

**BỘ GIÁO DỤC VÀ ĐÀO TẠO  
TRƯỜNG ĐẠI HỌC NHA TRANG  
KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN BÀI TẬP LỚN  
GAME BREAKOUT**

**Giảng viên hướng dẫn: Nguyễn Mạnh Cường  
Học phần : Lập trình Python  
Lớp học phần: 64.CNTT-4  
Sinh viên thực hiện: Vĩnh Thuận – 64132409 (NT)  
Phạm Tuấn Kiệt - 64131060**

*Nha Trang, ngày 21 tháng 10 năm 2024*

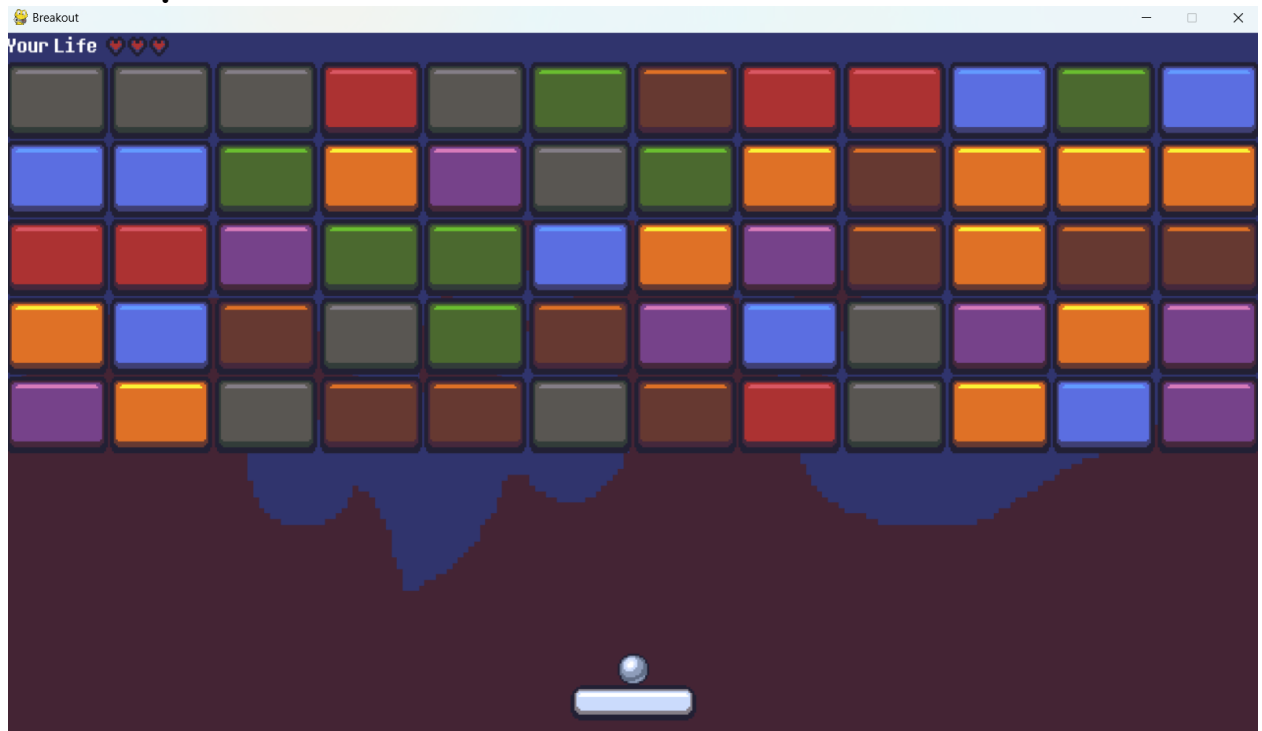


# Mục lục

I. Giới thiệu chung.....	1
1. Giới thiệu về BreakOut.....	1
2. Những cấu trúc dữ liệu và giải thuật sẽ dùng.....	1
II. Cấu trúc dữ liệu và giải thuật .....	3
1. Cấu trúc dữ liệu.....	3
1.1 List – Tuple – Dictionary .....	3
1.2 Vector2 – pygame.math.Vector2.....	4
2. Giải thuật .....	4
2.1 Giải thuật va chạm bóng (Block, Paddle, Display).....	4
2.2 Giải thuật xử lý sự kiện của người chơi .....	7
2.3 Giải thuật khóa paddle trong màn hình display.....	7
2.4 Giải thuật tương tác với vật phẩm nâng cấp .....	8
2.5 Giải thuật xử lý khi Block nhận sát thương.....	8
2.6 Giải thuật tạo Map Block ngẫu nhiên (Nâng cấp).....	9
III. Mô tả cách chơi BreakOut.....	10
IV. Cài đặt Project .....	10
V. Tài liệu tham khảo.....	
VI. Phân chia công việc.....	

## I. Giới thiệu chung




### 1. Giới thiệu về BreakOut



Hình 1.1: Ảnh minh họa Game BreakOut

BreakOut là một tựa game Arcade cổ điển nổi tiếng được phát hành vào 13/5/1976. Trong BreakOut, người chơi phải điều khiển một thanh ngang (paddle) theo chiều ngang để đỡ bóng (ball) không để rơi khỏi màn hình. Đồng thời, người chơi có thể nhặt các item hỗ trợ và phá hủy các lớp gạch được sinh ra ở trên cùng màn hình.[1]

Trong phiên bản cải tiến, ngoài lối chơi cơ bản, game sẽ có các tính năng bổ sung bao gồm các vật nâng cấp rơi ra khi người chơi phá hủy những viên gạch.

Vật phẩm				
Chức năng	Cung cấp cho người chơi khả năng bắn đạn phá hủy các khối dễ dàng hơn	Cung cấp cho người chơi thêm mạng. Tuy nhiên, người chơi chỉ có tối đa 3 mạng	Làm tăng kích thước của Paddle dài ra, tăng khả năng đỡ bóng	Làm tăng tốc độ duy chuyển của Paddle lên nhanh hơn

### 2. Những cấu trúc dữ liệu và giải thuật sẽ dùng

Game BreakOut sử dụng một số cấu trúc và dữ liệu:

- List – Tuple – Dictionary
- Player, Ball, Block, Upgrade, Projectile

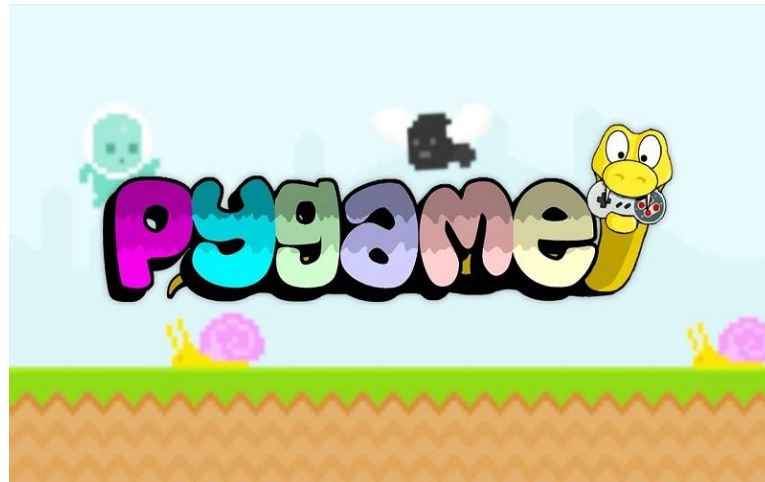
- Vector2 (Một cấu trúc dữ liệu để lưu trữ và thao tác với các tọa độ trong không gian 2 chiều)

Một số thuật toán:

- Kiểm tra va chạm (Collision)
- Cập nhật vị trí (Position)
- Tính toán thời gian (Time)
- Xử lý các sự kiện (Event)
- Xử lý các nâng cấp

Một số thư viện (Module/Package):

- **Pygame:** Đây là một thư viện quan trọng không thể thiếu để phát triển nhiều loại Game trên Python nói chung và BreakOut nói riêng. Nó tích hợp cả đồ họa máy tính và thư viện âm thanh được tạo ra đặc biệt để tương thích với ngôn ngữ lập trình Python.[2]



Hình 1.2: Thư viện Pygame

- **Random:** Đây là thư viện cung cấp các thao tác ngẫu nhiên. Diễn hình trong Game BreakOut như là tạo Map ngẫu nhiên với các khối gạch ngẫu nhiên, cung cấp các item nâng cấp ngẫu nhiên,...
- **Sys:** Đây là thư viện cung cấp các hàm và các biến được sử dụng để thao tác các phần khác nhau của môi trường chạy Python. Nó cho phép chúng ta truy cập các tham số và chức năng cụ thể của hệ thống.[3]
- **Time:** Đây là thư viện dùng để quản lý thời gian, cho phép thực hiện các thao tác như lấy thời gian hiện tại, tạm dừng chương trình, hoặc đo thời gian chạy của các đoạn mã.
- **Os:** Đây là thư viện cung cấp các chức năng được sử dụng để tương tác với hệ điều hành và cũng có được thông tin liên quan về nó.[3]

Một số thư viện (Module/Package) tự định nghĩa:

- **settings:** chứa các cài đặt chung cho game bao gồm kích thước cửa sổ, bản đồ các khối, bảng màu sắc của gạch, kích thước, khoảng cách của các khối và danh sách các nâng cấp.

- **sprites**: quản lý các đối tượng chính trong game như player, ball, blocks, upgrade và projectile. Module này giúp quản lý logic và hành vi của các đối tượng trên bao gồm chuyển động, va chạm và tương tác giữa chúng
- **surfacemaker**: có chức năng quản lý và tạo ra đồ họa các bề mặt (surfaces) cho các khối (blocks) với kích thước và hình dạng tùy chỉnh bằng cách kết hợp các hình ảnh đã được tải lên.

## II. Cấu trúc dữ liệu và giải thuật

## 1. Cấu trúc dữ liệu

Cấu trúc dữ liệu được sử dụng trong Project là List, Tuple, Vector2, Dictionary

## 1.1 List – Tuple – Dictionary

Tuple được sử dụng vì tính toàn vẹn không bị thay đổi dữ liệu và nhóm được các giá trị một cách gọn gàng. Trong project tuple được sử dụng chủ yếu để xác định vị trí (position) và kích thước (size) của các đối tượng trong trò chơi.

```
self.rect = self.image.get_rect(midbottom = pos)

self.image = surfacemaker.get_surf('player',(WINDOW_WIDTH // 10,WINDOW_HEIGHT // 20))
self.rect = self.image.get_rect(midbottom = (WINDOW_WIDTH // 2,WINDOW_HEIGHT - 20))
```

Hình 2.1.1: Minh họa sử dụng Tuple

Dictionary được sử dụng vì tốc độ truy xuất nhanh, dễ quản lý và dễ dàng trong việc thay đổi, mở rộng. Trong Project Dictionary được sử dụng để ánh xạ các ký tự đại diện cho các block sang màu sắc tương ứng.

```
COLOR_LEGEND = {'1': 'blue', '2': 'green', '3': 'red', '4': 'orange',  
                '5': 'purple', '6': 'bronze', '7': 'grey', }
```

Hình 2.1.2: Minh họa sử dụng Dictionary

List được sử dụng trong nhiều trường hợp nhằm tổ chức, lưu trữ liệu 1 cách rõ ràng. Ví dụ List được sử dụng để tạo Map Block, cụ thể là 1 danh sách chứa các chuỗi

```

    BLOCK_MAP = [
        '66666666666666',
        '44444444444444',
        '33333333333333',
        '22222222222222',
        '11111111111111',
        ' ',
        ' ',
        ' ',
        ' ',
        ' '
    ]

```

và mỗi chuỗi đại diện cho một hàng Block trong game. Nó cho phép dễ dàng xác định vị trí và loại khối mà người chơi sẽ tương tác.

*Hình 2.1.3: Minh họa sử dụng List*

## 1.2 Vector2 – pygame.math.Vector2

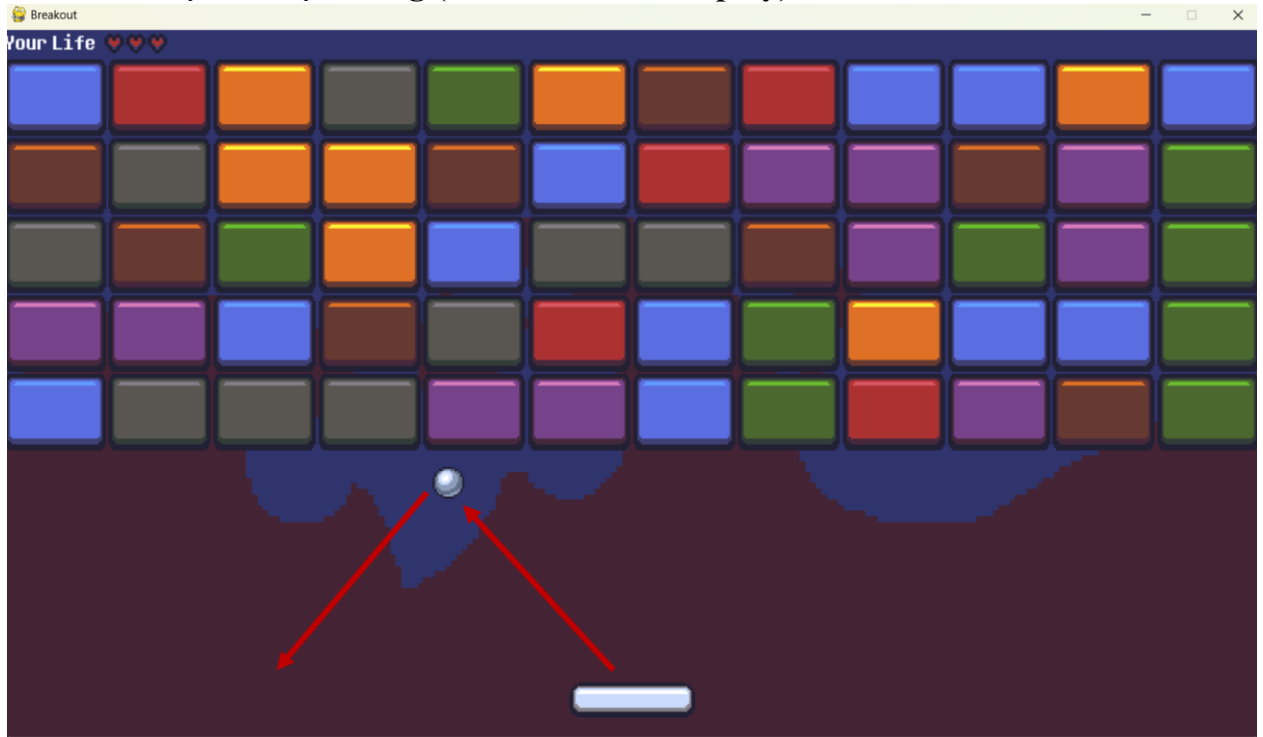
Vector2 giúp dễ dàng xử lý vị trí, thực hiện các phép toán trực tiếp trên vector làm đơn giản hóa code. Trong Project, Vector2 được sử dụng để quản lý vị trí và di chuyển của Ball, Item Upgrade, Projectile trong Project

```
self.pos = pygame.math.Vector2(self.rect.topleft)
self.direction = pygame.math.Vector2((choice((1,-1)), -1))
```

*Hình 2.1.4: Minh họa sử dụng Vector2*

## 2. Giải thuật

### 2.1 Giải thuật và chạm bóng (Block, Paddle, Display)



*Hình 2.2.1: Minh họa va chạm bóng*

Bóng sẽ được đặt ở chính giữa màn hình trên Paddle, khi bóng rời Paddle và va chạm với các phía của màn hình, bóng sẽ được đổi đi một hướng khác. Người chơi đảm bảo rằng bóng luôn được di chuyển bên trong màn hình, nếu bóng rời khỏi màn hình, tức là bóng chạm vào biên dưới màn thì sẽ mất đi một mạng.

```
def window_collision(self,direction):
    if direction == 'horizontal':
        if self.rect.left < 0:
            self.rect.left = 0
            self.pos.x = self.rect.x
            self.direction.x *= -1

        if self.rect.right > WINDOW_WIDTH:
            self.rect.right = WINDOW_WIDTH
            self.pos.x = self.rect.x
            self.direction.x *= -1

    if direction == 'vertical':
        if self.rect.top < 0:
            self.rect.top = 0
            self.pos.y = self.rect.y
            self.direction.y *= -1

        if self.rect.bottom > WINDOW_HEIGHT:
            self.active = False
            self.direction.y = -1
            self.player.hearts -= 1
            self.fail_sound.play()
```

Hình 2.2.2: Phương thức `window_collision`

Phương thức **`window_collision(self, direction)`** có chức năng kiểm tra và xử lý va chạm của đối tượng với biên của cửa sổ trò chơi. Với tham số `direction` ('horizontal' hoặc 'vertical') để xác định hướng va chạm.

Cách hoạt động:

- Hướng ngang – horizontal:
- + Nếu cạnh trái (*rect.left*) của đối tượng nhỏ hơn 0, đối tượng sẽ được đặt lại ở cạnh trái của cửa sổ (0), và hướng di chuyển ngang sẽ bị đảo ngược (*self.direction.x \*= -1*).
- + Nếu cạnh phải (*rect.right*) lớn hơn chiều rộng của cửa sổ (*WINDOW\_WIDTH*), đối tượng sẽ được đặt lại ở cạnh phải của cửa sổ, và hướng di chuyển cũng bị đảo ngược.
- Hướng dọc – vertical:
- + Nếu cạnh trên (*rect.top*) nhỏ hơn 0, đối tượng sẽ được đặt lại ở cạnh trên (0) và hướng di chuyển dọc sẽ bị đảo ngược.



- + Nếu cạnh dưới (*rect.bottom*) lớn hơn chiều cao của cửa sổ (*WINDOW\_HEIGHT*), đối tượng sẽ không còn hoạt động (*self.active = False*), hướng di chuyển dọc sẽ được đặt lại về -1, và số trái tim (hearts) của người chơi

```
def collision(self, direction):
    # find overlapping objects
    overlap_sprites = pygame.sprite.spritecollide(self, self.blocks, False)
    if self.rect.colliderect(self.player.rect):
        overlap_sprites.append(self.player)

    if overlap_sprites:
        if direction == 'horizontal':
            for sprite in overlap_sprites:
                if self.rect.right >= sprite.rect.left and self.old_rect.right <= sprite.old_rect.left:
                    self.rect.right = sprite.rect.left - 1
                    self.pos.x = self.rect.x
                    self.direction.x *= -1
                    self.impact_sound.play()

                if self.rect.left <= sprite.rect.right and self.old_rect.left >= sprite.old_rect.right:
                    self.rect.left = sprite.rect.right + 1
                    self.pos.x = self.rect.x
                    self.direction.x *= -1
                    self.impact_sound.play()

                if getattr(sprite, 'health', None):
                    sprite.get_damage(1)

            if direction == 'vertical':
                for sprite in overlap_sprites:
                    if self.rect.bottom >= sprite.rect.top and self.old_rect.bottom <= sprite.old_rect.top:
                        self.rect.bottom = sprite.rect.top - 1
                        self.pos.y = self.rect.y
                        self.direction.y *= -1
                        self.impact_sound.play()

                    if self.rect.top <= sprite.rect.bottom and self.old_rect.top >= sprite.old_rect.bottom:
                        self.rect.top = sprite.rect.bottom + 1
                        self.pos.y = self.rect.y
                        self.direction.y *= -1
                        self.impact_sound.play()

                if getattr(sprite, 'health', None):
                    sprite.get_damage(1)
```

sẽ giảm đi 1.

Hình 2.2.3: Phương thức *collision*

Phương thức **collision(self, direction)** có nhiệm vụ xử lý va chạm giữa Ball với các đối tượng khác (như Block và Player) và điều chỉnh vị trí của nó cũng như xử lý âm thanh va chạm và giảm máu của Block nếu có.

Cách hoạt động:

- Tìm đối tượng chồng lên
- + **pygame.sprite.spritecollide(self, self.blocks, False)**: Hàm này kiểm tra xem Ball có va chạm với bất kỳ đối tượng nào trong nhóm *self.blocks* hay không. Kết quả sẽ là một danh sách các đối tượng chồng lên (*overlap*).
- + **self.rect.colliderect(self.player.rect)**: Kiểm tra va chạm giữa Ball và Player. Nếu có va chạm, Player sẽ được thêm vào danh sách *overlap\_sprites*.
- Xử lý va chạm: Nếu có các đối tượng chồng lên (*if overlap\_sprites*), phương thức sẽ tiếp tục xử lý theo hướng va chạm (*direction*), có thể là *'horizontal'* hoặc *'vertical'*.

## 2.2 Giải thuật xử lý sự kiện của người chơi

```
def input(self):  
    keys = pygame.key.get_pressed()  
    if keys[pygame.K_RIGHT]:  
        self.direction.x = 1  
    elif keys[pygame.K_LEFT]:  
        self.direction.x = -1  
    else:  
        self.direction.x = 0
```

Hình 2.2.4: Phương thức `get_damage`

Phương thức **input(self)** sẽ nhận sự kiện ấn nút (`pygame.key.get_pressed()`) di chuyển qua trái (`pygame.K_LEFT`) hoặc phải (`pygame.K_RIGHT`) để điều khiển Paddle di chuyển qua lại đỡ Ball.

## 2.3 Giải thuật khóa paddle trong màn hình display

```
def screen_constraint(self):  
    if self.rect.right > WINDOW_WIDTH:  
        self.rect.right = WINDOW_WIDTH  
        self.pos.x = self.rect.x  
    if self.rect.left < 0:  
        self.rect.left = 0  
        self.pos.x = self.rect.x
```

Hình 2.2.5: Phương thức `screen_constraint`

Phương thức **screen\_constraint(self)** giữ cho Paddle khỏi việc di chuyển ra ngoài ranh giới của cửa sổ trò chơi. Phương thức này sẽ kiểm tra vị trí, xem thử vị trí hiện tại có vượt quá bên phải màn `self.rect.right > WINDOW_WIDTH` hoặc bên trái màn `self.rect.left < 0` hay không, nếu có sẽ cập nhật lại vị trí `self.pos.x = self.rect.x`

## 2.4 Giải thuật tương tác với vật phẩm nâng cấp

```
def upgrade(self, upgrade_type):
    if upgrade_type == 'speed':
        self.speed += 50
    if upgrade_type == 'heart':
        if self.hearts < 3:
            self.hearts += 1

    if upgrade_type == 'size':
        new_width = self.rect.width * 1.1
        self.image = self.surfacemaker.get_surf('player', (new_width, self.rect.height))
        self.rect = self.image.get_rect(center = self.rect.center)
        self.pos.x = self.rect.x

    if upgrade_type == 'laser':
        self.laser_amount += 1
```

Hình 2.2.6 Phương thức upgrade

Phương thức **upgrade(self, upgrade\_type)** cho phép người chơi nhận các nâng cấp khác nhau trong trò chơi, cải thiện khả năng di chuyển, sức sống, kích thước và khả năng tấn công. Mỗi loại nâng cấp có những ảnh hưởng riêng biệt đến lối chơi, tạo ra sự đa dạng cho người chơi.

## 2.5 Giải thuật xử lý khi Block nhận sát thương

```
def get_damage(self, amount):
    self.health -= amount

    if self.health > 0:
        self.image = self.surfacemaker.get_surf(COLOR_LEGEND[str(self.health)], (BLOCK_WIDTH, BLOCK_HEIGHT))
    else:
        if randint(0, 10) < 9:
            self.create_upgrade(self.rect.center)
        self.kill()
```

Hình 2.2.7 Phương thức get\_damage

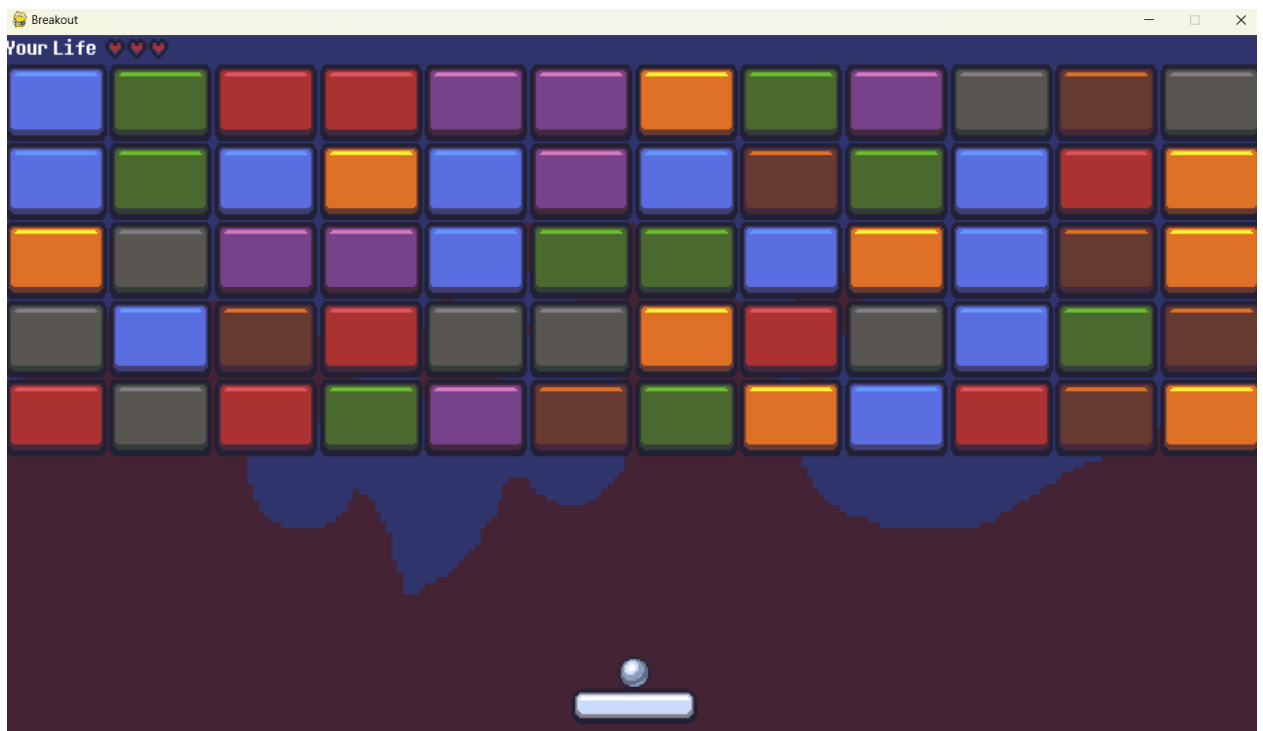
Phương thức **get\_damage(self, amount)** chịu trách nhiệm giảm sức khỏe của đối tượng khi bị tấn công – Block. Nếu sức khỏe còn lại sau khi giảm vẫn lớn hơn 0, Block sẽ được cập nhật hình ảnh tương ứng. Nếu sức khỏe giảm xuống 0, Block sẽ có khả năng tạo ra một nâng cấp và sau đó sẽ bị tiêu diệt.

## 2.6 Giải thuật tạo Map Block ngẫu nhiên (Nâng cấp)

```
def generate_block_map():  
    block_map = []  
    num_random_rows=5  
    total_rows=9  
    length=12  
    # Tạo các chuỗi ngẫu nhiên  
    for _ in range(num_random_rows):  
        random_string = ''.join(random.choice('1234567') for _ in range(length))  
        block_map.append(random_string)  
  
    # Thêm các chuỗi khoảng trắng  
    for _ in range(total_rows - num_random_rows):  
        block_map.append(' ' * length) # Tạo chuỗi khoảng trắng với độ dài cố định  
    return block_map  
  
# Sử dụng hàm để tạo BLOCK_MAP  
BLOCK_MAP = generate_block_map()
```

Hình 2.2.8: Phương thức `generate_block_map`

Phương thức `generate_block_map()` tạo ra một danh sách gồm các chuỗi ký tự ngẫu nhiên và chuỗi khoảng trắng với mục đích tạo ra một bản đồ lưới gồm 9 hàng, trong đó có 5 hàng chứa các ký tự đại diện cho các khối (dạng số từ 1 đến 7) và 4 hàng khoảng trắng.



Hình 2.2.9: Minh họa tạo map ngẫu nhiên

### III. Mô tả cách chơi BreakOut

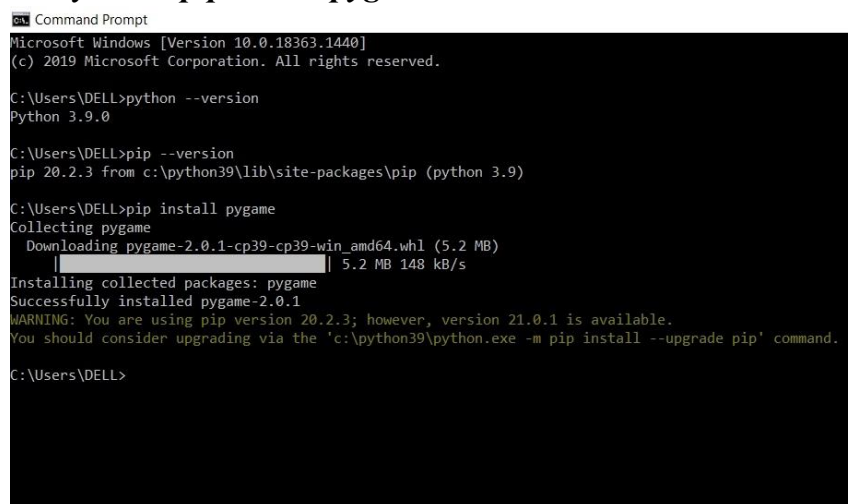
**BreakOut** là một trò chơi điện tử cổ điển thuộc thể loại game đập gạch. Người chơi điều khiển một thanh trượt (paddle) di chuyển ngang dưới màn hình để đỡ một quả bóng. Quả bóng sẽ liên tục di chuyển và va đập vào các viên gạch nằm phía trên. Mục tiêu của trò chơi là dùng quả bóng để phá vỡ hết các viên gạch mà không để bóng rơi ra khỏi màn hình, người chơi cũng có thể sử dụng các vật phẩm nâng cấp được rơi ra khi phá các khối, những vật phẩm này sẽ hỗ trợ người chơi có thể chơi game một cách nhanh hơn, thú vị hơn. Mỗi khi bóng đập vào một viên gạch, viên gạch sẽ biến mất và người chơi ghi điểm.

Người chơi phải khéo léo di chuyển thanh trượt để đỡ quả bóng, không để bóng rơi ra ngoài. Nếu người chơi để bóng rơi, họ sẽ mất một mạng. Trò chơi tiếp tục cho đến khi tất cả các viên gạch bị phá hủy hoặc người chơi mất hết mạng. BreakOut kết hợp giữa kỹ năng phản xạ và chiến lược trong việc lựa chọn góc đập bóng để phá vỡ nhiều gạch nhất có thể.

### IV. Cài đặt Project

Project được thực hiện bằng ngôn ngữ Python và được chạy trên **Visual Studio Code**. Để chạy được chương trình này cần phải có thêm thư viện hỗ trợ Pygame. Đây là thư viện hỗ trợ rất nhiều cho dự án và đặc biệt tạo giao diện cho người chơi tương tác với với Game.

Cách cài đặt Pygame: Pygame có thể được cài đặt dễ dàng bằng công cụ quản lý gói pip trong Python. Bạn chỉ cần mở terminal (hoặc Command Prompt trên Windows) và chạy lệnh **“pip install pygame”**



```
Microsoft Windows [Version 10.0.18363.1440]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\DELL>python --version
Python 3.9.0

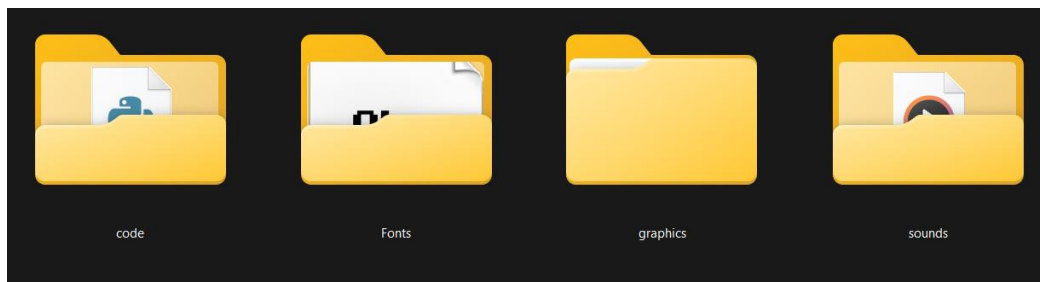
C:\Users\DELL>pip --version
pip 20.2.3 from c:\python39\lib\site-packages\pip (python 3.9)

C:\Users\DELL>pip install pygame
Collecting pygame
  Downloading pygame-2.0.1-cp39-cp39-win_amd64.whl (5.2 MB)
    |#####| 5.2 MB 148 kB/s
Installing collected packages: pygame
Successfully installed pygame-2.0.1
WARNING: You are using pip version 20.2.3; however, version 21.0.1 is available.
You should consider upgrading via the 'c:\python39\python.exe -m pip install --upgrade pip' command.

C:\Users\DELL>
```

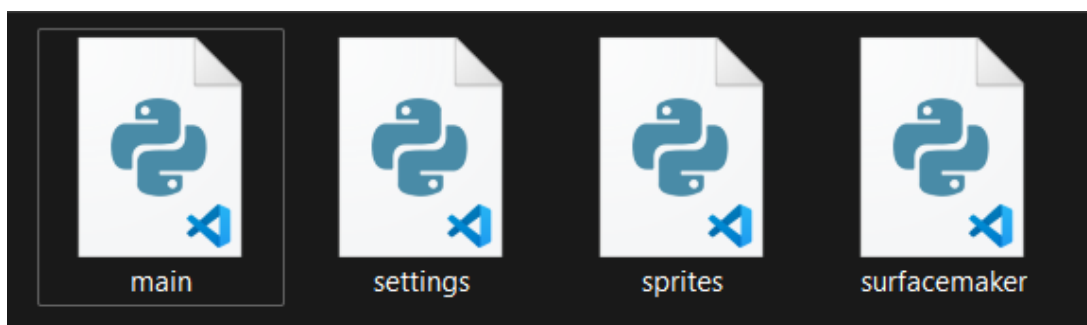
*Hình 3.1: Minh họa cài đặt Pygame*

Khi cài đặt Project cần phải có đầy đủ 4 folder code, Fonts, graphics, sounds



*Hình 3.2: 4 folder của project*

Folder code chứa các file Python của chương trình (main.py, settings.py, sprites.py, surfacemaker.py ). Chương trình sẽ hoạt động khi khởi động chạy file main.py

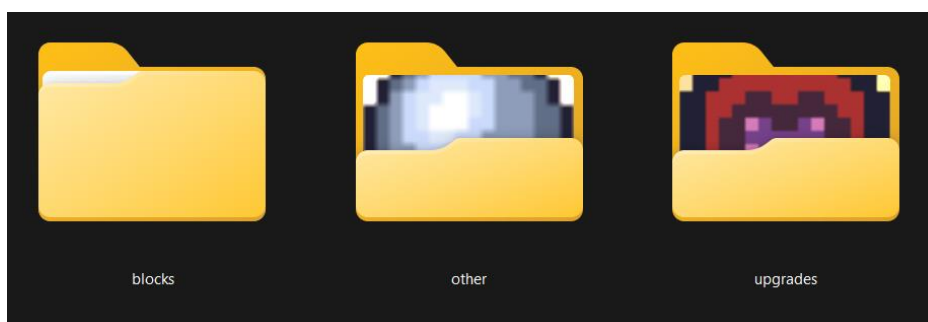


*Hình 3.3: 4 file của Folder code*

Folder Fonts chứa font chữ chính của Game

Folder sounds chứa các âm thanh hiệu ứng cho game.

Folder graphics chứa các hình ảnh đồ họa của các đối tượng như block, paddle, ball, hearts,...



*Hình 3.4: 3 folder chứa đồ họa của Game*

**V. Tài liệu tham khảo**

[1] [https://en.wikipedia.org/wiki/Breakout\\_\(video\\_game\)](https://en.wikipedia.org/wiki/Breakout_(video_game)) (Wikipedia contributors,12/7/2024, Breakout)

[2] Đại học trực tuyến FUNiX. (2023, November 1). *Pygame là gì? Lợi ích khi lập trình game với Pygame*. Học Trục Tuyển CNTT, Học Lập Trình Từ Cơ Bản Đến Nâng Cao.

[3] <https://viettuts.vn/> (*Học Lập Trình Online - Chia sẻ bài học miễn phí - VietTuts.Vn*, VietTuts. )

[4] <https://www.youtube.com/watch?v=4tVC1vhxiao&t=10634s> (Clear Code, 26/3/2022, *Breakout in python*)

**VI. Phân chia công việc**

Thành viên	Mã số	Nhiệm vụ
Vĩnh Thuận (Nhóm trưởng)	64132409	- Logic game (Player, Ball, Block) - Đồ họa, âm thanh - Viết báo cáo
Phạm Tuấn Kiệt	64131060	- Các vật phẩm nâng cấp - Giao diện game, âm thanh - Viết báo cáo
Khối lượng công việc của cả hai là như nhau		