Biểu mẫu BTL01

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: .Nguyễn Lam Sơn. . . . . .

Lớp: K58KTPM . . . . . .

Giáo viên GIẢNG DẠY: Nguyễn Văn Huy . . . . . .

Thái Nguyên – 2025

Biểu mẫu BTL02

TRƯỜNG ĐHKTCN
KHOA ĐIỆN TỬ

CỘNG HOÀ 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:Nguyễn Lam Sơn.....

Lớp: K58KTPM…….

Ngành: ……Kĩ Thuật Máy Tính……

Giáo viên hướng dẫn:Nguyễn Văn Huy.....

Ngày giao đề 20-05-2025..... Ngày hoàn thành 08-06-2025.....

Tên đề tài : Pizza Panic với pygame

Yêu cầu : Triên khai game Pizza Panic (Chapter I) ding pygame/ivewires: điều khiên cái chảo hứng

.....

.....

.....

.....

.....

.....

.....

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

Lời mở đầu

Chương trình "Pizza Panic" là một dự án nhỏ được thực hiện nhằm áp dụng các kiến thức lập trình đã học, đặc biệt trong lĩnh vực phát triển trò chơi 2D sử dụng thư viện Pygame. Với sự hỗ trợ của Python và các công cụ như Livewires, dự án này không chỉ là cơ hội để khám phá thế giới lập trình mà còn giúp rèn luyện kỹ năng tư duy logic, quản lý code, và giải quyết vấn đề thực tế. Được bắt đầu từ ý tưởng đơn giản về một trò chơi bắt pizza, dự án đã dần hoàn thiện với các tính năng như điều khiển nhân vật, quản lý điểm số, và hiển thị giao diện động, phản ánh nỗ lực học hỏi và sáng tạo trong suốt quá trình thực hiện.

CHƯƠNG 1 GIỚI THIỆU ĐẦU BÀI

"Pizza Panic " là một trò chơi đơn giản được phát triển dựa trên thư viện Pygame và Livewires, với mục tiêu tạo ra một trải nghiệm giải trí cơ bản nhưng đầy thách thức cho người chơi. Trò chơi tập trung vào việc điều khiển một nhân vật (được biểu diễn bằng một chiếc chảo) để bắt các đối tượng pizza rơi từ trên xuống, đồng thời tránh để pizza rơi quá nhiều lần dẫn đến kết thúc trò chơi. Đây là một bài tập lập trình phù hợp để người học làm quen với các khái niệm lập trình hướng đối tượng, xử lý đồ họa 2D, và quản lý logic trò chơi.

1.1 Mô tả bài toán

Chương trình "Pizza Panic" được thiết kế với các thành phần và yêu cầu cụ thể như sau:

1.1.1 Dữ liệu vào dữ ra:

- **Dữ liệu vào:**
  - Người chơi điều khiển vị trí của chiếc chảo (Pan) bằng cách sử dụng chuột để di chuyển ngang trên màn hình. Vị trí chuột sẽ xác định tọa độ ngang của chảo, với giới hạn là không vượt ra ngoài biên trái hoặc biên phải của màn hình.
  - Các đối tượng pizza xuất hiện ngẫu nhiên từ phía trên cùng của màn hình, với tốc độ rơi được tạo ngẫu nhiên trong khoảng từ 2 đến 5 pixel mỗi khung hình.
- **Dữ liệu ra:**
  - Khi chảo bắt được pizza, điểm số (score) tăng lên 1, và một pizza mới được tạo ra để thay thế pizza vừa bị bắt.
  - Nếu pizza rơi xuống dưới cùng màn hình mà không được bắt, số lần bỏ lỡ (missed_pizza) tăng lên 1. Khi số lần bỏ lỡ đạt 3, trò chơi kết thúc và hiển thị thông báo "Game Over".

- Giao diện hiển thị điểm số và số lần bỏ lỡ tại góc trên bên phải màn hình, cùng với nền đồ họa tùy chỉnh hoặc màu trắng làm nền mặc định nếu hình ảnh nền không tải được.

1.1.2 Tính năng của chương trình:

- **Khởi tạo và quản lý màn hình:** Chương trình thiết lập một cửa sổ trò chơi có kích thước 640x480 pixel, với tốc độ khung hình (FPS) cố định là 50 để đảm bảo hoạt động mượt mà.
- **Đồ họa và đối tượng:** Sử dụng hình ảnh tùy chỉnh cho nền (background), chảo (pan), và pizza nếu có, hoặc các hình dạng cơ bản (hình chữ nhật, hình tròn) với màu sắc thay thế (xanh dương, cam, đỏ) khi hình ảnh không tải được. Các đối tượng được scale để phù hợp với kích thước màn hình.
- **Logic trò chơi:**
    - Nhân vật chảo được di chuyển theo vị trí chuột, với giới hạn để tránh ra ngoài màn hình.
    - Pizza rơi từ trên xuống với tốc độ ngẫu nhiên và được xóa khi rơi ra ngoài hoặc va chạm với chảo.
    - Hiển thị thông báo "Game Over" khi số lần bỏ lỡ đạt 3, với thời gian trì hoãn 5 giây trước khi thoát trò chơi.
- **Quản lý điểm số:** Điểm số tăng khi bắt được pizza, và trò chơi kết thúc khi bỏ lỡ quá 3 lần.

1.1.3 Kiểm tra & kết quả đầu:

- **Hiệu ứng điểm số:** Mỗi khi bắt được 5 pizza, người chơi được cộng thêm 5 điểm.
- **Điều kiện kết thúc:** Trò chơi kết thúc khi số lần bỏ lỡ đạt 3 lần, hiển thị thông báo "Game Over" tại trung tâm màn hình.

1.2 Thách thức

Việc phát triển "Pizza Panic" không chỉ là một bài tập lập trình cơ bản mà còn đặt ra nhiều thách thức đáng kể, đòi hỏi người lập trình phải xử lý cẩn thận để đảm bảo trò chơi hoạt động trơn tru và mang lại trải nghiệm tốt cho người chơi:

- **Quản lý chuyển động và va chạm:** Một trong những thách thức lớn nhất là đảm bảo rằng chuyển động của pizza và chảo được đồng bộ hóa một cách mượt mà. Tốc độ rơi ngẫu nhiên của pizza

đòi hỏi thuật toán cập nhật vị trí phải được thực hiện chính xác trong mỗi khung hình. Ngoài ra, việc phát hiện va chạm (collision detection) giữa chảo và pizza phải được xử lý chính xác để tránh lỗi như ghi điểm nhầm hoặc bỏ sót va chạm.

- **Xử lý ngoại lệ và tài nguyên:** Chương trình sử dụng các tệp hình ảnh (background.jpg, pan.jpg, pizza.jpg), nhưng nếu các tệp này không tồn tại hoặc không tải được, cần có cơ chế thay thế (fallback) để vẽ các hình dạng cơ bản. Điều này đòi hỏi lập trình viên phải dự đoán và xử lý các lỗi tiềm ẩn như pygame.error, đảm bảo trò chơi vẫn hoạt động ngay cả khi tài nguyên bị thiếu.
- **Tối ưu hóa hiệu suất:** Với tốc độ khung hình cố định là 50 FPS, việc cập nhật trạng thái của nhiều đối tượng pizza cùng lúc (tối đa 3 pizza ban đầu, có thể tăng khi bắt được pizza) đòi hỏi tối ưu hóa vòng lặp chính (main loop) để tránh hiện tượng giật lag. Điều này bao gồm việc xóa các đối tượng pizza đã rơi ra ngoài màn hình và cập nhật giao diện một cách hiệu quả.
- **Trải nghiệm người dùng:** Giao diện trò chơi cần thân thiện, với thông tin điểm số và số lần bỏ lỡ hiển thị rõ ràng. Việc hiển thị "Game Over" và trì hoãn 5 giây trước khi thoát cũng đòi hỏi quản lý thời gian chính xác, tránh làm gián đoạn trải nghiệm người chơi.
- **Tính mở rộng:** Mặc dù đây là một trò chơi đơn giản, thách thức nằm ở khả năng mở rộng sau này, chẳng hạn như thêm cấp độ khó (tăng tốc độ pizza), âm thanh, hoặc nhiều loại đối tượng khác. Điều này đòi hỏi thiết kế code linh hoạt và dễ bảo trì.

1.3 Vận dụng kiến thức

Để hoàn thành chương trình "Pizza Panic," người lập trình cần áp dụng một loạt các kiến thức và kỹ năng lập trình, bao gồm:

- **Lập trình hướng đối tượng (OOP):** Chương trình sử dụng các lớp (class) như Pan, Pizza, và Game để tổ chức code một cách có cấu trúc. Lớp Pan quản lý nhân vật chảo, lớp Pizza xử lý các đối tượng pizza rơi, và lớp Game điều phối toàn bộ logic trò chơi. Các phương thức như **init**, update, và draw thể hiện cách sử dụng các thuộc tính (attributes) và phương thức (methods) trong OOP.
- **Xử lý sự kiện (Event Handling):** Thư viện Pygame cung cấp cơ chế xử lý sự kiện thông qua pygame.event.get(), cho phép chương trình phản hồi với hành động của người chơi như di chuyển chuột hoặc đóng cửa sổ (pygame.QUIT). Điều này yêu cầu hiểu biết về cách lắng nghe và xử lý các sự kiện trong thời gian thực.
- **Đồ họa 2D và quản lý màn hình:** Sử dụng Pygame để tạo và quản lý màn hình trò chơi (pygame.display.set_mode), vẽ đối tượng

(screen.blit), và scale hình ảnh (pygame.transform.scale). Kiến thức về màu sắc (RGB) và hình học cơ bản (hình chữ nhật, hình tròn) cũng được áp dụng khi tạo fallback shapes.

- **Quản lý thời gian và vòng lặp trò chơi:** Vòng lặp chính (while running) kết hợp với clock.tick(FPS) để kiểm soát tốc độ khung hình, đảm bảo trò chơi chạy ổn định. Việc sử dụng pygame.time.get_ticks() để quản lý thời gian trì hoãn "Game Over" cho thấy sự vận dụng kỹ thuật quản lý thời gian trong lập trình game.
- **Xử lý ngoại lệ và debug:** Việc sử dụng try-except để xử lý lỗi khi tải hình ảnh thể hiện kỹ năng debug và đảm bảo tính ổn định của chương trình. Điều này đòi hỏi hiểu biết về cách xử lý ngoại lệ trong Python.
- **Thuật toán và logic trò chơi:** Việc tạo ngẫu nhiên vị trí và tốc độ của pizza (random.randint), kiểm tra va chạm (colliderect), và cập nhật trạng thái (score, missed_pizza) đòi hỏi tư duy thuật toán và logic rõ ràng để xây dựng quy tắc trò chơi.

CHƯƠNG 2 CƠ SỞ LÝ THUYẾT

Chương 2 trình bày các nội dung chuyên môn được sử dụng trong chương trình "Pizza Panic," bao gồm các khái niệm, cấu trúc dữ liệu, và kỹ thuật lập trình cần thiết để xây dựng và vận hành trò chơi.

2.1 Lập trình hướng đối tượng (Object-Oriented Programming - OOP)

Lập trình hướng đối tượng là nền tảng chính để tổ chức code trong chương trình. Các lớp (class) như Pan, Pizza, và Game được định nghĩa để mô phỏng các đối tượng trong trò chơi:

- **Lớp Pan:** Đại diện cho nhân vật chảo, chứa thông tin về hình ảnh, vị trí (rect), và phương thức di chuyển dựa trên vị trí chuột.
- **Lớp Pizza:** Quản lý các đối tượng pizza rơi, bao gồm tọa độ ngẫu nhiên, tốc độ, và logic va chạm với chảo.
- **Lớp Game:** Điều phối toàn bộ trò chơi, quản lý danh sách pizza, điểm số, và trạng thái "Game Over." Sử dụng các phương thức như **init** (khởi tạo), update (cập nhật trạng thái), và draw (vẽ đối tượng) giúp code dễ mở rộng và bảo trì.

2.2 Thư viện Pygame và quản lý đồ họa 2D

Thư viện Pygame cung cấp công cụ để phát triển trò chơi 2D, bao gồm:

- **Khởi tạo màn hình:** pygame.display.set_mode() tạo cửa sổ trò chơi với kích thước 640x480 pixel, và pygame.display.set_caption() đặt tiêu đề "Pizza Catching Game."
- **Vẽ và scale hình ảnh:** pygame.image.load() tải hình ảnh, pygame.transform.scale() điều chỉnh kích thước, và screen.blit() vẽ đối tượng lên màn hình. Nếu hình ảnh không tải được, pygame.Surface() và pygame.draw.circle() được dùng để vẽ hình dạng cơ bản.
- **Quản lý thời gian:** clock.tick(FPS) duy trì tốc độ khung hình 50 FPS, đảm bảo trò chơi chạy mượt mà.

2.3 Xử lý sự kiện và vòng lặp trò chơi

- **Xử lý sự kiện:** pygame.event.get() lắng nghe các hành động như đóng cửa sổ (pygame.QUIT) hoặc di chuyển chuột, cho phép chương trình phản hồi theo thời gian thực.
- **Vòng lặp chính:** Vòng lặp while running kết hợp với game.update() và game.draw() đảm bảo cập nhật trạng thái và vẽ giao diện liên tục. Điều kiện thoát vòng lặp được kích hoạt khi "Game Over" và sau 5 giây trì hoãn (pygame.time.get_ticks()).

2.4 Cấu trúc dữ liệu và thuật toán

- **Danh sách (List):** Lớp Game sử dụng danh sách self.pizzas để lưu trữ các đối tượng pizza đang hoạt động. Phương thức append() thêm pizza mới, và remove() xóa pizza khi rơi ra ngoài hoặc va chạm.
- **Số ngẫu nhiên:** random.randint(0, SCREEN_WIDTH - self.rect.width) và random.randint(2, 5) tạo vị trí và tốc độ ngẫu nhiên cho pizza, tăng tính bất ngờ.
- **Kiểm tra va chạm:** self.rect.colliderect() phát hiện va chạm giữa chảo và pizza, điều chỉnh điểm số và danh sách pizza tương ứng.

2.5 Xử lý ngoại lệ và debug

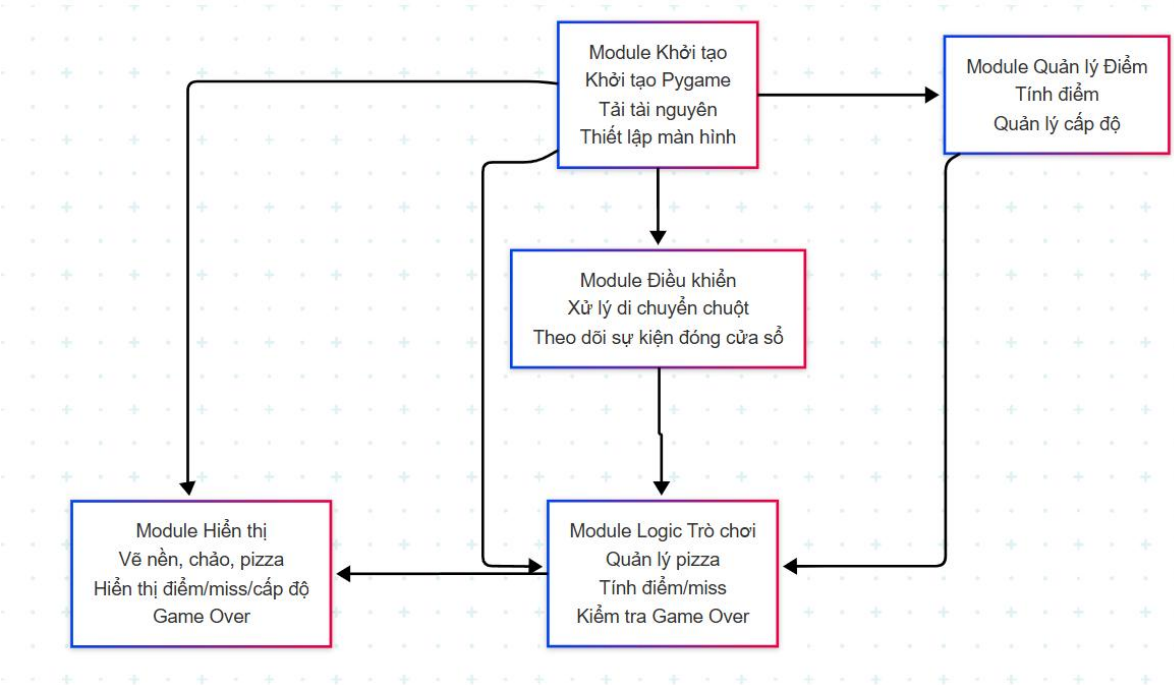
Sử dụng try-except để xử lý lỗi khi tải hình ảnh, đảm bảo chương trình chạy ổn định với fallback shapes. Điều này giúp tránh crash và cung cấp trải nghiệm người dùng liên tục ngay cả khi tài nguyên bị thiếu.

Tất cả các nội dung trên tạo nền tảng lý thuyết để phát triển "Pizza Panic," kết hợp giữa lập trình và thiết kế trò chơi một cách hiệu quả.

CHƯƠNG 3 THIẾT KẾ VÀ XÂY DỰNG CHƯƠNGTRÌNH

3.1. Sơ đồ khối hệ thống

- **Mô tả các module chính trong chương trình:**
  - **Module Khởi tạo (Initialization Module):** Chịu trách nhiệm khởi tạo thư viện Pygame trong file baitap.py, thiết lập màn hình trò chơi (640x480 pixel), và tải tài nguyên như hình ảnh nền, chảo, và pizza. Nếu tài nguyên không tải được, sử dụng hình dạng cơ bản làm thay thế.
  - **Module Điều khiển (Control Module):** Xử lý đầu vào từ người chơi thông qua di chuyển chuột để điều khiển vị trí chảo trong baitap.py, đồng thời theo dõi sự kiện đóng cửa sổ.
  - **Module Logic Trò chơi (Game Logic Module):** Quản lý trạng thái trò chơi trong baitap.py, bao gồm tạo và cập nhật danh sách pizza, tính điểm (tích hợp với beginners.py), và kiểm tra điều kiện "Game Over" khi bỏ lỡ 3 pizza.
  - **Module Hiển thị (Rendering Module):** Vẽ nền, chảo, pizza, và thông tin điểm số/miss trên màn hình trong baitap.py, cũng như hiển thị thông báo "Game Over" khi cần.
  - **Module Quản lý Điểm (Score Management Module):** Được định nghĩa trong file beginners.py, quản lý điểm số và cấp độ chơi, tăng cấp độ mỗi 5 điểm.
- **Biểu đồ phân cấp chức năng:**
  - **Khởi tạo trò chơi:** Thiết lập màn hình, tải tài nguyên, tạo đối tượng Game và ScoreManager.
  - **Cập nhật trạng thái:** Di chuyển chảo, cập nhật vị trí pizza, kiểm tra va chạm, tăng điểm/miss, và nâng cấp độ khi đạt 5 điểm.
  - **Vẽ giao diện:** Hiển thị nền, chảo, pizza, điểm số, miss, và cấp độ; hiển thị "Game Over" nếu cần.
  - **Kết thúc trò chơi:** Kiểm tra điều kiện thoát (miss ≥ 3) và trì hoãn 5 giây trước khi đóng.

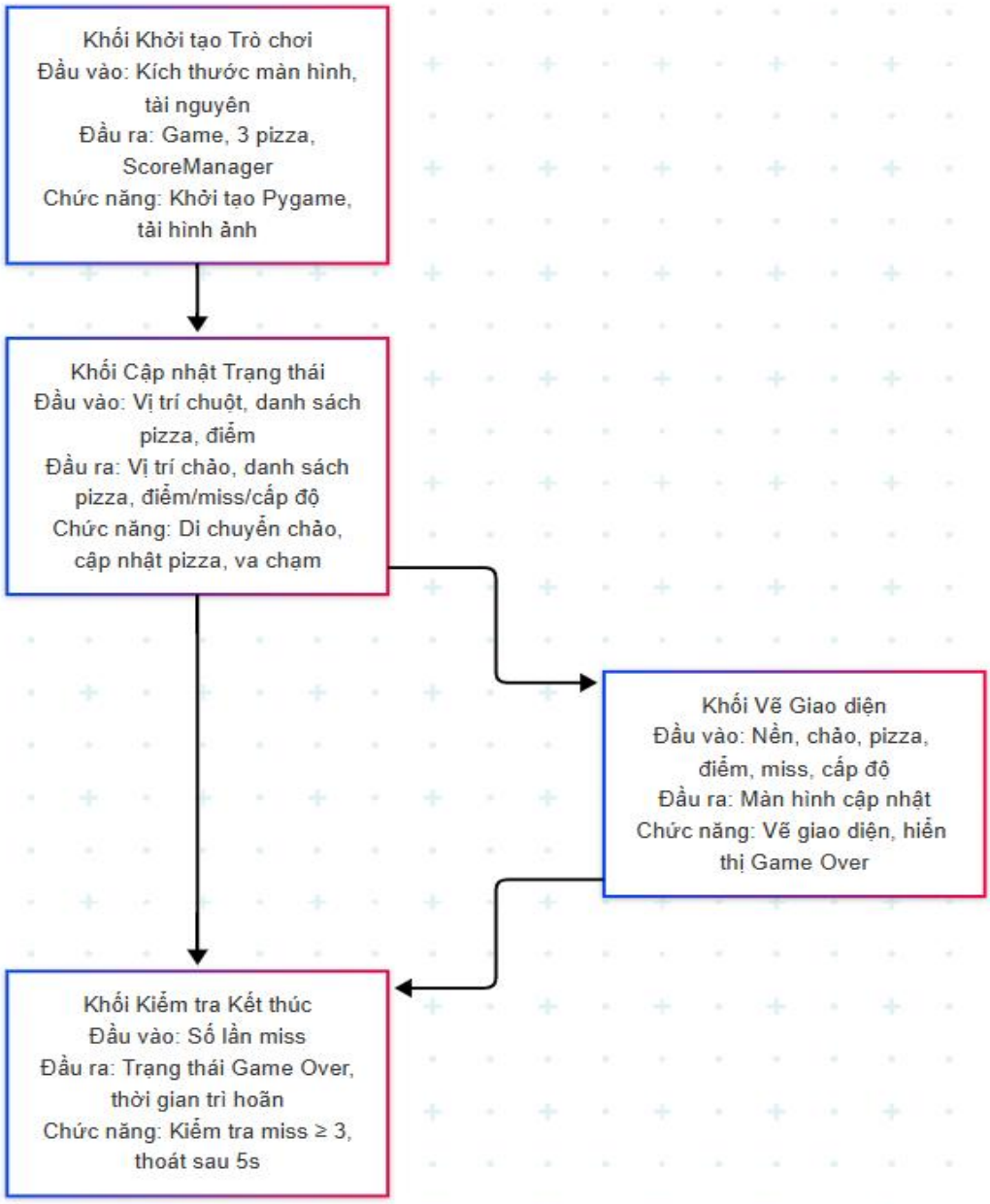


3.2. Sơ đồ khối các thuật toán chính

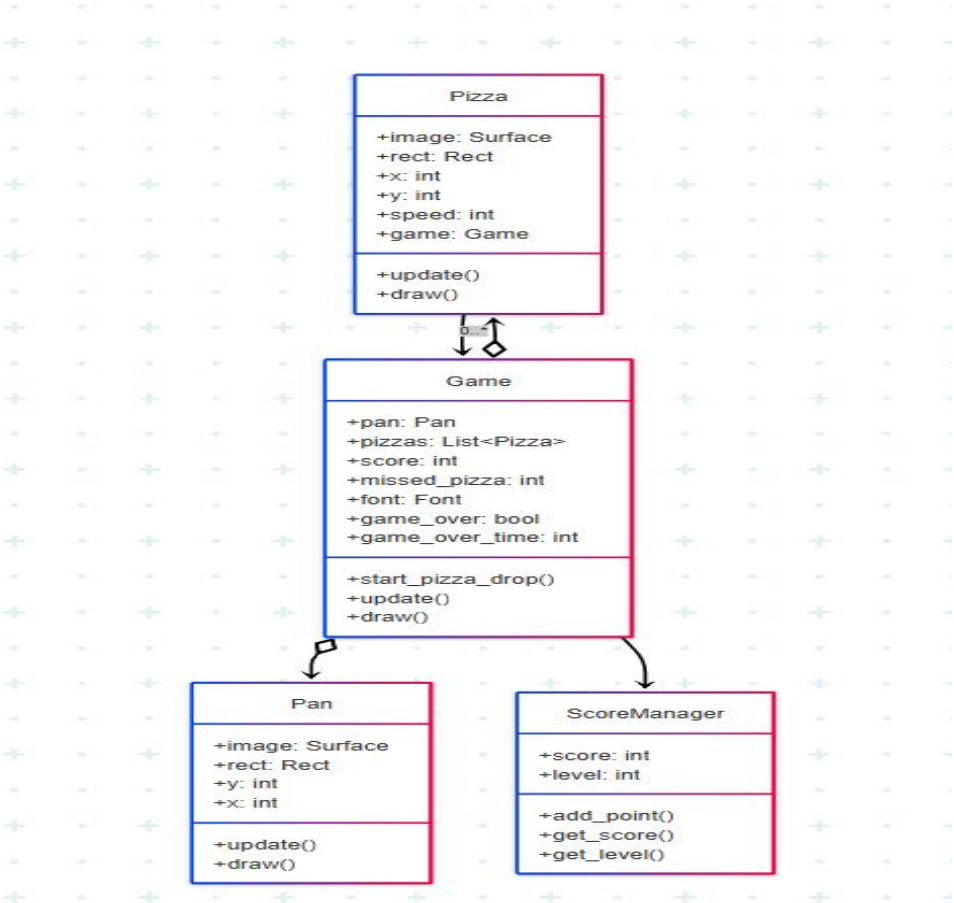
- **Mô tả các khối thuật toán chính, quan hệ đầu vào ra giữa các khối, chức năng từng thuật toán:**
    - **Khối Khởi tạo Trò chơi (Game Initialization):**
        - **Đầu vào:** Kích thước màn hình, tài nguyên hình ảnh.
        - **Đầu ra:** Đối tượng Game với pan, 3 pizza ban đầu, và ScoreManager.
        - **Chức năng:** Khởi tạo Pygame, tải hình ảnh, tạo lớp Game, thêm 3 pizza, và khởi tạo ScoreManager.
    - **Khối Cập nhật Trạng thái (State Update):**
        - **Đầu vào:** Vị trí chuột, danh sách pizza hiện tại, điểm số hiện tại.
        - **Đầu ra:** Vị trí mới của chảo, danh sách pizza cập nhật, điểm số/miss, và cấp độ.
        - **Chức năng:** Di chuyển chảo theo chuột, cập nhật vị trí pizza, kiểm tra va chạm, gọi add_point() từ ScoreManager khi bắt pizza, và kiểm tra trạng thái "Game Over."
    - **Khối Vẽ Giao diện (Rendering):**
        - **Đầu vào:** Nền, chảo, danh sách pizza, điểm số, miss, cấp độ.

- **Đầu ra:** Màn hình được vẽ với thông tin cập nhật.
- **Chức năng:** Vẽ nền, chảo, pizza, và hiển thị điểm/miss/cấp độ; hiển thị "Game Over" nếu cần.

- **Khối Kiểm tra Kết thúc (Game Over Check):**
  - **Đầu vào:** Số lần miss.
  - **Đầu ra:** Trạng thái "Game Over" và thời gian trì hoãn.
  - **Chức năng:** Kiểm tra miss ≥ 3, kích hoạt "Game Over," và thoát sau 5 giây.



3.3. Cấu trúc dữ liệu



- **Mô tả các bảng dữ liệu, các trường thông tin lưu trữ trong dữ liệu:**
    - **Bảng Pan (trong baitap.py):**
        - **image:** Hình ảnh chảo (Surface).
        - **rect:** Tọa độ và kích thước chảo (Rect).
        - **y:** Vị trí cố định theo trục y (400).
        - **x:** Vị trí động theo trục x (tùy vị trí chuột).
    - **Bảng Pizza (trong baitap.py):**
        - **image:** Hình ảnh pizza (Surface).
        - **rect:** Tọa độ và kích thước pizza (Rect).
        - **x:** Vị trí ngẫu nhiên theo trục x (0 đến SCREEN_WIDTH - width).
        - **y:** Vị trí động theo trục y (tăng theo speed).
        - **speed:** Tốc độ rơi (2 đến 5 pixel/khung hình).
        - **game:** Tham chiếu đến đối tượng Game.

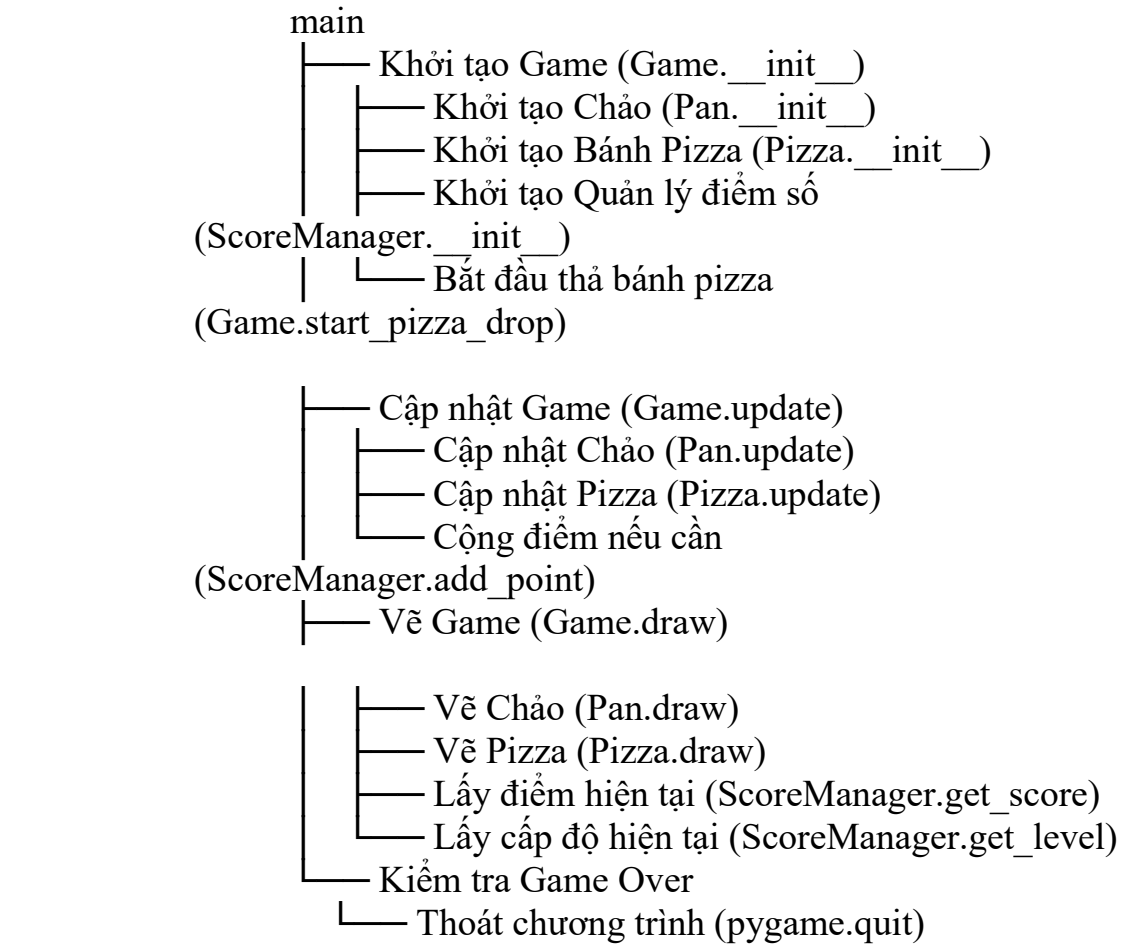
- **Bảng Game (trong baitap.py):**
    - **pan:** Đối tượng Pan.
    - **pizzas:** Danh sách các đối tượng Pizza.
    - **score:** Điểm số (số nguyên, tăng khi bắt pizza, liên kết với ScoreManager).
    - **missed_pizza:** Số lần bỏ lỡ (số nguyên, tăng khi pizza rơi ra ngoài).
    - **font:** Đối tượng font để vẽ chữ.
    - **game_over:** Trạng thái trò chơi (Boolean).
    - **game_over_time:** Thời gian kích hoạt "Game Over" (ticks).
- **Bảng ScoreManager (trong beginners.py):**
    - **score:** Điểm số hiện tại (số nguyên).
    - **level:** Cấp độ hiện tại (số nguyên, tăng mỗi 5 điểm).

3.4. Chương trình

- **Hàm trong chương trình chính:**
    - **init(self) trong Pan (baitap.py):**
        - Khởi tạo chảo với hình ảnh và vị trí ban đầu (x = SCREEN_WIDTH // 2, y = 400).
    - **update(self) trong Pan (baitap.py):**
        - Cập nhật vị trí x theo chuột, giới hạn trong biên màn hình.
    - **draw(self) trong Pan (baitap.py):**
        - Vẽ chảo lên màn hình.
    - **init(self, game) trong Pizza (baitap.py):**
        - Khởi tạo pizza với vị trí ngẫu nhiên, tốc độ, và tham chiếu game.
    - **update(self) trong Pizza (baitap.py):**
        - Cập nhật vị trí y, kiểm tra va chạm với chảo, và xóa nếu rơi ra ngoài.

- **draw(self) trong Pizza (baitap.py):**
  - Vẽ pizza lên màn hình.
- **init(self) trong Game (baitap.py):**
  - Khởi tạo pan, danh sách pizza, điểm số, và trạng thái ban đầu; gọi start_pizza_drop().
- **start_pizza_drop(self) trong Game (baitap.py):**
  - Thêm 3 pizza ban đầu vào danh sách.
- **update(self) trong Game (baitap.py):**
  - Cập nhật pan và pizza, kiểm tra điều kiện "Game Over."
- **draw(self) trong Game (baitap.py):**
  - Vẽ nền, pan, pizza, điểm số, và thông báo "Game Over."
- **init(self) trong ScoreManager (beginners.py):**
  - Khởi tạo điểm số và cấp độ ban đầu (score = 0, level = 1).
- **add_point(self) trong ScoreManager (beginners.py):**
  - Tăng điểm số, nâng cấp độ nếu score % 5 == 0.
- **get_score(self) trong ScoreManager (beginners.py):**
  - Trả về điểm số hiện tại.
- **get_level(self) trong ScoreManager (beginners.py):**
  - Trả về cấp độ hiện tại.
- **main() trong baitap.py:**
  - Chạy vòng lặp chính, xử lý sự kiện, cập nhật và vẽ giao diện, thoát khi "Game Over."

Sơ đồ phân cấp chức năng



CHƯƠNG 4 THỰC NGHIỆM VÀ KẾT LUẬN

4.1 Thực nghiệm

4.1.1. Giao diện vào game

4.1.2. Đếm số lần hứng trượt

4.1.3. khi hứng trượt 3 lần là game over

4.2. Kết luận

- **Sản phẩm đã làm được những gì:** Chương trình "Pizza Panic" đã được phát triển thành công với các tính năng chính như: khởi tạo màn hình trò chơi 640x480 pixel, điều khiển chảo bằng chuột để bắt pizza rơi ngẫu nhiên, tính điểm khi bắt được pizza, theo dõi số lần bỏ lỡ, và hiển thị thông báo "Game Over" khi bỏ lỡ 3 lần. Hệ thống cũng tích hợp quản lý điểm và cấp độ thông qua module ScoreManager, tăng cấp độ mỗi 5 điểm. Giao diện hỗ trợ tải hình ảnh tùy chỉnh hoặc dùng hình dạng cơ bản làm fallback, đảm bảo hoạt động ổn định ngay cả khi tài nguyên thiếu hụt.
- **Học được gì:** Qua quá trình phát triển, đã nắm vững các khái niệm lập trình hướng đối tượng (OOP) với các lớp như Pan, Pizza, Game, và ScoreManager. Đồng thời, học cách sử dụng thư viện Pygame để quản lý đồ họa 2D, xử lý sự kiện, và tối ưu hóa vòng lặp trò chơi. Kỹ năng xử lý ngoại lệ, quản lý thời gian thực, và tích hợp nhiều module (như baitap.py và beginners.py) cũng được cải thiện đáng kể.

- **Sẽ cải tiến gì:** Trong tương lai, có thể cải tiến bằng cách thêm cấp độ khó hơn với tốc độ pizza tăng dần, tích hợp âm thanh để tăng trải nghiệm, và phát triển giao diện người dùng thân thiện hơn (như nút khởi động lại trò chơi). Ngoài ra, sẽ tối ưu hóa hiệu suất khi xử lý nhiều đối tượng cùng lúc và thêm tính năng lưu điểm cao nhất để tăng tính cạnh tranh.

4.3 Link GitHub: NguyenLamSon-123/BAITAPLONPYTHON: K225480106076