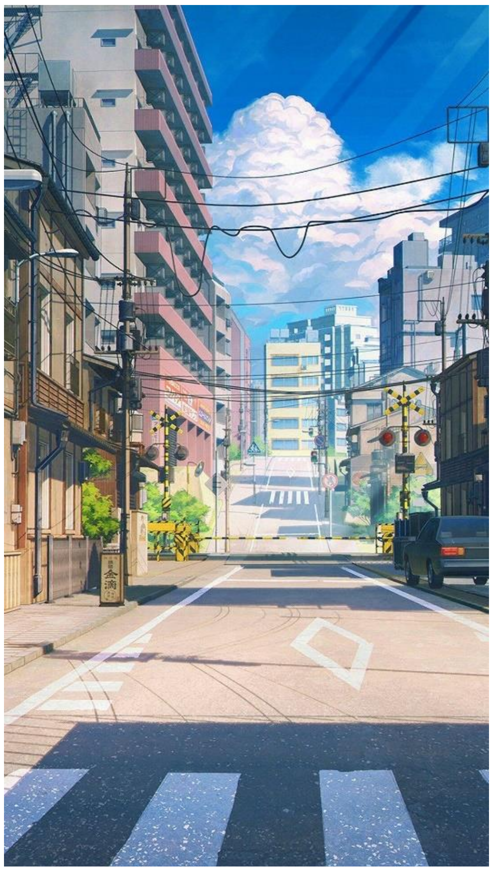
v



RPG Video Game

FINAL PROJECT

Haramain Emir Ben | Algorithm and Programming | ID: 2602206770

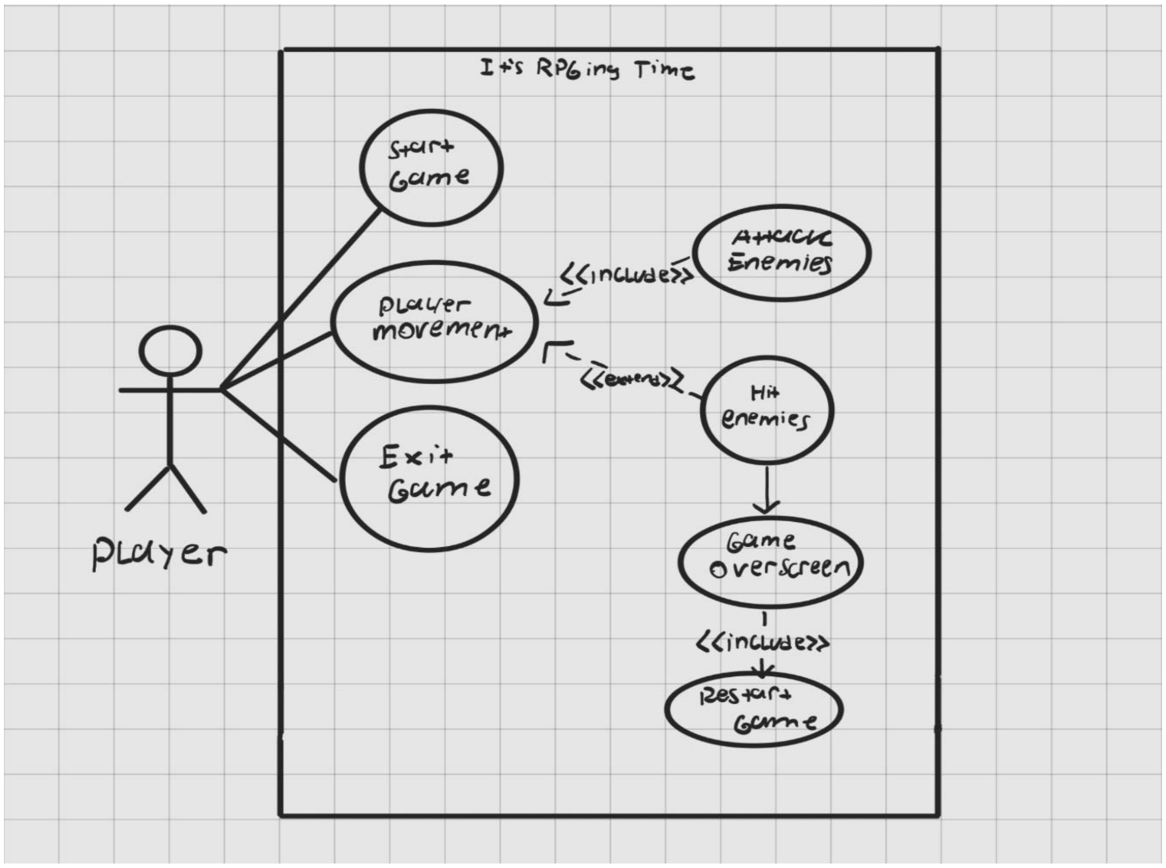
BRIEF DESCRIPTION

In this report, my final project for the algorithm and programming course I choose to create a video game, because I have no ideas left and, in my mind, this is only the option for me to make.

For this project I created an RPG style video game where you can move around and encounter enemies on the map, and if you touched the enemies you will die, and you will be sent to a game over screen but you can also restart.

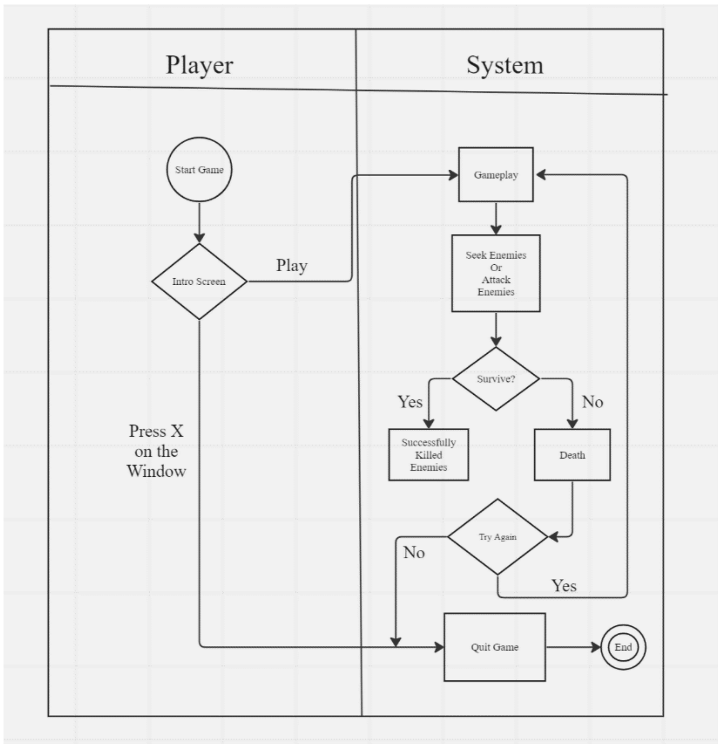
You also can attack an enemy and doing that will lead to the enemy disappear or killed to be exact.

USE CASE DIAGRAM

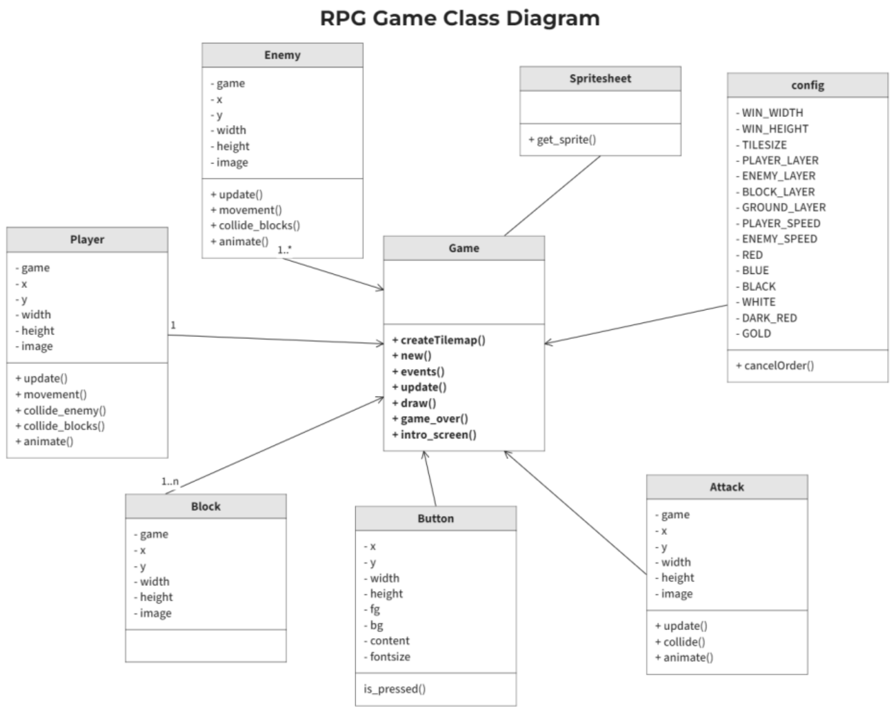


This is the Use case diagram for my Game project and in here it is explained that my program allows the player to attack the enemies and got hit by the enemy or in this case collide which result the game to be over, but you are able to restart the game every time you lost.

ACTIVITY DIAGRAM



CLASS DIAGRAM



MODULES

Here are the modules that I used in my program.

- PyGame

Pygame is my preferred module to create this RPG game because it is the one where I am most familiar because it is what I learned my class on university.

The Python programming language can **be used to create multimedia applications like video games** using the cross-platform Pygame package, which is free and open source. It employs a straightforward DirectMedia Layer library as well as numerous other well-known libraries to abstract the most typical functionalities, making it easier to write these apps.

- Sys

The sys module in Python **provides various functions and variables that are used to manipulate different parts of the Python runtime environment**. It allows operating on the interpreter as it provides access to the variables and functions that interact strongly with the interpreter.

By using a sys modules, you can use it for ending the program, but more importantly the module has a relevance with pygame so I always use this module if I were to create a game using pygame nonetheless. For example: pygame can use the SysFont to render any available sys modules 'font.

- Math

The Python 3 standard library includes a built-in module called math that offers common mathematical constants and functions. Numerous mathematical operations, including numerical, trigonometric, logarithmic, and exponential computations, can be carried out with the math module.

Basically, you can implement a mathematical equation inside your code. Such as constant provided by the math module. For Example:

- Pi
- Infinity
- Not a Number (NaN)

Having such constant saves the time writing the value of each constant every time we want to use it and that too with great precision.

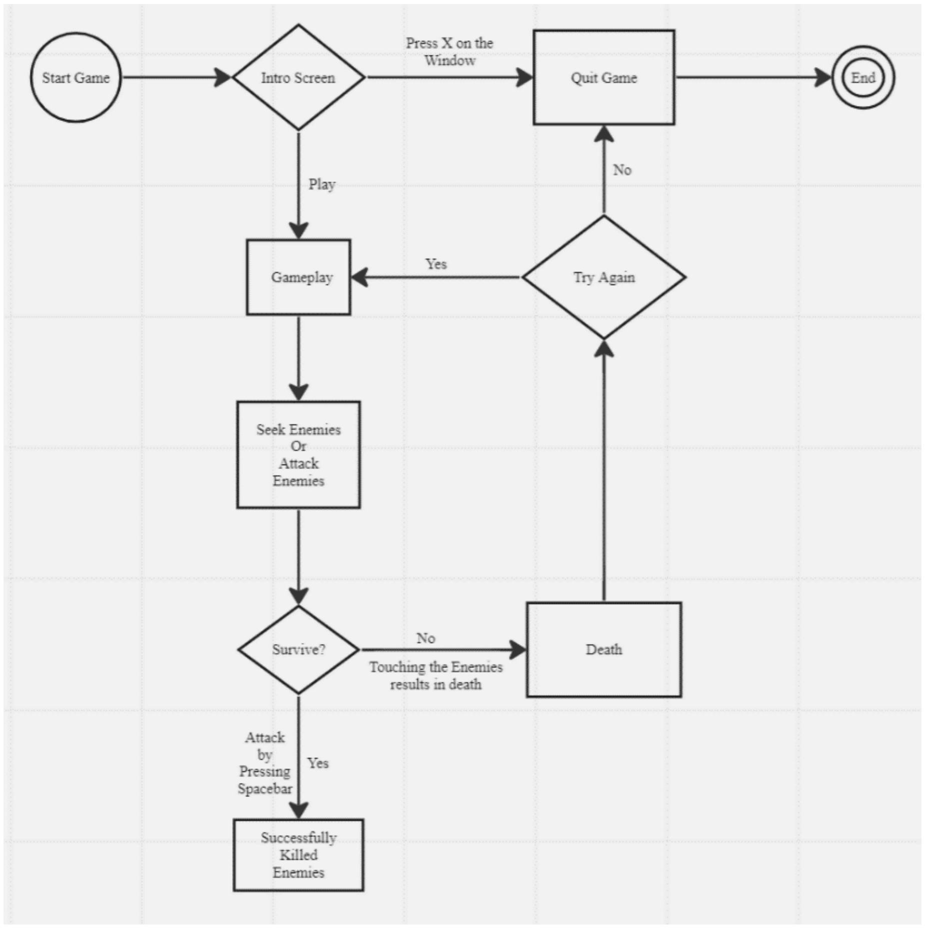
- Random

A Python built-in module called Python Random is used to create random integers. These numbers are not actually random because they are pseudo-random. This module can be used to generate random numbers, print a random value for a list or string, and do other random operations.

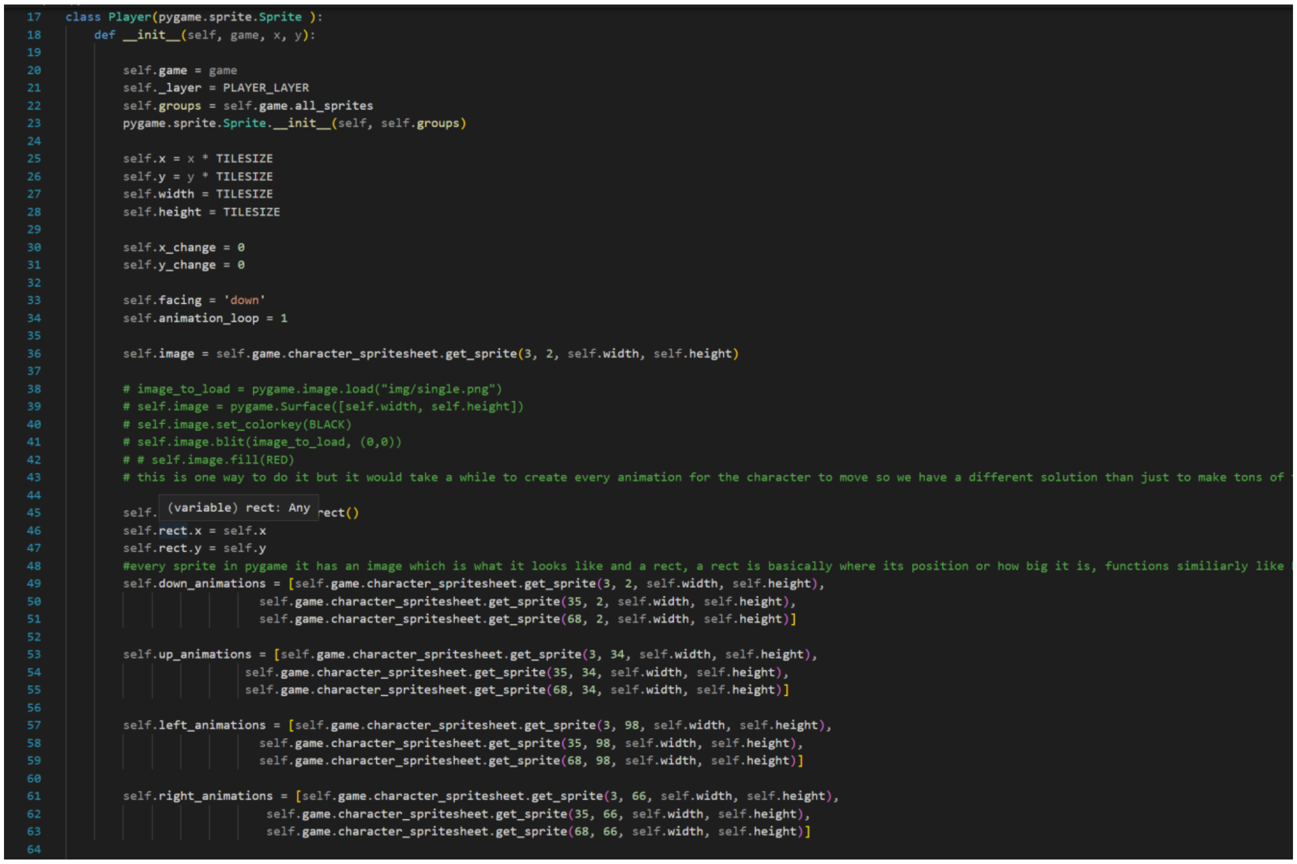
For Example :

- randint() = returns a random number between the given range

ESSENTIAL ALGORITHMS


Start Game
Intro Screen
Press X on the Window
Quit Game
End
Play
No
Gameplay
Yes
Try Again
Seek Enemies Or Attack Enemies
Survive?
No
Touching the Enemies results in death
Death
Attack by Pressing Spacebar
Yes
Successfully Killed Enemies

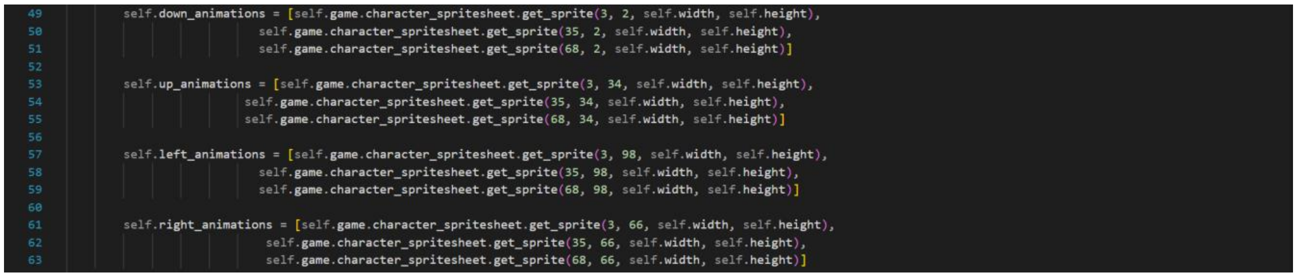
This is the flowchart for the program that I made, but for the essential algorithms I will explain very briefly in here.



```
class Player(pygame.sprite.Sprite ):
    def __init__(self, game, x, y):

        self.game = game
        self._layer = PLAYER_LAYER
        self.groups = self.game.all_sprites
        pygame.sprite.Sprite.__init__(self, self.groups)

        self.x = x * TILESIZE
        self.y = y * TILESIZE
        self.width = TILESIZE
        self.height = TILESIZE

        self.x_change = 0
        self.y_change = 0

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

        self.image = self.game.character_spritesheet.get_sprite(3, 2, self.width, self.height)

        # image_to_load = pygame.image.load("img/single.png")
        # self.image = pygame.Surface([self.width, self.height])
        # self.image.set_colorkey(BLACK)
        # self.image.blit(image_to_load, (0,0))
        # # self.image.fill(RED)
        # this is one way to do it but it would take a while to create every animation for the character to move so we have a different solution than just to make tons of

        self. (variable) rect: Any rect()
        self.rect.x = self.x
        self.rect.y = self.y
        #every sprite in pygame it has an image which is what it looks like and a rect, a rect is basically where its position or how big it is, functions similiarly like
        self.down_animations = [self.game.character_spritesheet.get_sprite(3, 2, self.width, self.height),
                                self.game.character_spritesheet.get_sprite(35, 2, self.width, self.height),
                                self.game.character_spritesheet.get_sprite(68, 2, self.width, self.height)]

        self.up_animations = [self.game.character_spritesheet.get_sprite(3, 34, self.width, self.height),
                              self.game.character_spritesheet.get_sprite(35, 34, self.width, self.height),
                              self.game.character_spritesheet.get_sprite(68, 34, self.width, self.height)]

        self.left_animations = [self.game.character_spritesheet.get_sprite(3, 98, self.width, self.height),
                                self.game.character_spritesheet.get_sprite(35, 98, self.width, self.height),
                                self.game.character_spritesheet.get_sprite(68, 98, self.width, self.height)]

        self.right_animations = [self.game.character_spritesheet.get_sprite(3, 66, self.width, self.height),
                                 self.game.character_spritesheet.get_sprite(35, 66, self.width, self.height),
                                 self.game.character_spritesheet.get_sprite(68, 66, self.width, self.height)]

```

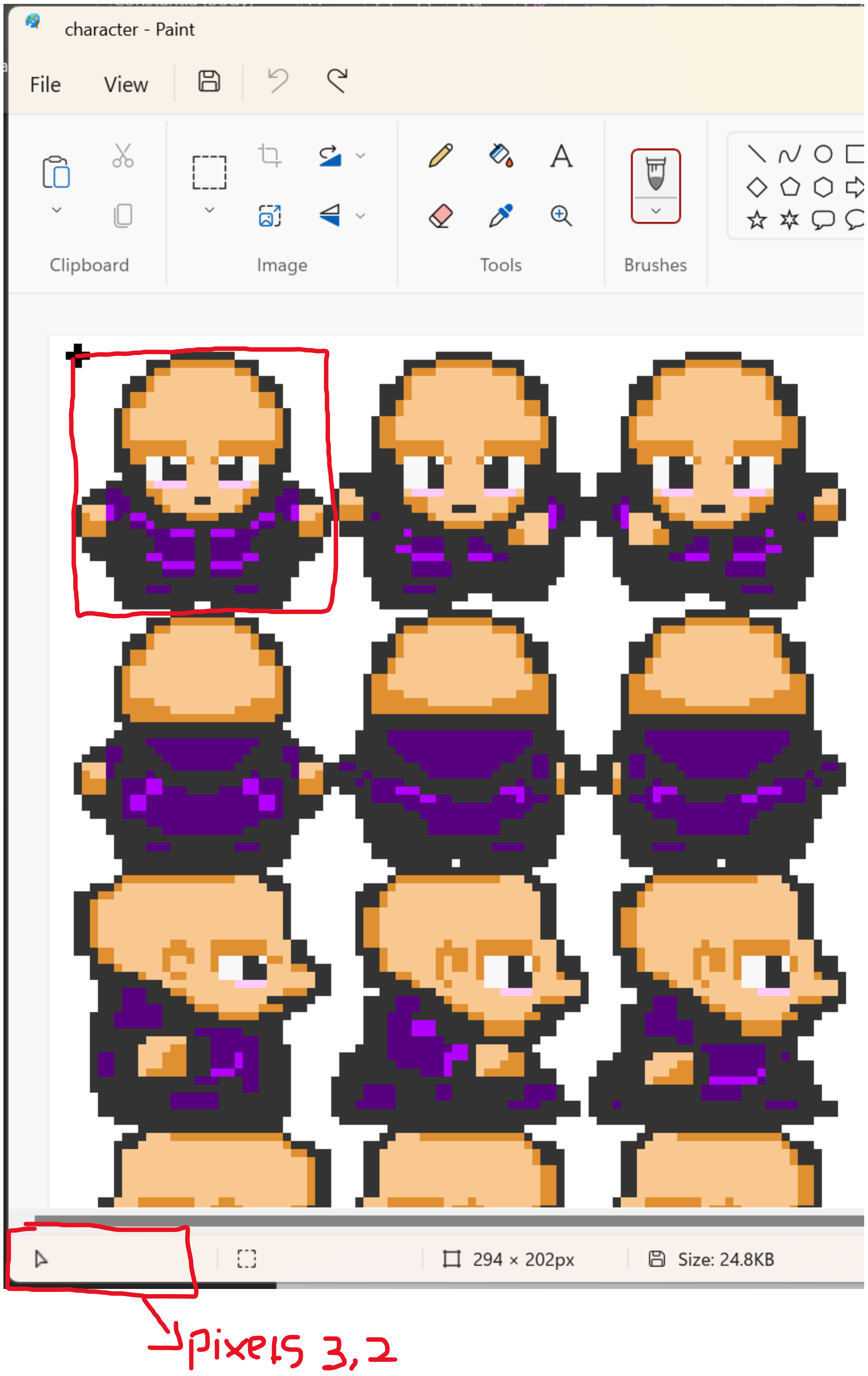
The codes above shows the Class for the Player and if you see on the bottom it shows that it is inside the init method at first I put the animation code on the animate() method on the class player and apparently this way it is very inefficient rather than copying this long list of codes I prefer to put the list on the init method and in turn I can just call these method into every other method inside of the class.

And quick note if you see from above on the Class Player(pygame.sprite.Sprite): it basically means that the class is inherits something from the pygame module, what it does is to make sprites easier to be exact.

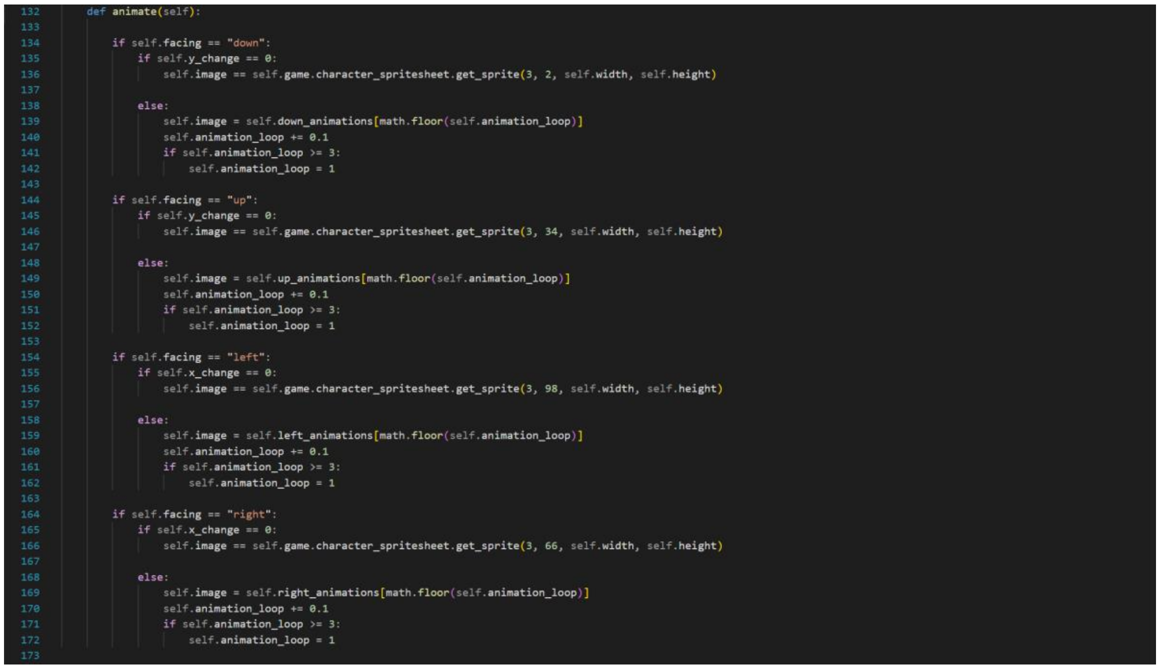


```
        self.down_animations = [self.game.character_spritesheet.get_sprite(3, 2, self.width, self.height),
                                self.game.character_spritesheet.get_sprite(35, 2, self.width, self.height),
                                self.game.character_spritesheet.get_sprite(68, 2, self.width, self.height)]

        self.up_animations = [self.game.character_spritesheet.get_sprite(3, 34, self.width, self.height),
                              self.game.character_spritesheet.get_sprite(35, 34, self.width, self.height),
                              self.game.character_spritesheet.get_sprite(68, 34, self.width, self.height)]

        self.left_animations = [self.game.character_spritesheet.get_sprite(3, 98, self.width, self.height),
                                self.game.character_spritesheet.get_sprite(35, 98, self.width, self.height),
                                self.game.character_spritesheet.get_sprite(68, 98, self.width, self.height)]

        self.right_animations = [self.game.character_spritesheet.get_sprite(3, 66, self.width, self.height),
                                 self.game.character_spritesheet.get_sprite(35, 66, self.width, self.height),
                                 self.game.character_spritesheet.get_sprite(68, 66, self.width, self.height)]
```

And how exactly do we count the math on the character sprite sheet?, well we need to see the sprite sheet of course in this case we will see the Character spritesheet.



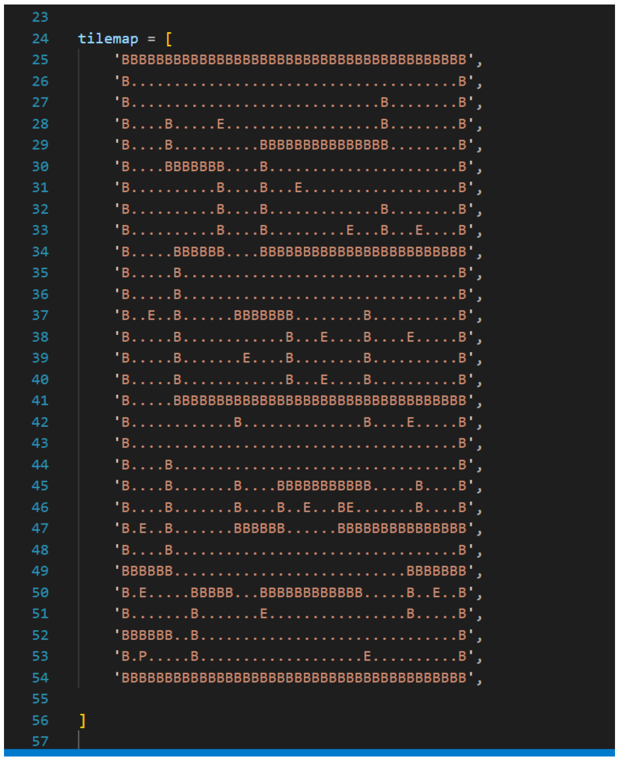
As you can see we need to count the pixels that are in the top left corner of each sprite sheet and from that on it will create a rectangle or rect to calculate the size of the sprite. To view it easy I use ms paint and the + symbol on top of the first character



```python
    def animate(self):

        if self.facing == "down":
            if self.y_change == 0:
                self.image == self.game.character_spritesheet.get_sprite(3, 2, self.width, self.height)

            else:
                self.image = self.down_animations[math.floor(self.animation_loop)]
                self.animation_loop += 0.1
                if self.animation_loop >= 3:
                    self.animation_loop = 1

        if self.facing == "up":
            if self.y_change == 0:
                self.image == self.game.character_spritesheet.get_sprite(3, 34, self.width, self.height)

            else:
                self.image = self.up_animations[math.floor(self.animation_loop)]
                self.animation_loop += 0.1
                if self.animation_loop >= 3:
                    self.animation_loop = 1

        if self.facing == "left":
            if self.x_change == 0:
                self.image == self.game.character_spritesheet.get_sprite(3, 98, self.width, self.height)

            else:
                self.image = self.left_animations[math.floor(self.animation_loop)]
                self.animation_loop += 0.1
                if self.animation_loop >= 3:
                    self.animation_loop = 1

        if self.facing == "right":
            if self.x_change == 0:
                self.image == self.game.character_spritesheet.get_sprite(3, 66, self.width, self.height)

            else:
                self.image = self.right_animations[math.floor(self.animation_loop)]
                self.animation_loop += 0.1
                if self.animation_loop >= 3:
                    self.animation_loop = 1
```

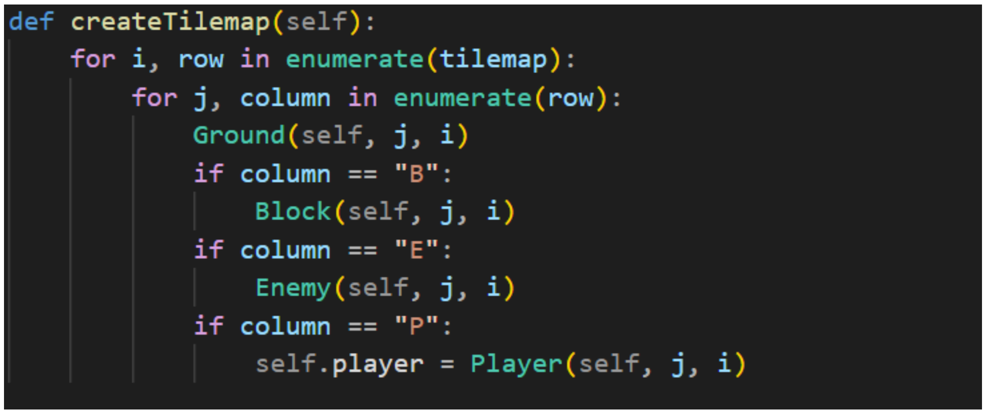
And this are the codes for all the If statements for each direction the player will face

And if you see my file on my repository on the sprite.py file I did this for the Enemy Class and the Attack class. As well.



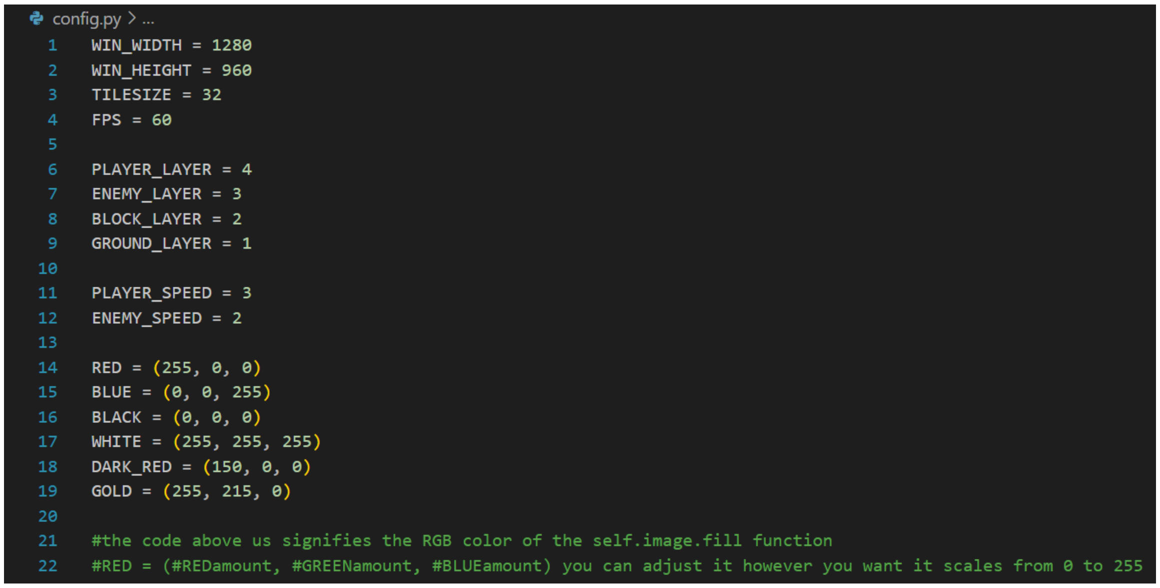
```python
tilemap = [
    'BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB',
    'B......................................B',
    'B.............................B........B',
    'B....B.....E..................B........B',
    'B....B..........BBBBBBBBBBBBBBB........B',
    'B....BBBBBBB....B......................B',
    'B..........B....B...E..................B',
    'B..........B....B.............B........B',
    'B..........B....B.........E...B...E....B',
    'B.....BBBBBB....BBBBBBBBBBBBBBBBBBBBBBBB',
    'B.....B................................B',
    'B.....B................................B',
    'B..E..B......BBBBBBB........B..........B',
    'B.....B............B...E....B....E.....B',
    'B.....B.......E....B........B..........B',
    'B.....B............B...E....B..........B',
    'B.....BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB',
    'B............B..............B....E.....B',
    'B......................................B',
    'B....B.................................B',
    'B....B.......B....BBBBBBBBBBB.....B....B',
    'B....B.......B....B..E...BE.......B....B',
    'B.E..B.......BBBBBB......BBBBBBBBBBBBBBB',
    'B....B.................................B',
    'BBBBBB...........................BBBBBBB',
    'B.E.....BBBBB...BBBBBBBBBBBB.....B..E..B',
    'B.......B.......E................B.....B',
    'BBBBBB..B..............................B',
    'B.P.....B...................E..........B',
    'BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB',

]
```

This is the Tilemap for designing the Map size of the Game the B stand for the Blocks The E stands for the enemy, and finally the P stands for the Player. You can change the declaration for each symbols on this map it can be anything but don't forget to declare it either way.



```python
def createTilemap(self):
    for i, row in enumerate(tilemap):
        for j, column in enumerate(row):
            Ground(self, j, i)
            if column == "B":
                Block(self, j, i)
            if column == "E":
                Enemy(self, j, i)
            if column == "P":
                self.player = Player(self, j, i)
```

As u can see above you just need to declare the string for the enemy block and player.

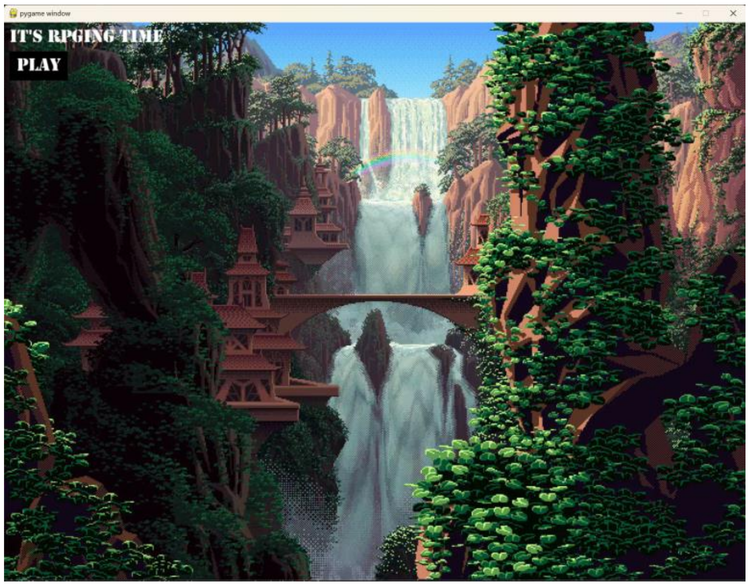


config.py > ...

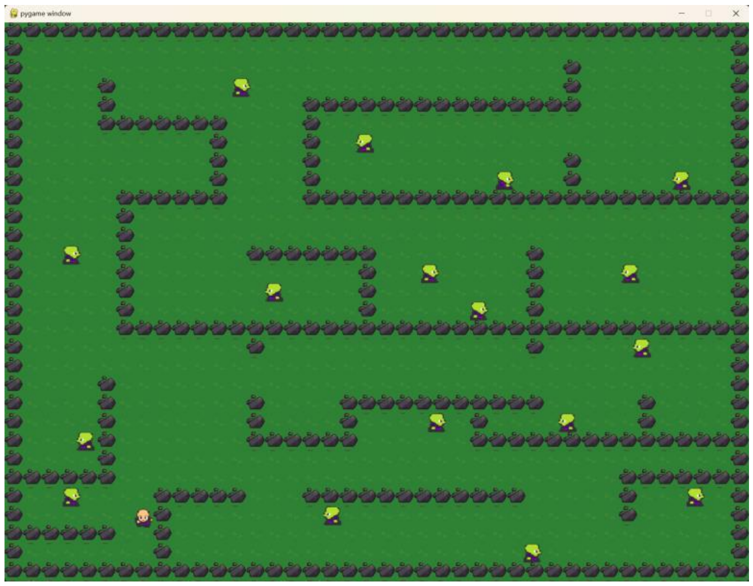
```
WIN_WIDTH = 1280
WIN_HEIGHT = 960
TILESIZE = 32
FPS = 60

PLAYER_LAYER = 4
ENEMY_LAYER = 3
BLOCK_LAYER = 2
GROUND_LAYER = 1

PLAYER_SPEED = 3
ENEMY_SPEED = 2

RED = (255, 0, 0)
BLUE = (0, 0, 255)
BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
DARK_RED = (150, 0, 0)
GOLD = (255, 215, 0)

#the code above us signifies the RGB color of the self.image.fill function
#RED = (#REDamount, #GREENamount, #BLUEamount) you can adjust it however you want it scales from 0 to 255
```

This is on the configurations for the game the WIN_WIDTH and WIN_HEIGHT is very self explanatory it is the aspect ratio for the game and you can mess around with the size as much as you can and you just need to divide the width and height with the tilesize and that shows you how much blocks you need to create for the entire screen.

SCREENSHOTS

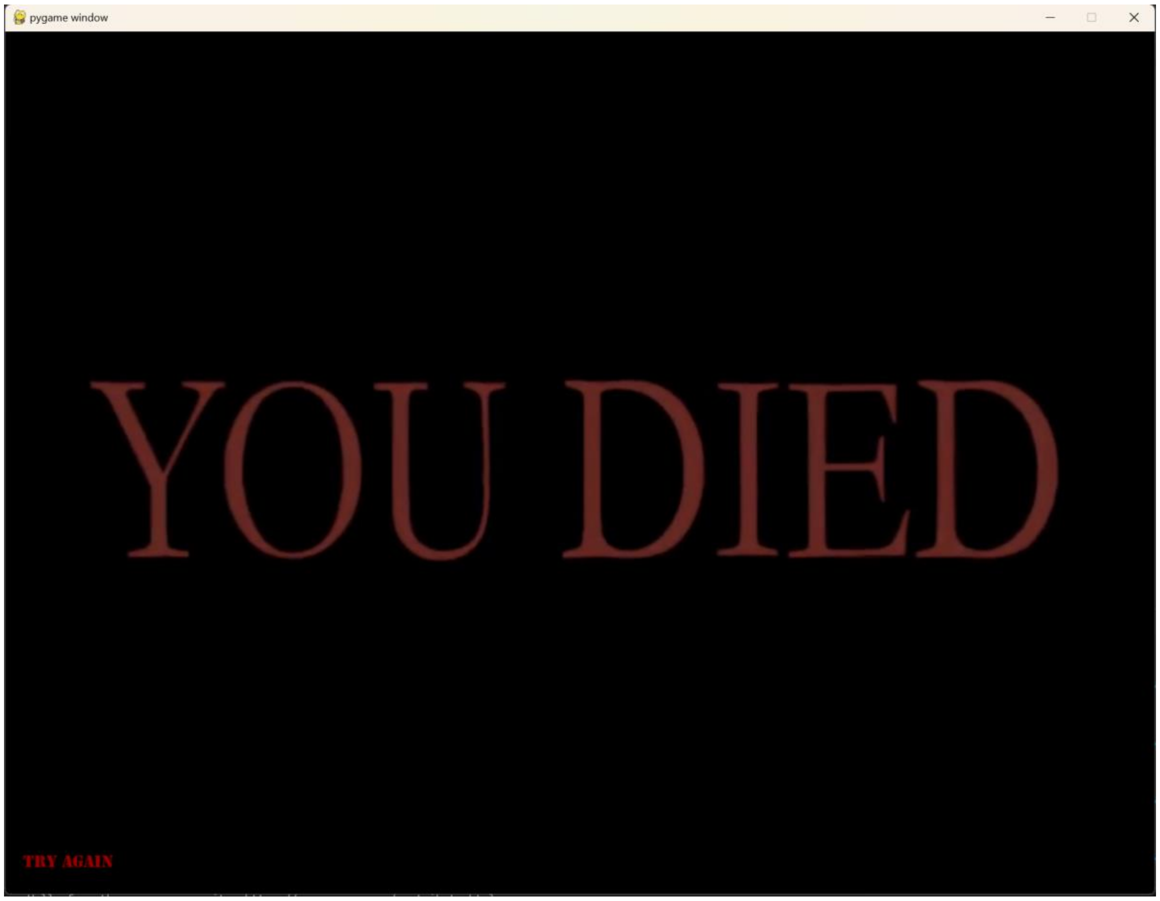


The Intro Screen, once you press the Play button Below the Title you can play the game, or you can press the X button on the window to exit anytime.



This is the gameplay you can move around the map as much as you want but watch out for the enemies because if you got caught you will be sent to the game over screen.

The enemies are the green skinned sprite that looks exactly like your character and they're moving randomly, you can attack them which leads them to disappear.



And this is the Game over screen where you can try again and retry the game again by pressing the Try Again Button on the Bottom Left corner there on the Screenshot.

REFLECTION

I learned something about pygame module which is a collection of libraries that contain tons of functions that can be used to create a game which is a back and forth searching when i were to create a game similar to mines.

I learned a lot from making this game, from making classes and functions inside of the class, at first I was not very confident in my skills as a programmer in the making especially in this case it is about game development which is the reason for me for taking this course in the first place.

I've got to give a lot of thanks to the Youtuber named ShawCode in helping me try to create a game using PyGame.

In this case I learned about how to manipulate the window screen by changing the aspect ratio in the config.py file and it corresponds to the tilesize as well for determining the amount of sprites inside the tilemap which is just to divide the window height and width with the Tilesize.