=

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

BÁO CÁO BÀI TẬP LỚN
HỌC PHẦN: LẬP TRÌNH PYTHON

ĐỀ TÀI: Dach's Enchanted Journey

Giảng viên:	Vũ Minh Mạnh
Nhóm môn học:	08
Nhóm BTL:	07
Thành viên:	
B22DCCN582	Nguyễn Thị Ngân
B22DCCN056	Trịnh Lê Xuân Bách
B22DCCN681	Lê Trọng Sang
B22DCCN030	Nguyễn Quang Anh

Hà Nội – 2024

PHÂN CÔNG NHIỆM VỤ NHÓM THỰC HIỆN

TT	Công việc / Nhiệm vụ	SV thực hiện	Thời hạn hoàn thành	Mức đóng góp
1	- Tạo bố cục, vẽ bản đồ. - Tạo màn hình Main Menu (có các button Play, Exit), màn hình Game Over. - Xử lý camera cho Player. - Xử lý va chạm giữa Player, Enemies với các vật thể có trên bản đồ. - Tạo và quản lý các đối tượng trên bản đồ (cây cối, nhà cửa, chướng ngại vật, NPC). - Xử lý hệ thống tính điểm, nhận điểm lưu điểm qua từng lượt chơi. - Quản lý trạng thái (thắng, thua, tạm dừng). - Kết nối giữa các đầu việc vào chung 1 hệ thống để game chạy. - Quản lý thông tin người chơi (Lưu và quản lý điểm cộng) - Tìm kiếm và cài đặt âm thanh cho các tác vụ cơ bản của game. - Hệ thống game phụ bản (snake và egg catch) - Xử lý NPC và dialog cho NPC. - Viết báo cáo.	Nguyễn Thị Ngân	15/11/2024	25%
2	- Thiết kế và tìm kiếm sprite cho nhân vật, kẻ địch. - Xử lý animation (chuyển động của nhân vật, quái vật).	Trịnh Lê Xuân Bách	15/11/2024	25%

	- Tạo các kỹ năng, hiệu ứng kỹ năng và cơ chế cooldown. - Xử lý di chuyển của player và quái. - Thiết kế slide.			
3	- Vòng lặp game (game loop). - Các yếu tố gameplay cốt lõi như tấn công, nhận sát thương, và hồi phục. - Thiết kế silde.	Lê Trọng Sang	15/11/2024	25%
4	- Quản lý thông tin người chơi (Tính điểm cộng) - Xử lý hệ thống tính điểm và tạo xu tính điểm cho nhân vật. - Viết báo cáo.	Nguyễn Quang Anh	15/11/2024	25%

NHÓM THỰC HIỆN TỰ ĐÁNH GIÁ

TT	SV thực hiện	Thái độ tham gia
1	Nguyễn Thị Ngân	3
2	Trịnh Lê Xuân Bách	3
3	Lê Trọng Sang	3
4	Nguyễn Quang Anh	3

Ghi chú:

Thái độ tham gia: Đánh giá điểm thái độ tham gia công việc chung của nhóm (từ 0 đến 4).

+ 0: Không tham gia bất kỳ hoạt động nào, không đóng góp, không phản hồi khi được liên hệ.

+ 1: Tham gia rất hạn chế, chỉ có mặt trong một số hoạt động, không đóng góp ý nghĩa, không chủ động.
+ 2: Tham gia cơ bản, hoàn thành một phần nhiệm vụ được giao, nhưng thiếu chủ động và trách nhiệm.
+ 3: Tham gia tích cực, hoàn thành tốt nhiệm vụ, hợp tác hiệu quả với nhóm, chủ động đề xuất ý kiến.
+ 4: Tham gia xuất sắc, đóng vai trò quan trọng, lãnh đạo nhóm, hỗ trợ thành viên khác và thúc đẩy tiến độ chung.

MỤC LỤC

LỜI MỞ ĐẦU

Chúng em xin gửi lời cảm ơn chân thành đến Học viện Công nghệ Bưu chính Viễnthông đã tạo điều kiện cho chúng em tham gia vào môn học Lập trình với Python. Môn học này không chỉ giúp chúng em nắm vững các kiến thức lập trình mà còn trang bị những kỹ năng cần thiết để phát triển các sản phẩm phần mềm, đặc biệt là game.

Chúng em đặc biệt muốn gửi lời cảm ơn đến thầy Vũ Minh Mạnh, giảng viên bộ môn, người đã tận tâm hướng dẫn và chia sẻ những kiến thức quý báu trong suốt học kỳ qua. Những bài giảng của thầy đã giúp chúng em vững vàng hơn trong quá trình phát triển game.

Bài tiểu luận này sẽ trình bày về quá trình phát triển trò chơi "Dach's Enchanted Journey", một game nhập vai 2D (RPG) do nhóm chúng em phát triển. Trong trò chơi, người chơi sẽ cùng nhân vật chính, Dách, một chàng trai thông minh và dũng cảm, tham gia vào hành trình đầy thử thách để giành lấy trái tim của Cẩm Lan – cô gái xinh đẹp và tài năng trong làng. Để chứng minh tình yêu và khả năng của mình, Dách phải vượt qua 5 thử thách khắc nghiệt mà gia đình Lan đặt ra, đồng thời đối mặt với những kẻ thù cạnh tranh và sinh vật huyền bí trong khu rừng già.

Chúng em hy vọng sản phẩm này sẽ phản ánh được sự nỗ lực và đam mê của nhóm, đồng thời cũng là một ứng dụng thực tế cho những kiến thức đã học trong môn Lập trình với Python. Chúng em mong nhận được những góp ý và phản hồi từ thầy để có thể hoàn thiện sản phẩm hơn nữa.

Trân trọng.

Chương 1. Tổng quan và giới thiệu về Dach's Enchanted Journey

1.1Giới thiệu

a. Mô tả:

- Tên dự án: Dach's Enchanted Journey
- Link dự án**:** https://github.com/chimcuccuccu/Dachs-Enchanted-Journey
- Thể loại: Top-down
- Lối chơi: Tính điểm, di chuyển xuyên qua nhiều thế giới, đánh quái, thực hiện các nhiệm vụ, ….
- Ngôn ngữ lập trình: Python
- Framework: Pygame
- Tác giả: Nhóm 7

b. Cốt truyện:

- Dách là một chàng trai thông minh, dũng cảm và tài hoa. Anh sống trong một ngôi làng nhỏ bên cạnh khu rừng già. Tình yêu của anh dành cho Cẩm Lan, cô gái xinh đẹp và tài năng trong làng, đã lớn dần theo năm tháng. Sau một thời gian dài tìm hiểu, Dách quyết định đến nhà Lan để xin phép cha mẹ cô cho hai người được kết hôn.

- Gia đình Lan là một gia đình có truyền thống lâu đời. Họ yêu cầu Dách phải vượt qua 5 thử thách để chứng minh tình yêu và khả năng của mình. Mỗi thử thách đều được ghi lại trong một hòm chứa bí mật cổ xưa, được truyền lại từ đời này sang đời khác trong gia đình.

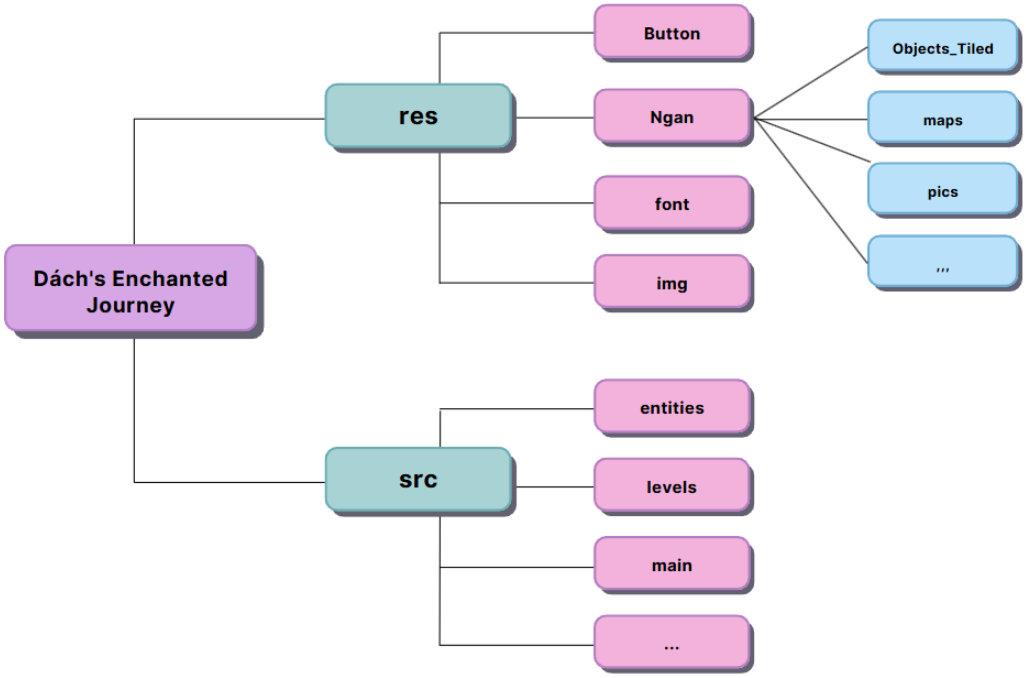
- Trong quá trình tìm kiếm, Dách phải đối mặt với những kẻ thù cạnh tranh, những người cũng muốn lấy được những hòm sính lễ. Họ có thể là những chàng trai khác trong làng, những kẻ xấu xa muốn chiếm đoạt kho báu hoặc những sinh vật huyền bí trong rừng.

c. Gameplay:

- Điều khiển Dach's đi qua khắp khu rừng để tìm kiếm các hòm sính lễ, chiến đấu với kẻ thù, làm nhiệm vụ khi mở các hòm bí mật để thu thập chúng.

1.2 Kiến trúc và các tính năng của game

- Cấu trúc dự án:



Sơ đồ các thư folder chính của dự án

Diễn giải:

- *res:* Chứa các tài nguyên như hình ảnh, âm thanh, map.
- *src:* Chứa các class chính và các hàm chức năng của game.

1.3 Kết chương

- Chương này đã giới thiệu tổng quan về trò chơi *Dach's Enchanted Journey*, bao gồm thông tin cơ bản, cốt truyện và lối chơi. Cốt truyện xoay quanh hành trình vượt qua thử thách của nhân vật chính Dách để chứng minh tình yêu chân thành với Cẩm Lan, kết hợp với lối chơi phiêu lưu và chiến đấu đầy hấp dẫn.
- Ngoài ra, cấu trúc dự án được xây dựng một cách rõ ràng và hợp lý, từ giao diện trực quan, cơ chế chơi mượt mà đến quản lý dữ liệu hiệu quả và tích hợp đồ họa, âm thanh sống động. Sự phân chia hợp lý này giúp đảm bảo tính tổ chức, tối ưu hóa quá trình phát triển và hỗ trợ nhóm phát triển duy trì tiến độ cũng như chất lượng sản phẩm.

- Những nội dung trên không chỉ đặt nền móng vững chắc cho quá trình xây dựng trò chơi mà còn hướng tới mục tiêu tạo ra một sản phẩm hoàn thiện, mang lại trải nghiệm thú vị và đáng nhớ cho người chơi,

Chương 2 . Thiết kế và cài đặt các folder cho dự án

2.1 Khái quát

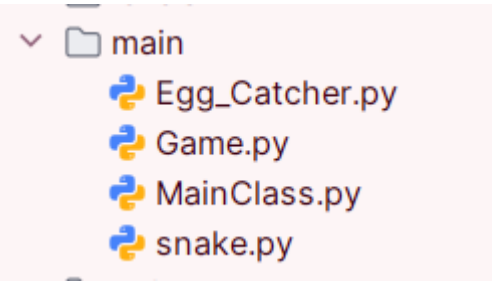
Như đã đề cập trong phần Mở đầu, chương này tập trung vào việc phân tích, thiết kế và tổ chức cấu trúc thư mục cho dự án game. Tôi dự định xây dựng cấu trúc thư mục hợp lý, bao gồm các phần dành cho tài nguyên đồ họa, âm thanh, và mã nguồn để hỗ trợ các chức năng chính của trò chơi.

Thư mục đồ họa sẽ quản lý toàn bộ hình ảnh, sprite, và hiệu ứng hình ảnh, trong khi thư mục âm thanh sẽ lưu trữ các tệp nhạc nền và hiệu ứng âm thanh. Mã nguồn được phân chia rõ ràng nhằm tổ chức các thành phần quan trọng như logic trò chơi, giao diện, và quản lý dữ liệu.

Việc thiết kế cấu trúc này không chỉ giúp dự án dễ dàng mở rộng và bảo trì mà còn đảm bảo tính khoa học và hiệu quả trong quá trình phát triển.

2.2 Thiết kế các folder và các file dữ liệu của dự án

a. main



Các class trong folder main

- MainClass: Chỉ có chức năng là chạy chương trình.

```
6   pygame.init()
7   main_menu()
8
9   g = Game()
10  g.new()
11  while g.running:
12      g.main()
13      g.game_over()
14
15  pygame.quit()
16  sys.exit()
```

Code chính của class MainClass,py

- *Game:* Có chức năng quản lý toàn bộ logic và giao diện trò chơi, từ khởi tạo, cập nhật trạng thái, vẽ màn hình, cho đến xử lý tương tác người chơi. Nó hoạt động như lớp điều phối chính, liên kết các lớp nhỏ hơn (như Player, Enemy, Map) để tạo ra một trò chơi hoàn chỉnh.

```
class Game:  3 usages  Ngan cute hme +2 *
    def __init__(self):...

    def createTilemap(self):...

    def new(self):...

    def events(self):...

    def update(self):...

    def draw(self):...

    def main(self):...

    def game_over(self):...
```

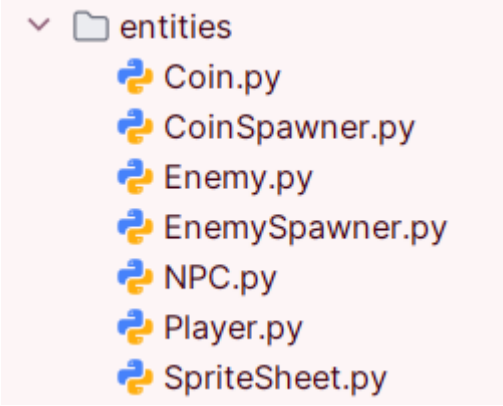
Các hàm đã sử dụng trong class Game

- Tài nguyên đồ họa:
    - Sử dụng các file spritesheet để tải hình ảnh nhân vật, kẻ thù, kỹ năng, đồng xu, và bản đồ.
    - Tạo các đối tượng Spritesheet để quản lý các hình ảnh này.
- Thông tin màn hình:
    - Lấy kích thước màn hình từ pygame.display.Info() để thiết lập giao diện.
- Cài đặt cơ bản:
    - Đồng hồ (clock) để điều chỉnh tốc độ khung hình.
    - Các nhóm sprite (LayeredUpdates) để quản lý nhân vật, kẻ thù, kỹ năng, đồng xu, và các đối tượng khác.
- Các phương thức chính:
- createTilemap():
    - Mục đích: Tạo bản đồ và khởi tạo các đối tượng trong trò chơi.
    - Chi tiết:
        - Tải bản đồ từ file .tmx bằng pytmx.
        - Tạo nhân vật chính Player, các kẻ thù (EnemySpawner), đồng xu (CoinSpawner), và các đối tượng khác như hộp (Box) và cửa (Door).
        - Thêm các đối tượng vào các nhóm visible_sprites và all_sprites.
- events()
    - Mục đích: Xử lý các sự kiện do người chơi hoặc hệ thống tạo ra.
    - Chi tiết:
        - Thoát trò chơi khi người chơi nhấn QUIT.
        - Kích hoạt các kỹ năng:
        - Tấn công thường (Attack): Kích hoạt khi nhấn phím SPACE.
        - Tấn công lửa (AttackFire): Kích hoạt khi nhấn phím H.
        - Hồi máu (Heal): Kích hoạt khi nhấn phím C.
- update()
    - Mục đích: Cập nhật trạng thái của tất cả các sprite trong trò chơi.
    - Chi tiết: Gọi phương thức update() cho từng sprite trong all_sprites.
- draw()
    - Mục đích: Vẽ toàn bộ nội dung lên màn hình.
    - Chi tiết:
        - Vẽ các sprite trong visible_sprites thông qua phương thức

custom_draw().

    - Hiển thị thanh máu của nhân vật, các biểu tượng cooldown kỹ năng, và bảng điểm
- main()
    - Mục đích: Chạy vòng lặp chính của trò chơi.
    - Chi tiết: Liên tục lặp qua các bước: xử lý sự kiện (events()), cập nhật (update()), và vẽ (draw()).
- Các tính năng dễ thấy và nổi bật
    - Quản lý sprite:
        - Dùng nhóm LayeredUpdates để tổ chức và cập nhật các đối tượng.
        - Sử dụng YSortCameraGroup để điều chỉnh camera và vẽ sprite theo thứ tự lớp (layer).
    - Hệ thống kỹ năng: Liên tục lặp qua các bước: xử lý sự kiện (events()), cập nhật (update()), và vẽ (draw()).
        - Tích hợp các kỹ năng tấn công thường, tấn công lửa, và hồi máu.
        - Cơ chế cooldown đảm bảo cân bằng trò chơi.
    - Bản đồ động:
        - Tải bản đồ từ file .tmx và tự động sinh kẻ thù, đồng xu, và các đối tượng khác.
- Snake, Egg_Catcher: Minigame cho nhiệm vụ.

b. entities



Các class chính của folder entities

- *Player:* Lớp Player kế thừa từ pygame.sprite.Sprite, đại diện cho nhân vật trong một trò chơi 2D được phát triển bằng Pygame. Lớp này xử lý di chuyển, va chạm, hoạt ảnh và thanh máu cho nhân vật.

```
39    def update(self):...
67
68    def update(self):...
100
101   def movement(self):...
119
120   def collide_coin(self):...
126
127   def check_collisions(self):...
139
140   def animate(self):...
195
196   def heal(self, amount):...
201
      def draw_health_bar(self):...
```

Các hàm đã sử dụng trong class Player

- Sử dụng lớp pygame.sprite.Sprite để kế thừa các chức năng cơ bản của sprite trong Pygame.

```
7   class Player(pygame.sprite.Sprite):  2 usages  Ngan cute hme +1
8       def __init__(self, game, x, y, scale_factor=1.3):  Ngan cute hme +1
9           self.game = game
10          self._layer = PLAYER_LAYER
11          self.groups = self.game.all_sprites
12          self.scale_factor = scale_factor
13          pygame.sprite.Sprite.__init__(self,  *groups: self.groups)
```

Code thể hiện dùng pygame.sprite.Sprite để kế thừa

- Thiết lập:
  - Vị trí ban đầu (x, y): Được xác định trong quá trình khởi tạo.
  - Kích thước (width, height): Cố định dựa trên kích thước ô lưới (TILESIZE).
  - Hình ảnh (image): Lấy từ spritesheet của trò chơi.
  - Hướng di chuyển ban đầu: Mặc định là "down".

```
        self.x = x
        self.y = y

        # Thiết lập kích thước và vị trí
        self.width = TILESIZE
        self.height = TILESIZE

        self.x_change = 0
        self.y_change = 0

        self.facing = 'down'
        self.animation_loop = 1

        # Tải hình ảnh ban đầu cho người chơi
        self.image = self.game.character_spritesheet.get_sprite(0, 0, self.width, self.height)
        self.rect = self.image.get_rect()
        self.rect.x = self.x
        self.rect.y = self.y

```

- Các phương thức chính
  - update()

```python
def update(self):  Ngan cute hme +1
    self.movement()
    self.animate()
    self.collide_coin()
    self.collider.collide_enemy()
    self.draw_health_bar()

    # Cập nhật vị trí người chơi
    self.rect.x += self.x_change
    for collidable in self.game.collidables:
        if self.rect.colliderect(collidable.rect):
            if self.x_change > 0:  # Moving right
                self.rect.right = collidable.rect.left
            elif self.x_change < 0:  # Moving left
                self.rect.left = collidable.rect.right
            break  # Stop checking further for efficiency

    self.rect.y += self.y_change
    for collidable in self.game.collidables:
        if self.rect.colliderect(collidable.rect):
            if self.y_change > 0:  # Moving down
                self.rect.bottom = collidable.rect.top
            elif self.y_change < 0:  # Moving up
                self.rect.top = collidable.rect.bottom
            break  # Stop checking further for efficiency

    self.x_change = 0
    self.y_change = 0
```

Code của hàm update() trong class player

- Cập nhật trạng thái của Player qua:
  - Di chuyển (movement()): Xử lý logic di chuyển dựa trên phím bấm.
  - Hoạt ảnh (animate()): Chuyển đổi hình ảnh dựa trên hướng di chuyển.
  - Va chạm:
  - Đồng xu: Xóa đồng xu và tăng điểm khi thu thập.
  - Kẻ thù: Gọi phương thức collider.collide_enemy().
  - Vật thể: Ngăn nhân vật đi xuyên qua.
  - Hiển thị thanh máu: Gọi draw_health_bar().
- movement()

```
101     def movement(self):  2 usages  Ngan cute hme
102         keys = pygame.key.get_pressed()
103         if keys[pygame.K_LEFT]:
104             if self.rect.left > 0:
105                 self.x_change -= PLAYER_SPEED
106                 self.facing = 'left'
107         if keys[pygame.K_RIGHT]:
108             if self.rect.right < self.game.map_width:
109                 self.x_change += PLAYER_SPEED
110                 self.facing = 'right'
111         if keys[pygame.K_UP]:
112             if self.rect.top > 0:
113                 self.y_change -= PLAYER_SPEED
114                 self.facing = 'up'
115         if keys[pygame.K_DOWN]:
116             if self.rect.bottom < self.game.map_height:
117                 self.y_change += PLAYER_SPEED
118                 self.facing = 'down'
119
```

Code xử lý điều khiển từ bàn phím

- Xử lý điều khiển di chuyển dựa trên các phím LEFT, RIGHT, UP, và DOWN.
- Cập nhật hướng (facing) và tọa độ thay đổi.

- animate(): Chuyển đổi hình ảnh dựa trên hướng (facing) và trạng thái di chuyển.

```
    if self.facing == 'down':
159     if self.y_change == 0:
160         self.image = self.game.character_spritesheet.get_sprite(64, 0, self.width, self.height)
161     else:
162         self.image = down_animations[math.floor(self.animation_loop)]
163         self.animation_loop += 0.1
164         if self.animation_loop >= 3:
165             self.animation_loop = 1
166
```

- Sử dụng danh sách ảnh động (animations) cho từng hướng: trái, phải, lên, xuống. Tương tự với các hướng còn lại.

```
140     def animate(self):  2 usages   Ngan cute hme
141         # Các ảnh động dựa trên hướng di chuyển
142         down_animations = [self.game.character_spritesheet.get_sprite(64, 0, self.width, self.height),
143                            self.game.character_spritesheet.get_sprite(32, 0, self.width, self.height),
144                            self.game.character_spritesheet.get_sprite(64, 0, self.width, self.height)]
145
```

- Tăng chỉ số animation_loop để thực hiện vòng lặp khung hình.
- heal(amount): Hồi lại một lượng máu cho nhân vật.

```
196     def heal(self, amount):  2 usages (2 dynamic)   Sang Le
197         """Tăng lượng máu cho nhân vật."""
198         self.current_health += amount
199         if self.current_health > self.max_health:
200             self.current_health = self.max_health  # Giới hạn máu không vượt quá tối đa
201
```

- Coin, CointSpawner: Khởi tạo và spawn đồng xu lên khắp bản đồ.
- Tương tự với Enemy, EnemySpawner.
- NPC: Đại diện cho các nhân vật không phải người chơi (Non-Playable Character) trong trò chơi, bao gồm các thuộc tính và chức năng liên quan đến việc tương tác và hiển thị hội thoại.

```
class NPC(pygame.sprite.Sprite):  3 usages   Ngan cute hme
    def __init__(self, game, x, y, width, height, image_path):...

    def update(self):...

    def draw(self):...
```

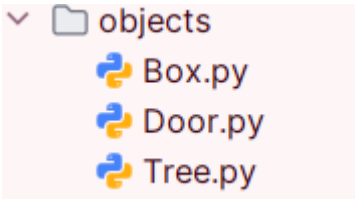
Các hàm trong class NPC

- Dialog: Hiển thị nội dung hội thoại, điều khiển luồng văn bản, và đảm bảo tính trực quan trong giao diện.

```
class Dialog:  1 usage   Ngan cute hme
    def __init__(self, screen, dialog_texts, font, text_color=(255, 255, 255), bg_color=(0, 0, 0), npc_rect=None, max_width=4(

    def draw(self):...

    def wrap_text(self, text):...

    def next_dialog(self):...

    def is_finished(self):...

    def reset(self):...
```

Các hàm trong class Dialog

c. object



Các class trong folder object

- Packet ojects: Dùng để khởi tạo các đối tượng cần sử dụng animation hoặc thao tác từ các sự kiện. Trong packet gồm:
- Class Box đại diện cho các hộp trong trò chơi. Nó xử lý trạng thái mở/đóng và các tương tác với người chơi như chơi minigame hoặc kích hoạt sự kiện.
-

```
class Box(pygame.sprite.Sprite):
    def __init__(self, game, x, y, box_id, scale_factor=4):...
    def open(self): ...
    def close(self): ...
    def update(self): ...
    def handle_yes_option(self): ...
    def handle_no_option(self): ...
```

Các hàm trong class Box

- Class Door đại diện cho cánh cửa trong trò chơi. Nó xử lý trạng thái mở/đóng cửa khi người chơi chạm vào và phát âm thanh khi mở.

```
class Door (pygame.sprite.Sprite):
    def __init__(self, game, x, y, scale_factor = 4): ...

    def open(self): ...

    def close(self): ...

    def update(self): ...
```

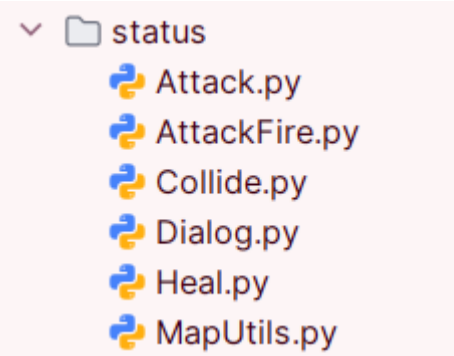
Các hàm trong class Door

- Class Tree đại diện cho các vật thể tĩnh (cây, đồ trang trí) trong trò chơi, với hiệu ứng chiều sâu dựa vào vị trí nhân vật.

```
class Door (pygame.sprite.Sprite):
    def __init__(self, game, x, y, scale_factor = 4): ...

    def open(self): ...

    def close(self): ...

    def update(self): ...
```

Các hàm trong class Tree

d. status



Các class trong folder status

- Lớp Attack và AttackFire có chức năng tương tự nhau nhưng khác nhau về loại đòn tấn công. Cả hai lớp đều kiểm tra thời gian hồi chiêu trước khi kích hoạt lại đòn tấn công và thực hiện các hoạt ảnh tương ứng với hướng tấn công. Lớp Attack xử lý đòn tấn công cơ bản, còn AttackFire là đòn tấn công có thêm khả năng di chuyển trên màn hình và có hình ảnh phóng lửa. Cả hai lớp đều kiểm tra va chạm với kẻ thù và thực hiện hành động tương ứng khi va chạm xảy ra. Cả hai đều có cơ chế hồi chiêu để giới hạn thời gian giữa các lần sử dụng đòn tấn công.

```
class Attack(pygame.sprite.Sprite):
    def __init__(self, game, x, y): ...
    @classmethod
    def can_create(cls): ...
    def use_skill(self): ...
    def update(self): ...
    def collide(self): ...
    def animate(self): ...
# hình ảnh ví dụ về các hàm trong class
```

Các hàm trong class Attack

- Class Collide xử lý va chạm giữa nhân vật (Player/Enemy) và kẻ địch. Khi va chạm, nó giảm máu của nhân vật, đẩy lùi nhân vật, và áp dụng hiệu ứng mờ. Nếu máu nhân vật xuống 0, nhân vật sẽ bị loại bỏ khỏi trò chơi và trò chơi kết thúc.

```
class Collide:
    def __init__(self, game, entity): ...

    def collide_enemy(self): ...

    def apply_fade_effect(self): ...
```

Các hàm trong class Collide

- Class Dialog trong đoạn mã này xử lý việc hiển thị một hộp thoại với hai lựa chọn trong trò chơi Pygame.

```
class Dialog:
    def __init__(self, screen, message, option1, option2): ...

    def show(self): ...
```

Các hàm trong class Dialog

- Các hàm quan trọng bao gồm: ___init___(): Khởi tạo đối tượng hộp thoại với màn hình, thông điệp, và hai lựa chọn. Xác định kích thước, vị trí của hộp thoại và các nút lựa chọn.show(): Hiển thị hộp thoại và xử lý các sự kiện người dùng, bao gồm sự kiện chuột nhấn vào một trong các lựa chọn. Khi người chơi chọn một lựa chọn, hàm trả về lựa chọn đó và thoát khỏi vòng lặp.

- Class Heal trong đoạn mã này xử lý kỹ năng hồi máu trong trò chơi Pygame, bao gồm việc quản lý cooldown, phát animation hồi máu và hồi phục một lượng máu cho nhân vật khi kỹ năng được sử dụng.

```
class Heal(pygame.sprite.Sprite): ...
    @classmethod
    def can_create(cls): ...
    def use_skill(self): ...
    def update(self): ...
    def animate(self): ...
    def apply_heal(self): ...
```

Các hàm trong class Heal

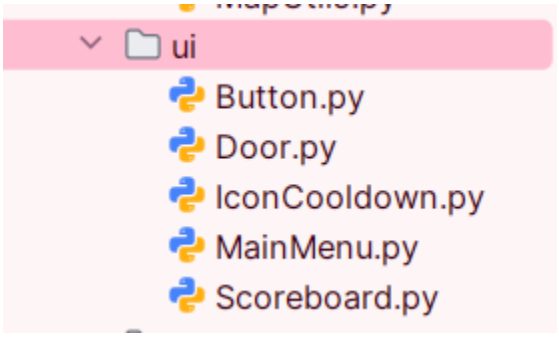
- Class MapUtils trong đoạn mã này cung cấp các phương thức tiện ích cho việc xử lý các đối tượng trên bản đồ trong trò chơi:

- get_house_positions: Lấy danh sách các vị trí của các đối tượng như nhà, cây, đá và trang trí từ dữ liệu TMX và chuyển đổi chúng thành các đối tượng pygame.Rect, với kích thước được nhân với một hệ số scale cho phù hợp.
- is_in_house: Kiểm tra xem một tọa độ (x, y) có nằm trong các khu vực nhà (hoặc các đối tượng khác như cây, đá) hay không, bằng cách sử dụng pygame.Rect để kiểm tra va chạm giữa vị trí và các khu vực nhà.

```python
class MapUtils:
    @staticmethod
    def get_house_positions(tmx_data, scale_factor=4):
        house_positions = []
        for obj in tmx_data.objects:
            if obj.name in ['House', 'Tree-Big', 'Tree-Mini', 'Tree-Tall', 'Rock', 'Decor']:
                house_positions.append(
                    pygame.Rect(obj.x * scale_factor, obj.y * scale_factor, obj.width * scale_factor, obj.height * scale_factor)
                )
        return house_positions
    @staticmethod
    def is_in_house(x, y, tile_size, house_positions):
        coin_rect = pygame.Rect(x * tile_size, y * tile_size, tile_size, tile_size)
        for house_rect in house_positions:
            if coin_rect.colliderect(house_rect):
                return True
        return False
```

Các hàm trong class MapUtils

e. ui



Các class trong folder ui

- Class Button trong Pygame tạo và quản lý các nút bấm trong giao diện trò chơi. Nó cho phép tùy chỉnh hình ảnh, văn bản, màu sắc và vị trí của nút. Khi chuột di qua, hình ảnh và màu sắc của nút thay đổi. Lớp này cũng kiểm tra sự kiện nhấp chuột để xác định liệu người dùng có nhấn vào nút hay không. Các nút có thể phát âm thanh khi nhấn và hỗ trợ văn bản tùy chỉnh hiển thị trên nút.

- Lớp Door trong trò chơi có chức năng quản lý và điều khiển các hành động mở và đóng cửa khi người chơi tương tác với cửa. Cụ thể:
    - Khởi tạo cửa: Khi tạo một đối tượng cửa, lớp sẽ lấy thông tin vị trí và kích thước từ các tham số đầu vào. Cửa được vẽ từ một sprite sheet và có thể thay đổi kích thước dựa trên yếu tố tỷ lệ (scale_factor). Màu sắc nền (colorkey) của cửa được thiết lập để loại bỏ các pixel không cần thiết.
    - Mở và Đóng cửa: Khi cửa được mở, hình ảnh cửa sẽ thay đổi để hiển thị trạng thái "mở". Đồng thời, trạng thái is_open sẽ được cập nhật thành True.
    - Khi cửa đóng lại, hình ảnh cửa được thay đổi để hiển thị trạng thái "đóng", và trạng thái is_open được cập nhật thành False.
    - Cập nhật trạng thái cửa: Trong mỗi vòng lặp game, lớp Door kiểm tra va chạm giữa cửa và người chơi. Nếu người chơi tiếp cận cửa (va chạm với rect của cửa), cửa sẽ mở và phát âm thanh mở cửa. Nếu người chơi rời khỏi cửa, cửa sẽ tự động đóng lại.

```
class Door (pygame.sprite.Sprite):
    def __init__(self, game, x, y, scale_factor = 4): ...

    def open(self): ...

    def close(self): ...

    def update(self): ...
```

Các hàm trong class Door

- Class IconCooldown chịu trách nhiệm quản lý và vẽ các biểu tượng kỹ năng cùng với thời gian hồi chiru trong trò chơi. Các chức năng chính bao gồm:
    - Khởi tạo các biểu tượng và vị trí: Lớp lấy các biểu tượng kỹ năng từ trò chơi (game.attack_icon, game.attackfire_icon, game.heal_icon) và xác định vị trí hiển thị các biểu tượng này trên màn hình.
    - Vẽ biểu tượng và thời gian hồi chiêu:
    - Dựa vào trạng thái cooldown của các kỹ năng (kiểm tra xem kỹ năng có thể sử dụng được hay không), lớp sẽ vẽ biểu tượng mờ nếu kỹ năng đang trong thời gian hồi chiêu.
    - Nếu kỹ năng đã sẵn sàng, biểu tượng sẽ được hiển thị bình thường.
    - Nếu kỹ năng đang trong cooldown, thời gian hồi chiêu còn lại (tính bằng giây) sẽ được vẽ bên dưới biểu tượng.
    - Hiệu ứng mờ khi hồi chiêu: Hàm dim_icon tạo ra hiệu ứng mờ cho biểu tượng kỹ năng khi đang trong trạng thái hồi chiêu, thông qua việc thay đổi độ alpha của hình ảnh.
    - Vẽ văn bản: Hàm draw_text vẽ thời gian hồi chiêu dưới dạng text trên màn hình, giúp người chơi dễ dàng theo dõi.

```
class IconCooldown:
    def __init__(self, game): ...

    def draw(self): ...

    def dim_icon(self, icon): ...

    def draw_text(self, text, x, y): ...
```

Các hàm trong class IconCooldown

- Class Scoreboard dùng để quản lý điểm số trong trò chơi, bao gồm chức năng lưu và tải điểm cao từ file JSON, cập nhật điểm, vẽ điểm hiện tại và bảng xếp hạng. Các phương thức chính:

- __init_(self, game, font_path: str, max_high_scores: int): Khởi tạo các thuộc tính của scoreboard như điểm hiện tại, danh sách điểm cao, font chữ, màu sắc và tải điểm cao từ file JSON.
- save_high_scores(self, filename: str): Lưu danh sách điểm cao vào file JSON. Dữ liệu được lưu dưới dạng mảng các đối tượng chứa thứ hạng và điểm số.
- load_high_scores(self, filename: str): Tải danh sách điểm cao từ file JSON. Nếu file không tồn tại hoặc có lỗi khi đọc, class sẽ khởi tạo danh sách điểm cao trống và lưu lại file JSON mới.
- update_score(self, value: int): Cập nhật điểm hiện tại của người chơi bằng cách cộng thêm giá trị vào điểm số.
- add_high_score(self): Thêm điểm hiện tại vào danh sách điểm cao và giữ lại top N điểm cao nhất (giới hạn theo max_high_scores).
- reset_score(self): Đặt lại điểm số hiện tại về 0.
- reset_high_scores(self, filename: str): Xóa toàn bộ điểm cao và lưu lại danh sách trống vào file JSON.
- draw(self): Vẽ điểm số hiện tại và bảng xếp hạng (nếu được bật). Nếu show_leaderboard là True, bảng xếp hạng sẽ được hiển thị.
- _draw_score(self): Vẽ điểm số hiện tại ở vị trí trên cùng của màn hình.
- _draw_leaderboard(self): Vẽ bảng xếp hạng với top điểm cao, hiển thị trong một hộp chữ nhật với tiêu đề và danh sách điểm.

```
class Scoreboard:
    def __init__(self, game, font_path: str = '../../res/fonts/ElecstromRegular-vmyoy.otf', max_high_scores: int = 5): ...
    def save_high_scores(self, filename: str = '../../src/ui/high_scores.json') -> None: ...
    def load_high_scores(self, filename: str = '../../src/ui/high_scores.json') -> None: ...
    def update_score(self, value: int) -> None: ...
    def add_high_score(self) -> None: ...
    def reset_high_scores(self, filename: str = '../../src/ui/high_scores.json') -> None: ...
    def draw(self) -> None: ...
    def _draw_score(self) -> None: ...
    def _draw_leaderboard(self) -> None: ...
```

Các hàm trong class Scoreboard

f. utilz

Các class trong folder utilz

File config.py thiết lập các thông số cấu hình cơ bản cho trò chơi 2D RPG, với mục đích tổ chức và quản lý các yếu tố trong trò chơi. Các thông số được chia thành các nhóm sau:

- Kích thước cửa sổ và FPS:
    - WIN_WIDTH và WIN_HEIGHT xác định kích thước của cửa sổ game, với chiều rộng là 640 pixel và chiều cao là 480 pixel, tạo ra một không gian chơi game vừa đủ.
    - TILESIZE: Chỉ định kích thước mỗi ô (tile) trên bản đồ là 32x32 pixel. Đây là kích thước cơ bản cho các đối tượng trong game, như đất, tường, hoặc các vật thể khác.
    - FPS: Số khung hình mỗi giây được thiết lập là 60, đảm bảo trò chơi chạy mượt mà và có tốc độ phản hồi tốt.
- Các Layer (Lớp):
    - Các biến như PLAYER_LAYER, ENEMY_LAYER, GROUND_LAYER, và COIN_LAYER giúp phân loại các đối tượng trong game và xác định thứ tự vẽ chúng. Các lớp này có giá trị số khác nhau để xác định thứ tự ưu tiên vẽ, lớp có giá trị cao hơn sẽ được vẽ trên lớp có giá trị thấp hơn.
    - Ví dụ: GRASS_LAYER và STREET_LAYER dùng để vẽ cỏ và đường phố, trong khi NPC_LAYER và DIALOG_LAYER dùng cho các nhân vật không phải người chơi và các hộp thoại. Điều này giúp các đối tượng trong game không bị che khuất nhau.
- Tốc độ di chuyển:
    - PLAYER_SPEED xác định tốc độ di chuyển của nhân vật người chơi là 3 đơn vị mỗi lần cập nhật. Điều này giúp người chơi có thể di chuyển dễ dàng trong môi trường game.
    - ENEMY_SPEED chỉ định tốc độ di chuyển của kẻ thù là 2, cho phép tạo sự khác biệt giữa hành vi của người chơi và kẻ thù.
- Màu sắc:

- Các giá trị màu sắc RED, BLACK, BLUE, và WHITE được định nghĩa dưới dạng mã RGB (Red, Green, Blue). Những màu này sẽ được sử dụng để tô màu các đối tượng trong game, như nền, nhân vật, và các hiệu ứng đặc biệt.

- Những thông số này giúp quản lý các đối tượng trong game, xác định cách vẽ chúng lên màn hình và điều khiển các yếu tố trong trò chơi. Việc cấu hình rõ ràng các lớp, tốc độ và màu sắc tạo ra một trò chơi dễ kiểm soát và có thể tùy chỉnh theo yêu cầu phát triển.

```
WIN_WIDTH = 640
WIN_HEIGHT = 480
TILESIZE = 32
FPS = 60


PLAYER_LAYER = 12
ENEMY_LAYER = 12
GROUND_LAYER = 2
COIN_LAYER = 12

GRASS_LAYER = 3
STREET_LAYER = 3
FLOOR_LAYER = 2
WALL_HOUSE_LAYER = 6
TOP_WALL_HOUSE_LAYER = 13
DECOR_LAYER = 13
TREE_LAYER = 10
DOOR_LAYER = 12
CARPET_LAYER = 10
NPC_LAYER = 15
DIALOG_LAYER = 20

BOX_LAYER = 15

PLAYER_SPEED = 3
ENEMY_SPEED = 2

RED = (255, 0, 0)
BLACK = (0, 0, 0)
BLUE = (0, 0, 255)
WHITE = (255, 255, 255)
```

Các giá trị của game

2.3 Kết chương

Chương này đã trình bày quá trình phân tích, thiết kế và tổ chức cấu trúc thư mục cho dự án game. Việc xây dựng một cấu trúc thư mục hợp lý, với các phần dành cho tài nguyên đồ họa, âm thanh và mã nguồn, sẽ giúp tối ưu hóa quá trình phát triển và bảo trì trò chơi. Sự phân chia này không chỉ đảm bảo tính khoa học trong tổ chức dự án mà còn hỗ trợ việc mở rộng và cải tiến game trong tương lai. Đây là bước quan trọng để dự án có thể phát triển mượt mà và hiệu quả, đồng thời giúp nhóm dễ dàng quản lý và triển khai các tính năng mới.

Chương 3. Thử nghiệm và đánh giá

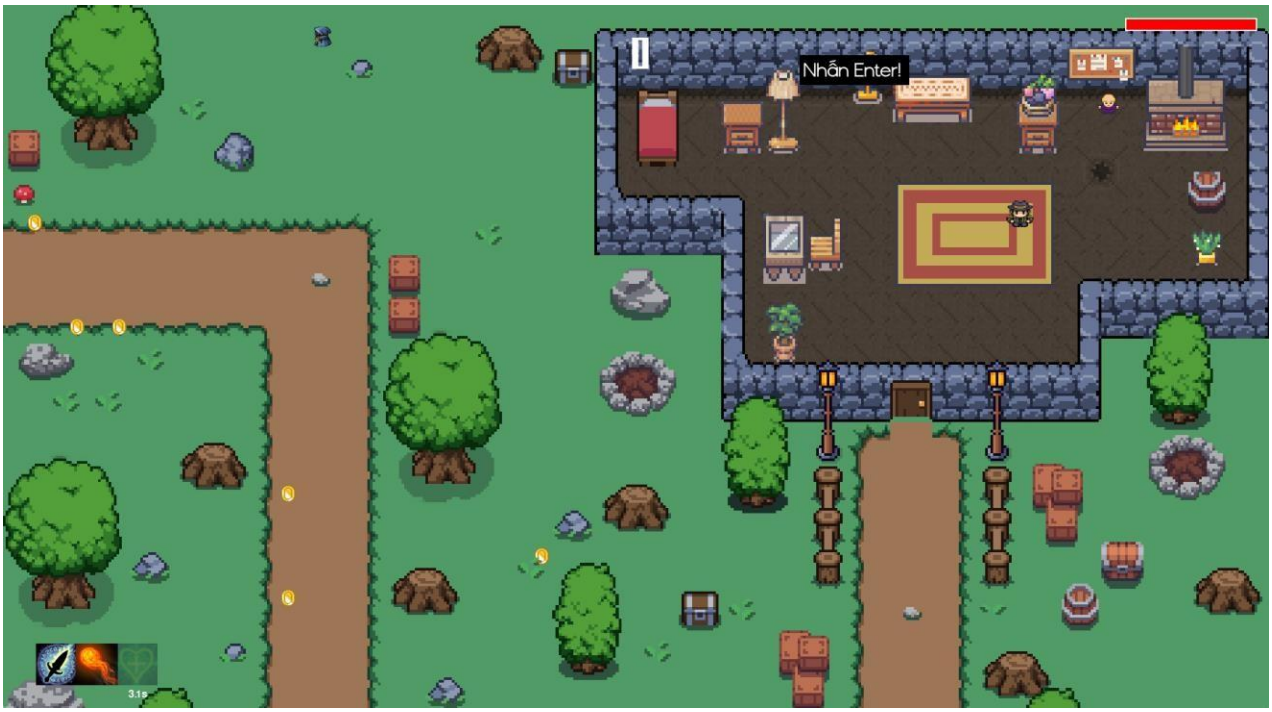
3.1 Triển khai game và chơi thử

Màn hình chính của game

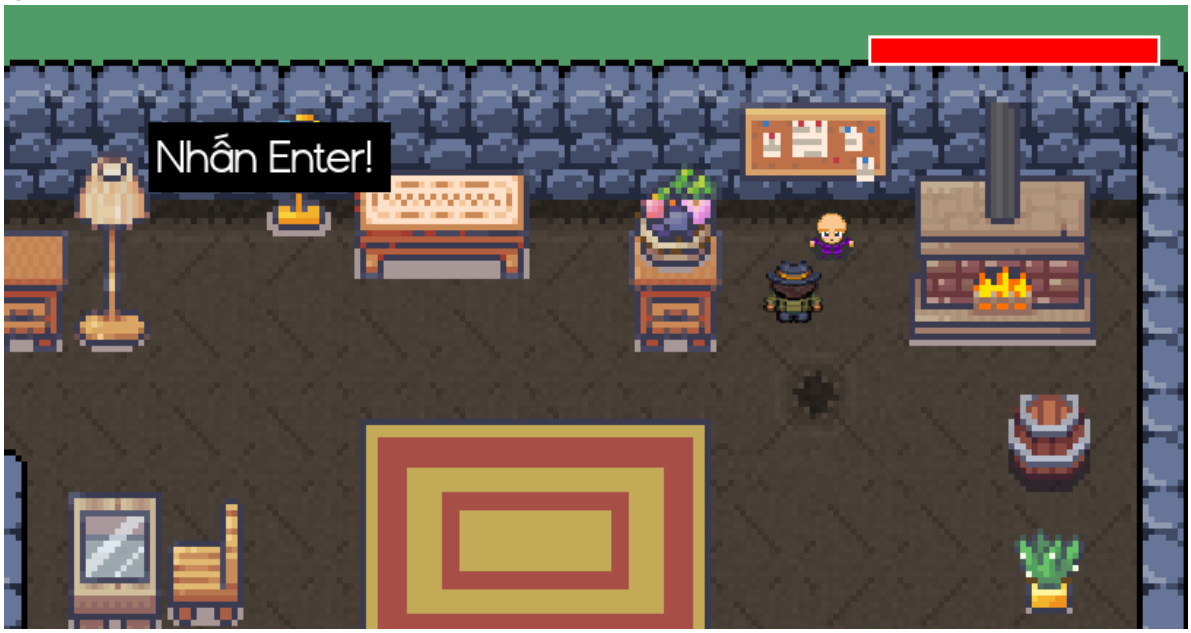
Khi ấn vào nút PLAY thì sẽ đưa tới giao diện chính của trò chơi

Khi ấn vào nút QUIT thì sẽ đưa về màn hình chính của máy tính

- Sau khi ấn vào nút PLAY giao diện sẽ chuyển sang giao diện sau



- Khi đứng cạnh NPC sẽ hiện ra nút ấn Enter để người chơi có thể trò chuyện cùng.

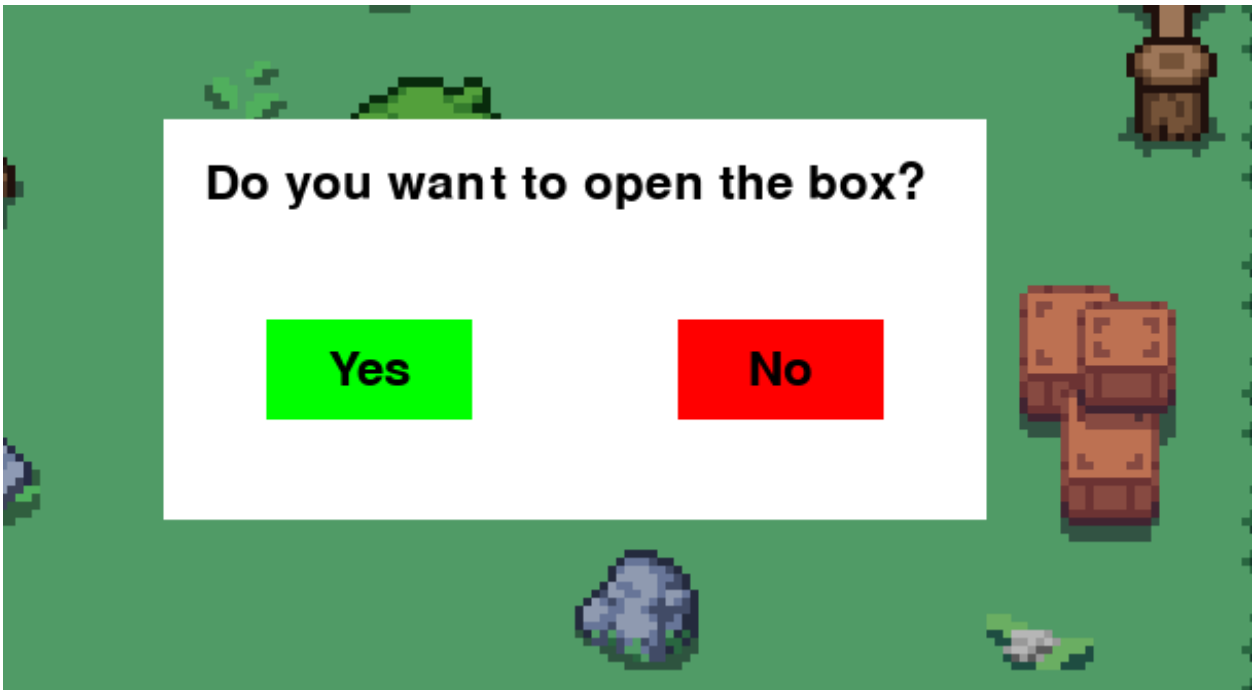


Hình ảnh minh họa cuộc nói chuyện giữa Player và NPC

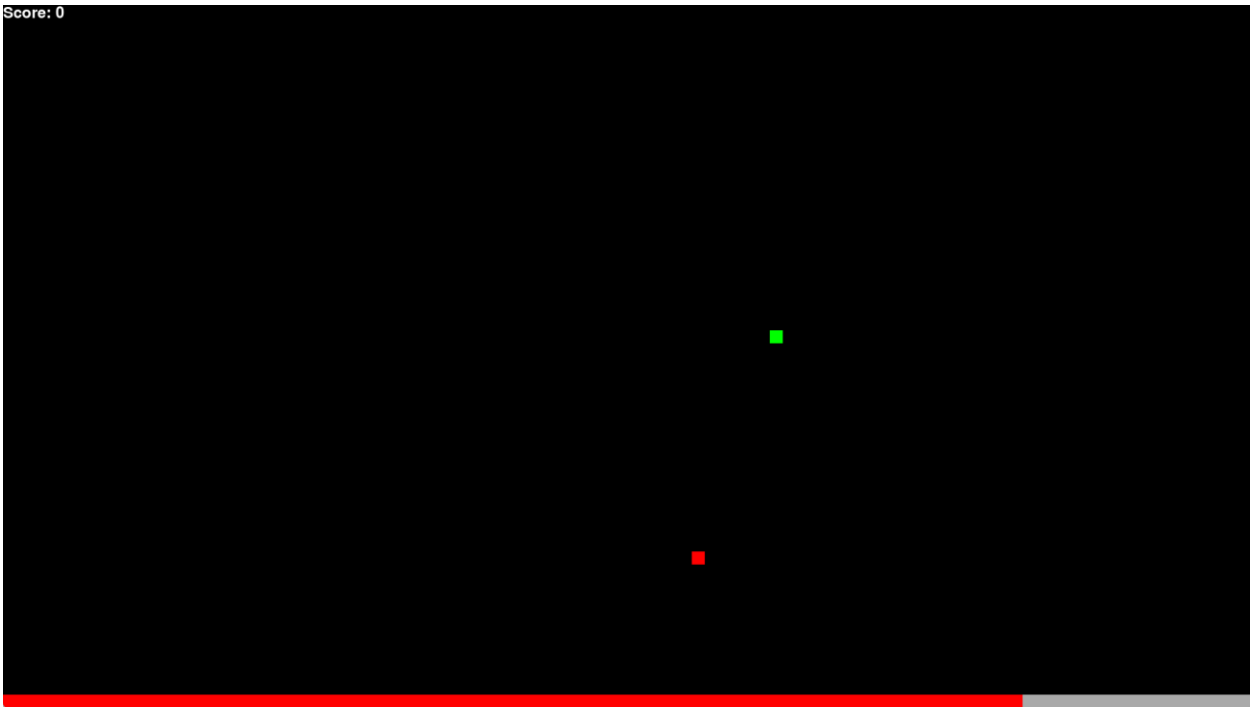
- Ngoài ra trong game đã đặt sẵn 2 hộp quà, khi mở hộp bất kì thì sẽ phải làm mini game trong đó để mang lại điểm thưởng

Vị trí của các hộp kho báu

- Khi di chuyển chạm vào hộp kho báu thì sẽ hiện ra thanh thông báo chọn Yes or No để làm nhiệm vụ.



~ Thanh thông báo ~



Game rắn săn mồi (game phụ trong hộp kho báu)

Game hứng trứng (game phụ trong hộp kho báu)

- Ngoài ra còn các chiêu của nhân vật như đánh ra lửa, đánh tay và hồi máu.

Nhân vật đánh ra đòn đánh lửa

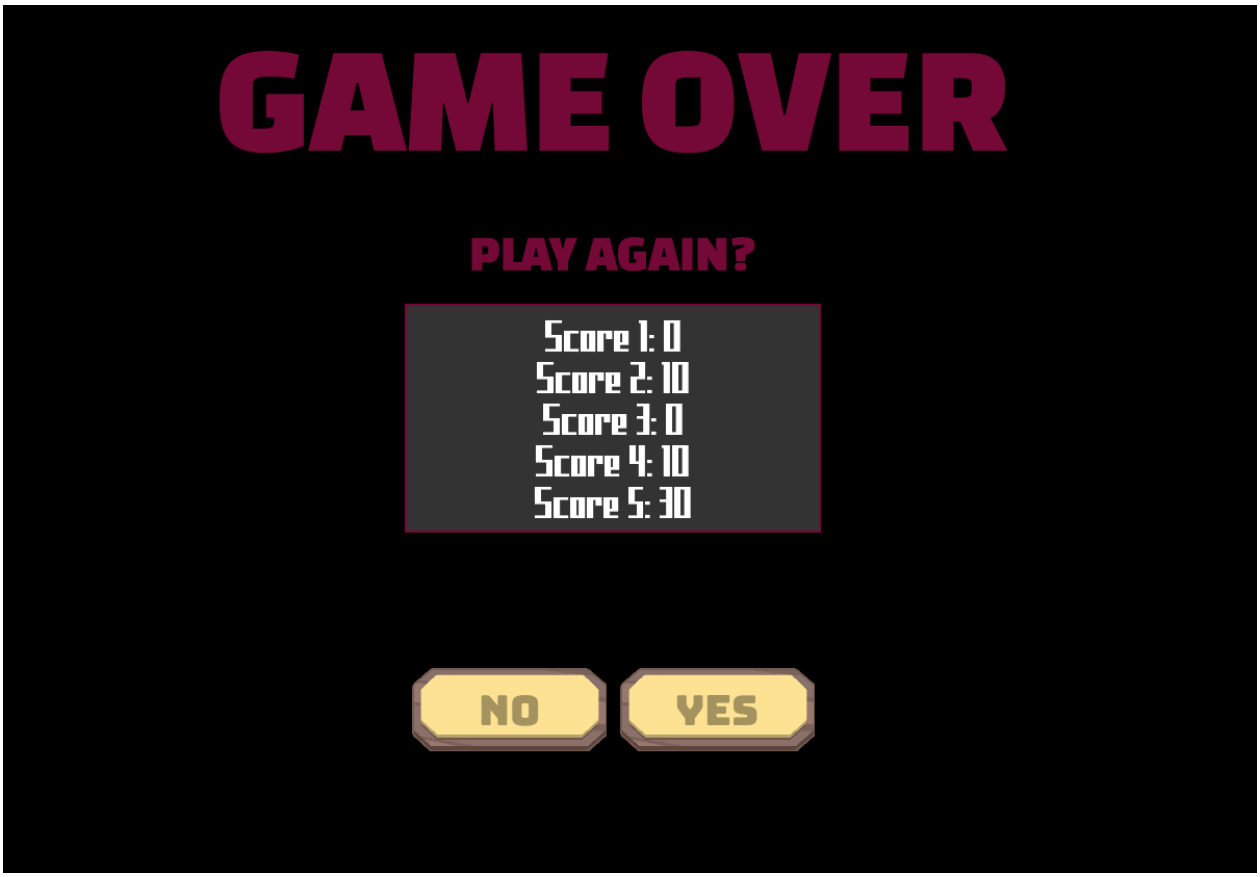
- Nhân vật cũng có thể nhặt các đồng xu để gia tăng điểm số

Hình ảnh đồng xu và điểm được cập nhật sau khi ăn đồng xu

- Ngoài ra game còn có các quái vật và khi mình va chạm với quái vật thì nhân vật sẽ bị mất máu và bật ra xa.

Hình ảnh chỉ ra quái vật và thanh máu sau khi bị quái vật va chạm

- Hệ thống bảng điểm để lưu lại điểm của người chơi



Bảng điểm của người chơi sẽ được cập nhật khi người chơi thua cuộc

3.2 Kết chương

Chương này đã mô tả quá trình triển khai và chơi thử *Dach's Enchanted Journey*. Sau khi hoàn thành việc thiết kế và phát triển các tính năng chính của trò chơi, việc triển khai đã giúp kiểm tra tính khả thi của các cơ chế gameplay, đảm bảo mọi chức năng hoạt động đúng như mong đợi. Qua quá trình chơi thử, các phản hồi và lỗi phát sinh đã được ghi nhận để tinh chỉnh và cải thiện trò chơi. Việc thử nghiệm không chỉ giúp phát hiện và khắc phục các vấn đề kỹ thuật mà còn góp phần nâng cao trải nghiệm người chơi. Đây là bước quan trọng để đảm bảo trò chơi không chỉ hoàn thiện về mặt chức năng mà còn mang lại một trải nghiệm mượt mà và thú vị cho người chơi.

KẾT LUẬN

A. Các kết quả đạt được

Mặc dù *Dach's Enchanted Journey* chưa hoàn chỉnh, nhưng trong quá trình phát triển, chúng em đã đạt được một số kết quả quan trọng và đã có được nền tảng vững chắc cho sản phẩm cuối cùng.

Đầu tiên, các cơ chế gameplay cơ bản như di chuyển nhân vật, chiến đấu và tương tác với môi trường đã được triển khai và hoạt động ổn định trong các thử nghiệm. Mặc dù vẫn còn một vài lỗi nhỏ và cần cải thiện, nhưng chúng em đã có thể xây dựng được nền tảng vững chắc để tiếp tục phát triển thêm các tính năng phức tạp hơn.

Cốt truyện của trò chơi đã được xây dựng, với các nhiệm vụ chính xoay quanh hành trình của Dách. Các nhiệm vụ này đã được thiết kế để đưa người chơi vào thế giới của game, nhưng chúng em nhận thấy vẫn cần bổ sung và phát triển thêm để tạo ra sự đa dạng và thử thách hơn nữa cho người chơi.

Về giao diện và đồ họa, mặc dù chúng em đã thiết kế và phát triển các yếu tố cơ bản, nhưng đồ họa và hiệu ứng vẫn cần được hoàn thiện để mang lại trải nghiệm tốt hơn cho người chơi. Đặc biệt, cần làm rõ hơn về các hiệu ứng trong chiến đấu và môi trường xung quanh để tạo ra một thế giới sinh động và hấp dẫn hơn.

Tính ổn định và hiệu suất của trò chơi cũng đã được cải thiện sau quá trình thử nghiệm, mặc dù vẫn còn một số vấn đề cần tối ưu hóa, nhất là khi có nhiều đối tượng xuất hiện cùng lúc. Tuy nhiên, việc phát hiện và khắc phục các lỗi cơ bản trong giai đoạn này đã giúp chúng em có cái nhìn rõ ràng hơn về các vấn đề cần giải quyết.

Cuối cùng, mặc dù trò chơi chưa hoàn thiện, nhưng chúng em cảm thấy rất tự tin với những gì đã đạt được và nền tảng vững chắc đã có. Những kết quả này mở ra nhiều cơ hội để tiếp tục phát triển trò chơi, bổ sung thêm tính năng và cải thiện các phần còn thiếu để hoàn thiện sản phẩm trong tương lai.

B. Phương hướng phát triển

- Mở rộng cốt truyện và nhân vật: Thêm các tình tiết mới, mở rộng thế giới và bổ sung nhiều nhân vật với nhiệm vụ phụ và khả năng đặc biệt. Cốt truyện có

thể mở rộng với các lựa chọn dẫn đến kết thúc khác nhau, tạo sự lôi cuốn cho người chơi.

- Cải thiện gameplay: Phát triển hệ thống chiến đấu với các kỹ năng mới, combo và thêm kẻ thù, boss mới với chiến thuật độc đáo. Các cơ chế gameplay có thể được điều chỉnh để tăng sự thách thức và tính chiến lược trong các trận chiến.
- Nâng cấp đồ họa và âm thanh: Cải tiến đồ họa với hiệu ứng ánh sáng và đổ bóng, làm cho thế giới game trở nên sống động hơn. Âm thanh sẽ được bổ sung thêm cho các tình huống đặc biệt như chiến đấu, mở khóa kho báu hoặc các cuộc đối thoại quan trọng.
- Tính năng Multiplayer: Phát triển chế độ chơi nhiều người, cho phép hợp tác hoặc thi đấu giữa người chơi. Người chơi có thể tham gia vào các trận đấu PvP hoặc làm việc nhóm để hoàn thành nhiệm vụ.
- Thêm nhiệm vụ và minigames: Tạo thêm nhiệm vụ phụ và minigame để tăng tính thú vị và đa dạng cho gameplay. Các minigame có thể đa dạng về thể loại, từ các trò chơi giải đố đến các thử thách đua xe hoặc thu thập vật phẩm.
- Lưu game và đa nền tảng: Tính năng lưu game và hỗ trợ chơi trên nhiều nền tảng như PC, Mac và thiết bị di động, cho phép người chơi tiếp tục chơi trên nhiều thiết bị mà không bị gián đoạn.
- Cải thiện AI: Nâng cấp AI của kẻ thù và NPC, giúp gameplay trở nên thử thách và hấp dẫn hơn. AI kẻ thù có thể thay đổi chiến thuật tùy theo tình huống, trong khi NPC có thể đưa ra những quyết định thông minh hơn, tạo cảm giác sống động.
- Hệ thống thành tựu: Thêm hệ thống thành tựu và thách thức để khuyến khích người chơi tiếp tục tham gia và khám phá trò chơi. Hệ thống này có thể liên kết với điểm số hoặc mở khóa các phần thưởng đặc biệt.

TÀI LIỆU THAM KHẢO

1. https://www.pygame.org
2. https://chatgpt.com
3. Pygame RPG Tutorial #1 - Pygame

Tutorial - YouTube