HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN 1

_____ o0o _____

BÀI TẬP LỚN

Đề tài: Lập trình Game bằng framework Pygame

Môn học: Lập trình Python
Số thứ tự nhóm: 01

Nguyễn Đăng Huy	**MSSV: B21DCCN062**
Ngọ Văn Trọng	**MSSV: B21DCCN726**
Nguyễn Thị Lan	**MSSV: B21DCCN818**
Phạm Thế Anh	**MSSV: B21DCCN009**
Lê Bá Quang	**MSSV: B21DCCN624**

Giảng viên hướng dẫn: Ths. Nguyễn Trọng Khánh

HÀ NỘI, 11/2023

LỜI CẢM ƠN

Lời cảm ơn của nhóm sinh viên tới thầy đã giúp đỡ và hướng dẫn chúng em trong định hướng các phương án bài tập lớn để nhóm chúng em có thể thực hiện hoàn thành Bài tập lớn Môn Lập trình Python

TÓM TẮT NỘI DUNG ĐỀ TÀI

Với đề tài chính của nhóm chúng em là lập trình 1 game cơ bản với thể loại là Platform. Xoay quanh đề tài, chúng em sử dụng framework Pygame là công cụ chính để hiện thực hóa được ý tưởng của nhóm.

Tựa game của nhóm chúng em vạch ra là một trò chơi Platform, với nguồn cảm hứng là những tựa game như Super Mario Series với phong cách chơi đơn giản hóa nhất, Celeste với yếu tố khám phá và phong cách thiết kế màn chơi, Hollow Knight với yếu tố điều khiển người chơi một cách linh hoạt với nhiều chức năng như lướt, trượt tường,... Với ý tưởng ban đầu khi chúng em muốn tạo ra trò chơi là một thế giới Fantasy được đặt bối cảnh xung quanh một chàng Ronin (được lấy cảm hứng từ nhân vật Sekiro trong trò chơi Sekiro: Shadows Die Twice) và độ khó của trò chơi một phần được dựa cảm hứng trên thể loại Soul-like, với độ khó cao đòi hỏi người chơi không phạm sai lầm khi chơi. Bên cạnh đó, giao diện thân thiện, dễ dàng điều khiển, mang đến trải nghiệm thú vị mà người chơi có thể thưởng thức một cách thoải mái.

Để hiện thực hóa được ý tưởng trên, thì sự lựa chọn của Pygame là một quyết định xuất sắc, vì nó cung cấp một nền tảng mạnh mẽ và linh hoạt để phát triển trò chơi với đồ hoạ và gameplay mạnh mẽ. Qua đó, việc lấy cảm hứng từ nhân vật Ronin, một hình ảnh của sự kiên định, sức mạnh và khả năng kiểm soát linh hoạt, và kết hợp với tiềm năng của Pygame, trò chơi sẽ có một phong cách nghệ thuật độc đáo cùng với việc tạo ra các môi trường đa dạng và phong phú, kèm theo đó bối cảnh được xây dựng sẽ kết hợp giữa yếu tố giả tưởng, mang đến cho người chơi cảm giác khám phá thú vị và đầy lôi cuốn. Một trong những yếu tố quan trọng nữa là việc tạo ra giao diện thân thiện và dễ dàng điều khiển. Điều này giúp người chơi dễ dàng tiếp cận trò chơi và tận hưởng trải nghiệm một cách thoải mái, không bị rối mắt hoặc gặp khó khăn trong việc tương tác với game. Cuối cùng, việc phát triển một trò chơi platform đòi hỏi sự cân nhắc cẩn thận giữa độ khó và trải nghiệm người chơi. Sự kết hợp giữa yếu tố thách thức cao cùng với một môi trường thân thiện, đẹp mắt và gameplay sáng tạo sẽ là chìa khóa quan trọng để thu hút và giữ chân người chơi trong thế giới game của bạn.

ABSTRACT

The main focus of our group is to develop a basic game in the Platform genre. To materialize this idea, we have chosen Pygame as the primary tool to execute our concept.

Our game project is inspired by titles such as the Super Mario Series, known for its simplified gameplay style, Celeste, which incorporates exploratory elements and unique level design, and Hollow Knight, celebrated for flexible player control and various functionalities like sliding and wall jumping. Initially, our idea revolves around creating a Fantasy world centered on a Ronin character, drawing inspiration from the protagonist Sekiro in the game Sekiro: Shadows Die Twice. The game's difficulty is partially influenced by the "Soul-like" genre, demanding players to perform flawlessly, offering a high level of challenge. Moreover, we aim to provide an intuitive user interface, allowing players to comfortably enjoy the engaging experience.

In order to bring this vision to life, the selection of Pygame is an excellent decision. This framework offers a robust and flexible platform for game development, empowering us to create powerful graphics and gameplay elements. Drawing inspiration from the Ronin, symbolizing determination, strength, and agile control, and leveraging the potential of Pygame, our game will feature a unique artistic style. The diverse and richly designed environments, blending fantasy elements, will captivate players and encourage a sense of thrilling exploration.

Another crucial aspect is to develop a user-friendly interface and easy game controls. This accessibility ensures players can effortlessly engage with the game and enjoy a comfortable experience, without experiencing visual clutter or interaction difficulties.

Ultimately, developing a platform game requires a delicate balance between challenge and player experience. The combination of high-level challenge with an attractive and friendly environment, coupled with innovative gameplay, will be the key to captivating and retaining players within our game world.

MỤC LỤC

CHƯƠNG 1. GIỚI THIỆU CÁCH CHƠI

1.1 Giới thiệu chung

1.1.1 Màn hình bắt đầu

Trò chơi bắt đầu bằng một Intro, sau đấy sẽ trò chơi sẽ chuyển đến trạng thái Menu. Khi ở Menu, người chơi có thể tùy chọn các chế độ như New Game (Chơi mới), Edit Map (Tự tạo map theo cá nhân), Custom Play (Nếu như người chơi đã tạo ra một map riêng bằng chế độ Edit Map thì nút chọn chế độ này sẽ hiện ra để người chơi có thể truy cập map riêng để chơi)

Hình 1.1: Intro trò chơi

Hình 1.2: Menu trò chơi

1.2 Bắt đầu trò chơi

Khi người chơi chạy New Game, trò chơi sẽ chạy màn chơi đầu tiên. Lúc này người chơi sẽ được hướng dẫn chơi qua các hình ảnh để dễ nắm rõ trò chơi

Hình 1.3: Hướng dẫn di chuyển

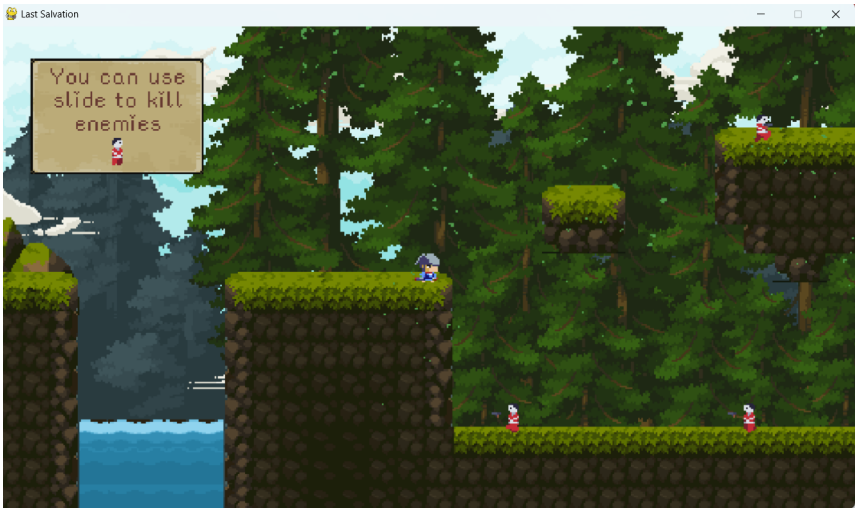
Hình 1.4: Hướng dẫn nhảy

Như các trò chơi Platform thông thường, tựa game của nhóm cũng đưa ra cho người chơi một mục tiêu là điều khiển nhân vật trong game vượt qua các chướng ngại vật trong game bằng cách leo trèo, trượt tường, lướt,... tiêu diệt hết kẻ thù và chuyển sang màn mới.

Kẻ thù trong game được đưa vào như là một công cụ giúp game trở nên thử thách hơn, với các kẻ địch là những "Samurai màu đỏ" với khả năng bắn từ xa, đòi hòi người chơi phải khéo léo hơn khi đi màn. Bởi vì mỗi lần người chơi chết, thì trò chơi sẽ tự động reset mọi thứ trong màn chơi và đưa người chơi về vạch xuất phát - vốn dĩ trò chơi không có điểm checkpoint, vì nhóm chúng em muốn tạo cho tựa game một cảm giác khá giống với dòng "Soul-like" để tựa game đòi hỏi người chơi phải khéo léo và thực sự dùng kỹ năng để có thể vượt qua được màn chơi.

Hình 1.5: Hướng dẫn trượt tường

Hình 1.6: Hướng dẫn lướt

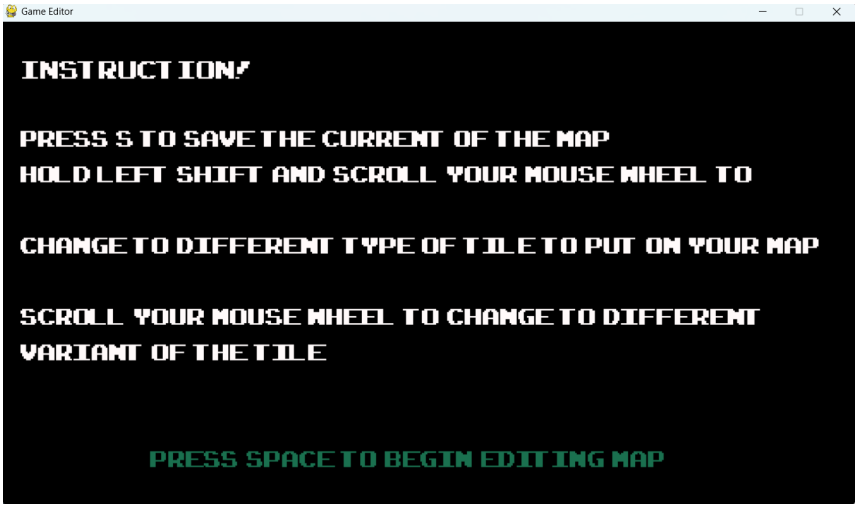


Hình 1.7: Kẻ thù trong game

1.3 Tự thiết kế màn chơi

Trò chơi cho phép người chơi có thể thoải mái tự tạo màn chơi theo sở thích mong muốn, ngoài thúc đẩy khả năng chơi lại của trò chơi của nhóm, đồng thời chức năng này sẽ cho phép người chơi có thể thỏa sức sáng tạo để tạo ra các map chơi cá nhân nhằm tự thử thách bản thân.

Khi bắt đầu chế độ chơi, trò chơi sẽ mở một danh mục hướng dẫn qua về cách chế độ này hoạt động như nào để người chơi có thể hiểu rõ cách hoạt động và dễ dàng sử dụng chế độ Edit Map



Hình 1.8: Hướng dẫn thao tác nút Edit Map

Hình 1.9: Edit Map

Ở chế độ này người chơi sẽ được đặt các khối có sẵn trong tài nguyên của trò chơi, như: đất, cỏ, nước, cây cối, đá trang trí, đá có rêu trang trí,...Với các thao tác cơ bản như: nhấn chuột trái để đặt khối, nhấn nút G để thoải mái đặt các khối theo tọa độ số thực, nhấn nút S để lưu màn chơi, lăn chuột để thay đổi các kiểu khối khác của cùng một loại khối hiện tại để đặt lên màn chơi(ví dụ như thay thành các

loại khối cỏ khác để tạo sự đa dạng trong hình ảnh), giữ Shift và lăn chuột để thay đổi các loại khối khác để đặt lên màn chơi (ví dụ thay đổi từ khối cỏ thành khối đá).

1.4 Một số chức năng khác

Trò chơi có một số chức năng như: tạm dừng trò chơi, bật tắt nhạc, hiệu ứng chuyển cảnh, hiệu ứng trò chơi (như hiệu ứng nhân vật lướt, hiệu ứng tia đạn, hiệu ứng chết, hiệu ứng lá rơi, mây,...) hệ thống mô phỏng chuyển động ở background, hệ thống hướng dẫn cách chơi cho người chơi, hệ thống chuyển màn chơi, hệ thống cắt cảnh, hệ thống vật lý, hệ thống tilemap, và hệ thống chức năng hỗ trợ Developer phát triển game,...

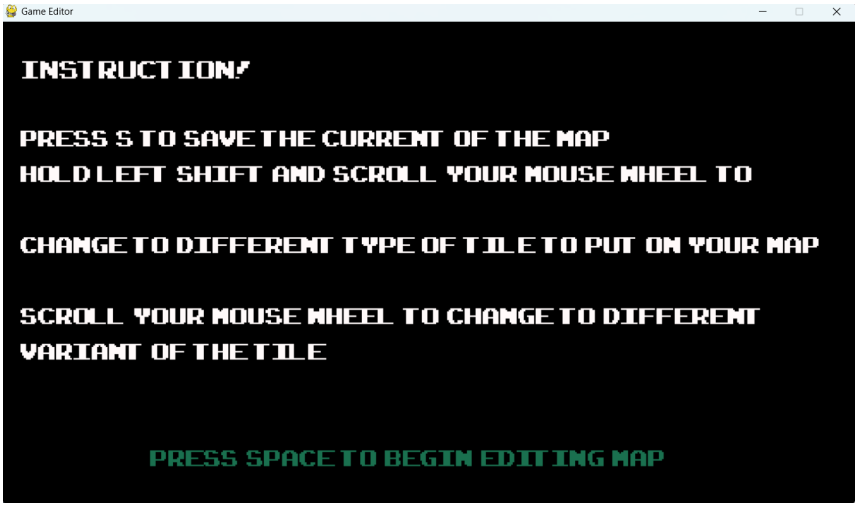
CHƯƠNG 2. MÔ HÌNH CODE

Để hiểu được bản chất của cách trò chơi hoạt động như nào, chúng ta sẽ đi sâu vào cách các mã nguồn hoạt động ra sao và bản chất của trò chơi là như nào

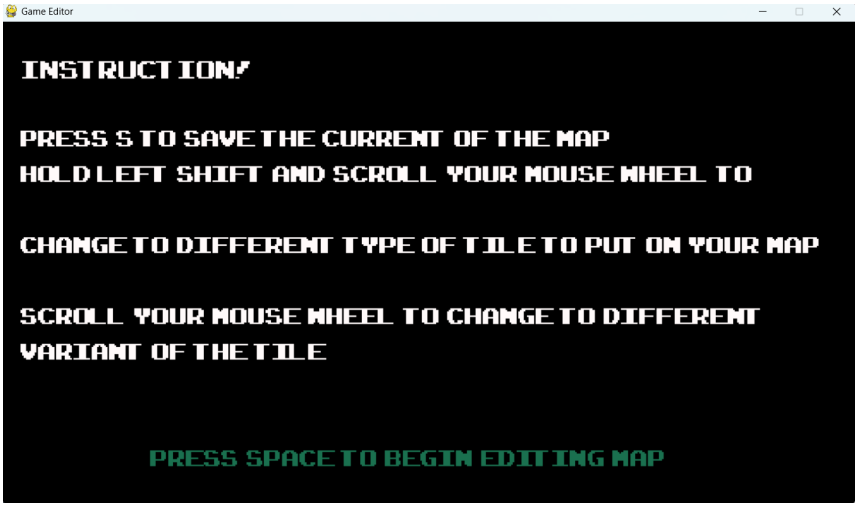
Những mục sau sẽ được trình bày theo thứ tự cách trò chơi của nhóm chúng em được phát triển

2.1 Chuẩn bị tài nguyên trò chơi

Tài nguyên trò chơi bao gồm: Hình ảnh, hiệu ứng âm thanh, font chữ. Các tài nguyên hình ảnh sẽ bao gồm: Hình ảnh background, hình ảnh cho hệ thống GUI, hình ảnh cho các vật thể như nhân vật người chơi và địch, hình ảnh cho hiệu ứng trò chơi,...Đa phần hình ảnh nhóm chúng em sử dụng là đa phần được tự vẽ để phù hợp với phong cách hình ảnh của trò chơi.



Hình 2.1: Tài nguyên trò chơi



Hình 2.2: Cấu trúc tài nguyên hình ảnh trò chơi

2.2 Tạo ra cửa sổ trò chơi

Với việc sử dụng framework Pygame cho phép việc xuất ra một màn hình có khả năng tương tác để nhóm có thể làm việc và xuất game ra cửa sổ màn hình đấy

```
26  class Game:
27      def __init__(self, mode):
28          pygame.init()
29
30          pygame.display.set_caption('Last Salvation')
31          self.screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
32
33          self.display = pygame.Surface((DISPLAY_WIDTH, DISPLAY_HEIGHT), pygame.SRCALPHA)
34
35          self.clock = pygame.time.Clock()
36
```

Hình 2.3: Chương trình khởi tạo cửa sổ

2.3 Xây dựng cấu trúc lớp thực thể, và nền tảng vật lý của trò chơi

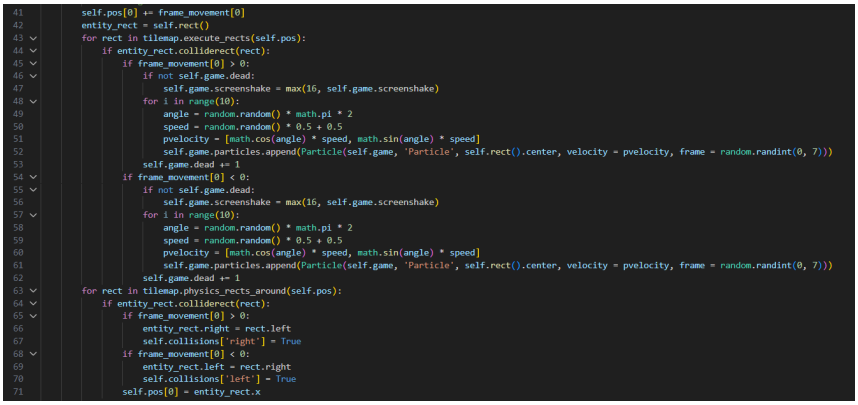
Sau khi chúng ta đã có cửa sổ để làm việc, chúng ta sẽ xây dựng nền tàng cơ bản nhất của một trò chơi, đấy chính là Gameplay. Với thể loại trò chơi là Platform, nền tàng cơ bản cần xây dựng ngay lúc này là nhân vật chơi và vật lý của trò chơi.

```
9   class PhysicsEntity:
10      def __init__(self, game, e_type, pos, size):···
25
26      def rect(self):···
28
29      def set_action(self, action):···
33
34      def update(self, tilemap, movement = (0, 0)):···
```

Hình 2.4: Cấu trúc lớp Entities (Thực thể)

Dựa vào khả năng di chuyển khối đối tượng của framework Pygame, nhóm đã có thể mô phỏng việc nhân vật chơi di chuyển bằng cách xoay xung quanh việc thay đổi tọa độ của các khối Rect của Pygame và được kiểm soát qua một List (self.movement) với 2 biến boolean để biết nhân vật có đang di chuyển hay không. Với đại lượng thứ nhất đại diện cho việc di chuyển theo trục Ox và đại lượng thứ hai đại diện cho việc di chuyển theo trục Oy, kèm theo sẽ thêm một số biến để thay đổi các yếu tố khác như List velocity (vận tốc) gồm 2 biến đặc trưng cho việc tốc độ di chuyển theo Ox và cách trọng lực ảnh hưởng đến nhân vật theo Oy.

Với nền tảng trò chơi là một tựa game Platform nên việc quan trọng nhất để thể hiện được tính chất của thể loại trên là di chuyển giữa các mặt phẳng, từ đó trò chơi phải cho phép người chơi có thể chạy nhảy giữa các mặt phẳng của trò chơi. Hệ thống collision được phát triển theo hướng, khi nhân vật chạy qua các khối được cho là vật cản thì nhân vật sẽ không được phép đi qua



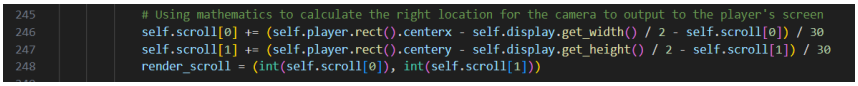
```
        self.pos[0] += frame_movement[0]
        entity_rect = self.rect()
        for rect in tilemap.execute_rects(self.pos):
            if entity_rect.colliderect(rect):
                if frame_movement[0] > 0:
                    if not self.game.dead:
                        self.game.screenshake = max(16, self.game.screenshake)
                    for i in range(10):
                        angle = random.random() * math.pi * 2
                        speed = random.random() * 0.5 + 0.5
                        pvelocity = [math.cos(angle) * speed, math.sin(angle) * speed]
                        self.game.particles.append(Particle(self.game, 'Particle', self.rect().center, velocity = pvelocity, frame = random.randint(0, 7)))
                    self.game.dead += 1
                if frame_movement[0] < 0:
                    if not self.game.dead:
                        self.game.screenshake = max(16, self.game.screenshake)
                    for i in range(10):
                        angle = random.random() * math.pi * 2
                        speed = random.random() * 0.5 + 0.5
                        pvelocity = [math.cos(angle) * speed, math.sin(angle) * speed]
                        self.game.particles.append(Particle(self.game, 'Particle', self.rect().center, velocity = pvelocity, frame = random.randint(0, 7)))
                    self.game.dead += 1
        for rect in tilemap.physics_rects_around(self.pos):
            if entity_rect.colliderect(rect):
                if frame_movement[0] > 0:
                    entity_rect.right = rect.left
                    self.collisions['right'] = True
                if frame_movement[0] < 0:
                    entity_rect.left = rect.right
                    self.collisions['left'] = True
                self.pos[0] = entity_rect.x
```

Hình 2.5: Vật lý nền tảng của trò chơi

2.4 Góc nhìn của người chơi

Sau khi các nền tảng của thể loại Platform đã hoàn thành, trò chơi sẽ cần một hệ thống Camera để đưa góc nhìn của người chơi theo nhân vật, để người chơi có thể trải nghiệm được một màn chơi theo hướng di chuyển từ vị trí A-Z.

Bởi vì vốn dĩ Pygame không có hỗ trợ tạo ra các chức năng Camera như các nền tảng Engine đồ họa khác như Unity hay Unreal Engine, nên thứ để chúng ta có thể có sát với chức năng đó, chính là tạo ra một ảo ảnh như cửa sổ trò chơi đang di chuyển theo nhân vật. Để làm được điều đó, chúng ta sẽ cho các khối tile di chuyển ngược với hướng di chuyển hiện tại của người chơi (cụ thể hơn là nếu người chơi di chuyển phải và tọa độ của người chơi có thay đổi, thì dựa vào tọa độ của người chơi được thay đổi như nào thì các khối ở môi trường xung quanh sẽ thay đổi ngược lại)



```
            # Using mathematics to calculate the right location for the camera to output to the player's screen
            self.scroll[0] += (self.player.rect().centerx - self.display.get_width() / 2 - self.scroll[0]) / 30
            self.scroll[1] += (self.player.rect().centery - self.display.get_height() / 2 - self.scroll[1]) / 30
            render_scroll = (int(self.scroll[0]), int(self.scroll[1]))
```

Hình 2.6: Camera

2.5 Tối ưu hóa

Python, dù là một ngôn ngữ lập trình mạnh mẽ, nhưng thường gặp phải vấn đề về tốc độ thực thi khi sử dụng cùng với framework Pygame. Tốc độ thực thi chậm này có thể dẫn đến hiện tượng giật khung hình và giảm hiệu suất xử lý trong các trò chơi.

Để xử lý vấn đề này, hai giải pháp cơ bản và hợp lý nhất là tối ưu hóa mã nguồn và tải trước tài nguyên trò chơi (Preloading game resources).

Trước hết, việc tối ưu hóa mã nguồn đòi hỏi việc kiểm tra và điều chỉnh mã nguồn trò chơi. Bằng cách loại bỏ những hoạt động không cần thiết, tối ưu cấu trúc dữ liệu, và tránh các phép toán phức tạp không cần thiết, chúng ta có thể cải thiện hiệu suất của trò chơi. Thứ hai, việc tải trước tài nguyên trò chơi là một cách hiệu

```python
BASE_IMG_PATH = 'Data/Images/'

def load_transparency_image(path):
    img = pygame.image.load(BASE_IMG_PATH + path).convert_alpha()
    img.set_colorkey((0, 0, 0))
    return img

def load_image_no_cv(path):
    img = pygame.image.load(BASE_IMG_PATH + path)
    img.set_colorkey((0, 0, 0))
    return img

def load_image(path):
    img = pygame.image.load(BASE_IMG_PATH + path).convert()
    img.set_colorkey((0, 0, 0))
    return img

def load_images(path):
    images = []
    for img_name in sorted(os.listdir(BASE_IMG_PATH + path)):
        images.append(load_image(path + '/' + img_name))
    return images

class Animation:
    def __init__(self, images, img_dur = 5, loop = True):
        self.images = images
        self.loop = loop
        self.img_duration = img_dur
        self.done = False
        self.frame = 0

    def copy(self):
        return Animation(self.images, self.img_duration, self.loop)

    def update(self):
        if self.loop:
            self.frame = (self.frame + 1) % (self.img_duration * len(self.images))
        else:
            self.frame = min(self.frame + 1, self.img_duration * len(self.images) - 1)
            if self.frame >= self.img_duration * len(self.images) - 1:
                self.done = True

    def img(self):
        return self.images[int(self.frame / self.img_duration)]
```

Hình 2.7: Preloading Game Resources

quả để giảm bớt tải của Pygame trong quá trình chơi. Bằng cách tải trước các hình ảnh, âm thanh, và các tài nguyên khác cần thiết, chúng ta có thể giảm thời gian xử lý khi trò chơi đang diễn ra Kết hợp cả hai giải pháp này có thể giúp cải thiện hiệu suất và trải nghiệm người chơi trong trò chơi được phát triển bằng Python và Pygame.

2.6 Level Editor

Đây là tiền đề ban đầu của chế độ Edit Map sau này, ban đầu chức năng này được viết ra để dễ dàng thiết kế một Tilemap (một bản đồ khối dưới dạng một file .json chứa một dictionary để từ đó Pygame có thể đọc và lấy được thông tin phải render những khối nào ở những vị trí nào với mục đích cụ thể)

Tất cả các công cụ thiết kế trực tuyến bản đồ tile như Tiled, Tile Editor, Aseprite, thường xuất dữ liệu dưới định dạng CSV. Tuy nhiên, sử dụng CSV để render bản đồ trong Pygame có nhược điểm. Để vượt qua hạn chế này, nhóm đã đưa ra một giải pháp thông qua việc xây dựng một lớp Tilemap đặc biệt. Lớp Tilemap này không chỉ giúp vẽ bản đồ một cách linh hoạt mà còn lưu trữ dữ liệu dưới định dạng tệp

```
# Game assets
self.assets = {
    # Game graphic asset
    'Barrier': load_images('Tiles/Barrier'),
    'Grass': load_images('Tiles/Grass'),
    'Water': load_images('Tiles/Water'),
    'Stone': load_images('Tiles/Stone'),
    'Stone_Surface': load_images('Tiles/Stone Surface'),
    'Spike': load_images('Tiles/Spike'),
    'Clouds': load_images('Tiles/Decor/Clouds'),
    'Background': load_image('Background/Background.png'),
    'Forest_Background': load_image('Background/Background_1.png'),
    'Cave_Background': load_image('Background/Cave_Background.png'),
    'Decor_Grasses': load_images('Tiles/Decor/Grasses'),
    'Decor_Rock': load_images('Tiles/Decor/Prop Rock'),
    'Decor_Mossy_Rock': load_images('Tiles/Decor/Prop Mossy Rock'),
    'Decor_Tree': load_images('Tiles/Decor/Tree'),
    # Player's asset
    'Player/Idle': Animation(load_images('Entities/Player/Idle'), img_dur = 10),
    'Player/Run': Animation(load_images('Entities/Player/Run'), img_dur = 5),
    'Player/Jump': Animation(load_images('Entities/Player/Jump')),
    'Player/Wall_Slide': Animation(load_images('Entities/Player/Wall Slide')),
    # Enemies's asset
    'Enemy/Idle': Animation(load_images('Entities/Enemy/Idle'), img_dur = 6),
    'Enemy/Run': Animation(load_images('Entities/Enemy/Run'), img_dur = 4),
    'Gun': load_image('gun.png'),
    'Projectile': load_image('projectile.png'),
    # Particles
    'Particle/Leaf': Animation(load_images('Particles/Leaf'), img_dur = 20, loop = False),
    'Particle/Particle': Animation(load_images('Particles/Particle'), img_dur = 6, loop = False),
    # Tutorial
    'Tutorial/Moving': Animation(load_images('Tutorial/Moving'), img_dur = 20),
    'Tutorial/Double_Jump': Animation(load_images('Tutorial/Double Jump'), img_dur = 20),
    'Tutorial/Jumping':Animation(load_images('Tutorial/Jumping'), img_dur = 20),
    'Tutorial/Dash': Animation(load_images('Tutorial/Slide'), img_dur = 20),
    'Tutorial/Wall_Slide': Animation(load_images('Tutorial/Wall Slide'), img_dur = 30),
    'Tutorial/Notification_1': load_image('Tutorial/Notification/00.png'),
    'Tutorial/Notification_2': load_image('Tutorial/Notification/01.png'),
    'Tutorial/Notification_3': load_image('Tutorial/Notification/02.png'),
}
```

Hình 2.8: Tối ưu hóa bằng cách Preloading Game Resources

.json. Điều này không chỉ giúp chúng ta vượt qua những hạn chế của CSV mà còn tạo điều kiện linh hoạt và hiệu quả hơn cho việc render bản đồ trong Pygame.

Các chức năng của class Tilemap bao gồm xuất, đọc file .json và kiểm tra va chạm giữa các khối tile, kiểm tra tính chất vật lý của khối tile với nhân vật của người chơi, truyền chức năng tiêu diệt của nhân vật cho các khối tile thích ứng,...

2.7 Hiệu ứng hình ảnh

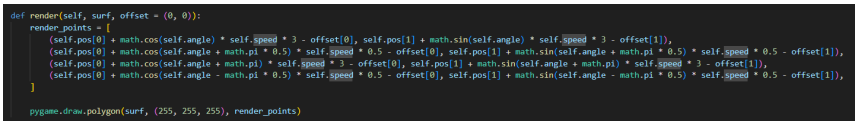
Một trò chơi ngoài gameplay là công cụ chính để tương tác với người chơi ra thì đồ họa cũng là một yếu tố quan trọng không kém để người chơi có thể tận hưởng một trò chơi một cách trọn vẹn nhất có thể. Từ những lý do đó, trò chơi của nhóm cũng đã được cho rất nhiều chức năng về hiệu ứng hình ảnh.

Một số hiệu ứng hình ảnh có trong trò chơi bao gồm: hiệu ứng tóe tia đạn, hiệu ứng lá rơi, hiệu ứng mây, hiệu ứng lướt, hiệu ứng chuyển cảnh,...

```
# Rendering leaf from trees, by using
for rect in self.leaf_spawners:
    if random.random() * 49999 < rect.width * rect.height:
        pos = (rect.x + random.random() * rect.width, rect.y + random.random() * rect.height)
        self.particles.append(Particle(self, 'Leaf', pos, velocity = [-0.1, 0.3], frame = random.randint(0, 20)))
```

Hình 2.9: Particle

Hầu hết các hiệu ứng hình ảnh ở trong trò chơi đều là các hình ảnh Particle có



```
def render(self, surf, offset = (0, 0)):
    render_points = [
        (self.pos[0] + math.cos(self.angle) * self.speed * 3 - offset[0], self.pos[1] + math.sin(self.angle) * self.speed * 3 - offset[1]),
        (self.pos[0] + math.cos(self.angle + math.pi * 0.5) * self.speed * 0.5 - offset[0], self.pos[1] + math.sin(self.angle + math.pi * 0.5) * self.speed * 0.5 - offset[1]),
        (self.pos[0] + math.cos(self.angle + math.pi) * self.speed * 3 - offset[0], self.pos[1] + math.sin(self.angle + math.pi) * self.speed * 3 - offset[1]),
        (self.pos[0] + math.cos(self.angle - math.pi * 0.5) * self.speed * 0.5 - offset[0], self.pos[1] + math.sin(self.angle - math.pi * 0.5) * self.speed * 0.5 - offset[1]),
    ]

    pygame.draw.polygon(surf, (255, 255, 255), render_points)
```

Hình 2.10: Một ví dụ biểu thức toán học dùng để tạo ảo ảnh hiệu ứng

sẵn trong tài nguyên, kết hợp chức năng di chuyển sprite của Pygame để tạo ra các hiệu ứng ảo ảnh.

Ví dụ như hiệu ứng lá được sử dụng phương trình sóng kết hợp thư viện random để tạo ra các con số ngẫu nhiên, từ đó mỗi chiếc lá sẽ rơi theo chu trình sóng và mỗi lá sẽ có một chu trình sóng khác.

Ví dụ khác là hiệu ứng chết của nhân vật được sử dụng phương trình ax + b thông thường nhưng được kết hợp thêm đại lượng tần số góc để tạo ra chu trình hình tròn, từ đó sẽ thành một hình tròn được mở rộng. Tương tự với kiến thức đó, hiệu ứng chuyển cảnh giữa các màn và các game state cũng được dùng phương trình hình tròn để tạo một màn cắt cảnh phóng to thu nhỏ,...

CHƯƠNG 3. KẾT LUẬN

3.1 Kết luận

Trong quá trình phát triển BTL game của nhóm, đã có nhiều vấn đề quan trọng mà nhóm đã xác định và giải quyết thành công. Một trong những thành tựu đáng chú ý là việc tối ưu hóa hiệu suất game, đặc biệt là vấn đề liên quan đến giảm hiện tượng giật hình và cải thiện tốc độ xử lý. Nhóm đã áp dụng các kỹ thuật tối ưu hóa mã nguồn và điều chỉnh quá trình xử lý để tạo ra một trải nghiệm chơi game mượt mà hơn, đồng thời giảm tải cho hệ thống.

Tuy nhiên, một số vấn đề vẫn còn tồn đọng của nhóm. Một trong số đó là vì thời gian để hoàn thành BTL chưa được nhiều nên chưa mở rộng tính năng của trò chơi nhiều hơn để tạo ra một trải nghiệm đa dạng và hấp dẫn hơn cho người chơi. Điều này bao gồm việc thêm các cấp độ mới, tính năng tương tác phong phú và môi trường game đa dạng. Đồng thời, hệ thống AI của kẻ thù còn chưa đủ thông minh để tạo ra một thử thách lớn với người chơi.

3.2 Hướng phát triển trong tương lai

Nếu như trong tương lai được cơ hội và có động lực thúc đẩy phù hợp, nhóm sinh viên của chúng em sẽ đặt ra kế hoạch phát triển hứa hẹn hơn. Nhóm sẽ tập trung vào việc mở rộng phạm vi và tính năng của trò chơi để tạo ra một trải nghiệm chơi game sâu sắc hơn và đa dạng hơn. Kế hoạch bao gồm việc thêm các cấp độ mới, tính năng tương tác phong phú và môi trường game đa dạng hơn.

Đồng thời, chúng tôi đang tích cực nghiên cứu và phát triển hệ thống trí tuệ nhân tạo (AI) để cải thiện trải nghiệm chơi game, tạo ra sự tương tác thông minh hơn và cung cấp trải nghiệm chơi game linh hoạt hơn cho người chơi.

Một mục tiêu quan trọng khác của chúng tôi là tối ưu hóa việc sử dụng tài nguyên, giúp trò chơi hoạt động mượt mà và hiệu quả trên nhiều nền tảng và thiết bị khác nhau, đồng thời giảm tải cho hệ thống.

TÀI LIỆU THAM KHẢO

[1] Jesse N. Schell, “Story and Game Structures can be Artfully Merged with Indirect Control," *Game Design*, vol. 16, pp. 283-293, 2008

[2] R. Nystrom, *Game Programming Patterns*. Genever Benning, 2011.

[3] Katie S. Tekinbas and E. Zimmerman, *Rules of Play: Game Design Fundamentals*. MIT Press, 2003.

[4] Pete Shinners, *pygame – pygame v2.6.0 documentation*. [Online]. Available: `https://www.pygame.org/docs/ref/pygame.html` (visited on 08/11/2023).