a lEVEL

COMPUTER SCIENCE PROJECT

Computer Game [Shadow Slash]

Contents

[Analysis 2](#_Toc116041896)

[The Problem 2](#_Toc116041897)

[Why the program is computational? 2](#_Toc116041898)

[Stakeholder 6](#_Toc116041899)

[Research 7](#_Toc116041900)

[Game Reference/Competition 7](#_Toc116041901)

[Why combat orientated? 8](#_Toc116041902)

[Questions and Answers 11](#_Toc116041903)

[Features 11](#_Toc116041904)

[Limitations 11](#_Toc116041905)

[Requirements 12](#_Toc116041906)

[Hardware Requirements 12](#_Toc116041907)

[Software Requirements 13](#_Toc116041908)

[Success Criteria 14](#_Toc116041909)

[Design 16](#_Toc116041910)

[Solution to make it computational 16](#_Toc116041911)

[Top-Down Design 16](#_Toc116041912)

[Development methodology 30](#_Toc116041913)

[User Interface 30](#_Toc116041914)

[Usability Features 30](#_Toc116041915)

# Analysis

## The Problem

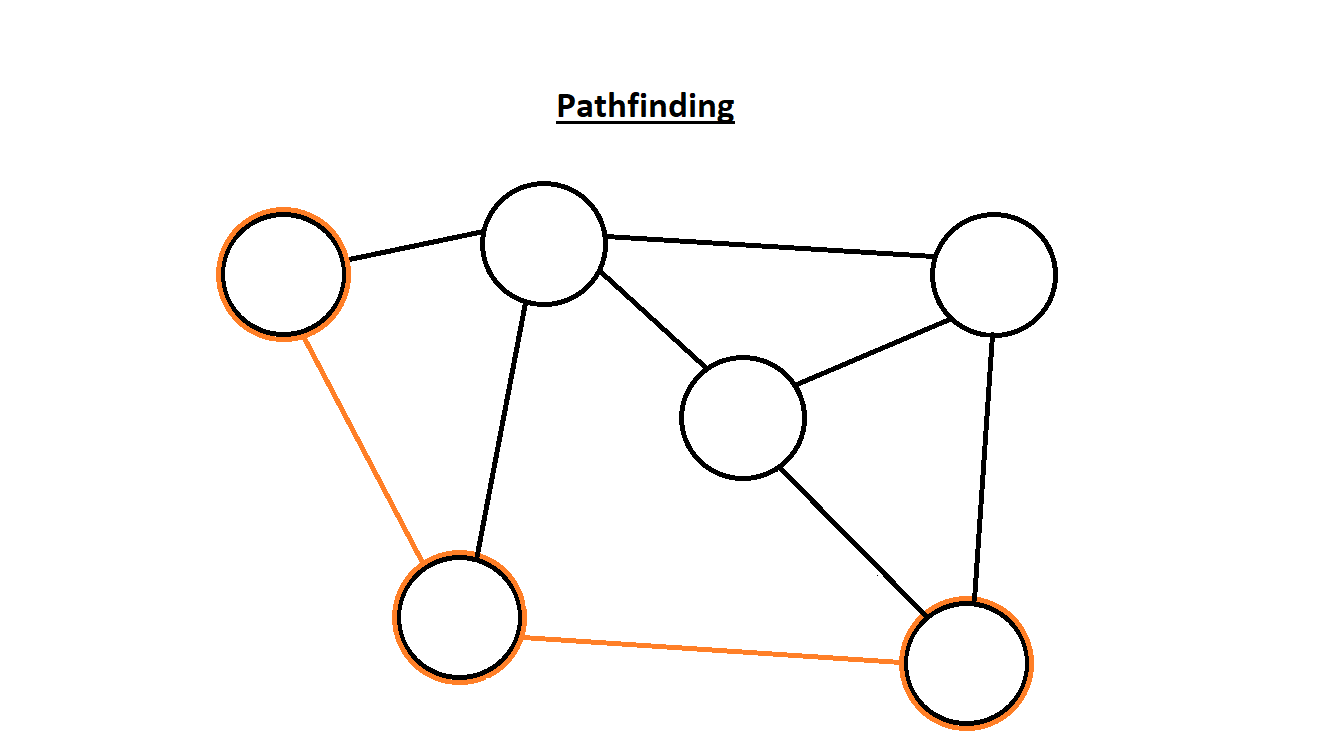
The immersion of games. The concept is given to those who are immersed in games believing that they are within the game itself. When people play games, it allows them to leave the real world. It allows people to forget their struggles and problems. In addition, games are a fantastic way to rethink and take different approaches to real-world struggles. Why? Well because games allow people to take breaks from using their brain; it reduces stress by creating a flow of sensation and satisfaction – similar to meditation. When people are stressed, they tend to take a different perspective of the situation by changing their pace of workflow which is the main reason why I am creating a game. One way to combat stress is to use combat itself. Creating a game that is fast pace and requires the player to focus, allows the concentration to be shifted away from the stressful stimuli. But why a video game? Well, video games are engaging. Unlike taking a stroll in the park, video games allow people to completely forget about stress. Taking a stroll in the park, even though for some it relieves the pressure, for many since there is not anything for the brain to be distracted with, the brain signals out the stimuli, causing the person to be possessed with the stress. The program is going to be a 2-dimensional (2D) ‘Hack and Slash’. This is where combat is emphasized with melee-based weapons. The aim of the game is for the player to defeat waves of enemies, and reach a high score. Therefore, the main purpose of the action-packed and fast pace Hack and Slash will be for the player to be immersed in the realms of the game to mitigate stress.

## Why the program is computational?

Games heavily rely on computational methods. Without computational methods, the idea of creating a game will be so overly complex, that games, in general, will not have a place in the market, let alone exist. Therefore, for many games, different methods are required in order to create a successful game.

**Decomposition (thinking procedurally):**

One way in which my Hack and Slash game utilises these methods is the use of decomposition. By breaking down the functionality, I am able to make sub-functions for the sprite, actions, camera, and many other sub-layers that conclude the foundation of the final game. One main sub function is pathfinding/pathing. This is the concept of entities finding the shortest route to the final destination. For that reason, the enemies are going to be using pathing to track the position of the player.



*(Pathing)*

The illustration [source: Pathfinding] tells us how pathing works. We know that pathing allows objects to follow the shortest route to their destination. Therefore, the illustration shows that the orange outlines are the fastest route to the destination compared to the other possible routes. However, the diagram is abstracted since there could be obstacles blocking the entities from taking different routes. With this in mind, I am going to be implementing a similar design for the enemy pathing.

|  |
| --- |
| *(Top-down design: Key Functions)* |

Decomposition generally makes solving and understanding the functions much easier. This is because it helps the programmer understand the situation and the steps required to achieve the main function. When decomposing, the top-down design is the most suitable since the stages are in a hierarchal format. The top gives an overview of the main functions while the bottom is the sub-functions that are the building blocks to create the main function. The key problems to be solved are stated through visualisation on my top-down design [source ‘Key Functions’]. Note that the diagram is a simplified and abstracted example, the more comprehensive top-down design diagrams will be stated in the analysis section.

* Controls:
  + The main control consists of using the keyboard. With the main keys being ‘W’/’Space’ (jumping), ‘A’ (moving left) and ‘D’ (moving right). With these keys, the player has the ability to move the character in any position they wish. This makes the game interactive by allowing players to be active rather than being static.
  + The use of the mouse is also crucial. Without the mouse, the player is not able to attack, resulting in the game idea being useless. This is because the main point of the game is for combat.
  + The help/instruction is an option for the player to see what controls they need to use. If the player enters the game without the knowledge of what controls to use, then the game would not be played.
* Graphics:
  + For the design element, I want to create the art aesthetically pleasing. This is because, when the visuals look beautiful it satisfies our senses and does not strain our eyes. The visuals are an important criterion to hit. Since my program is to help alleviate stress, having the visual look suffocating does not relax the mind. This will result in the problem surrounding stress not to dissipate, but elevate.
* Options:
  + Without different options, it will make the player jump into the game without any warning. Having them start the game straight away will cause some confusion since the menu screen allows the user to control the settings. The settings are important as permission is given to the player to change the quality of the setting. This may be due to their low-end computer. But due to the time limit, the implementation of different settings will not be added.

Furthermore, having the play button rather than kick-starting the game is important as it will allow the user to load the game at their own pace. In addition, the controls button helps players understand the main keys to be used when playing the game. Without knowing the main controls, players will be clueless about what to do. Also, the quit button will close the game. If the user decides to play the game in full screen, then the close ‘x’ button will not show. This may leave some inexperienced players confused on how to quit the game. Finally, the choice of pause menu allows the user to pause the game whenever they want.

* Sound:
  + Sound is important for stress. The experience of calm sounds helps ease the player's stress. Music that sounds appealing to the ear can cause many health benefits. When we are healthy, the amount of stress accumulated (if stress had an arbitrary number) falls to near 0. The implementation of satisfying music can help players mollify their stress. This is because of the power music does to our minds. Depending on the tempo of the sound, it can change our pulses, which mends our minds together.

**General Methods (e.g., Calculations):**

Another computational method that is going to be used is calculations. Without calculations, players will be incapable of doing any basic movement. Additionally, calculations are going to be a vital implementation of the game, since it is heavily relied upon calculating the damage done and taken. The use of calculations will make the health easier to develop, the score the user receives and the movement speed.

Furthermore, the use of input recognition is important. Without the use of inputs, the game will not operate. Inputs are crucial for my game since it is needed for certain actions. For example, the use of the mouse click is vital since it is used to attack enemies. If the left click had no input, there will not be any responds and the game will be a still image, where the user can only watch while the enemies depletes their life. Also, the use of the keyboard allows the user to move. Without these input responses, the user is not able to move around the game.

**Visualisation:**

The use of visualisation is important. Since it is easier to understand the aim of the problem through illustrations. It helps break down and understand complex instructions in a more simplistic format. This is because the use of visual aspects aids people to solve information quicker and more efficiently rather than reading a passage. It has been scientifically proven, that many people who see problems through images, graphs, flow charts, etc solve dilemmas much quicker. However, the use of visualisation is not only to solve problems. Another technique used is analysing and summarising concepts. This is a key feature of visualisation since it can outline unintelligible problems straightforwardly. The way I am going to be implementing the visualisation aspect is by graphically representing large chunks of information. This can be seen in the design section where I break down the main functionality into sub-functions to assist me in understanding the aim of the problem and the required steps to follow to solve the issues.

|  |
| --- |
| *(Design Concept)* |

The [source: ‘Design Concept’] diagram is an example of how users can visualise the layout of the game. Without saying a word, users can understand what is going on and what the game is going to look like. Here, we can see that the health bar for the player is on the top left. The player is represented as a yellow star, while the enemies are pink bolts. There is also going to be a ground layer and a backdrop. With only showing images, people’s brains decode concepts easier and faster than if it was text based.

**Backtracking:**

Backtracking is the use of multiple paths. The use of algorithms to solve problems does not always succeed. Therefore, when the program does not run, we can return to our previous path in order to try a different approach. For my program, the use of backtracking is going to be crucial. This is because, even though I know the different functions required to create the game, I do not know how to construct the code together without a few trials and errors. Therefore, the use of backtracking allows me to return to my previous path of code if the newly produced code does not work.

**Abstraction:**

On top of the other computational methods, the use of abstractions is going to be essential for the game. Without separating ideas from reality, the game will get overly complex. The main reason is that having a complex concept of the game will cause; firstly, the programmer unable to understand what they are coding and secondly, the player unable to understand the concept of the game. Without a good vision of what the game is supposed to do in its most basic form, just the thought of playing it will become useless. By ignoring the unimportant detail and focusing on the most important parts, it will allow me to scope out the functionality and key sections of the game. For that reason, I am going to abstract the ideas of my game from reality. One way that I am going to use abstraction is through games. By allowing the player to immerse themselves within a combat game, rather than in real-life combat, allows physical punches to change into virtual forms. Resulting in abstraction being used. Additionally, I want the player to gain experience in other ‘worldly’ situations. Even though neither I nor them have not entered other realms, in games, I do not have to think about realism. Just leaving it up to the game master helps you go through a new adventure and opens new chapters in your life. This all links to my game as I want everyone who plays to be immersed in my game. I want them to be submerged in my game so their stresses can be relieved.

Furthermore, the use of abstraction is also going to be used for the design element. By abstracting the design, we essentially make the game out of basic shapes. This is powerful since it helps us clearly understand the variables that needs to be added and the visuals of the project. If we decided to design the visuals without any abstractions, then time is could be wasted if changes are required. Additionally, without a clear thought of the design, then designers will be confused about what the visuals are supposed to look like. At the [source: ‘Design Concept’] diagram, I have broken down how the game is going to look in its simplest form and only added the essential features the game is going to have. Once those main features are added, then I am going to improve on them and make them look pleasing. In addition, I will also add some features that have not been added to the image. This is because the additional features are less important than the main features being displayed.

## Stakeholder

The target audience for this program is anyone who is stressed. The game's purpose is to ease the stress whereby it allows the player to think of alternative methods to their real-world issues. However, when thinking of some individual target audiences, I think teenagers are vulnerable to stress. Therefore, are the most suitable option for targets.

Teenagers are in the most important stages of their lives. From developing physically to taking important exams that their future is dependent on. When teenagers grow up, they go through a phase called puberty. This is the process of transformation when a child's body develops into an adult body. During this time, the human changes physically and mentally. This is stressful since it can increase stress-related dysfunctions[[1]](#footnote-1). Moreover, the addition of exams causes stress as it is an important qualification for their future. All these important life struggles colliding with each other, it causes teenagers to go through a period of their lives of agonising stress.

|  |  |  |  |
| --- | --- | --- | --- |
| Stakeholder | Role | Interaction | Availability |
| Jahangir | University Student | - Functionality ideas  - Additional Feature to add | Weekly |
| Ray | UBS Employee | - Adding depth: different viable options the game offers | Weekly |

When speaking to Jahangir about my plans, he decided that health that regenerates would be important. He told me that having regeneration lets the player play for longer periods of time. Without regeneration, player might die quicker and will not have much play time before they die. This is an important feature to add since stress will increase with frustration. If lower skilled players decide to play, their dodging skills might not be at the optimum level since the game is fast pace. Therefore, they will die quicker which will lead them to be frustrated as they die before they can even get started. Therefore, not adding regeneration will cause frustration, increasing stress which is not the point of my game.

## Research

### Game Reference/Competition

The Hack and Slash game, ‘Castle Crashers’ is a perfect example of what my game ideas revolve around. From the simple graphics to the smooth combat, the game is a perfect example of how many can immerse within the game and unwind. ‘Castle Crashers’ is a 2D Hack and Slash where it contains a simple story. The player has to save the princess and the king’s magic crystal. For this to happen, the player must defeat enemies with their melee-based weapon. On the image [source: ‘Castle Crasher’] we can see the orange playable knight using a melee weapon and partaking in combat. This is the main concept of Hack and Slash, where the player’s goal is to defeat enemies.

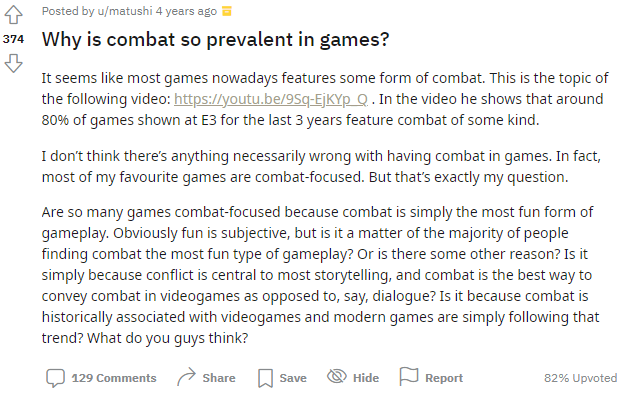
|  |
| --- |
| C:\Users\16AliMo\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\7F563D67.tmp *(Castle Crasher)* |

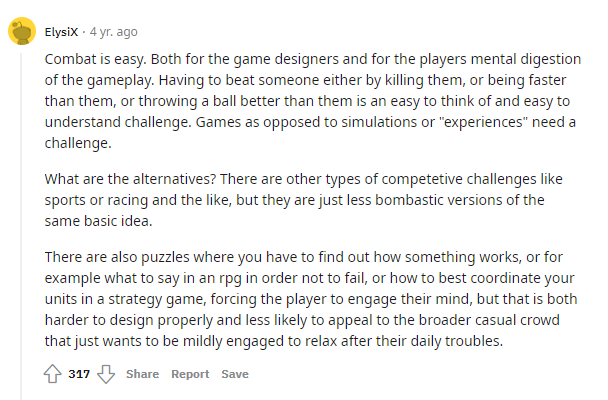
However, the problem lies within the engaging storyline. ‘Castle Crashers’ has a simple but addictive story. Therefore, the player will be immersed in the game for hours. As a result, they forget about their real-world tasks and deadlines and before they know it, they are stressed once again. When people want to destress, they want to take a brief period away to relax the brain. Usually, just taking 10 – 15 minutes to escape the real world is enough for someone to destress. Therefore, when the player starts playing games with storylines, they submerge themselves into more stress.

### Why combat orientated?

The program is exclusively going to contain only combat. Having combat as the core mechanic will divert the player's attention away from stress. This is due to the fast pace and engaging environment. A reddit[[2]](#footnote-2) user ‘ElysiX’ replied to a user ‘matushi ’questioning “Why is combat so prevalent in games? [Source: ‘matushi Question.’] Firstly, the question is asking why combat is increasing within games. He stated while watching E3 (Electronic Entertainment Expo), a gaming event for publishers to release new games. That for the consecutive years from 2015, around 80% of games contained combat. ‘ElysiX’ mentions [source: ‘ElysiX Reply’] that combat is easy for the player's mental digestion. He was implying that consuming a combat game allows a mental process that causes the clearing of a person’s stomach, reducing stress. On top of that, he mentions the use of challenge. Comparing puzzles, sports, or racing games also contains a challenge but it is the same basic idea. However, combat is thrilling as no combat is the same. From being faster than the other players/enemies to learning different control and combos to defeat enemies, all the combat is different. Combat as ‘ElysiX’ mentions is *‘mildly engaged to relax after their daily troubles.’* suggesting that combat has the power to grant people relaxation and composure. It allows someone to loosen the clouds of clustered thoughts in their head. For that reason, combat is the optimum choice when someone wants to destress.

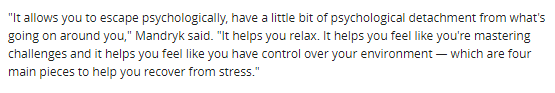
A way I am thinking why the game should be combat orientated is by seeing the enemies as the stress and the real-world problems. Having the player as the standalone character within a pixel word, gives them a satisfying reward since they slash through their problems all by themselves. By doing this themselves, it creates a sense of reward and relief. The achievement of removing a certain difficult task removes the burden, but also you feel ‘right’ when you complete something yourself.

  
 *(matushi Question)*

  
 *(ElysiX Reply)*

In addition, it has been researched that action games in general relieve stress.

During this period, the research was done during the Covid-19[[3]](#footnote-3) pandemic, this was one of the most stressful times for the entire world. One way many coped through the stressful time was through gaming. A professor at the University of Saskatchewan, Regan Mandryk[[4]](#footnote-4) said that *“gaming can help reduce stress and improve mental health.”* They looked at people from the ages of 18 to 55 where Mandryk said that playing video games can benefit people’s emotions and mental health. They gave the example of the game ‘Fortnite’[[5]](#footnote-5), on [source ‘Mandryk Quote’].



(Mandryk Quote)

By going through this passage, they mention that *“it allows you to escape psychologically.”* This is referring to the psychological damage done to the body when stressed. Therefore, when gaming, it allows someone to forget their problems and rethink different approaches to tackle their stress. Additionally, *“mastering challenges”* and *“have control over your environment”* helps individuals overcome their stress. By breaking this down, we can see stress is a problem humans face when they have a challenge that is too difficult to comprehend. Therefore, when they play a simple combat game, it helps them feel like they have control of their surroundings.

## Questions and Answers

## Features

Adding various features to the program allows ease of accessibility. Firstly, without features, a program might find it harder to stand out from the rest of the competition. This will cause many problems as money, time and effort would have been wasted on the project. Secondly, features make it easier to use the program. For example, having a game a login screen makes it easier for the user to log in straight away and access the game. That being said, my program is also going to contain many key features.

* GUI (Graphical User Interface):
  + A GUI is an essential feature for modern games. GUIs are not only used for games, but are used everywhere else. This is because GUI allows users to interact with the software by either clicking or tapping the screen. It helps users understand what is happening in the program and what they might do to combat certain situations. Without a GUI, the other alternative is command-line interface (CL). Command-line interface is a text-based interface that can also be used to run programs, interact with the computer, etc. However, this is much harder to decipher since the use of text rather than graphics is difficult to understand. For that reason, I am going to be incorporating GUI into my game. This is because games require GUIs to play and make the user recognise what the tasks are. For example, it will be easier to see combat, health, death, game over, and many other systems rather than seeing them as text. In addition, GUI will relieve stress than using command-line interface since GUI are easier to interpret. Command-line interface will just increase stress since reading massive amounts of text on the screen is not thrilling. But, the use of GUI allows the user to immerse themselves easily into the game since it is easier to interpret what is going on.

## Limitations

I think that the main limitation is the time constraint to create different maps, characters, and enemy designs. Pixel art is the main art style for my game. In pixel art, many individual pixels need to be coloured. In addition, the canvas size is 1280x720, and it takes a long time just for a single map to be constructed. Therefore, having multiple maps will take a long time to create, especially having uniquely comprised maps that will make my games stand out. Another issue is the character and enemy designs. Firstly, just coming up with a design for a character causes problem since I would need to think of distinctive characteristics and features that make up the character. This is because when creating and designing a character I need to think about the different components of what makes them visually appealing. After settling down with a design, I would need to create the same variation of the art, but make some small adjustments to the sprite for the different actions. Therefore, creating multiple characters and enemies will require a lot of time and effort to create them.

As I mentioned before, sprites are game assets that represent images. Each character and enemy are made from sprites. Therefore, when creating just a single character or enemy, there are going to be many sprites just for movement. This is the main reason I am making a 2D game since the only sprites that I am required to create is the left and right movement rather than up and down. Having multiple sprites does not only count for movement, it also counts for the attacks, jumping, taking damage and death. Therefore, it requires a lot of different sprites that comprise the final character/enemy. For that reason, the animation of the game is going to be hard to create due to the time needed to develop. Which is the main limitation since I lack the time required to pull off many designs.

However, a limitation to adding multiple maps, different sprites for each character and enemy, and various animations, is the amount of memory required to store the assets. If I do decide to add distinctive abilities for each character and enemy, it is also going to take more memory space since I will have to implement different sprites and animations. In addition, having all the unique designs and maps will require different sounds for each of them. This is a problem since the amount of memory required to store each image does not require an extensive number of bytes, however, having multiple of them is. In addition, the audio files are going to take the most amount of storage because there are going to be multiple. Not to mention, if a user has a low-end computer, it can cause lag, which is frame rate dropped, causing more frustration for the player. This would not reduce their stress.

## Requirements

### Hardware Requirements

Since this is a computer game, the user requires certain peripherals and software to play the game. Here are the hardware and software requirements and justification

* A monitor is required to provide a display of the visuals since the program is going to make use of GUI (graphical user interface).
* The keyboard is required for the player to move around. Using the ‘A’ (left movement), ‘D’ (right movement), and ‘Space’ (for jump).
* The mouse is needed to attack enemies.
* Speakers or headphones are needed since the game is going to output sound.

The internal hardware does not have to be complicated;

* it requires a basic GPU (Graphic Processing Unit) for the game to run. This is because we need a GPU to process information that is graphical.
* The game requires at least 1GB of RAM to play. Without it, poor performance and a loss of frame rates can occur.
* VRAM (video RAM) is required since I am creating a 2D game. The use of a graphical interface allows a smooth execution of the game.
* Having HDD or SSD is required to store sprites, images, animations, and sound since we need it to handle data.

In general, having a decent computer will allow this game to run without issues.

### Software Requirements

* The user is required to install the latest version python, since the entire program will be running on the python language.
* Pygame is essential for the user to install to their computer. Pygame is a python module that allows ease of creating games such as my game.
* Using the same up-to-date operating system (OS) is also required since the game will only operate on Windows.
* The user should have the appropriate drivers to play the game: keyboard driver, mouse driver, speaker driver, graphical driver

**Stakeholder input for requirements:**

**Main Requirements**

|  |  |  |  |
| --- | --- | --- | --- |
| Number | Requirements | Solves | Justification |
| 1 | Menu Screen | Displays options for the player and allows the player to start when they like | Helps the user understand what they can do when entering the game. |
| 2 | Play button | Lets the user enter the game to play | The player may not be ready, just starting the game without any preparations can cause problems |
| 3 | Controls button | Lets the user to see the controls required to play the game | If the user does not know the controls, then they will not be able to play the game |
| 4 | Quit button | Lets the user quit the game | The game could be played on full screen therefore, there will not be a way to quit the game |
| 5 | Display the map when entering the game | The background of the game is required to make the game aesthetically pleasing | The visuals will help the player ease their stress since the game looks nice |
| 6 | Display the entities | Show the enemies and the character on the screen | If the enemy is not shown then damage will be taken randomly. In addition, if the character is not shown, then the user will not know what they are controlling |
| 7 | Movement | Allows the user to move in any direction and have the enemy track the players movement | If the player and enemy are static. Then the game will only be an image. Therefore, sprites to move around makes it a game |
| 8 | Animation | Having animations makes the game look good | Without the animations, the game will not look aesthetically pleasing and will look bland |
| 9 | Border | The map should have a border | The entities and map border should have a box around it so that it prevents any entity from leaving the map |
| 10 | Collision | Actions should not collide | When actions of jump and attack should not collide with each other |
| 11 | Health bar | Displays the health of the player | The user should know what health they have so that they can prevent death |
| 12 | Damage | Allows the entities to take damage | This should kill the enemies when their health drops to 0 or display game over when the player dies |
| 13 | Spawn positions | Spawn the entities | Spawn the player in the middle of the map while the enemies should spawn randomly on the map |
| 14 | Game over | Displays game over | When the players health drops to 0, it should display game over |
| 15 | Sound | Make sound for the correct actions | When certain actions are performed, the corresponding sound should be played |
| 16 | Pause | Stops the game | When the game is paused, it should stop the game and allows the user to resume playing or quit. This may be because the player is disrupted midway. |

## Success Criteria

**Essential Features**

1. Display the menu screen with ‘play’, ‘controls’ and ‘quit’ options so that users have a choice of what they want to do when entering the game
2. Correspond the appropriate inputs with different actions. When the player either moves or attacks, then the game should respond to those action. E.g., if the player presses ‘A’ it should allows the user to move left
3. When the ‘play’ button is pressed, it should allow the player to play the game
4. When the ‘controls’ button is pressed, it should display the controls
5. When the ‘quit’ button is pressed, it should quit the game
6. Display the map when entering the game, which shows that the user has entered the game
7. Display sprites for both player character and enemies, which shows that the game has begun
8. Allows the player to move left, right, jump and attack
9. Add animation for the different actions performed by the player or enemy (e.g., attack, jump, death, etc). There will be a sprite sheet with all the different sprites that make up the animations
10. The entities should not be able to leave the map. For that reason, adding a border should prevent this
11. A combination of different actions should not collide with each other (e.g., attack and death). This is because having death as well as attacking will cause a problem since the entity should have died
12. Display the health bar for the player to know when they might lose
13. Reduce health of enemy or player when taking damage
14. Enemies should spawn at random positions; this makes it unique and helps the player not be bombarded with enemies. In addition, it reduces the likelihood of entity cramming, where there are too many entities in an area, causing lag. I can prevent this by limiting the amount of spawns. I am going to do this by having a new enemy spawn once another enemy dies
15. When health reaches 0 for the enemy they should die and a new one should spawn
16. When the player dies, it should display the game over and return to the main menu
17. Play sound according to the action, so the game will not sound empty
18. Allow the user to pause displaying the ‘resume’ and ‘quit’ options

**Advanced Features (Low priority + cuttable)**

1. Restore health when lost after a certain time. When the player loses health, it should return back to full if they have not taken damage after a while
2. Having different playable characters. This helps the game be more enjoyable as there are many different options
3. Add different styles of enemies with unique animations, as it can make the game more worthwhile
4. Having different playable maps for variation as the back drop can get repetitive
5. Implementing a high score function that counts scores for number of kills, which can make the game more competitive
6. Add another option which represent the high score of players
7. Add special abilities (e.g., pressing the ‘E’ keys allows different moves to show) to make combat more fun
8. Add crouching to the game as a dodging mechanic
9. Add stamina so that the player cannot spam attacks and requires the player to be more strategic
10. Give every enemy their own health bar. Similar to the player, each enemy is going to have a visual health bar, that is unique to all of them, which can show the player the health the enemies have.

# Design

## Solution to make it computational

### Top-Down Design

When describing decomposition on the analysis stage, I described how the top-down design is the most suitable method to solve problems. The main reason is the ease of understanding what the main functions are. By breaking down the main functions into smaller sub-functions, it makes coding the program easier since I know what is supposed to be added. This is known as modular programming where I am going to break down the core functions into smaller independent functions since it is easier to understand the main objectives needed to follow in order to complete the program. Therefore, when breaking down the functions into smaller and smaller sub-functions, we reach many subroutines. With these subroutines, I can understand the steps involved to code the functions to make the whole game.

In addition, the top-down design is more of a visual approach to seeing things. As I stated in the visualisation section in the analysis section, *‘Without saying a word, users can understand what is going on’*. Even though there are some words within each box, the hierarchal format shows me the steps required to complete sections of program. The use of a visualisation rather than having plain script of instructions makes it easier to understand the key functions without missing a step. For that reason, there are going to be many top-down design diagrams that will unite to make an entire main function. Down below is going to contain the many different subroutines that I am going to be needing. By giving bold headings to my designs, I am able to understand different individual functions required to do.

**[Overall Concept]**

Justification: The diagram represents the main concept of the game. The subsystems will be broken down into even more clear and in-depth functions. This will tell me what the aim and goals are to complete the program. It lets me see the essential components to the game and the steps to achieve them. These subsystems are the main system, which will be broken down into more specific and in-depth instructions. These main systems that will be broken down include (not in specific order):

* GUI
* Input
* Sound
* Animation
* Play Game
* Menus
* Enemies
* Health System

These all will be justified why they are the main systems and vital for the game.

**[User Interface]**

Justification: The user interface as mentioned in the features section part of analysis, is going to contain a GUI (Graphical User Interface). This is because it is much easier for the user to interact with the program, and easier for the user to destress since having aesthetically pleasing graphics ensures satisfaction to the player.

In addition, the GUI is going to contain colour. Having colour to a game also ensures satisfaction but additionally, a game containing colour allows the player to identify different objects. Having interactable entities and non-interactable objects as the same colour will make the game confusing since players will have to guess what is interactable. An example of this is door colours. In certain games, the use of having different shaded doors indicates that certain doors are interactable while the other ones are there for art purposes. The use of different shades makes the player know that they can interact with certain objects.

* Animations:
  + Animations are important to games. Without them, it is similar to like watching a still image. Adding animations brings life to a game, it allows people to read the character and enemies. In a more in-depth way of explaining, having characters animated brings out their personality, it expresses how the character is like and how the game might play out. For example, having a character tripping over countless of times indicates that they are clumsy and might belong in a simulation game. Moreover, adding animations adds an entertaining addition to the game’s movement. Having clean-cut movement makes the game easier to look at. Games with beautiful animations submerges the user into the game, believing that they within the game itself.
* Main menu screens:
  + Within the interface, it is going to contain the different menus screen. I will not go into detail about the menu screens, since I went in-depth about them in the [Menu] top-down design section. However, the different menus are going to be interactable through the GUI and they are going to contain colour. In addition, when navigating through the controls button in the main menu screen, it will take the player to another interface where the player will be able to see the controls.
* Display the game:
  + Display the Map: When the player presses the play button, they should be directed to the game’s GUI. From there, the first thing they should see is the map. The map will contain colour so that it looks pleasing to the eye. This is because we want a background to my game. Having a plain background does not tell a story. Having a complete background tells the player what setting they are in and who the character is. For example, having a post-apocalyptic background tells the player that they are in a world where there is no other civilisation. That the world they are in is dead with no signs of life and that they have to protect themselves from danger. ‘The immersion of games’ is the quote I mentioned in the problem section at analysis where I want the player to feel as if they have been consumed by the game. Having the player consumed means that they have sunken so far into the game, that they have forgotten their real-world problems and have overcome their stress. This is the main problem my game is solving.
  + In addition, the map is going to be containing borders. The border is not going to be visible to the player. Having visible borders will be distracting since the player will keep diverting their eyes to toward the out of place border. Therefore, having them invisible will make it easier for the player to concentrate on the game. One way I will make the player know that they have hit the border is through collision. When the player rectangle (which is a box around the player) touches the border rectangle, then the player is not able to move in that certain direction. This tells the player that there is a border and that they cannot proceed further.
* Display sprites:
  + Along with the map loading, the individual sprites should load. The first sprite that should load is the character sprite, where they will spawn right in the middle of the map. The second sprites that should appear on screen are the enemy sprites. The enemies are going to be spawning from different ends of the map. The character and enemy sprites are going to have unique animation that will me justified in the animation diagram below.

**[Inputs]**

Justification: This diagram represents the different inputs the program should recognise and execute.

* Keyboard Input
  + The keyboard is going to contain 4 keys that the user can press. When these keys are pressed, the program should correspond with the correct actions.



The image outlines the keys that is needed for play the game. The keys ‘A’, ‘D’, ‘Space’ and ‘Esc’ is outlined in yellow indicating that these keys will be used. The ‘A’ key will be used to move the sprite left. The ‘D’ key will move the sprite right while the ‘Space’ key will allow the sprite to jump. The ‘Esc’ key will allow the player to pause the game. When the player needs some time off due to problems outside of the game or they do not want to play, they can pause the game. When the game is paused, it should pop up a pause menu where the user is given the choice to either resume playing or quit the game.

* Mouse Input
  + The mouse will be needed for the player to attack. A mouse is commonly used in games since it easier to use and control. Mouse’s make it easier to respond to certain stimuli.



The image of the mouse shows a yellow outline. This yellow outline highlights the left click. When the player clicks on the left click, then the game should produce a response when the character sprite attacks. The player will know when they have attacked is when the sprite animation occurs.

**[Sound]**

Justification: Sound is an important factor to take in. Without sound in games, it will make them dull and boring. Having sound immerses someone into the virtual world. This is because sound usually tells a story. Depending on the mood of the game, the right corresponding sound or music can play, which can rush emotions into the player as if they are experiencing the characters emotions themselves. This creates a strong attachment to the game since it can provoke a connection for the player, believing that they are in the same wavelength with the game character. This can aid players into having a deep links with the character itself, allowing them to destress when playing that game again. This is ultimately the main goal for my game. Having the correct sound played when the players play so the game can destress them.

* Background music
  + Within opening the game, when the main menu screen is displayed, the music should play. This will allow the player to sync in to the game and feel the flow of the game. Once the user hears the music, it creates an immersion where the player will understand the sort of game it is and how it will turn out.
  + When the user enters the game, music should play. The music is going to be an 8-bit fast pace music it will set the players pace.
* Sprite Sound
  + When the player does certain actions or the enemy AI does an action, which are: attack, jump, take damage or die when HP (hit point) goes to 0, different sound should play that corresponds to each action. Firstly, when the AI or character attacks, it should make a slash sound where it indicates that an attack action has been done. The jump, which only the player character can do should make a ‘boing’ noise that shows that the player has jumped. The damage being taken for both player characters and enemies will sound different from each other. The sound of the damage taken will have a fade in, fade out noise. By having the sound fade in and out, is an easier way to understand that damage has been taken. Finally, the death sound should sound would be different for both player character and enemy. The different change in sound will indicate whether the player died or the enemy died.

**[Animations]**

Justification: The inclusion of animations will bring the game to life. Having animations tells the personality of the world and entities. Therefore, when creating the different animations, I want them to be unique so it brings a powerful expression to the player.

* Player animations
  + The player animation is going to be comprised into 4 main sprite sheets. These sprite sections include: idle, move, attack, jump, and death. The first animation is the idle animation. This is where the sprite it standing still as there is no user input. When the sprite is at a standstill, the sprite should bop up and down (the sprite should move up and down) at a slow pace, displaying to the player that the sprite is not moving.
  + The second animation is the attack animation. When the player clicks the left click on the mouse, the game should respond with the character sprite doing a slash animation. This is where a white streak will appear showing that the player is attacking. When the enemy is attacked, they should blink white to show that the players attack has gone through. This will indicate to the player that the hitbox of the enemy has been hit.
  + The third animation for the player is the movement. The player should be able to move left and right of the screen. When moving either left or right of the screen, a running animation should be displayed. However, there is a limit to how much the player can run left or right. This is due to the borders. When the player moves to far to the left, they will eventually hit the border. When the border is hit, the user will not be able to move left and the sprite will stand still changing to the idle animation. The same thing will happen to the right movement.
  + The next is the jump animation. When the player presses the ‘Space’ key, the character entity should respond by jumping. When the player jumps, the corresponding animation should be displayed. This is where the sprite bends the knees and then lifts of from the ground. When the sprite jumps, there will be particle effects indicating that the sprite has jumped. However, the sprite will not stay airborne forever. After a few seconds, the sprite will fall down back to the surface due to gravity. For the animation of when the character falls, the sprite will have their arm up in the air. Once the character hits the ground, then the sprite will change to the idle animation if no other input processed.
  + The final animation for the player character is the death animation. When the health integer value drops to 0, then that is an indication that the player has died. When the player dies, there will be a sprite animation. The animation will be the sprite falling, where after a second, the sprite will turn into particle flying back into the atmosphere.
* Enemy animation
  + The enemy animation is going to be made of 3 sprite sheets. Within the sprite sheet, the enemy is going to contain attack, move and death. The first animation is the attack animation. When the enemy attacks the player, there will be an attack animation. This attack animation is going to be represented by a red swipe. When the player is hit, they should blink red to indicate that the player has taken damage.
  + The next animation is the move animation. When the enemy moves, they can only move left or right. Usually, I do not have to worry about the enemies having to walk too far too the left or right, since they will be tracking the player’s entity. However, in the circumstances that they move too far to the left or right, like the player, then they will stand still since there will be a collision with the border and the enemies hitbox. The animation of the enemy moving left or right will be a semi-paced run, trying to chase to the player.
  + The final animations of the enemy sprite are the death animation. When the integer value of the enemy drops to 0, then that means the enemy has died. When the enemy dies, the animation that would be displayed them falling to the ground and turning into ashes, floating into the atmosphere until they dissipate.

**[Play Game]**

**[MENU]**

Justification: This diagram represents the different menu systems. By having these menu systems, it ensures that the user is in control of the game, allowing them to start, pause and quit the game whenever they like.

* Main Menu:
  + Firstly, when we head into the ‘Main menu screen’, we have three options to pick from. At the start, there is a ‘Play button’ this allows the user to enter the game. After, there is a ‘Controls button’ which should warp the user to another interface where it displays the keyboard and mouse controls. Finally, on our main menu screen, it should display the ‘Quit button’ where it allows the user to quit the game. These will conclude the main menu and what the user is supposed to expect.
* Pause Menu:
  + Furthermore, there is a pause menu. On the diagram, we can see that ‘Pause menu’ is defined. The use of the pause menu allows the user to pause the game at whatever point they want. When the pause menu pops up, it should display two options. Firstly, it will show the ‘Resume button’ where it allows the user to return back to the game from where they left off. Secondly, there is going to be a ‘Quit button’ where it allows the user to quit the game.

This concludes the menus the game is going to include.

**[Enemy]**

Justification: This diagram represents the enemy. This tells me how the enemy is going to be laid out and its key functionality. This will make it easier for me to see the main practicality of the enemy.

* Spawning
  + When the game starts, the enemy should spawn at either end of the map. When they spawn, only a set number of enemies can spawn. This is because having too many enemies can result in several problems. The first problem will cause the game to crash. This is because not having a set number of entities on the screen can overload the system and cause frame rate drops and lag. Eventually, the game will crash. The second reason is that even if the game did not crash, it could obstruct the player’s view. This is because if there are too many enemies swarming the player, it could cause too many enemies on the screen at the same time, resulting in the player being unable to see their character. When the enemies spawn, they will spawn at the far left and the far right. This will allow the player to adjust themselves and get ready for the combat. When an enemy dies, another enemy should spawn after that where they will spawn at either end of the map.
* Health
  + Health is an important variable for the game and enemies. When setting the health for the enemy, they will have an integer value set with them. The enemy will only lose health when the player attacks them. When the enemy is attacked and the hitbox is hit, then the enemy will take damage. When damage is taken, then the maximum integer value will lose a set amount of health. When the integer value falls to 0, the enemy dies. When the enemy dies, then we return to the spawning mechanic. The enemies will not have too much health since I want the player to defeat the enemies at a relative pace. In a more gaming term, the enemies that take a long time to defeat will be known as ‘tanks’. This means that they have an incredible amount of health and take a long time to kill. If I have enough time, I think the implementation of adding ‘tanks’ will be on the list of my priorities. This is because I think if the player progresses through the game for a long period, I think introducing stronger enemies and ‘tanks’ will add more levels of complexity. In addition, these ‘tanks’ could be bosses when the player reaches a certain distance into the game. However, as stated before, this will be a lower priority due to the time required to create the game.
* Movements
  + I went into detail about the movement of enemies in the animation diagram. To reiterate, the enemies can move left and right. However, when the enemy move left and right, they cannot move too far to the left or right. This is because of the world border. Since the map is not infinite, the entities cannot move out of bounds. Therefore, when the enemy reaches the border, it will clash with their hitboxes and stop them from proceeding in a certain direction. Another movement is the attack and death. When the enemy attacks, it should have a visual representation that the enemy is attacking. Similarly, there will be a death animation.
  + The most important movement for the enemy is tracking the player's movement. This is vital since I want the enemy AI to seem as if another human is controlling them. As soon as the player clicks the ‘play game’ button, the enemies will spawn and track the player. When tracking the player, it will move towards them in the direction the player is. When the enemy hitbox collides with the player's hitbox, then the enemy will start the attack animation. This will make the game more realistic, since in real life when someone is at a perfect distance to reach someone else, they will engage in combat. For that reason, when the enemy is at the perfect range (when the hitboxes collide) they will attack the player. However, for them to do that, they will have to track the player's movement.

**[Health system]**

Justification: The central part of the game requires a health system. Without a health system, it could result in the game just being an end-to-end fight. This will make the game boring since the only thing the player is doing is fighting and not having a main goal. When adding the health system, the player has a goal of not dying. The challenge and thrill of not having their health to 0 and trying to survive as long as possible will bring the desire and passion to defeat enemies. The player must think on the spot about the different actions they need to do to stay alive. Having the enemies have health adds a reward element. This is due to satisfaction, as defeating opponents that the player has been fighting endlessly until the enemy’s health drops to 0 is an achievement in the player’s eye. During the research section in the analysis under the ‘Why combat orientated?’ heading, I mentioned that I see the enemies as real-world problems and stress where the player aims to defeat as many of their problems. Therefore, having enemies that disappear when they defeat them brings in reward for the player. One question might be: “Then why does the player have health, if they die, will that not mean that the player has lost to their problems?”. With that, I respond by saying that: since the game is a fast pace game and has a lot of action, I think the player will be immersed too much into the game to ever think that they have lost to their problems. Allowing them to retry the game as much as possible, ensures that they will not give up. Another way of thinking about this is: losing to the enemies gives the player the passion for finding another way to think of alternative routes and solutions to solve their problems.

* Player
  + The player is going to have a set integer value for their health which will be represented at the top of the screen. When the player takes damage by getting hit by the enemies, then their health will decrease. The integer value given to the player will decrease by the amount of damage taken. Over time, if the player does not take any damage, or is taking significant lower damage, then their health will regenerate. This is where an addition will be done to the integer value, increasing it until the health reaches the maximum value. However, if too much damage is taken, the integer value will drop to 0, meaning that the player has lost. This then should display a game over screen.
* Enemy
  + Enemies also have a similar concept to the player’s health. The enemy will have a set integer value that is represented as the health. However, the difference between the player and the enemy is that the enemy is going to have a different integer value from the player. Also, the enemy is not going to have their health displayed like the player. I think exhibiting the enemy’s health will distract the player since there would be many health boxes that will cover too much of the screen. Additionally, there are going to be enemies respawning once the old enemies die, which can mean implementing new health boxes for each of them. This means that I have to remove the old boxes with the new ones but, due to the time required to create the game, it will need to be added as a low-priority/cuttable feature.

## Development methodology

The development methodology that I am going to be using is agile/iterative. If I used the waterfall methodology, then there would be a high risk. This is because, unlike the agile model, the waterfall model is not suitable for object-orientated projects. Since my code is going to make use of object-orientated code, the waterfall method is not able to handle the complexity of the code. In addition, there is a lack of user feedback. Without the feedback from my stakeholders, it could cause many faults to my game. Jahangir decided to add health that regenerates when lost. This helped me understand that players want to play the game longer without any disruptions. For that reason, having health that regenerates allow the player to play the game longer. This suggests that stakeholder feedback is vital for my game's success.

Since I am using a top-down design, many of the problems are going to be broken down into many sections. For that reason, the use of agile is going to be essential. This is because when using agile, each problem is worked through iterations. The use of making multiple iterations helps me test my ideas quickly and improve from the last one. Agile has a flexible approach since it will allow me to alter any idea or change solutions. This is important due to the time constraint. Since my program is going to be large, and since I am the only sole developer of the program, it is going to take much longer to create since I do not have a team of programmers to help me. Therefore, due to the limited amount of time required to create the game, having the flexibility and stakeholder feedback benefits me to progress through my game efficiently.

[Why appropriate than other methodology]

## User Interface

## Usability Features

1. <https://en.wikipedia.org/wiki/Puberty> (26/09/2022) [↑](#footnote-ref-1)
2. https://www.reddit.com/r/truegaming/comments/9a5sy0/why\_is\_combat\_so\_prevalent\_in\_games/ (23/09/2022) [↑](#footnote-ref-2)
3. <https://en.wikipedia.org/wiki/COVID-19_pandemic> (26/09/2022) [↑](#footnote-ref-3)
4. <https://www.cbc.ca/news/canada/saskatoon/u-of-s-research-finds-video-games-can-relieve-stress-improve-mental-health-1.5563824> (26/09/2022) [↑](#footnote-ref-4)
5. <https://fortnite.fandom.com/wiki/Fortnite_Wiki> (26/09/2022) [↑](#footnote-ref-5)