Generative Dungeon Crawler Game with Ai Combat Simulation, Procedural Generation, and Player Performance Analysis

*Jared Tan, Leo Hanqing, Thu Ya*

[**Abstract And Summary 0**](#_30j0zll)

[**Introduction And Background 1**](#_1fob9te)

[**Project Plan 2**](#_3znysh7)

[Aims 3](#_2et92p0)

[**Methodology 3**](#_tyjcwt)

[1) Menu Interaction 3](#_3dy6vkm)

[2) Procedural Map Generation 6](#_1t3h5sf)

[2a) Procedural Generation 6](#_4d34og8)

[2b) Player Navigation 8](#_2s8eyo1)

[3) Combat System Simulation 9](#_17dp8vu)

[3a) Ai Combat Simulation 10](#_3rdcrjn)

[3b) Defining Graphics & Environment 12](#_26in1rg)

[3) Data Analysis 15](#_djm6qjamvf5)

[**Evaluation: Retrospection 15**](#_lnxbz9)

[Limitations and Moving Forward 16](#_35nkun2)

[**References 16**](#_1ksv4uv)

[Image Sources 16](#_44sinio)

# Abstract And Summary

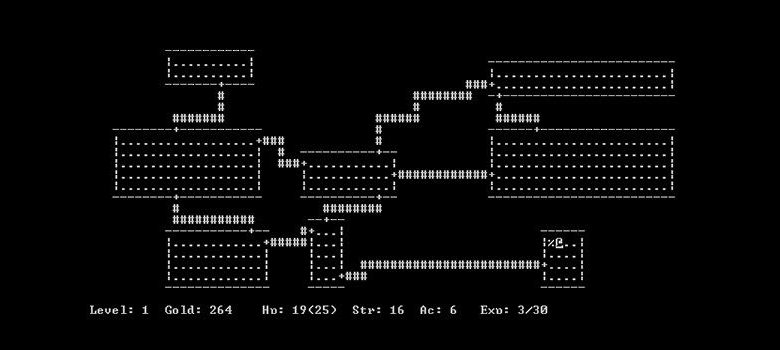
Using the Pygame framework from Python, this project investigates and implements intriguing coding concepts such as procedural content creation, ai combat simulation, and data manipulation into the genre of ‘dungeon crawler’ video games. We've created a special Pygame setting where a player must traverse and clear out an procedurally generated dungeon. This enables us to experiment with various parameters and investigate intricate emergent phenomena. We'll go into detail about the development process in the sections that follow, alongside an explanation of the genre itself and the respective code and concepts that were produced in order to create a functional demonstration of the project. The ultimate goal of this project is to demonstrate Pygame's ability to simulate and analyze complex systems in addition to producing entertaining games. This project will include information of the history and software regarding ‘Dungeon Crawlers’ whilst showcasing the development process of the game itself.

# Introduction And Background

A videogame under the ‘Dungeon Crawler’ category is a combination of multiple genres. A ‘Dungeon Crawler’ tends to include these elements: Copious amounts of rooms that are either endless due to the usage of procedural generation or containing a start and end due to maintaining a narrative agenda or in other words, in order to tell a particular story. Dungeoncrawlers also contain a common objective of clearing out rooms by clearing out enemies during combat. This unique genre also tends to include the addition of items, spells, or general loot that would aid in this venture. This genre was a convenient category to expand upon with modern technology.

This genre of games stemmed back to the tabletop role playing game known as ‘Dungeons and Dragons’ which was first revealed to the public by Gary Gygax and Dave Arnesan in 1974 as emphasized by Brewer Nathan (2016). ‘Dungeons and Dragons’ (or more informally known as ‘D&D’) was a physical board game in which a person who is appointed as the ‘Dungeon Master’ narrates and distorts the story and the events within the game to the other remaining human players, who would interact with the characters and the revolving world using role-play. Many of the items are premade and most of the combat and interaction’s successes and/or failures would be dictated by the roles of multiple sided dice which would simulate a mechanical random number generator.

The generational popularity of this game is what would fuel many aspects of commercialized video games in the coming years. In 1975, the earliest recorded role-playing game was displayed in the PLATO (Programmed Logic for Automatic Teaching Operations) which was the first generalized computer-assisted instruction system, located in the University of Illinois. ‘The Dungeon’ was a game made by Rusty Rutherford in which a player-generated character with character statistics such as Strength, Dexterity, Constitution, Intelligence, Wisdom, and Charisma would explore a premade dungeon and in each room, the player would encounter randomly generated enemies presented as fantasy monsters.



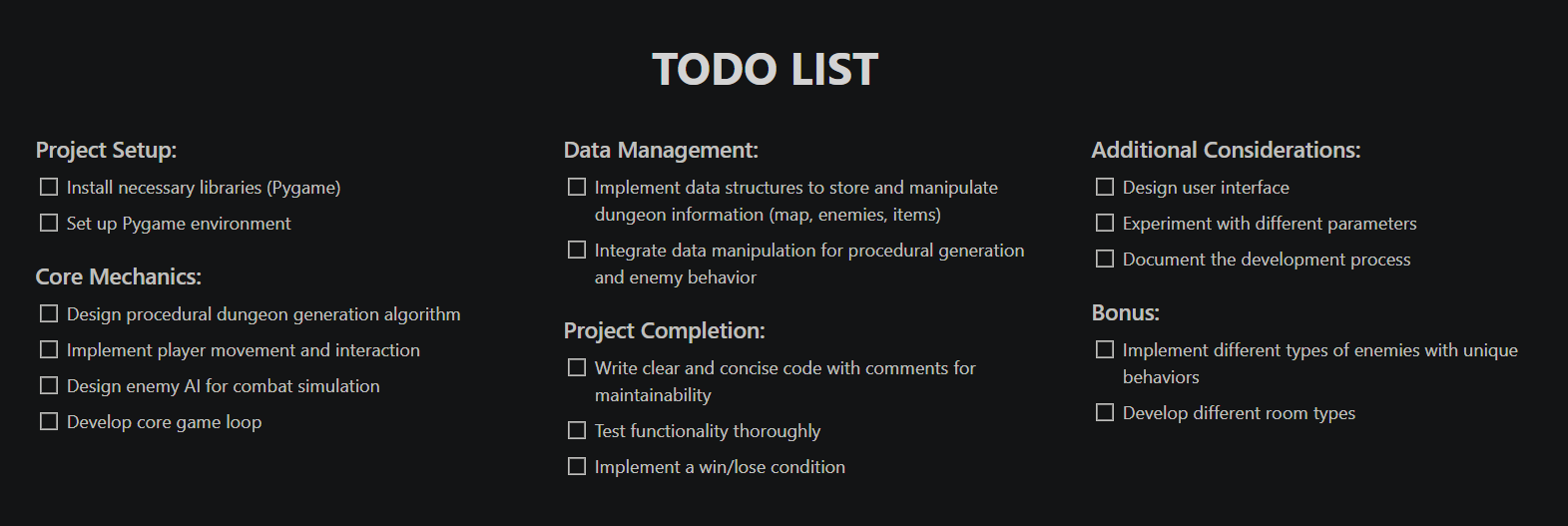
‘The Dungeon’ was a game made by Rusty Rutherford

This game would be the structure in which newer developers would expand upon. This game alongside the creation of ‘Dungeons and Dragons’ will be the structural foundation in which the game’s thematic approach and gameplay would grow upon. The code within the program could also inspire many others to implement into real world scenarios. Procedural generation and artificial intelligence simulations could be used for advancements in machine learning in order to automate experiences such as for 2D and 3D graphics, Artificial Intelligence image generation, as well as artificial general intelligence to name a few

# Project Plan

During the early stages of the project’s lifespan. A general task list was devised in order to keep track and maintain progress throughout the development of the dungeon crawler game. The todo list was made in an application known as ‘Notion’ and within it, displayed a pseudo-Kanban style task board. This todo list displays the general goals of the project itself and should have included and succinctly broken down all important aspects and factors that needed to be considered during the stages of development.

The project was intended to be made using the **AGILE** software development life cycle which includes an iterative and cyclical progression approach. This would allow all members to work on part of the code autonomously and in a laisser-faire manner without the worry of any particular member impacting the speed of the intended progression itself. Furthermore, it ensured that through these many iterations, bugs and glitches would be swiftly removed to allow for a convenient and reduced work-load on the final debugging stage nearing the end of the development on the project. In order to track and maintain the iterations, Pycharmm’s inbuilt ‘stop and debug’ function is utilized in order to conveniently check for variables during each iteration.



Initial task list made in Notion

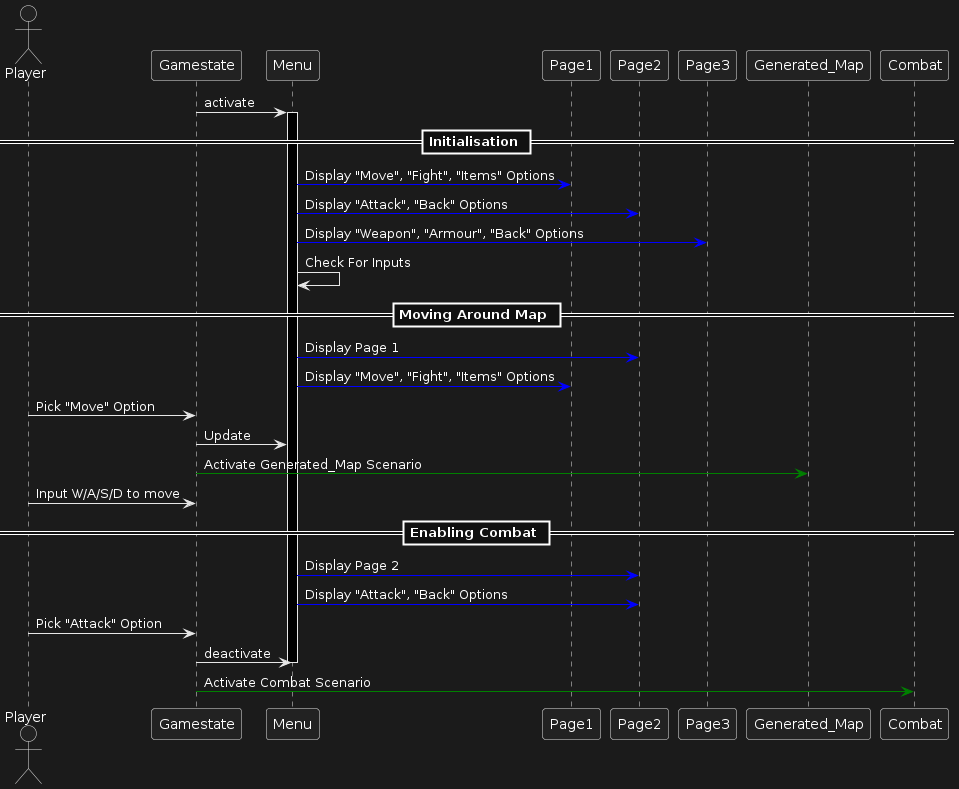
### Aims

This project aims to create a Dungeon Crawler video game using Pygame with the game having included a procedurally generated map with a fully interactable character which means having the ability to move and react. Furthermore, the game should include an artificial intelligence or algorithm in order to simulate combat with enemies. The program itself should include elements of Python, Object Oriented Programming, and should have research containing some data prior to the game’s creation.

# Methodology

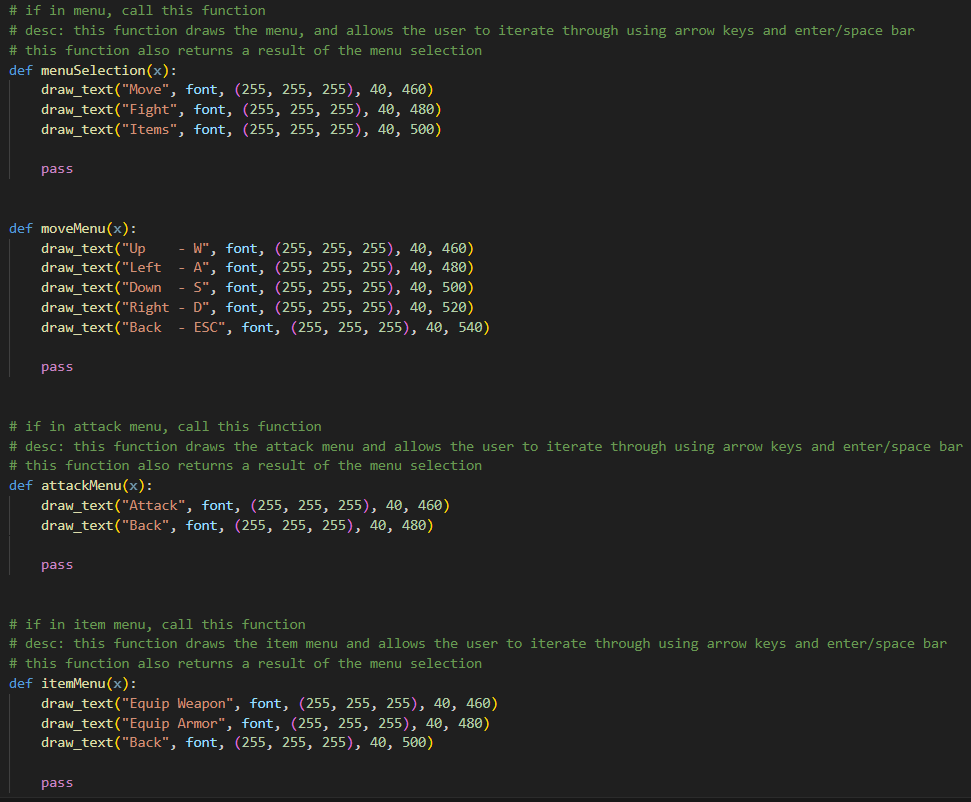
## 1) Menu Interaction

In order to navigate around the map and to customize your player, there will be an integrated menu system inside the game outside of combat scenarios. As the game is split into combat scenarios and Generated Map scenarios, this menu will only be applicable during the Generated Map scenario.



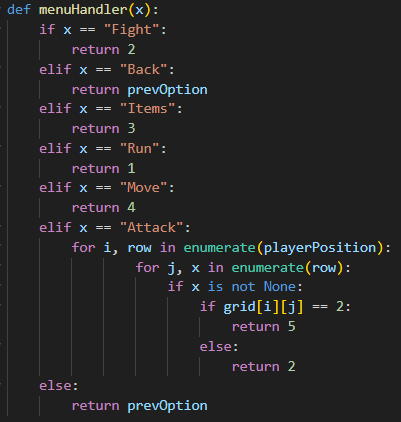
Initial UML Class Diagram for the menu system

Each of these functions draws texts in which displays the various different options the player is allowed to control during the map generation scenario. Each subsection of the menu would display multiple options alongside a dot which would show your current position on the menu itself. As shown in the attached image below, there is documentation on some of the functions which were able to be made during the limited time span.



Initial code for the menu system (Pre-integration Module)

These functions below show the various inputs that will be detected. An integer marker and switch cases were used to not only detect which option had been selected but to also trigger the following operations afterwards. Note that in order to navigate the menu, use the arrow keys and confirm with the ‘enter’ or ‘return’ key. These are conventional and intuitive user interface keys for video games in the modern era which were practiced in this project as well



Initial code for the menu system (Pre-integration Module)

## 2) Procedural Map Generation

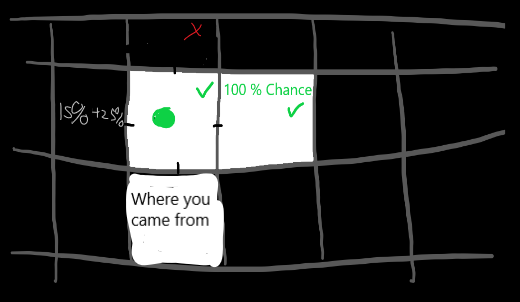
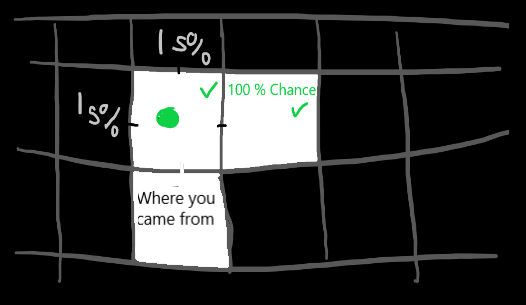
At the heart of ‘Dungeon Crawlers’ are the dungeons itself. This project contained a drastic decision in wanting to pursue a greater venture of containing a procedurally generated map as part of the gameplay loop for the project. Essentially, the player will be navigating through a grid in which will display different generated rooms in random procedurally generated directions, those in which could contain encounters and/or bosses.

### 2a) Procedural Generation

The player is represented by a green dot in a large grid.They are able to move horizontally and vertically once square at a time. In this dungeon, when the player moves to an adjacent square, check the 4 adjacent squares. One square would already be discovered as that was where the player came from. For the 3 remaining squares, there will be a 100% chance that one of them will be undiscovered (a path that a player cna travel too but has not yet). For the remaining 2 tiles, there is a base 15% chance that the tile will convert to being undiscovered.

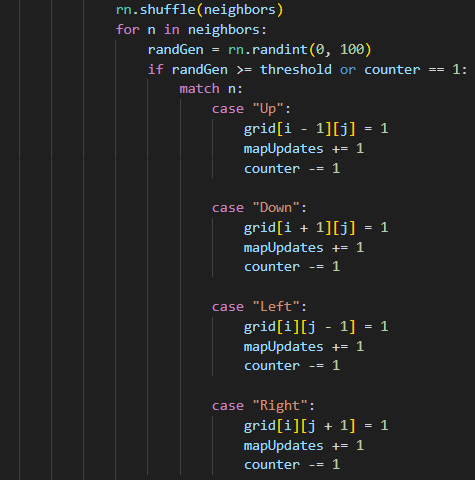
Here presents 2 situations. If the tile fails at becoming undiscovered, the tile will stay dormant and the next adjacent tile would have a 25% increase in the probability that it will become an undiscovered tile. In the second scenario, the tile becomes discovered. The next tile will continue to stay at 15% chance. Every change that occurs on the map will be recorded as the mapUpdates variable will be increased by one.

Also note that when a square is generated, white squares are discovered squares, light-gray are undiscovered (once stepped on, will be discovered), red are enemy tiles, purple are boss tiles. Both red and purple tiles trigger combat mode.



Conceptual Art for Procedural Map Generation

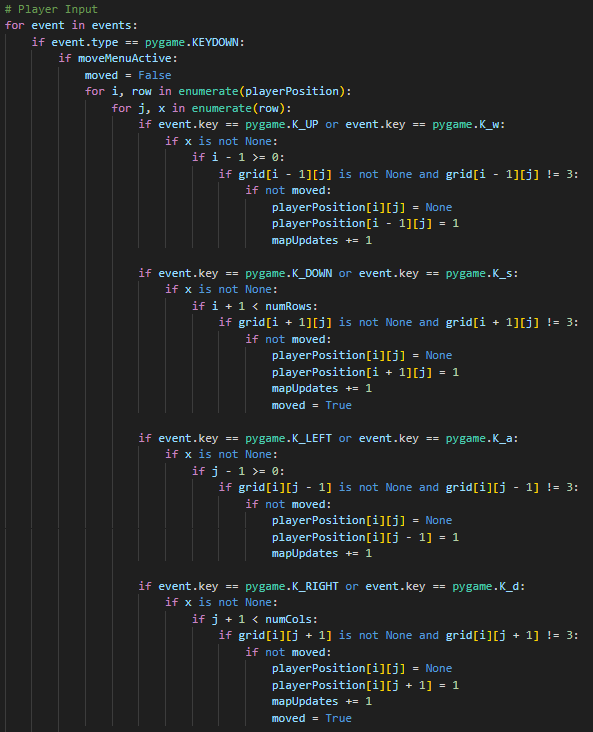




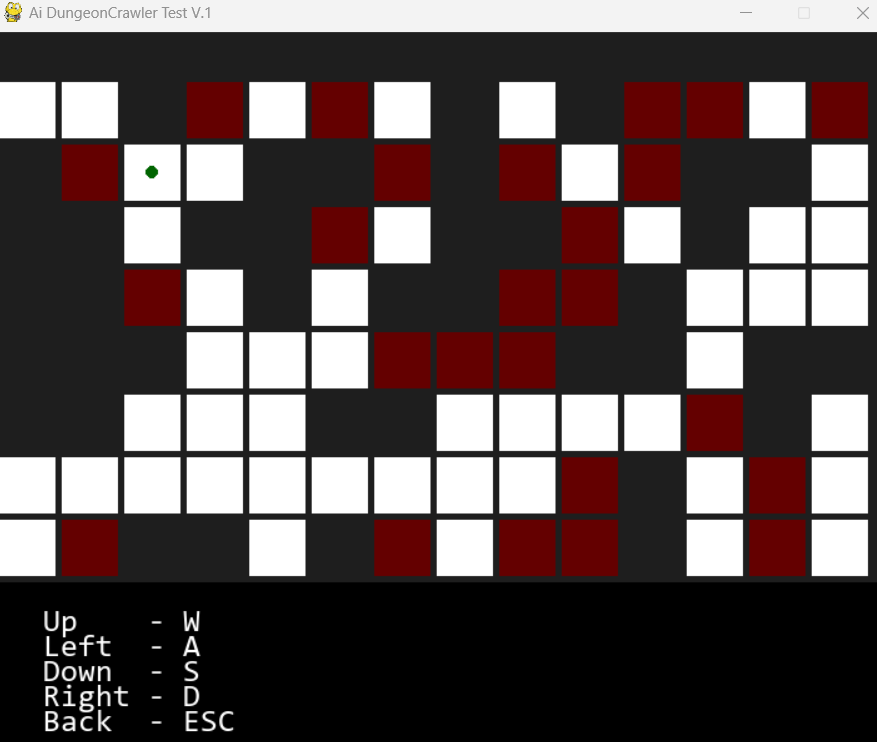
Code for Procedural Map Generation

### 2b) Player Navigation

Using the Pygame exclusive ‘events’ and ‘pygame.KEYDOWN’, the program is able to detect keyboard inputs from the player. Whilst the menu is active, enumeration is used to not only get the particular row or z coordinate in the grid but also the column or y coordinate of the grid and to move the player accordingly to the intuitive input of the W/A/S/D movement system. Go up a column when the W key is pressed, down when the S is pressed, left when the A is pressed, right when the D is pressed. Update map accordingly after movement.



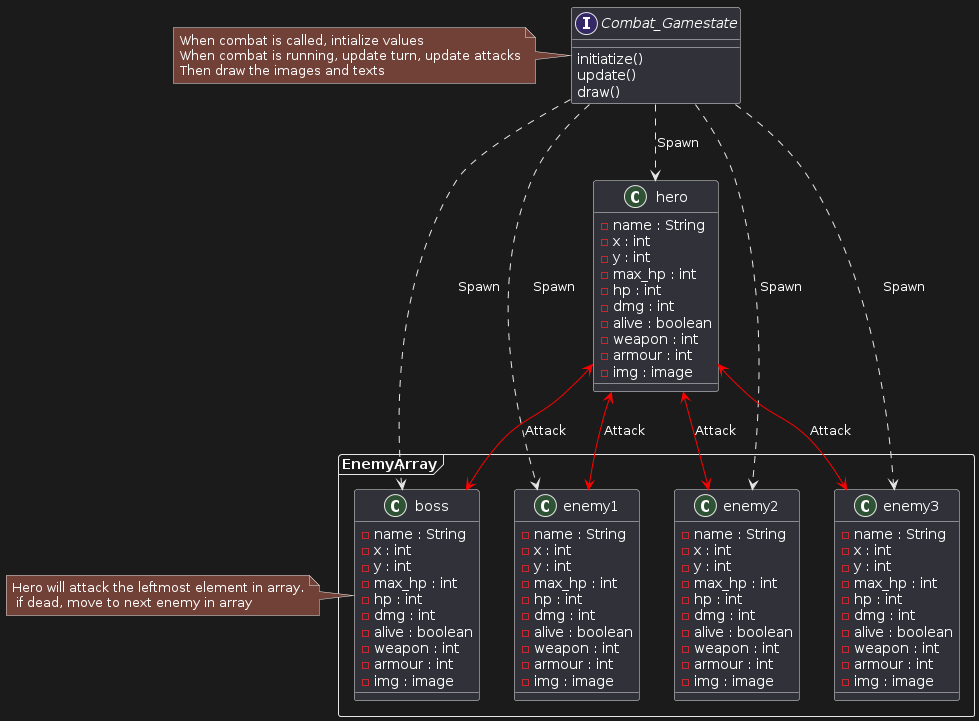
Code for Player Movement



User interface of the generated dungeon

## 3) Combat System Simulation

The combat system is a completely separate system that is added to fulfill the combat and strategic gameplay elements to the project. As the game is inspired by ‘Dungeons and Dragons’, their respective prevalence themes of military, weaponry, and fighting as supported by Trammell Aaron (2018) would also carry fourth in this system. Since the combat system was developed alongside the procedural generation, the combat system simulation needed to be modular



Initial UML Class Diagram for the combat system

### 3a) Ai Combat Simulation

Firstly, a class was created that will be shared with all entities in the game. It displays the position of they character on the x and y axis, the name of the character, the maximum hp (which is also the initial hp) alongside a hp tracker with a variable named ‘hp’, and finally the potential damage the entity is capable of alongside it’s status on whether it’s alive or dead.

class entity():

def \_\_init\_\_(self, x, y, name, max\_hp, dmg):

self.name = name

self.x = x

self.y = y

self.name = name

self.max\_hp = max\_hp

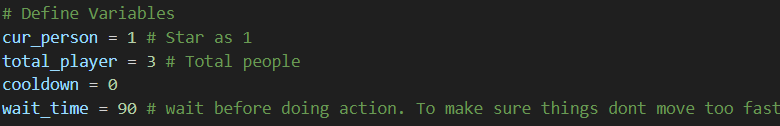
self.hp = max\_hp

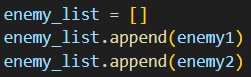
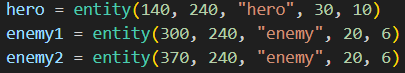
self.dmg = dmg

self.alive = True

**First Prototype** of the code of the entity class.

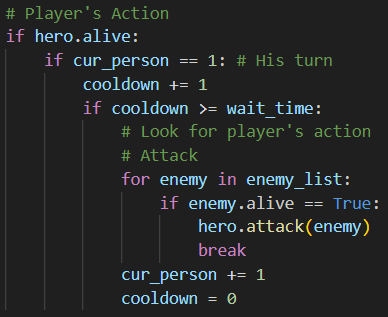
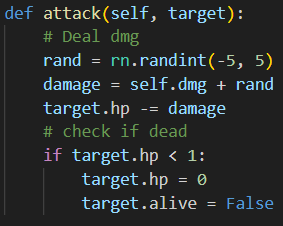
To determine which individual’s turn it is, we have a cur\_person variable and a total\_player variable to know when to reset the cur\_person variable. Furthermore there is a cooldown and a wait\_time to give some time before every action taken. If this is not implemented, each individual’s actions will happen simultaneously. As an example, here is initialization of the player, prompt named ‘hero, and the enemies. When the game is running, the enemies would be put into a list so that each enemy can be checked without manually typing ‘enemy1’ and ‘enemy2’

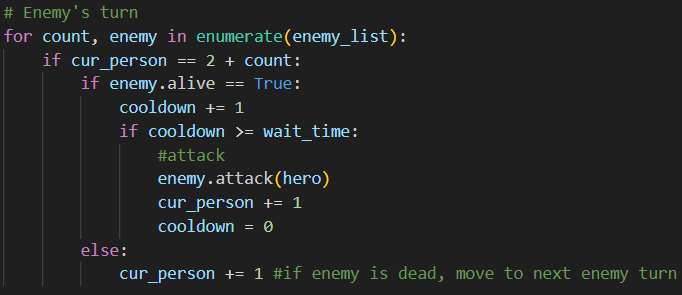


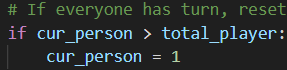


Code of the defined variable (Pre-integration Module)

While the game is running, the code check’s each individual’s turn. If the player is alive, it’s his turn, and if the player waits for 90 frames/ticks, takes the first enemy in enemy\_list and attacks the enemy, the damage will be its class’s damage followed by a random number from -5 to 5. If the target has a hp that is less than 0, they are dead hence the ‘alive’ variable will be false. During the enemies turn, they will do the same action with the same logic. The enumerate() function was used to not only get a counter (labeled as ‘count’) but also get each element in the array (labeled ‘enemy’) to attack.

****

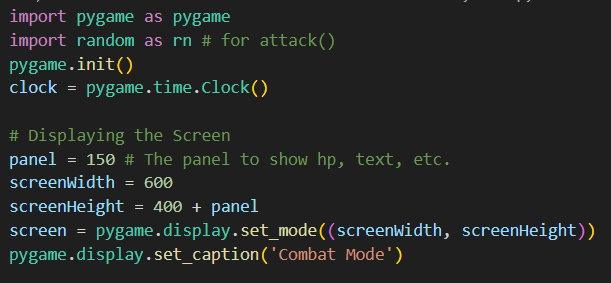




Code of the turn-based system (Pre-integration Module)

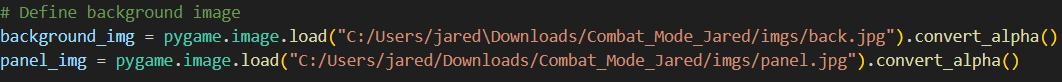
### 3b) Defining Graphics & Environment

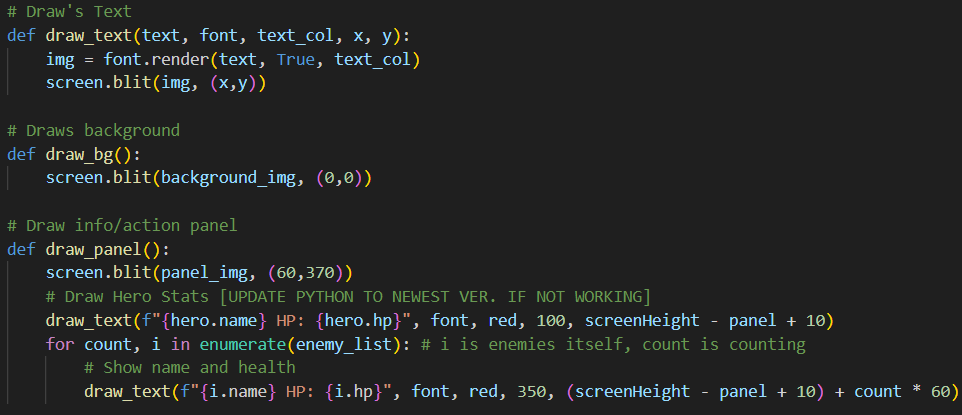
In order to initialize the game. These essential variables are needed. A clock is needed to count the time, the screen width and screen height needs to be defined as it would be the window in which the game will be displayed, and an extra space for a ‘panel’ which would be the user interface to display health. The ‘screen’ variable sets the size of the window and the caption ‘Combat Mode’ defines the name of the window



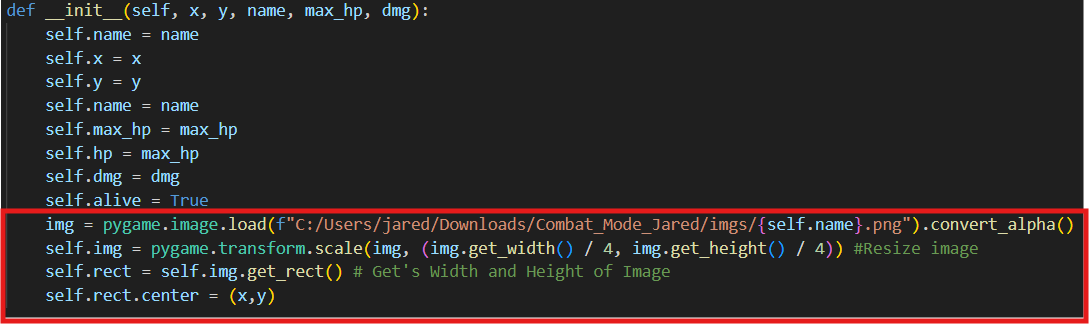
Code of the graphical initialization (Pre-integration Module)

These functions will be necessary during the initialization phase in order to display certain assets onto the screen. The draw\_bg() take’s an image of the background and displays it on the screen using the blit() function. When it comes to displaying text, **Pygame has some limitations** as text needs to be in an image format before displaying onto the screen hence the need to render() the text in the draw\_text() function. Finally, to draw the user interface/ panel, it will draw its background, followed by the HP (which represents health) of the hero (which is what the player is called), and a list of the enemies’ hp.



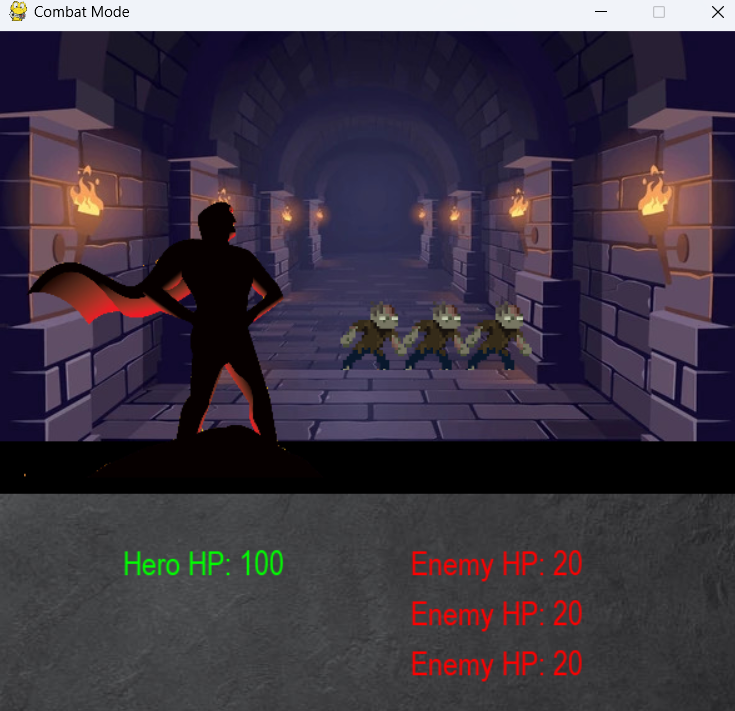
  
Code for displaying graphics (Pre-integration Module)

In order to load the images for the player and the enemies, in the same entity class, there is an additional amount of generalized code in which pastes a resized image onto a rectangle and then displays it in the game. The reason the image is pasted on a rectangle is so that the x and y coordinates of the image can be easily manipulated.



Code for image for each individual entity (Pre-integration Module)

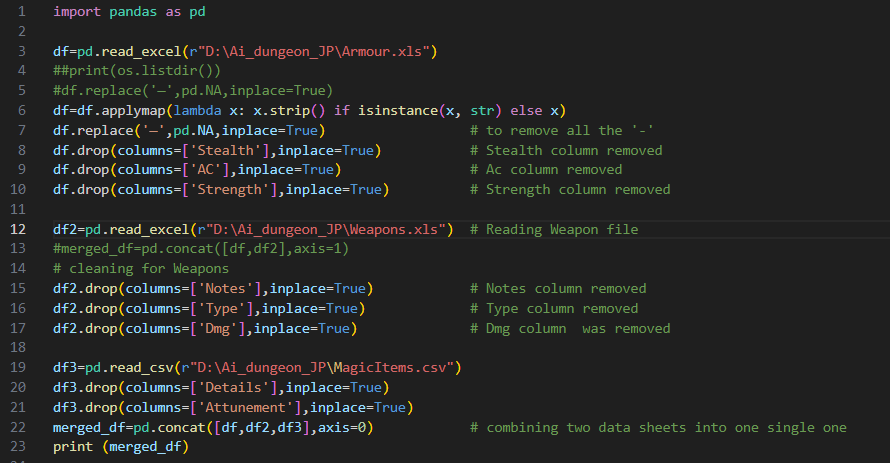
The end result is a functional interface in which displays the changes of each entity’s hit points during battle. The player or ‘hero’ is displayed on the left with its accompanying hp. He/she is displayed slightly larger in order to create the effect of the character being closer towards the camera. Furthermore a silhouette was chosen in order to allow the user to imagine and role-play their own character, similar to ‘D&D’ as supported by Brewer Nathan (2016). The enemies themselves are displayed to the right of the dungeon’s hallway with their accompanying hp. The game can accommodate a maximum of 3 enemies at a time for balancing purposes/reasons with their respective pre-set displayed locations. However, if it happens to be a boss scenario, there will be a change in the location in which the enemy will be printed. Note that a ‘boss enemy’ is only counted by the turn counter as a singular entity. To pay homage to ‘Dungeons and Dragons’, the final boss is presented as a ‘Beholder’ which is ‘Wizard Of The Coast’, The game’s parent company’s intellectual property and a staple mascot to the tabletop roleplaying game.

****

Combat system (Pre-integration Module)

## 3) Data Analysis

In order to determine the right balance of difficulty during the gameplay look to ensure maximum enjoyability and player retention, reviewing data would be an important solution. ‘Dungeon Crawlers’ tend to have combat and movement synergies with weaponry, and other categories of equipment. For the Data analysis section of the project, there were 4 initial csv files of weaponry, regular items, magical items, and armor originally from ‘dungeons and Dragons’ specifically from the Firth Edition of the ‘Player’s handbook’. From these data, there were some missing values and poorly formatted variables. This was a golden opportunity to incorporate data cleaning into the project.



Data Cleaning (In Development)

# Evaluation: Retrospection

This project turned out to be a fantastic educational opportunity for every member of the team. The crucial components of replayability, procedural dungeon generation, was successfully accomplished, and its connection with the combat system and data structures were achieved. Both systems' modular designs provide easy maintenance and potential expansion. It was really interesting to experiment with the procedural generation options and see how it influenced player tactics and emergent occurrences. The project did, however, also emphasize the significance of striking a balance between complexity and maintainability in code, as well as the necessity of extensive testing and well selected data structures. The development process was greatly aided by the comprehensive documentation. All things considered, the project is proof of Pygame's ability to simulate intricate systems and produce entertaining games that offer insightful lessons for further game development endeavors.

## Limitations and Moving Forward

The data analysis aspect of the project is lacking. Furthermore, due to time constraints, player performance analysis was not completed. For programmers planning to continue to the project// prospects for the future of this genre of games, a player performance analysis and a database of weapons, armor, equipment, and monsters would be a step in the right direction

# References

Brewer Nathan (2016), ‘Going Rogue: A Brief History of the Computerized Dungeon Crawl’, *InSight IEEE USA,* <https://insight.ieeeusa.org/articles/going-rogue-a-brief-history-of-the-computerized-dungeon-crawl/>

Trammell Aaron (2018), ‘Militarism and Masculinity in Dungeons & Dragons’, Masculinities in Play, pp 129-147, <https://link.springer.com/chapter/10.1007/978-3-319-90581-5_8>

### Image Sources

* <https://www.enworld.org/threads/gaming-with-gygax.204729/>
* <https://www.cdw.com/content/cdw/en/research-hub.html>
* <https://www.shutterstock.com/image-vector/dark-dungeon-long-medieval-castle-corridor-2118070751?irclickid=WBGzch0C8xyKTphylxV-gzUBUkHTuZTH3VKrxc0&irgwc=1&pl=77643-108110&utm_campaign=TinEye&utm_content=108110&utm_medium=Affiliate&utm_source=77643&utm_term=>

Copyright Disclaimer Under Section 107 of the Copyright Act 1976, allowance is made for "fair use" for purposes such as criticism, comment, news reporting, teaching, scholarship, and research. Fair use is a use permitted by copyright statute that might otherwise be infringing. Non-profit, educational or personal use tips the balance in favor of fair use. ©