**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC NHA TRANG**

**KHOA CÔNG NGHỆ THÔNG TIN**

Logo, icon

Description automatically generated

BÀI TẬP NHÓM CUỐI KỲ 30%

**LẬP TRÌNH PYTHON**

PROJECT: HUNGRY SNAKE

Giảng viên hướng dẫn: TS. Nguyễn Mạnh Cương

|  |  |  |
| --- | --- | --- |
| Sinh viên thực hiện | Mã số sinh viên | Đóng góp từng thành viên |
| Lê Văn Nhật Quân | 64131909 | 50% |
| Phan Nhựt Hào | 64130608 | 50% |

KHÁNH HÒA-2024

MỤC LỤC

[Chương 1: Giới thiệu Project/Bài Tập Lớn/Đồ Án Môn Học 3](#_Toc183687280)

[a) Giới thiệu tổng quan Project 3](#_Toc183687281)

[b) Cấu trúc dữ liệu (CTDL) và Giải thuật để thực hiện Project 3](#_Toc183687282)

[Chương 2 Tất cả cấu trúc dữ liệu và các giải thuật sử dụng trong Project 4](#_Toc183687283)

[a) Game Board 4](#_Toc183687284)

[1. Cấu trúc dữ liệu bảng điều khiển (Game Board): 4](#_Toc183687285)

[2. Các thành phần chính 4](#_Toc183687286)

[b) Tọa độ 6](#_Toc183687287)

[1. Cơ chế định vị tọa độ 6](#_Toc183687288)

[2. Phương pháp xác định tọa độ 6](#_Toc183687289)

[3. Kiểm soát và ràng buộc tọa độ 7](#_Toc183687290)

[c) Lưu trữ 8](#_Toc183687291)

[d) Lưu trữ thức ăn 9](#_Toc183687292)

[e) Các thuộc tính 10](#_Toc183687293)

[f) Theo dõi điểm số 11](#_Toc183687294)

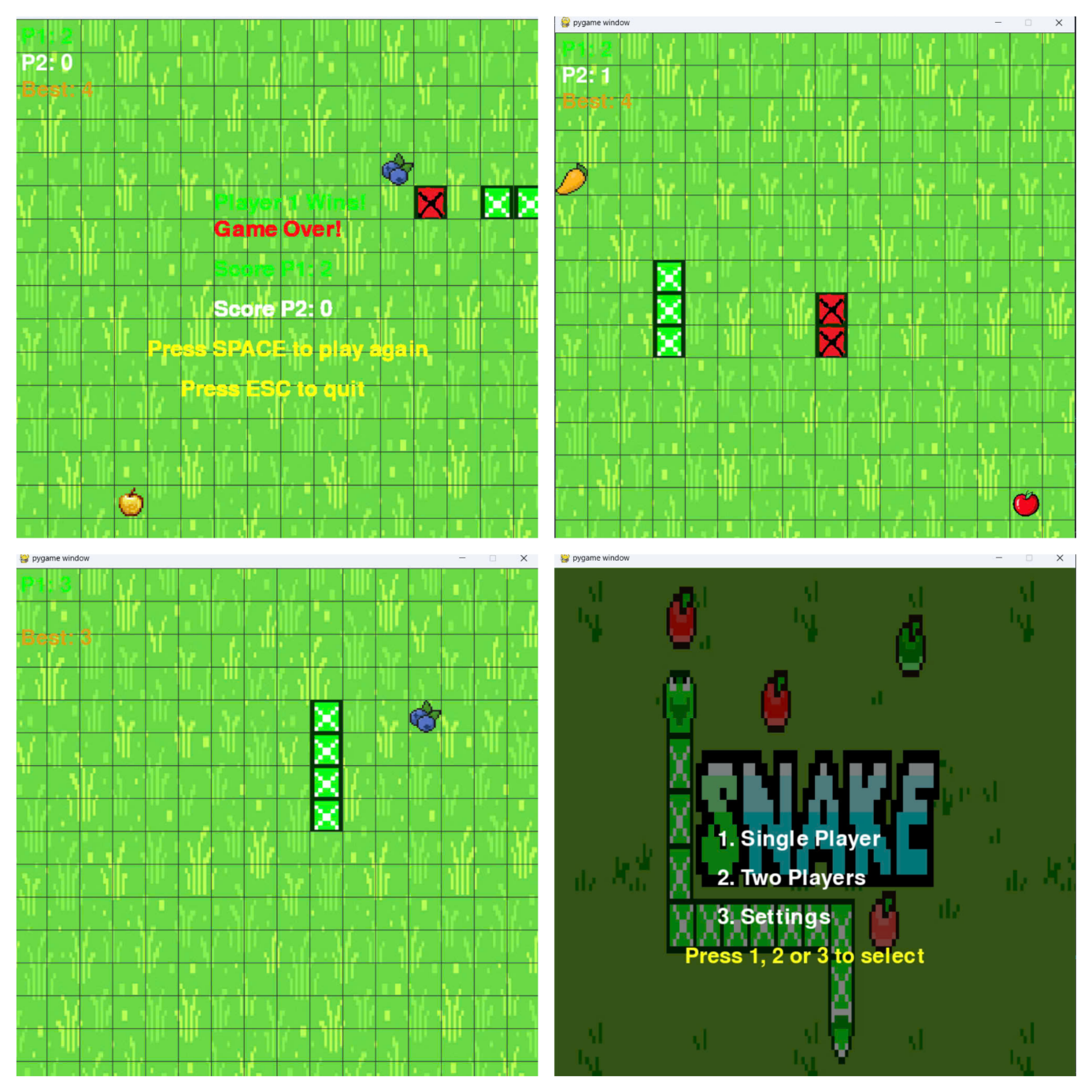
[g) Điều khiển cho 2 người chơi 11](#_Toc183687295)

[Chương 3: Tài liệu tham khảo 11](#_Toc183687296)

# Chương 1: Giới thiệu Project/Bài Tập Lớn/Đồ Án Môn Học

1. Giới thiệu tổng quan Project

Game Rắn Săn Mồi là một trò chơi điện tử cổ điển được phát triển bằng thư viện Pygame. Trò chơi cho phép một hoặc hai người chơi điều khiển rắn để ăn thức ăn và tăng điểm số.



1. Cấu trúc dữ liệu (CTDL) và Giải thuật để thực hiện Project
2. Cấu trúc dữ liệu:

* Game Board: Sử dụng screen của pygame
* Tọa độ thân rắn: Tuple (x,y)
* Thân rắn: List chứa các tuple tọa độ (body\_snake1, body\_snake2)
* Food: List chứa tuple tọa độ (food\_positions)
* Điều khiển: Dictionary (CONTROLS)
* Điểm số: Biến số nguyên (score1, score2)
* Trạng thái game: Biến Boolean

1. Giải thuật:

* Random generation (sinh số ngẫu nhiên)
* Collision detection (phát hiện va chạm)
* Timer-based movement (di chuyển dựa trên timer)
* Grid-based positioning (định vị trên lưới)
* State management (quản lý trạng thái)
* Event handling (xử lý sự kiện)
* Score tracking (theo dõi điểm số)

1. Những thư viện (module/package) có trong Project
2. pygame

* Khởi tạo cửa sổ game và đồ họa
* Xử lý âm thanh (mixer)
* Xử lý sự kiện bàn phím
* Vẽ đồ họa (hình ảnh, text, grid)
* Quản lý FPS và thời gian
* Load và xử lý hình ảnh

1. random

* Sinh vị trí ngẫu nhiên cho thức ăn
* Chọn ngẫu nhiên hình ảnh thức ăn từ danh sách

1. sys

* Thoát game (sys.exit())
* Xử lý tắt cửa sổ game

1. json

* Đọc/ghi file cấu hình điều khiển (controls.json)
* Lưu trữ và nạp các thiết lập phím điều khiển của người chơi

# Chương 2 Tất cả cấu trúc dữ liệu và các giải thuật sử dụng trong Project

1. Game Board
2. Cấu trúc dữ liệu bảng điều khiển (Game Board):

#### 1.1. Thông số kỹ thuật cơ bản

* Kích thước cửa sổ: 800x800 pixels (WINDOW\_SIZE)
* Kích thước mỗi ô: 50x50 pixels (SNAKE\_PART)
* Số ô trên mỗi chiều: 16x16 ô (GRID\_SIZE)
* Độ phân giải: 16x16 ô vuông đồng nhất

1. Các thành phần chính

#### 2.1. Cửa sổ game

A black background with white text and colorful text

Description automatically generated

* Tạo cửa sổ vuông sử dụng Pygame
* Cung cấp không gian hiển thị cho tất cả thành phần game

#### 2.2. Lưới game

A screen shot of a computer code

Description automatically generated

* Sử dụng hình nền từ file ảnh
* Vẽ lưới trong suốt phía trên hình nền
* Màu lưới: RGB(40,40,50) với alpha=128
* Đường lưới cách đều 50 pixels

#### 2.3. Hệ thống tọa độ và định vị

##### 2.3.1. Tọa độ trên lưới

* Gốc tọa độ (0,0) tại góc trên bên trái
* Trục x tăng từ trái sang phải
* Trục y tăng từ trên xuống dưới
* Phạm vi tọa độ:
  + x: [0, WINDOW\_SIZE-1]
  + y: [0, WINDOW\_SIZE-1]

##### 2.3.2. Chuyển đổi tọa độ

* Công thức: Tọa độ pixel = Chỉ số ô × SNAKE\_PART
* Ví dụ: Ô (3,4) → (150,200) pixels

#### 2.4. Quản lý đối tượng trên bảng

##### 2.4.1. Hiển thị rắn

A screen shot of a computer code

Description automatically generated

* Sử dụng danh sách các tọa độ (body\_snake1/2)
* Vẽ hình ảnh thân rắn tại mỗi tọa độ
* Kích thước phần thân rắn: 50x50 pixels

##### 2.4.2. Hiển thị thức ăn

A screen shot of a computer program

Description automatically generated

* Vị trí thức ăn được lưu trong food\_positions
* Hình ảnh thức ăn được scale phù hợp với kích thước ô
* Có hai loại thức ăn: thường và bonus

#### 2.5. Hệ thống kiểm tra va chạm

A computer screen shot of text

Description automatically generated

Các loại va chạm được kiểm tra:

* Va chạm với tường biên
* Va chạm với thân rắn
* Va chạm giữa hai rắn (chế độ 2 người chơi)

#### 2.6. Hiển thị thông tin trò chơi

A screenshot of a computer code

Description automatically generated

* Hiển thị điểm số người chơi
* Hiển thị điểm cao nhất
* Sử dụng màu sắc phân biệt cho từng loại thông tin

1. Tọa độ
2. Cơ chế định vị tọa độ

* Game sử dụng hệ tọa độ 2D với gốc tọa độ (0,0) ở góc trên bên trái màn hình
* Trục X: tăng từ trái sang phải
* Trục Y: tăng từ trên xuống dưới
* Đơn vị: pixel

1. Phương pháp xác định tọa độ

#### 2.1. Tọa độ ban đầu

A blue text on a black background

Description automatically generated

* Rắn 1 bắt đầu ở vị trí 1/4 màn hình
* Rắn 2 bắt đầu ở vị trí 3/4 màn hình
* Đảm bảo rắn không xuất hiện ở biên màn hình

#### 2.2. Cập nhật tọa độ khi di chuyển

A screenshot of a computer

Description automatically generated

#### 2.3. Quy tắc di chuyển

A screen shot of a computer code

Description automatically generated

Các hướng di chuyển:

* Trái: (-50, 0)
* Phải: (50, 0)
* Lên: (0, -50)
* Xuống: (0, 50)

1. Kiểm soát và ràng buộc tọa độ

3.1. Kiểm tra va chạm biên

A computer screen with text

Description automatically generated

* Tọa độ X không được < 0 hoặc >= 800
* Tọa độ Y không được < 0 hoặc >= 800
* Vi phạm = game over

#### 3.2. Căn chỉnh lưới (Grid Alignment)

* Tọa độ luôn là bội số của SNAKE\_PART (50)
* Đảm bảo rắn di chuyển chính xác trên lưới
* Ví dụ các tọa độ hợp lệ:
* (0, 0): Góc trên trái
* (50, 50): Ô thứ hai theo cả hai chiều
* (750, 750): Ô cuối cùng bên phải dưới

#### 3.3. Quản lý thân rắn

A screen shot of a computer code

Description automatically generated

* Mỗi phần tử thân rắn được lưu dưới dạng tuple (x, y)
* Cập nhật liên tục theo nguyên tắc FIFO (First In First Out)
* Độ dài thân rắn được kiểm soát bằng biến length

1. Lưu trữ

Để lưu trữ từng phần của thân rắn, chúng ta sử dụng danh sách (list) trong Python. Danh sách là một cấu trúc dữ liệu rất linh hoạt, cho phép lưu trữ nhiều loại dữ liệu và thực hiện các thao tác như thêm, xóa, và truy cập phần tử một cách dễ dàng.

Cách lưu trữ

* Các phần của thân rắn được lưu trữ trong các biến body\_snake1 và body\_snake2. Cụ thể, chúng được định nghĩa như sau:
* body\_snake1 = [] : Danh sách lưu trữ các phần của thân rắn người chơi 1
* body\_snake2 = [] : Danh sách lưu trữ các phần của thân rắn người chơi 2

1. Mỗi phần của thân rắn

* Mỗi phần của thân rắn được biểu diễn dưới dạng một tuple (cặp giá trị) chứa tọa độ (x, y). Ví dụ, nếu đầu rắn ở vị trí (100, 150), chúng ta có thể lưu trữ nó như sau:
* body\_snake1.append((100, 150)) : Thêm phần đầu rắn vào danh sách

1. Cập nhật thân rắn

* Khi rắn di chuyển, tọa độ mới của đầu rắn sẽ được tính toán và thêm vào danh sách body\_snake
* Cập nhật tọa độ đầu rắn
* x1 += x1\_change
* y1 += y1\_change
* Thêm tọa độ mới của đầu rắn vào danh sách
* body\_snake1.append((x1, y1))

1. Giữ lại chiều dài rắn

* Để duy trì chiều dài của thân rắn, chúng ta cần kiểm tra xem chiều dài của danh sách có vượt quá chiều dài tối đa của rắn hay không. Biến length đại diện cho chiều dài hiện tại của rắn. Khi chiều dài của thân rắn vượt quá giá trị này, phần đầu tiên (phần cũ nhất) sẽ bị xóa:
* Xóa phần đầu tiên nếu chiều dài vượt quá
* if len(body\_snake1) > length1:

del body\_snake1[0] : Xóa phần đầu tiên trong danh sách

1. Lưu trữ thức ăn

Thức ăn được lưu trữ trong các biến danh sách.Có hai loại thức ăn:

* Thức ăn chính: Thức ăn thông thường mà rắn có thể ăn để tăng chiều dài và điểm số.
* Thức ăn bonus: Thức ăn đặc biệt giúp rắn tăng chiều dài nhiều hơn và có thể có các hiệu ứng bổ sung.

Các biến lưu trữ thức ăn được định nghĩa như sau:

* food\_positions = [(0, 0)] : Danh sách lưu trữ vị trí của thức ăn chính
* bonus\_food\_position = [(0, 0)] : Danh sách lưu trữ vị trí của thức ăn bonus

Cách thức lưu trữ và cập nhật vị trí thức ăn

1. Vị trí thức ăn

* Mỗi loại thức ăn được lưu trữ dưới dạng một tuple (x, y), trong đó:
* **x**: tọa độ ngang của thức ăn trên màn hình.
* **y**: tọa độ dọc của thức ăn trên màn hình.

1. Sinh thức ăn

* Khi trò chơi bắt đầu hoặc khi rắn ăn thức ăn, chúng ta cần sinh ra vị trí mới cho thức ăn. Hàm spawn\_food() sẽ được gọi để cập nhật vị trí của thức ăn chính:

A black background with white text

Description automatically generated

* Trong hàm này, vị trí của thức ăn được sinh ra ngẫu nhiên trong lưới, đảm bảo rằng tọa độ của thức ăn là bội số của kích thước phần rắn (SNAKE\_PART).

1. Thức ăn bonus

* Tương tự như thức ăn chính, thức ăn bonus cũng được lưu trữ và sinh ra bằng cách sử dụng một hàm riêng:

A black screen with white text

Description automatically generated

1. Va chạm thức ăn

* Khi rắn di chuyển, trò chơi sẽ kiểm tra xem rắn có ăn thức ăn hay không. Nếu tọa độ của đầu rắn trùng với tọa độ của thức ăn, rắn sẽ ăn thức ăn và chiều dài cùng với điểm số sẽ được cập nhật:
* if x1 == food\_x and y1 == food\_y:

length1 += 1 : Tăng chiều dài của rắn

score1 += 1 : Tăng điểm số

spawn\_food() : Sinh thức ăn mới

* Tương tự, nếu rắn ăn thức ăn bonus:
* elif x1 == bonus\_x and y1 == bonus\_y:

length1 += 2 : Tăng chiều dài thêm 2

score1 += 2 : Tăng điểm số thêm 2

spawn\_bonus\_food() : Sinh thức ăn bonus mới

1. Các thuộc tính

Mỗi phần tử trong trò chơi đều có những thuộc tính riêng biệt ảnh hưởng đến cách thức hoạt động của trò chơi. Các thuộc tính này không chỉ xác định hình dạng, màu sắc và vị trí của các phần tử mà còn ảnh hưởng đến cách mà người chơi tương tác với trò chơi.

Các phần tử chính :

1. Rắn
2. Tọa độ (x, y)

* Mô tả: Tọa độ xác định vị trí của thức ăn trên màn hình.
* Ý nghĩa: Vị trí của thức ăn cần phải được sinh ra ngẫu nhiên trong không gian trò chơi. Điều này tạo ra sự thú vị và thử thách cho người chơi khi họ phải di chuyển rắn đến vị trí của thức ăn.

1. Loại thức ăn

* Mô tả: Có hai loại thức ăn chính: thức ăn thường và thức ăn bonus.
* Ý nghĩa: Mỗi loại thức ăn có ảnh hưởng khác nhau đến rắn. Thức ăn thường tăng chiều dài rắn một đơn vị, trong khi thức ăn bonus có thể tăng chiều dài nhiều hơn. Người lập trình cần phải xác định loại thức ăn và cập nhật các thuộc tính tương ứng khi rắn ăn.

1. Thức ăn
2. Tọa độ (x, y)

* Mô tả: Tọa độ xác định vị trí của thức ăn trên màn hình.
* Ý nghĩa: Vị trí của thức ăn cần phải được sinh ra ngẫu nhiên trong không gian trò chơi. Điều này tạo ra sự thú vị và thử thách cho người chơi khi họ phải di chuyển rắn đến vị trí của thức ăn.

1. Loại thức ăn

* Mô tả: Có hai loại thức ăn chính: thức ăn thường và thức ăn bonus.
* Ý nghĩa: Mỗi loại thức ăn có ảnh hưởng khác nhau đến rắn. Thức ăn thường tăng chiều dài rắn một đơn vị, trong khi thức ăn bonus có thể tăng chiều dài nhiều hơn. Người lập trình cần phải xác định loại thức ăn và cập nhật các thuộc tính tương ứng khi rắn ăn.

1. Màn hình
2. Kích thước màn hình

* Mô tả: Kích thước của cửa sổ trò chơi được xác định bởi các biến WINDOW\_SIZE, SNAKE\_PART, và GRID\_SIZE.
* Ý nghĩa: Kích thước màn hình quyết định không gian mà rắn có thể di chuyển. Người lập trình cần đảm bảo rằng mọi phần tử đều nằm trong giới hạn này để tránh lỗi và va chạm không mong muốn.

1. Màu sắc và hình ảnh

* Mô tả: Mỗi phần tử trong trò chơi (rắn, thức ăn, nền) đều có màu sắc và hình ảnh riêng biệt.
* Ý nghĩa: Màu sắc và hình ảnh giúp phân biệt các phần tử trong trò chơi, tạo sự hấp dẫn và sinh động cho trải nghiệm người chơi.

1. Theo dõi điểm số
2. Sử dụng biến toàn cục
3. Khai báo biến

score1 = 0

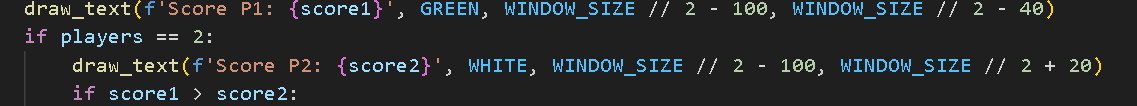
score2 = 0

1. Cập nhật biến

Mỗi khi người chơi ăn thức ăn, điểm số sẽ được cập nhật:

* score1 += 1 : Khi người chơi 1 ăn thức ăn

1. Hiện thị điểm số mức độ trên màn hình



* Hiện thị điểm số người chơi khi game kết thúc

1. Điều khiển cho 2 người chơi

* Đặt các phím điều khiển cho người chơi 1 và người chơi 2 khác nhau để tránh xung đột. Ví dụ: người chơi 1 sử dụng W, A, S, D, trong khi người chơi 2 sử dụng các phím mũi tên.

A screen shot of a computer program

Description automatically generated

# Chương 3: Tài liệu tham khảo

Tham khảo tại : [Lập trình game bằng python và pygame: Tạo game rắn săn mồi | How to code snake game](https://www.youtube.com/watch?v=F8s1aQswQqw)