

ITP & PDM FINAL PROJECT  
COVER LETTER

|  |  |  |  |
| --- | --- | --- | --- |
| Student Information | Surname:  Hassan | Given Name:  Hassan | Student ID:  2440105785 |

Course Code: COMP6056 & COMP6502

Course Name: Program Design Methods and Introduction to Programming

Class: L1AC

Major: Computer Science

Title of Assignment: Dungeon Crawler

Name of Lecturer: Jude Joseph Lamug Martinez

Plagiarism/Cheating

Binus International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

Declaration of Originality

By signing this assignment, I understand, accept and consent to BiNus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student: Hassan Mohamed Hassan

**Program Description**:

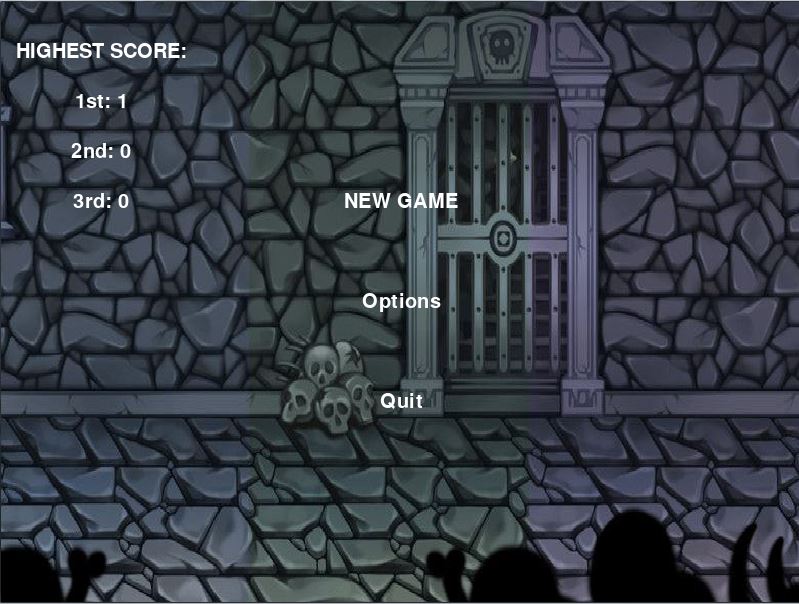
The first thing to load up is the Menu screen. It has three buttons, "New Game" starts a new game and loads up the dugeon. “Options” to change the music and sound volume, with press Backspace taking you back to the Main Menu(all settings are auto-saved). And finally, the "Quit" button to exit the program(which can also be done using the Escape(Esc) button). This Menu also shows the three Highest scores, though it always shows zeros at the first run of the game. The player can always come back to the Menu screen by pressing M though their progress will be lost. They also come back naturally after they die.

Once the game starts the player is loaded into the dugeon level and at the same time three monsters that scale to his level are auto spawned(represented by the purple vortexes). The player can use the WASD or the arrow keys to move around and must touch the vortexes to initiate battle and start the turn-based face of the game.

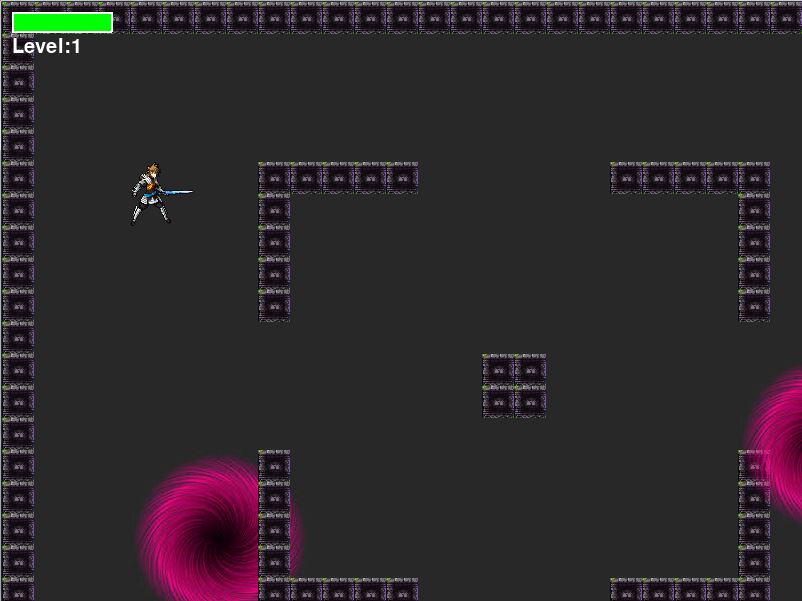
In the turn-based, the player can use the F key to attack the monster or the R key to Defend himself increasing the chances of the monster missing. And then they each take turns until one dies. If the player wins, they must go on and defeat all the other monsters in the level to advance to the next level and delve deeper into the dungeon. If the player loses their level is saved and they are sent back to the Menu. If their level was high enough it will go on the leader board.

**Game Interface:**

Game Menu:



Normal game level:



Battle Screen:

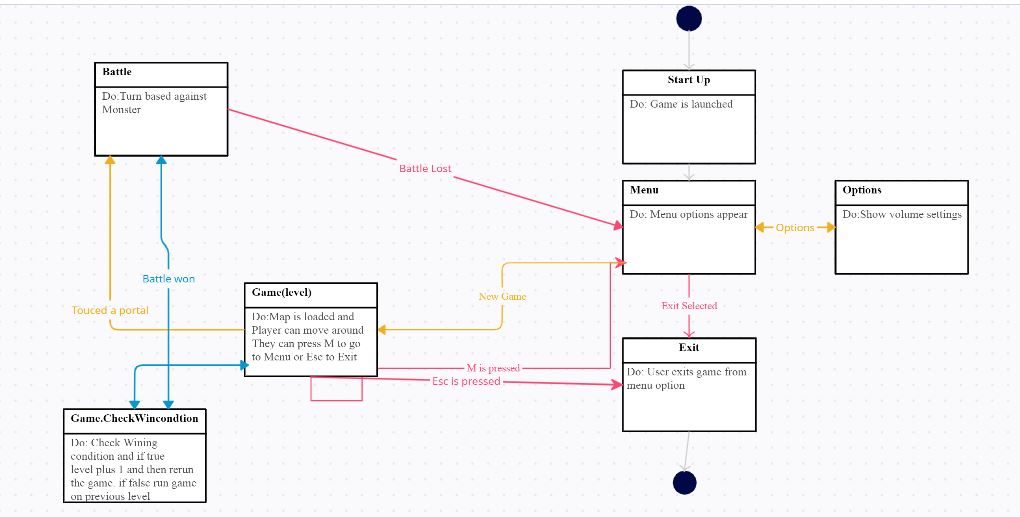


**Class Diagram:**

I decided not draw a cla diagram because no of my classes inherite each other and I make a spearate diagram for each one serves no purpose. The is inheritance as all the “sprite” classes (Wall , Event ,Player) inheritant the pygame.sprite.Sprite so they can be grouped. Instead I decided to make a game loop.

Key:

Orange is one way. Blue is a loop. Red is going back to previous face.

****

**Libraries:**

Pygame – this most popular library for developing games and GUI heavy programs. I used it to design and animate the game.

Random – used this to generate random numbers so that monsters can have a slight variation and do different damage on every attack.

Os(path) – to navigate between folders and get files.

Sys – to close the game when the player quits

**Lessons Learned:**

I learned that OOP(Object Oriented Programming) is not an easy thing to do especially if you have not planed all the class functions beforehand, something I couldn't do as it was my first time working with pygame. There were so many times I had to delete class functions because they became redundant or I have found a better way to do things. But once I figured out what I wanted to do with the game and learned more about pygame working with classes became so much fun. I did not need to worry about consistently redoing work, I just need to run the function that did those things and just change the arguments. My code became so much easier to manage and though it gets a little hectic once I need to move files around everything else was easier.

I also learned a lot about game development, especially about frames and game loops. It was my first time working with a code that is constantly running and for a while, it kept throwing my calculations off. As "a = a+2" is not run one time but once every second. I learned about using class methods and if statements to overcome these and hurdles and keep the game running. All and all it was a really fun and educational project and certainly I want to keep working. I got to add that magic at some point.

**Code Explained:**

The game mainly consitence of 2 major loop with smaller loops inside of it. The main loop is the Menu loop that keeps the menu loaded and manges all the user interactions with the menu. The other is the main Game loop wich manges the player interactions with game itself.

The Program begins with:

while True:  
 g.MainMenu()

and this loop never ends until the program is closed. This what allows the player to always go back to the starter screen every time instead of the game shutting down.

Inside of the Menu loop are few smaller loops that handle the player interactions with the menu. Those are the main menu loop and options loop. This is done so that when a player clicks on a specific part of the screen, they can get two different results depending on the loop they are in. You must remember that in the way the code is written there is only one screen so all the "buttons" are on this screen and sometimes their coordinates might overlap so the game needs which button is active and which is not. Like down below.

for event in pg.event.get():  
 if event.type == pg.QUIT:  
 self.quit()  
 if event.type == pg.KEYDOWN:  
 if event.key == pg.K\_ESCAPE:  
 self.quit()  
 if event.type == pg.MOUSEBUTTONDOWN:  
 if pg.mouse.get\_pressed()[0] and self.MainMenu\_on == True:

...  
 if pg.mouse.get\_pressed()[0] and self.Options\_on== True:

...

And now based on the user input and the loop current running the game figures out what button was pressed and successfully runs the corresponding buttons. But calling them buttons would be a lie as we are merely checking the coordinates of where the mouse clicked on and checking if it falls on where we have displayed some text previously. And with this example below the code is written so that the "button" is accurate no matter the screen size. (in the example bellow pos is a tuple with the mouse x,y)

elif(pos[1] > (HEIGHT/ 2)-18 and pos[1] < (HEIGHT/ 2)+18):  
 print("Options")  
 self.Options\_on = True  
 self.Options()

...

self.draw\_text('Options', 20, WIDTH / 2, HEIGHT / 2)

...

def Options(self):  
 self.MainMenu\_on = False...

The same principle of smaller loops within the main loop applies for the game loop. But this time the smaller loops are not checking for user-input but game faces. There are two faces within the game. This is the exploration face and battle face. In the exploration face, the player can walk around and interact with the map. While the battle face is a turn-based battle between the player and a random battle where there is no movement.

Let us start with the exploration face as that is the first loop that starts when the game is loaded. Before anything happens the game first loads up all the pictures, file paths, and maps at the creation of the game object.

def load\_data(self,volume\_Music,volume\_Sound):  
 game\_folder = path.dirname(\_\_file\_\_)  
 Map\_folder = (path.join(game\_folder, '../files/Maps'))...

And then a random map is generated using one of the two map.txt files. The code for the generation is very simple, a for loop that goes through every character within the file and loads it as a tile on the screen where running the corresponding classes.

for row, tiles in enumerate(self.map.map\_data):  
 for col, tile in enumerate(tiles):  
 if tile == '1':  
 Wall(self, col, row)  
 if tile == 'P':  
 self.player = Player.Player(self, col, row,"Rick")  
 if tile == 'M':  
 self.pionts\_goal = self.pionts\_goal +1  
 Event(self, col, row)

Where there is '1' a wall is loaded and where is an 'M' the portal for the Battle face, we will get there, is loaded, and finally, the player is loaded on 'P'. The reason why Wall and Eent classes objects are not assigned to any specific variable is that because they are all under the same sprite group and the program updates them all automatically. (self.pionts\_goal is there so the game knows how many monsters must be killed later all in the winning condition.)

After this, the main game loop strats running and within it, the exploration loop.

while True: 🡨Main game loop  
 g.new()  
 g.run(level) 🡨Exploration Loop  
 if not g.LevelCleared:<-checks if the level is cleared,all monsters are killed)  
 if not g.Menu:🡨 (checks if the player has not pressed M to go to main menu)  
 g.gg() 🡨 rungs the death screen  
 break 🡨 breaks the main loop and goes to menu  
 else:  
 g.ggez() 🡨 plays the win screen  
 level = level +1 🡨 Tells the game loop that the plyaer has leveled up

----^Main game loop----------------v Exploration loop

def run(self,level):

while self.playing:  
 self.dt = self.clock.tick(self.FPS)/1000  
 self.events()  
 self.update()  
 self.draw()

This loop(the def run loop) is straight forward. It will play as long as the level is not cleared. It checks for events which are player keyboard inputs, it updates all the sprites(things like movement are calculated here) and finally, all the new calculations are drawn on the screen. And this completion of the loop is what is normally called a frame.

Now when the player touches the portal or an "Event" object the running code loop is switch for.

def update(self):  
 if pg.sprite.spritecollideany(self, self.game.players):  
 self.kill()

...

self.game.Battle()

---------------------------------------------------------------------------------------------------------------------

#The code above is within the Event class and the one below is within the game class(do this just to show both codes at once.

def Battle(self):  
 #Generate the monster  
 self.timer = 0;  
 self.sound['Go\_Portal'].play()  
 self.AttackMode = True

When Attackmode is turned on the loop changes to the turn-based loop. In this loop. The player can’t move.

ef update(self):  
 if not self.game.AttackMode:  
 self.get\_keys()  
 self.x += self.vx \* self.game.dt

The map isn’t drawn.

def draw(self):  
 if not self.AttackMode:  
 self.screen.fill(BGCOLOR)  
 for sprite in self.all\_sprites:#This loop draws all the sprites  
 self.screen.blit(sprite.image, self.camera.apply(sprite))

Instead the battle screen above is drawn.

if self.AttackMode:  
 pg.display.set\_caption("{:.2f}".format(self.clock.get\_fps()))  
 self.screen.fill(BLACK)  
 self.screen.blit(self.Menu\_img, (0, 0))

...

And the Fight function is run.

def update(self):  
 # update the game loop  
 self.all\_sprites.update()  
 self.camera.update(self.player)  
 pg.mixer.music.set\_volume(self.volume\_Music)  
 pg.mixer.music.set\_volume(self.volume\_Sound)  
 if self.AttackMode:  
 pg.mixer.music.set\_volume(0)  
 self.Fight()🡨This guy

Before we get to the Fight function, I need to talk about how the monsters are created in the game. The Battle function doesn't just turn Attackmode on but also generates one random monster based on the player's level. And the monster is states are bases on simple formulas manipulating the player's level plus some random numbers to make the monster unique. The monster is also assigned a skillset based on its level but the code is not that complicated so I wouldn't show it.

Now to Fight function. This is also quite simple. Every time the function is called it checks if both the player and the monster are both alive. If they are then it checks who is supposed to go and then that person goes, next time it runs it checks who went last so that the other person goes. But the most interesting part of the code is that part in the middle.

elif self.player.turn == False and self.monster.turn== False:  
 time =pg.time.get\_ticks()/1000  
 self.timer= self.timer + 1  
 if self.timer >= 50:  
 print("Works")  
 if self.lastturn == "Player":

...

This is a buffer state with a timer clock. Like I said before the Fight function is called upon in every frame which is fine when doing calculations but it also means that all the animation happening within the function has to happen with that frame as well. And a single frame of animation is not an animation. That is why this buffer face exits to allow the animation and sound effects to finish before the next person takes their turn. With this the code went from Player🡪Monster🡪 to Player🡪Animation/SF🡪Monster🡪Animation/SF

After this the code is straightforward, check if the player is still alive after the fight. If no, Game over. If yes, continue exploration with one less monster. The exploration loop goes on until all the monsters are killed and then the player levels up and the game loop is run once more.

**Project Link:**

<https://github.com/hassan701/Introduction-to-programing/tree/master/Project_turnbased%20Game>

**Project Refrence:**

The Stolen Crown - A Fantasy RPG written in Python using Pygame: https://www.youtube.com/watch?v=MkZXaDQfTSo&t=1438s

Kamidori Alchemy Meister: Origenal inspirationa but i could never get the movement right.

**Credits:**

Code and Tutorials used/fllowed to develop this program:

Tile-based game development tutorial by KidsCanCode(youtube): https://www.youtube.com/watch?v=txUvD5\_ROIU&list=PLmz3pPh6QC2VOLIFgybOmZRJZGZ1HQcb0

Pygame Menu System Tutorial Part 2- Building the Menu and States - YouTube, https://www.youtube.com/watch?v=bmRFi7-gy5Y&t=17s

Pygame Turn Based Battle System Tutorial - YouTube: https://www.youtube.com/watch?v=JGxFPCkJzJM

Python Pokemon - Python Class Tutorial for Beginners - YouTube: https://i.ytimg.com/an\_webp/Pbs6jQZrZA4/mqdefault\_6s.webp?du=3000&sqp=CKTB-v8F&rs=AOn4CLCKjz7a3Z5KJO37uU7PIik\_05cjaw

**Sprites:**

Rock Golem:

https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcS0JoW2Z444aWDqP9p9NdKrcMCNdKCEGvWG6g&usqp=CAU

VINe Monster:https://i.pinimg.com/736x/df/39/a0/df39a0c939f9448cf40470791dea6a4c.jpg

Zompie wolf: https://i.pinimg.com/originals/c0/3c/48/c03c481b812ab6c521ecf0ed49bf6fa0.jpg

Tree Wolf:

https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.pngwing.com%2Fes%2Fsearch%3Fq%3DPlanta%2Bcarn%25C3%25ADvora&psig=AOvVaw1e9QlLNxfWV8m4kOV3qHDU&ust=1610613265174000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCPja2MXAmO4CFQAAAAAdAAAAABAD

Skeleton:https://s3.amazonaws.com/GameArtPartners/Images/MonsterBox1\_Gallery2.jpg

Dragon:https://i.pinimg.com/474x/4c/ab/4c/4cab4c3000198f5924bf1db7268b267d.jpg

Slime Lizard:

https://www.google.com/imgres?imgurl=https%3A%2F%2Fimg.favpng.com%2F25%2F17%2F19%2Frpg-maker-monster-role-playing-game-role-playing-video-game-legendary-creature-png-favpng-fGSyRXHQ3xJfsraRyKi2qcPNU.jpg&imgrefurl=https%3A%2F%2Ffavpng.com%2Fpng\_view%2Frpg-rpg-maker-monster-role-playing-game-role-playing-video-game-legendary-creature-png%2F8KSacNw5&tbnid=3otId6F6w4X1AM&vet=12ahUKEwjT2syNwJjuAhXKwoUKHS52AvAQMygDegQIARAf..i&docid=Dm5cB4EUoWJzHM&w=820&h=888&itg=1&q=holy%20family%20catholic%20church&hl=en-SO&ved=2ahUKEwjT2syNwJjuAhXKwoUKHS52AvAQMygDegQIARAf

Portal:https://cdn.xl.thumbs.canstockphoto.com/dark-gate-vortex-dark-portal-abstracthard-vortex-lines-version-stock-illustrations\_csp24305558.jpg

Floor(didn't use it,but..):https://cdn.gamedevmarket.net/wp-content/uploads/20191203180928/6f45c221273b7ca1fddb55ac560f5012.png

Wall:https://cdn.gamedevmarket.net/wp-content/uploads/20191203180928/6f45c221273b7ca1fddb55ac560f5012.png

**Background:**

Menu: https://assetstorev1-prd-cdn.unity3d.com/package-screenshot/df9d9252-8c14-4f62-a0a8-ad811625a7c8\_scaled.jpg

Battle: https://assetstorev1-prd-cdn.unity3d.com/package-screenshot/3ec04946-604a-4011-9e42-027d362fcbb7\_scaled.jpg

**Music and Sound:**

Monster Growl: https://www.youtube.com/watch?v=FEW5a7ebxvQ, Sound Effect

Sword Slash: https://www.youtube.com/watch?v=X3liPsg21Cg, SoundEffectsArchive

Level Up: https://www.youtube.com/watch?v=P\_u0k2uElHI, CPhT Fluke

DEAH: Dark Souls ' You Died ' Sound Effect

Portal: https://www.youtube.com/watch?v=80bdxAqF4GU, SOUND Effects Public Domain

Bensound.com 🡪 Memories

Bensound.com 🡪 Creativeminds