Akibul Hoque

NEA project word document

2018/19/20

Centre number 51407

Candidate number 4380

Project Idea – 2D Adventure RPG

“Trek for Atonement”

# Contents

[Contents 2](#_Toc34899618)

[Section 1 – Analysis 4](#_Toc34899619)

[Problem definition: 4](#_Toc34899620)

[Brief 4](#_Toc34899621)

[Who is the end user, what do they do and what are their needs? 4](#_Toc34899622)

[Current system and why it is not meeting their needs (Market) 5](#_Toc34899623)

[Current system in terms of my inspiration 8](#_Toc34899624)

[Project Investigation: 10](#_Toc34899625)

[Questionnaire 10](#_Toc34899626)

[Existing solutions 17](#_Toc34899627)

[Justification of Computational Methods: 18](#_Toc34899628)

[Methodology 19](#_Toc34899629)

[Creating games 19](#_Toc34899630)

[Limitations 21](#_Toc34899631)

[End User Requirements 21](#_Toc34899632)

[Section 2 – Design 22](#_Toc34899633)

[Design objectives: 22](#_Toc34899634)

[Decomposition of Design objectives: 22](#_Toc34899635)

[Enemies & NPCs (Input): 22](#_Toc34899636)

[In-game Map Creator (Input): 23](#_Toc34899637)

[Interchangeable controls (Input & Process): 24](#_Toc34899638)

[OOP (Process): 25](#_Toc34899639)

[Saving & Loading (Process): 25](#_Toc34899640)

[Experience & Levelling (Process & Output): 25](#_Toc34899641)

[Decomposition of the whole problem: 25](#_Toc34899642)

[Gameplay decomposition: 26](#_Toc34899643)

[Graphics decomposition: 27](#_Toc34899644)

[Controls decomposition: 28](#_Toc34899645)

[Audio decomposition: 29](#_Toc34899646)

[Save/load decomposition: 30](#_Toc34899647)

[Describing the Solution 31](#_Toc34899648)

[Menus analysis and MFD 31](#_Toc34899649)

[Menu mock-ups (Input): 33](#_Toc34899650)

[Menu mock-ups (Output): 35](#_Toc34899651)

[Main Algorithms: 36](#_Toc34899652)

[Section 3 – Implementation 40](#_Toc34899653)

[Completeness of Solution: 40](#_Toc34899654)

[All Important Technical Skills Used: 43](#_Toc34899655)

[Group A skills 43](#_Toc34899656)

[Group B skills 50](#_Toc34899657)

[Research/code sources: 52](#_Toc34899658)

[Section 4 – Testing 52](#_Toc34899659)

[Video Testing Timestamps 52](#_Toc34899660)

[Manual Testing 53](#_Toc34899661)

[Section 5 – Evaluation 54](#_Toc34899662)

# Section 1 – Analysis

## Problem definition:

### Brief

My friends and a lot of people on the internet (Including myself) share their discontent with the lack of games similar to older games as recent games have moved on from them. They lack this entertainment as their desires seem to be a minority in the industry of games, so it is left up to smaller developers to fill the gap. They want a game that includes elements of:

* The action-adventure genre
* Nostalgia
  + The idea of nostalgia will be analysed later
* Good sound design
* Well thought out story

### Who is the end user, what do they do and what are their needs?

**Age**

To first answer this question, we must look at my main inspiration for my project. I have based most of my game ideas from a video game called Zelda 2 and also have some other ideas from the older versions of Pokémon games in the Pokémon franchise; what this means for me is that my game will have more of an older feel to it which would say a lot about the target demographic it would attract. Couple this with the fact that my main intention with this is to be a combination of feelings of nostalgia and also of a new experience for those who pick up the game I am to create, means that this project would mostly appeal to an older audience in terms of gaming age groups, which would seem to be old teens to young adults. This gives me an age range of about 16 to 24-year olds.

**Gender**

But this does not factor in the issue of a gender target/appeal for my project to aim for. But due to the general genres my project will inherit from my inspiration that are close to unavoidable such as action, my game would naturally tend to a more male audience as games that are action orientated tend to see a more male audience generally according to this article from newzoo.com ([LINK](https://newzoo.com/insights/articles/male-and-female-gamers-how-their-similarities-and-differences-shape-the-games-market/)) which means that my project would be orientated to more of a male audience. But according to the article, even though men are more into action games than women, they aren’t in front by too much of a difference and so that means I personally will try my best to keep my audience appeal neutral in terms of gender to appeal to a wider audience. The thing is a wider appeal means more people try the product but at the cost of not being able to specifically cater to a smaller audience meaning that it is harder to make the game higher acclaimed as it would be harder to please one without costing the other, etc. This is something I should keep in mind when designing my project.

**Experience**

Then there is the issue of how much experience my ideal user would have, the two mainly being someone quite new to gaming and/or new to the action adventure genre compared to somebody that is well accustomed to gaming and want a challenge to test their skills. But the thing is, here it is pretty easy to hit two birds with one stone as all that is needed to appeal to both target demographics is to have the game be based on a skill curve and have it appeal to both audiences. In terms of the two types of players that I just mentioned, that means that a good skill curve starts off slow and lenient to the casual players to ease them into the game and how it works, but not have the start be also unique to give the more experiences players more of a reason to continue on, and later the game must ramp up in difficulty by the right about not to dissuade the casual gamers from dropping the game but also to give the experienced players what they want, a challenge. Looking in terms of balance, the fact that there are generally more casual players over seasoned gamers that would play these kinds of games and so that gives an idea of what to aim for in terms of difficult in the design phase.

So now that we have a general idea of to whom I am aiming to create my project for, now we can form an idea of what their needs are and set a sort of criteria from that information. First of all, there are the ideas already mentioned, which include:

* Action/adventure style
* Gender neutral
* Beginner friendly to start

There are more needs the customer has but that will be covered in a point a little further down.

### Current system and why it is not meeting their needs (Market)

**Multiplayer vs Single player**

To talk about the system in terms of video games is to talk about the industry and market of video games in the present and compare it to the past state of affairs. Currently the system is that there is a large focus on largely multiplayer experiences as these seem to almost always guarantee a large monetary consistency even after initial sales start to dwindle as large producers could have a base game which is engaging but then include extensions onto gameplay to boost player progress for money. This incentivises a lot of customers even on top of any money they have spent on purchasing a title, to purchase extra player upgrades (or cosmetics) in attempt to out-do any competition from other players which tends to get out of hand the more the focus on this the developers put on this as it would segment communities of games with the players at the top of the global leader boards being filled with people who spend the most money on the game rather than the most skilled and hardworking of players which of whom could potentially spend hours on a game grinding for a level up or an item to just be out done by somebody who bought all the best items in a game. This can be seen by the fact that the majority of AAA titles now fit into that category with only a few exceptions only being from the biggest companies which just want to retain company tradition such as Nintendo by continuing loved single player franchises that only works because they are guaranteed high sale numbers purely on the fact that they are using the name of a loved franchise that customers will always want to buy, an article from forbes.com highlights this tend well ([LINK](https://www.forbes.com/sites/davidthier/2017/10/18/star-wars-visceral-yes-aaa-single-player-games-are-dying-thats-fine/#129b495468aa)).

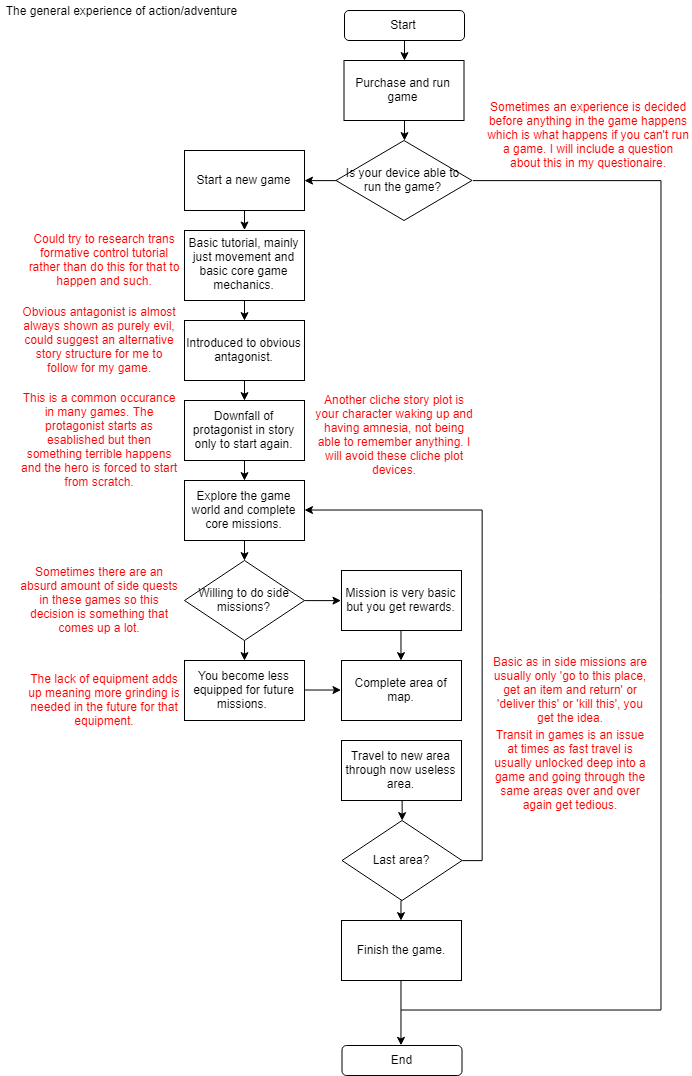
But then how does this compare to its past and how does it link into my project? Well if we look back a decade or two, we can see that this shift in focus is clearly apparent. Back then the only sort of multiplayer seemed to be local multiplayer with friends and family in the same house but that only seeming to be a gimmick in pretty much every case while the focus was on a single player experience and the really well received and well produces titles trumping sales for all other games of its time. That is how long-standing franchises started out such as Mario, The Legend of Zelda, etc. which all seem to be household names now. Whether or not the advancement in technology or other factors is the reason this shift has happened is of no relevance to this project as of yet, but rather the repercussions of the change is what is. Now a lot of single player games are up to smaller developers to create and really the only way for quality, well acclaimed single player games is to be produced by these developers is for a lot of creative talent for a game to stand out and have new, unthought of or uniquely used gimmicks which means only truly good single player experiences come out few and far in between. These small studios do not have the funds or facilities to hire large writing teams for their games unlike large producers.

Why this matters is that people that grew up in the golden age of single player experiences are now neglected in their needs for a good consistency in the rate that great single player titles are released as there are only a few AAA single player games produced.

**Accessibility**

There are issues due to accessibility of games. If a game is only multiplayer, this means that those who are not able to secure a good consistent internet connection is not able to enjoy the video game to its fullest extent. Also, if a user is not able to have a good device to properly run the new titles and are instead stuck with a low tier work computer let’s say, they won’t have access to much entertainment from that device from the limited capabilities of the system as now games are created to be played on high end platforms or gaming consoles. Which is why I want to go for a minimalistic approach in my project, so my game is runnable even on the lower tier systems.

Here is a flow chart of current games in the general genres that I am to make my game on:



No

Yes

No

Yes

No

Yes

This flowchart is to show the general formula for a lot of games of this genre. My aim out of creating this flowchart is to identify those clichés and to make sure I don’t follow these just so my game is memorable and is avoiding being stale, etc.

### Current system in terms of my inspiration

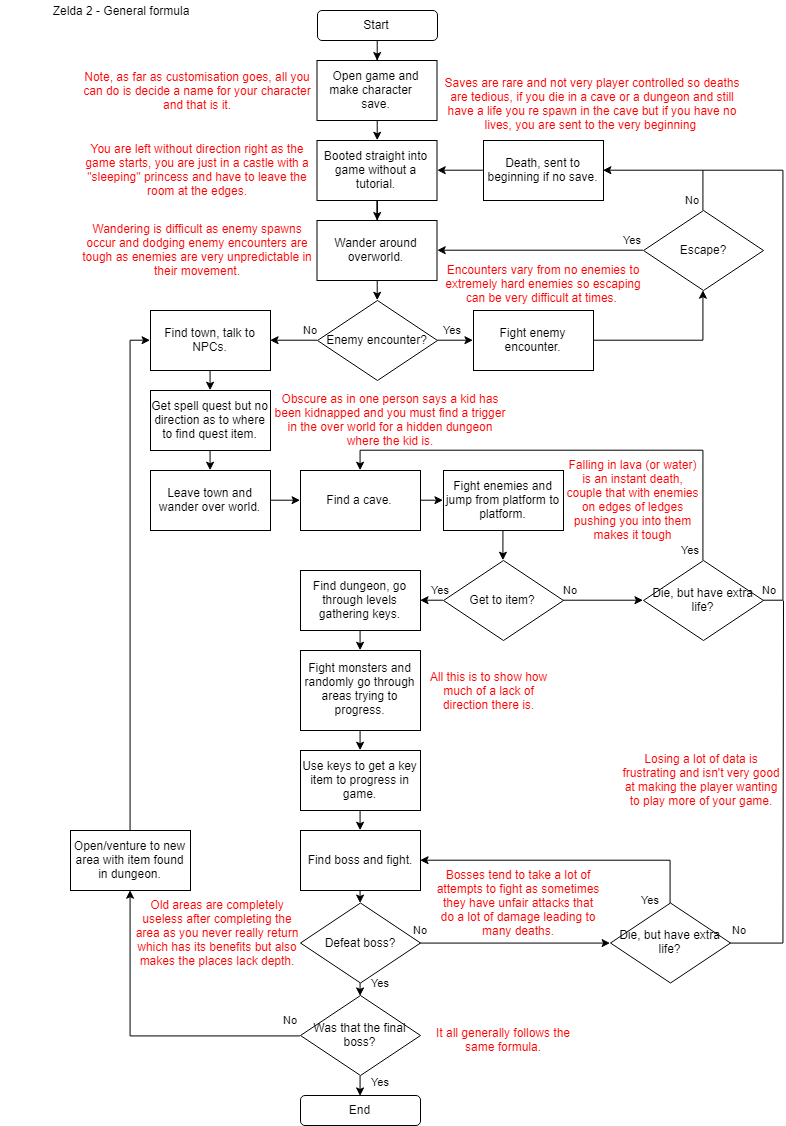
**Inspiration**

Now to talk about the game I am basing my project on. The game is called The Legend of Zelda II (2): The Adventure of Link and is a perfect blend of all the desires that my audience have asked me to base a game on mainly that being an action adventure with good nostalgia attributed to the game among a lot of the people that purchased the game a while back when it was released. The game is an action adventure where you battle through monsters in a 2D side scrolling environment and then travel through the world in an overworld 2D grid to make your way to fight an end game boss where he is attempting to receive the Tri-force to revive princess Zelda. For more details on the story, the Zelda would elaborate ([LINK](https://zelda.gamepedia.com/Zelda_II:_The_Adventure_of_Link#Story)). The game incorporates element from the RPG genre as well as adventure and action which was a unique combination at the time. There is an EXP system where you gain experience after battling monsters to level up your stats as a character such as attack damage, magic and defence.

**Problems**

There is a lot of things that the game lacked such as polish and direction as well as issues in the game such as bugs and glitches. To start let’s talk about the direction. When you press play you are booted straight into the game being told nothing about the controls or where you are meant to go or do. In fact, there is really never a clear indication in the game throughout its entirety on where to go with the closest being talking to every NPC until you get a brief sentence on what to do, which is a problem in on itself as there are many NPCs that are completely pointless and serve no relevance to the user whatsoever. A lot of the time users must look up walkthroughs on the internet for an idea on what to do which means that the people that don’t want to do that would just quit playing the game in its entirety without a very good impression of the game.

Here is a flowchart of the game in general:



The ideas we get from this flowchart is that a few things in the game that can be improved on is that the game can have some more variety and that would add to replay-ability. The lack of direction in this case was a sort of way for the company to sell more external merchandise in terms of this game as in at the time of the game release, there was not as wide spread of an internet as there is today so if people wanted to figure out what to do in a game, they had to purchase a Nintendo magazine which would showcase what you would need to do in this game. This segments the community as some people aren’t interested in those magazines. Also, another issue with the game that is highlighted in this flowchart is the difficulty of the game. The game is extremely difficult to complete and doesn’t start that easy either. These are aspects I am to take into consideration when making my project as that means I can take this inspiration and take out all the good bits and improve the negative aspects of the game to make them positives and make a generally better experience.

## Project Investigation:

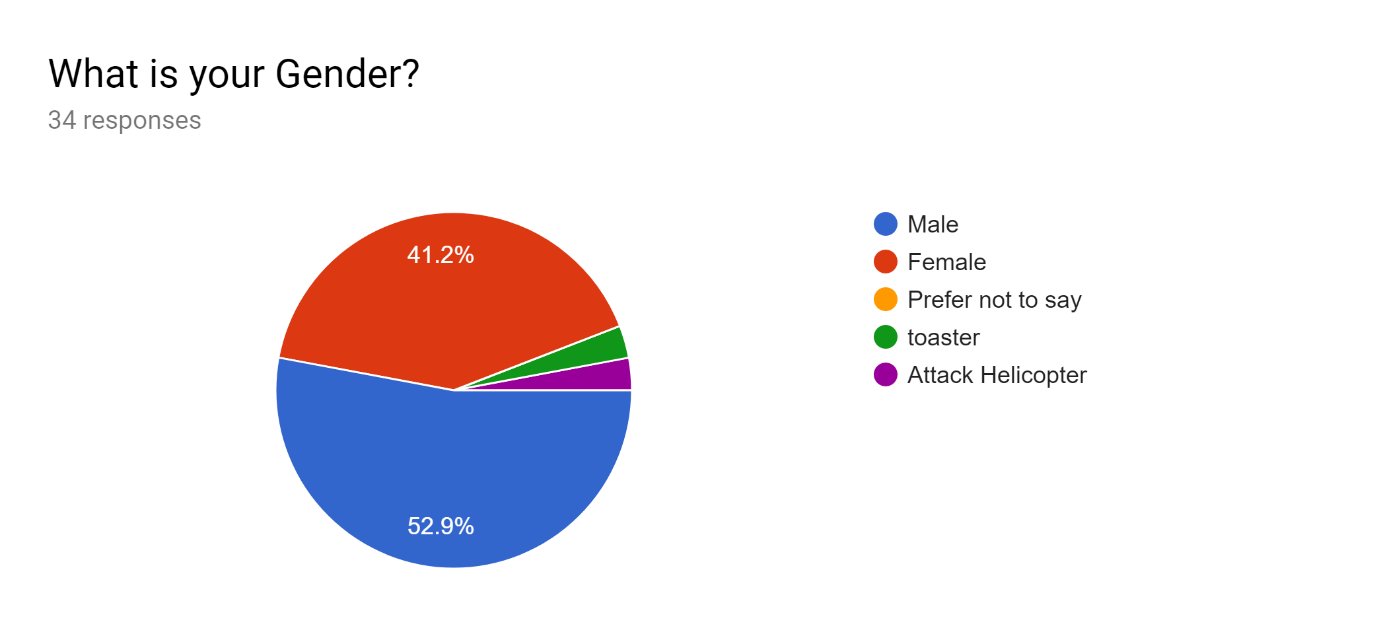
So now that we have our problem analysed on the surface, it is time to dive deeper and get some primary data of my own to see what extent all that I have analysed effects my problem and hopefully make some further analysis on the direction of my project and to see possible ways to develop the game.

### Questionnaire

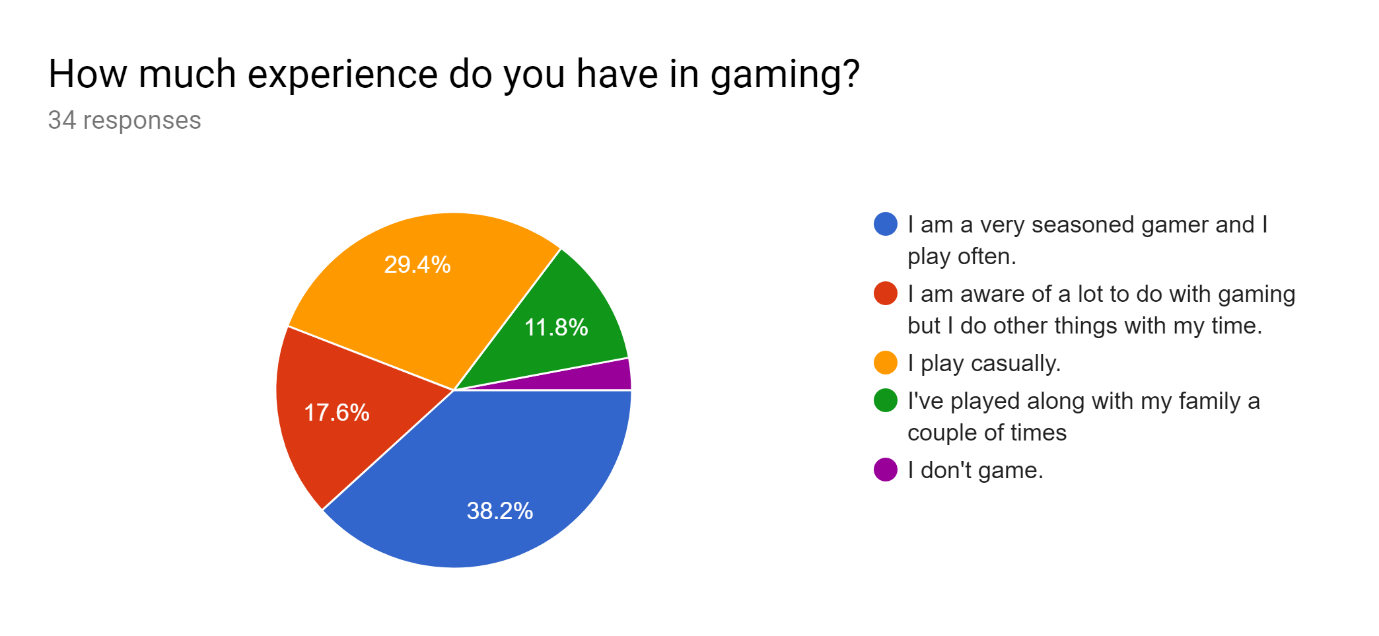
I set up a questionnaire to consult information about gaming from an audience of people in the age group that I desire for my project. Here are the results after a few days of keeping the questionnaire open (And receiving a total of 34 responses).

I split up my questions with some to do with the view of the market of gaming and preferences in terms of that and also general preferences that my audience has in terms of content in games and why. But to start let’s analyse the audience that voted:

#### Audience



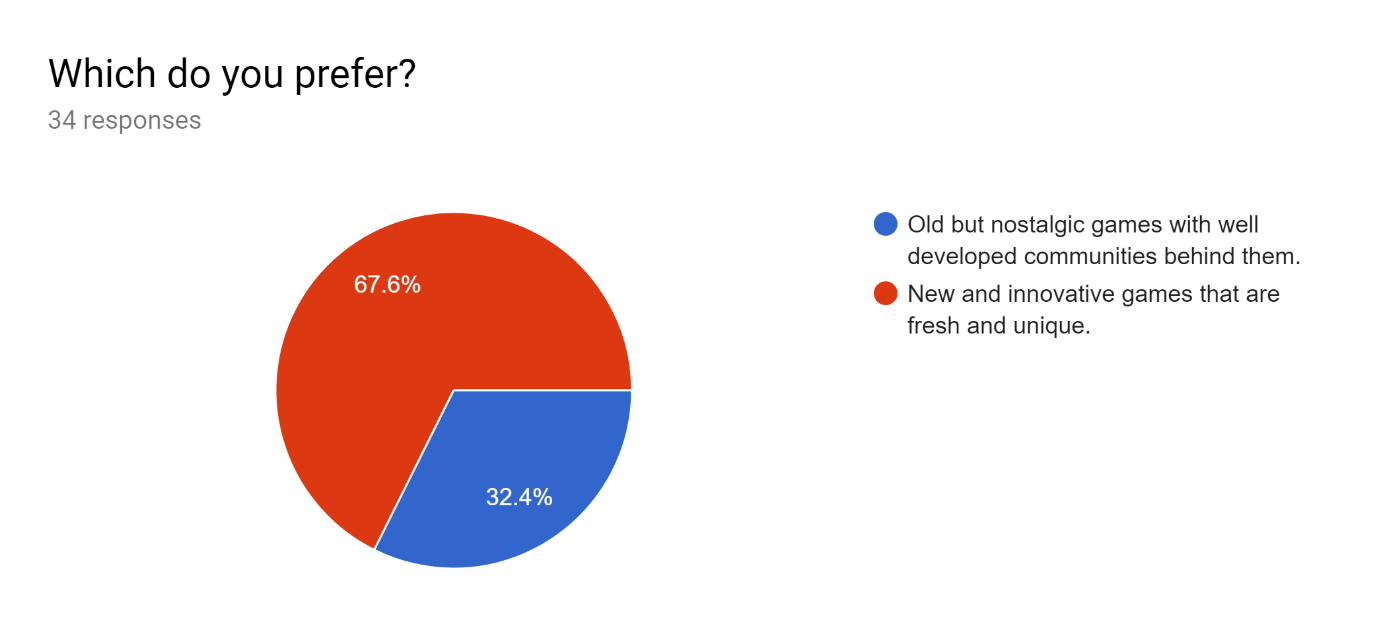
This shows that the majority of votes were from males but not a completely overwhelming majority. Since there is quite a balance in the genders, the part where I wrote about being as gender neutral as possible when making my project would be something that I will stick with from seeing this data.



This data shows that the general majority of the sample tends to be either serious games or casual games which a good thing as that means that I can stick with my ideas on skill curve in my game and making the game satisfying in general which would increase overall appeal.

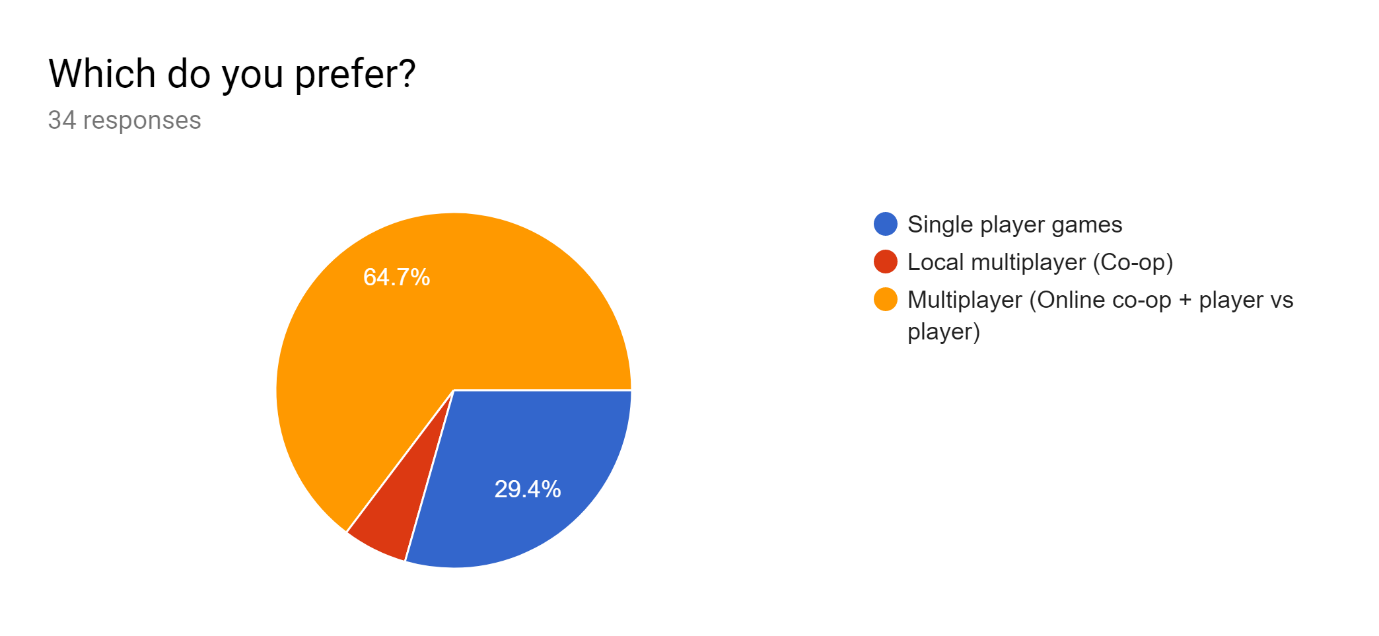
#### Market

These questions on market are so that I could analyse how much my audience views issues in the industry and what they tend to prefer:



An overwhelming majority voted for new and innovative games which was what I was expecting as there is massive and instant success of games of this genre such as Fortnite, Playerunknown’s Battlegrounds, etc. especially in the bottom end of my age demographic which is mainly where I got this data from. But since the majority that want these new innovative games have been catered for by big companies that produce many games of this calibre, the minority that want games with that older game vibe are left with not much to work with and is why I want to create a game in that theme.

This does not mean I cannot take aspects of these new games’ success and implement them into these older games though as these older games seemed to lack polish which was for some practical reasons, as back in the day pretty much every game was produced and distributed on a cartridge or a disk to be played on a non-internet enabled console which meant that after the cartridges were distributed, there was no good way to fix bugs that slipped through play testing. Now however since games could be distributed and downloaded online, patches could be made from feedback from customers and a patched version of a game can be instantly accessible to the people who purchased the original product. Why this matters is that it is one of the aspects from these newer titles and technology that I can use to create a game with that old game vibe but have the new game kind of polish.



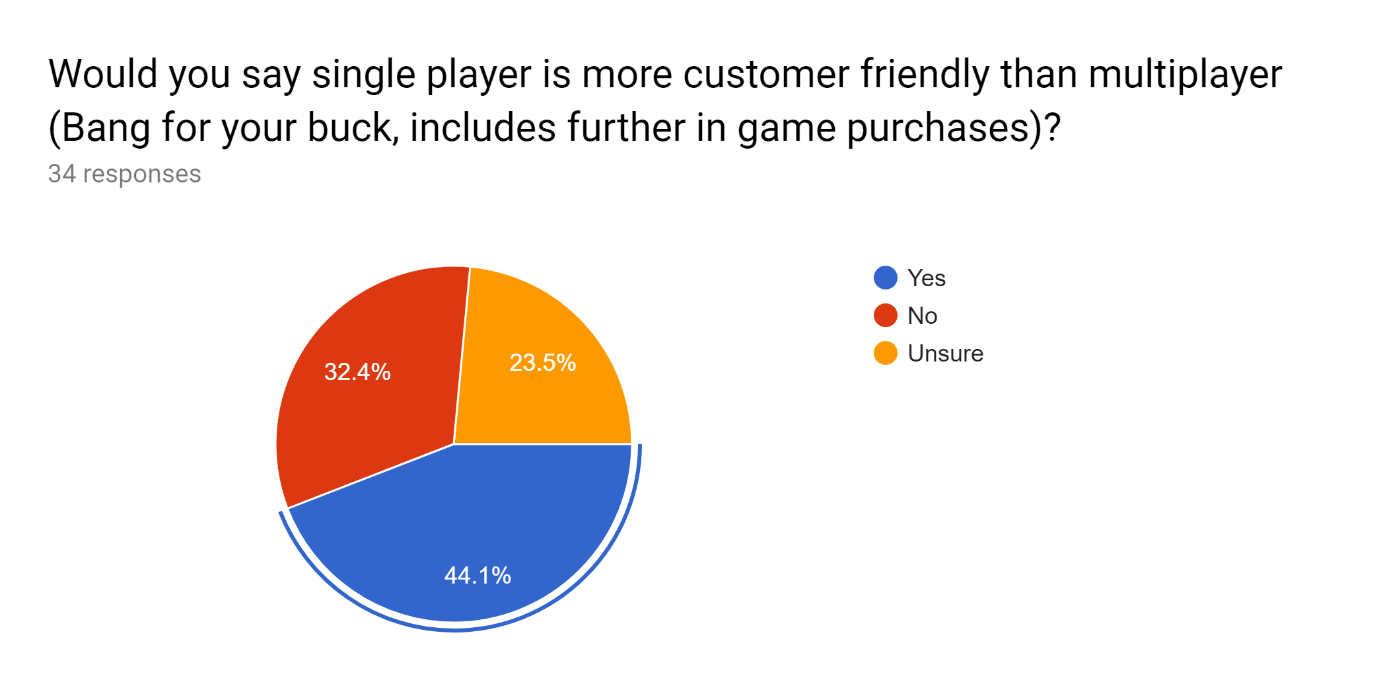
This question (like the last one) has the multiplayer option being the overwhelming majority for the same reasons as the new vs. old game had an overwhelming new game preference. But to elaborate more, the customers were able to give a reason as to why they preferred their choice over the other two.

The majority in favour of online multiplayer gave reasons that they found that medium more fun and that they could play with and against their friends with one response being “Teamwork makes the dream work”. This tells us that these kinds of players prefer the medium so they could bond with their friends for laughs (We can tell through the very jokey replies to do with this as well) and time together rather than to enjoy a storyline which could mean they play these games as a lot of people play them so they would get a lot of chances to play with these people. Another big reason stated was the competitive nature of a lot of these big multiplayer titles was appealing which could be seen as there being a big thrill on self-improvement to get higher on leader boards, or the rush people get from outsmarting their opponents as this kind of game tends to extremely fast paced naturally which only a few genres of single player games could possibly consistently achieve such as the horror genre or the games that were built for speed running. Other replies include the thought that there is more potential for variety for multiplayer games and that single player games get boring after you complete the main storyline and that multiplayer games won’t get boring as such, which is an interesting point which I hope to analyse further after finishing with this set of data.

This leaves us with the single player voters. This was the minority which for the same reasons as discussed before leaving this customer base as generally underserved and understated. People that voted for this option tended to have different desires in a game to the multiplayer player base with people valuing “a better story” and that these sorts of games are more “immersive” which are general arguments as to which single player games have more potentially to fit these motives. But one response was interesting as it stated that “They are usually depend on better programming rather than constant updates” which allures to single player games being given out in their entirety on release more often than not and rely on good game design rather than constant changes and updates and even in a lot of cases in multiplayer games, paid DLC even on release of a product which is argued that it should not be extra content that should be charged for but rather should come with the initial purchase of a game. One other person talked about how they are “not very good” which is also a reason for them to not be able to enjoy multiplayer games which require skill at a lot of times to find enjoyment out of it.

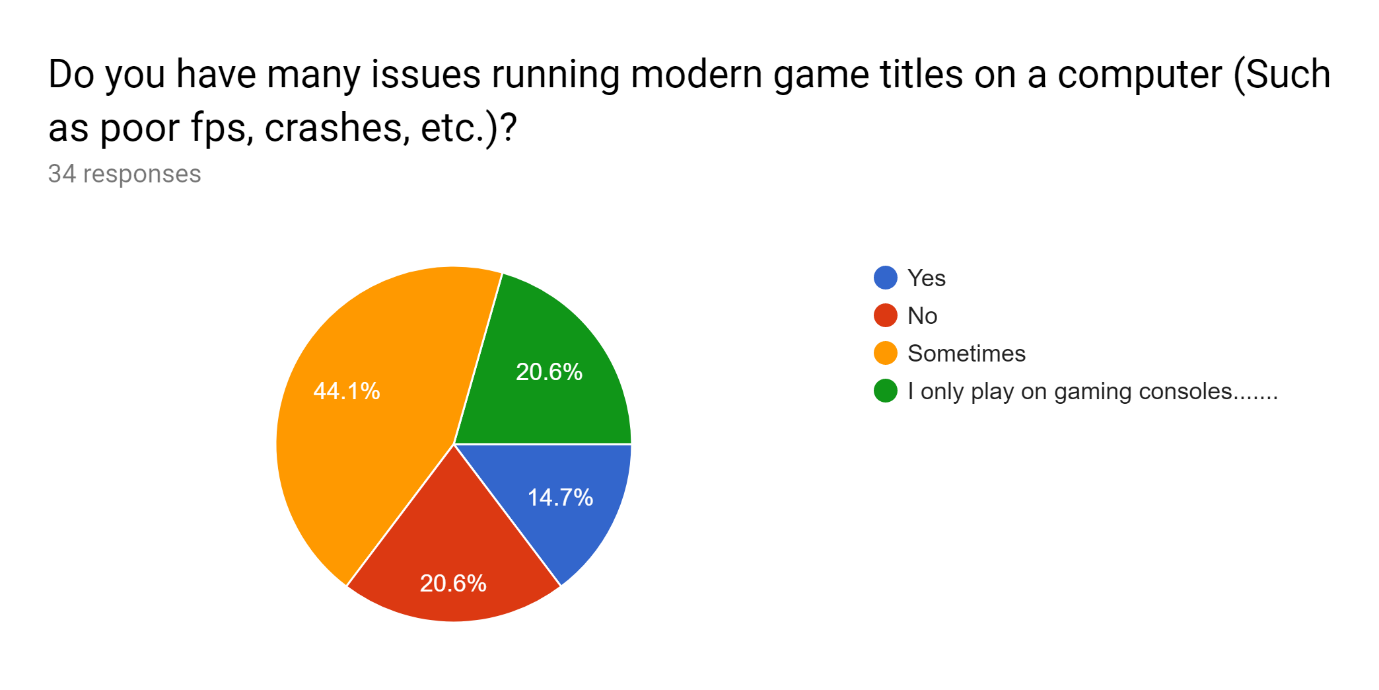
Now that that is out of the way let’s talk about the replay ability of games after initial completion. This issue encompasses a lot of things in a game from the type of extra content there is after initial completion of a game, to how much actual extra content there is to other mechanics that try to increase playability. First there is the type of content, this is the issue as in what can you possibly do after completing a core part of a game. Let’s talk about the bad ways that this is done, a lot of lazy developers don’t think about the importance of post-game content as this is seen as less marketable as there is more emphasis on the main part of a game and this kind of content is only really seen as good if there is a multitude of unique things to do and so in many cases, the only post-game content are very basic side-quests masked as something else which serves no importance to the player other than for a chance to beef up the player some more. The reason that this kind of post-game content does work at all is that there is no incentive for the player to carry out these kinds of menial tasks. In the storyline the player was encouraged to do these missions so that it would make their lives easier when beating an end game but here it seems to be just aimless reward giving for the sake of reward giving. Then there is the issue of how much content there is which is self-explanatory in itself as the more (quality) content there is the better. Finally, there is the mechanics that are in the game from the beginning that give the player to carry on a game; these include: collectible items, achievements or maybe even stats leader boards. These kinds of things tend to be in every game, but the thing is that not many games use these systems in a useful or innovative way that truly encourage players to go that extra mile to 100% a game. This is because of mainly that they lack depth and seem to be just a ploy to get players to keep playing the game rather than an actual reason to continue playing.

There are some exemplar games however that do this well, but the thing is it is easier to implement in some games than others. Such as open worlds games, depth could be much easier to implement to add compared to linear games which would require more individual levels to be programmed. The same with games that have good procedural generation, can continue with that procedural generation with little alterations to spice things up. Games such as my favourite “Pokémon mystery dungeon” has a lot of extra content after the story that you have to work to unlock the extra content which is one way to make this content actually be not like a half-ditched effort at getting play time. Other specific ways this is done includes having great mod support for a game so that extra content could be made by the community for the community. Here is a good conversation for the topic on an IGN forum ([LINK](https://www.ign.com/boards/threads/why-do-so-many-games-this-gen-lack-single-player-replay-value.452777979/)).

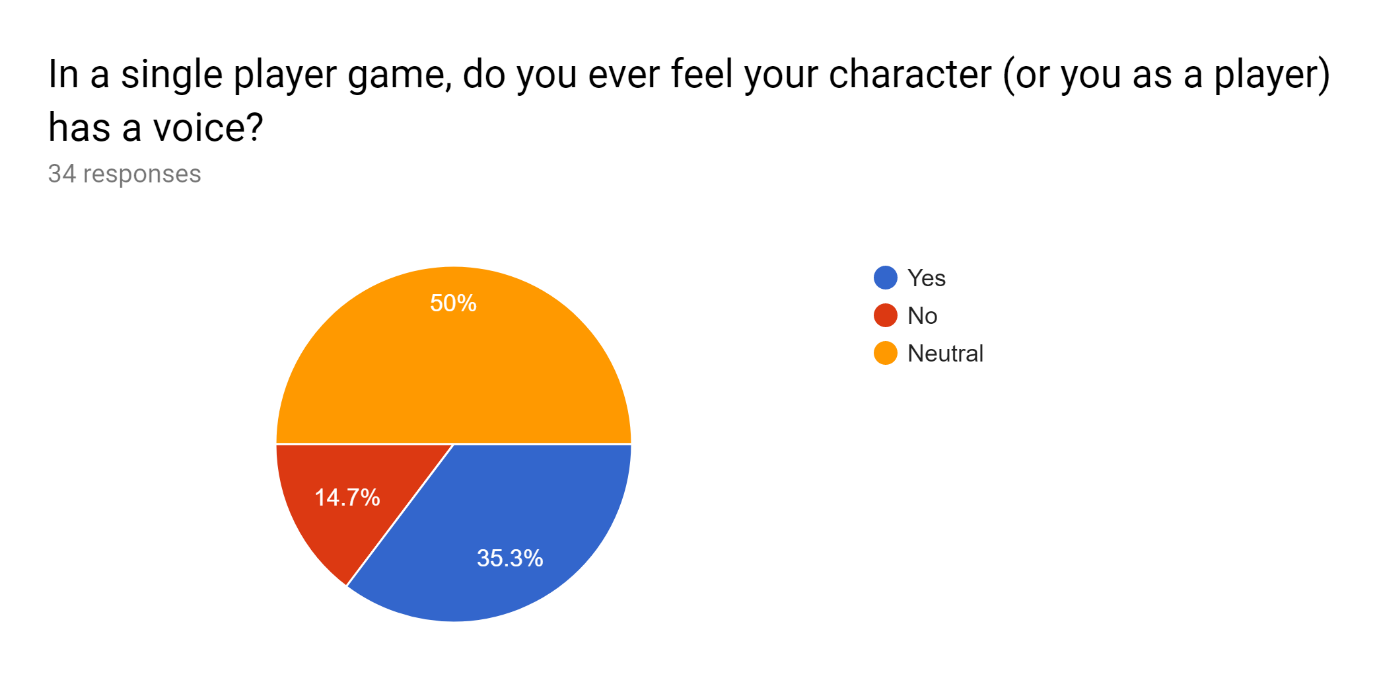


This data tells us the public opinion of the industry on how companies manage extra costs in games and that people feel that single player games give people more for one purchase. The vote of majority being yes shows this to be true in the view of the consumers as well. A lot of this has been mentioned previously so there is no need to elaborate further.

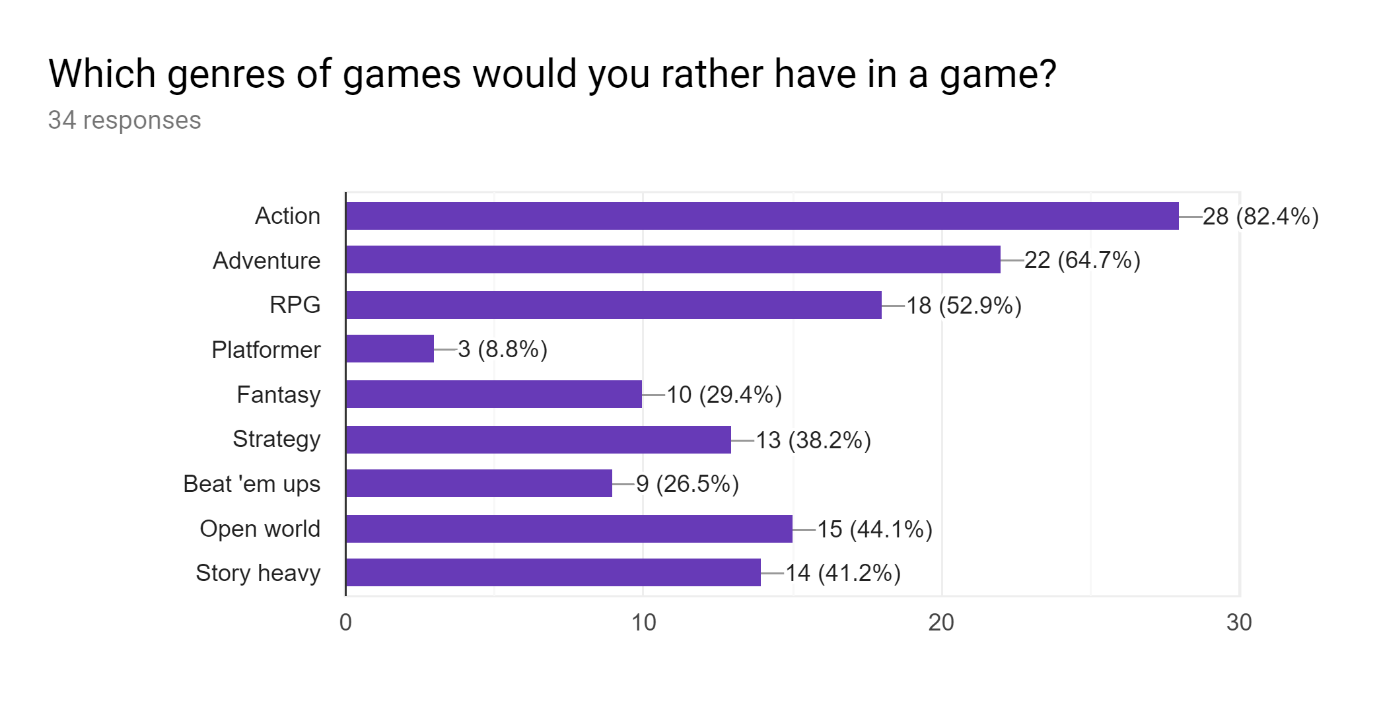
#### Preference/Other



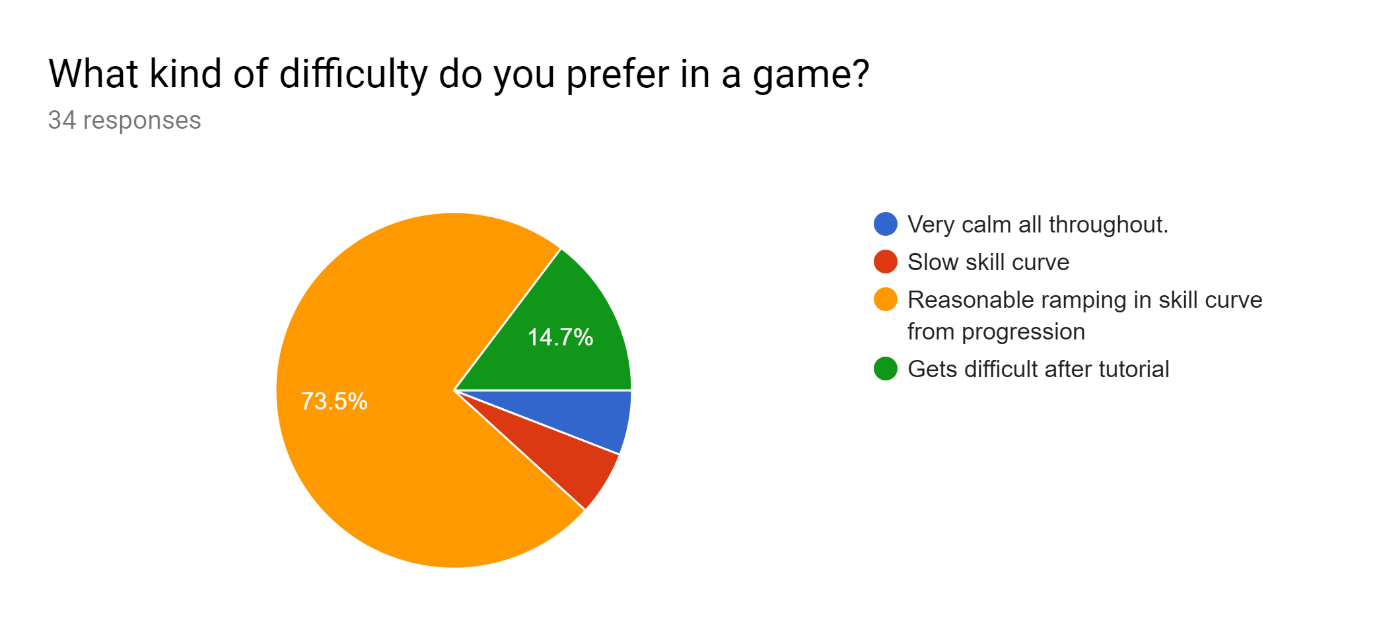
The majority lean towards having somewhat issues with running games on their computers which means that making a game that isn’t very demanding in terms of pc specs would seem a welcome thing. Also, the fact that 20% were console gamers mean that they would most likely not pick up a game from me as I am only going to make the game available on a computer (Windows) right now but if I have too much free time at the end of my project, console porting could be an interesting idea to play around with.



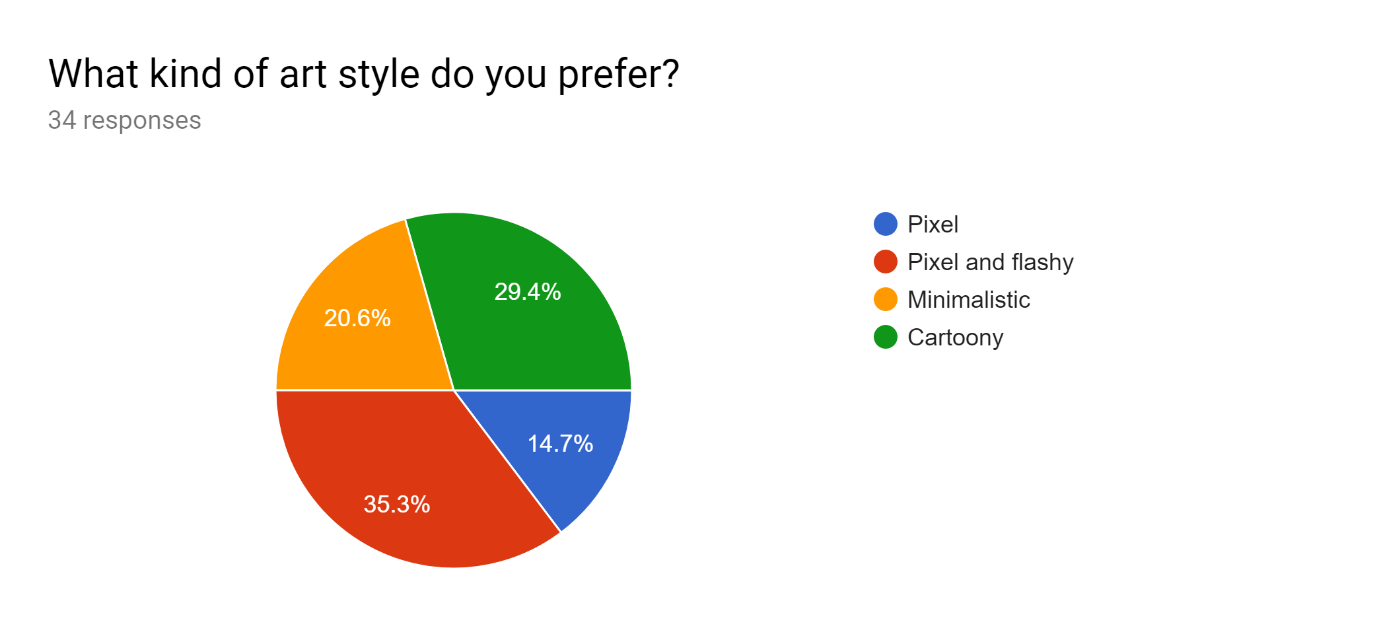
The general consensus is that people don’t generally feel the character doesn’t have a voice in many games. This means that I won’t need to do anything special in terms of this as just a general approach with minimal character involvement in the say of the story would suffice.



The fact that a platformer is the least popular out of all the genres means that I am going to definitely focus less on this aspect of the game as I was initially thinking of working heavily on making the game quite platformer heavy. The top three results is what I was told to set out and create a game on and so it is not a surprise that these 3 genres are the most popular choice out of my sample. The idea of an open world game which is also story heavy is interesting as well as it will be a little tricky to also fit a convincing and deep story within all the other genres.



This is self-explanatory, linking back to when I was talking about skill curve in terms of the experience of a user and what would work. Seems like a skill curve that attempts to satisfy both sides of the spectrum seem to be overly popular compared to all the other options which are dwarfed.



The results of this question were very split with pixel and flashy graphics just barely winning out over the rest. Flashy graphics could cause issues such as seizures in some people so I have to be sure not to get too carried away with the graphics or this could happen to a client which has its obvious repercussions.

### Existing solutions

When I’m talking about existing solutions to my problem, I’m talking about remakes of the game that is my inspiration of that being Zelda 2 (As my solution will be quite similar to this anyway). Here I have found a few examples of these remakes ([LINK](https://www.youtube.com/watch?v=Ay4UDrEHJJM)) ([LINK](https://www.youtube.com/watch?v=1orsteUF9Y8)) ([LINK](https://www.youtube.com/watch?v=5JBpTk1TMGM)). But what all these other projects had in common is that they are not completed by the people who were working on them and rather left in a half-completed state, even if they had a lot of potential. These games looked to have a lot of potential and I can take some ideas of what these games attempted to create and put them into my thought process of my game. Here is a link to a forum discussing the thought that these old games that gave Nintendo a big name has not received any extra love from the company ([LINK](https://www.resetera.com/threads/its-pretty-incredible-that-zelda-1-and-2-havent-ever-been-remade.19793/)).

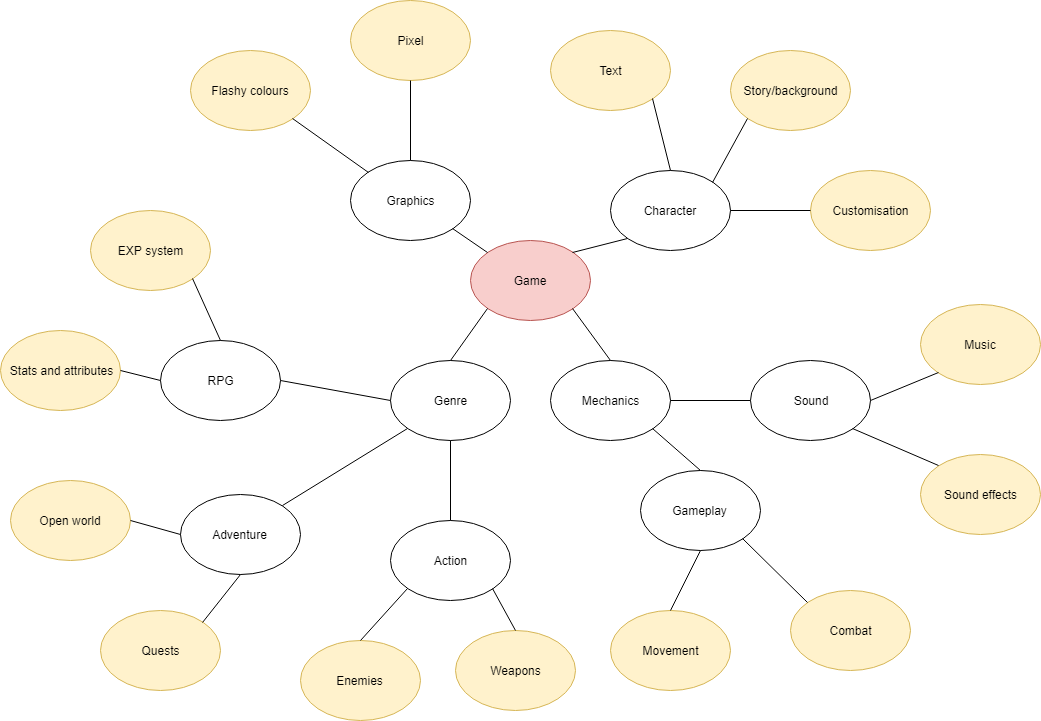
### Justification of Computational Methods:

#### Abstraction

Abstraction is one of the four cores of solving problems in Computer Science. It represents the idea of concentrating on core aspects of a problem and solely the core aspects while ignoring more minute details, so the overarching problem can be solved as that is what matters.

In context of my project, abstraction can be applied in a variety of places. Being chronological, the first would be imagining what an outcome of the project would be. If we leave out a lot of the little details I have analysed in this document, then we have a simple idea of the project. With this we could just focus on core game mechanics over stuff like art and sound to start off with as if we make our game work first in a basic sense then we can adapt to that easily.

#### Decomposition

Decomposition is the idea of breaking down a complex problem into multiple little problems that could each be solved on their own individually until the initial problem has been solved. This is probably the computational method that I like the most and will use the most as this method can be applied to pretty much every problem while other computational methods such as abstraction cannot be used in every single problem.

This diagram shows a very brief way of how the decomposition process will go for me and my project. The yellow bubbles would be the problems that I would tackle each individually and I could decompose even further for some of these.

### Methodology

When talking about Methodology, we mainly talk about the development techniques used to create a project with the two main being:

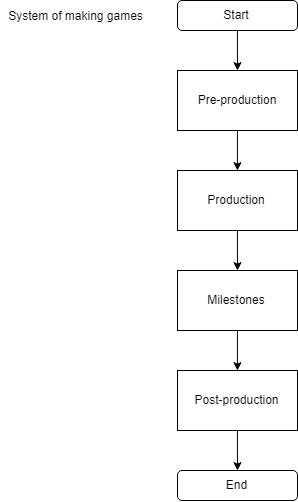
* Waterfall
* RAD (Rapid application development)

Waterfall is generally described as a way or development where everything flows down sequential goal, and everything sticks to a strict schedule where changes in direction are only ever made very early in the process.

RAD however is different in the sense that it a more adaptable approach to a deadline. It is said to be built on 4 core concepts ([LINK](https://kissflow.com/rad/rad-waterfall-agile-which-one-is-best-for-you/)), that being reusable code, quick prototyping, constant client feedback and building usable software as quickly as possible. Why there is emphasis on speed is that the more you fail, the more you can adapt the final outcome to better suit the client’s desires and so this medium heavily relies on feedback from the client. For smaller projects such as mine with a one man development team this would be the more preferable as the flexibility is not an impact on communication as it will only be me, plus this kind of methodology would allow me to be less restrained to adapt to add new trends to a project, etc.

### Creating games

We can use decomposition as a computation method to break down the sections of creating a game generally. To start with this, we can draw a flowchart to make this more visual:



This info came from an article on Wikipedia ([LINK](https://en.wikipedia.org/wiki/Video_game_development#Design)).

**Pre-production**

This is the planning phase of the project (a lot like this analysis section) and is self-explanatory in why this section is hugely important. From this we get out concept through talking about needs of a customer and then putting those desires into a broken down.

**Production**

Here is the main design phase of a project where the ideas make from planning are put into practice and is where I will spend the majority of my time on my project. Everything from design of the game’s aspects such as gameplay, sound and writing is drawn out and then after all this is collated into a nice organised structure, is then programmed. A lot of procedural testing happens on this phase as well done by the developer.

**Milestones**

This can be said to be a sub-section of production, but it is separate here to elaborate what they are. Milestones in a project include Alpha, Beta and release. Alpha is the part of the development cycle where the product includes all the core gameplay functionality is implemented either to a very basic standard or not. Alpha testing can be done to get feedback on these mechanics and to have people play around with these to try to find extra bugs and unintended stuff in these mechanics. Beta is the next stage of the cycle and is where the only thing left in the game to work on is bugs as the main aspects of the game are completed and require close to no alteration.

**Post-production**

This includes processes like patches after launch to improve aspects of a game that were not intended and slipped all previous testing phases. There will not be much emphasis on this section for me just for the purposes of this one project.

### Limitations

**Physical**

The only real limitations would be that the client’s computers must have the windows operating system and the reason for this is because my program in theory will be able to run on pretty much any modern pc even if it is not very good as the game will be very basic with bitmap graphics with nothing 3D. Plus python programs aren’t very hard to run.

**Limitations of proposed solution**

A game cannot solve this problem forever just to keep in mind, as this is a different kind of problem. After the game has been enjoyed to its fullest extent, the client will not continue to play the game even more as they would not have a reason to play this game forever. This is not something that I can do too much about as it is a single player game and attempting to continue such a thing would entail its own issues such as how hard it would be to continue the same story without repeating any of the same story devices and concepts just to name one issue.

## End User Requirements

1. Implement OOP in a variety of ways
   * This is what I am mainly being assessed on when creating my project, so it is important that I am able to achieve this point successfully and have the implementation of this clearly show my understanding for the principles of OOP.
2. Complete a full, comprehensive and coherent storyline
   * A lot of my audience want this, so it is a requirement
3. Create a save/load system that works
   * This is something that I’ve highlighted in my inspiration, so I must fix the problem when creating my program
4. Have my game able to display different screens, one being the overworld and one being when inside a level
5. Have controls that can be changed according to the client’s desires
   * I want to have this as an end user requirement just to remind me to build my project on robust, easily changeable parts.
   * Also, just because this is a great quality of life inclusion in a game
6. Have a full system for enemies to give EXP and the user to be able to alter stats

# Section 2 – Design

## Design objectives:

1. Create a system to be able to add new enemies or NPCs without needing to change any code. **INPUT**
2. Create an in-game map creator so levels can be created with an in-game GUI. **INPUT**
3. Create a system for the controls of the character to be interchanged depending on the user’s desires. **INPUT & PROCESS**
4. Implement an OOP structure for the main character as well as a separate one for enemies. **PROCESS**
5. Have a saving and loading system. **PROCESS**
6. Have a system of experience points being given to the player when monsters are defeated to level up. **PROCESS & OUTPUT**

## Decomposition of Design objectives:

### Enemies & NPCs (Input):

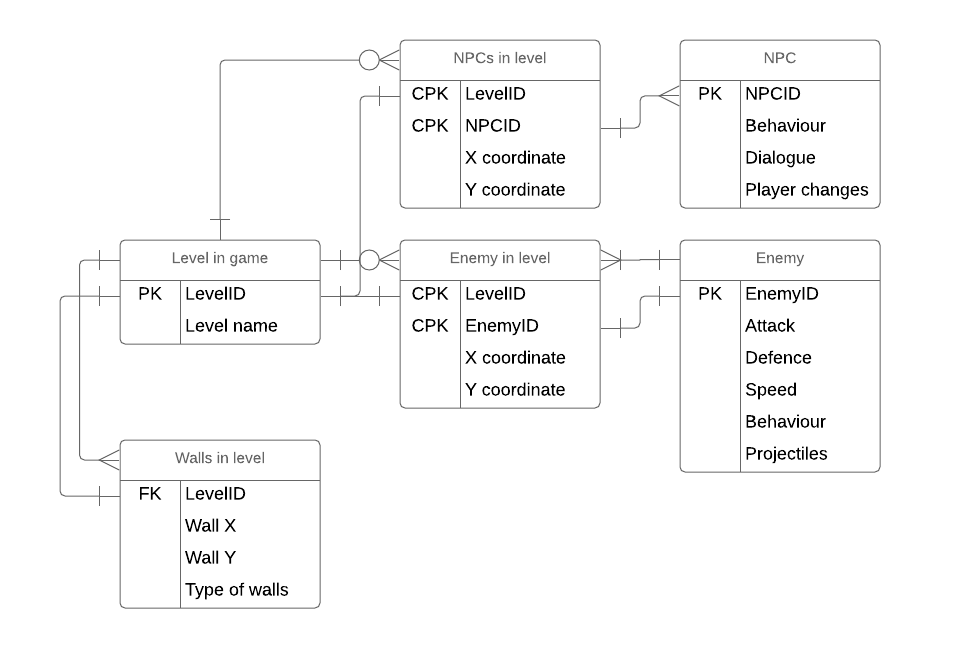
1. **Create a system to be able to add new enemies or NPCs without needing to change any code.**

Although this objective is not a feature that is going to be the backbone of my project where the program will not be able to be created without, it does serve as a way to both make the code way more efficient as well as saving my a lot of time on adding enemies and NPCs which in its own way would be just a very repetitive process and so automating this aspect more would leave me more time to focus on other problems in my project.

First to approach this we must separate enemies and friendly NPCs as they both tend to act very differently when interacting with the player as enemies tend to pursue the player with intentions to attack while NPCs generally hover around or sometimes do not move at all from their designated spot and wait to be interacted with by the player. To separate them I will make them completely different objects with different classes used to instantiate them, this would be better than putting them all in one class as since they both have different functions in a game, they would have different sub-routines letting them interact differently with the player.

To be able to satisfy the next part of this criteria, that being not needing to change any code. What would need to happen is that I would need to have a database for all the enemies and all the NPCs and have it so that these databases could be easily added with images being imported and recognised through file names being added to the database as well as different behavioural patterns for example I could have it that there are 5 types of behaviours that the program recognises, “1” being a passive enemy that doesn’t chase the player but does damage when touched, “2” being an enemy that just chases the player, etc. These would be pre-programmed in the code and so are assigned to the enemies once they are instantiated.

But to make this work we must have it so that each level has a database where each entity that is to be included in the level will be added to the database, so if Level 1 has 4 enemies, when the level is loaded the game gets the ID for all 4 of the enemies and then scans it from the enemies database and then adds those enemies and all their stats into the level with its designated x and y coordinate. It will be done this way so that an enemy can be used multiple times if it is inputted into the level data multiple times with different X and Y coordinates. Here is an ER diagram showing this relationship.



### In-game Map Creator (Input):

1. **Create an in-game map creator so levels can be created with an in-game GUI.**

This feature is pretty much necessary if I want to implement a lot of levels into my game as adding this means that I do not need to type up the coordinates of each and every wall block in every level.

The aim will be to reverse the process that will be used to read a database for the data of a level and to display a level with that data. Reversing this means that instead of reading a file for all the data, we will be able to create the level in the game and when the process is done, save the data of that in the format that the game can later read and interpret. This means that I will have a database for every level which keeps the data for the position of every wall along with a database for each level that includes the metadata of a level.

This metadata includes:

* The LevelID
* The length of the level in blocks
* Which ends of the level are level change detectors (And where it leads)
* The pointer to the database that includes the position of all the walls
* The type of walls used
* The background art image file names.
* Music file name
* Enemies in the level
* NPCs in the level

Once I code the process of changing a level into code, I need to add a GUI to let me paint a level of its walls, drag and drop any enemies that I have created and are in an enemy’s database and then on completion set the metadata so that the necessary files can be created off of my input.

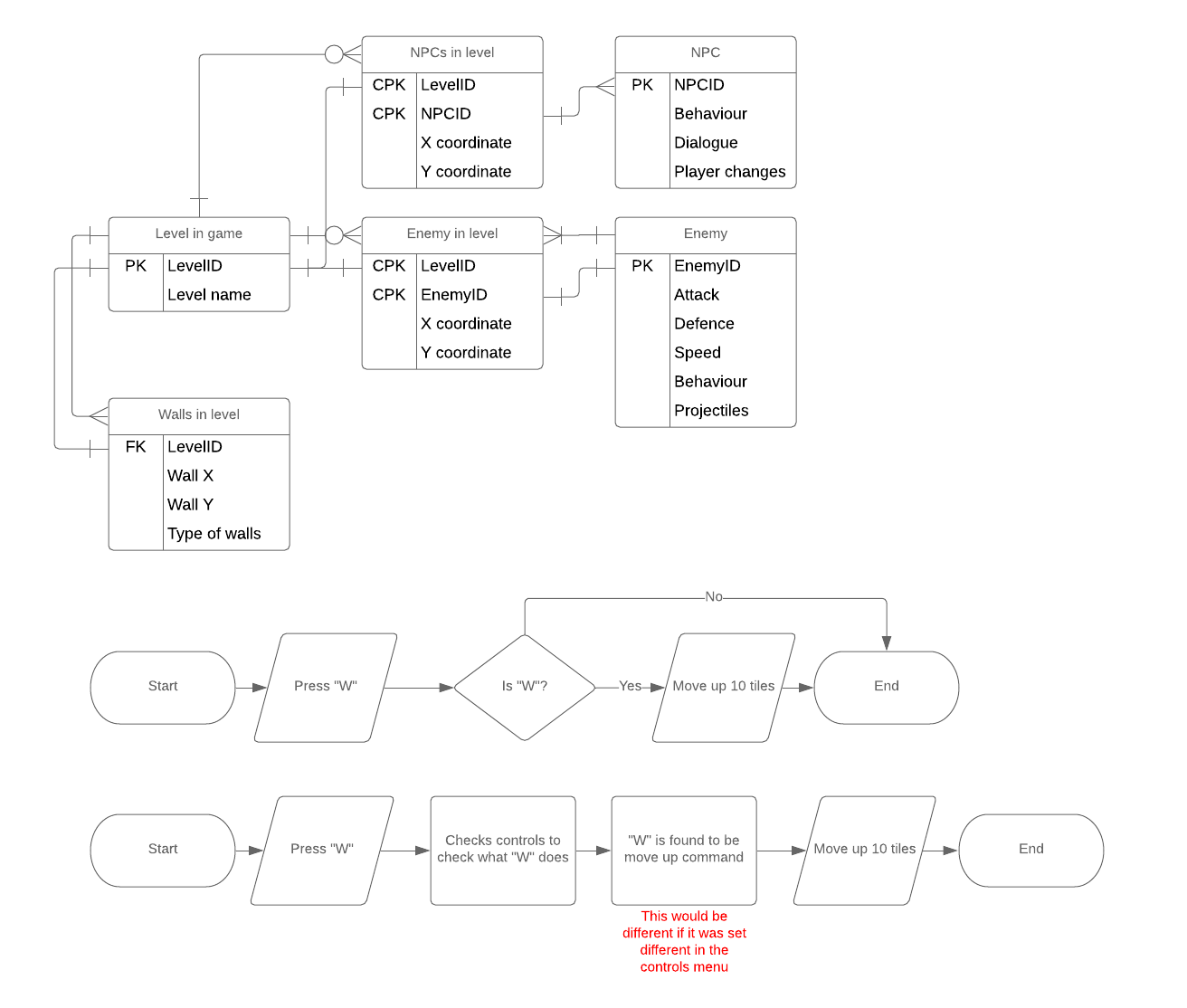
### Interchangeable controls (Input & Process):

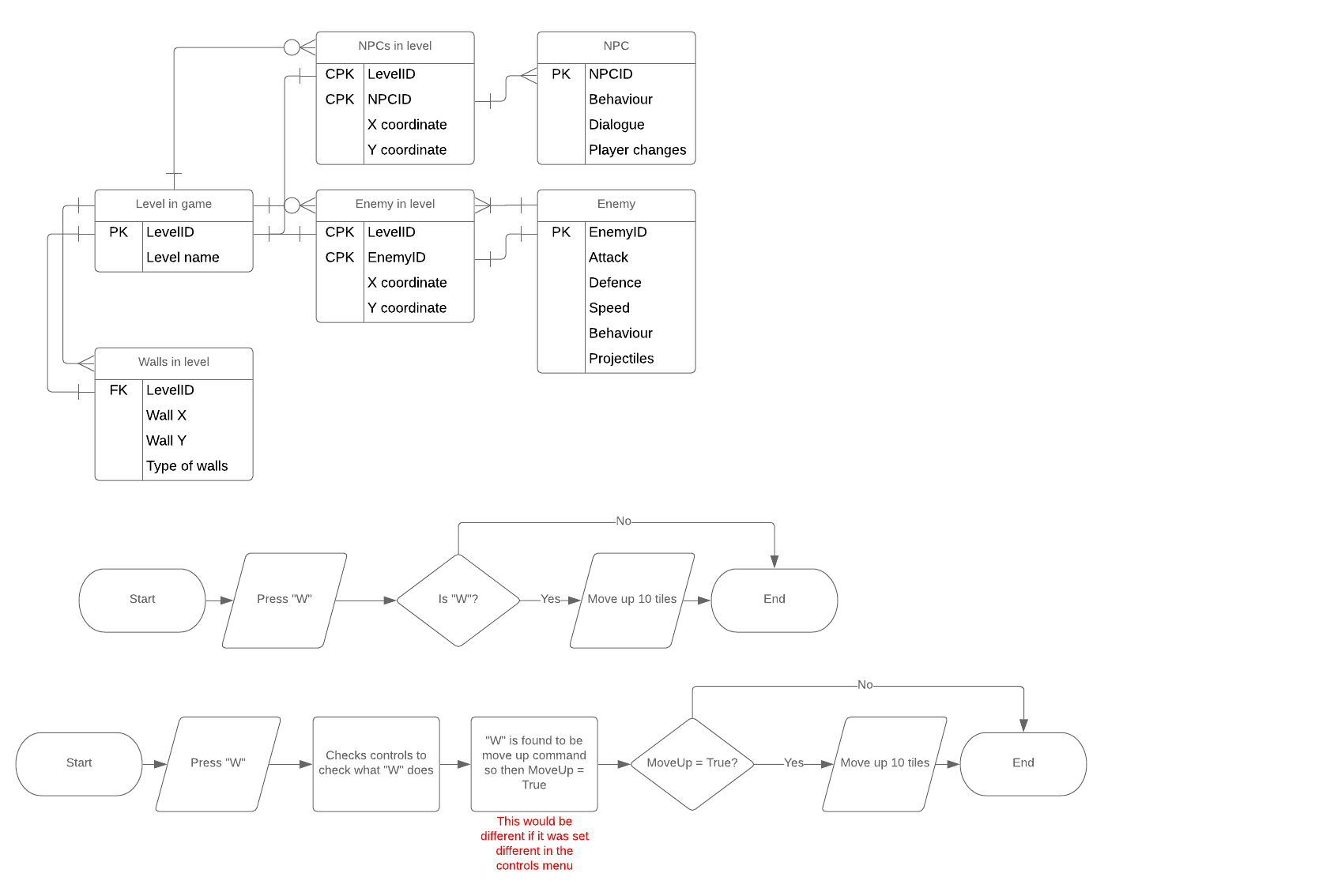
1. **Create a system for the controls of the character to be interchanged depending on the user’s desires.**

This is a quality of life design objective which would make the game more appealing to play as the player will have the freedom to choose the controls they want.

This objective is more straightforward than my other objectives as all this really means to me is that when I am coding my game, instead of having an input straight away formulate into an action, instead I need to add a middle variable which is changed by the input matching one of the changed controls conditions which is then checked by the action. Here is a diagram to explain:

Before:



After:

So, adding this middle ground means that the controls can be interchanged.

### OOP (Process):

1. **Implement an OOP structure for the main character as well as a separate one for enemies.**

This design objective is so my code would be neater to avoid clutter later on in my later stages of programming.

Creating a class for the player means that we can create the main character as an object with many sub-routines for the player all inside the class. Subroutines would include actions such as:

* Take damage
* Attack
* Move in any direction
* Jump

OOP structure for enemies would work differently than the player as there would be more than one object created from the enemies class. Each instance of this class would be one singular enemy entity and how it interacts with the level with subroutines and this does mean that there could be multiple of the same enemies in the same level as objects.

### Saving & Loading (Process):

1. **Have a saving and loading system.**

To save my game, I need to be able to represent all the game information of progress a player has made and then convert that into a text-based text file and then I need to be able to reverse the process by reading a save file and then converting the text format into structured game data.

I don’t need to encode the data as this is a single player game and so altering save files is going to be the player’s decision. But this depends on how I save the game data and what it would look like.

### Experience & Levelling (Process & Output):

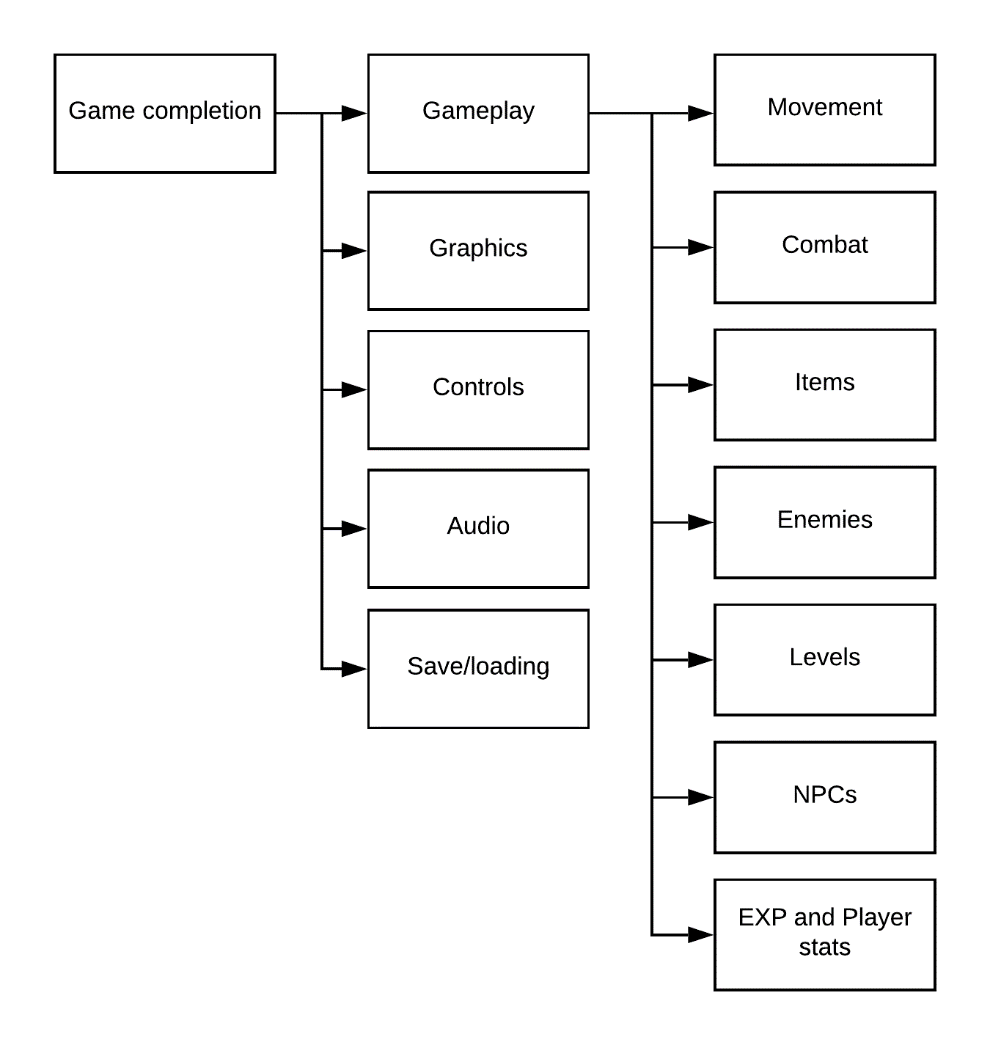
1. **Have a system of experience points being given to the player when monsters are defeated to level up.**

Experience will be given to the player after an enemy is slain or a dungeon is completed. If the player reached a certain required amount of experience points for a character upgrade, the player will have the chance to either upgrade that skill or save up their points for another upgrade that requires more exp. This exp counter will always be present on the screen of the game display and a small menu to assign exp will pop up once either the player presses the button to access it or the player has enough exp points to upgrade a skill (but will not repeat popping up if you decide to save your exp points).

## Decomposition of the whole problem:

This game will have a lot of parts to it, so a full decomposition will be very helpful.

### Gameplay decomposition:



Movement:

* Collision detection
* Movement changing items
* Movement changing enemies/effects
* Knockback

Combat:

* Swords and the different kind of attacks that can be performed with one
* Attack stat on the player
* Items that can damage enemies
* Enemy knockback

Items:

* Item picking up collision
* Using items and in each unique way
* How it interacts with environment
* How it interacts with the player
* Switching between items

Enemies:

* Importing into levels
  + Their X and Y co-ordinates
  + Their stats for things like attack, defence, speed, etc.
* Creating enemies (In terms of my first design objective)
  + Their stats
  + Their placement in a level
  + Their behaviour (AI)
  + EXP given to player
* What eliminating them does
  + EXP given to player
* Projectiles

Levels:

* Generate walls with designated textures
* Level creator (As explained in my decomposition of my second design objective)

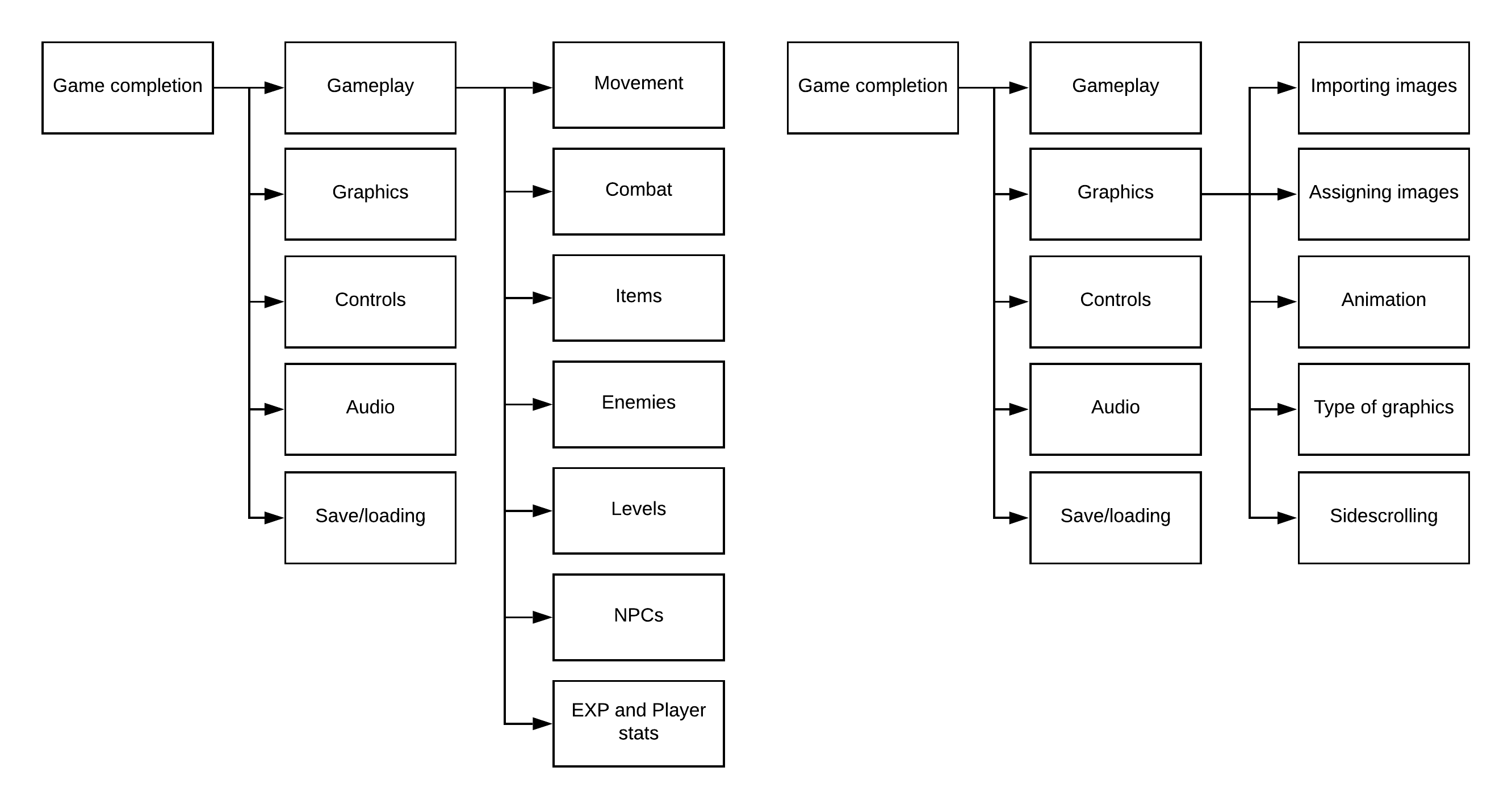
NPCs:

* Dialogue
* Update objectives menu for directions on where the player should go
* Behaviour

EXP:

* Levelling up stats with exp points
* Skills menu
  + Attack
  + Life

### Graphics decomposition:



Importing images:

* Finding the files in folders
* Creating those images

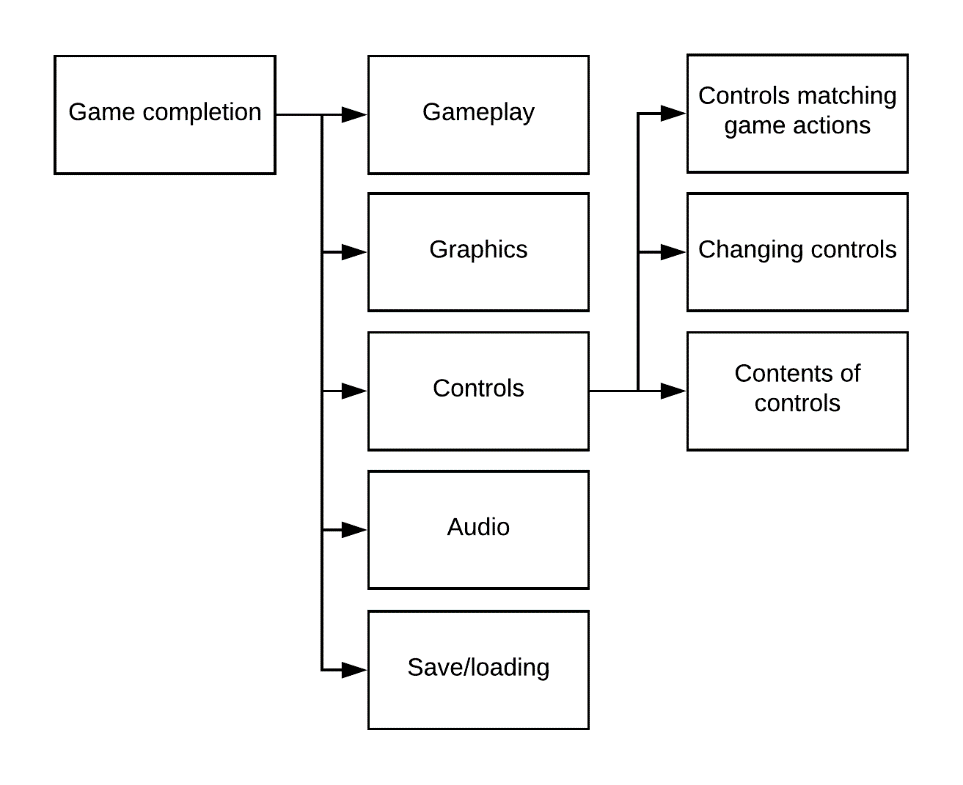
Assigning images:

* Walls images
* Player image
* Enemy images
* Backdrop art
* Menu images (Import/draw)
* Item images

Animation:

* Sword swings
* Player walk/jump
* Enemy walk/jump (If they can jump)
* Projectiles

### Controls decomposition:



Controls matching game actions:

* Already decomposed in decomposition of design objective 3

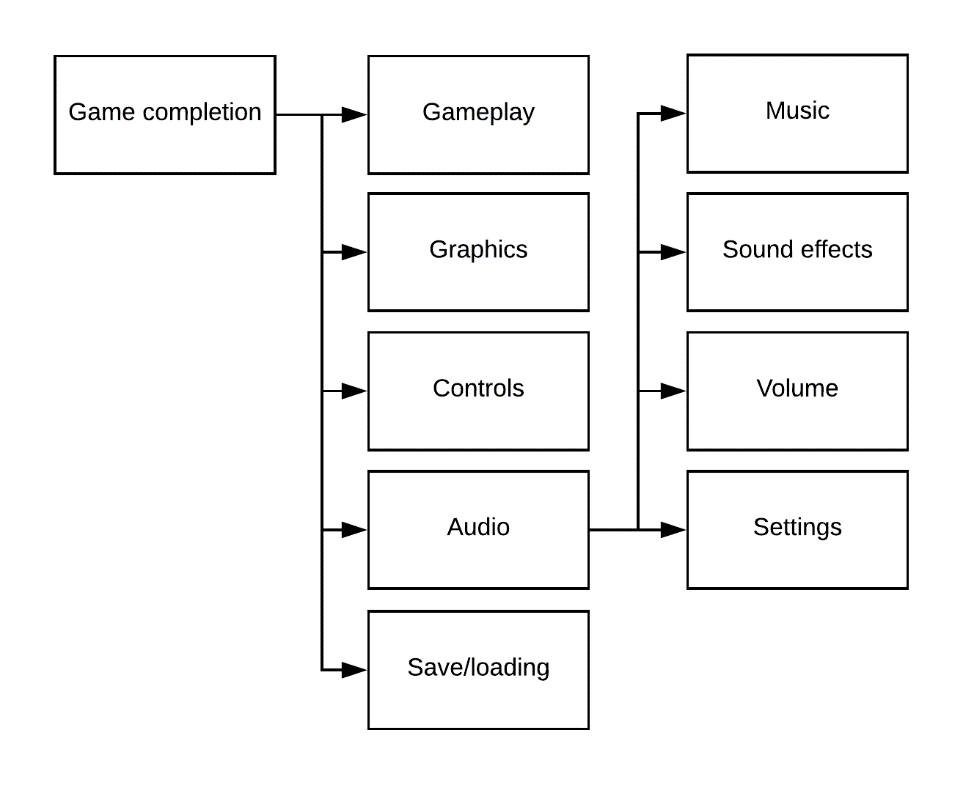
Changing controls:

* Already decomposed in decomposition of design objective 3

Contents of controls:

* Movement
* Attack
* Items
* Aim

### Audio decomposition:



Music:

* Background music in a level
* Matching up correct music to a level

Sound effects:

* Attack miss
* Attack hit
* Jumping and hitting ground
* Walking

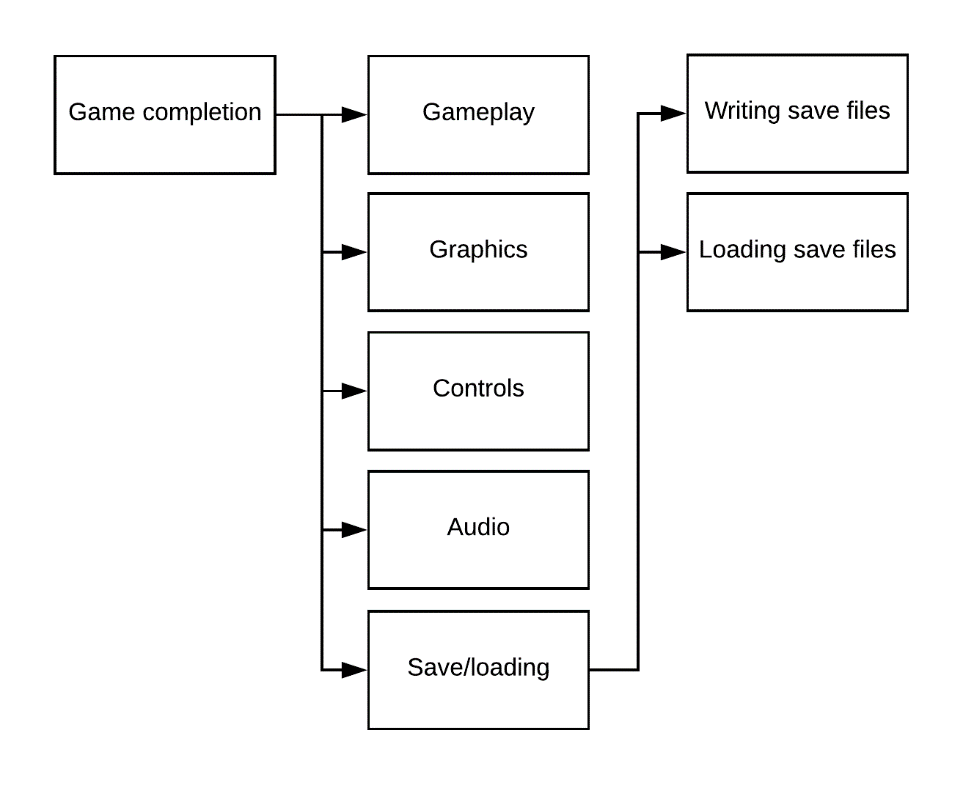
Volume:

* Correct balance for default volume

Settings:

* Master volume
* Sound effect volume
* Music volume
* Ambiance volume possibly

### Save/load decomposition:



Writing save files:

* Player stats
* Player exp
* Player game time
* Player progress (Items gotten; dungeons completed)
* Player save point
* Player life points at time of save
* Changing all this data into a format that can be put into a save file/database

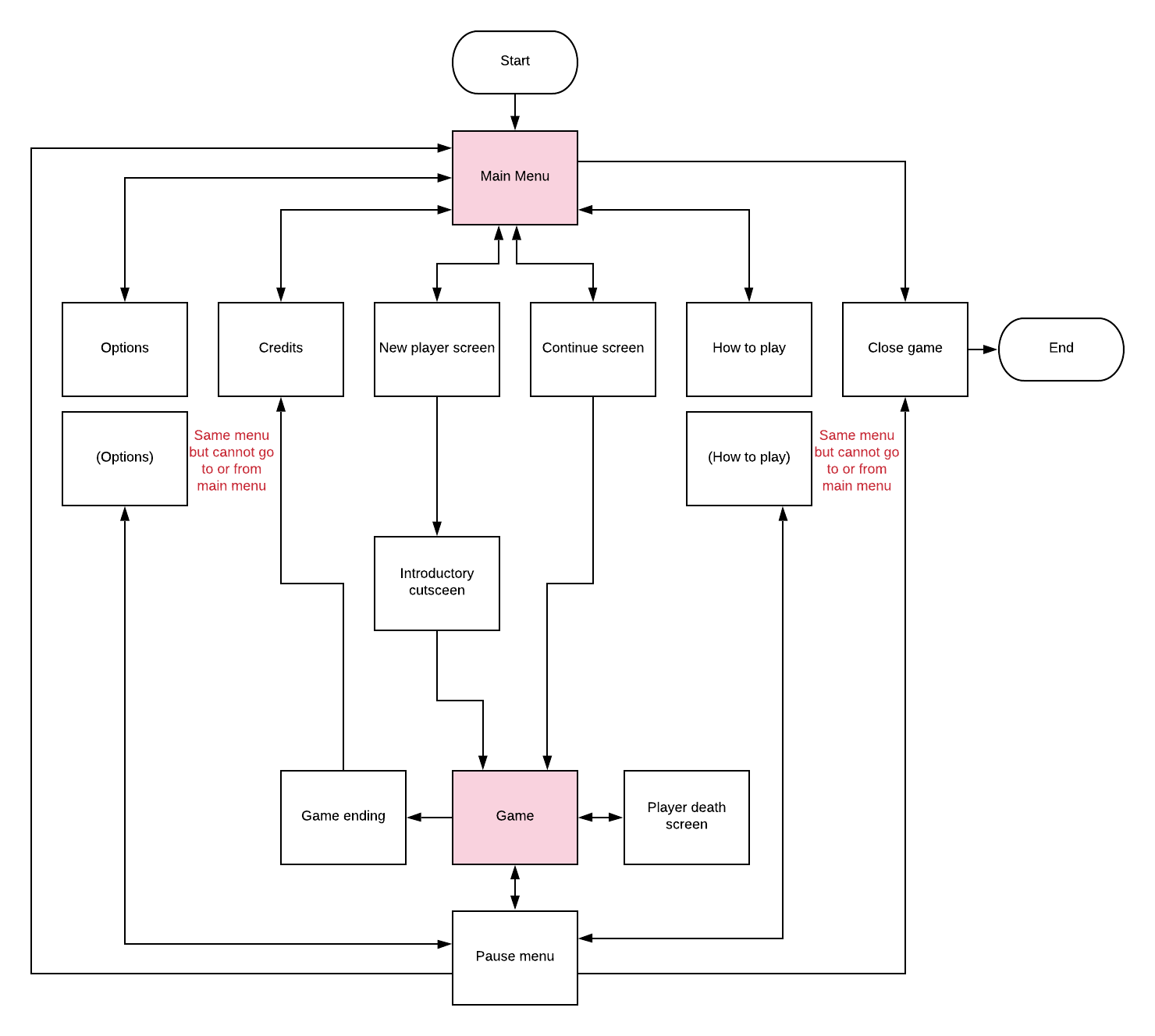
Loading save files:

* Interpreting the data of a save file and convert into in-game data

## Describing the Solution

### Menus analysis and MFD

Since my project is a game, it would be good to start off describing the solution with a Menu Flow Diagram as the project is more graphical than data orientated:



Main screens

#### Simple menus (From main menu):

* **Credits:** Simply accrediting my project and work to myself, mainly just text on a screen
* **How to play:** In both iterations where it is in the in-game screen and in the main menu screen it will just be text describing controls and objectives with basic illustrations. This would me multi paged if detail is needed and the player would navigate the pages by clicking arrows.
* **Close game:** This will come with a confirmation of closing game and then closes the program.

#### Options menu:

The option menu will have tabs for different types of settings one for:

* Audio settings – Music and sound effects will be able to be scaled up and down
* Video settings – Screen resolution and full screen will be interchangeable changed here
* Control settings – Controls can be mapped to different keys

|  |  |  |
| --- | --- | --- |
| **Action** | **Default Key** | **Alternate Default Key** |
| Move Up / Jump | W | UP ARROW |
| Move Left | A | LEFT ARROW |
| Move Down / Crouch | S | DOWN ARROW |
| Move Right | D | RIGHT ARROW |
| Interact | E |  |
| Attack | Left Click (Mouse 1) |  |
| Use Item | Right Click (Mouse 2) |  |
| Item Menu | Enter |  |
| Menu | Esc |  |
| Next Item | MOUSE WHEEL UP |  |
| Previous Item | MOUSE WHEEL DOWN |  |

#### New player screen:

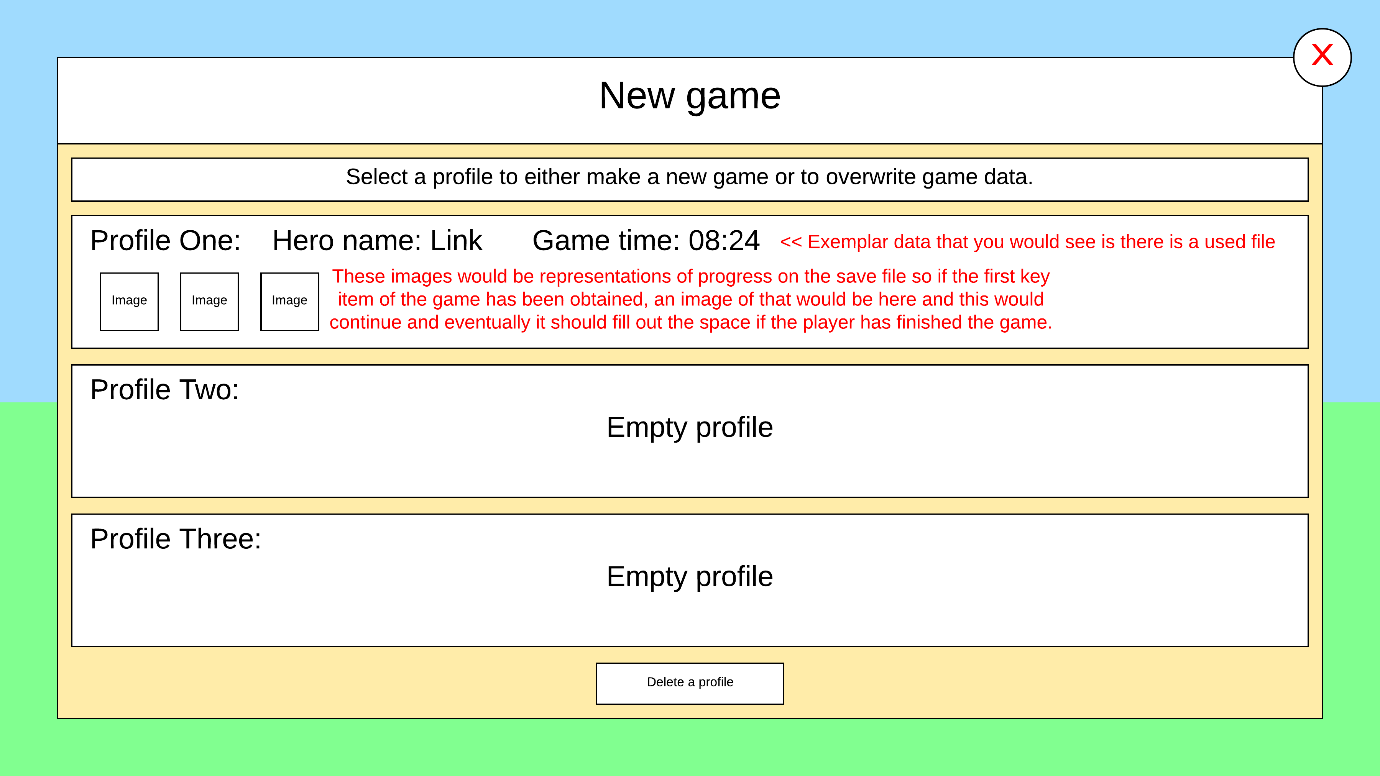
The player will have the choice to either create a new save file in one of 3 files or navigate back to the main menu screen. If the player decides to create a new save file in a **slot that already has game data**, then a confirmation message pops up and if accepted, then that **save file is reset** and made into a new one. Also, on this page there will be options to just **delete a save file** if that is what the player wants.

### Menu mock-ups (Input):

Main menu:



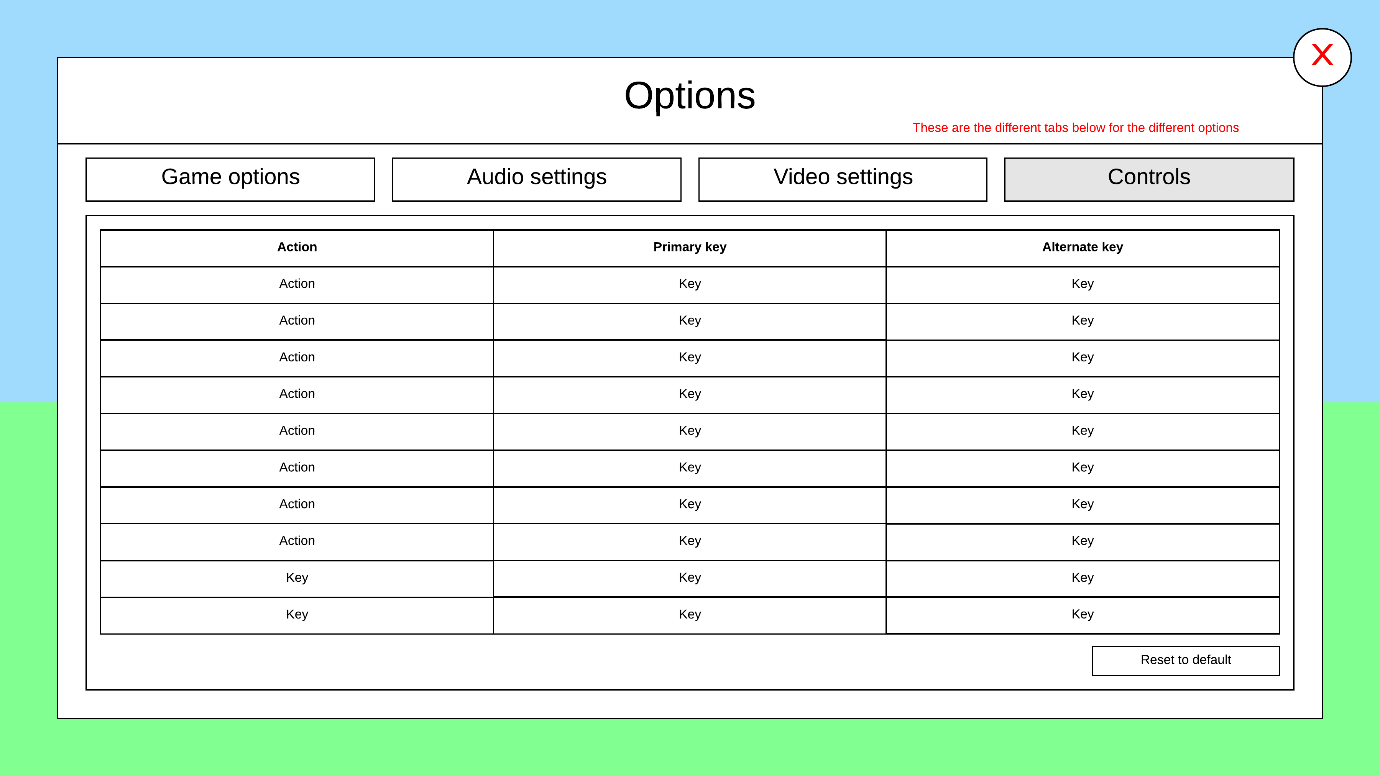
Main menu --> New Game:



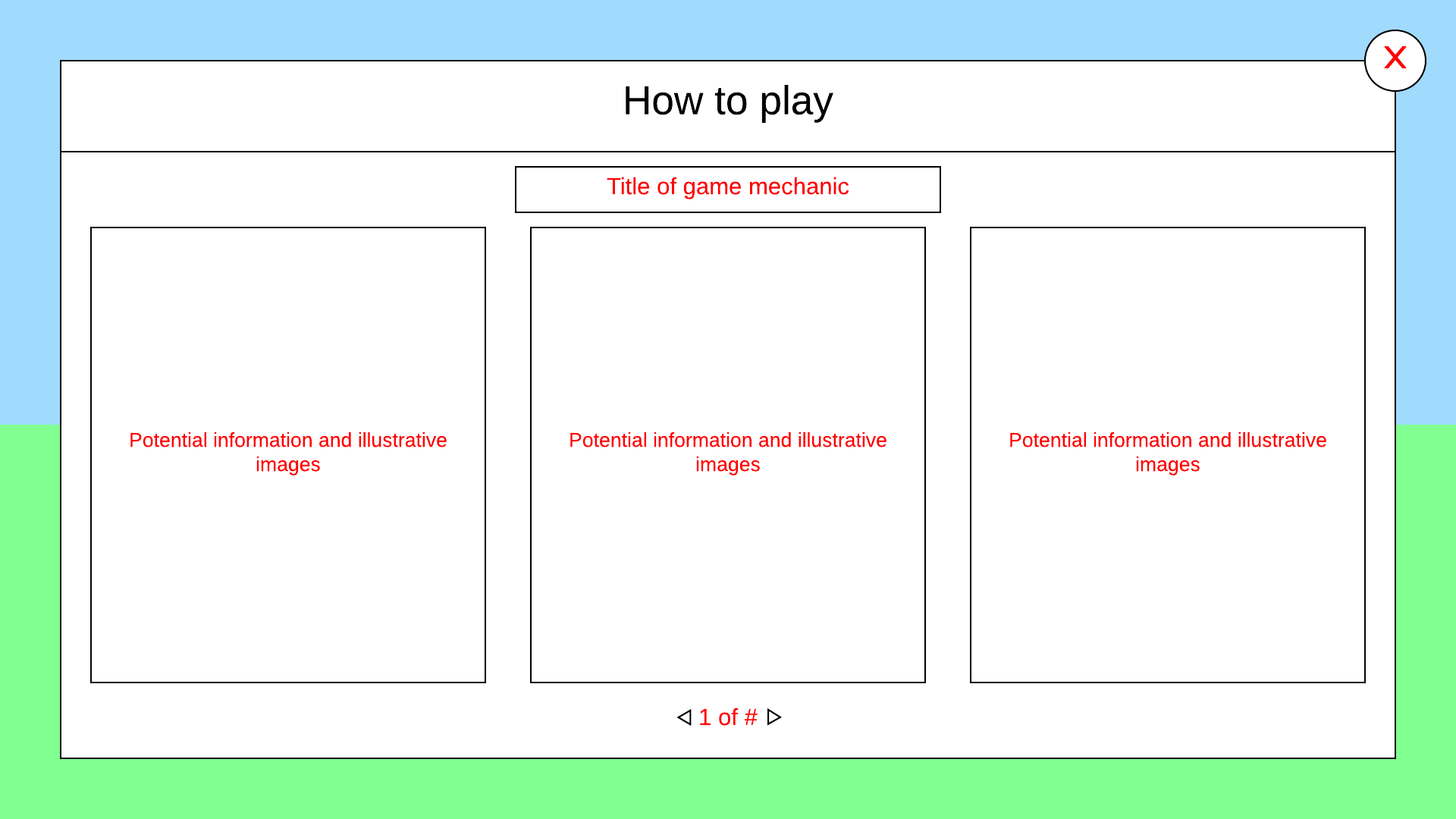
Main menu --> Load Game:



Main menu --> Options:



Main menu --> How to play:



These arrow pointers will be used to change between pages of information.

### Menu mock-ups (Output):

Game screen:

Here is a screenshot of the game that I am getting inspiration for my project. The game screen here is similar to what I would like to have as my game screen.

There will not be magic in my game

This is the exp bar and to reduce clutter I won’t have it on the game screen but rather have it on the exp menu purely.



Life Bar

Also, the enemies have animations as so they are not static images when moving, same as the player.

There are multiple of the same enemies in the same level and they have all the same stats but just are positioned differently when the level is loaded.

The floors have a repeated texture, and this is changed in other levels like here.



Here the player is drawn on top of the floor of the level and id drawn in front of the background art with some background detail here being this foliage.

## Main Algorithms:

**Spawning enemies:**

Enemies will spawn once, only when the player enters a string of levels and the string of levels depends entirely on level adjacency. I will use a depth first traversal to find all the levels that are connected to each other from the overworld.

Overworld map

Level

3

Level

2

Level

4

Level

1

Level

???

As soon as the player enters Level 1 from the Overworld map, then the algorithm will load the enemies from level 1 to 4 into their respective levels so that enemies that are killed won’t respawn while you are still in the dungeon.

A dictionary will be used to show adjacency like {Level 2: [Level 1, Level 2, Level 4] … } and so I will run the depth first traversal on this but when it comes to level 1 and it sees that “Overworld Map” is adjacent, it will not count it as a level and do the rest of the levels. The outcome will be a returned list of all the levels other than the overworld map in that chain which will then be iterated through and relayed into spawning the respective enemies in each level from the text files.

**Pseudocode (Recursive Depth First Traversal)**

SUBROUTINE DepthFirstTraversal (adjacency\_list, current\_node, visited\_list):

APPEND current\_node to Visited\_list

FOR Node INSIDE adjacency\_list[current\_node]:

IF node NOT INSIDE visited\_list:

DepthFirstTraversal(adjacency\_list, node, visited\_list)

RETURN visited\_list

Levels = DepthFirstTraversal(adjacency\_list of every level in game unless adjacency is to overworld, level just entered, [])

FOR level INSIDE Levels

OPEN level.txt from enemy directory

CREATE enemy objects based on the contents of that text file

APPEND enemy objects to enemy list attribute

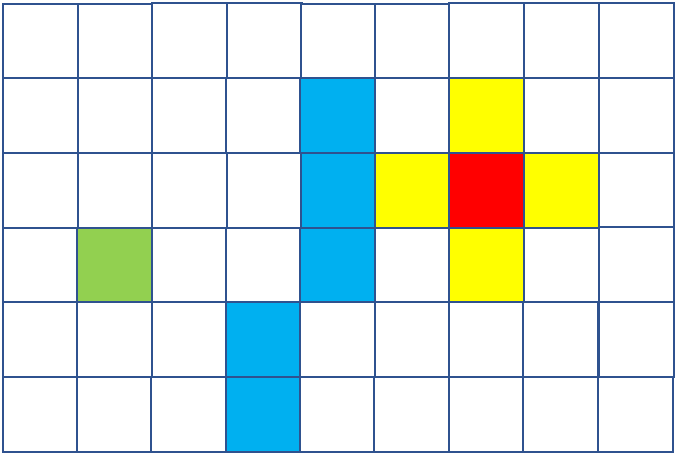
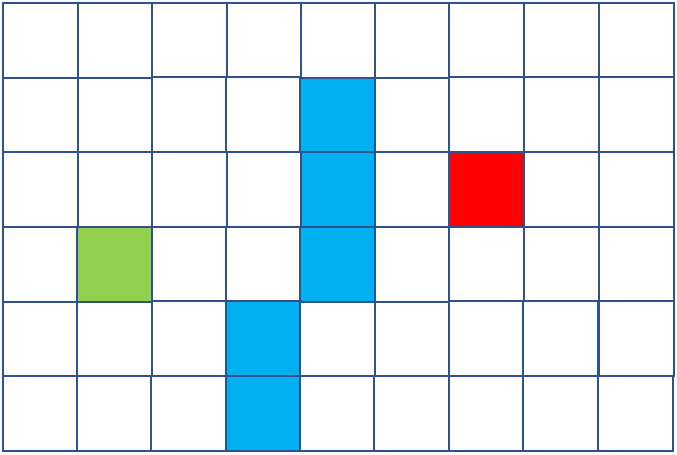
**Pathfinding:**

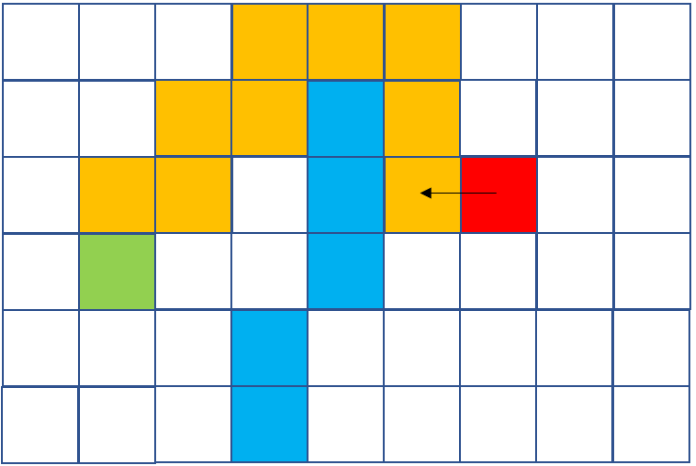
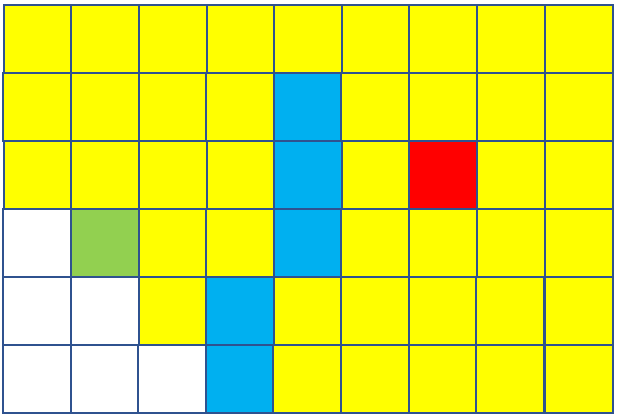
On my overworld map I want to have enemy encounters. How this would work is enemies would chase the player until they expire after a certain time or they catch the player. If they expire, they disappear and a new one will appear later but if the player is caught, then he will be kicked into a level full of enemies and has to escape.

For this to function I will need to use a pathfinding algorithm. It is a choice between A\* and Dijkstra’s algorithm but I have decided to use Dijkstra’s algorithm as my enemy will spawn in a diamond shape around the player so the enemy will always be in close proximity to the player for Dijkstra’s algorithm to be quick enough, with only ever a max of 6 tiles distance from player and enemy.

The goal of one run of the algorithm is to find the shortest route in that frame of movement so the player will not be faster than the enemy so he can catch up.

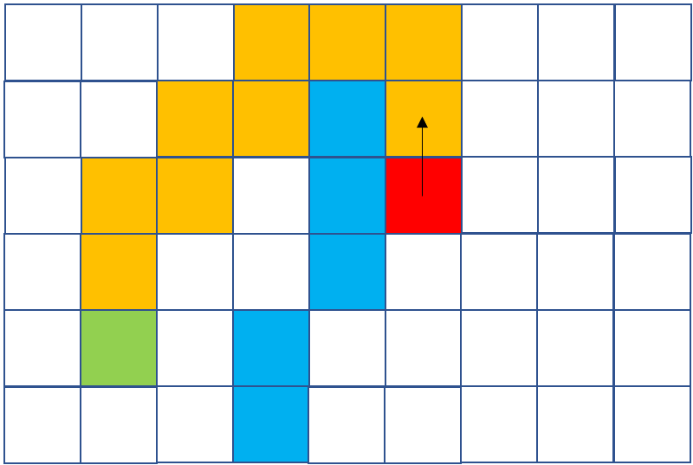
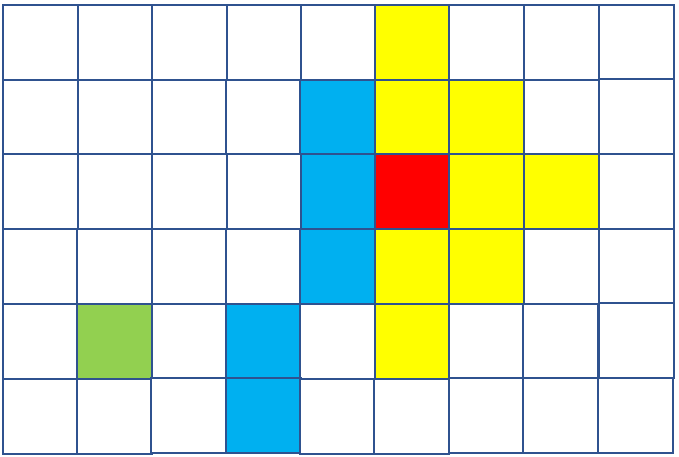
First Dijkstra is run on the enemy to find out which direction to move.





The path to the player starts by moving left.

While the enemy is moving the player can move a tile also, which is shown below so then Dijkstra is run again (no matter if the player has moved or not).



**Pseudocode**

SUBROUTINE RecurPath (spawn\_node, player\_node, path, current\_node):

If path[current\_node] == start\_node:

RETURN current\_node

Else:

RETURN RecurPath (spawn\_node, player\_node, path[current\_node])

SUBROUTINE Dijkstra (node\_queue, adjacency\_list, distances, path, player\_node, spawn\_node):

WHILE LENGTH OF node\_queue NOT EQUAL TO 0 LOOP:

POP node\_queue INTO current\_node

IF current\_node NOT EQUAL TO player\_node:

FOR node IN adjacency\_list[current\_node]:

IF Distances[current\_node] + 1 < distances[node]:

Distances[node] = Distances[current\_node] + 1

Path[node] = current\_node

Noque\_queue = MergeSort (distances, node\_queue)

ELSE

Node\_queue = []

Next\_coord = RecurPath (spawn\_node, player\_node, path, player\_node)

RETURN next\_coord

GET spawn\_node – Random place where the enemy spawns around the player

GET node\_queue – Use depth first traversal to get all valid nodes around enemy spawn

GET adjacency\_list – From node\_queue check all 4 directions to determine adjacent nodes

GET distances – Set all of nodes from node queue to value of 999999 of distances except spawn node

GET path – Set all nodes from node queue to (0,0)

GET player\_node – Find the coordinate of the player

Next\_coord = Dijkstra (node\_queue, adjacency\_list, distances, path, player\_node, spawn\_node)

**Creating levels:**

I will have a level editor that first you would choose a level and then when you are in the editor, you click on a cell to paint with a block pallet at the bottom of the screen which will then alter the text file for that level in the position of the block that you clicked and change that block to the block you have chosen on the block palette.

Blue is selected

And along with that the text file will be updated so when the screen is redrawn, the new blocks are there:



**Pseudocode:**

SUBROUTINE ReplaceBlockInFile (coord, file):

OPEN file.txt

###Line is coord[0] and 3 letters to replace is [coord[0]:coord[0]+3]

File.txt[coord[1]][coord[0]:coord[0]+3] = self.paint\_block

SAVE file.txt

INPUT file

LOOP UNTIL EXIT

IF mouse clicked on screen:

GET coord – Coordinates of block clicked on screen

ReplaceBlockInFile (coord, file)

ELIF mouse clicked on palette:

Select block clicked

self.paint\_block = ID of block clicked, will be in form “001”

# Section 3 – Implementation

I am using python with the module pygame 1.9.4

## Completeness of Solution:

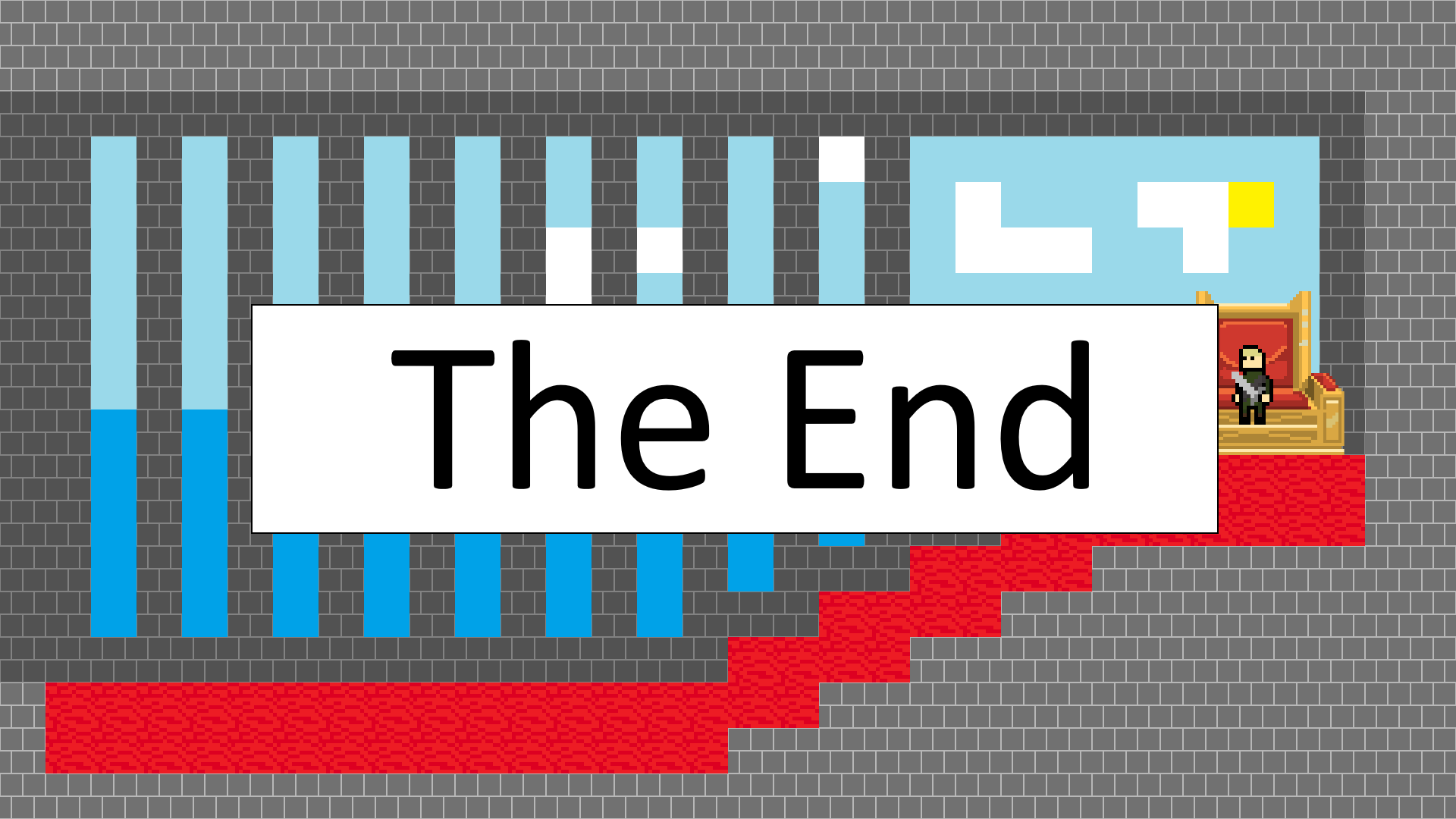
Here I will evidence my completion of my objectives from my analysis and design stages.

**Implement OOP in a variety of ways**

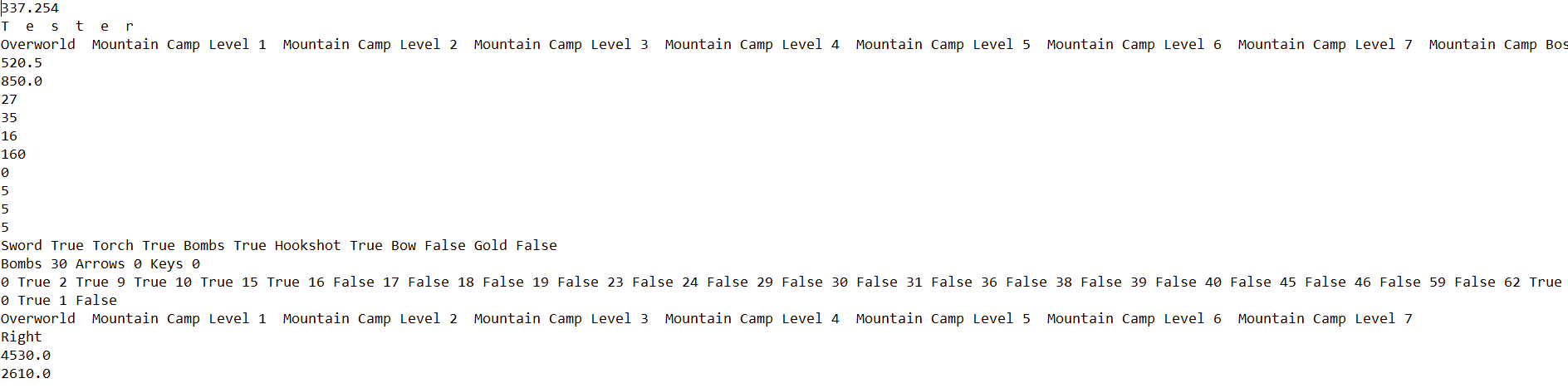
Look down into technical skills used for evidence, I have used: Inheritance, polymorphism/overriding, static and dynamic generation of objects.

**Complete a full, comprehensive and coherent storyline**

The game contains a couple of cutscenes and conversations all about the storyline and it is complete with the final frame of the game here:

****

**Create a save/load system that works**

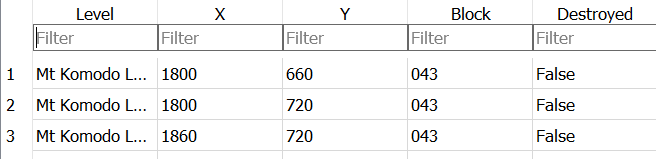
When the game is saved, all necessary data is saved into a text file dependant on which profile the player is playing on:



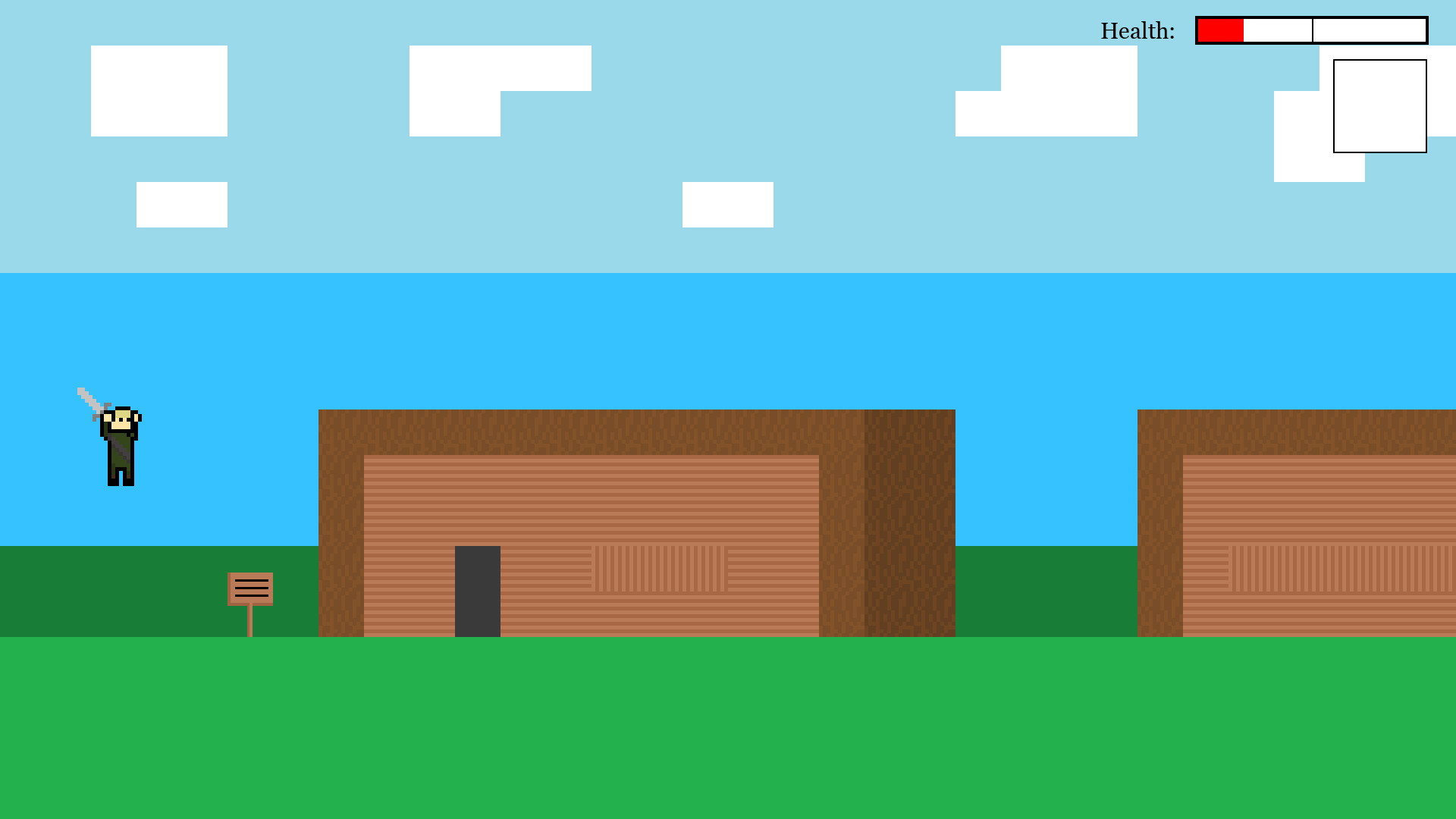
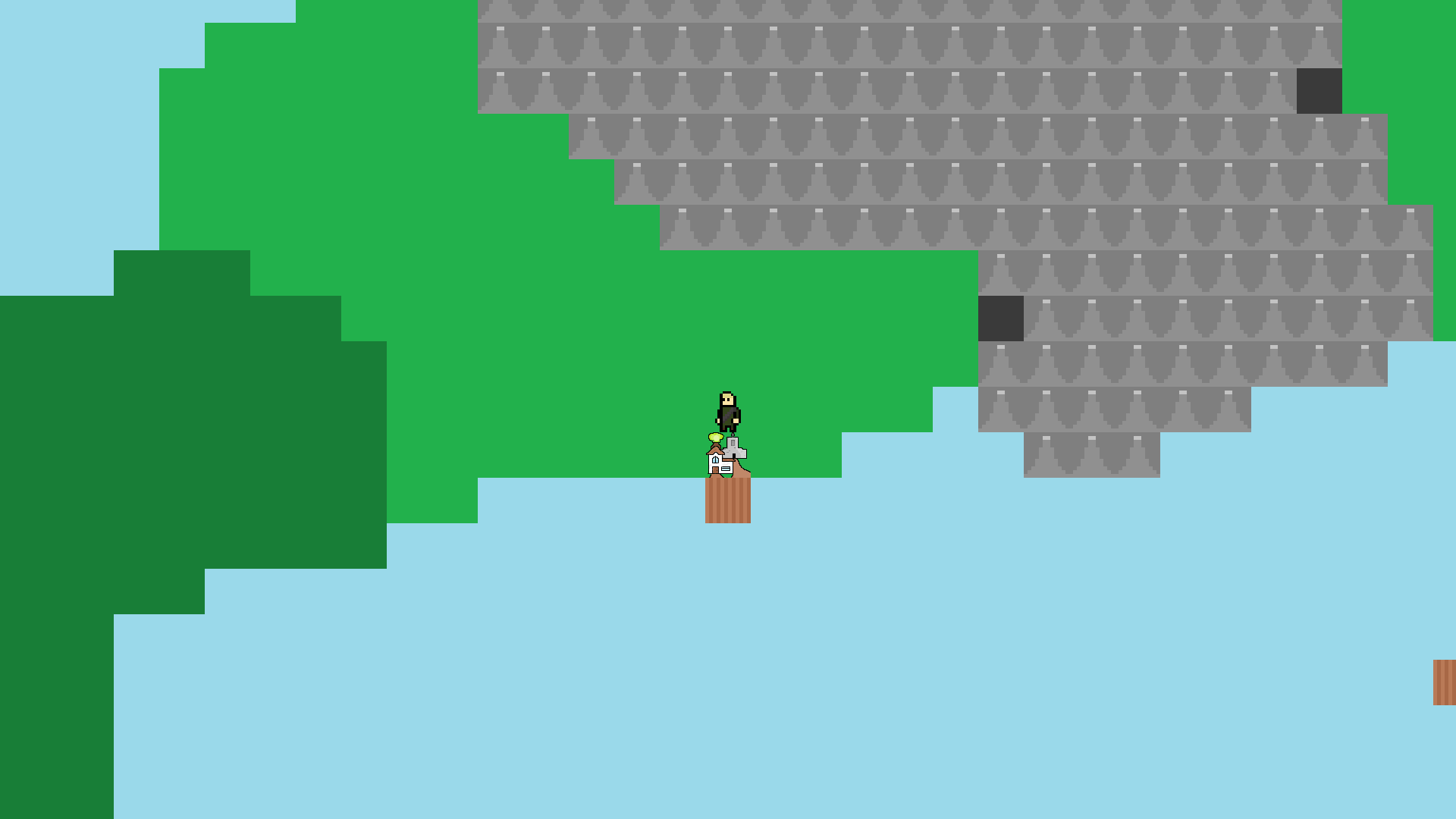
(**Python code - Line 8601** – For all

Attributes that are saved)

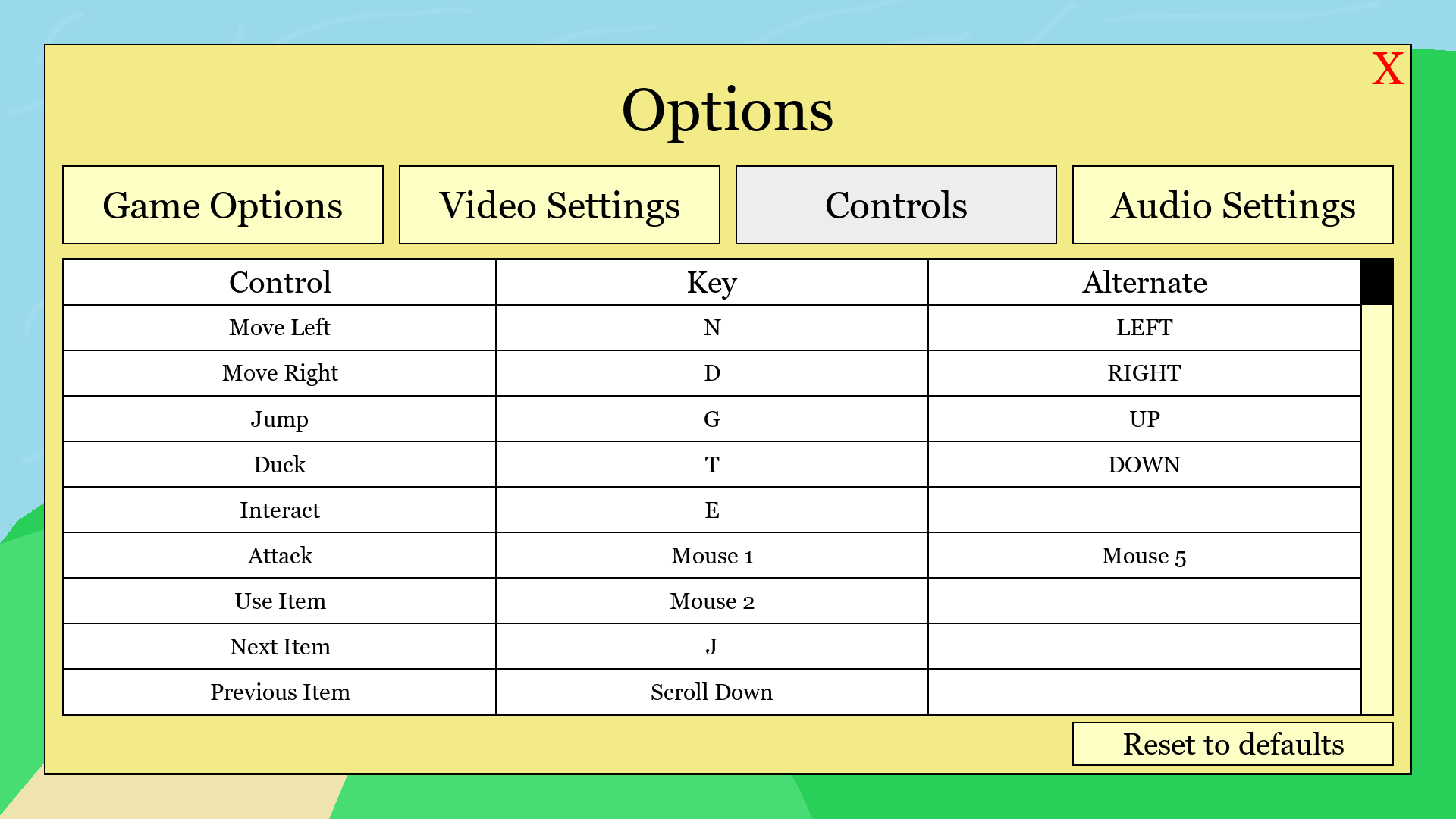
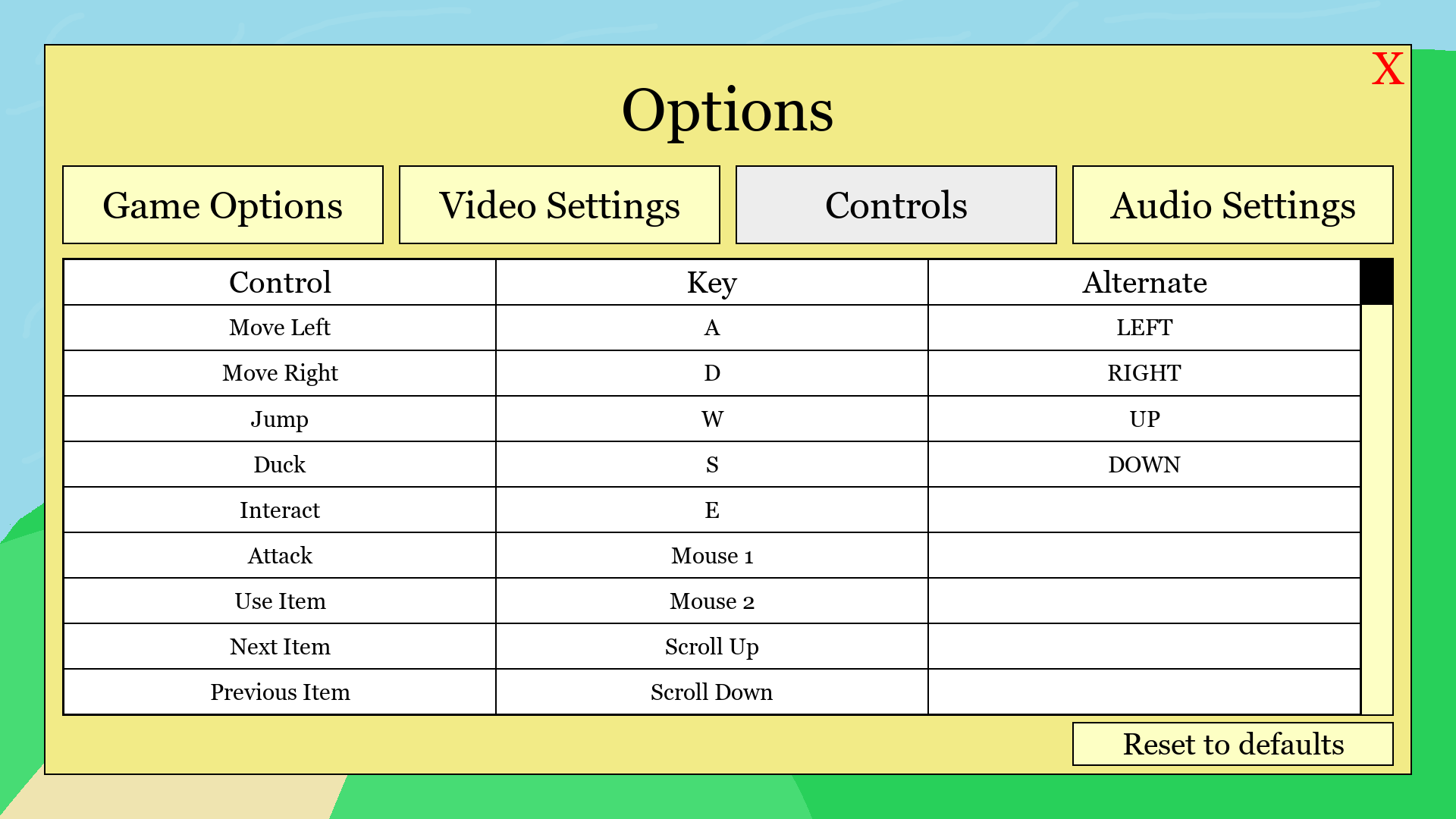
Also, a database containing data on if breakables walls are destroyed or not are updated:



**Have my game able to display different screens, one being the overworld and one being when inside a level**

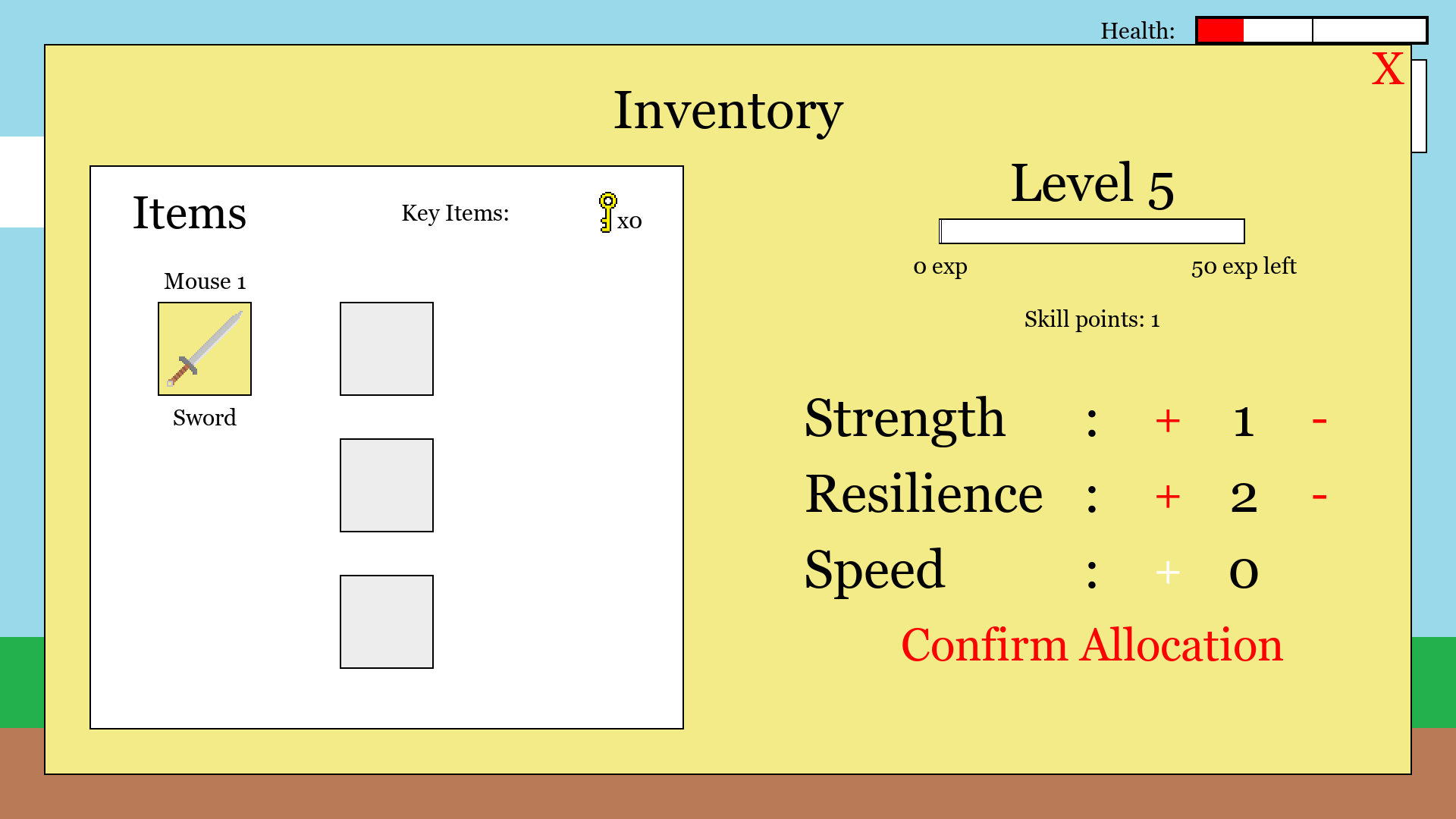
**Have controls that can be changed according to the client’s desires**



**Have a full system for enemies to give EXP and the user to be able to alter stats**

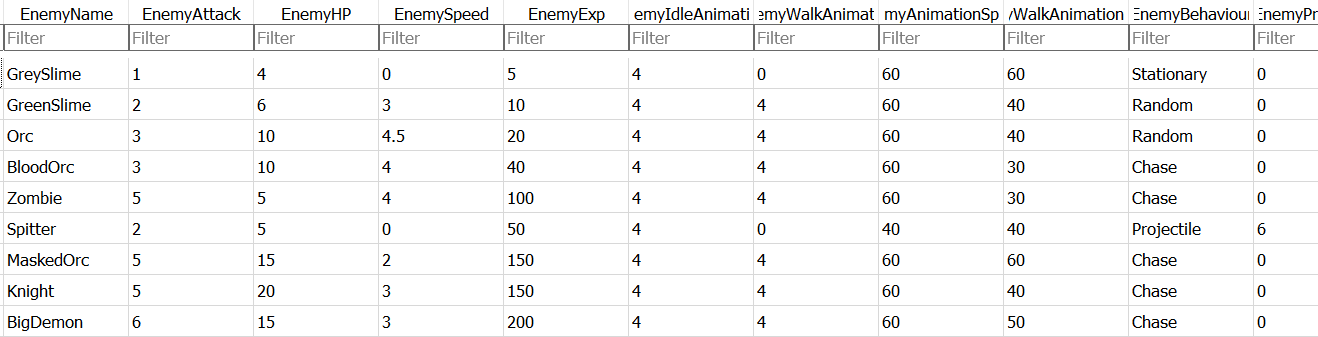
Here is me killing an enemy and levelling up and getting some skill points:

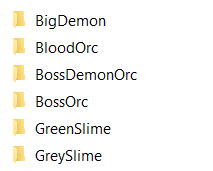
 



**Create a system to be able to add new enemies or NPCs without needing to change any code**

I have created a system for adding enemies by changing very minimal code, I need to first make a new record in the database for the enemy then add all of the animation frames to the images folder under the enemy’s name:





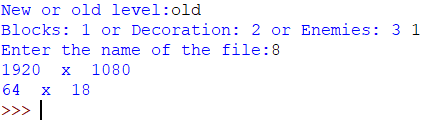
The only python code I need to alter upon adding an enemy is in my level editor code, when I want to have another enemy as an option to place down in a level I need to add one to the number of times I iterate in the function that adds buttons to select enemies:

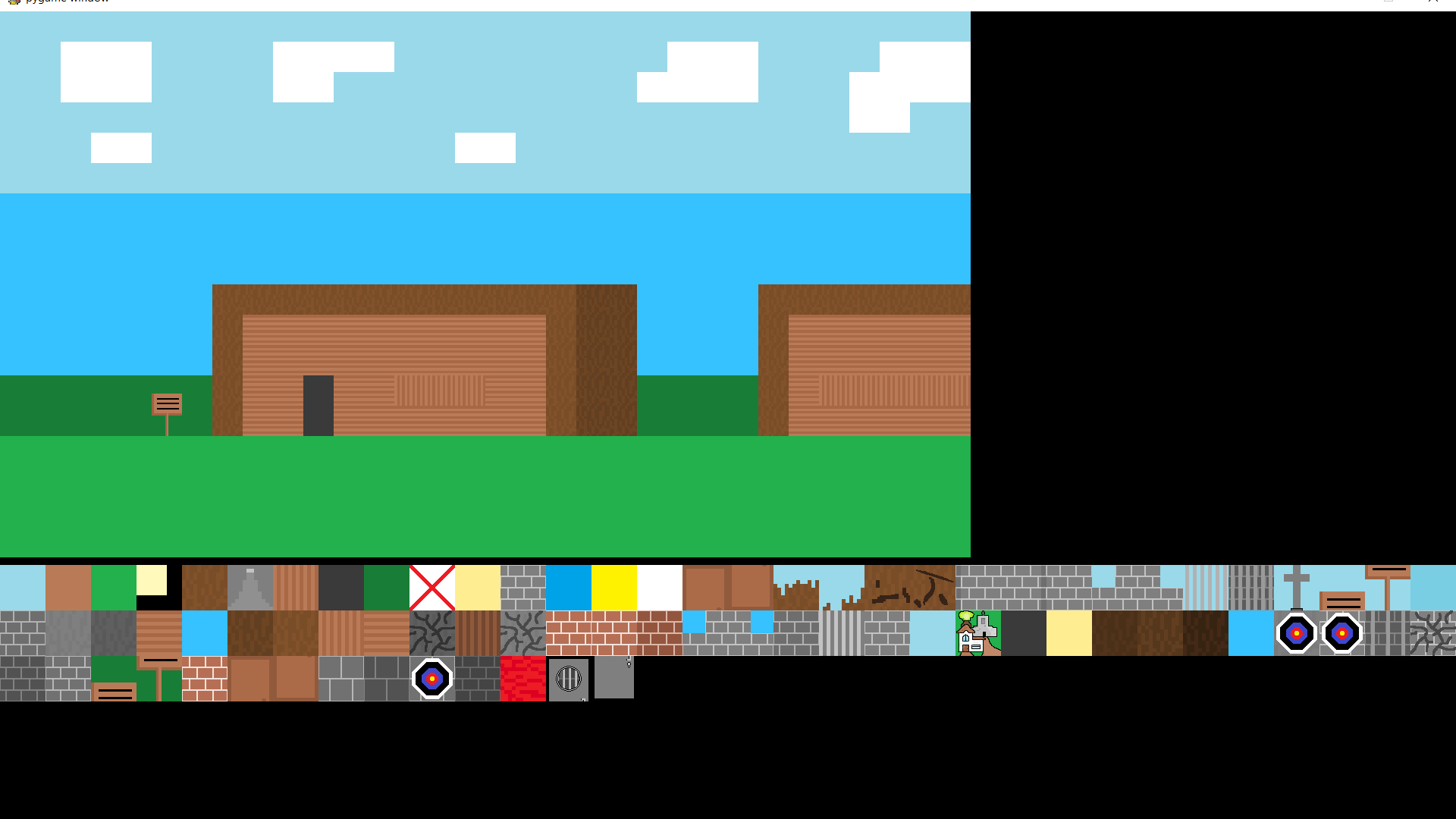
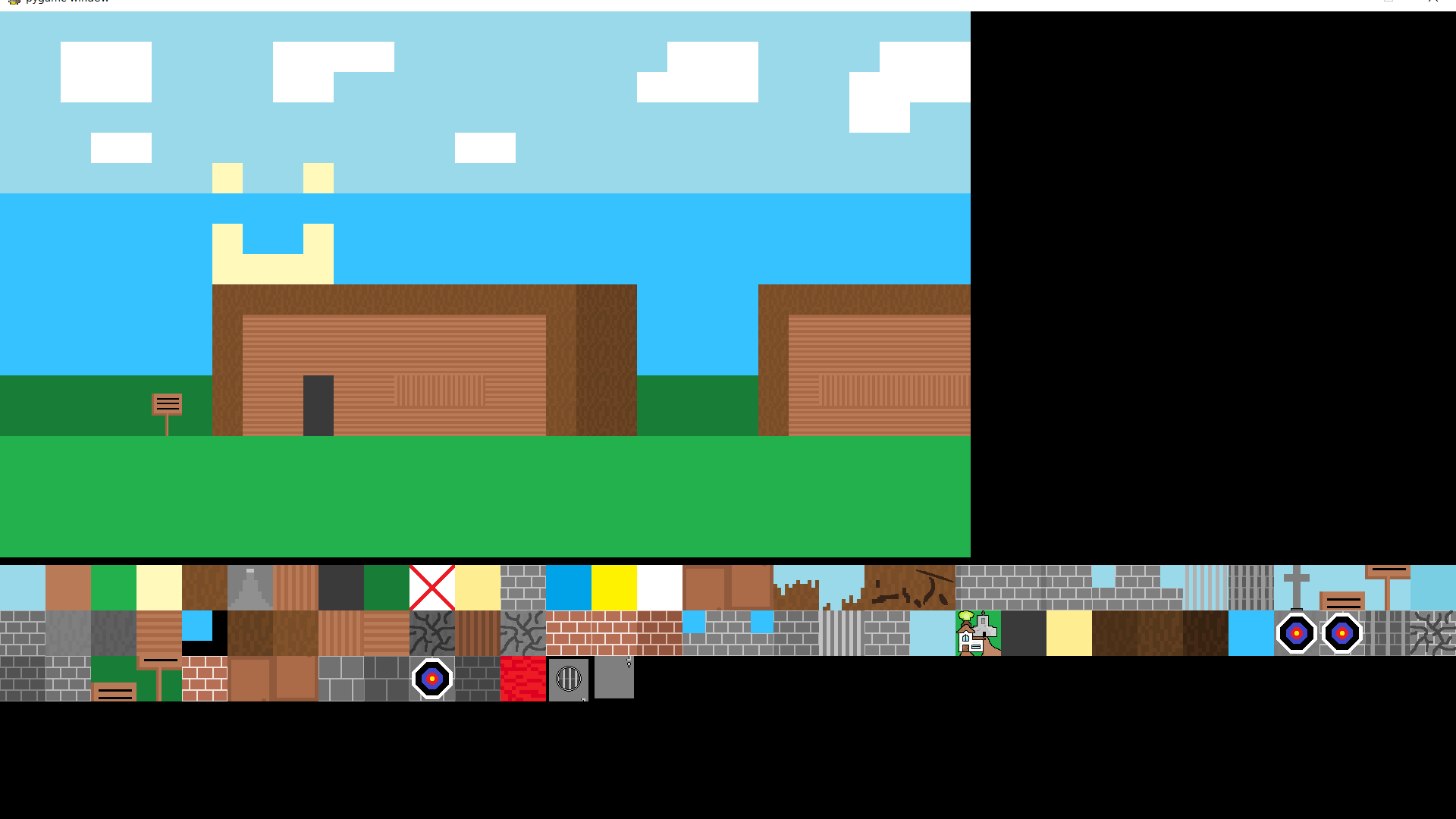
A screenshot of a cell phone

Description automatically generated

**Create an in-game map creator so levels can be created with an in-game GUI**

In a separate file but works perfectly:



**Implement an OOP structure for the main character as well as a separate one for enemies**

 (**Python code – Line 9470**)

 (**Python code – Line 10498**)

## All Important Technical Skills Used:

My source code is very long so I will include line numbers in this document to refer to parts of my code and talk about it in this document.

**Python code** – Refers to my main source code: “NEA main code – Akibul Hoque.py”.

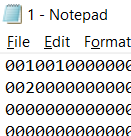
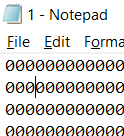
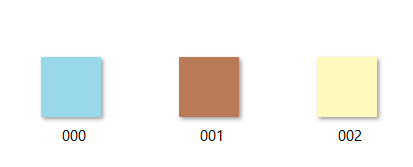
**Level editor code** – Refers to my level editor code: “NEA level creator code – Akibul Hoque.py”.

### Group A skills

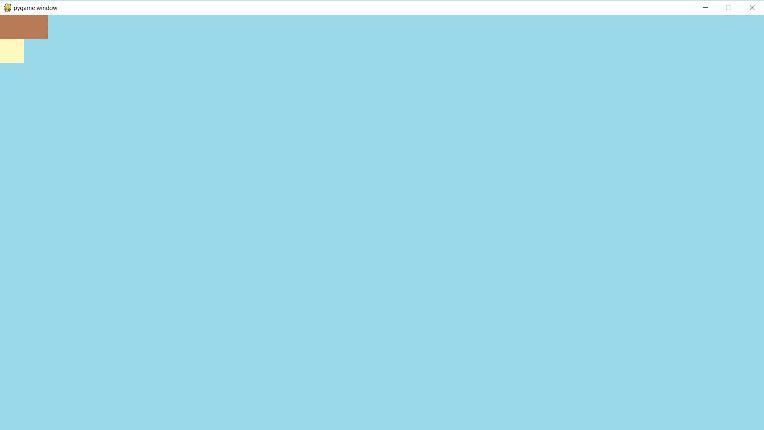
Dynamic Generation of Objects based on complex OOP model

* Walls
  + Objects generated dependant on text files in >>>/levels/<>.txt
  + **Python code – Line 9152 & 9153**
  + Text file is iterated through and for each wall, a wall object is created and added to walls list.
  + Without this skill I would need to define each wall as an object which is impossible or I would have to not have my walls as objects and instead have them being integrated with in a non-efficient way like the player checking if the screen below them is a certain colour which is not ideal.

To show how the program loads a level:



After changing the text file and as shown on the right the screen changes from the screenshot above to the one below (With the tiles codes shown on the right):



And every block is an object as shown on the code reference.

* Deco(rations)
  + Objects generated dependant on text files in >>>/levels/decoration/<>.txt
  + **Python code – Line 9117 & 9118**
  + Text file is iterated through and for each decoration there is, it is instantiated with the deco id, x and y position.
* Enemies
  + Objects generated dependant on text files in >>>/levels/enemies/<>.txt
  + **Python code – Method on Line 9237**
  + Text file is iterated through and for each enemy there is, their name is queried in the enemies database to find their behaviour to then instantiate from the correct child class of “EnemyMaster”.
  + Without this skill, making new enemies would be a lot slower as I scale my game up with more and more levels as I would have to define each enemy in each level with some levels having more or less making things a lot less efficient.

For example:

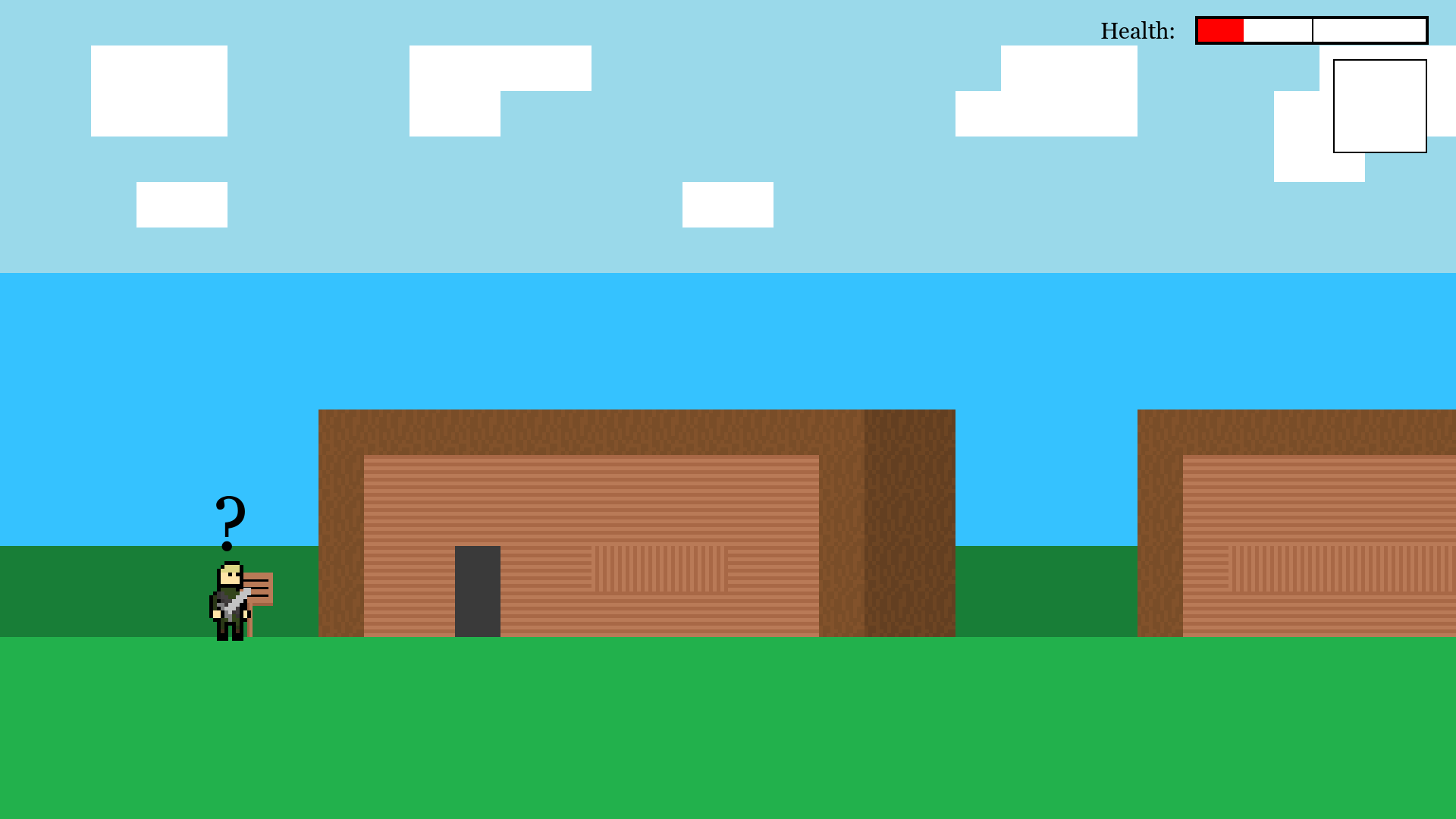




Three enemies in the text file spawn three enemies on the level and all the stats of the enemy come from the database and you can see that these enemies are shooting projectiles.

* Interactables
  + Objects generated dependant on contents of “self.interactables”.
  + **Python code – Line 3875**
  + A list above that contains all of the interactable events and so here, all of the ones that are still valid to be interacted with are made into objects to be later handled.

Example of an interactable:



Here I interact with a sign and the interactable is loaded upon level entry from this:



**Python code – Line 3804**

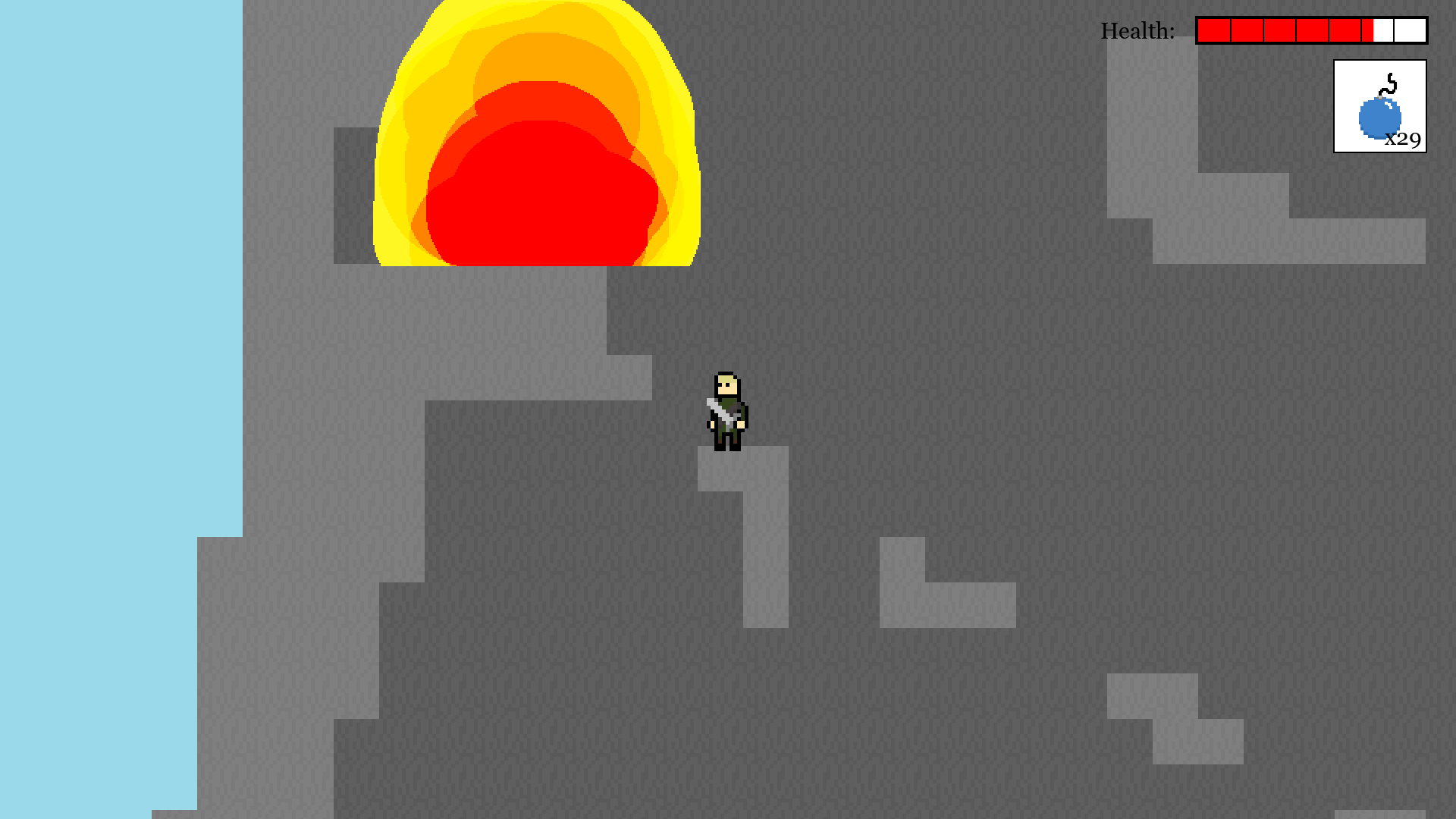
* Cutscenes
  + Objects generated dependant on contents of “self.cutscenes”.
  + **Python code – Line 3885**
  + A list above that contains all of the cutscenes, very similar to Interactables but instead trigger when you walk into the hitbox.
* Projectiles
  + Objects generated dependant on if a shooting enemy has met the conditions to shoot a projectile or if the player has shot any projectiles.
  + **Python code – Line 9319 & 9321**
  + A list above that contains all of the cutscenes, very similar to Interactables but instead trigger when you walk into the hitbox.
  + So, an enemy projectile is a dynamic object created from another dynamic object.
* OverworldEnemy
  + Object generated dependant on player time on overworld and his coordinates.
  + **Python code – Line 7729 & 7730**
  + Spawns an overworld enemy that chases the player and spawns a couple tiles away from player in a sort of diamond shape dependant on what tiles are valid to move to and from, such as sea is invalid while grass is valid.

Uses Dijkstra to chase player:



* TemporaryWalls
  + Objects generated dependant on database tempwalls and whether a record has the final item of “False”.
  + **Python code – Line 7440 & 7441**
  + A wall that can be blown up (removed) if in collision with a bomb that has exploded is added to the list “self.temp\_walls”.

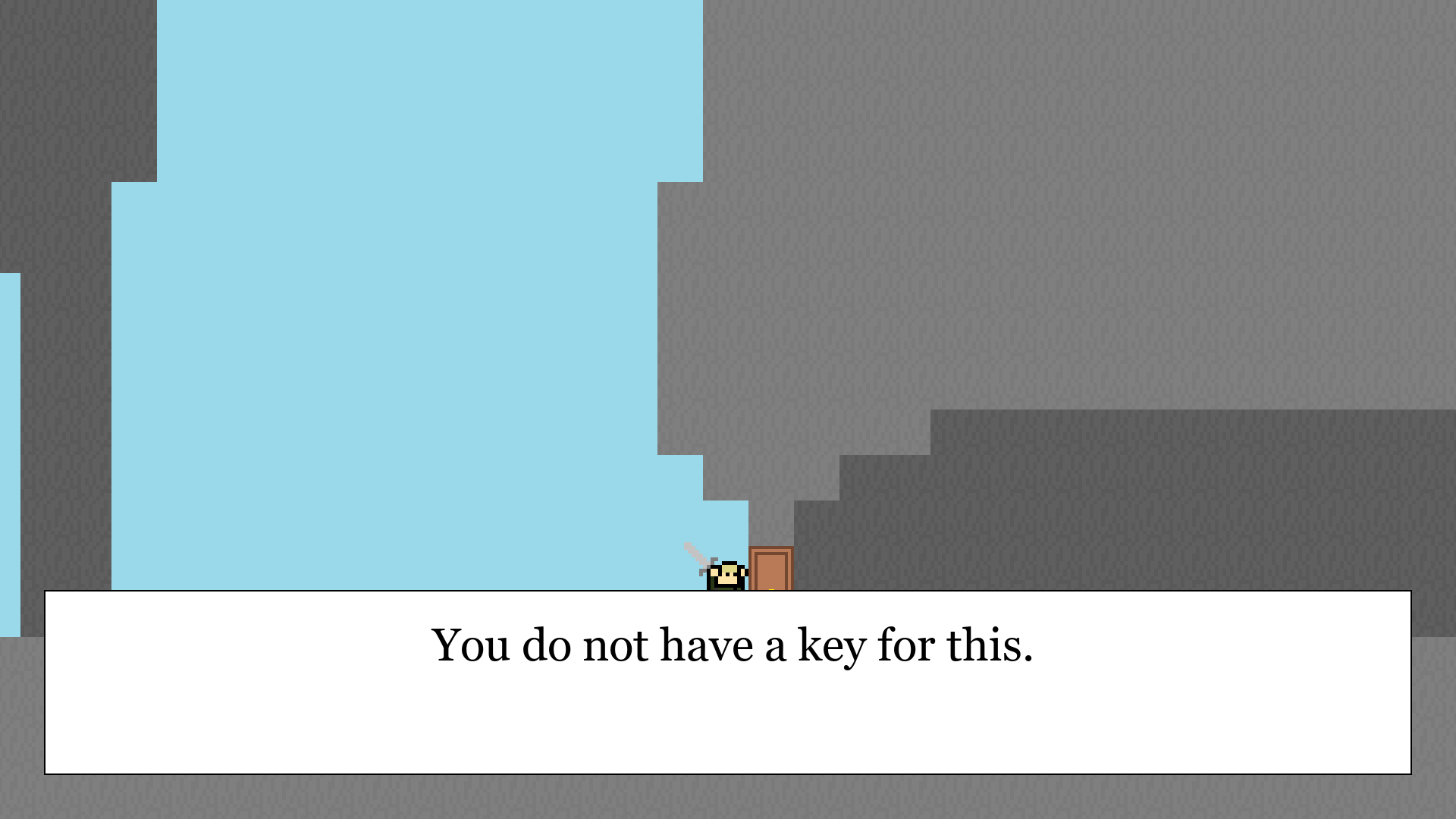
Here is a breakable wall in work:





* KeyDoor
  + Objects generated dependant on database tempwalls and whether a record has the final item of “False”.
  + **Python code – Line 7437 & 7438**
  + A door that can be opened if the player interacts with it and has a key in their inventory.

When interacting with a door either, first screenshot is if you have no key and second and third is if you do have a key in your inventory:





Inheritance

Each subclass follows the mother-class directly

* Buttons - Mother class, has methods for drawing and detecting if mouse is over and if button is clicked – **Python code – Line 3103.**
  + MenuButton – Changes menu state.
  + SubMenuButton – Changes sub-menu state (In options).
  + AttributeButton – Changes the attribute of an option.
* EnemyMaster – Mother class of enemies, different behaving enemies perform different actions, but all have the same methods for collision with different things, for drawing and for basic movement dependant on x and y velocity – **Python code – Line 10498.**
  + StaticEnemy – X and Y velocity never change and the enemy just stares in the player’s direction.
  + RandomEnemy – X and Y velocity are randomly decided at different times and does not always just stare at player.
  + ChaseEnemy – X velocity dependant on if the player is on the left or right of enemy.
  + ProjectileEnemy – Stares at player, only one enemy of this type moves but after a certain amount of frames, shoots a projectile.
    - Without this inheritance, coding in different behaviours would be much different, needed many more methods each with different identifies to do the same thing but with slight alterations so no overriding can take place to make space complexity much worse.
* ProjectileObject – Mother class of all projectiles contains common methods such as drawing, moving and setting whether the projectile is dead or not – **Python code – Line 11732.**
  + PlayerArrows – An arrow let out from the player’s bow that hurts enemies.
  + PlayerBombs – A bomb put down by the player and can damage enemies, the player and can destroy cracked walls.
  + PlayerHookshot – The hook of the hookshot that can latch to a target block and reel the player in.
  + EnemyProjectile – Bullets that some enemies shoot out that can damage the player.
    - This inheritance allowed me for polymorphism later of behaviours of these projects as they have a lot in common with each other.

Polymorphism/Overriding:

* Child classes of EnemyMaster
  + Each contain the method “EnemyMain()” which differs in different classes, calling different methods in each variation but the different variations are all called from one place.
  + Called from – **Python code – Line 10598**
  + Method is on – **Python code – Line 10983 & 11012 & 11062 & 11106**
* Child classes of ProjectileObject
  + Each contain the method “ProjMain()” which differs in different classes, calling different methods in each variation but the different variations are all called from one place.
  + Called from – **Python code – Line 7528 & 7652**
  + Method is on – **Python code – Line 11898 & 12087 & 12266 & 12346**

Dijkstra’s Algorithm

Uses a dictionary value for each coordinate to compare distances and to sort into priority queue.

**Python code – Line 11514**

Merge Sort

* Sort queue in Dijkstra’s algorithm into a priority queue.
  + **Python code – Line 11543**
  + This was pretty much compulsory due to the nature of my Dijkstra’s algorithm as my algorithm had to handle a lot of nodes (36 max nodes) and also I had to run the algorithm about twice a second on the overworld map to find out the direction the enemy should move so if I used a less efficient and slower algorithm then my program would have some heavy slowdowns.

Graph Traversal

* Depth first traversal – **Python code – Line 13176 (This used twice)**
  + Dijkstra’s algorithm uses depth first traversal to get all the nodes for the node queue **Python code – Line 11530**
  + Depth first used in my levels to determine all levels connected together in one location of the overworld to spawn all enemies once at start – **Python code – Line 8964 & 9002.**
    - Without this enemies would respawn upon any level changing making my game much harder to beat from the player’s perspective.

Recursion

* End of Dijkstra’s Algorithm to recur path.
  + **Python code – Line 11545**
  + To find final node, which is the node next to the enemy’s start position.
* Depth first traversal method is recursive, making the algorithm very simple – **Python code – Line 13191.**

Queues and stacks operations

* Queue for Dialogue
  + Object generated dependant on which text file has been passed in. Used in many places in my events part of my main function (GameState), a couple of examples:
  + **Python code – Line 4428 or 4456 or 4504 or 4571…**
  + Grabs all text from the text file passed in and then passes words into a sentence and checks width, if the width is bigger than the text box then make a new line for text or reserve that text for the next box.
    - Without this I would need to program each box of text in my game instead of being able to just type out an encounter and tell the program that there is that encounter there, slowing down production a lot and making my code longer and inefficient
* Queue for Dijkstra’s algorithm
  + Node queue (which is sorted into a priority queue) and passed into the algorithm.
  + **Python code – Line 11530**
* Stack for entering the player name in new game
  + The player types their name and if they backspace then the last letter entered is popped.
  + Pop and appending on – **Python code – Line 824 & 829**
* Stack for the player’s location
  + Instantiated on – **Python code – Line 3200**
  + Handled on – **Python code – Line 8689 & 8697 & 8717 & 8723**
    - Handled dependant on the dictionary attribute level\_exits.

Aggregate SQL function:

* Used COUNT to find how many records are in a file and then use that number to calculate a limit to scrolling in the controls menu – **Python code – Line 12828.**

Cross-table parameterised SQL:

* In settings.db all 4 tables (tblAudio, tblControls, tbl Game, tblVideo) include a replica table of the default controls. The first field acts as a foreign key for each table and when the player clicks “Reset to Defaults” in the game, the normal table’s fields gets assigned as the default values from the default table – **Python code – Line 12789.**
  + This is a feature that is present in pretty much every game so having it in my game is very important.

List operations:

* .append()
  + Used multiple times for: Stacks, queues, appending objects to lists.
* .pop()
  + Used multiple times for: Stacks, queues.
* .remove()
  + Main use for removing specific objects from a list attribute.
* .split()
  + Main use when reading from text files and sorting the separate lines of text into separate items in the list.
  + Main examples: **Python code – Line 9060 & 9072 & 9084 & 11190**
* Len()
  + Used in location to find the final item in the list (len(self.location)-1).

### Group B skills

Multi-dimensional arrays

I will include my most complicated ones.

* Interactables 2D list
  + **Python code – Line 3781**
* Cutscenes 2D list
  + **Python code – Line 3836**
* Key door 2D list
  + **Python code – Line 3858**

Dictionaries

* List inside a dictionary inside a dictionary (For level adjacency)
  + **Python code – Line 3375**
* Majority of dictionaries are on **Python code – Line 3273 to 3777**

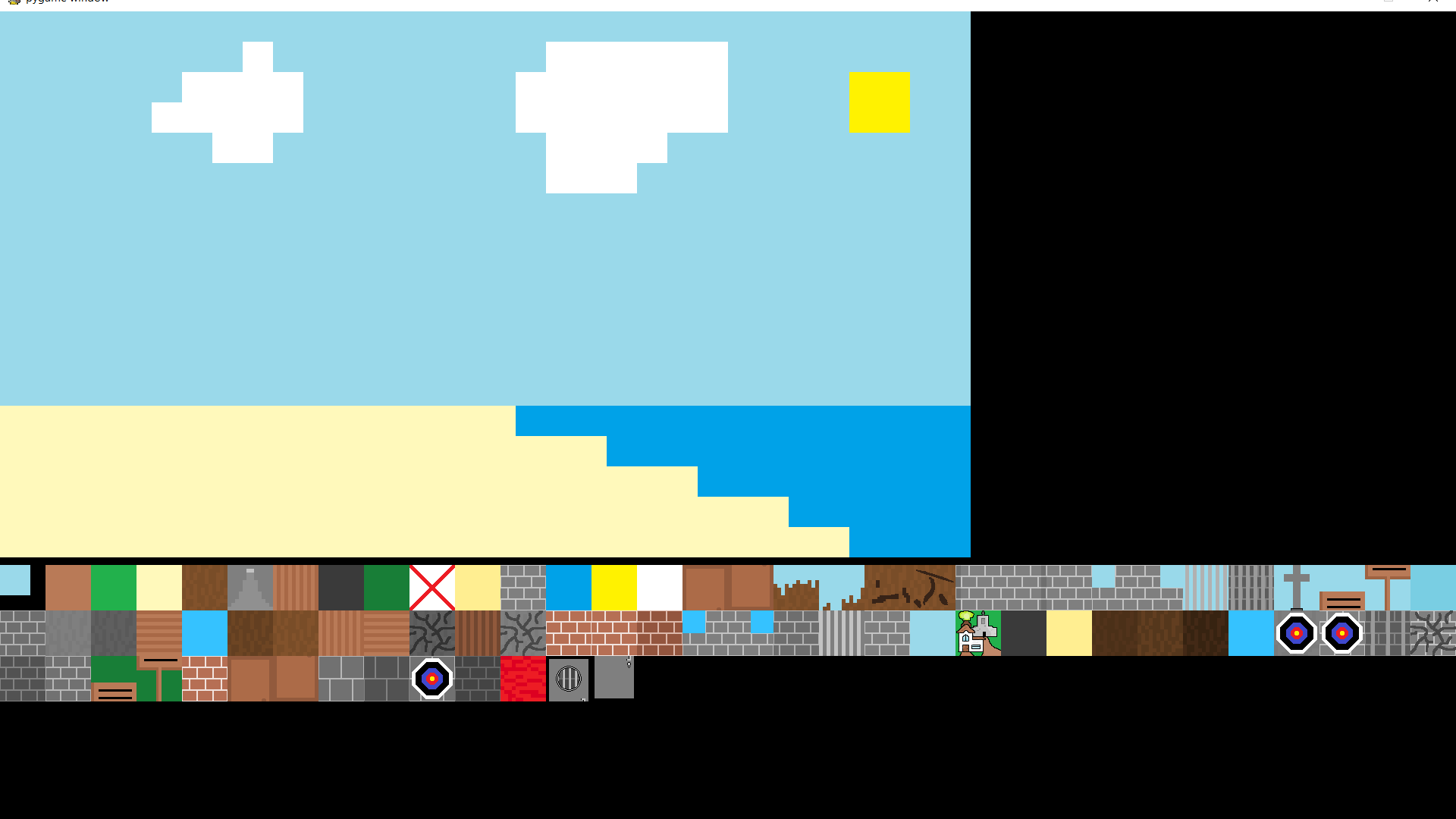
Basic OOP

* Main menu (Screen)
  + **Python code – Line 21**
  + Just contains all the methods for displaying buttons and navigating through the main menu.
* Game
  + **Python code – Line 3180**
  + Contains the main game loop and majority of game related method.
* Player class
  + **Python code – Line 9470**
  + Contains methods and attributes.
* Sword class
  + **Python code – Line 10475**
  + Makes the sword an object so that the attributes can be taken easily.

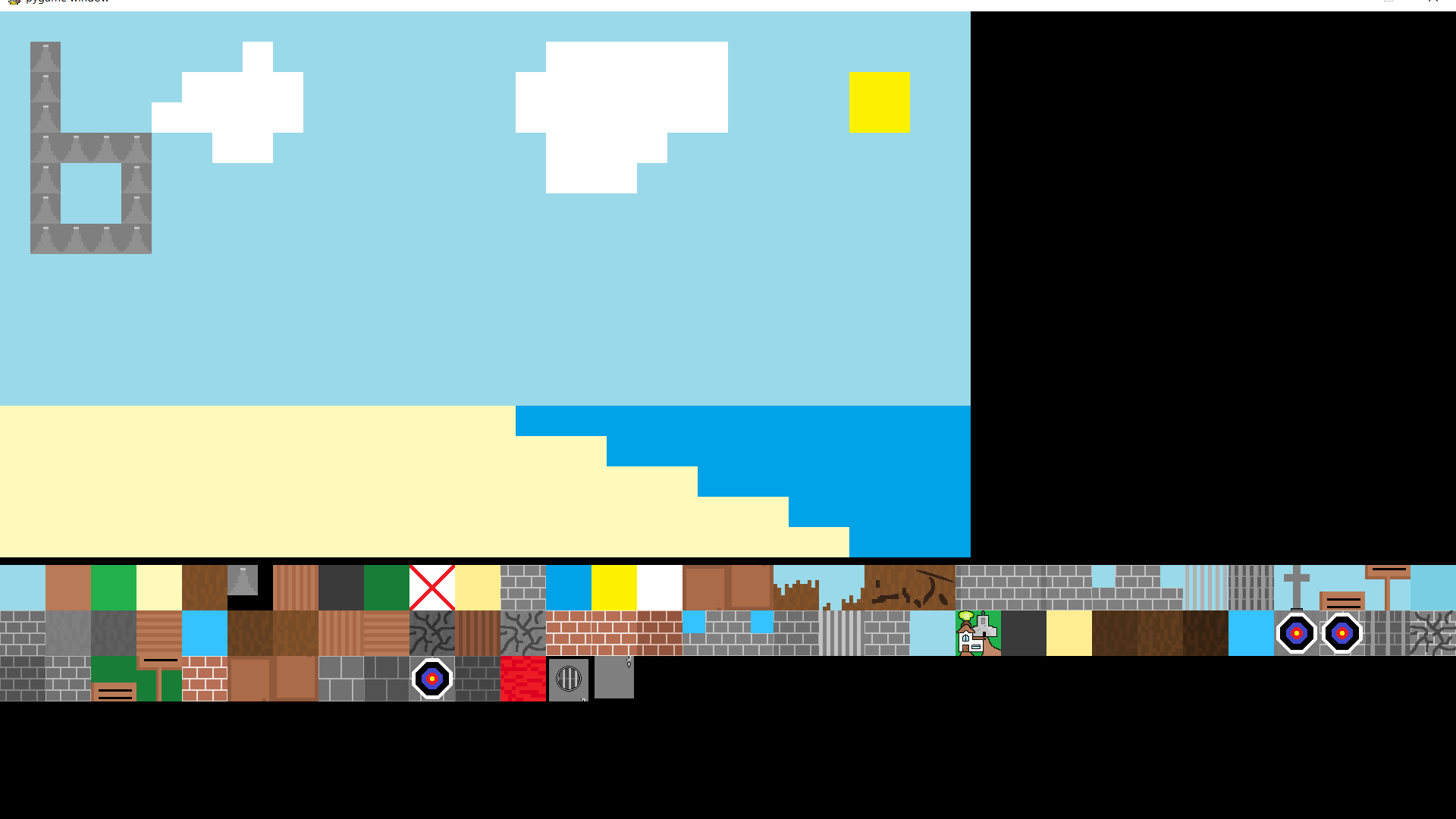
File handling

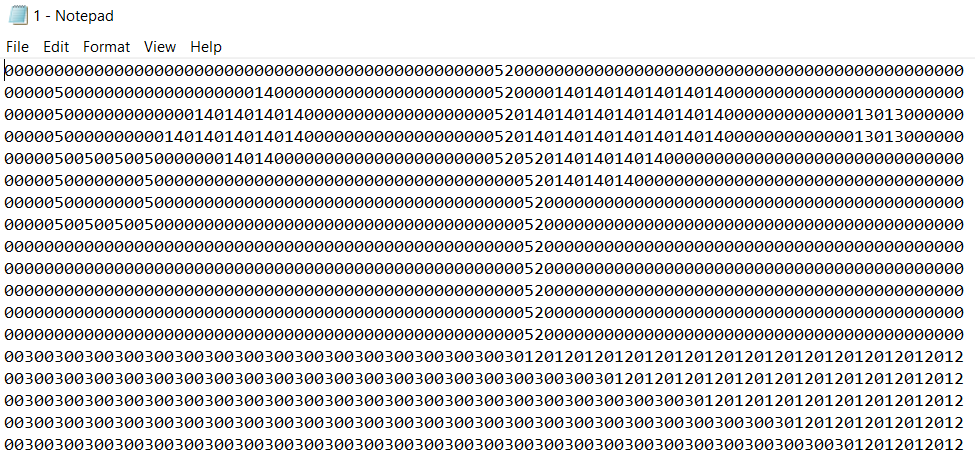
* Level editor that writes to a text file based on
  + **Level editor code – Line 245**
  + First, I choose a level to edit, whether it is the blocks of that level, enemies or the decorations I want to edit and then I can choose where in the screen I click, grid 32x18, and scroll using arrows and it writes to the levels file dependant on what block I am drawing with and where on the screen I click.

First:



After I draw:





## Research/code sources:

Resolution Fix for Fullscreen in pygame using ctypes (Using library once without understanding): <https://stackoverflow.com/questions/39274460/pygame-fullscreen-display-flag-creates-a-game-screen-that-is-too-large-for-the-s>

* So Fullscreen fits the screen properly in pygame

SQL query fix:

<https://stackoverflow.com/questions/13880786/python-sqlite3-string-variable-in-execute>

* So that variables in python can be put into SQL queries

Pygame.event.wait() to get a single keyboard input: <http://www.poketcode.com/en/pygame/keyboard/index.html>

* For changing controls in controls menu

Rotating images in pygame:

<https://gamedev.stackexchange.com/questions/126353/how-to-rotate-an-image-in-pygame-without-losing-quality-or-increasing-size-or-mo>

* For rotating my hookshot projectile and arrow projectile dependant on where the user clicks

# Section 4 – Testing

## Video Testing Timestamps

Along with this document is included a video text file that is the main testing and timestamps for that video are here (Along with extra comments not in video):

(Alternate public YouTube link for video: <https://youtu.be/YK-E8iU3-Rc>)

0:15 : Database handling for settings and controls

2:51 : Overworld and level screen being separate plus interacting

3:05 : A cutscene being triggered

3:26 : Sword is gotten

3:36 : Overworld and Dijkstra with enemy encounters

4:07 : Enemies being loaded dynamically from a text file

4:57 : Exp and skill points system

5:41 : Save / Load working

6:33 : Dialogue queue overflow handling

7:35 : Enemies loading correctly using depth first graph traversal

8:28 : Database handling for destructible walls

10:06 : Enemy drops occurring as dynamic objects

10:23 : Pick up a key

10:28 : Key door and hook shot item

12:11 : Enemy projectiles

12:23 : Boss + Database handling for boss doors

13:11 : Overworld enemy Dijkstra respawn (Because I set a max distance)

13:24 : Projectile collision with wall making it be removed as an object, and if moves off screen

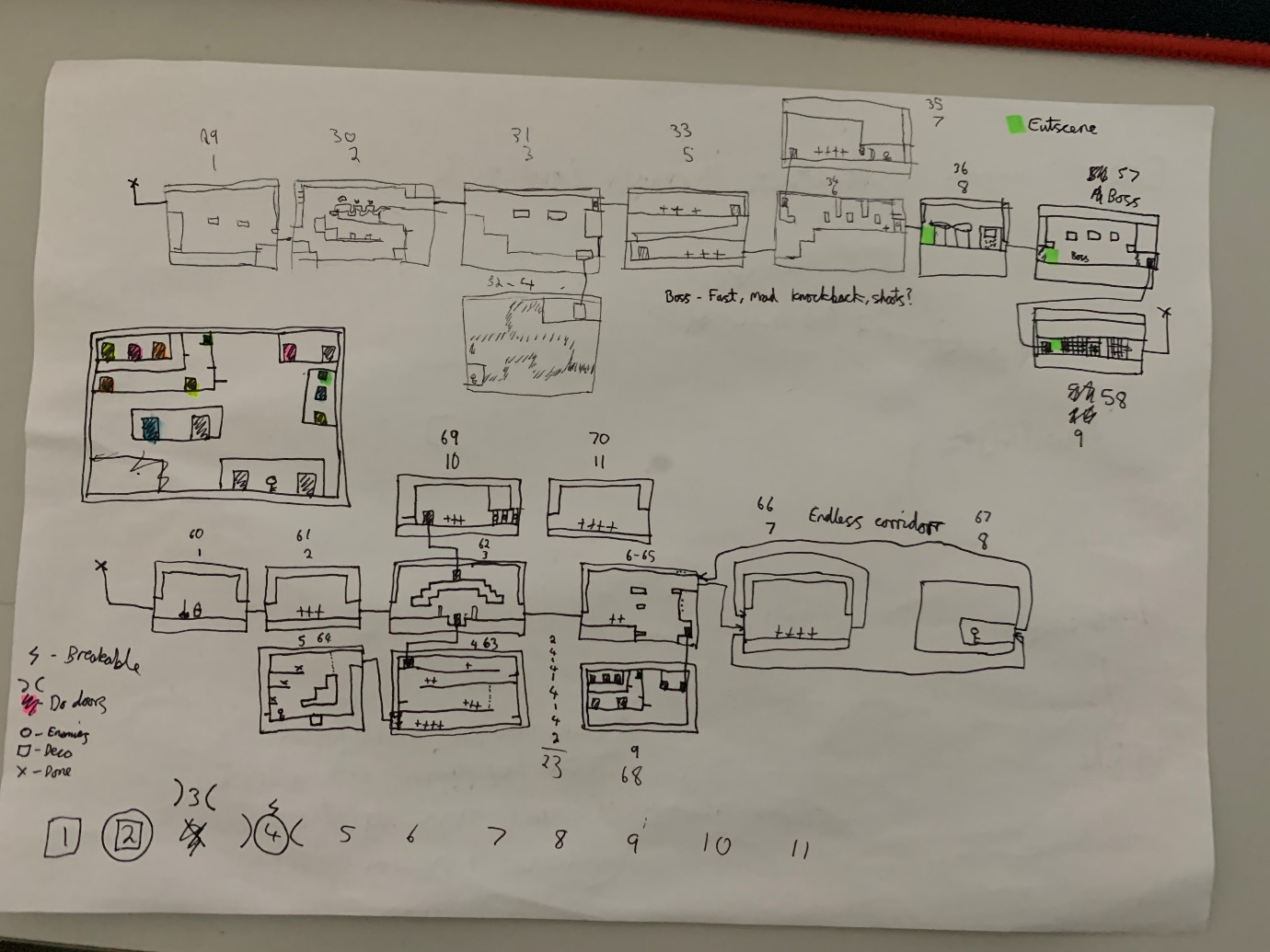
13:40 : Bow & Arrow being obtained and used

## Manual Testing

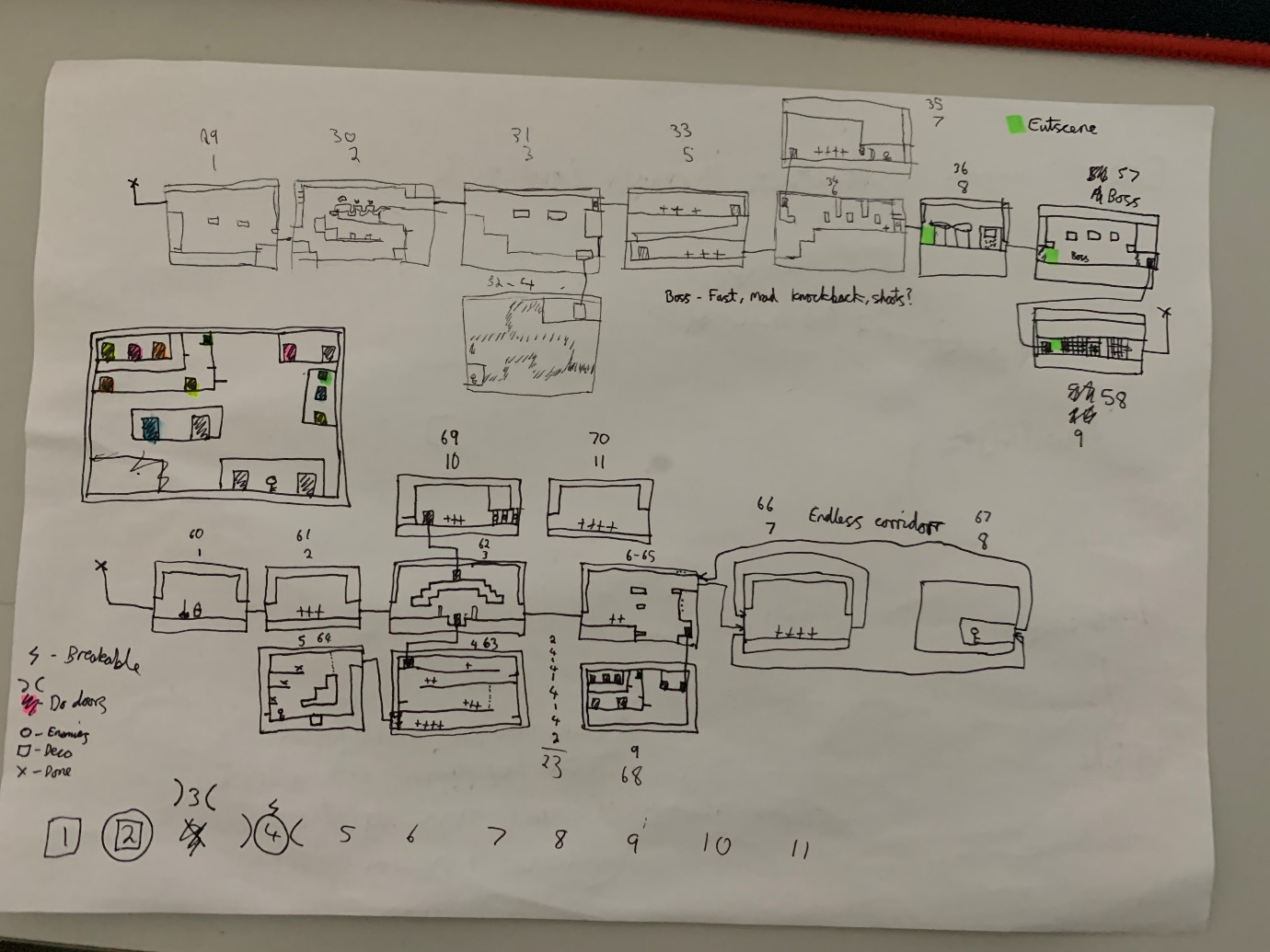
To check my depth first algorithm works as intended I printed it’s results when I loaded my game in “Research Lab Level 6”:



Here is the image of my level design with the level that they are above or beside:



So, if we follow the algorithm out on this direction then we do see that the algorithm did work correctly:



Branch 4

Branch 3

Branch 2

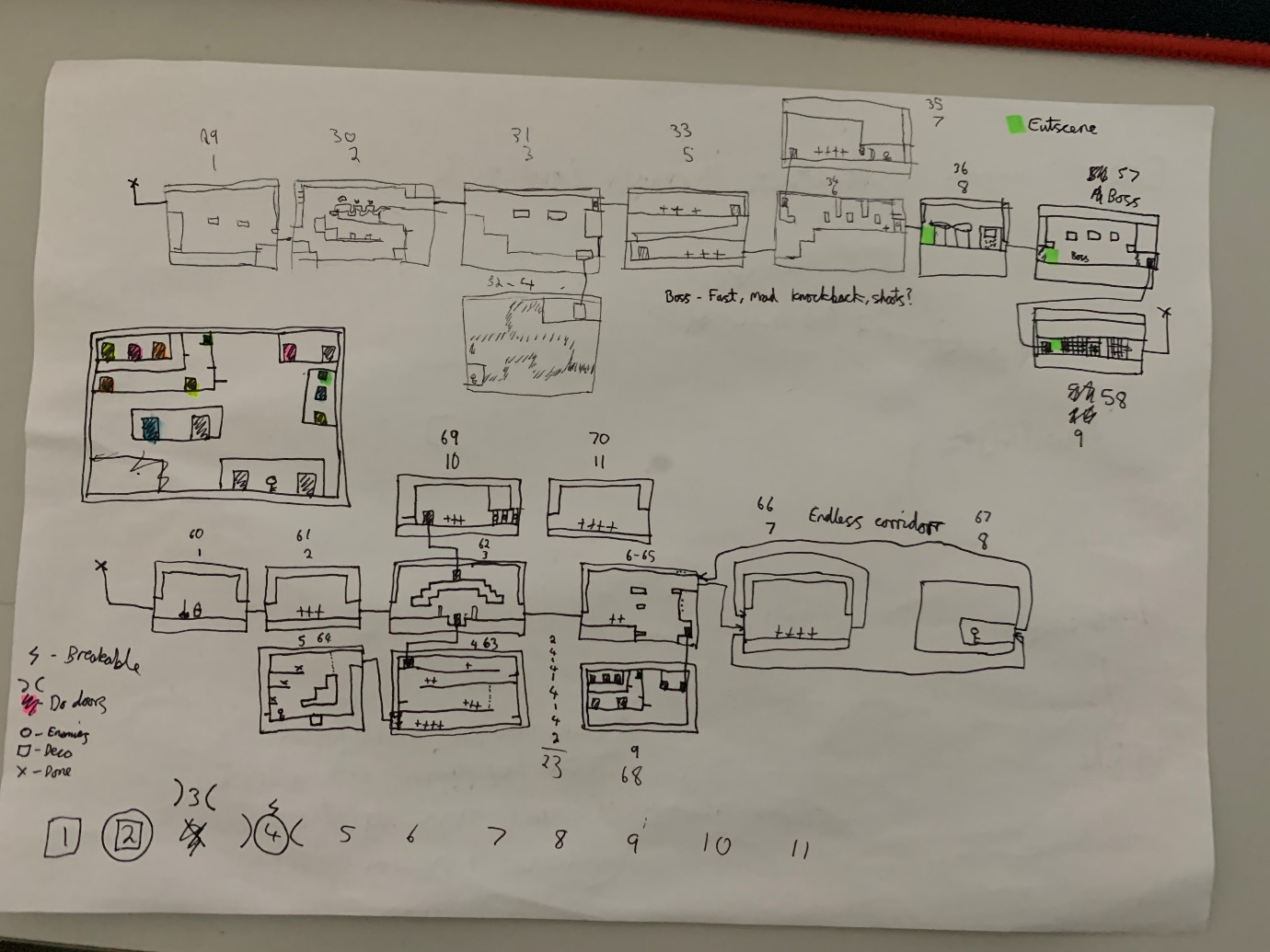
Branch 1

Start node

Then I did it from the start of the level and I got this:



So, it is instead like this which follows the algorithm correctly:



Branch 3

Branch 2

Branch 1

Start node

# Section 5 – Evaluation

Reflection on all the objectives I have set for myself tell me that I have met everything that I had said I wanted to achieve. My implementation section provided the evidence that I have achieved all of those objectives, however playing my game through, I do feel some inconsistencies with what I set out to accomplish in a final product. I got some informal feedback from people around the age group that I targeted to play the game.

“The game is pretty hard at the beginning.” This relates to the skill curve I was aiming for in my analysis. Although I did try my best to have my game progress in difficulty from an easy point by adding harder enemies and harder platforming, but it turns out the biggest issue was instead the difficulty in movement. It was slippery at times to make certain jumps in a level and you could fall and die simply. These are issues I did try my best to fix in the development process and I thought I did manage to fix them as I did improve them a lot from how they were in development so they seemed good enough to me, even though collision was one of the most difficult aspects of coding this program for me.

“There is no sound.” I overlooked this before and during development and thought that I could have a relatively good amount of time for sound design at the end of my project, but it turns out that I did not. I started my project in the earliest time that I could and I still couldn’t make time for sound; now playing through my game I do realise that sound design is something that I tend to overlook in a game and that it adds a lot to a gaming experience so it is a shame that I did not manage to get any.

A lot of aspects of my game I was planning on adding was either changed or removed completely. One of them is me removing the idea of a shield as I realised it would be very gimmicky to implement with the way enemies function in my game currently and the way I track enemy hits would make it a very difficult task to add a shield. To counter the lack of the idea I added more knockback in general to help with evasion and the fact that you can attack down one block from high up. Another major change is my initial idea of having NPCs as objects. I changed this as I found that implementing them would be very similar to how I already deal with events such as interactions and cutscenes and so adding NPCs as objects would be a waste of my time and make my game slower when I could just make interactions with people similar to how I read signs and have more complex encounters just be cutscenes with images drawn on and moving specifically.

Overall this project was a success.