a lEVEL

COMPUTER SCIENCE PROJECT

Computer Game [Shadow Slash]

Contents

[Analysis 2](#_Toc115337040)

[The Problem 2](#_Toc115337041)

[Why the program is computational? 2](#_Toc115337042)

[Stakeholder 6](#_Toc115337043)

[Research 7](#_Toc115337044)

[‘Game Reference/Competition’ 7](#_Toc115337045)

[Why combat orientated? 8](#_Toc115337046)

[Questions and Answers 9](#_Toc115337047)

[Features 9](#_Toc115337048)

[Limitations 10](#_Toc115337049)

[Requirements 10](#_Toc115337050)

[Hardware Requirements 10](#_Toc115337051)

[Software Requirements 11](#_Toc115337052)

[Success Criteria 12](#_Toc115337053)

[Design 13](#_Toc115337054)

[Solution to make it computational 13](#_Toc115337055)

[Top-Down Design 13](#_Toc115337056)

[[Overall Concept] 14](#_Toc115337057)

[[Menus] 15](#_Toc115337058)

[[User Interface] 16](#_Toc115337059)

[[Play Game] 16](#_Toc115337060)

[[Inputs] 16](#_Toc115337061)

[[Enemy] 16](#_Toc115337062)

[[Health system] 16](#_Toc115337063)

[[Sound] 16](#_Toc115337064)

[Development methodology 17](#_Toc115337065)

# 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.

|  |
| --- |
| The illustration on the left 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 example, the actual 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’ 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’, 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. When lower skilled players die quicker, they will get frustrated as they lose health quick. 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?

For that reason, 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.

|  |  |
| --- | --- |
| C:\Users\16AliMo\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\24A26715.tmp  (matushi Question) | C:\Users\16AliMo\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\F01FFF1C.tmp  (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), where they said on source ‘Mandryk Quote’

|  |
| --- |
| C:\Users\16AliMo\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\E411AD44.tmp  (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 to play the game.

* 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

**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 actions with the different options. If they do not, then it will leave bad reviews
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
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

# 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. Since this is a large program, there are going to be many functions. 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. 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]**

This

**[Menus]**

This diagram represents the different menu systems. 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. Furthermore, there is a pause menu. On the diagram, we can see that ‘Pause menu’ is defined. The user 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. These concluded the menus the game is going to include.

**[User Interface]**

**[Play Game]**

**[Inputs]**

**[Enemy]**

**[Health system]**

**[Sound]**

## 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 allows 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.

1. <https://en.wikipedia.org/wiki/Puberty> [↑](#footnote-ref-1)
2. https://www.reddit.com/r/truegaming/comments/9a5sy0/why\_is\_combat\_so\_prevalent\_in\_games/  [↑](#footnote-ref-2)
3. https://en.wikipedia.org/wiki/COVID-19\_pandemic [↑](#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 [↑](#footnote-ref-4)
5. https://fortnite.fandom.com/wiki/Fortnite\_Wiki [↑](#footnote-ref-5)