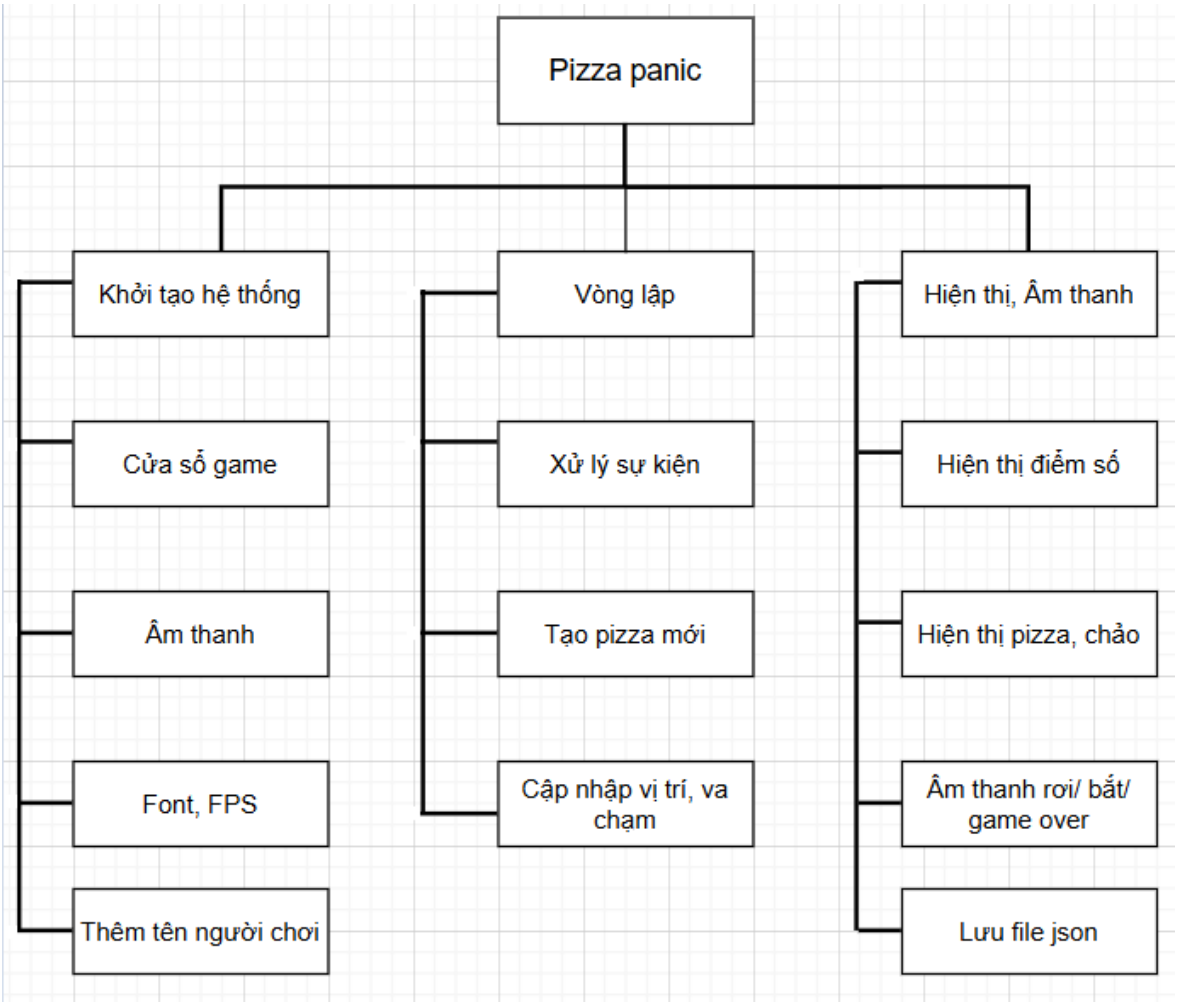
• Bao gồm: draw_text()

catch_sound.play(), miss_sound.play(), gameover_sound.play()

*Ghi file json

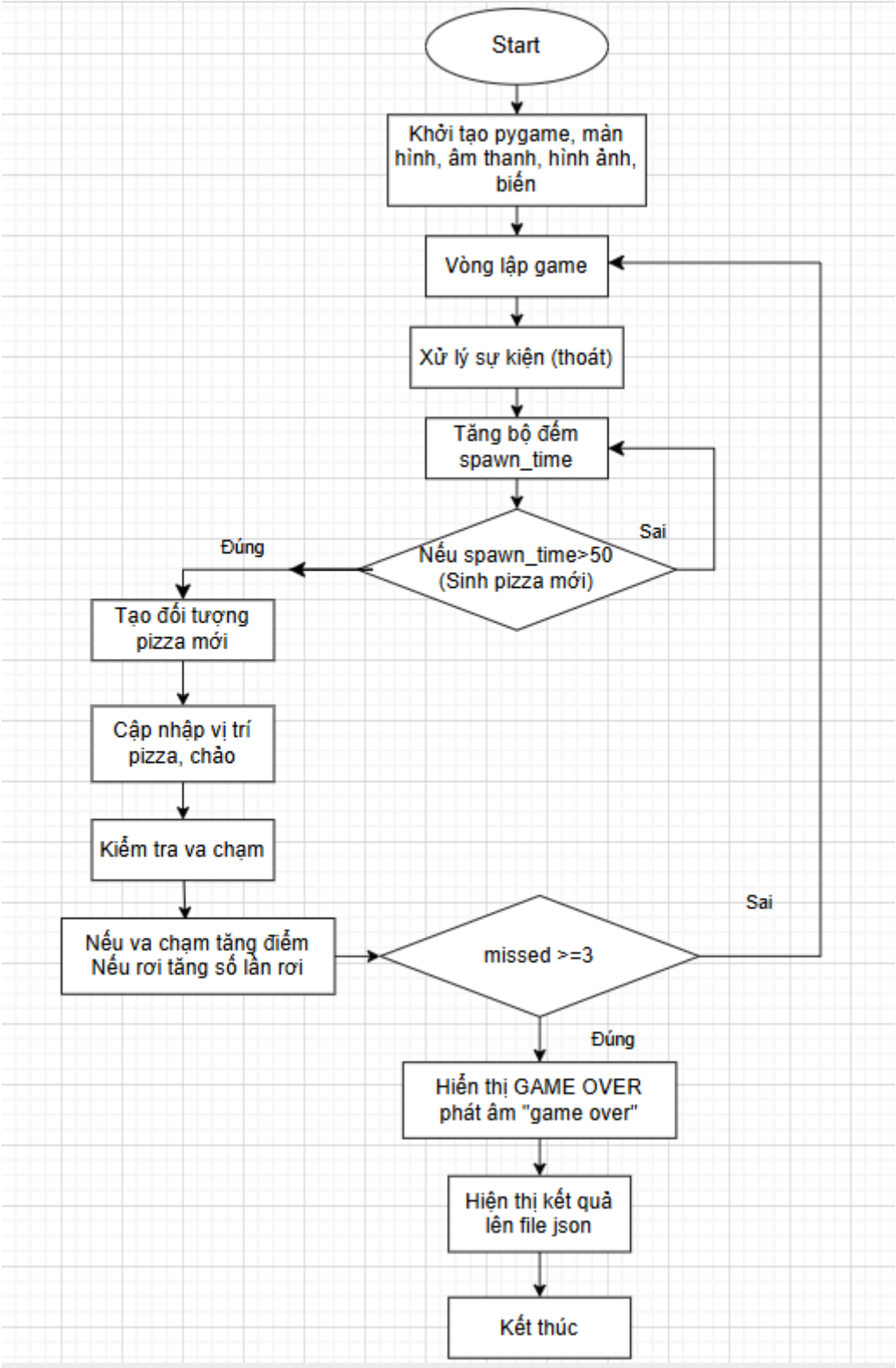
Ghi lại tên người chơi, điểm số và thời gian

3.1.2 Trình bày biểu đồ phân cấp chức năng



Hình 3.1 Biểu đồ chức năng

3.2. Sơ đồ khối các thuật toán chính



Hình 3.2 Sơ đồ khối

3.3. Cấu trúc dữ liệu

Tên trường	Kiểu dữ liệu	Mô tả
Nguoi_choi	String	Tên người chơi
Diemso	int	Điểm số đạt được (Số pizza đã bắt được)
So_pizza_roi	int	Số pizza không bắt được
tong_so_pizza	int	Số pizza được tạo ra bao gồm bắt được và bỏ lỡ
bat_dau_choi	String	Thời gian bắt đầu chơi
ket_thuc_choi	String	Thời gian kết thúc trò chơi

Bảng 3.3 Bảng dữ liệu trò chơi

3.4. Chương trình

```
import pygame
import random
import sys
import json
from datetime import datetime
import os

# ======= NHẬP TÊN NGƯỜI CHƠI =======
while True:
    Nguoi_choi = input("Nhập tên người chơi: ").strip()
    if Nguoi_choi:
        break
    print("Bạn chưa nhập tên. Vui lòng nhập lại!!!")
bat_dau_choi = datetime.now()
```

```
# ======= KHỞI TẠO PYGAME =======
pygame.init()
pygame.mixer.init()

# Âm thanh
catch_sound = pygame.mixer.Sound("catch.wav")
miss_sound = pygame.mixer.Sound("miss.wav")
gameover_sound = pygame.mixer.Sound("gameover.wav")

# Màn hình
WIDTH, HEIGHT = 640, 480
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Pizza Panic")
clock = pygame.time.Clock()

# ======= TẠO CHẢO =======
pan_img = pygame.Surface((100, 20), pygame.SRCALPHA)
pygame.draw.ellipse(pan_img, (200, 0, 0), pan_img.get_rect())

# ======= TẠO PIZZA =======
pizza_img = pygame.Surface((40, 40), pygame.SRCALPHA)
pygame.draw.circle(pizza_img, (255, 200, 0), (20, 20), 20)
for _ in range(5):
    x = random.randint(8, 32)
    y = random.randint(8, 32)
    pygame.draw.circle(pizza_img, (139, 69, 19), (x, y), 3)

font = pygame.font.SysFont(None, 36)

Diemso = 0
So_pizza_roi = 0
```

```
fall_speed = 3

# ======= CLASS CHẢO =======
class Pan(pygame.sprite.Sprite):
    def __init__(self):
        super().__init__()
        self.image = pan_img
        self.rect = self.image.get_rect()
        self.rect.y = HEIGHT - 40

    def update(self):
        self.rect.x = pygame.mouse.get_pos()[0] - self.rect.width // 2
        self.rect.left = max(0, self.rect.left)
        self.rect.right = min(WIDTH, self.rect.right)

# ======= CLASS PIZZA =======
class Pizza(pygame.sprite.Sprite):
    def __init__(self):
        super().__init__()
        self.image = pizza_img
        self.rect = self.image.get_rect()
        self.rect.x = random.randint(0, WIDTH - self.rect.width)
        self.rect.y = -self.rect.height

    def update(self):
        global So_pizza_roi
        self.rect.y += fall_speed
        if self.rect.top > HEIGHT:
            So_pizza_roi += 1
            miss_sound.play()
            self.kill()
```

```
# ======= SPRITE GROUPS =======
pan = Pan()
all_sprites = pygame.sprite.Group(pan)
pizza_group = pygame.sprite.Group()

# ======= VẼ CHỮ =======
def draw_text(surface, text, x, y, color=(0, 0, 0)):
    img = font.render(text, True, color)
    surface.blit(img, (x, y))

# ======= VÒNG LẶP GAME =======
running = True
spawn_timer = 0

while running:
    clock.tick(60)
    screen.fill((135, 206, 250))

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    spawn_timer += 1
    if spawn_timer > 50:
        pizza = Pizza()
        all_sprites.add(pizza)
        pizza_group.add(pizza)
        spawn_timer = 0

    all_sprites.update()
```

```
    hits = pygame.sprite.spritecollide(pan, pizza_group, True)
    if hits:
        catch_sound.play()
    Diemso += len(hits)

    fall_speed = 3 + Diemso // 5

    all_sprites.draw(screen)
    draw_text(screen, f"Score: {Diemso}", 10, 10)
    draw_text(screen, f"Missed: {So_pizza_roi}/3", 10, 40)

    if So_pizza_roi >= 3:
        gameover_sound.play()
        draw_text(screen, "GAME OVER", WIDTH // 2 - 100, HEIGHT // 2, (255, 0,
0))
        pygame.display.flip()
        pygame.time.wait(2000)
        running = False

    pygame.display.flip()

# ======= GHI KẾT QUẢ GAME =======
ket_thuc_choi = datetime.now()

ket_qua = {
    "Nguoi_choi": Nguoi_choi,
    "Diemso": Diemso,
    "So_pizza_roi": So_pizza_roi,
    "tong_so_pizza": Diemso + So_pizza_roi,
    "bat_dau_choi": bat_dau_choi.strftime("%Y-%m-%d %H:%M:%S"),
```

```
    "ket_thuc_choi": ket_thuc_choi.strftime("%Y-%m-%d %H:%M:%S")
}

file_path = "DLgame.json"
if os.path.exists(file_path):
    with open(file_path, "r") as f:
        try:
            data = json.load(f)
        except json.JSONDecodeError:
            data = []
else:
    data = []

data.append(ket_qua)

with open(file_path, "w") as f:
    json.dump(data, f, indent=4)

print("Kết quả game đã được lưu vào DLgame.json")

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

Chương 4 Thực nghiệm và kết luận

4.1. Thực nghiệm

4.1.1 Đánh giá chất lượng sản phẩm

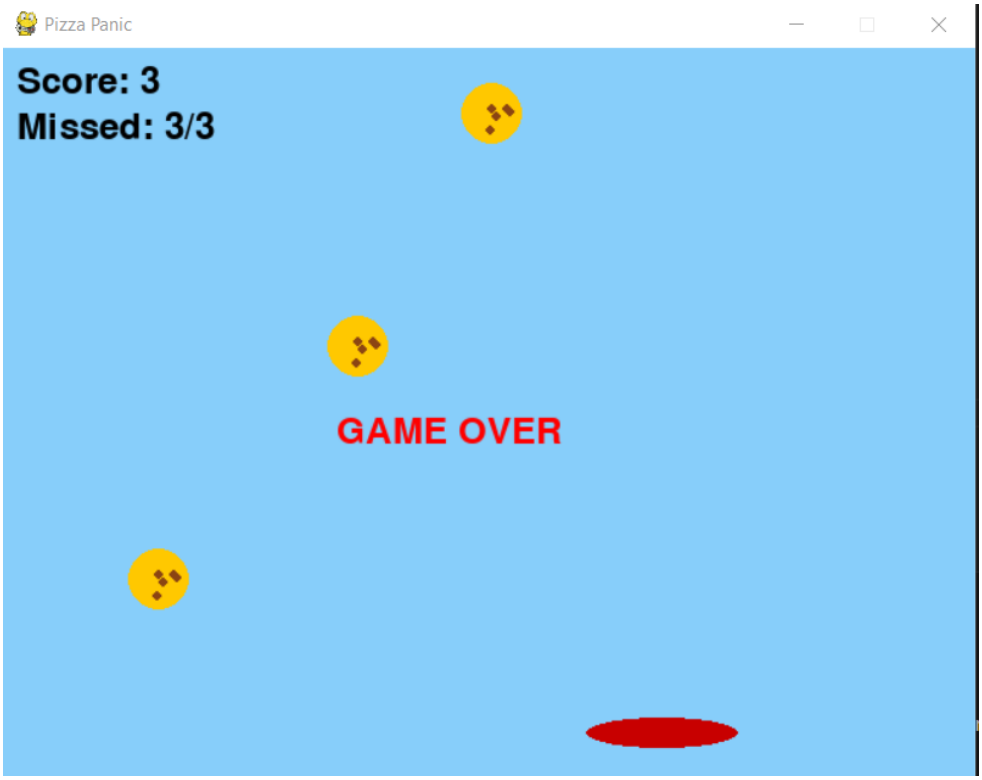
Sản phẩm hoạt động tốt, đạt được mục tiêu đề ra về tính năng của một game nhỏ hay giải trí đơn giản.

- Tính ổn định: Trò chơi hoạt động mượt mà, không bị lỗi trong quá trình chơi.
- Tính dễ sử dụng: Người chơi chỉ cần di chuột, thao tác đơn giản, phù hợp nhiều lứa tuổi.
- Khả năng lưu trữ: Dữ liệu được lưu đầy đủ, có thể mở rộng để thống kê hoặc phân tích sau.
- Khả năng mở rộng: Có thể dễ dàng thêm tính năng như tăng cấp độ, thêm loại pizza khác, ...

STT	Mô tả tình huống kiểm thử	Dữ liệu đầu vào	Kết quả mong đợi	Kết quả thực tế	Đạt/Không đạt
1	Nhập tên người chơi	"nhi"	Tên được ghi lại trong kết quả JSON	Đúng	Đạt
2	Bắt thành công pizza	Bắt đúng các pizza rơi xuống	Điểm tăng, âm thanh "catch" phát ra	Đúng	Đạt
3	Bỏ lỡ pizza 3 lần	Không bắt pizza	Hiện chữ "GAME OVER", kết thúc trò chơi	Đúng	Đạt
4	Dữ liệu ghi vào file JSON	-	File DLgame.json chứa thông tin đúng định dạng	Chính xác	Đạt
5	Hiển thị giao diện và hiệu ứng	-	Có nền, chảo, pizza, văn bản rõ ràng	Hiển thị tốt	Đạt
6	Tính thời gian bắt đầu và kết thúc	-	Trường bat_dau_choi và ket_thuc_choi hợp lệ	Thời gian chính xác	Đạt

Bảng 4.1 Bảng mô tả tính năng trò chơi

4.1.2 Hình ảnh mô tả các tính năng



Hình 4.2 Hình mô tả trò chơi

pizza.py DLgame.json

DLgame.json > {} 8 > ket_thuc_choi

```
52      {
55          "So_pizza_roi": 3,
56          "tong_so_pizza": 6,
57          "bat_dau_choi": "2025-05-30 14:39:05",
58          "ket_thuc_choi": "2025-05-30 14:39:15"
59      },
60      {
61          "Nguoi_choi": "n\u1ecb",
62          "Diemso": 3,
63          "So_pizza_roi": 3,
64          "tong_so_pizza": 6,
65          "bat_dau_choi": "2025-05-30 14:57:44",
66          "ket_thuc_choi": "2025-05-30 14:57:54"
67      }
68  ]
```

Hình 4.3 Hình mô tả dữ liệu trong file json

4.2. Kết luận

* Sản phẩm đã thực hiện được những gì

- Phát triển thành công trò chơi mini bằng thư viện Pygame với chủ đề "Bắt pizza".
- Tương tác đơn giản: người chơi điều khiển chảo để bắt pizza rơi bằng chuột.
- Giao diện trực quan, màu sắc tươi sáng, có âm thanh tương tác khi bắt/trượt pizza.
- Xử lý được điều kiện kết thúc trò chơi (khi bỏ lỡ 3 pizza).
- Ghi lại kết quả chơi bao gồm: tên người chơi, điểm, số pizza rơi, tổng số pizza, thời gian bắt đầu và kết thúc.
- Lưu kết quả vào file DLgame.json dưới dạng danh sách dữ liệu JSON để thuận tiện cho lưu trữ và phân tích sau này.

* Bài học thu được

- Hiểu và thực hành lập trình game cơ bản bằng Pygame.
- Biết cách xử lý sự kiện, cập nhật và vẽ lại màn hình theo vòng lặp game.
- Làm quen với khái niệm sprite, nhóm sprite, xử lý va chạm và hoạt ảnh đơn giản.
- Áp dụng thư viện datetime để lấy và định dạng thời gian thực.
- Biết cách đọc, ghi dữ liệu bằng định dạng JSON để lưu trữ trạng thái game.
- Cải thiện kỹ năng tổ chức mã, phân chia module theo chức năng.

* Hướng cải tiến trong tương lai

- Thêm nhiều cấp độ chơi với tốc độ rơi tăng dần, độ khó cao hơn.
- Thiết kế đồ họa hấp dẫn hơn: dùng hình ảnh thực tế thay vì vẽ bằng lệnh pygame đơn giản.
- Bảng xếp hạng: hiển thị top người chơi có điểm cao nhất từ file JSON.
- Thêm các loại pizza đặc biệt (ví dụ: pizza + điểm, pizza - điểm...).
- Tối ưu hóa giao diện người dùng: thêm màn hình bắt đầu, hướng dẫn chơi, nút thoát.
- Tối ưu cấu trúc mã nguồn theo mô hình hướng đối tượng rõ ràng hơn.

Link github: [hoangquyencode/BTCM_Python]