**Docstring**

**1. File main.py**

class Game:

    '''

    Class Game, dùng để tạo 1 phiên game mới.

    '''

    def \_\_init\_\_(self):

        '''

        Hàm khởi tạo init

        Attributes:

            #sounds

                - self.main\_sound (pygame.mixer.Sound): Nhạc nền

                - self.accept (pygame.mixer.Sound): Nhạc khi chọn các thuộc tính của UI

                - self.state (str): Trạng thái của game

                - self.started (bool): Game bắt đầu hay chưa

                - self.font (pygame.font.Font): font chữ

                - self.screen\_rect (pygame.Rect): Rect màn hình game

                - self.restart\_f (bool): Trạng thái state restart

            #title

                - title\_surf (pygame.Surface): Surface cho title

                - self.title\_rect (pygame.Rect): Rect cho title

                - off\_text\_surf (pygame.Surface): Surface tắt dùng để tạo hiệu ứng chớp tắt

                - self.blink\_surfaces : hàm cycle dùng để chuyển giữa 2 trạng thái chớp tắt cho title

                - self.blink\_surf =: hàm next để chọn suface tiếp theo

                - self.instruct\_image (pygame.image): Hình ảnh hướng dẫn chơi

                - self.instruct\_rect (pygame.Rect): Rect của instruct\_image

        '''

    def intro(self):

        '''

        Hàm chạy state 'intro' cho game lúc bắt đầu (Menu khởi đầu)

        input:

            self

        output:

            chạy visible\_sprites.custom\_draw

            Nếu ấn quit thì tắt game

            Nếu ấn Enter thì chuyển state sang 'main\_game' bắt đầu level.timer

            Nếu BLINK\_EVENT(hiệu ứng chớp tắt của title) thì chuyển Surface của title sang surface blink tiếp theo

            Vẽ title và hướng dẫn chơi

            chạy pygame.display.update() để cập nhật hình ảnh game

        '''

    def state\_manager(self):

        '''

        Hàm quản lý trạng thái state của game, gồm 3 state: intro, main\_game, restart.

        '''

    def main\_game(self):

        '''

        Hàm game chính dùng để tạo level và xử lý gameplay.

        output:

            Lấp màn hình game bằng WATER\_COLOR (trong settings.py)

            Chạy level.run()

            Nếu người chơi đã chết trước đó thì kiểm tra restart\_f, Nếu restart\_f == True thì gán restart\_f = False, phát âm thanh restart

            Kiểm tra thao tác menu của người chơi. Nếu ấn tab chạy level.toggle\_menu(). Nếu ấn R chạy level.toggle\_ranking().

            Kiểm tra nếu máu người chơi <0 và level.player.restart\_pressed == True thì chuyển state sang 'restart'

        '''

    def run(self):

        '''

        Hàm chạy vòng lặp để chạy state\_manager

        '''

**2. File Level.py**

class Level:

    '''

    Class Level dùng để xử lý game play chính như tạo map, tạo quái, drop item, tạo phép, vẽ particles, tính toán sát thương cho quái và người chơi

    '''

    def \_\_init\_\_(self):

        '''

        Hàm khởi tạo cho class Level

        Attributes:

            - self.game\_paused (bool): Game có pause hay không

            - self.ui (UI): object ui

            - self.display\_surface (pygame.display.get\_surface): Surface màn hình

            - self.timer (Timer): object Timer dùng để đếm thời gian

            #sprite group

            - self.visible\_sprites (YSortCameraGroup): Group dùng để vẽ các vật thể dựa theo người chơi

            - self.obstacle\_sprites (pygame.sprite.Group): Group các vật thể cản đường

            #attack sprites

            - self.current\_attack (Weapon): object vũ khí để xử lý ma thuật và đòn tấn công

            - self.attack\_sprites (pygame.sprite.Group): Group các sprite đòn tấn công

            - self.attackable\_sprites (pygame.sprite.Group): Group các sprite có thể bị người chơi tấn công

            #sprite setup

            - self.layouts: lưu các đường dẫn đến file csv dùng để tạo map cho game

            - self.graphics: lưu các đường dẫn đến file hình ảnh dùng để tạo map cho game

            #particles

            - self.animation\_player (AnimationPlayer): Object dùng để chạy các animation hình ảnh cho sprite

            - self.magic\_player (MagicPlayer): Object dùng để chạy các animation cho magic

            #upgrade

            - self.upgrade (Upgrade): Object dùng để tạo trình nâng cấp cho người chơi

            #monster spawn

            - self.difficulty (int): Độ khó

            - self.monster\_spawn\_radius (int): Bán kính tạo quái

            - self.monster\_spawn\_cd (int): cooldown tạo quái

            - self.monster\_spawn\_time (float): thời gian tạo quái lần cuối

            - self.screen\_shake (int): độ rung màn hình

            - self.render\_offset (list): độ lệch của rung màn hình

            - self.player\_attacked (int): tick ghi nhận khi người chơi tấn công

            - self.hit\_sound (pygame.mixer.Sound): Âm thanh khi người chơi bị tấn công

            - self.paused\_upgrade (bool): Kiểm tra người chơi có đang pause ở màn hình upgrade hay không

            - self.paused\_gacha (bool): Kiểm tra người chơi có đang pause ở màn hình gacha hay không

            - self.paused\_ranking (bool): Kiểm tra người chơi có đang pause ở màn hình ranking hay không

            - self.game\_start (bool): Kiểm tra game bắt đầu hay chưa

            - self.highscore = Highscore()

            - self.spawn\_timer (int): Timer cooldown tạo quái

        '''

    def create\_map(self):

        '''

        Hàm tạo map bằng cách lọc qua các file csv trong self.layouts. Tạo các object tại x và y tương ứng với style.

        input:

            self.layouts

        output:

            Tạo các object tại x và y tương ứng với style.

        '''

    def create\_magic(self, style, strength, cost):

        '''

        Hàm tạo hiệu ứng magic dựa theo input.

        input:

            style (str), strength (int), cost (int).

        output:

            chạy hàm magic\_player khi thỏa điều kiện

        '''

    def destroy\_attack(self):

        '''

        Hàm huỷ hiệu ứng attack sau khi tạo(True)

        Nếu current\_attack == True thì huỷ hiệu ứng.

        '''

    def damage\_player(self, amount, attack\_type):

        '''

        Hàm trừ máu người chơi khi quái vật tấn công.

        input:

            amount (int): Lượng sát thương.

            attack\_type (str): Loại tấn công.

        output:

            Gây sát thương cho người chơi nếu player.vulnerable == True.

            Trừ máu người chơi = amount.

            Tạo hiệu ứng particle đòn đánh của quái.

            Tạo rung màn hình.

        '''

    def trigger\_death\_particles(self, pos, particle\_type):

        '''

        Hàm tạo hiệu ứng particle chết cho quái và tạo item rớt ra khi quái chết.

        input:

            self.animation\_player

            pos (tuple x, y): vị trí quái chết.

            particle\_type (str): Loại particle.

        Attributes:

            chance (float): dùng random để xác định item drop.

            output:

            Dùng pos để tạo particle ở vị trí chết bằng hàm create\_particles .

            Dùng chance để xác định item drop và tạo object item dựa trên toạ độ pos.

        '''

    def player\_attack\_logic(self):

        '''

        Hàm logic về các đòn tấn công của player

        input:

            self.attack\_sprites

            self.attackable\_sprites

        output:

            Tìm các object được gán vào group attack\_sprites, kiểm tra nếu có va chạm giữa các attackable\_sprites thì tạo va chạm giữa chúng.

            Gây damage cho attackable\_sprite bằng hàm get\_damage.

        '''

    def spawn\_monster(self):

        '''

        Hàm tạo quái vật bằng cách random dựa trên tính toán thời gian hiện tại - thời gian tạo quái lần cuối > self.monster\_spawn\_cd.

        Attributes:

            current\_time (pygame.time): Lấy thời gian hiện tại

            x,y (int): toạ độ dùng để tạo quái random \* TILESIZE (pixel của sprite)

            enemy\_vec (Vector2): Vector của quái

            player\_vec (Vector2): Vector của người chơi

            distance (int): khoảng cách giữa người chơi và quái

        input:

            self.monster\_spawn\_time: thời gian tạo quái lần cuối

            self.monster\_spawn\_cd: cooldown tạo quái

            self.player

        output:

            Tạo object Enemy với các group visible\_sprites, attackable\_sprites.

        '''

    def spawnrate(self):

        '''

        Hàm dùng để giảm cooldown tạo quái dựa trên độ khó của game theo biến spawn\_timer.

        input:

            self.spawn\_timer (int): Giá trị sẽ giảm theo từng tick để xác định lúc giảm thời gian cooldown spawn quái.

        output:

            chỉnh sửa self.spawn\_cd tuỳ theo self.spawn\_timer

        '''

    def increase\_difficulty(self, time):

        '''

        Hàm tăng độ khó cho game dựa trên thời gian chơi.

        input:

            time

            self.difficulty

        output:

            Thay đổi độ khó self.difficulty dựa trên time.

        '''

    def create\_attack(self):

        '''

        Hàm tạo object Weapon, dùng cho các đòn tấn công và sử dụng phép thuật.

        '''

    def check\_screen\_shake(self):

        '''

        Hàm dùng để điều chỉnh các biến số liên quan đến rung màn mình để tính toán.

        input:

            self.screen\_shake (int):

            self.render\_offset (tuple):

            self.player\_attacked (int): đếm ngược tick người chơi tấn công (đang thử nghiệm chưa dùng).

        output:

            Điều chỉnh các biến số liên quan đến rung màn mình để tính toán.

        '''

    def run(self):

        '''

        Hàm tạo vòng lặp cho level, chạy hầu hết các method của Level.

        '''

    def add\_exp(self, amount):

        '''

        Hàm cộng exp cho người chơi khi giết quái.

        input:

            self.player.exp

        output:

            Cộng exp theo amount mà quái có.

        '''

    def toggle\_menu(self):

        '''

        Hàm điều chỉnh biến game\_paused, paused\_upgrade.

        '''

    def toggle\_ranking(self):

        '''

        Hàm điều chỉnh biến game\_paused, paused\_ranking.

        '''

    def toggle\_gacha(self):

        '''

        Hàm điều chỉnh biến paused\_gacha và tạo ra object Gacha mới mỗi khi người chơi nhặt rương báu.

        '''

class YSortCameraGroup(pygame.sprite.Group):

    '''

    Class dùng để nhóm các vật thể cần vẽ lên màn hình và theo thứ tự ưu tiên hiển thị toạ độ y lớn hơn.

    Thừa kế class pygame.sprite.Group

    '''

    def \_\_init\_\_(self):

        '''

        Hàm khởi tạo cho class YSortCameraGroup

        Attributes:

            - self.display\_surface (pygame.Surface): Surface màn hình game

            - self.half\_width (int): phân nửa động rộng màn hình

            - self.half\_height (int): phân nửa độ cao màn hình

            - self.offset (Vector2): vector độ chênh lệch

            #floor creation

            - self.floor\_sur (pygame.Surface): Surface phần hình ảnh màn chơi

            - self.floor\_rect (pygame.Rect): Rect của hình ảnh màn chơi

            - self.sparks (Spark): tỉnh năng spark đang thử nghiệm

            - self.s (pygame.Surface): dùng để vẽ bóng cho game

        '''

    def custom\_draw(self, player, render\_offset, player\_attacked):

        '''

        Hàm vẽ các visible\_sprites theo offset để tạo hiệu ứng camera di chuyển theo người chơi.

        Đồng thời xử lý bóng của các visible\_sprites lên màn hình.

        input:

            player (Player): Object người chơi

            render\_offset (list): các thông số dùng để rung màn hình

            player\_attacked (int): Biến thử nghiệm, chưa dùng tới

        output:

        Blit các surface lên màn hình game:

            Surface các visible\_sprites

            Surface bóng của visible\_sprites

        Tính năng tối ưu hoá:

            Không hiển thị hình ảnh nếu khoảng cách giữa vật thể và người chơi > 800

        '''

    def enemy\_update(self, player):

        '''

        Hàm lọc qua các visible\_sprites là enemy và cập nhật bằng việc chạy hàm enemy\_update

        '''

**3. File tile.py**

class Tile(pygame.sprite.Sprite):

    '''

    Class Tile dùng để tạo tile cho game và điều chỉnh sự tương tác giữa player với tile

    '''

    def \_\_init\_\_(self,pos,groups,sprite\_type, surface = pygame.Surface((TILESIZE, TILESIZE))):

        '''

        Hàm khởi tạo init

        Atribute:

            - self.sprite\_type : loại sprite

            - y\_offset : offset của sprite\_type

        input :

            - sprite\_type

        output :

            - self.rect

            - self.hitbox

        '''

**4. File Entity.py**

class Entity(pygame.sprite.Sprite):

    '''

    class này dùng để tạo các thực thể có trong game

    '''

    def \_\_init\_\_(self, groups):

    '''

    Attribute

    - self.frame\_index : cho frame\_index mặc định bằng 0

    - self.animation\_speed : độ nhanh hoạt ảnh của thực thể

    - self.direction : phương hướng

    '''

    def move(self,speed):

        '''

        Hàm tạo các bước di chuyển và hitbox của thực thể

        input : self.direction.magnitude, self.direction.x, self.direction.y

        output : self.direction , self.hitbox.x, self.hitbox.y

        '''

    def collision(self, direction):

        '''

        Hàm điều chỉnh va chạm collision của các thực thể trong game

            input : direction

            output : self.direction và self.hitbox

        '''

    def wave\_value(self):

        '''

        Hàm dùng để tạo tương tác giữa player và thực thể khi bị đánh trúng( khi bị đánh thì thực thể sẽ thay đổi màu trong chốt lát)

            input : value

            output : 255 hoặc 0

        '''

**5. File Enemy.py**

class Enemy(Entity):

    '''

    class Enemy là class tạo ra các quái vật trong game và điều chỉnh tương tác giữa player và enemy

    '''

    def \_\_init\_\_(self,monster\_name, pos, groups, obstacle\_sprites, damage\_player, trigger\_death\_particles, add\_exp, difficulty, visible\_sprites):

        '''

        Attribute:

            #general setup

        - self.sprite\_type : sprite\_type là 'enemy'

        - self.status : status là 'idle'

        - self.difficulty : độ khó của quái

        - self.visible\_sprites : group các thực thể enemy trong trò chơi

            #graphics setup

        - self.import\_graphics(monster\_name) : đưa hình ảnh của quái vật vào game

        - self.image : tạo hình ảnh của enemy

        - self.rect : tạo rect cho enemy

            #movement

        - self.hitbox : hitbox cho quái

        - self.obtacle\_sprites : group các vật thể có thể cản đường người chơi

            #enemy stats

        - self.monster\_name : tên của quái vật

        - monster\_info: list chứa dữ liệu của quái vật như health, exp, speed, attack\_damage, ...

            #player interaction

        - self.can\_attack : có thể tấn công

        - self.attack\_time : thời gian tấn công

        - self.attack\_cooldown : khoảng thời gian chờ giữa các đòn tấn công

        - self.damage\_player : sát thương có thể gây ra cho player

        - self.trigger\_death\_particles : kích hoạt hiệu ứng khi quái bị đánh bại

        - self.add\_exp : exp của quái mà người chơi nhận được khi giết thành công

            #invincibility timer

        - self.vulnerable : quái có thể bị tấn công

        - self.hit\_time : khoảng thời gian chờ khi bị tấn công

        - self.invincibility\_duration : khoảng thời gian khi quái không thể bị nhận sát thương

            #sounds

        - self.death\_sound : âm thanh khi quái chết

        - self.hit\_sound = âm thanh khi bị đánh trúng

        - self.death\_sound.set\_volume : điều chỉnh đô lớn âm thanh

        - self.hit\_sound.set\_volume : điều chỉnh âm thanh khi bị đánh

        - self.attack\_sound = âm thanh khi quái tấn công player

        - self.attack\_sound.set\_volume : điều chỉnh âm thanh khi tấn công

        '''

    def import\_graphics(self, name):

        '''

        Hàm dùng để đưa hình ảnh vào game

            input : self.animation

            output : vẽ ra các hoạt ảnh từng trạng thái của quái vật như idle, move hay attack

        '''

    def get\_player\_distance\_direction(self, player):

        '''

        Hàm dùng để tạo sự tương tác giữa player và quái khi hai bên va chạm nhau trong khoảng cách nhất định

            input : distance

            output : distance, direction

        '''

    def get\_status(self, player):

        '''

        Hàm này xuất ra trạng thái của quái tùy thuộc khoảng cách của quái và player

            input : distance

            output : trạng thái của quái

        '''

    def animate(self):

        '''

        Hàm dùng để tạo animate cho quái

            input : self.frame\_index, self.status, self.vulnerable

            output : self.can\_attack, self.frame\_index, self.wave\_value

        '''

    def cooldowns(self):

        '''

        Hàm tạo cooldown cho các đòn tấn công của quái cũng như thời gian tấn công của player

            input : current\_time, self.can\_attack, self.attack\_cooldown, self.vulnerable

            output : self.can\_attack = True

                     self.vulnerable = True

        '''

    def get\_damage(self, player, attack\_type):

        '''

        Hàm dùng để xuất ra những sự kiện như trừ máu player, trừ máu quái vật  sau khi quái và player tấn công lẫn nhau

            input : self.vulnerable

            output : hit\_sound.play()

                     health của quái sau khi bị tấn công

        '''

    def check\_death(self):

        '''

        Hàm dùng để kiểm tra sự kiện chết của quái

            input : self.health

            output : chạy kill(), trigger\_death\_particles, add\_exp và âm thanh khi quái chết đi

        '''

    def hit\_reaction(self):

    def actions(self, player):

        '''

        Hàm thể hiện các hành động của quái khi đạt các điều kiện khoảng cách

            input : self.status

            output : attack\_sound.play()

                     direction

        '''

    def update(self):

        '''

        Hàm chạy các hàm hit\_reaction(),move(),animate(),cooldowns(),check\_deck()

        '''

    def enemy\_update(self, player):

        '''

        Hàm chạy các hàm get\_status(),actions()

        '''

**6. File Player.py**

class Player(Entity):

    '''

    Class dùng để tạo object Player. Thừa kế từ class cha Entity.

    '''

    def \_\_init\_\_(self, pos, groups, obstacle\_sprites, create\_attack, destroy\_attack, create\_magic, visible\_sprites, toggle\_gachapon, timer):

        '''

        Hàm khởi tạo của class Player.

        input:

            - pos ((x,y)): vị trí người chơi

            - groups (pygame.Group): các group object

            - self.obstacle\_sprites (pygame.Group): group các obstacle\_sprites

            - self.create\_attack: hàm truyền vào create\_attack

            - self.destroy\_attack: hàm truyền vào destroy\_attack

            - self.create\_magic: hàm truyễn vào create\_magic

            - self.visible\_sprites (pygame.Group): group các visible\_sprites

            - self.toggle\_gacha: hàm kích hoạt toggle\_gacha

            - timer (Timer): bộ đếm thời gian

        Attributes:

            - self.image (pygame.image): hình ảnh người chơi

            - self.rect (pygame.Rect): Rect của người chơi dựa trên hình ảnh

            - self.hitbox (pygame.hitbox): hitbox của người chơi dựa trên rect

            - self.visible\_sprites (pygame.Group): group các visible\_sprites

            - self.timer (Timer): object Timer

            #graphics setup

            - self.status (str): trạng thái người chơi

            - self.display\_surface (pygame.Surface): Surface màn hình game

            #task

            - self.sprite\_type (str): loại sprite

            - self.dead (bool): Trạng thái chết

            #movement

            - self.attacking (bool): Trạng thái tấn công

            - self.attack\_time (time): thời gian tấn công

            - self.obstacle\_sprites (pygame.Group): group các obstacle\_sprites

            #weapon

            - self.create\_attack: hàm truyền vào create\_attack

            - self.destroy\_attack: hàm truyền vào destroy\_attack

            - self.weapon\_index (int): index vũ khí

            - self.weapon (list): list các vũ khí

            - self.can\_switch\_weapon (bool): Đổi vũ khí

            - self.weapon\_switch\_time (time): thời gian đổi vũ khí

            - self.switch\_duration\_cooldown (int): cooldown đổi vũ khí

            #magic

            - self.create\_magic: hàm truyền vào create\_magic

            - self.magic\_index (int): index của magic

            - self.magic (list): list các magic có thể dùng

            - self.can\_switch\_magic (bool): Đổi loại magic

            - self.magic\_switch\_time (time): thời gian đổi magic

            - self.switch\_duration\_cooldown (int): cooldown đổi magic

            #stats

            - self.stats (dict): stat mặc định của người chơi

            - self.max\_stats (dict): stat tối đa của người chơi

            - self.upgrade\_cost (dict): cost của mỗi nâng cấp

            - self.health (int): máu hiện tại của người chơi

            - self.energy (int): energy hiện tại của người chơi

            - self.exp (int): exp hiện tại của người chơi

            - self.speed (int): speed hiện tại của người chơi

            - self.recovery\_rate (float): tốc độ phục hồi energy

            - self.attack\_cooldown (int): cooldown tấn công của người chơi

            #invincibility timer

            - self.vulnerable (bool): trạng thái vulnerable của người chơi

            - self.hurt\_time (time): thời gian nhận damage từ quái

            - self.invulnerability\_duration (int): thời gian không bị tấn công

            #import sounds

            - self.weapon\_attack\_sound (pygame.mixer.Sound): âm thanh khi tấn công bằng vũ khí

            - self.g\_o\_ft (bool): kiểm tra game over chưa

            - self.game\_over (pygame.mixer.Sound): âm thanh khi game over

            - self.is\_player (bool): kiểm tra là player

            - self.font (pygame.font.Font): font in màn hình chết

            - self.restart\_pressed (bool): kiểm tra ấn phím restart chưa

            #gacha

            - self.toggle\_gacha: hàm kích hoạt toggle\_gacha

        '''

    def input(self):

        '''

        Hàm input:

            - Nhận vào các input đầu vào của player và thay đổi status của player.

            - Đồng thời xử lý các input tấn công và input thay đổi vũ khí/magic.

        '''

    def import\_player\_assets(self):

        '''

        Hàm import các hình ảnh của player vào animations tương ứng với các status của player.

        Attributes:

            - character\_path (str): đường dẫn đến thư mục asset của player

            - self.animations (dict): dictionary chứa asset

        '''

    def write\_to\_file(self, data):

        '''

        Hàm ghi nhận highscore vào file save.

        input:

            data (int): thời gian mà player còn sống

        '''

    def cooldowns(self):

        '''

        Hàm quản lý các cooldown của người chơi. Bao gồm:

                    - Thời gian tấn công,

                    - Thời gian thay đổi vũ khí/magic,

                    - Thời gian không nhận sát thương từ quái.

        Attributes:

            current\_time (time): thời gian hiện tại

        '''

    def energy\_recovery(self):

        '''

        Hàm xử lý việc hồi năng lượng theo thời gian.

        Đồng thời ngăn chặn hồi năng lượng vượt ngưỡng năng lượng tối đa

        '''

    def animate(self):

        '''

        Hàm vẽ animation cho người chơi dựa trên status tương ứng.

        output:

            - self.image = image status tương ứng

            - Tạo hiệu ứng flicker dựa trên hàm wave\_value của entity

        '''

    def get\_status(self):

        '''

        Hàm lấy trạng thái status của người chơi dựa trên vị trí x,y hoặc trạng thái tấn công.

        output:

            Trả về status hiện tại

        '''

    def get\_full\_weapon\_damage(self):

        '''

        Hàm lấy damage của người chơi +  damage của vũ khí

        return:

            Damage của người chơi + damage của vũ khí

        '''

    def get\_full\_magic\_damage(self):

        '''

        Hàm lấy damage của người chơi +  damage của magic

        return:

            Damage của người chơi + damage của magic

        '''

    def get\_value\_by\_index(self, index):

        '''

        Hàm lấy giá trị stat của người chơi dựa theo index.

        return:

            Giá trị stat của người chơi dựa theo index.

        '''

    def get\_cost\_by\_index(self, index):

        '''

        Hàm lấy giá trị cost của người chơi dựa theo index.

        return:

            Giá trị cost của người chơi dựa theo index.

        '''

    def get\_current\_values\_by\_index(self, index):

        '''

        Hàm lấy giá trị stat hiện tại dựa theo index.

        return:

            Giá trị stat hiện tại dựa theo index.

        '''

    def item\_pickup(self):

        '''

        Hàm xử lý nhặt vật phẩm cho người chơi.

        - Sort qua các visible\_sprites, nếu visible\_sprites có sprite\_type và sprite\_type == 'item'

        thì cho người chơi nhặt vật phẩm để nhận các chỉ số cộng thêm.

        - Sau khi nhặt xong thì kill object vật phẩm ấy.

        '''

    def check\_death(self):

        '''

        Hàm kiểm tra trạng thái chết.

        - Nếu health < 0 sẽ tạm dừng và dùng hàm write\_to\_file để ghi lại thời gian player đã sống và

        set dead = True.

        - In ra màn hình yêu cầu restart và gọi hiệu ứng âm thanh game over

        - Nếu input người chơi là phím ENTER thì restart\_pressed = True

    def update(self):

        '''

        Hàm chạy các method của class Player và update các method.

        '''

**7. File weapon.py**

import pygame

class Weapon(pygame.sprite.Sprite):

    '''

    Class này dùng để tạo ra các vũ khí và tạo các chuyển động khi dùng vũ khí đó để tấn công

    '''

    def \_\_init\_\_(self, player, groups):

'''

    Hàm khởi tạo

    Attribute :

      - self.sprite\_type(string) : loại sprite dùng trong class này là 'weapon'

      - direction(string) : phương hướng của vũ khí khi tấn công

      - full\_path : đường dẫn tới file hình ảnh

      - self.image : load hình ảnh thông qua đường dẫn full\_path

    input :

        Các hướng direction

    output :

        Tùy theo các hướng direction sẽ cho ra các kết quả tương ứng

    '''

**8. File magic.py**

class MagicPlayer:

    '''

    Class này dùng để tạo phần sử dụng phép thuật cho player

    '''

    def \_\_init\_\_(self, animation\_player):

        '''

        Hàm khởi tạo init

        Attribute:

        - self.animation\_player : hoạt ảnh của player

        - self.sound : âm thanh khi dùng phép thuật để tấn công hoặc hồi máu

        - self.sounds['heal'].set\_volume(0.05) : điều chỉnh âm thanh

        - self.sounds['flame'].set\_volume(0.05) : điều chỉnh âm thanh

        '''

    def heal(self, player, strength, cost, groups):

        '''

        Hàm tạo phương thức heal cho player

            input : player.energy

            output : player.health tăng lên và self.sound['heal'].play()

        '''

    def flame(self, player, cost, groups):

        '''

        Hàm tạo phương thức flame cho player

            input : player.energy

            output : direction lúc sử dụng phép tấn công ở hướng nào và

                     tạo các particles của phép flame trên màn hình

                     self.sound['flame'].play()

        '''

**9. File Particles.py**

class AnimationPlayer:

    '''

    Class dùng để tạo các particles cho hoạt ảnh của player

    '''

    def \_\_init\_\_(self):

        '''

        Hàm khởi tạo init

        Attribute

        self.frame : frame của các loại hoạt ảnh magic, attacks, monster deaths, leafs.

        '''

    def reflect\_images(self, frames):

        '''

        Hàm tạo các ảnh lặp lại

            input : frames

            output : new\_frames

        '''

    def create\_grass\_particles(self, pos, groups):

        '''

        Hàm tạo hoạt ảnh của grass

            input : animation\_frames

            output : ParticleEffect

        '''

    def create\_particles(self, animation\_type, pos, groups):

        '''

        Hàm tạo hoạt ảnh của các frames

            input : animation\_frames

            output : ParticleEffect

        '''

class ParticleEffect(pygame.sprite.Sprite):

    '''

    Class dùng để tạo các hiệu ứng đặt biệt

    '''

    def \_\_init\_\_(self, pos, animation\_frames, groups):

        '''

        Hàm khởi tạo init

        Attribute

        - self.frame\_index : cho frame index bằng 0

        - self.animation\_speed : tốc độ của hoạt ảnh

        - self.frames : các hoạt ảnh frames

        - self.image : hình ảnh của frames

        - self.rect : tạo rect cho self.image

        '''

    def animate(self):

        '''

        Hàm tạo các hoạt ảnh

            input : self.frame\_index

            output : self.kill() hoặc self.image

        '''

    def update(self):

        '''

        Chạy hàm animate()

        '''

**10. File Upgrade.py**

import math

import pygame

from settings import \*

class Upgrade:

    '''

    Class dùng để thiết lập cơ chế nâng cấp nhân vật khi đủ lượng Exp yêu cầu

    '''

    def \_\_init\_\_(self, player):

        '''

        Hàm khởi tạo init

        Attribute:

          #general setup

          - self.display\_surface : Surface màn hình

          - self.player : người chơi

          - self.attribute\_nr : là độ lớn các thuộc tính của người chơi

          - self.attribute\_names : là tên các thuộc tính

          - self.font : font các thuộc tính

          - self.max\_value : là chỉ số thuộc tính tối đa

          - self.current\_values : là chỉ số thuộc tính hiện tại mà người chơi đang có

          #item creation

          - self.height : là chiều cao của item

          - self.width : là độ rộng của item

          - self.create\_item() : tạo item

          #selection system

          - self.selection\_time : là thời gian chọn nâng cấp

          - self.can\_move : di chuyển khi lựa chọn nâng cấp các thuộc tính nhân vật

          - self.move\_fx : bỏ âm thanh vào menu

          - self.move\_fx.set\_volume(0.3) : set độ lớn âm thanh của menu

        '''

        #general setup

        self.display\_surface = pygame.display.get\_surface()

        self.player = player

        self.attribute\_nr = len(player.stats)

        self.attribute\_names = list(player.stats.keys())

        self.font = pygame.font.Font(UI\_FONT, UI\_FONT\_SIZE)

        self.max\_values = list(player.max\_stats.values())

        self.current\_values = list(player.stats.values())

        #item creation

        self.height = self.display\_surface.get\_size()[1] \* 0.8

        self.width = self.display\_surface.get\_size()[0] // 6

        self.create\_items()

        #selection system

        self.selection\_index = 0

        self.selection\_time = None

        self.can\_move = True

        self.move\_fx = pygame.mixer.Sound('./audio/Menu1.wav')

        self.move\_fx.set\_volume(0.3)

    def input(self):

        '''

        Hàm dùng để nhập các phương thức để lựa chọn các thuộc tính cần nâng cấp

          input :

                keys[pygame.K\_RIGHT],keys[pygame.K\_LEFT],keys[pygame.K\_UP]

          output :

                phím Left, Right để chọn các thanh nâng cấp thuộc tính của nhân vật

                phím Up để nâng cấp stat nếu đủ điểm

        '''

        keys = pygame.key.get\_pressed()

        if self.can\_move:

            if keys[pygame.K\_RIGHT] and self.selection\_index < self.attribute\_nr - 1:

                self.selection\_index += 1

                self.move\_fx.play()

                self.can\_move = False

                self.selection\_time = pygame.time.get\_ticks()

            elif keys [pygame.K\_LEFT] and self.selection\_index >= 1:

                self.selection\_index -= 1

                self.move\_fx.play()

                self.can\_move = False

                self.selection\_time = pygame.time.get\_ticks()

            if keys[pygame.K\_UP]:

                self.can\_move = False

                self.selection\_time = pygame.time.get\_ticks()

                self.item\_list[self.selection\_index].trigger(self.player)

    def create\_items(self):

        '''

        Hàm dùng để tạo các item chứa thuộc tính

        Attribute

          self.item\_list : là danh sách các item

          #horizontal position và #vertical position :

              Tạo tương tác giữa item và player( khoảng cách giữa các iteam, vị trí, kich thước)

          #create object :

              Để tạo item và đưa item vào trong game

        '''

        self.item\_list = []

        for item, index in enumerate(range(self.attribute\_nr)):

            #horizontal position

            full\_width = self.display\_surface.get\_size()[0]

            increment = full\_width // self.attribute\_nr

            left = (item \* increment) + (increment - self.width) // 2

            #vertical position

            top = self.display\_surface.get\_size()[1] \* 0.1

            #create object

            item = Item(left, top, self.width, self.height, index, self.font)

            self.item\_list.append(item)

    def selection\_cooldown(self):

        '''

        Hàm dùng để set up thời gian chọn

          input :

                self.can\_move

          output :

                nếu self.can\_move not True thì current\_time(pygame.time.get\_ticks)

                và nếu current\_time - self.selection\_time >= 150 thì self.can\_move thành True

        '''

        if not self.can\_move:

            current\_time = pygame.time.get\_ticks()

            if current\_time - self.selection\_time >= 150:

                self.can\_move = True

    def display(self):

        '''

        Hàm hiện thông tin các item thông qua hai hàm input() và selection\_cooldown

          input :

              index, item

          output :

              name, value, max\_value, current\_value, cost từng thuộc tính có thể nâng cấp

        '''

        self.input()

        self.selection\_cooldown()

        for index, item in enumerate(self.item\_list):

            #get attributes

            name = self.attribute\_names[index]

            value = self.player.get\_value\_by\_index(index)

            max\_value = self.max\_values[index]

            current\_values = self.player.get\_current\_values\_by\_index(index)

            cost = self.player.get\_cost\_by\_index(index)

            item.display(self.display\_surface, self.selection\_index, name, value, max\_value, cost, current\_values)

class Item:

    '''

    Class dùng để tạo các item và thanh bar

    '''

    def \_\_init\_\_(self, l, t, w, h, index, font):

        '''

        Hàm khởi tạo init

        Attribute

          - self.rect : tạo rect item theo left, top, width, height

          - self.index : tạo index từng item

          - self.font : tạo font chữ

          - self.up : âm thanh của item

          - self.up.set\_volume : điều chỉnh độ lớn âm thanh

          - self.c\_up : âm thanh menu màn hình Upgrade

          - self.c\_up.set\_volume : điều chỉnh độ lớn âm thanh

        '''

    def display\_names(self, surface, name, cost, selected):

        #title

        '''

          title\_surf : tạo title trên surface

          title\_rect : tạo rect cho title\_surf

        '''

        #cost

        '''

          cost\_surf : tạo cost lên surface

          cost\_rect : tạo rect cho cost\_surf

        '''

        #draw

        '''

          surface.blit(title\_surf, title\_rect) : vẽ title lên màn hình game

          surface.blit(cost\_surf, cost\_rect) : vẽ cost lên màn hình game

        '''

    def display\_bar(self, surface, value, max\_value, selected, current\_values):

        '''

        Hàm này dùng để vẽ thanh bar lên màn hình Upgrade. Thanh bar sẽ thay đổi màu và màu chữ trong lúc đang chọn

        Attribute

          #drawing setup

          - top với bottom : xác định vị trí bằng pygame.math.Vector2

          - color : - input : selected(thuộc tính đang chọn)

                    - output : BAR\_COLOR\_SELECTED hoặc BAR\_COLOR

          - text\_color :  - input : selected(thuộc tính đang chọn)

                          - output : TEXT\_COLOR\_SELECTED hoặc TEXT\_COLOR

          #bar setup

          - full\_height : set up chiều cao của thanh bar

          - relative\_number : các số liên quan tới bar

          - value\_rect : tạo rect cho value

          #current values

          - curr\_surf : tạo current value trên surf

          - curr\_rect : tạo rect cho curr\_surf

          #draw elements

          - pygame.draw.line() và pygame.draw.rect : vẽ surface

          - surface.blit(curr\_surf, curr\_rect) : vẽ current value lên màn hình

        '''

    def trigger(self, player):

        '''

        Hàm dùng để tạo sự tương tác giữa player và thanh bar thuộc tính theo giá trị exp của nhân vật

          input :

              - player.exp

              - player.upgrade\_cost

              - upgrade\_attribute

              - player.max\_stat

          output :

              - player.stats[upgrade\_attribute]

              - player.upgrade\_cost[upgrade\_attribute]

          Nếu điểm thuộc tính đã nâng cấp vượt qua giá trị tối đa mà thuộc tính có thể có thì

          set điểm thuộc tính đó bằng giá trị tối đa của thuộc tính đó.

        '''

    def display(self, surface, selection\_num, name, value, max\_value, cost, current\_values):

        '''

        Hàm dùng để vẽ ra màn hình các thuộc tính của upgrade như name, cost, max\_value, current\_value

          input :

                - self.index

          output :

                - Bắt đầu vẽ ra hình ảnh các thuộc tính của upgrade vào màn hình game

        '''

**11. File Highscore.py**

class Highscore:

    '''

    Class tạo object Highscore, dùng để xếp hạng giờ chơi.

    '''

    def \_\_init\_\_(self):

        '''

        Hàm khởi tạo cho class Highscore.

        Attribute:

            #general setup

            self.display\_surface (pygame.Surface): Surface màn hình game

            self.score\_list (list): list các highscore

            self.attribute\_nr (int): số lượng highscore

            self.font (pygame.font.Font): font chữ

            self.font2 (pygame.font.Font): font chữ title

            #item creation

            self.height (int): độ cao item

            self.width (int): độ rộng item

        '''

    def read\_score(self):

        '''

        Hàm đọc và sắp xếp điểm từ file txt (theo thứ tự từ lớn đến bé)

        Attributes:

            x (list): list các phần tử thời gian xếp từ lớn đến bé

        return:

            5 phần tử đầu tiên của x

        '''

    def create\_items(self):

        '''

        Hàm tạo các phần tử trong list item có thể roll

        Attributes:

            self.item\_list (list): list các item (ui)

            full\_width (int): độ dài màn hình

            increment (int): khoảng cách giữa các phần tử

            left (int): vị trí các phần tử

            item (Item): tạo object item

        output: thêm các item vào list tương ứng

        '''

    def display(self):

        '''

        Hàm thực thi các method của class Gacha, đồng thời hiển thị các phần tử

        '''

class Item:

    '''

    Class Item chuyên dùng để tạo các phần tử cho class Highscore.

    '''

    def \_\_init\_\_(self, l, t, w, h, index, font):

        '''

        Hàm khởi tạo cho class Item (Highscore)

        Attributes:

            - l (int): vị trí left

            - t (int): vị trí top

            - w (int): độ rộng

            - h (int): độ cao

            - index (int): index của item

            - font (pygame.font): font chữ chính

            - self.rect (pygame.Rect): rect item

            - self.index (int): index item được chọn.

            - self.font (pygame.font): font chữ chính.

        '''

    def display\_names(self, surface, name):

        '''

        Hàm hiển thị tên của phần tử, thay đổi màu chữ tuỳ theo trạng thái.

        input:

            - surface (pygame.Surface): Surface màn hình game

            - name (str): tên phần tử

            - selected (bool): trạng thái chọn

        Attributes:

            - color (color): màu chữ

            - name (str): tên phần tử

        '''

    def display(self, surface, name):

        '''

        Hàm thực thi các method của class Item.

        '''

**11. File Gacha.py**

class Gacha:

    '''

    Class tạo object Gacha dùng để quay các vật phẩm khi nhân vật nhặt rương báu

    '''

    def \_\_init\_\_(self, player):

        '''

        Hàm khởi tạo cho class Gachapon.

        input:

            player (Player): người chơi

         Attributes:

            #general setup

            - self.display\_surface (pygame.Surface): Surface màn hình game

            - self.player (Player): người chơi từ input level

            - self.font (pygame.font): font chữ

            - self.display\_gacha (bool): Biến kiểm tra gacha có đang bật hay không

            - self.rolling (bool): Biến kiểm tra gacha có đang quay hay không

            - self.give (bool): biến kiểm tra có khả năng trao đồ hay không

            - self.done (bool): biến kiểm tra gacha đã hoàn thành hay chưa

            #item creation

            - self.height (int): độ height item

            - self.width  (int): độ width item

            - self.rollable\_items (list): list các item có thể roll

            - self.attribute\_nr (int): số item có thể roll

            - self.create\_items(): hàm tạo item dựa trên các item có thể roll

            - self.rand\_item (int): random thời gian roll item

            - self.timer (int): thời gian đệm sau khi roll xong item

            #selection system

            - self.selection\_index (int): index item hiện tại

            - self.selection\_time (time): thời gian select item

            - self.can\_move (bool): biến kiểm tra khả năng di chuyển

            - self.move\_fx (pygame.mixer.Sound): âm thanh khi di chuyển

            - self.done\_fx (pygame.mixer.Sound): âm thanh khi hoàn thành gacha

        '''

        #general setup

    def rolling\_item(self):

        '''

        Hàm roll item cho Gachapon.

        input:

            - self.rand\_item (int): random thời gian roll item

            - self.rolling (bool): Biến kiểm tra gacha có đang quay hay không

            - self.can\_move (bool): biến kiểm tra khả năng di chuyển

            - self.selection\_index (int): index item hiện tại

            - self.attribute\_nr (int): số item có thể roll

        output:

            Nếu hoàn thành xong việc lấy index item roll ra thì set self.rolling = False

        '''

    def give\_item(self):

        '''

        Hàm đưa item cho người chơi

        Attributes:

            - self.gift

        output:

            - Tạo biến gift = item được chọn

            - Tăng chỉ số thuộc tính cho người chơi dựa trên tên item

            - set self.give = False

        '''

    def create\_items(self):

        '''

        Hàm tạo các phần tử trong list item có thể roll

        Attributes:

            - self.item\_list (list): list các item

            - full\_width (int): độ rộng màn hình

            - increment (int): khoảng cách giữa các phần tử

            - left (int): vị trí các phần tử

            - item (Item): tạo object item

        output: thêm các item vào list tương ứng

        '''

    def selection\_cooldown(self):

        '''

        Hàm cooldown khi roll các phần tử.

        '''

    def display(self):

        '''

        Hàm thực thi các method của class Gacha, đồng thời hiển thị các phần tử item

        '''

    def isdone(self):

        '''

        Hàm set trạng thái biến display\_gacha = false

        '''

class Item:

    '''

    Class Item chuyên dùng để tạo các phần tử cho class Gacha.

    '''

    def \_\_init\_\_(self, l, t, w, h, index, font):

        '''

        Hàm khởi tạo cho class Item (Gachapon)

        input:

            - l (int): vị trí left

            - t (int): vị trí top

            - w (int): độ rộng width

            - h (int): độ cao height

            - index (int): index của item

            - font (pygame.font): font chữ

        Attributes:

            - self.rect (pygame.Rect): rect item

            - self.index (int): index item được chọn

            - self.font (pygame.font): font chữ

        '''

    def display\_names(self, surface, name, selected):

        '''

        Hàm hiển thị tên của phần tử, thay đổi màu chữ tuỳ theo trạng thái chưa roll và được roll trúng.

        input:

            - surface (pygame.Surface): Surface màn hình game

            - name (str): tên phần tử

            - selected (bool): trạng thái được chọn

        Attributes:

            - color (color): màu chữ

            - name (str): tên phần tử

        '''

    def display\_images(self, surface, name, selected):

        '''

        Hàm hiển thị hình ảnh của phần tử

        input:

            - surface (pygame.Surface): Surface màn hình game

            - name (str): tên phần tử

            selected (bool): trạng thái chọn

        Attributes:

            - full\_path (path): đường dẫn của hình ảnh item

            - image\_surf (pygame.Surface): surface của hình ảnh item

        '''

    def display(self, surface, selection\_num, name):

        '''

        Hàm thực thi các method của class Item và hiện tên và ảnh của item

        '''

**12. File Item.py**

class Item(Entity):

    '''

    Class này dùng để tạo các item rớt ra từ quái và âm thanh khi nhặt item

    '''

    def \_\_init\_\_(self, item\_name, pos, groups):

        '''

        Hàm khởi tạo cho class Item (Entity)

        Attributes:

            #general setup

            - self.sprite\_type : loại sprite (item)

            - self.status : trạng thái

            - self.item\_name : tên item

            #graphics setup

            - self.import\_graphics(item\_name) : đưa hình ảnh vào

            - self.image : trạng thái của hình ảnh

            - self.image(pygame.transform.scale(self.image, (64, 64))) : chuyển hình ảnh thành kích cỡ 64x64

            - self.rect : tạo rect cho hình ảnh đó

            #movement

            - self.hitbox : hitbox của item

            - self.pick\_up\_sound : âm thanh khi nhặt item

            - self.pick\_up\_sound.set\_volume(0.4) : set up độ lớn âm thanh

        '''

    def import\_graphics(self, name):

        '''

        Hàm dùng để đưa ảnh của item vào trong game

        - self.animation : lấy trạng thái hình ảnh 'idle'

        - main\_path : đường dẫn tới ảnh của item đó

        input :

        Animation trong self.animation.keys()

        ouput :

        Đưa hình ảnh và hoạt ảnh của item đó vào trong game

        '''

**13. File Timer.py**

class Timer:

    '''

    Class Timer dùng để tạo bộ đếm thời gian cho game

    '''

    def \_\_init\_\_(self):

        '''

        Hàm khởi tạo cho class Timer.

        Attributes:

            - self.accumulated\_time (int): Thời gian đã tích luỹ

            - self.start\_time (pygame.Time): Thời gian bắt đầu

            - self.started (bool): hàm kiểm tra Timer chạy hay chưa

            - self.running (bool): hàm kiểm tra Timer có đang chạy hay không

            - self.display\_surface (pygame.Surface): Surface màn hình game

            - self.font (pygame.font): font chữ

        '''

    def get(self):

        '''

        Hàm get lấy thời gian chạy của class Timer

        return:

            Nếu đang chạy thì

                - self.accumulated\_time + (pygame.time.get\_ticks() - self.start\_time)

            còn không phải thì

                - self.accumulated\_time

        '''

    def pause(self):

        '''

        Hàm tạm dừng pause cho class Timer

        '''

    def resume(self):

        '''

        Hàm tiếp tục resume cho class Timer

        '''

    def update(self):

        '''

        Hàm update hiển thị thời gian theo format định sẵn qua các tính toán

        Attributes:

            s: thời gian đã tích luỹ (giây)

        '''

**14. File Ui.py**

class UI:

    '''

    Class dùng để tạo object UI và xử lý các tương tác của giao diện UI với game.

    '''

    def \_\_init\_\_(self):

        '''

        Hàm khởi tạo cho object UI

        Attributes:

            - self.display\_surface (pygame.Surface): Surface màn hình game

            - self.font (pygame.font): font chữ chính của UI

            #bar setup

            - self.health\_bar\_rect (pygame.Rect): các thông số như độ cao, độ rộng, vị trí cho health bar

            - self.energy\_bar\_rect (pygame.Rect): các thông số như độ cao, độ rộng, vị trí cho energy bar

            #convert weapon dictionary

            - self.weapon\_graphics (list): list chứa graphics của vũ khí

            #convert magic dictionary

            - self.magic\_graphics (list): list chứa graphics của các phép

        '''

    def show\_bar(self, current, max\_amount, bg\_rect, color):

        '''

        Hàm vẽ energy và health bar lên màn hình.

        Đồng thời cập nhật thanh energy và health bar

        input:

            - current (int): số lượng hiện tại

            - max\_amount (int): số lượng max

            - bg\_rect (pygame.Rect): rect background cho các thanh

            - color (color): màu của thanh trạng thái

        '''

    def show\_exp(self,exp):

        '''

        Hàm thể hiện số lượng exp người chơi mà hiện tại đang có

        input:

            exp (int): Exp của người chơi

        '''

    def show\_difficulty(self, difficulty):

        '''

        Hàm hiển thị độ khó hiện tại của level

        input:

            difficulty (float): độ khó của màn

        '''

    def weapon\_overlay(self, weapon\_index, has\_switched):

        '''

        Hàm hiển thị khung chứa vũ khí

        input:

            - weapon\_index (int): index vũ khí hiện tại

            - has\_switched (bool): kiểm tra khả năng switch vũ khí hiện tại

        output:

            - blit khung hình và vũ khí hiện tại lên màn hình

        '''

    def magic\_overlay(self, magic\_index, has\_switched):

        '''

        Hàm hiển thị khung magic

        input:

            - maigc\_index (int): index magic hiện tại

            - has\_switched (bool): kiểm tra khả năng switch magic hiện tại

        output:

            - blit khung hình và magic hiện tại lên màn hình

        '''

    def selection\_box(self,left,top, has\_switched):

        '''

        Hàm vẽ khung lựa chọn khi thay đổi vũ khí/magic

        input:

            - left (int): toạ độ left

            - top (int): toạ độ top

            - has\_switched (bool): kiểm tra khả năng switch vũ khí/magic hiện tại

        output:

            - Vẽ khung UI\_BORDER\_COLOR\_ACTIVE cho khung đang được switch

            - Hiển thị UI\_BORDER\_COLOR cho khung đã switch xong

        '''

    def display(self, player, difficulty):

        '''

        Hàm chạy các method của class UI

        '''

**15. File support.py**

def import\_csv\_layout(path):

    '''

    Hàm import các file csv vào list

    Attributes:

        - terrain\_map (list): chứa các list chứa các ký hiệu của file csv

    return:

        - List chứa nhiều list tạo thành ma trận 2 chiều có thông tin của bản đồ

    '''

def import\_folder(path):

    '''

    Hàm import folder, đường dẫn đến folder chứa các file hình ảnh

    Attributes:

        surface\_list (list): chứa đường dẫn đến tất cả các file trong folder

    return:

        list chứa đường dẫn đến các file trong folder

    '''

**16. File settings.py**

'''

File chứa các cài đặt chung của game như:

- Kích thước màn hình chơi game

- Các hitbox offset

- Thiết lập kích thước, màu sắc, font chữ của UI

- Các loại màu sắc cho menu, background, border background, text,...

- Các thông tin, thuộc tính của weapon, magic, quái thú

'''