OCR GCE A

COMPUTER SCIENCE PROJECT

H446-03

Name: Riley Stevenson

Candidate Number: 7509

Tuxford Academy: 28148

Title of Project: UNDECIDED

H446-03 – Project CONTENTS

Table of Contents

[A. Analysis 3](#_Toc452555018)

[B. Design 8](#_Toc452555019)

[Systems diagram 9](#_Toc452555020)

[C. Developing the coded solution (“The development story”) 9](#_Toc452555021)

[D. Evaluation 9](#_Toc452555022)

[Project Appendixes 10](#_Toc452555023)

# 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.

For my game I 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.

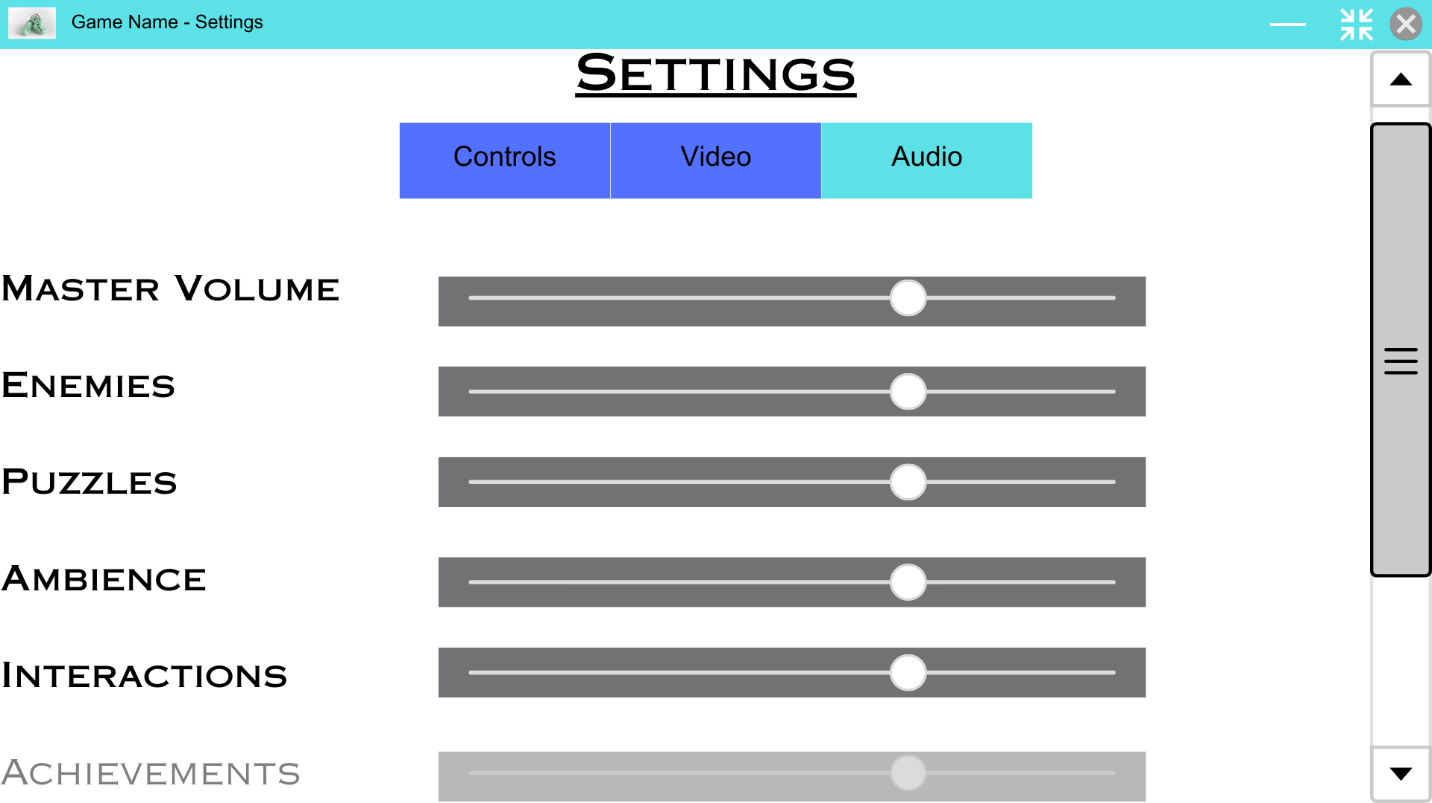
When I am designing the assets, it may look different to the current setup as well as the other designs shown here, because I am not entirely sure on this design at the moment

A screenshot of a video game

Description automatically generated with low confidence

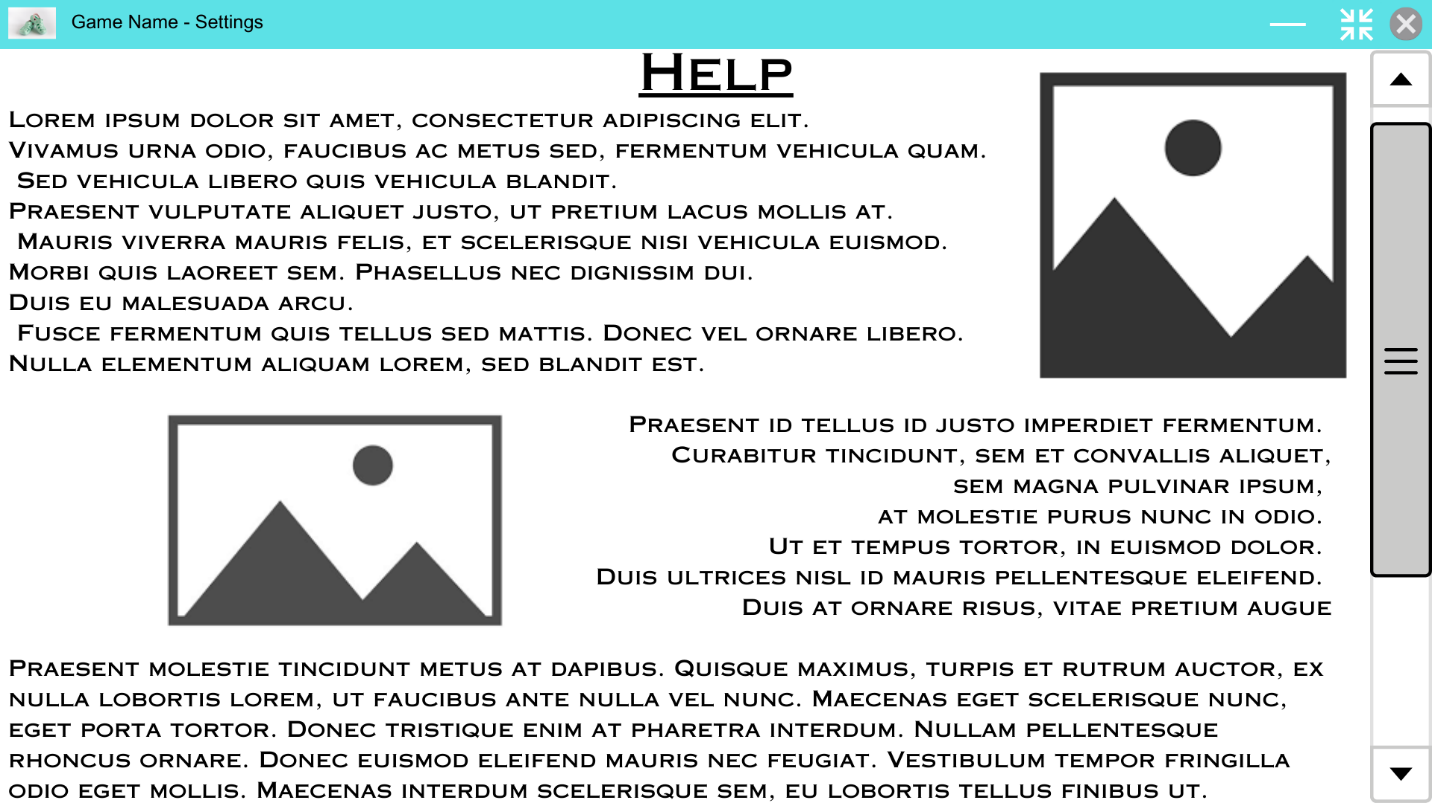
In the second section, for graphics, I want there to be a list of options with various methods to change them, such as a slider, a drop-down list, checkboxes and maybe more. These options will be focused on the graphics of the game such as limiting fps so the player can limit the CPU usage of the program.

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:

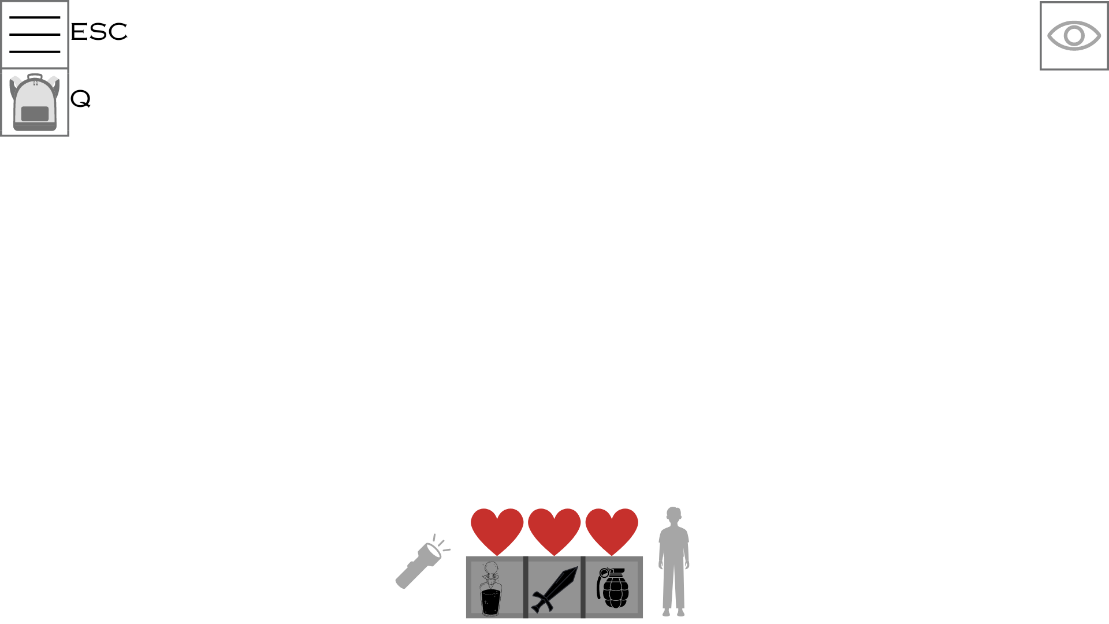
A screenshot of a game

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

# 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.

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.

### 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.

### 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.

### 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.

### 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 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.

### 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.

### 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.

### 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 using a circle class makes it much easier and flexible to use.

### 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 ad 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.

### 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

## 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

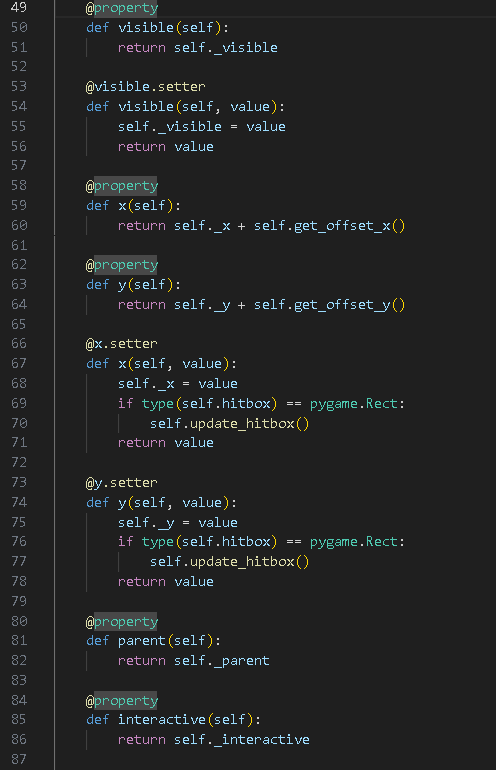
## Evidence of element implementation

### 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**

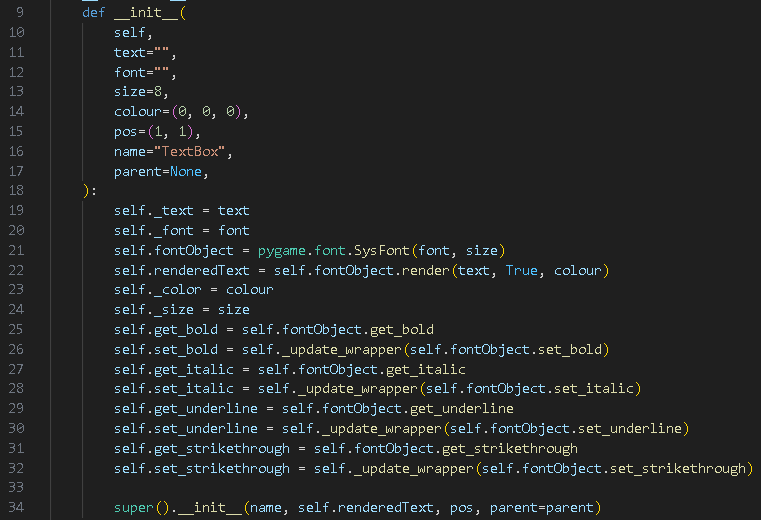
### Main Update Method

**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

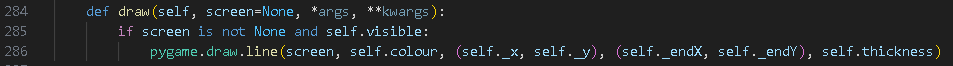
Description automatically generated

### 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

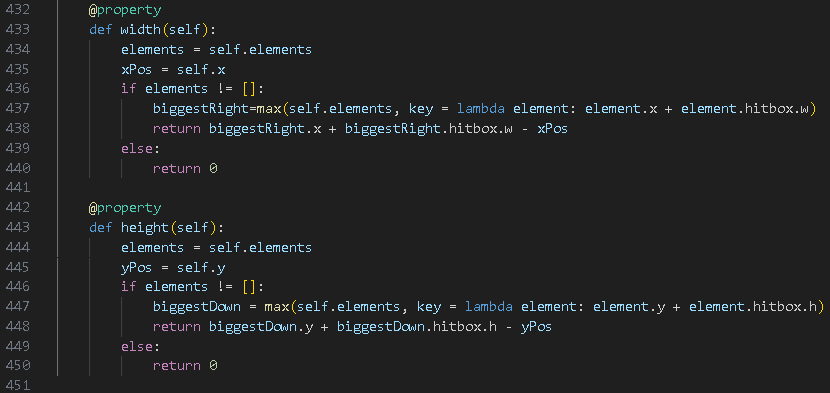
Description automatically generated

### Properties

A screenshot of a computer program

Description automatically generated





### Append Element

A screen shot of a computer code

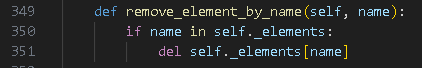
Description automatically generated

### Remove Element

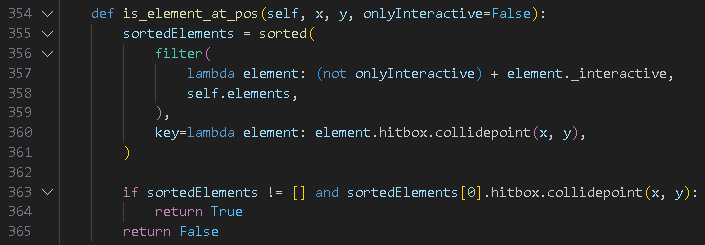
A screen shot of a computer code

Description automatically generated

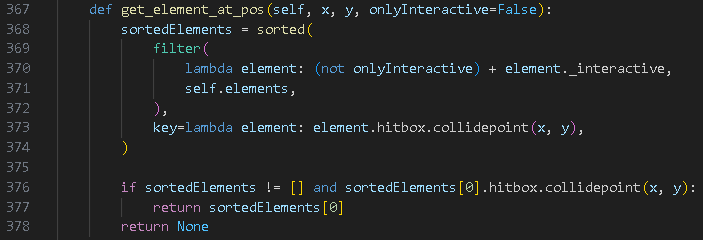
### Remove element by name



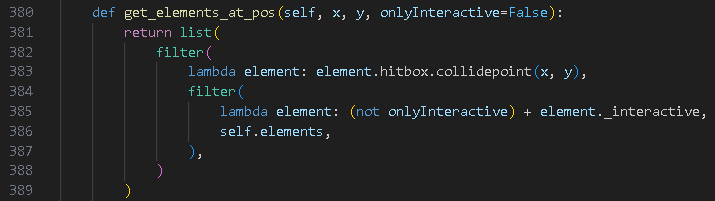
### Is element at position



### Get element at position



### Get elements at position



# D. Evaluation

<See H446-03 Project Advice Booklet for help and guidance of what must go here.>

# Project Appendixes

Insert as many project appendixes as you need for your project.

These might include, but are not limited to:

* Complete Code Listing (ESSENTIAL)
* Interview Transcripts
* Meeting notes
* Observation notes or questionnaires