OCR GCE A

COMPUTER SCIENCE PROJECT

H446-03

Name: Riley Stevenson

Candidate Number: 7509

Tuxford Academy: 28148

Title of Project: Dunzle

H446-03 – Project CONTENTS

Table of Contents

[A. Analysis 8](#_Toc164169866)

[Problem identification 8](#_Toc164169867)

[Stakeholders 8](#_Toc164169868)

[Research 8](#_Toc164169869)

[Essential Features 14](#_Toc164169870)

[Limitations 15](#_Toc164169871)

[Solution Requirements 15](#_Toc164169872)

[B. Design 16](#_Toc164169873)

[Decompose Problem 16](#_Toc164169874)

[User interface 16](#_Toc164169875)

[Systems diagram 24](#_Toc164169876)

[C. Developing the coded solution 28](#_Toc164169877)

[Developing the overlay class 28](#_Toc164169878)

[Structure 29](#_Toc164169879)

[Developing the overlay manager. 30](#_Toc164169880)

[Structure 31](#_Toc164169881)

[Developing the element class 33](#_Toc164169882)

[Structure 33](#_Toc164169883)

[Developing the element list class 36](#_Toc164169884)

[Structure 37](#_Toc164169885)

[Developing the element dictionary class 37](#_Toc164169886)

[Structure 37](#_Toc164169887)

[Developing the text class 37](#_Toc164169888)

[Structure 38](#_Toc164169889)

[Developing the still image class 39](#_Toc164169890)

[Structure 39](#_Toc164169891)

[Developing the rectangle class 39](#_Toc164169892)

[Structure 39](#_Toc164169893)

[Developing the Circle class 40](#_Toc164169894)

[Structure 40](#_Toc164169895)

[Developing the line class 40](#_Toc164169896)

[Structure 41](#_Toc164169897)

[Developing the group class 41](#_Toc164169898)

[Structure 41](#_Toc164169899)

[Developing the dungeon definitions 42](#_Toc164169900)

[Load Models 43](#_Toc164169901)

[Developing the dungeon 43](#_Toc164169902)

[Generate Tiles 43](#_Toc164169903)

[Generate Image Layout 44](#_Toc164169904)

[Get Layers 44](#_Toc164169905)

[Structure 45](#_Toc164169906)

[Developing the Entity class 46](#_Toc164169907)

[Developing the AI class 46](#_Toc164169908)

[Errors of element implementation 48](#_Toc164169909)

[Copy Method 48](#_Toc164169910)

[Errors of element list implementation 48](#_Toc164169911)

[BEFORE IDEA 48](#_Toc164169912)

[AFTER IDEA 49](#_Toc164169913)

[Errors of overlay managers implementation 49](#_Toc164169914)

[Main menu background errors 49](#_Toc164169915)

[Index Error 50](#_Toc164169916)

[Highlight Error 50](#_Toc164169917)

[Evidence of element implementation 51](#_Toc164169918)

[Constructor Method Implementation 51](#_Toc164169919)

[Property Implementations 51](#_Toc164169920)

[Update Hitbox Method 52](#_Toc164169921)

[Empty Draw Method 52](#_Toc164169922)

[Image Manipulation Methods 53](#_Toc164169923)

[Align Methods Implementation 53](#_Toc164169924)

[Update Methods 54](#_Toc164169925)

[Element Drawing Order Manipulation 54](#_Toc164169926)

[Copy Method 55](#_Toc164169927)

[Evidence of element list implementation 55](#_Toc164169928)

[Element List 55](#_Toc164169929)

[Evidence of element dictionary implementation 55](#_Toc164169930)

[Element Dictionary 55](#_Toc164169931)

[Evidence of overlay Implementation 56](#_Toc164169932)

[Constructor Method Implementation 56](#_Toc164169933)

[Property Implementations 56](#_Toc164169934)

[Property Setter Implementations 57](#_Toc164169935)

[Update Hitbox Method Implementation 58](#_Toc164169936)

[Align Methods Implementation 58](#_Toc164169937)

[Element Dictionary/List Manipulation 58](#_Toc164169938)

[Element Collision 59](#_Toc164169939)

[Update Methods 59](#_Toc164169940)

[Draw Method 60](#_Toc164169941)

[Evidence of overlay managers implementation 60](#_Toc164169942)

[Constructor Method Implementation 60](#_Toc164169943)

[Properties 61](#_Toc164169944)

[Getter and Setter for events states 62](#_Toc164169945)

[Append, Remove and Insert function for the overlays 62](#_Toc164169946)

[Move overlays in list 63](#_Toc164169947)

[Overlay at pos? 63](#_Toc164169948)

[Template Update Methods 64](#_Toc164169949)

[Main Update Method 65](#_Toc164169950)

[Evidence of text element implementation 65](#_Toc164169951)

[Constructor Method Implementation 65](#_Toc164169952)

[Renderer 66](#_Toc164169953)

[Properties 66](#_Toc164169954)

[Update Text 67](#_Toc164169955)

[Draw 67](#_Toc164169956)

[Evidence of still image implementation 67](#_Toc164169957)

[Evidence of rectangle implementation 67](#_Toc164169958)

[Constructor Method Implementation 67](#_Toc164169959)

[Draw Method 67](#_Toc164169960)

[Evidence of circle implementation 68](#_Toc164169961)

[Constructor method implementation 68](#_Toc164169962)

[Update Method 68](#_Toc164169963)

[Draw Method 68](#_Toc164169964)

[Evidence of Line implementation 68](#_Toc164169965)

[Constructor Method Implementation 69](#_Toc164169966)

[Update Hitbox 69](#_Toc164169967)

[Properties 69](#_Toc164169968)

[Property Setters 69](#_Toc164169969)

[Draw Method 70](#_Toc164169970)

[Evidence of Group implementation 70](#_Toc164169971)

[Constructor Method Implementation 70](#_Toc164169972)

[Properties 71](#_Toc164169973)

[Append Element 72](#_Toc164169974)

[Remove Element 72](#_Toc164169975)

[Remove element by name 73](#_Toc164169976)

[Is element at position 73](#_Toc164169977)

[Get element at position 73](#_Toc164169978)

[Get elements at position 73](#_Toc164169979)

[Evidence of Dungeon Implementation 73](#_Toc164169980)

[Constructor Method Implementation 74](#_Toc164169981)

[Unlock Room 74](#_Toc164169982)

[Generate Tiles 74](#_Toc164169983)

[Generate Image Layout 78](#_Toc164169984)

[Evidence of Dungeon Definitions 79](#_Toc164169985)

[Grid Square Sizes 80](#_Toc164169986)

[Load Models Function 80](#_Toc164169987)

[Screen Size + Fog Of War (Related-ish) 80](#_Toc164169988)

[Tile Information 80](#_Toc164169989)

[Rooms 81](#_Toc164169990)

[Evidence of entity implementation 86](#_Toc164169991)

[Construction methods implementation 86](#_Toc164169992)

[Evidence of AI implementation 87](#_Toc164169993)

[Construction Method Implementation 87](#_Toc164169994)

[Room Property 88](#_Toc164169995)

[Movement 88](#_Toc164169996)

[Targets 88](#_Toc164169997)

[Distance to player 89](#_Toc164169998)

[Update Method 90](#_Toc164169999)

[Evidence of shadow implementation 90](#_Toc164170000)

[D. Evaluation 90](#_Toc164170001)

# A. Analysis

## Problem identification

The project I wish to complete is a 2.5D (Visually is 3D but in terms of mechanics is 2D) Top-Down puzzle solving game, where the player can explore a series of dungeons (caves) that will contain puzzles to access rooms, monsters that defends those rooms as well as an incentive to dive into these dungeons. The “loot” that can be earnt in these dungeons could be weapons, armour, valuables, etc. The loot would be generated based on the “level” of the dungeon, the type of puzzles/enemies inside, and the player. The dungeons would also be generated. However, they would be generated procedurally with pre-made art/tiles, meaning it will be rare that two consecutive dungeons share the same structure. Although, if I have extra time, I will try to implement procedurally generated art.

## Stakeholders

The stakeholders for my project would be any gamers that may enjoy a puzzle/fighting game that will keep them entertained, for the stakeholders my game has to be enjoyable, and repeatable (They will want to come back and play it again after playing it), attractive (looks good so they will want to play it in the first place).

The game has to, by the bare minimum, have a good game loop meaning the player can keep playing and the gameplay not cut off abruptly. The stakeholders will consist of casual gamers, since the game won’t be too intense or laid back it lands in the middle of competitive and relaxing games.

## Research

To better understand who my stakeholders are and what they would want in a game, I created a form that was distributed across some classes in my school and also across some others that I believe would have insight into what a game should have and what should have the main focus.

The form consisted of questions like:

* How long have you been playing games? (I prioritise the responses with the higher number of hours and lowest number of hours, this is so I can see what a newer player expects and what the experienced players need)
* What genres of games do you play and why? (So, I can see what features/ genres hold players interest)
* What platforms do you play on?
* Rate features in order of importance? (So, I can see what features I should focus on the most during development)
* What is your gender? (This was didn’t really hold much value for the project, I was just intrigued to see how much difference there is in how people are brought up and how exposed to games they are based on their gender, there was limited data on this though due to the majority of IT students being male[89%] )
* What other features would you like? (This was so I could gather more feature ideas that I didn’t already think of)

At the time of writing this I have 28 responses to these questions.

A picture containing screenshot, diagram, text

Description automatically generated

With this first question, I asked the age of the person doing the form. This is so I can compare the different answers of different age groups and see how they differ and what group likes what features.

For question 2 and 3, I asked the email and name of the person so I could potentially contact them in the future for more insight, I will not be showing photos of these questions for privacy purposes.

A picture containing text, screenshot, font, diagram

Description automatically generated

For question 4, I asked the persons gender, as you can see there seems to be an overwhelming majority of people who identify as male in the IT classes who completed the form.

A screenshot of a graph

Description automatically generated with low confidence

For question 5, I asked how often the person plays games on a daily basis, (other was for if someone wanted to be more specific). Here it seems that the majority of people player between 1-3 hours, so more casual gaming. This data lines up with the stakeholders I am looking for, people who play casually, so I can keep the people who played 1-3 hours in mind for future testing.

A picture containing text, screenshot, diagram, colorfulness

Description automatically generated

Question 6, I asked what platform people play on, the majority of people answered PC. This means in the future when I am developing the code, I can focus on making it work best on PC, but still add support for controllers since a decent amount of people play on console.

A picture containing screenshot, text, diagram, colorfulness

Description automatically generated

Question 7, I asked how many years the individual has been playing games, this was so I could support experienced players and non-experienced players. By doing this, I can consider what a new player wants and experienced players need.

Skipped over question 8 since it’s not related to the topic.

A picture containing text, screenshot, colorfulness, diagram

Description automatically generated

For question 9, I asked what genres people played. From the group of individuals, they seem to have favoured Action more over anything. Since my idea is of a stealth/puzzle mix it seems it may be at a disadvantage within this group, so to make sure it is still favoured I will make sure that there is a healthy amount of action. Not enough to change the core idea of the game but enough that the game is appealing to a wider audience.

A screenshot of a computer

Description automatically generated with low confidence

For Question 10, I asked a rather simple question, I didn’t really mind all too much about the answers, and since it was simple it led to mostly simple answers such as the most common answer being “fun”. However, the individuals that did answer with more detail have saved me some research since they have provided features and what genre they relate to. With this knowledge I can sway the genre of my game by introducing some of the features these individuals have pointed out to be enjoyable.

Skipping 11 since it is similar to 10 and didn’t get too much useable content.

A picture containing text, screenshot, font, line

Description automatically generated

For question 12, I asked the user to rate a selection of features in order of importance to the players experience. This allows me to prioritize during development in order to get what the player wants done sooner with more detail. It seems that the majority believe that the story in a game is most important to the user experience, however because I am making the game, I think I might only make an outline of the story and if I have time at the end, I will fluff out the story. I was a bit surprised to see character customization above lighting but it seems that choosing the characters appearance is important for some people.

A screenshot of a computer

Description automatically generated with medium confidence

Question 13 is similar to 10, it wasn’t too much of an important question, only there to potentially save some time researching in the future. A common theme with this question is the detail of the environment, despite my game idea being 2.5D (looks 3D but mechanics are 2D) I do want to add more detail but that will be second to getting it to work.

A picture containing text, font, line, screenshot

Description automatically generated

For Question 14, I did a similar thing to 12 and asked users to choose what they prefer and why (why is Question 15), most people seem to like Pre-made worlds for the reason that more detail can be put into them, in 2nd was procedural worlds where the environment around the player is never the same across saves and generates procedurally. Some people still liked user designed content, such as levels in super Mario maker, but it isn’t voted as good as the other 2. For my game I’m thinking of making procedurally generated levels. However, I may have some puzzles created by hand as a sort of tutorial, so its fairer for new players.

The last question (No.16) was just asking if anyone had anything else to add. This question only brought 1 valuable answer and that was to potentially add a secret ending, or an achievement like system. This will encourage players to keep playing the game and give them goals to achieve as sometimes, games without goals can get repetitive and lose interest.

Games like my project idea already exist, such as “Hogwarts Legacy” which contains trials for the player. These trials are similar to my idea in the fact that they are a sort of dungeon that contains enemies and puzzles which rewards the player, it is also similar to “Dungeon and Puzzles”(2021) in which the character goes through dungeons and has to complete puzzles to advance further, there is also the game “Binding Of Isaac” which is the most like my game idea.

**Overview:**

In Hogwarts Legacy ,the user can go through trials to progress in the game, there are 4 trials, one for each “Keeper” in the game. These trials consist of some small puzzles and fighting enemies. From my knowledge there is a boss at the end of every trial, these bosses usually have a lot of health and have attacks that forces the player to either dodge or perform some sort of mechanic to avoid damage. These boss battles are usually quite difficult and don’t offer too much reward but progress in the main quest line.

The player experience while going through these trials is usually to prepare beforehand by gathering health potions, so they can regain health while in battle, and slowly but smoothly progress through the trial, completing puzzles and killing enemies gaining various items throughout their journey, this experience continues up and until the boss battle at the end where it is much more difficult, if the player dies during the battle they respawn just before the battle so they can try again. The player will continually attempt to kill the boss before finally leaving the trial victorious.

**Relevance To My Project:**

In my project, I plan to avoid some of these mechanics but adopt others. I like the aspect of while progressing through the trial the player can complete side-puzzles to gain extra items or experience points towards a quest or skill, all the while killing enemies and unlocking more rooms. I, however, do not like the aspect of only being able to leave by either aborting the quest or completing it fully. I would like to have a system so that the player can leave the trial/dungeon at the end of any room, however leaving will have a drawback on final rewards and quests. I would also like to give the player the ability to abandon the boss fight and leave the dungeon if it is too hard.

Overall, Hogwarts Legacy is a good game and has some great features that I wish to include in my project, but it also contains some features that I would like to avoid since they increase difficulty for not too much reward.

**Overview:**

In Dungeons and Puzzles, on each level there is a puzzle that the user has to complete that allows access to the next level, the player can challenge themselves to make the least amount of moves by thinking logically about the puzzle. The user can use a sword to attack monsters, or a bow and an arrow to destroy enemies from afar. They can also push obstacles back with a shield and pull the monsters with a pair of gloves that can be picked up if on a level.

The dungeon has 150 hand-crafted levels to complete, that the player can go through and attempt to complete and optimise their approach. It also has nice top-down pixel art that give the illusion of a 3D environment with mechanics being 2D in nature. For my project I wish to replicate this type of art and have similar mechanics.

**Relevance To My Project:**

In my project, I wish to have a similar approach to this, where the player can progress through levels in a dungeon completing puzzles and killing enemies, but I want it so the player can move freely throughout the level and have it so movement doesn’t use a tile system, but the environment will use a tile-based system for generation and collision.

A picture containing text, animation, screenshot, fiction

Description automatically generated **Overview:**

The Binding Of Isaac is a roguelike game inspired by the biblical story of Isaac, it is a 2D top-down dungeon crawler game.

The player controls Isaac, or one of seven other unlockable characters, throughout a procedurally generated dungeon in his mother’s basement. The combat system is real-time meaning that the player and enemy don’t have to wait to attack and can do so at any point. The player can move throughout the dungeon collecting items, power-ups and fighting enemies. The goal is to defeat all of the monster bosses and eventually Isaac’s mother.

When playing the game if the player dies, then it is a permanent death, meaning they will lose that playthrough and won’t be able to play as that character (No respawning).

**Relevance To My Project:**

This game has a lot of features that I plan to have in my project such as levels that the player will progress through, and in “The Binding of Isaac” the rooms once cleared of enemies, will stay cleared and the player can go backwards through the dungeon. In my project I want to make it so that once a player completes a room and leaves it the puzzle will either reset or monsters reform, but will deactivate, meaning that the player can walk past and go backwards and won’t have to worry about refighting them again, but they will still be there, this could be used in the story later.

In my project, I also want to have multiple dungeons that the player can go through and complete or fail, that may change in some way once the player finishes them, such as reforming, a new room appearing, a room disappearing, a theme shifts inside of the dungeon, new enemies etc. This will make the player want to redo completed dungeons in the future.

## Essential Features

For my game I have some features in mind that need to be in the project in order for me to count it as a success.

My game needs to have a main menu screen through some sort of overlay feature, this wraps back around to the stakeholders because in order for them to enjoy the game and/or want to keep coming back, there needs to be a nice gentle entry into the game, with that being the main menu.

This overlay feature needed to be able to be used across the entire game to provide UI for the player and for starting the game. This feature is incredibly important because the games I researched all have some form of UI to give the player information and let them manipulate the world around them. For example, Hogwarts Legacy, has UI that shows the player, the current spells they have in their “spell wheel”, the spells current state (can it be cast or is it on delay), and more. Without this info, the game may be harder to play and wont provide a better experience for the stakeholders. **However**, some games can use a lack of UI to make the game more immersive, they usually give you controls at the beginning and then give information about your players state in different ways, like health being shown by how red the screen is, or stamina by your players breathing. Games without UI are usually in my experience, more horror/thriller focused, however that’s not really the feeling I’m going for, so I’m going to use UI.

The game also needs to have dungeon generation, this feature is needed in order to generate the rooms the player will be able to traverse, I could use prebuilt dungeon levels, however, not only is it time consuming to build a lot of unique dungeon levels, you would also need to consistently update the game to add more levels which reduces the amount of time I could use to focus on other components of the game. Dungeon procedural generation, adds in a sense of randomness to the game, but also in the end, saves me time. In my research, I found that Hogwarts legacy uses more prebuilt then randomised, this is because they want the map to follow the structure of Harry Potters world and randomness would take that away, but also because it is heavily story based, and it is much harder to make sure a story players out smoothly in a randomised world.However, in my game, I don’t have a story or a map im taking inspiration from, so random generation is my best choice. So in the end, my game will share more features with The Binding Of Isaac, which does use procedural generation to make its levels and therefore provides a unique experience every time.

The game will need its enemies for the player to fight in order to provide an engaging experience. This feature point is important because each game I researched has some form of enemy or goal to achieve, fighting enemies is important to engage the stockholders as it provides a thrilling experience that draws them in to replay the game.

I also want it to have good lighting whether through ray-casting or some other method, this light will be able to cast shadows which could be used to sneak around as an extra mechanic…

## Limitations

During the design and implementation of my project there will be limitations I may encounter.

For my project, I have chosen python, and while python is a good language because of its readability and easy to use, it does come with limitations that may affect the performance of my project.

Python is known to be relatively slow when compared to other languages. I will have to work around this by making my project as streamlined as possible.

It can also be quite heavy in memory, so I will have to try limit how much memory it uses, I could approach this by, for example, when you go into another room it will have a short loading screen where it removes the previous room from memory and writes it to storage, it can then load the next room and have the loading screen end.

The loading screen could also be an animation of their character walking down a corridor or something similar so it looks less like a loading screen and more like a nice cutscene, however I will put a symbol, so the player won’t close the game while its doing this. I could use multiple threads to achieve this, one to update the screen and some others to load the models, generate the dungeon and another to generate the lighting.

## Solution Requirements

For my final solution, I want to have a game that can be played multiple times, has multiple save files so that player can create new saves, delete/overwrite old saves, and load saves. The game must be enjoyable, look good, and be fully functional. I want it to have an item collection system with gear and weapons, that can be collected and potentially upgraded. I want there to be a sense of progression which can be shown by collected items/weapons but also potentially have achievements.

I want at minimum, there to be a generation system for the dungeons that can be fully explored. Having a world generation system outside of dungeons is optional and low in the list of things I need done, it may be a feature added after the project is done.

Each dungeon must have multiple rooms where the player can fight and solve puzzles; the rooms must also change to reflect the player and slowly ramp up in difficulty if it’s too easy for the player.

The game must have user friendly menus and a title screen that shows off the users past achievements and gives new players info/help on the game such as mechanics, controls and maybe access to a wiki.

# B. Design

## Decompose Problem

My programs design can be broken down into smaller categories, such as:

**User Interface:**

Such as the games Main Menu, Health, Inventory, Stamina etc.

The user interface is highly important because it gives key information to the player.

Without the User interface, the gameplay would be horrible, and it would be near impossible to

navigate the game with some sort of ease.

**Dungeon Generation:**

The games dungeon generation is highly important because it decides the difficulty of the game,

as well as how “nice” the game feels to play.

If the dungeon is generated in a line, then the game would feel repetitive with a limited sense of exploration, however with good dungeon generation, the player will have choices of where to explore.

**Game Mechanics:**

The game needs to have good mechanics, meaning enemies have to be hard to push the player,

but not too hard that the player will quit.

The loot must be good but not too good etc.

There also needs to be a good feel of progression for the player, I need to make sure that the

game feels like they are consistently progressing, with only a few roadblocks here and there, that the player can work around with time and effort.

## User interface

**Title Screen**



For the title screen of my game, I want to have two scrolling columns, one on each side of the screen, that show the players best dungeon runs and achievements throughout their different playthroughs.

I also want to have, in the middle of the screen, the games name which hasn’t been decided yet, which is followed by 5 buttons, which should be New Run, Load Run, Settings, Help and Quit. These buttons are rather self-explanatory, “New Run” will be used to make a new dungeon run save, I want to have 5 save slots, this forces users to either play all the way through or overwrite old saves. But I do want to have a sort of ‘archive’ features, where you can save a game save outside of the main menu, so it doesn’t take up one of your slots, and can later be imported back into a slot.

The ”Help” button is there for new players, I will make a sort of wiki on how to play the game, base controls, and functionality.

This menu is the “title screen” and will appear when you open the game, its use is to give the user options and info, but to also gradually lower them into the game, instead of throwing the user into gameplay as soon as they open the program.

This links to the solution requirements because it is user friendly, due to it being similar to other games but also the difference in features that the user can interact with and the non-functional elements.

**New Run Screen**

A screenshot of a game

Description automatically generated with medium confidence

On this screen the player will be able to choose one of the five save slots to create a new game, and if there is already a game saved in the slot you choose you will get a prompt asking if you want to overwrite the save file.

This links to the solution requirements, because in order for my project to be considered a success I want the user to be able to save progress.

The load run button will be pretty much the same screen; however, the title will be “Load Run” and won’t open the game creation menu when you click on a save slot and will instead open it.

A screenshot of a computer

Description automatically generated with medium confidence

For my settings menu I want three sections, the section currently selected will be highlighted so that the user can easily tell which one they have currently selected. A lot of different types of software use this type of feature to help the user, and its use is easily picked up on, because of this, no explanation is needed for the user for them to understand how the menu system works.

The first one for controls will have three columns, one for the action names, one for the button that activates it and then a reset key for each to reset the actions button to default.

Each action should be categorized by their effect, for example the first 5 rows are separated from the others because they are movement related.

I also want to have the scroll bar on the side so the player can scroll down to see the rest of the settings.

When I am designing the assets, it may look different to the current setup as well as the other designs shown here, because I am not entirely sure on this design at the moment

A screenshot of a video game

Description automatically generated with low confidence

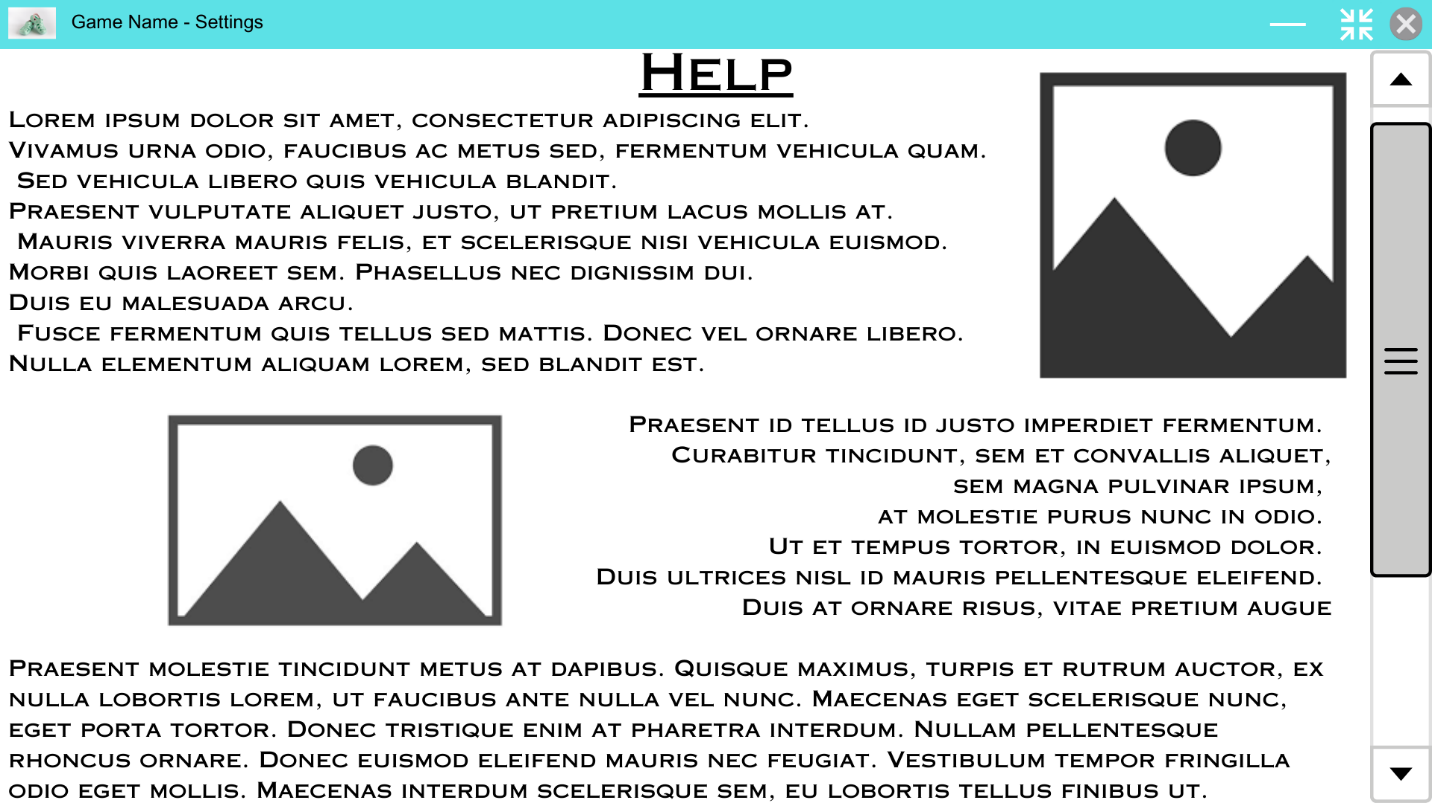
In the second section, for graphics, I want there to be a list of options with various methods to change them, such as a slider, a drop-down list, checkboxes and maybe more. These options will be focused on the graphics of the game such as limiting fps so the player can limit the CPU usage of the program.

The different types of options are used in many other programs and are intuitive to use, such as the drop-down menu. The arrow on the drop-down boxes helps make the user aware that it is interactable and by hovering over it, a menu appears below the box showing different options. A drop-down menu is the best choice for settings such as “Resolution” because it is compact and allows the user to select one option out of a list that takes up the same amount of space no matter how many options are in the list.



The Audio section will contain a list of the various types of sounds that can play in the game along with sliders to change the volume of that type, setting the slider to its lowest will mute that type of sound entirely. As the entries within the list are within a certain range of the bottom of the screen, they will gradually get more transparent. There is a scroll bar on the side so the user can access any of the other types of sounds.

Scrolls bar are found in a lot of programs and so are easily recognizable, and user friendly. This means that no previous knowledge of the game will be needed to navigate the menus.



For the most part, the help menu will consist of text, with a variety of images scattered throughout to help guide the user. However, at the bottom of the page, I wish to include a link which opens a wiki that delves into more detail of the game and has a better guide on it.

I have used Lorem Ipsum as placeholder text for the help page since I have not fully settled on what kind of help will be on this page.

The help menu won’t have any interactive elements except for a back button and the wikis link, this is because it is meant to act as a small guide and not as something which affects the users playthrough of the game.

Throughout the help menu there will be many titles which show off the different parts of the small guide, the user will be able to select these different sections through a separate menu that could slide from the side of the screen into view.

A screenshot of a phone

Description automatically generated

This is an example of what that side menu could look like when extended.

It would include a list of all the titles on the help page, each of which acts as a link to that part of the document.

**In all of the settings sections, I have made the buttons have a large appearance and have a high contrast with the background so that the experience is easier for most users and consequently, there is less white space left over, which if there was more of could make the menu more boring.**

**Game Screen**

The game screen will be where the actual game content will be drawn, such as the player, the environment, enemies etc.

The other screens shown so far will be overlays that get drawn on top of the game screen if they are open.

When the program is run, a background image will be drawn onto the game screen since a game hasn’t been loaded yet, this could look like:

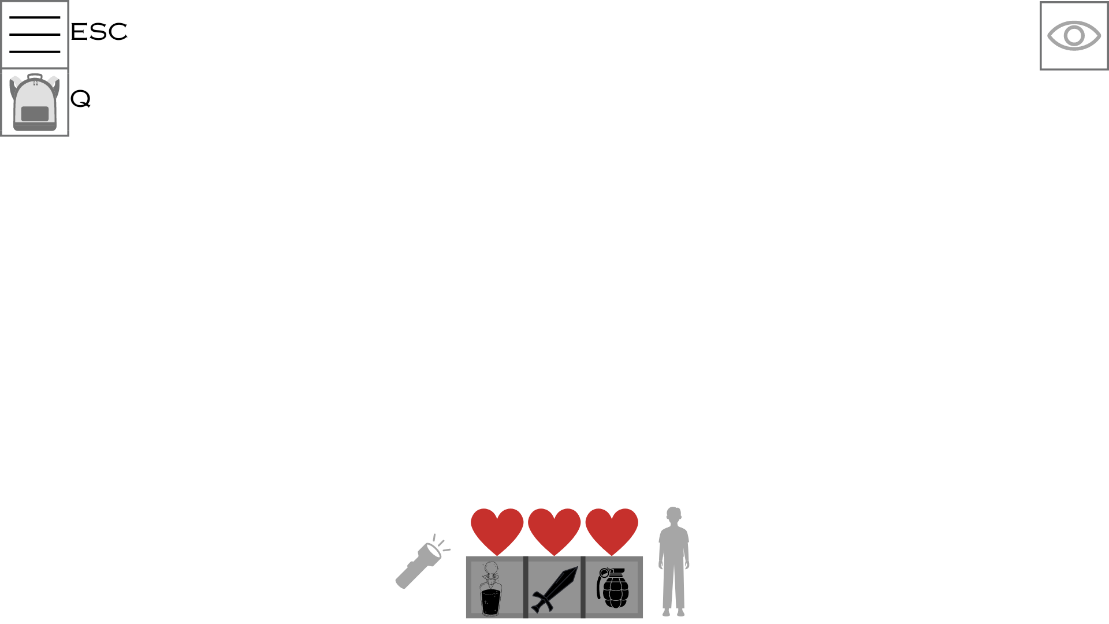
A screenshot of a game

Description automatically generated

This system will be helpful since it will make other overlays later in the game such as a pause menu easier to implement. If I make the overlays work on an element-based system then I could develop different types of elements such as a button, that would allow me to not only design these overlays easier, it will make the system easier to develop in the future, by implementing different elements.

When the player has loaded a game whether it be a new save or an existing save, any open overlays will be closed and the program will start the game and draw the environment and everything else needed onto the game screen.

Any in-game overlays will then be drawn on top, this will include the HUD (Heads-up display), the HUD will show information such as health as well as what items the player has and any other information that is necessary.



This is what the HUD overlay could look like when its implemented.

Overlays accessible by buttons.

At the top left of the overlay, there are some symbols. The three lines is in most games recognized as a pause menu, this will contain different options that the user can use to change their gameplay, such as changing the max fps, graphics etc. Pressing the shortcut related to the button will open the pause menu overlay and close the HUD overlay.

Underneath the pause menu symbol there is a backpack symbol, when the keyboard button is pressed, the game will open the users’ inventory and display it on screen in boxes like the hot bar (bottom middle). The inventory will have a max size of 9 slots which can be upgraded later in the game.

As shown on the HUD, whenever a button has a keyboard shortcut it will be displayed to the side of the box, (this can be turned off in the pause menu), this is for new players to help them learn the controls.

Hotbar

The hotbar can hold 3 items chosen by the player, this allows the user to easily switch what item they are currently holding by switching between the three slots using the 1,2 and 3 numbered buttons on the keyboard, as well as potentially using the mouse wheel.

The user can choose what items show up in the hotbar through the inventory, by dragging a slot of the inventory onto a slot of the hotbar, this will overwrite the item displayed in that slot of the hotbar. The item shown in the hotbar will not be taken away from its slot in the inventory, instead it will act as a sort of shortcut to the item in the inventory, so If the player uses up that item (item gets destroyed), it will be removed from the inventory and therefore be removed from the hotbar.

Torch

To the left of the hotbar is a torch icon, this is a simple graphic on the overlay that will show to the player if their torch is on.

I may also add a battery charge to the torch, shown by a percentage bar underneath the icon, to add an extra level of gameplay, where the player may have to wind up the torch to charge it, this will take time but will also produce noise that may make enemies investigate.

Crouching Mechanic

The crouching mechanic is one that I’m interested in at the moment.

If the player is standing normally then the icon to the right of the hotbar will show a character stood up, however if the player is crouched it will show a crouched character and the player will a debuff to their speed.

Crouching will mean that the player will be able to hide behind and inside certain objects to avoid detection.

(May not be implemented)

Health

Above the hotbar are 3 hearts.

These hearts can be full, half depleted and empty. If all hearts are empty, the player is dead.

Damage dealt by attacks from enemies will be rolled as a dice, and can be anywhere between 0 damage, 1 damage (half a heart) up to 1 and a half damage.

At the start of a dungeon run the player will have full health, they may obtain potions throughout the game that will heal them a bit, drinking a potion may increase your heart rate.

Eyeball

At the top right of the HUD, there is an eyeball, if the eye is open, it will mean that the player has been detected and is being ‘hunted’ by the enemy, if the player can avoid the enemy’s eyesight long enough, eventually they will give up and the player will be ‘hidden’ again.

## Systems diagram

## Key Variables/Classes

|  |  |  |
| --- | --- | --- |
| **Variable/Class Name** | **Data Type** | **Description** |
| Overlay Manager | Class | Acts as the parent to all overlays, will have functionality to manage the separate overlays and render them. |
| Overlay | Class | Acts as a parent to elements. Adds functionality to render and manage elements. |
| Element | Class | Acts as the smallest class of the overlay system. This will be used to define the behavior and rendering of shapes,images,text etc. Anything that is needed to make UI |
| Dungeon | Class | Acts as a parent to Rooms and Tiles. The dungeon is the environment that the player will explore and complete. It will use methods to generate rooms that will fit into a generated layout. |
| Room | Class | Acts as a blueprint for a room inside the dungeon, will have a room type and a shape. This shape contains strings that represent different tile types. |
| Renderer | Class | This class defines how to render the dungeon to the screen. This is highly important, without this class the user would not be able to see the game environment. |
| Screen | Pygame.surface | This is the variable that holds the reference to the window/surface that the game screen will be rendered to. |
| AI | Class | This class holds the AI that determines the movement patterns and other aspects of entities that are not player controlled. |

## Algorithms

### Dungeon Generation

For the dungeon, I need an algorithm that will generate the room layout. The rooms shouldn’t be all cluttered in the middle and should sort of branch away, but in a more natural way then just sharp corridors that sometimes branch.

Generated amount of rooms = 0

max amount of rooms = some number

While generated amount of rooms is less then the max:

* A room is chosen (Current Room)
* The Sides of the room are entered into a variable (sidesTaken)
* If the current room has more than 1 room and it is locked, then it will choose a different random room and restart the loop
* The current max zone id will be updated if the current room has a value higher than the current max.
* If the current room is locked and not in the list of locked rooms, then it will be appended
* If the current room has all 4 sides open or a random chance is less then the focus chance, then it will randomly choose a side.
  + If that side is currently inside of the gen data dictionary
    - it will then check if that room is locked or its zone id is not equal to the current rooms ID then a new room will be chosen.
  + If the last bullet point does not happen and the current room doesn’t have a side where the chosen side is.
  + it will then generate a room there and change the current room variable to the new room, and then the current loop will end.
* Else if a random chance is less then the ‘changeFocusEntirelyChance’ and the current room is in the rooms dictionary then
  + A random room will be selected from the dictionary of rooms and currentRoom will be updated to hold the randomly selected room
  + The current iteration will then end
* Else
  + A new list newSides will be created which holds all the sides of the current room that are closed
  + A side from this list will be randomly selected (chosenSide)
    - If there is a room on the chosen side and that room is locked, a different room will be chosen at random and the current iteration will end
    - Otherwise a new room will be generated at that side and the current iteration will end.

This algorithm will determine the shape of the dungeon layout by branching and randomising its focus across already generated rooms. This means that the rooms won’t be as crowded near the centre. However, with a large max number of rooms, it is more likely that more rooms will still generate at the middle due to an unweighted random selection. Meaning there will still be a large density of rooms at the middle, just less so with a smaller number of rooms.

### Entity pathfinding – A\*

The entities must have pathfinding to get to the player, or at least wander around their environment, I want to use a mix of algorithms, I plan to use A\* when the entities are near the player, and when they are not, then they should use a teleport/timer-based situation.

A\*:

* Create a Open dictionary, this dictionary will hold all the cell locations that are “open” as keys and their data as values
  + The data will consist of the cells g score, h score and its “parent” cell
  + G score = cost to get to this cell from the start
  + H score = estimated cost to get from this cell to the end cell
* Create a closed dictionary
  + once all of a cells neighbors have been placed in the open dictionary, a cell shall become “closed”
* LOOP:
  + Select the cell in the open dictionary with the cheapest f score
    - F score = g score + h score
  + For each neighbor
    - If neighbor doesn’t exist in open or closed and cell is walkable
      * Calculate the g score
      * Calculate the h score
      * Set its parent to the selected cell and add it into the open cell
    - Else if cell is walkable
      * Calculate the g score
      * Calculate the h score
      * If its g score + h score is less than the existing cell
        + Set the parent of the existing cell to the selected cell
      * Else
        + Skip this neighbor
  + Remove cell from the open dictionary and append it to the closed dictionary
  + If selected cell == final cell
    - Break loop
  + if open dictionary is empty
    - break loop
* if current cell == final cell
  + create steps list
  + LOOP
    - Add current cell to steps
    - Set current cell equal to the parent of current cell
    - If current cell == None (The starting cell has no parent)
      * Break
  + Reverse the steps list (We have now backtracked through the steps taken to get to the final cell, reversing the list just makes sure the steps are in the right order start>finish)
  + Return the list (this list is the path needed to get to the finish point)
* Else
  + Return empty list since there is no valid path to the destination

### Entity pathfinding – Teleport/Timer-Based:

* Select a random room that is either adjacent to the entities current room or is the current room.
* Make a list called possible targets that includes all of the walkable tiles inside of the selected room
* Choose a random coord from the list
* Determine the distance between the entities current position and the target
* Walk time = Divide the distance by the entities walking speed
* Create a timer that runs a function after the calculated walk time
* The function when ran, will set the entities coordinates to the target.

# C. Developing the coded solution

## Developing the overlay class

The Overlay class is a level under the Overlay Manager, the overlays are what will contain the buttons, text, images for a UI, the overlay manager simply defines the order they get drawn in, as well as streamlines how the user interacts with the overlays.  
I decided on using overlays because it reminds me of other games where the user can gain information about their characters or inventory state through sorts of “overlays” that appear on screen.

I also believe it to introduce a nice grouping sort of feature, where an overlay can be a grouping of “elements”.

I love this idea as it takes a sort of hierarchical structure and applies it to my game. It also makes it easy to understand and use.

Overlay Manager = group of overlays

Overlay = group of elements

Elements = visual and/or interactive item on screen e.g. button

### Structure

1. Variables:
   1. Elements (Dictionary) (Private)
      1. Has the keys “allElements” and “interactive”.
      2. The key “allElements” has a value of type ElementList, it will hold all elements inside of the Overlay.
      3. The key “interactive” has a value of type ELementList, it will only hold the visible elements inside of the overlay
   2. Screen (pygame.Surface)
      1. Has the screen surface of which the elements will be rendered onto.
   3. Hitbox (pygame.Rect)
      1. Holds the x,y,width and height components of the overlay.
   4. Name (String)
      1. Holds the name of the overlay.
      2. May be used to identify it by the overlay manager.
   5. Screen Flags (List)
      1. Holds all the screen flags for the screen surface.
   6. Pos (int,int)
      1. Holds the x and y component of the overlay.
   7. Size (int,int)
      1. Holds the width and height component of the overlay
   8. Parent (Overlay Manager)
      1. May the hold the parent overlay manager of the overlay
   9. Visible (Boolean)
      1. Determines whether or not the overlay is updated and rendered to the overlay managers surface
   10. Highlighted (Element)
       1. Holds the currently “highlighted” element, e.g. the one currently being held down by the user.
2. Unique Property
   1. Elements
      1. This property returns the contents of the key “allElements” from the Elements variable
   2. Interactive Elements
      1. This property returns the contents of the key “interactive” from the Elements variable
   3. Elements By Name
      1. This property returns the contents of the “allElements” key, formatted as {\*Elements Name: Element}
3. Methods
   1. Align To Center
      1. Changes the (Get Offset) methods so that when the offset is applied the element is centered

Constructor:

VARIABLES:

Elements -> List, contains all elements within the control of the Overlay

Screen -> pygame.Surface, the surface that the overlays elements will get drawn to

Name -> String, contains the name of the overlay so it can be found easier

Screen Flags -> List, Contains all of the flag integers for the screen, e.g. pygame.SRCALPHA – Allows for transparency

Size -> (int, int), Contains the width and height of the screen object

Hitbox -> pygame.Rect(), Contains the x,y,width and height position of the screen in one place

Parent -> OverlayManager, stores the overlay manager that the overlay is stored within

Pos -> (int, int), stores the x and y position of the Overlay within an overlay manager so it gets drawn in the right area

Visible -> Boolean, stores the overlays state of visibility

Highlighted -> Element|None, Stores the element contained in the overlay that is currently being held down/highlighted

## Developing the overlay manager.

During this stage of development, I have to make the overlay manager, as well as some other classes that will work within it.

Overlays will be the driving force behind the User Interface, for example, the game will have 1 overlay manager and will contain overlays, Overlays are groups of elements such as text boxes, buttons etc. The New Game screen will be an overlay which will contain 5 buttons which represent game slots.

The overlay manager was a nice addition to the structure I believe, as it again groups the overlays together and allows an easy way to manage/access them without me having to put them in a dictionary or something later on.

An overlay manager can hold as many overlays as needed. The list of overlays that the overlay manager is important as the order of the list determines which overlays get drawn first. For example, you would want a overlay that shows health information to be at the start of the list, so it’s the first thing to get drawn onto the game screen, so if you have an inventory overlay, when it gets opened, it will render over the health so the health doesn’t block anything in the inventory.

The overlay manager will also only render overlays that are visible, this is to allow the game to have multiple overlays and only use a few at once, like in a main menu. Each screen is technically there, some just are hidden until you open them.

### Structure

1. Variables:
   1. Overlays (List)
      1. Will contain a list of the overlays contained within the Overlay Manager, these are the overlays that the manager will have control over.
      2. This list may increase or decrease in size throughout the program.
   2. Screen (pygame.Surface)
      1. A pygame surface which is the surface that the overlays inside of the manager will be rendered on.
      2. This surface can then be taken and drawn onto the main screen surface where the player can then see it.
   3. Name (String)
      1. This will contain the name of the Overlay Manager
   4. Screen Flags (List)
      1. This will contain any flags for the screen surface, such as pygame.SRCALPHA which will mean that each pixel can on the surface can have an alpha value and therefore be transparent.
      2. This flag is recommended for the overlay manager since this module is intended to be used to draw interactive “overlays” that lay over a game or something else.
   5. Size (List(Int,Int))
      1. This contains the width and height (int,int) of the screen surface
   6. Held Down Keys (Dictionary)
      1. A dictionary that is intended to hold a Boolean value of true for any key the user is currently holding down, and reset it to false if the user lets go.
   7. Hitbox (pygame.Rect)
      1. Holds the x,y,width and height components of the overlay manager.
   8. X,Y (Int)
      1. The x and y components of the overlay manager (probably wont be used)
2. Methods:
   1. Update Hitbox
      1. This method will be used to update the hitbox variables components if the x,y,width or height components of the object are changed.
   2. Get Key State Event
      1. This method will attempt to retrieve the Boolean held down state of the key from the held down dictionary.
      2. if no such key exists in the dictionary it will return False, else it will return the keys state (True or False)
   3. Set State Event
      1. Sets a key inside the held down dictionary and will give it a Boolean value for status.
   4. Append Overlay
      1. Will append an overlay to the list Overlays inside of the manager.
      2. If the overlay provided is already inside of another manager, it will remove it from that manager and continue with placing it inside of its self.
   5. Remove Overlay
      1. Will remove a given overlay from the list Overlays inside of the manager.
   6. Insert Overlay
      1. Will place an overlay inside of the manager at a given index.
      2. It will attempt to remove the overlay from the list first to make sure no duplicates are made.
   7. Move Overlay Forward
      1. Will move the given overlay forward in the list Overlays by a given amount.
      2. This is used to make sure overlays are rendered in the correct order.
   8. Move Overlay Backward
      1. Will move the given overlay backward in the list Overlays by a given amount.
      2. This is used to make sure overlays are rendered in the correct order.
   9. Is Overlay At Position
      1. This method will return True if a given vector (x,y) collides with an overlay inside of the list.
      2. Else it will return False.
   10. Get Overlay At Position
       1. This method will return the topmost overlay at position (x,y).
       2. Will return None if an overlay isn’t found.
   11. Get Overlays At Position
       1. This method will return all overlays at position (x,y).
       2. Will return [] if an overlay isn’t found.
   12. Get Overlay By Name
       1. This method will return the first overlay found with a matching name
   13. Get Visible Overlays
       1. This method will return all overlays that are currently visible
   14. Set Overlay Visible
       1. Will set the provided overlays visibility to the Boolean value given
   15. Pre Update (Replaceable)
       1. Empty method that will be called before starting to update overlays.
       2. May be replaced by outside code.
   16. Post Update (Replaceable)
       1. Empty method that will be called after finishing the overlay update loop.
       2. May be replaced by outside code.
   17. Pre Draw (Replaceable)
       1. Empty method that will be called before starting to draw overlays onto the manager surface.
       2. May be replaced by outside code.
   18. Post Draw (Replaceable)
       1. Empty method that will be called after drawing overlays onto the manager surface.
       2. May be replaced by outside code.
   19. Pre Overlay Update (Replaceable)
       1. Empty method that will be called before every overlay update.
       2. Gets passed the overlay manager and current overlay that is updating.
       3. May be replaced by outside code.
   20. Post Overlay Update (Replaceable)
       1. Empty method that will be called after every overlay update.
       2. Gets passed the overlay manager and current overlay that finished updating.
       3. May be replaced by outside code.
   21. Pre Overlay Draw (Replaceable)
       1. Empty method that will be called before every overlay gets rendered.
       2. Will get passed the overlay manager and overlay
       3. May be replaced by outside code.
   22. Post Overlay Draw (Replaceable)
       1. Empty method that will be called after every overlay gets rendered.
       2. Will get passed the overlay manager and overlay
       3. May be replaced by outside code.
   23. Update
       1. Will get all visible overlays.
       2. Clears the overlay manager screen surface.
       3. It will then call all the other update methods and overlay methods in order.
       4. Then it will call the draw methods for the manager and overlay.
       5. Called every frame.

## Developing the element class

When developing the element class I decided that the Element class is the base/smallest type In the Interactive Overlays module.

The element is highly important as it is the base of everything, the element could be text, an image, a line etc.

I called them elements, partially due to hearing other similar objects in programs be called elements, but mostly because they remind me of elements in science. They do nothing by themselves, but when put together, they are reactive. These elements cant and wont do anything by themselves, they are only definitions of behaviours, however when put into an overlay, the user will be able to interact with them through mouse, keyboard, etc.

For example, if someone is playing a game, they wouldn’t want their inventory to open and close, flickering across the screen, without any cause relating to the user. You would want the user to interact with an icon or maybe a keybind to open and close the inventory. These inputs will be passed through an overlay, which will then determine which element was targeted and will active one or more of the behaviours defined within it.

### Structure

1. “Action” Methods must:
   1. Be replaceable, so that when an Element is created it can be given a function that defines its behaviour E.g. A rectangle Element being given a function that shuts the program when pressed.
   2. Must always take parameters in the pattern (self, \*args, \*\*kwargs), this is to avoid any errors that may arise if an appropriate function is not provided and an action method is called and given information that does not match.
2. Variables:
   1. Image:
      1. May not be used in some elements, but is important to have so that I can later pass an image for an element to use in its drawing method for any extra detail.
   2. Base Image:
      1. This variable is needed to store the original image before any manipulations such as cropping is done, this is needed to maintain the images quality, as manipulations will use the base image, instead of the current image (The current image will usually be manipulated and therefore have lost quality)
   3. Name:
      1. This variable will simply hold the name of the element so that an overlay may do a search by name. This acts as an identifier and therefore should be unique
   4. Interactive:
      1. This variable should simply hold a true or false value, that indicates if it can be interacted with by the player. If an overlay detects a user input that originates from the element, then it will check if it is interactable, if not it will ignore this input.
   5. Visible:
      1. Another true or false value that lets the parent overlay decide if the element should be rendered to the screen or not.
   6. Hitbox:
      1. The hitbox simply holds the outline of the element/ holds an x,y,width and height value for the element, it is updated by a method if any of the other values change.
   7. Parent:
      1. This is a highly important variable since it holds a reference to the elements parent overlay. If a element does not have a parent, then it simply wont be rendered anywhere and cant be interacted with.
   8. X,Y:
      1. Two separate position variables that hold the X and Y components of the elements position on screen
3. Methods:
   1. Get Offset X (Replaceable)
      1. This method is changed by the align methods within the class, it is used to return the offset values for the X component of the element, it is represented as a method because the offset values are expressions whose results may change if other properties of the element is changed.
   2. Get Offset Y (Replaceable)
      1. This method is changed by the align methods within the class, it is used to return the offset values for the Y component of the element, it is represented as a method because the offset values are expressions whose results may change if other properties of the element is changed.
   3. Pressed (Action)(Replaceable)
      1. This method defines the behaviour of the element when a user presses on the element with their mouse. It may be changed by code outside of the object. This is because different elements may want different behaviours when “used”.
   4. Released (Action)(Replaceable)
      1. This method is a replaceable method. This method defines the behaviour of the element when a user releases their mouse on the element. It may be changed by code outside of the object. This method will always eventually be called after the Pressed method is called.
   5. Held Down (Action)(Replaceable)
      1. This method is a replaceable method. This method defines the behaviour of the element that should happen if the player continues to hold down their mouse button on the element when they press on the element.
   6. Other Element Pressed (Action)(Replaceable)
      1. This method is a replaceable method. If the user presses on a different element then this one, then this method will be called. This is useful for something like a drop down list so that it closes if the player interacts with something else.
   7. Update Method (Replaceable)
      1. This method is replaced by code outside of the class. This method is called during the update cycle of elements, outside code may define the logic behind when elements are updated.
   8. Update Hitbox
      1. This method should just update all properties of the hitbox (pygame.Rect) object of the element in order to keep it up to date after any changes.
   9. Draw (Replaceable)
      1. This method should contain all of the steps that are taken when drawing this element to the screen. This method may be replaced by code outside of the element so that elements draw steps may be customized.
4. Methods (Continued)
   1. Resize Image / Resize Image By Amount
      1. There should be 2 methods like this. One that will resize the image by integer increments/decrements in its width and height and one that will set the width and height. These functions will take the base image, resize it and set the image variable to the result.
   2. Align Methods
      1. The Align Methods will set the “get offset x/y” methods to their respective equations. There will be 9 align methods   
         (top left, top middle, top right)  
         (middle left, center, middle right)  
         (bottom left, bottom middle, bottom right)
   3. Pre/Post Update (Replaceable)
      1. The pre update method is called before the update method and the post update is called after. These can just be used to separate update methods into easier to understand chunks.
   4. Update Method
      1. This method will call all update and draw methods in the right order and pass the parameters (self, screen, \*args, \*\*kwargs) to each update/draw method.
   5. Front/Back Movement Methods
      1. Move the element back or forwards in the parent overlays list, this will change the order in which the elements are rendered to the overlays surface.  
         Will take in the amount to move relatively as a parameter
   6. Bring To Front/ Send To Back
      1. Similar to the previous movement methods, this will change the location of the element in its parents list, however these methods will either send all the way to the front of the list, or the end. Meaning the element will either be rendered first, or rendered last based on which method is called.
   7. Copy
      1. This will iterate over the dictionary representation of the element and make a copy of every key/value and place it into a new copy of the element.

## Developing the element list class

When developing the method ‘copy’ in the Group element I needed the ability to make a copy of the groups list of elements, however it wouldn’t make a copy of the elements when using a normal list, instead it would keep a reference to the original elements meaning the group copy didn’t work. So I had to make a separate class that works exactly like a list, it would however use a custom copy method that makes a new list and iterates through all items and uses a copy method on them.

I decided to do this by making the Element List class that would inherit from the built-in list and overwrite the copy method.

I originally tried to edit the built-in list class from python, which doesn’t work due to it being immutable so instead I came up with this idea. I would make a custom list class that would inherit from the python list and would define any custom behaviour I wanted.

This was needed as the built-in list wouldn’t allow for a feature that I wanted.

### Structure

1. Methods
   1. Copy
      1. Makes a new element list, iterates through all of the elements in the current list and makes copies of them and puts them into the new list. The new list is then returned.

## Developing the element dictionary class

This class was put into development for the same reason of the element list class. I needed ‘Copy’ methods inside of other classes to copy elements correctly and so I put into development a custom version of the built-in dictionary class.

It would need to work exactly like a dictionary except replace the copy method, this was easy by inheriting the built-in dict class and just defining a copy method.

Again, I tried to edit the built-in dict class which didn’t work out, so I did the same thing I did with the element list class and designed my own that would define custom behaviour while inheriting from the python dictionary class.

### Structure

1. Methods
   1. Copy
      1. Makes a new element dictionary, iterates through all of the keys and elements in the current dictionary and makes copies of them and puts them into the new dictionary. The new dictionary is then returned.

## Developing the text class

When I have done projects before text in pygame has always been a bit confusing and hard to remember and I figured making text as a sub-element would be a great use of the element and would follow the structure of the element/overlay/overlay manager hierarchy quite well.

It was pretty easy to make as all I really had to do was make a new class that wrapped around the already made pygame.fontObject class. I would like to believe that my class is more understandable and easier to use then the pygame.fontObject class, so far while i have been using it I would say that I find it better.

Making text into an element sub-type was important and an integral part of any game as it would allow for me to display dynamic texts, such as the players current level, what zone they are in, and potentially names of items.

This was one of the very first element sub-types I desired to make due to it not only being a good test of the element class structure, but also due to it having such a high need within my project.

### Structure

1. Variables
   1. Text
      1. (String) Stores the text of the text of the object
      2. if updated it will then call the update\_text method
   2. Font
      1. (String) Stores the font of the object
      2. If updated will call update\_text method
   3. FontObject
      1. (Pygame Font Object) Pygame Font object holds details about the text object
   4. RenderedText
      1. (Pygame Surface) The rendered version of the text that can be drawn to the screen
      2. Will get updated if any of the texts variables that affect the render are changed
   5. Colour
      1. (int,int,int) Controls the colour of the rendered text
      2. If updated will call update\_text method
   6. Size
      1. (Int) Max height of characters in pixels
      2. If updated will call update\_text method
2. Methods
   1. Get Bold
      1. Returns a Boolean value indicating if the object is formatted as bold
   2. Set Bold
      1. Takes a Boolean value and will format the text as bold if the given value is true otherwise it will make it not bold
      2. Takes a boolean parameter
   3. Get Italic
      1. Returns a Boolean value indicating if the text is italic
   4. Set Italic
      1. Sets the italic boolean value for the text object
      2. Takes a boolean parameter
   5. Get Underline
      1. Returns a Boolean value indicating if the text is Underlined
   6. Set Underline
      1. Sets the Underline boolean value for the text object
      2. Takes a boolean parameter
   7. Get Strikethrough
      1. Returns a Boolean value indicating if the text is Strikethroughed
   8. Set Strikethrough
      1. Sets the Strikethroughed boolean value for the text object
      2. Takes a boolean parameter
   9. Renderer
      1. Returns a rendered surface of the text object
      2. Takes in text as a string
      3. Would be used in the scenario of needing to use the “renderer” of another text object without needing to make a new text object
   10. Update Text
       1. Whenever a change is made to the parameters of the text that affect it visually, update text will be called to update the visuals.
       2. Changes could be made to streamline this method by only updating once at the end of frame before everything is drawn, instead of potential repetitive updates throughout one frame.
   11. Draw
       1. Will draw the text to the parent overlay, if the object is visible and has a parent overlay.

## Developing the still image class

For this class all I wanted was an element that could display a still image, it inherits all of its variables from the element class and only defines a draw method.

This was most likely, the easiest element sub-type to make, due to it not having much difference from the base element, and not being very dynamic. The image may change using outside code, but other than that, it is designed to be static, for potentially displaying item icons or acting as a background for elements/overlays.

### Structure

1. Methods
   1. Draw
      1. Will draw the still image to the parent overlay if the object is visible and has a parent overlay

## Developing the rectangle class

For this class I wanted a simple rectangle element that I could give coordinates and it will draw a rectangle.

This was also easy like the image due to it being relatively simple and matching the base element class very well without adding too much extra code.

### Structure

1. Variables
   1. Hitbox
      1. Holds the hitbox (\*left corner, \*size)
   2. Colour
      1. The colour of the rectangle when it gets drawn to its parent overlay
   3. Border Thickness
      1. The thickness of the border of the rectangle
      2. If changed from its default value then the rectangle shall not have a fill, and will instead be a border
   4. Border Radisus
      1. Will determine the overall radius of the corners of the rectangle
      2. If left alone, the angles would be sharp 90 degrees
   5. (Top|Bottom Left|Right) Border Radius
      1. Determines the radius of a given corner of the rectangle rectangle
      2. Left alone will default to border radius
2. Methods
   1. Draw
      1. Draws the rectangle onto its parent overlay

## Developing the Circle class

Similar to the rectangle class, I wanted a simple base circle class. This could have been achieved with just a circle image and then using the still image class. But I wanted to make it as its own class so it would be scaled properly and would be more functional.

### Structure

1. Variables
   1. Hitbox
      1. Holds the top left|right corners of the circle and its diameter as width and height
   2. Radius
      1. Holds the radius of the circle
   3. Colour
      1. Holds the colour of the circle
   4. Border Thickness
      1. Holds the thickness of the circles border
      2. If changed from its default value the circle will not have a solid fill and will instead be a border
2. Methods
   1. Draw
      1. Draws the circle to its parent overlay

## Developing the line class

The line class was a bit harder to develop then the other classes due to it using two sets of coordinates. It was originally confusing to develop as I had to plan how I was going to do since the element class that the line would inherit from only uses one set of coordinates.

It ended up not being much of a hindrance however and was relatively easy to implement in the end. I simply overcomplicated it at first.

I fixed it by just defining a new set of end coordinates in the line class.

It did take a bit of trial and error and overthinking the problem a bit, and the answer ended up being quite simple, the line class is a bit less useful then the other sub-types of the element, but it still has its uses, and is also planned to be used in a slider element sort.

### Structure

1. Variables
   1. End X
      1. Holds the end x coordinate of the line
   2. End Y
      1. Holds the end y coordinate of the line
   3. Hitbox
      1. Stores the start coordinate of the line as well as the x,y distance from the end coordinate
      2. This creates a rectangle shape around the line
   4. Colour
      1. Stores the colour of the line
   5. Thickness
      1. The thickness of the line
2. Methods
   1. Update Hitbox
      1. Updates the hitbox in case any of the lines parameters change
   2. Draw
      1. Draw the line onto the parent overlay

## Developing the group class

The group class was an interesting element to develop, because it acts a lot like an overlay, however I needed an object to act like an overlay but be one step lower.

It also introduced a lot of issues as I realised a lot of my code for other elements wasn’t written with this level of complexity in mind. However, the issues were rather easy to fix.

The group will hold other elements within an internal list inside of the group element. Due to its unique nature, it needed a lot more thought, planning and code then the other elements, but ended up being rather simple In nature. Unlike the overlay, the elements coordinates inside of the group stay relative to the overlay. The group is simply a list of elements, it doesn’t define any new behaviour for the elements, it just may change their position/size if the groups properties are edited.

### Structure

1. Variables
   1. Visible
      1. Determines if the group and its internal elements are visible
   2. Parent
      1. Determines the parent for all elements contained
      2. Any changes to this variable will propagate through the sub-elements of the group
   3. Elements
      1. Returns an ElementList object containing a list of the elements contained in the group
   4. X
      1. Returns the top left x position of the rectangle containing all sub-elements in the group
      2. Any changes to this will offset the sub-elements so they stay relatively the same position in the group
   5. Y
      1. Returns the top left y position of the rectangle containing all sub-elements in the group
      2. Any changes to this will offset the sub-elements so they stay relatively the same position in the group
   6. Width
      1. Distance between the top left position and the top right position of the rectangle containing the sub-elements
   7. Height
      1. Distance between the top left position and the bottom left position of the rectangle containing the sub-elements
2. Methods
   1. Append Element
      1. Appends an element to the group
   2. Remove Element
      1. Removes an element from the group
   3. Remove Element By Name
      1. Searches for an element in the group with a given name, and deletes the first it finds
   4. Is Element At Pos
      1. Using a given coordinate, searches for a element at the position and returns True if it finds one
   5. Get Element At Pos
      1. Using a given coordinate, searches for a element at the position and returns the element if it finds one
   6. Get Elements At Pos
      1. Using a given coordinate, searches for any elements at the position and returns all found

## Developing the dungeon definitions

The dungeon definitions is a separate file from the dungeon files and holds a lot of the definitions for the dungeon, such as room shapes, tiles within the different types of rooms that can be generated, etc.

It will also hold most of the values that remain constant throughout the dungeon files and throughout runtime.

This was rather easy to develop since any value I identified as constant, I just moved it over to this file while developing.

There are no classes defined within this file, only 1 function.

### Load Models

This function just loads all the models found in a certain folder path, it will generate model objects and store the images according to their model and their height.

The information of which image is which is stored within the images and folders name. The images name is its layer height. The folder holding the image has the name of the variation of the model. And the folder holding the variations holds the models name.

Each model has a “variation” system, so for a model “stone” I could make 2 variations “stone” and “mossy stone”. When generated, in some areas one could be preferred more than another. The variations are just to make generation look less boring.

## Developing the dungeon

The dungeon is an integral part of the project as it generates the “world” the player will explore.

### Generate Tiles

The generate tiles function will generate the room layout of the dungeon by a rather simple algorithm.

Each room has an Zone ID that describes the zone level of the current room

While generated amount of rooms is less then the max:

* A room is chosen (Current Room)
* The Sides of the room are entered into a variable (sidesTaken)
* If the current room has more than 1 room and it is locked, then it will choose a different random room and restart the loop
* The current max zone id will be updated if the current room has a value higher than the current max.
* If the current room is locked and not in the list of locked rooms, then it will be appended
* If the current room has all 4 sides open or a random chance is less then the focus chance, then it will randomly choose a side.
  + If that side is currently inside of the gen data dictionary
    - it will then check if that room is locked or its zone id is not equal to the current rooms ID then a new room will be chosen.
  + If the last bullet point does not happen and the current room doesn’t have a side where the chosen side is.
  + it will then generate a room there and change the current room variable to the new room, and then the current loop will end.
* Else if a random chance is less then the ‘changeFocusEntirelyChance’ and the current room is in the rooms dictionary then
  + A random room will be selected from the dictionary of rooms and currentRoom will be updated to hold the randomly selected room
  + The current iteration will then end
* Else
  + A new list newSides will be created which holds all the sides of the current room that are closed
  + A side from this list will be randomly selected (chosenSide)
    - If there is a room on the chosen side and that room is locked, a different room will be chosen at random and the current iteration will end
    - Otherwise a new room will be generated at that side and the current iteration will end.

Once the while loop has ended, a for loop will then iterate through the generated rooms and will generate their structure.

For room in rooms:

* Foreach room a banned rooms list is temporarily created
* This list will get entries based on conditions, currently the only condition planned is for the entrance room to have some sort of walls and not to be fully open.
* A Room object is created. The room class is one developed to hold all the attributes of each room, such as the enemies currently inside, the layout etc.
* The room object will be passed all the info of the generated room.
* The room object will then be inserted into the dungeons room dictionary.
* The room object will also be appended to the list relevant to the rooms zone id
* END

Now that the generated rooms have been converted into actual room objects instead of their previous dictionary format and have been placed into the actual room dictionary, the room dictionary has then got to be looped through in order to update all of the walls of the rooms so that the sides that are connected to other rooms have a door, after which the rooms tiles (each room is split into tiles, each of which are building blocks).

And finally, the function will then loop through all of the tiles inside of the dungeon, if the tile has less then 4 neighbouring tiles, then the tile will be converted to a wall. This is to avoid the user from walking into the void.

### Generate Image Layout

The function generate image layout will take the dictionary of tiles that was generated by “GenerateTiles”.

It will loop through the list and retrieve the models (lists of 2d images that get stacked to make a 3d model),

It will then place each 2d “slice” into a list inside of a layers dictionary with an index of its “height”, this height is used to offset its y value on screen to produce a 3d look.

### Get Layers

This will build the “layers” to be rendered on the screen, each layer is like 1 step in height.

For each step in height, It will “load” a small radius around the player to be rendered. Should only be enough to fit on screen, no need to render more off screen. This area will be drawn onto 1 layer (pygame surface) which will then be added to a dictionary.

Once each height is generated, the dictionary will be returned.

### Structure

1. Variables
   1. X
      1. Holds the x position of the dungeon
   2. Y
      1. Holds the y position of the dungeon
   3. Min Rooms
      1. Holds the minimum amount of rooms to be generated
   4. Max Rooms
      1. Holds the maximum amount of rooms to be generated
   5. Rooms
      1. Dictionary: dict[tuple[int,int], Room]
      2. Will hold the room objects once they are generated by the generate tiles function
      3. The key to the room objects are their positions e.g. (0,1),(0,0),…
   6. Starter Room
      1. Holds the coordinates of the very first room to be generated and from which the dungeon will branch off of
   7. Tiles
      1. Dictionary: dict[tuple[int,int], int]
      2. Holds the coordinates as keys, and int representation of the types of tiles
   8. Dungeon Layout
      1. Is a dictionary of layers, each layer is a dictionary and the key to the layers are their heights.
      2. The layer dictionaries holds coordinates as keys and pygame surfaces as values.
      3. Holds all of the images to be drawn to screen that represent each tile
   9. Levels
      1. Is a dictionary that holds all of the zone IDs as keys and the rooms that correspond to each zone as the values
2. Variables (cont…)
   1. Max Zone Id
      1. Holds the highest zone id that was generated
   2. Player
      1. Holds the player object
3. Methods
   1. Unlock Room
      1. Will unlock a room, so its no longer locked and can be entered by the player.
   2. Generate Tiles
      1. Generates all of the rooms and tiles for the dungeon
   3. Generate Image Layout
      1. Fills the dungeon layout dictionary with all of the layers of the dungeon
   4. Get Layers
      1. Gets the layers to be drawn around the player.

## Developing the Entity class

The entity class is a base class for all entities in the game, it holds health, walk speed, etc.

## Developing the AI class

AIs are very important as they are the base class for all the enemies the player will fight against, the will also need to be able to pathfind and move around.

Originally I wanted them all to pathfind in real-time, the issue is however, is with a large dungeon, there may be lots of enemies, and if a lot of enemies are pathfinding at the same time, that may be an issue.

So for this class, I introduced two very important decisions.

First of all, I introduced threading, threading was important as it means the AIs can be processed quicker. Secondly, I introduced separate AI movement methods. If an enemy is far enough away from the player, then the AI wont use pathfinding, instead it will choose a neighbouring room and a random open “air” tile it can move to, and it will wait a calculated period of time that matches with distance until it moves there. This was done by making a custom thread timer, it works by putting a thread to sleep and then once its done it will complete a function (which teleports the entity to its location). This is a fast way to process the entities movement, as the entity doesn’t actually have to move normally off screen, as the player wont see it.

The AI class needed a lot of extra content made even when inheriting from the entity class, due to it needing methods to pathfind.

1. Variables
   1. Layers
      1. (class variable)
      2. Holds the different layers of the enemies models
   2. All
      1. (class variable)
      2. Holds all of the Ais that are made
   3. Moving
      1. (class object)
      2. Dictionary that holds movement types as keys and holds all of the Ais that are currently using that movement type
   4. Current Id
      1. (class object)
      2. Each AI needs a unique id, this variable holds a unique id that a entity can take, and then it will increment
   5. Frame
      1. The current frame of animation, if it has one
   6. Id
      1. The entities id
   7. Dungeon
      1. The dungeon the entity is contained within
   8. Dungeon Level
      1. The current dungeon level the entity is within
   9. Current Target
      1. The current target of the AI
2. Variables (Cont…)
   1. Room
      1. Current room the AI is within
   2. Move Timer
      1. Holds a timer object if the AI is destined to move
   3. Out Of Range
      1. True or false value based on if the AI is out of range of the player to experience regular movement or be rendered
   4. Player X/Y (2 separate variables)
      1. Holds the players position, this is needed to decide what movement type the AI should use
   5. Static period
      1. List that holds the min wait time and the max wait time
      2. The AI will wait a random time between these two values before moving again after moving.
   6. Movement Type
      1. The Current movement type the AI is using
3. Methods
   1. Move To
      1. Moves the AI to a given position
   2. Target Activation
      1. Sets the activation level of the target to a given value
   3. Choose Target
      1. Browses a random room connected to the current room, it chooses an open tile and sets a movement timer which will move the AI to that tile after a calculated amount of time relevant to the amount of distance between the current and end position.
   4. Distance to player
      1. Returns the current distance to the player object
   5. Update
      1. Updates the AI
      2. This method will affect movement

## Errors of element implementation

### Copy Method

When designing the copy method for the element class, I originally tried designing a copy method for all the sub-element classes such as Rectangle. This eventually led to errors or a lot of pain in trying to pass in the right arguments for every different copy method or getting errors from a certain sub-element class not having a copy method defined.

A screenshot of a computer program

Description automatically generatedOne of the old copy methods

Due to the many issues I was facing, I finally realized a much simpler and better approach to the issues, and that was to make a single fit-all copy method that would simply iterate through the dictionary representation of an element and change the values of a new element to that of the copy. This works for every sub-element and is much better, it also takes up a lot less space.

A screen shot of a computer program

Description automatically generatedNew Copy Method

## Errors of element list implementation

### BEFORE IDEA

My original idea was to edit the built-in list.copy method without making any other classes. And so I developed a copy function and then set list.copy equal to my new function.

list.copy = elementCopy

This didn’t work, and I received an error. This led to me learning/realising that the built-in classes are immutable unlike any classes made by the programmer.

ERROR > TypeError: cannot set 'copy' attribute of immutable type 'list'

I was developing a copy function for the dictionary class as well at the same time. So I received this error technically for both.

After I received this error, it felt obvious that I wouldn’t be able to change the built-in classes but it was worth a shot and led me to do this instead, which now that ive done it, is much better then what I was going to do because now I have an original list class and my new one, which makes sense and looks better.

### AFTER IDEA

When developing the element list classes copy method, I received quite a few logic errors. Such as the elements not copying right, or the elements not having a method that matched the name.

I fixed the missing copy method by implementing an if statement that tries to find the copy method before trying to call it.



If it doesn’t find it, it will instead just append the element as it is without making a copy.

## Errors of overlay managers implementation

When implementing the overlay manager I encountered many issues.

While developing the code, I also coded the Main Menu which not only let me progress in the menus creation but also let me test the overlay manager, and other parts of my code.

This test code was extremely helpful in development because it let me see errors within my code and figure out how to fix them.

### Main menu background errors

When developing the main menu I put a background image behind the options, as can be seen here:



The background looks decent but it also caused me to be able to find many issues within my code.

### Index Error

One of the first errors I came across when testing the main menu was that if the player didn’t press a button and instead clicked on the background an error would be raised. This was a rather simple error to track down and fix because in the test file, when I click on the screen, the program would use the “get\_elements\_at\_pos” method under the Overlays class to retrieve the buttons the user pressed.

It would then get the last element rendered (The topmost element) within the list and process the click. The error came about because I was not checking if the list actually had any items inside it before indexing.

A screen shot of a computer

Description automatically generated

The red means lines deleted and green equals lines added.

As you can see, I got rid of the loop since the loop made no sense in this case and I added an if statement to make sure that the elements list isn’t empty before indexing. This way we don’t get the index error.

### Highlight Error

This error was encountered rather soon after the previous error.

When the user presses on an element, it gets saved to the overlay it originates from in the variable “Highlighted”, this is so when the user stops holding the button down, we can call that buttons release function and get the element from the Highlighted variable.

The error came about when the player releases the button but Highlighted isn’t set to None, this means when the player presses on anything that isn’t interactive, the last pressed elements function will be called again due to it still being in Highlighted.

This was fixed by setting Highlighted equal to None when the element is released.

A green board with white text

Description automatically generated

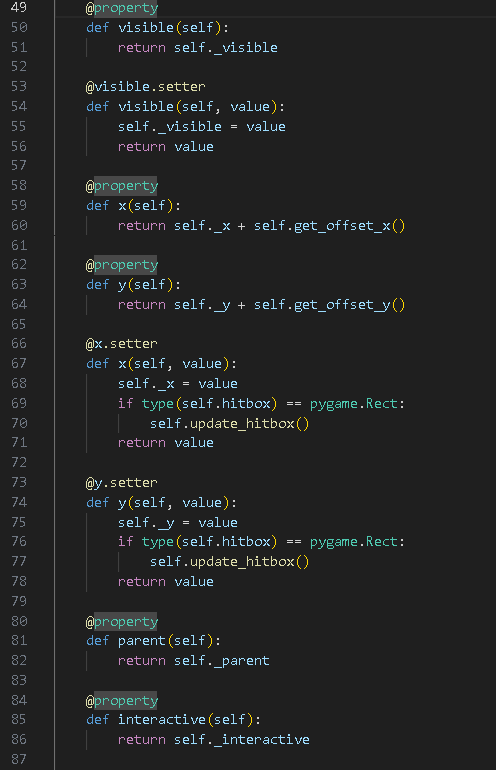
## Evidence of element implementation

### Constructor Method Implementation

A screenshot of a computer program

Description automatically generated

### Property Implementations



A screen shot of a computer program

Description automatically generated

A black screen with colorful text

Description automatically generated

### Update Hitbox Method

A black background with white text

Description automatically generated

### Empty Draw Method

****

### Image Manipulation Methods

**A computer screen shot of text

Description automatically generated**

### Align Methods Implementation

**A screen shot of a computer program

Description automatically generated**

**A screen shot of a computer program

Description automatically generated**

**A screen shot of a computer code

Description automatically generated**

### Update Methods

**A screen shot of a computer program

Description automatically generated**

### Element Drawing Order Manipulation

**A computer screen shot of text

Description automatically generated**

### Copy Method

**A screen shot of a computer program

Description automatically generated**

## Evidence of element list implementation

### Element List

A screen shot of a computer program

Description automatically generated

## Evidence of element dictionary implementation

### Element Dictionary

A screen shot of a computer program

Description automatically generated

## Evidence of overlay Implementation

### Constructor Method Implementation

**A computer screen shot of a program code

Description automatically generated**

### Property Implementations

**A screen shot of a computer program

Description automatically generated**

### Property Setter Implementations

**A screen shot of a computer program

Description automatically generated**

### Update Hitbox Method Implementation

**A screen shot of a computer code

Description automatically generated**

### Align Methods Implementation

**A screenshot of a computer program

Description automatically generated**

### Element Dictionary/List Manipulation

**A screen shot of a computer program

Description automatically generated**

**A screen shot of a computer code

Description automatically generated**

### Element Collision

**A computer screen with text

Description automatically generated**

**A screen shot of a computer code

Description automatically generated**

### Update Methods

**A screen shot of a computer program

Description automatically generated**

### Draw Method

**A computer screen shot of a program code

Description automatically generated**

## Evidence of overlay managers implementation

### Constructor Method Implementation

A screenshot of a computer program

Description automatically generated

### Properties

**A screen shot of a computer program

Description automatically generated**

### Getter and Setter for events states

**A screen shot of a computer program

Description automatically generated**

### Append, Remove and Insert function for the overlays

**A screen shot of a computer program

Description automatically generated**

### Move overlays in list

**A screen shot of a computer code

Description automatically generated**

### Overlay at pos?

**A screen shot of a computer program

Description automatically generated**

**A computer screen shot of text

Description automatically generated**

### Template Update Methods

**A screen shot of a computer program

Description automatically generated**

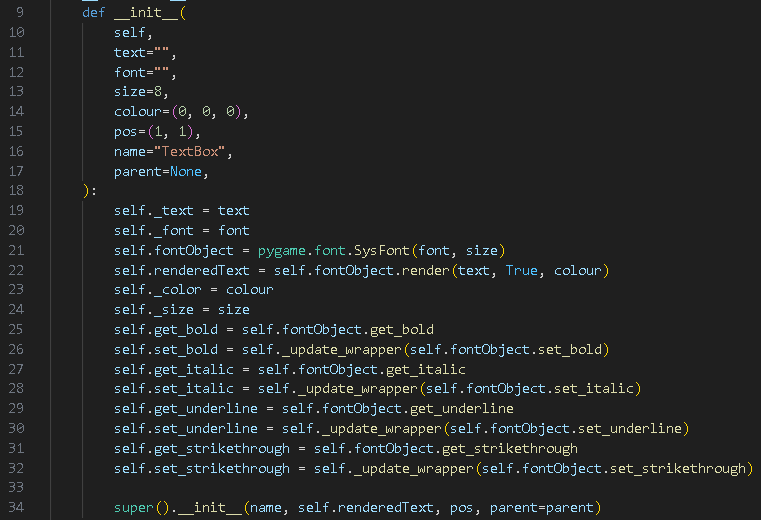
### Main Update Method

**A screenshot of a computer program

Description automatically generated**

## Evidence of text element implementation

### Constructor Method Implementation



### Renderer



### Properties

A screen shot of a computer program

Description automatically generated

### Update Text

A screen shot of a computer code

Description automatically generated

### Draw

A black background with blue and white text

Description automatically generated

## Evidence of still image implementation

A screen shot of a computer program

Description automatically generated

## Evidence of rectangle implementation

### Constructor Method Implementation

A screen shot of a computer program

Description automatically generated

### Draw Method

A screen shot of a computer program

Description automatically generated

## Evidence of circle implementation

### Constructor method implementation

A computer screen shot of a program

Description automatically generated

### Update Method

A screenshot of a computer

Description automatically generated

### Draw Method

A screen shot of a computer program

Description automatically generated

## Evidence of Line implementation

### Constructor Method Implementation

A computer screen shot of a program code

Description automatically generated

### Update Hitbox

A screen shot of a computer code

Description automatically generated

### Properties

A screenshot of a computer program

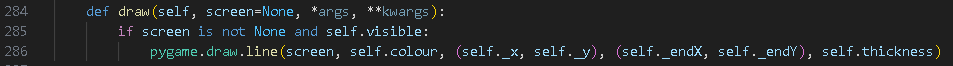
Description automatically generated

### Property Setters

A screen shot of a computer program

Description automatically generated

### Draw Method



## Evidence of Group implementation

### Constructor Method Implementation

A computer screen shot of a program code

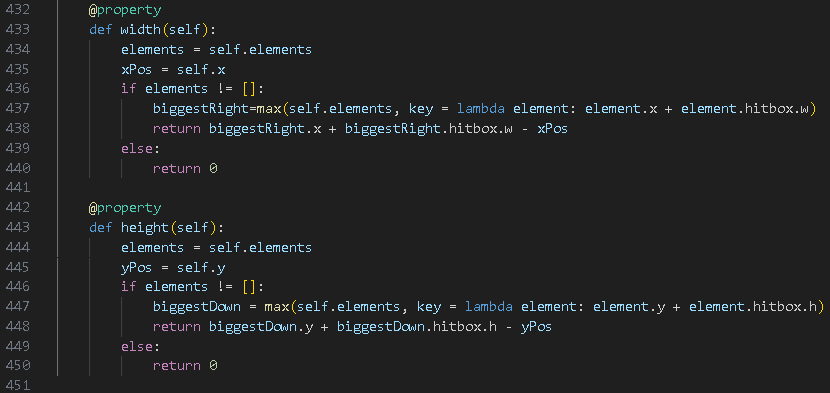
Description automatically generated

### Properties

A screenshot of a computer program

Description automatically generated





### Append Element

A screen shot of a computer code

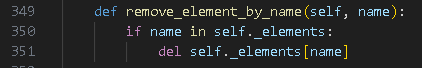
Description automatically generated

### Remove Element

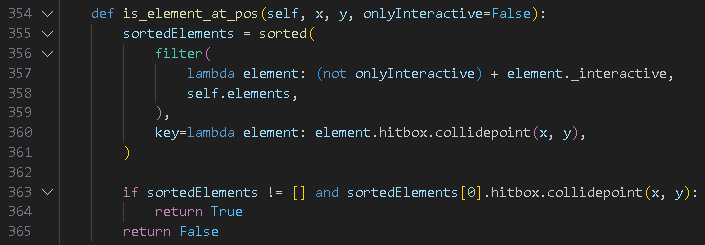
A screen shot of a computer code

Description automatically generated

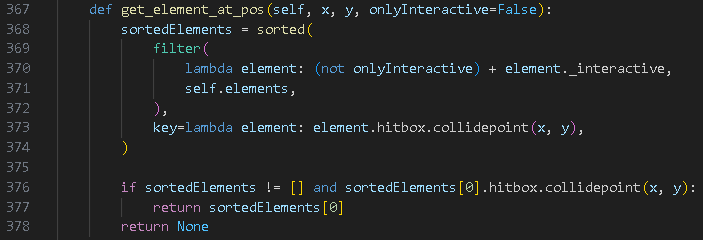
### Remove element by name



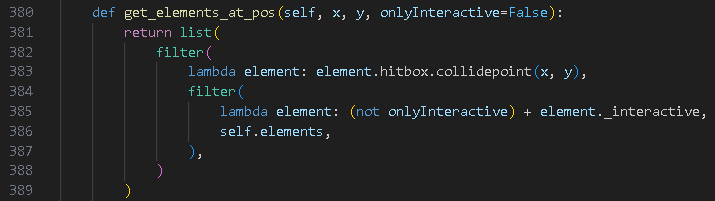
### Is element at position



### Get element at position

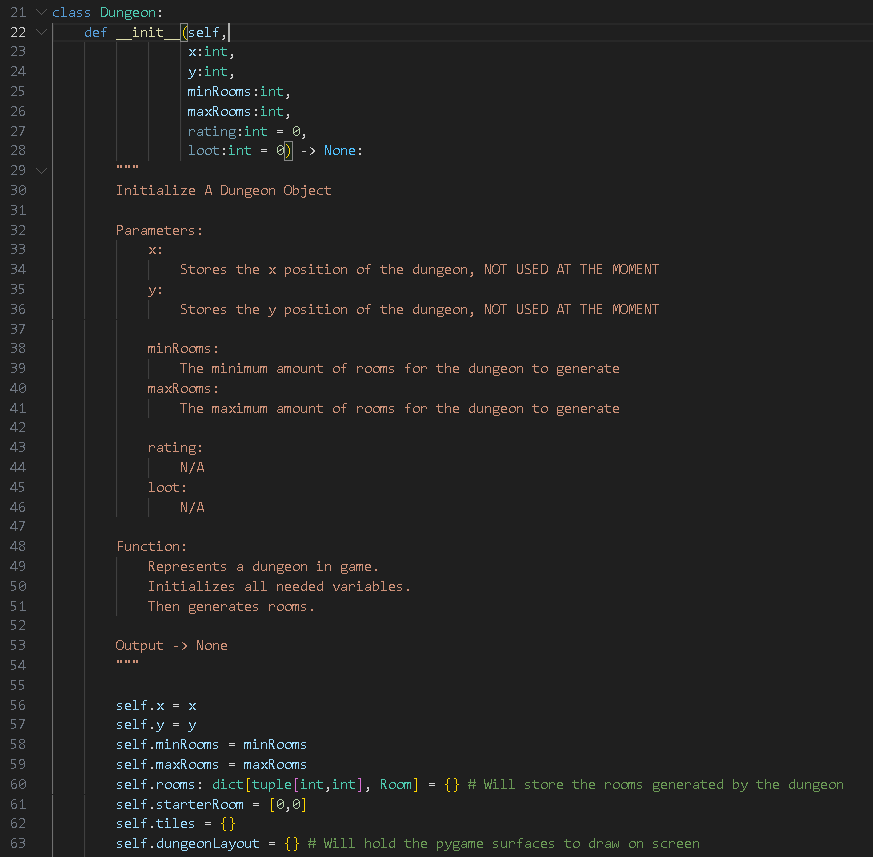


### Get elements at position



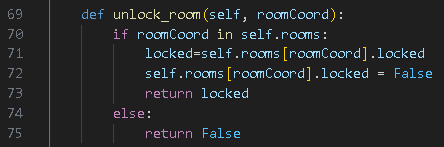
## Evidence of Dungeon Implementation

### Constructor Method Implementation





### Unlock Room

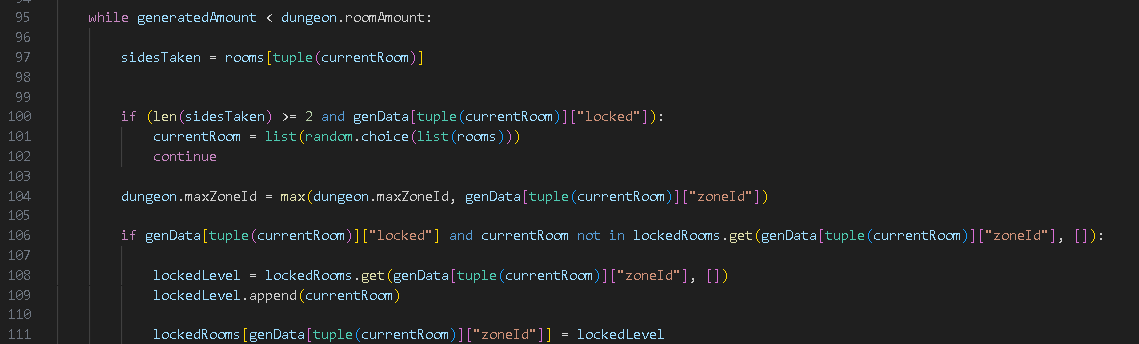


### Generate Tiles

#### pRE-lOOP SETUP



#### sTART-lOOP

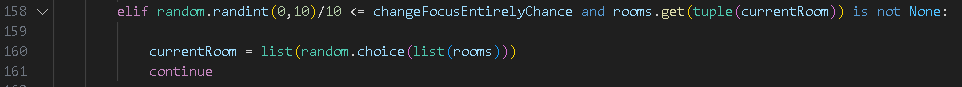


#### 1st Major If Statement





#### 2nd Major If



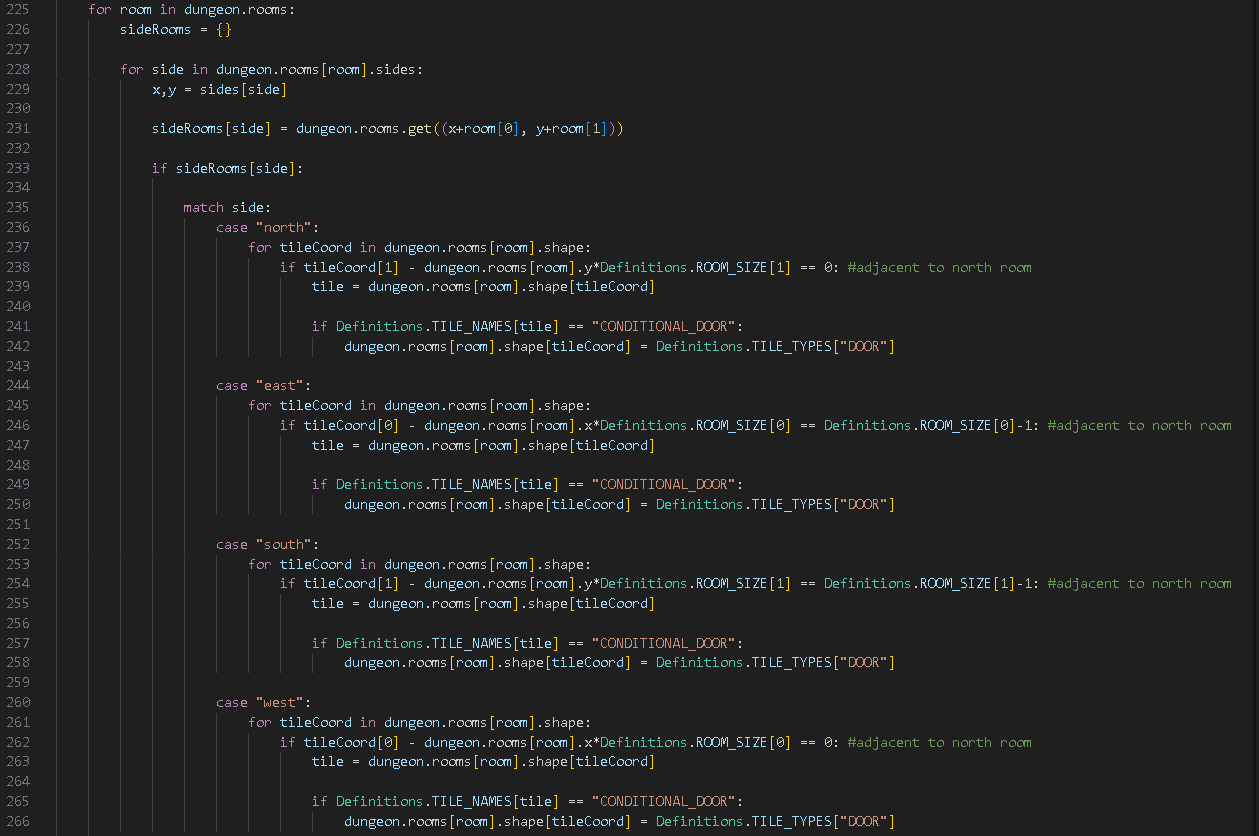
#### 3rd Major If



#### Generate Room Objects

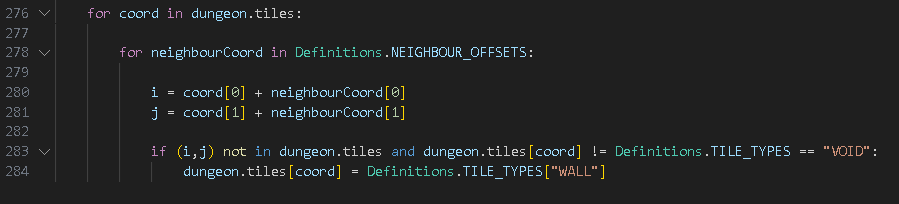


#### Fill In Door Tiles

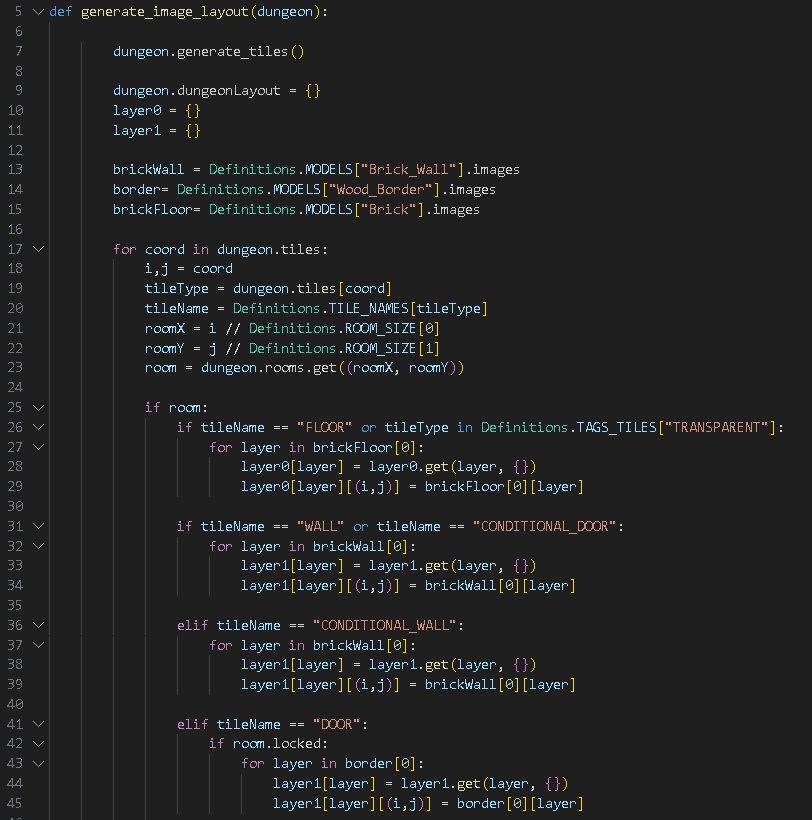


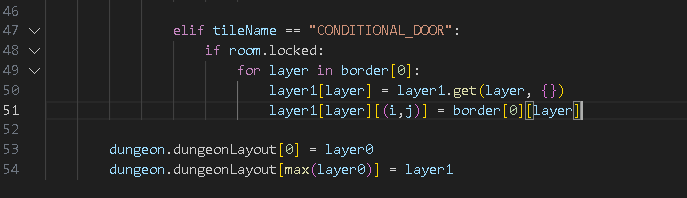


#### Block Out Void



### Generate Image Layout



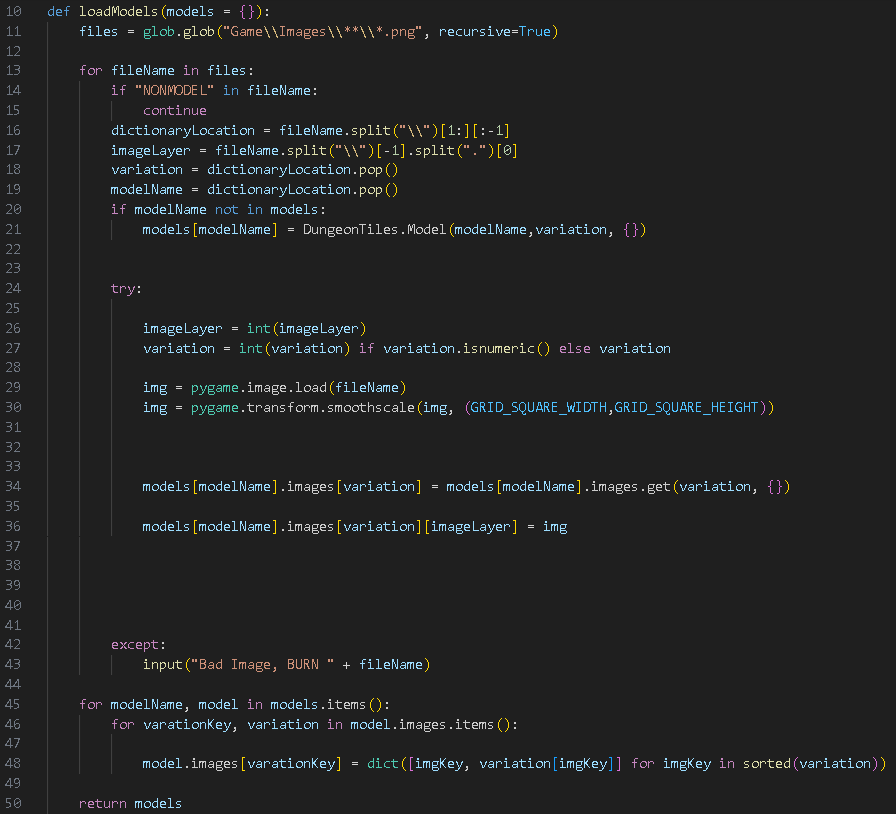


## Evidence of Dungeon Definitions

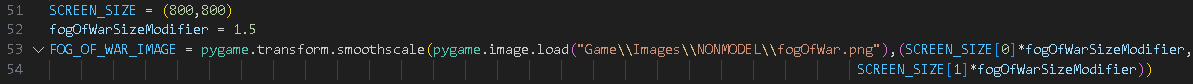
### Grid Square Sizes



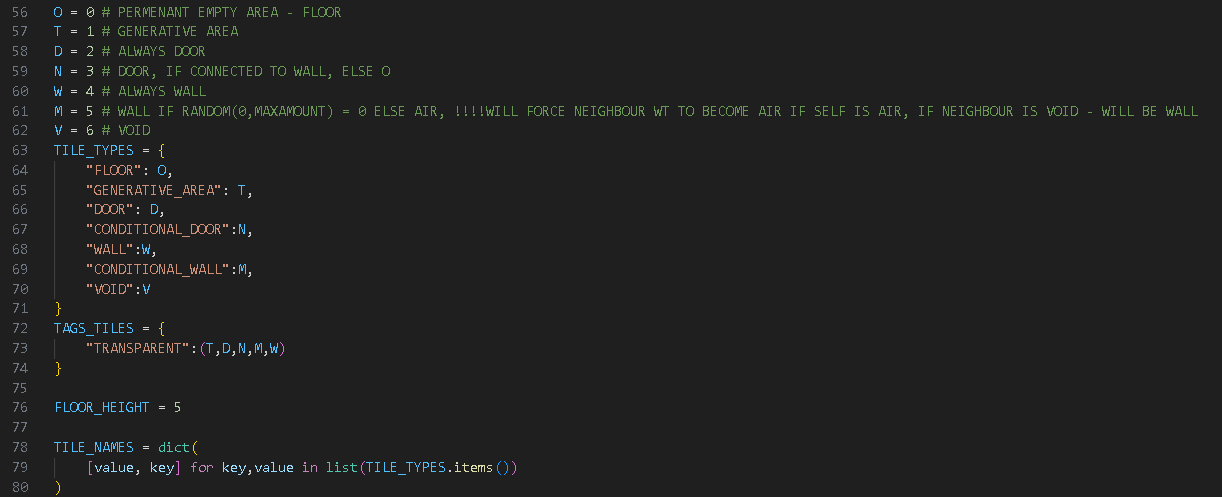
### Load Models Function



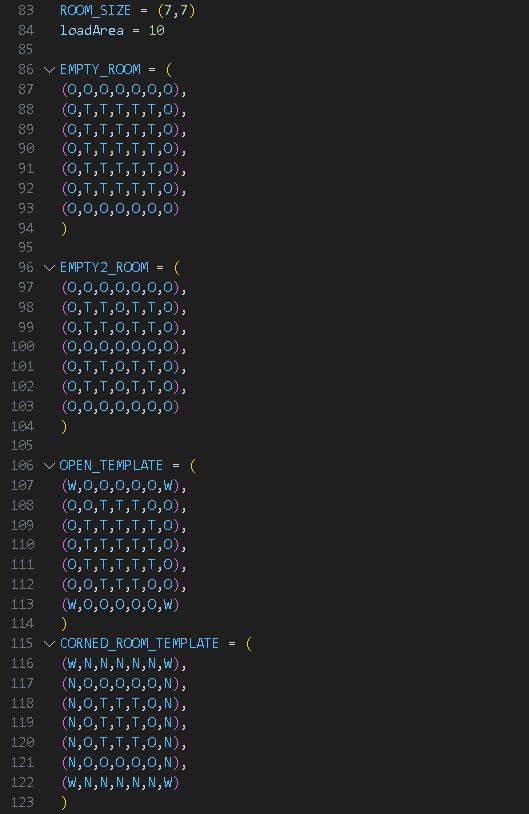
### Screen Size + Fog Of War (Related-ish)

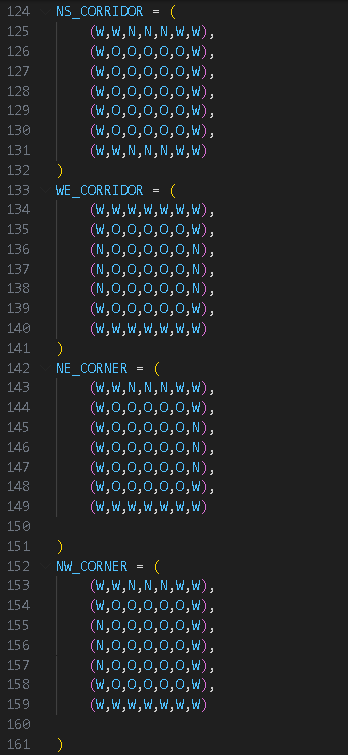


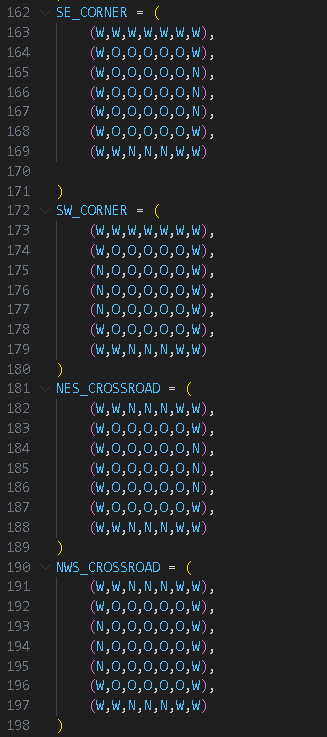
### Tile Information

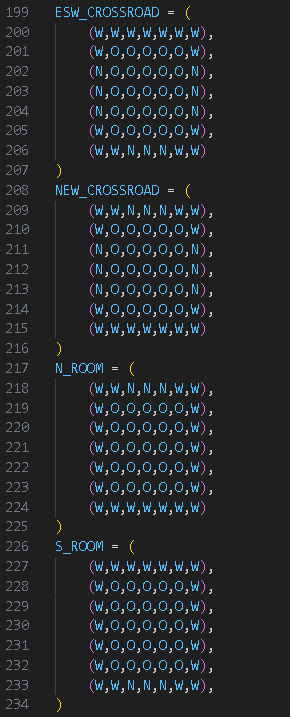
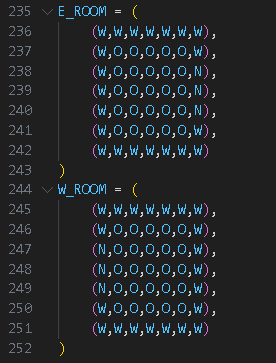


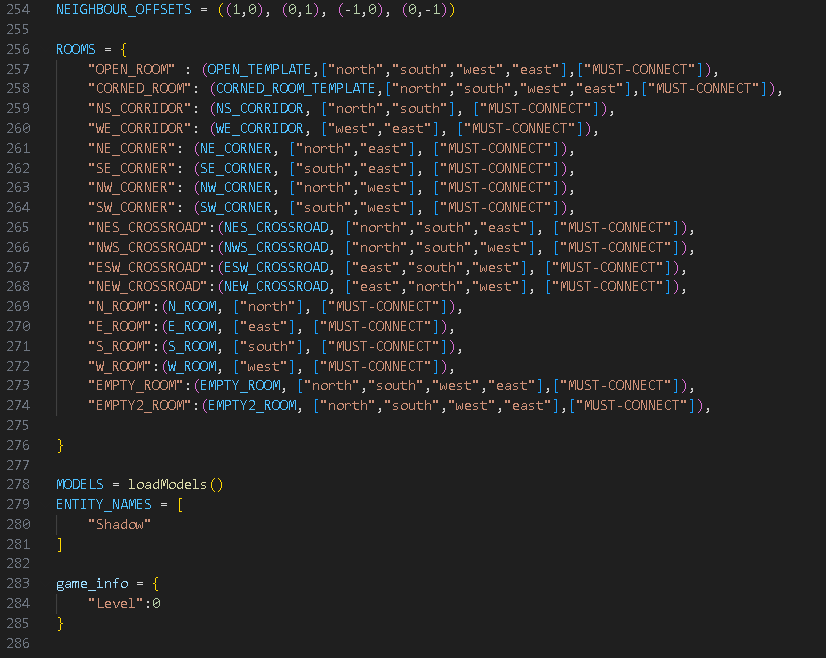
### Rooms









## Evidence of entity implementation

### Construction methods implementation

A computer screen shot of text

Description automatically generated

## Evidence of AI implementation

### Construction Method Implementation

A screen shot of a computer program

Description automatically generated

### Room Property

A computer code on a black background

Description automatically generated

### Movement

A screenshot of a computer program

Description automatically generated

### Targets

A black screen with white text

Description automatically generated

A screenshot of a computer program

Description automatically generated

### Distance to player

A screen shot of a computer program

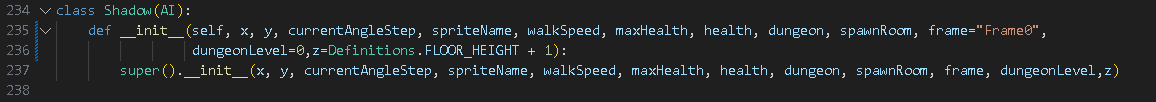
Description automatically generated

### Update Method

A screenshot of a computer program

Description automatically generated

### Evidence of shadow implementation



# D. Evaluation

## General

Throughout my project, there are many things I wish I could change. Such as spending less time on the Overlay system, I spent most of my time developing the overlay system and fleshing it out to be a standalone module that I wish I had spent more of working on the actual gameplay. Due to this misplacement of time, I had ended up not having enough time to make the game itself and lost a lot of motivation.

However, due to this project, I do have an overlay system that will work for any future projects in my future, so I am thankful for that.

I also have some regret for not using a game development framework such as Unity or Godot, my original idea was to make my project without using software such as Unity so I could have more space and develop my own framework. However, due to this, it ended taking me much longer to program.

Of the main features I wanted…

“My game needs to have a main menu screen through some sort of overlay feature.”:

* I did this, I implemented a main menu with an overlay module I developed
* It doesn’t however currently interface with the game itself

“This overlay feature needed to be able to be used across the entire game to provide UI for the player and for starting the game.”:

* I would say im not fully sure on what to rank this
* At this point of the project I do have a test where UI was developed to show information for the game screen, however it wasn’t merged yet
* But the overlay feature is able to be used across the game.

“The game also needs to have dungeon generation, this feature is needed in order to generate the rooms the player will be able to traverse.”:

* Dungeon generation was mostly completed, the layout of the dungeon is generated and so are the tiles of the dungeon, however I wasn’t able to add in procedurally generated tiles.

“The game will need its enemies for the player to fight in order to provide an engaging experience.”:

* I would say this condition wasn’t met. Enemies do appear on screen, however they do not use A\* path finding, instead they teleport
* They also cannot fight, there also isn’t a player object to fight.

“I also want it to have good lighting whether through ray-casting or some other method, this light will be able to cast shadows which could be used to sneak around as an extra mechanic…”:

* This wasn’t met, as of now, there is no lighting, other then a simple fog of war image that gets drawn over the rendered dungeon.

I was planning originally to be able to make a puzzle-solving dungeon crawler game. However, as of now, my project consists of a interactive overlays module, and a generative dungeon.

## Testing

I only had some tests to perform at the end, such as framerates and functionality.

In order to perform these tests I made this file 

This file contains code that will use the dungeon renderer to render the game screen and set the windows name to the framerate of the game.

### Passive Framerate Test

This is a singular frame of the test,



As you can see, the title of the window says around 30, this means that the game is running around 30 fps, I have analysed this number and it seems to fluctuate between 28 and 32 fps, so averaging 30fps.

This result in my opinion, is not up to the standard I wanted, however I believe this low framerate to be related to the renderer. This belief comes from the testing of entities and dungeon updates. These updates show that these methods/threads run extremely fast compared to the renderer, this may be accredited to that a lot of time is waster redrawing pixels. This is due to the innefficieny of sprite stacking, by stacking sprites I am effectively redrawing layers over already drawn layers, in order to achieve a 3d effect, this is inefficient because most of the old layers will get redrawn with only outside pixels remaining. Without this, the game would achieve much faster speeds.

A screenshot of a computer

Description automatically generated

As you may see, when the layers are not being drawn, we can effectively quadruple the frame rate, this shows that the game is running slower due to the major inefficencies of the sprite stacking algorithm.

If I were to redo my project, I would maybe choose a different algorithim or spend more time on optimising the current.

### Moving Framerate Test

I needed to also test the framerate when the camera is moving. This is important because when the player is moving, we don’t want to see a decrease in framerate, or at least a noticeable one.

A screenshot of a video game

Description automatically generated

In this test I made a bigger dungeon to explore, the bigger dungeon with between 150 and 300 rooms had no noticeable effect, I did this so more open areas would generate so I could move quickly without worry of navigating the dungeon. Therefore, assuming the effect is negligible, when moving the camera through the rooms as fast as I could, I hit a minimum of 17fps and a maximum of around 24. So, on average, moving has an effect of -10 frames per second on the passive average.

This decrease is unwanted, but it is also linked to the inefficiency of the sprite stacking system. Every time the camera moves, I have to re-render the layers, this re-rendering means collecting the tiles for every step in height and rendering them in the correct layers and place on screen. This could be made more efficient by only collecting the new tiles that are moving onto screen and just shifting the rest and cropping the layers. However, while in development I did try this optimization, but it proved difficult due to how the system already works.

If I had more time, fixing the performance issues of the re-rendering of layers would be in high priority since the player would notice the relatively large amount of fps change when moving.

### Testing Conclusion

The game through testing, doesn’t fully reach my expectations, but I am still impressed with how far I made it, considering python is quite slow especially with looping, and sprite stacking is a very loopy algorithm. So while it might not be the best algorithm for python, I am proud that I was able to get it to the point it is currently at.

## Stakeholders

In the final stages of my project, I don’t believe my game meets the full expectations I had for it to be played by the stakeholders, this is further backed by testing which revealed sub-optimal results in performance. I said near the beginning, I believe I wanted my game to be replayable and look good, unfortunately, I may have only fully achieved the latter.

I was able to achieve the looks I wanted for my game, by using smooth resizing of sprites when loading, I was able to achieve a sort of outlining on the barriers that wasn’t originally planned, this outline however makes the objects stand out more and introduces contrast. So, I quickly made this a feature.

I managed to fully get sprite stacking to work in order to achieve a semi 3D effect like I desired. However, the game as of now, isn’t playable. This is due to a lot of focus on making the UI module and a lot of focus on dungeon generation/optimisation, and not enough time on being able to play the game. I believe this may have some connection to the lack of dedicated software to making games that was used and instead coding in python with the pygame module. Due to my method, the game world and player are lot more connected then how they would be in an engine such as Godot, in Godot the player could be developed separately and at any time, same as the dungeon. However, with my structure, I needed the dungeon to be complete before I could start work on the player. If the dungeon wasn’t done, then the player couldn’t be started because then I wouldn’t have an environment to test the player in, but also because a lot of the player features resolve around the dungeon.

This is a negative towards the stakeholders because it means there was no player object at the end of the project meaning that the stakeholders don’t have a “middle man” to play through in order to interact with the game world. Instead what is currently made is a camera that lets you move through the dungeon and observe but not interact.

I made the camera in order to debug dungeon generation and try to get dungeon rendered to the screen. It was not designed to be used by stakeholders.