Contents

[- Analysis 2](#_Toc21900520)

[1.1 – Introduction 2](#_Toc21900521)

[Background: 2](#_Toc21900522)

[How the game will play: 3](#_Toc21900523)

[The client/supervisor: 4](#_Toc21900524)

[Project Scope: 4](#_Toc21900525)

[Analysis plan 5](#_Toc21900526)

[1.2 - Investigation 6](#_Toc21900527)

[Existing games 6](#_Toc21900528)

[Invisible inc: 6](#_Toc21900529)

[The X-Com series: 10](#_Toc21900535)

[The Binding of Isaac: 13](#_Toc21900541)

[Questionnaire 15](#_Toc21900546)

[Programming languages 21](#_Toc21900547)

[Python: 21](#_Toc21900548)

[JavaScript: 22](#_Toc21900549)

[Prototyping 23](#_Toc21900550)

[Maze Generation Algorithm (verbal): 23](#_Toc21900551)

[Class diagram: 25](#_Toc21900552)

[Data Flow Diagrams 26](#_Toc21900553)

[Objectives / Constraints 26](#_Toc21900554)

[Main objectives 27](#_Toc21900555)

[Optional Objectives 28](#_Toc21900556)

# 1 - Analysis

## 1.1 – Introduction

### Background:

I will be creating a strategy based stealth game that will use a top down view of a grid map where players must complete certain objectives by interacting with certain objects across each level to progress. It will follow a similar 2D top down game format to most arcade games.

The plan is to add different terrain types and have each level procedurally generated with enemies spawning randomly but in locations which will not make the game impossible (hence procedural).

Another aspect would be to create a “fog of war” effect over each level so that the player can only see what is in locations that are in direct view of the player or sections the player has already seen.

Enemies will likely have a line of sight algorithm implemented so that they are only activated if the player is in direct line of sight of the enemy. This can be achieved by “drawing” a segment of a circle that will be the enemy’s field of view, the radius being the max view distance.

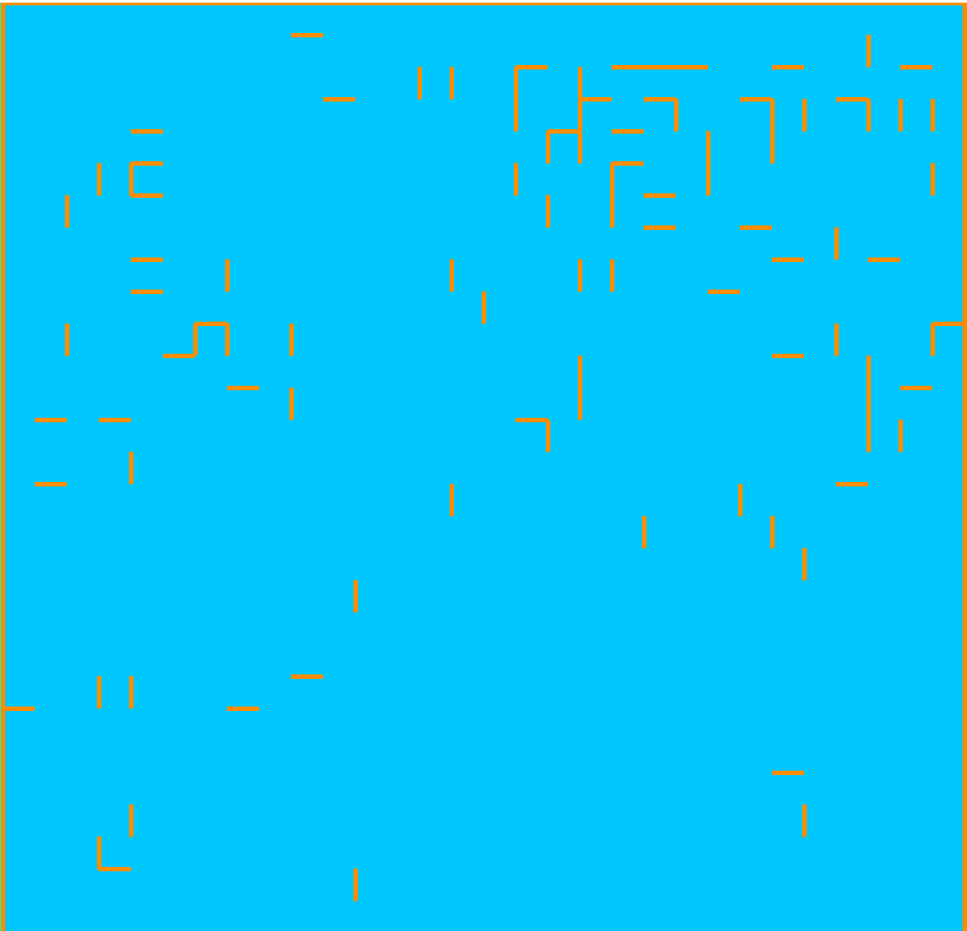
It will be similar in some ways to games like Invisible inc. or Scotland yard where players must escape the enemies / collect a certain item and head to extraction to succeed.

When I have finished with this game, I will test it with fellow students (aged 17+), friends and family as well as conducting my own stress testing to see if I can break the game using boundary and extreme data. The game will only be runnable on windows however there may be portability to Linux depending on whether I have time by the end of this project.

I will be researching how to procedurally generate a maze and how to create line of sight and fog of war algorithms for games. This project will likely use Djikstra or A-Star efficient path finding algorithms (which find the shortest path from one location to another) since it is a point and click, turn and tile-based game.

I also need some market research to find out what my target audience is, what their preferences are and how to accommodate for them. Furthermore, I may need to find out whether I can cater to the feedback within my constraints.

### How the game will play:

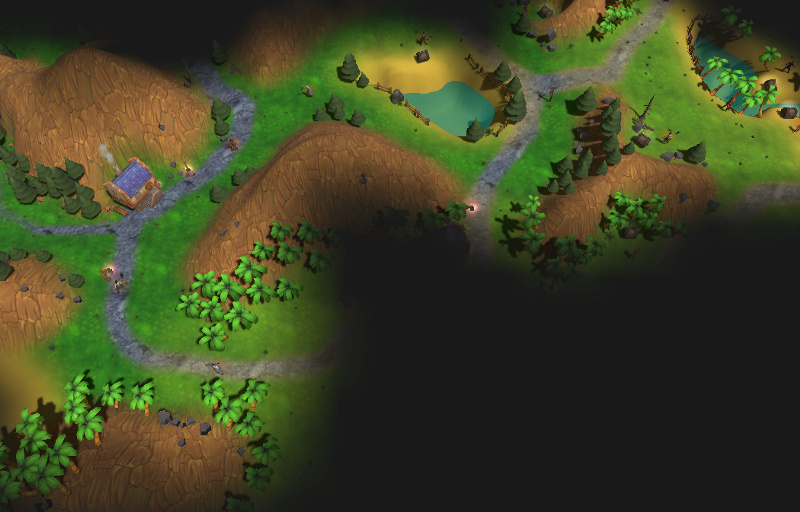
The game, as mentioned before, will use a top down view of a grid-based map as shown below.

Screenshot from project prototype

Each map is procedurally generated (randomly with restrictions/ rules). Every position (node) on the grid must be reachable. This is to prevent players spawning inside a blocked of portion with the exit being outside.

A player is placed randomly on the map at a position. Enemies are also placed randomly with restrictions on the proximity to players. This is done to prevent games becoming impossible. Items such as upgrades and perks will be placed randomly and one or more randomly placed exit nodes (positions) will exist.

The game will be turn and tile based. Players will be restricted on the number of places they can travel (steps). The player will choose a location to travel to by clicking on the node (position) and by likely hitting a go button to confirm their decision. At the end of the players turn the sprite that resembles the player will move to that location taking the most efficient path. The player and enemies will have line of sight algorithms in place that calculate what the sprite can see directly in front of it (line of sight). This will be used to calculate what the player has seen and can see to remove the fog of war effect over areas that are or were visible. A fog of war effect is an effect that covers a map from users preventing them from seeing ahead.

 Figure Fog of war effect source: <https://bit.ly/2MpXD81>

The enemy AI will then make their turns using the same rules as the player. Enemies can be in two states: searching and ambushing.

Whilst a player hasn’t passed the line of sight of an enemy the AI will follow a routine path around the map. This is to make them predictable so that players can calculate their next move.

If a player passes by the line of sight of an enemy, that enemy will be in the ambush state. The enemy will calculate the shortest path to player and take that path each turn (perhaps with some throw off variable to prevent the game being impossible to beat). The enemy may also fire projectiles at a player whilst they aren’t in cover. The player can either defeat the enemy by attacking or they can hide out of the line of sight of enemies for a while until the enemy is reset to the search state. The enemy will however travel towards the last seen location of the player until the player isn’t found within a radius of that position (again using line of sight that cannot pass through walls).

The aim is to get to the exit of each level and collect as much loot (points) as possible. Some levels may require the player to collect a key or some other item to unlock the exit.

This is subject to change however whilst I am programming (only minor changes to rules or improvements to the game).

## The client/supervisor:

The client will be (insert client here). I will discuss my progress on the project in intervals regularly.

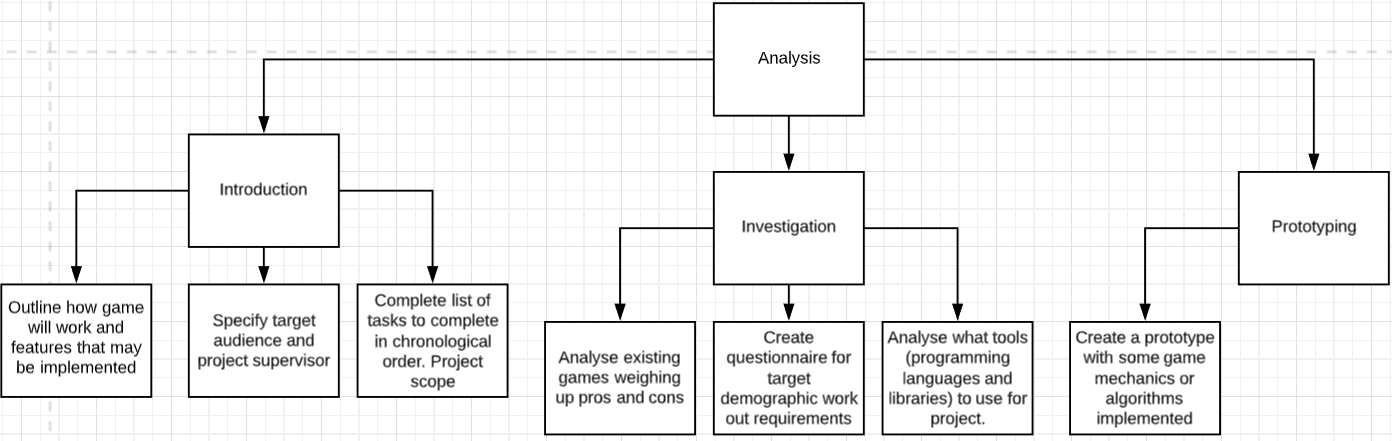
## Project Scope:

I plan to finish this project at least two weeks before the deadline to a high standard which I consider to be complete. This gives me time to notice any errors I may have made. The following is a table with an outline of what I will do in order and the relative time it will take. This may be slightly different realistically, but it gives me a good guide on what to finish when.

|  |  |  |
| --- | --- | --- |
| # | Task | Duration |
| 1 | Introduction | 1 |
| 2 | Investigation | 3 |
| 3 | Constraints, requirements, limitations | 2 |
| 4 | Objectives | 1 |
| 5 | Proposed solution | 1 |
| 6 | Finish Analysis |  |
| 7 | Documented Design |  |

### Prospective users / target audience

The game will be targeted towards younger children and teenagers, the game will be somewhat strategic so it may not be appropriate for much younger players. However, it will have difficulty levels that may be able to compensate for this. The game must be easy to pick up and play therefore I will have some form of tutorial or guide to playing the game. The game will be turn based and point and click so the actual game mechanics shouldn’t be too hard to learn.

Analysis plan

# 1.2 - Investigation

### Existing games

### Invisible inc:

Invisible inc is a turn-based strategy game that gets the player to traverse a grid-based map with multiple enemies and items/ tools which they need to interact with in order to finish the game.



### User interface

At the very start, the game features a simple graphical user interface which gives the user four options: play game, options, credits and exit game. The UI on this start screen is basic and suitable for the game since it doesn’t confuse the player on how to start a game. From this I have determined that my project should have a similar user interface at the start to maintain a player’s interest in the game since anything that is too complex may deter the player. Another feature of the UI is that each button lights up when hovered over to give a pleasing visual aesthetic. This would be simple to code and would improve the project in a subtle way that a player may not notice but will have an impact on the players opinion on the game.

The game also has save files that separate players (or one player with multiple runs of the game) can use to save the current game when finished and load the game when returning.

Character selection:

The game has multiple characters. Each character has different abilities and qualities. The 4 qualities are hacking, speed, strength and anarchy. The game also gives the player perks (as seen on the right side).

Each quality has 4 different levels that show how good a character is in each field. These qualities will influence the way the game plays and may make certain levels harder or easier which will give the player a reason to change characters to suite each level.

Verdict: From this I am considering adding abilities like Invisible inc to my own project that will influence how the game play for example by making it cost turns to perform certain actions based on each ability level. This will add variety to each playthrough and add a strategy element to the game.

The game starts out by giving the player an option of two characters to choose from and then allows for more powerful characters as the game progresses.



You play as two characters each level which allows for more complex strategies to be employed during each level. Despite this there is no multiplayer aspect to the game which seems like a missed opportunity.

Verdict: Consider adding a multiplayer element to the project where two players can co-ordinate to win.

### Rules of the game:

* Each character gets a certain number of tiles (steps) they can move to per turn
* Each character can only step on tiles which are not occupied, not too far away and can be visited by a path of adjacent tiles.
* If a room has not been visited it will not be shown on the screen (fog of war effect)
* During each turn a character moves, and all existing enemies move following their routine path after both characters have expended their turns.
* Opening doors and interacting with items uses steps (the number is based on the weighting of the interaction and ability level)
* Characters have two modes ambush or hide if a character is set to ambush, they will wait until an enemy visits a neighbouring tile and then neutralise them.
* There is a security level which dictates how easily alerted enemies are and whether new enemies will be introduced.
* There is an objective to complete and an extraction point which users must head towards.

### Level design:

Each level is designed by a grid. Each player can move several tiles and may interact with certain objects inside such as computers and cameras and terminals. Each action costs a certain number of movement points. There are multiple rooms in each level and an extraction point which characters must head to after completing their mission.



Any unvisited/ unseen areas will be masked by a fog of war and during enemy turns if the player can’t see the enemy through a camera or otherwise the enemy’s movement will be illustrated by a noise wave effect for each step they take.

### HUD design:

A HUD (heads up display) is arguably the most important part of most games since it is what is interacted with the most and gives the game it’s “feel” or atmosphere as well as highlighting game mechanics. The HUD in Invisible inc is very minimal with little writing and features technological visual style. It also allows user to see which character is selected what their statistics are and their inventory status. It also shows objectives in small writing at the bottom right corner. Furthermore, it allows users to end their turn start their turn reset their move or switch between different tactical views which are angles at viewing the map (since it is somewhat 3D). It also displays the alertness level constantly, so players always know how difficult the level currently is. It shows what turn the player is on and allows the user to access an external menu that allows the user to quit or deal with options.



|  |  |
| --- | --- |
| Positives: | Negatives |
| The user interface is clear, simple and concise | There is no scoreboard / leader board to give a sense of competition in the game |
| The player gets objectives and missions to complete and rewards for those missions in the form of unlockable characters and perks | There is no multiplayer despite there being multiple characters in a level at once |
| The game has very intricate yet simple game mechanics which give the game a good polished feel and smooth gameplay | The first level introduces complex game mechanics right off the start that may overwhelm a player (although they are not required, they may become confusing) |
| Music adds tension to the game and will become intense depending on the situation | Objective is not clearly displayed on the screen so players that didn’t focus before the level may lose track of what they are doing |
| The game has a save feature which allows users to leave the game and return later | HUD can be said to be a little too minimal and perhaps too small which can create confusion and accessibility problems. |

### Conclusion:

Through my analysis of this system I have concluded that I should consider the following:

* A simple User Interface
* Multiple characters with different perks
* A turn-based movement system
* Weighting to tiles on the grid and to interactions
* Exit points to leave the level and objectives to complete
* An alertness level which indicates how difficult enemies will be
* A fixed number of moves each character can take
* Clear indication as to what the objective of each level is
* A multiplayer element to the game
* A simple HUD which shows the current game status and allows users to visit options and see inventory
* HUD accessibility settings to allow for font size increases or colour-blind modes

### The X-Com series:

X-Com is another turn based strategy game with less emphasis on stealth and more emphasis on tactics and strategy. The game has highly detailed graphics and many different weapons and environments to play with / in. It allows players to choose a squad (group of characters with different traits) and send them to a level. The players goal is to complete some sort of mission objective and safely leave. Maps are premade, however there is many maps with regular DLC which makes the game fresh and varied instead of being repetitive and monotonous. However, this has come at cost to the company designing the game and may not be an appropriate decision for my scenario since I will only release one version of the program (once completed). X-Com is available on multiple operating systems including most console, phone and desktop operating systems. This allows them to reach a much larger market.

### User Interface:

(Enemy Within) X-Com’s user interface is simplistic, featuring options to load games, play single player or multiplayer and options to tweak game settings (such as audio and graphics). The start screen also has an exit button (on PC builds of the game) which will make the game more accessible for users who don’t know how to close out of the game or for users with slow hardware (since forcing a game closed usually slows a pc down). The game has different difficulty levels that will set the quality of the characters you may be able to add to your squad and will also affect the strength and difficulty of the games main opposing AI.

### Game Mechanics:

The game itself is turn based with each turn giving the player the option to reposition their squad members and attack or take cover from enemy forces. Cover works in the form of obstacles such as cars walls or boxes and each one of these types of cover reduces the chance of getting hit by enemy fire the next turn by a different amount. That brings us onto another game mechanic which is chance. The game is heavily dependent on chance. Enemy AI is somewhat unpredictable and landing or missing shots is entirely random and based on the level of cover the target has. Each character can move a limited amount of spaces depending on their character traits/ skills and the chances of landing shots or dodging shots are also altered by this.

When a player clicks on a character (or selects them by some other means) the player will see the range of locations that character can visit (in blue and orange) and the amount of steps and shortest path to the location that is being hovered over in a blue line. It will also show the cover stats of that location as well as any other relevant information. Both the game AI and players have some form of path finding algorithm in place to calculate the shortest path to a point.

Although the game is designed in 3D the game is, in a basic form, two dimensional which means any 3D rendering is done only for the sake of making the game look better. This will likely not be a priority for my project.

The game mainly features ranged weapons and, though they do exist, melee weapons are rarely used outside of stealth missions. When attacking there is random number generation (RNG) involved reducing the games emphasis on strategy and making the game a little frustrating since each run is inconsistent.

### Level design:

Each level is unique and created by the developers and each level (or mission) has its own objectives that may change throughout the level. The map is covered by a fog of war preventing players from seeing where exactly the enemies are or will be coming from. Maps are made of a grid layout and there are usually multiple layers to the grid layout (3 dimensional). Furthermore, enemies are placed in strategic locations to make the game somewhat challenging and the quantity or difficulty of enemies is determined by the difficulty level of the game.



### HUD:

Objectives are listed at the top left corner of the screen allowing users to constantly be able to see what the goal of the level is and what completed or new goals have appeared. At the bottom right corner, the current characters health stats and weapon are displayed making it a lot easier for players to make better strategic decisions throughout the game. Options to change character or change angle of 3D view are available in the top right corner and all enemies that are visible to the squad or squad members have their stats highlighted above them. Also, at the bottom of the screen there are options to use character perks of the selected character. Each of these options / functions in the game are on the peripherals of the screen and are minimalistic and non-intrusive to the gameplay experience. This is an important design aspect because it prevents the HUD from being frustrating rather than useful. Also, interaction with the HUD using the mouse may reduce the pace of the game so the game has hotkeys in place (like 1,2,3,4) to allow users to efficiently play the game.

### Enemy AI:

Enemy AI moves strategically from cover to cover (however this will depend on difficulty and type of enemy) and will ambush attack or defend from enemy players. Enemies may also be unalerted at the start of levels until a squad member passes their line of sight which will cause enemies to be alerted of a player’s presence and may have them ambush or protect from players. The enemy AI will actively try to hinder a player’s progress by either attacking players or moving / protecting objectives (for example hostages).

|  |  |
| --- | --- |
| Positives | Negatives |
| Is strategy based and therefore is engaging due to the thought involved in playing the game | RNG is a heavy aspect of the game which means that playthroughs may be inconsistent. |
| Game AI is unpredictable yet predictable allowing users to improve at each level consistently yet keeping each playthrough fresh with new challenges or different play styles | Also, there are issues with players closing and reopening the game if they fail a mission allowing them to avoid the consequences of failure (in solo play) |
| Has simple game mechanics that are easy to pick up and easy to follow (such as hovering over grid positions will highlight the path taken so players can take evasive measures) | Game requires relatively strong hardware which may be too costly for some users (except in the case of android and IOS). |
| Has many levels and regular DLC which keeps the game fresh for very long times | No option to speed up game and pass through slow enemy turns |
| Has tutorial modes and accessibility settings and is available to most of the common operating systems allowing for appeal to a very broad audience. | Constant updates are costly to game developers when random map generation would have been a much more efficient option (like in Minecraft) |
| Has engaging graphics which makes users enjoy the game more than if it was plain |  |
| Has simple user interface; non-intrusive and easy to use via hotkeys |  |

### The Binding of Isaac:

The binding of Isaac is a 2D procedurally generated dungeon game (similar in many ways to my own project). It has smooth gameplay clean graphics and is polished to a high standard. It is rated highly among critics. It doesn’t require strong hardware to play and was relatively inexpensive to make despite its quality and popularity. The aim of the game is to defeat multiple stages and earn powerups that will help with the final boss battle. After this, the game goes into endless mode which allows players to achieve ever increasing high scores. One issue with the game is the high skill ceiling and variability which takes a lot of getting used to. Another problem is the rare occurrence of impossible to beat levels. Though this seems to have been patched well, this may be an issue I will run into.

### Game Mechanics:

At the start of each run you start with no weapons or perks other than the ability to fire tears. You must navigate through different rooms killing enemies along the way. There is a map that originally only contains the room you are in. As you progress previously visited areas will be shown on the map to prevent the player from running in circles. Each stage has a limited amount of rooms and the aim is to reach a golden door whilst carrying the key. The problem is that you don’t know in what direction the door is. This is repeated on multiple stages (each stage is accessible through the golden doors) until the boss stage is reached. Along the way you may encounter and pickup special perks or weapons that you can use to progress more easily. Enemies progressively get harder as you progress however enemy AI is simple and repetitive much like in early Zelda therefore it isn’t hard to learn patterns and use them to your advantage.



### HUD:

The HUD consists of three simple aspects; a mini-map, heart/health count, coins bombs and key count. This makes it a lot easier to find out your current stats and location in game which is crucial for a game as fast paced as this.

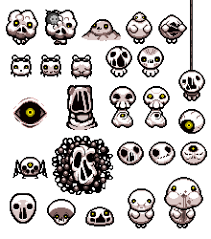
The game also has a very simple user interface with basic options such as volume gore settings and exit. Like with the previous games the game features a save option.

### Level design and procedural generation:

Levels are, like before, grid based with blocks as obstacles, pits or interactable objects. The game is procedurally generated so each room is random with random enemies within a threshold of AI difficulties and size. Furthermore, the game must be beatable, so the procedural generation accommodates for this.

There are different obstacle types each with different levels of breakability and each with a possibility of containing some item. Some objects are only affectable using certain perks or weapons. AI may also be able to affect the terrain of each level by creating pits, obstacles or by removing obstacles.

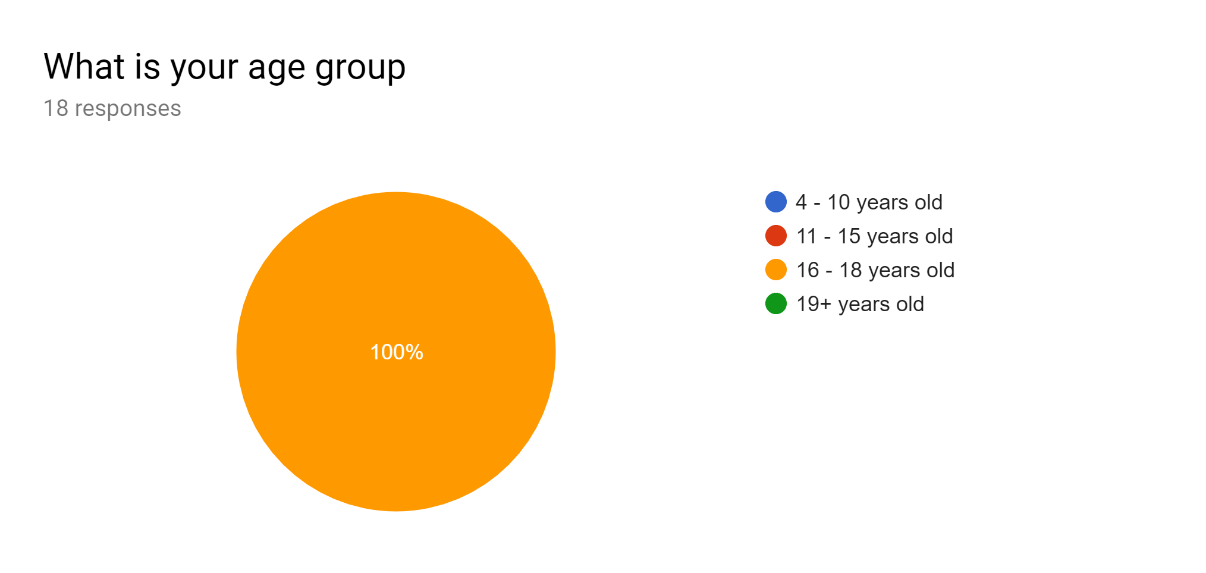
### Enemy AI:

There are multiple enemies each with different movement and attack patterns. Some enemies such as eyes fire lasers in straight lines periodically and just move left and right or up and down. Others bounce around randomly like spiders and haarder enemies usually explicitly target the player. The AI for this game isn’t particularly complex however this is made up for by the uniqueness and the relationships between different enemy types. Boss enemies are hard and can spawn other enemy types.

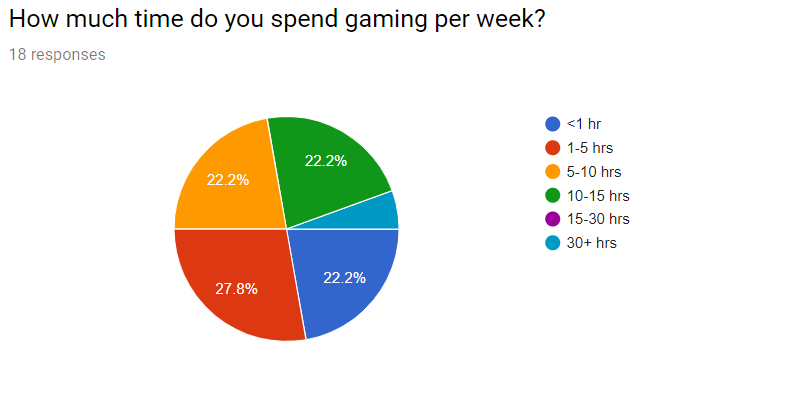
|  |  |
| --- | --- |
| Positives | Negatives |
| The game is procedurally generated which keeps the game fresh at relatively low cost and allows for unlimited replay-ability | There are issues with the game being unbeatable at some points which is due to bugs with the procedural generation |
| The game is simple to pick up and play and most of the controls are intuitive | Even at lower difficulties the game can become hard which creates high skill ceilings |
| The HUD is simple and has all information placed compact in the edges of the screen making it easy to find out current stats. | After death the game ends and players must restart which can make the game frustrating. This is easily fixed with a game mode. |
| Game has multiple difficulties and the game progressively gets harder as time goes on keeping the game challenging | There is no guidance on what various items do which makes the game difficult to learn creating a steep learning curve |
| Game isn’t taxing on hardware therefore it can be played on most devices including PC and console. | No accessibility settings for players with joint problems which means there is a lot of button mashing |
| The game allows players to copy and find out their seeds which keeps the game somewhat consistent if players desire | Ports to other devices don’t function well or as well as the original version. |

## Questionnaire

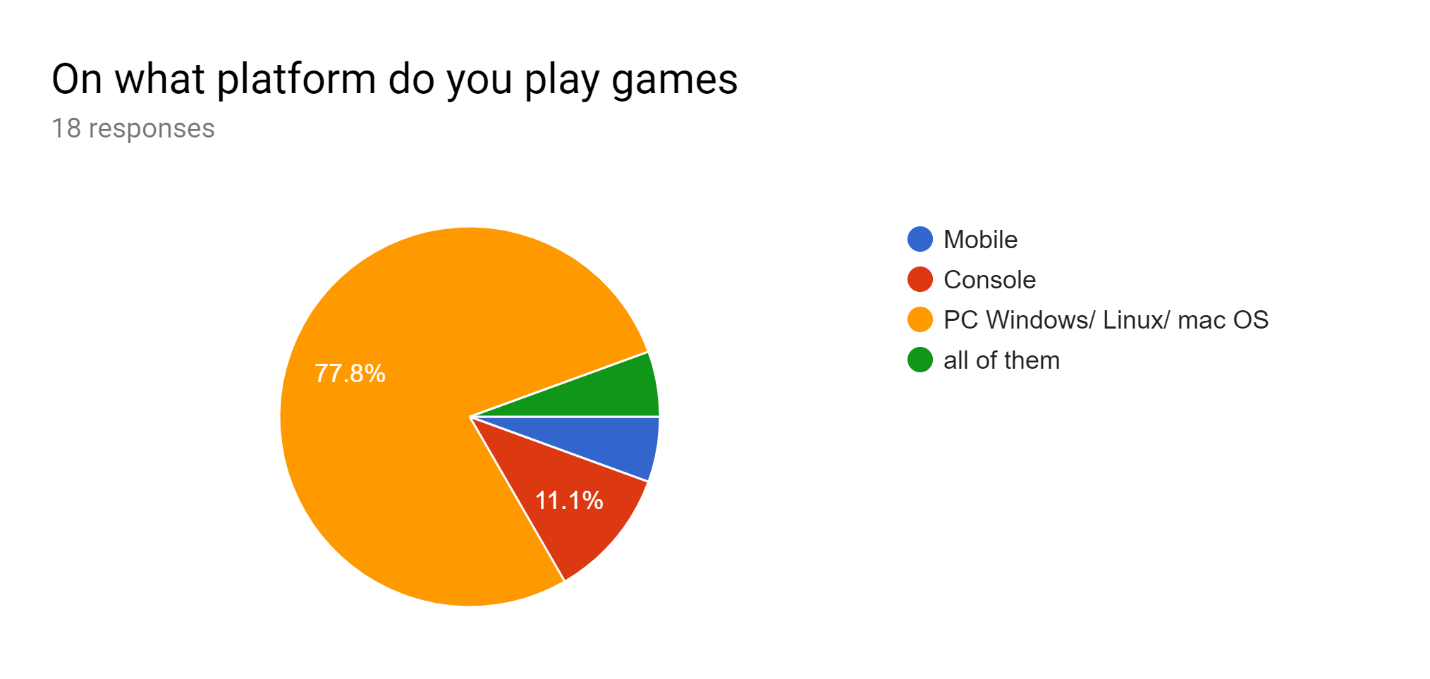
I wrote a questionnaire that was filled by several classmates and family members. I believe they are a good sample of my target market since this is a game for teens and younger. The questionnaire was designed to create a good idea on who my target audience may be and what their desires may be for a video game.



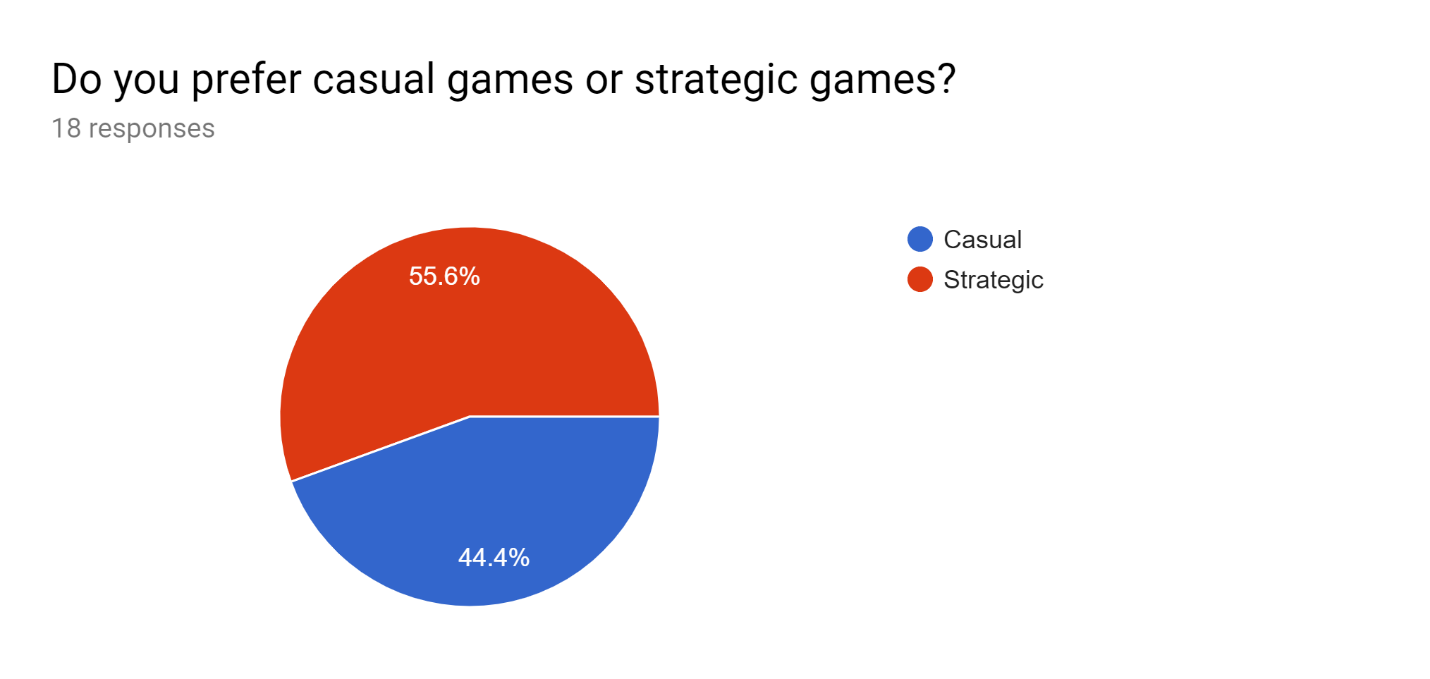
This question in particular wasn’t placed well since all of the responses have come from 16–18 year olds. The only deduction that can be made from this is that the responses in the questionnaire will be representative of this age demographic. Due to my target demographics age an assumption of affinity with technology would be reasonable.



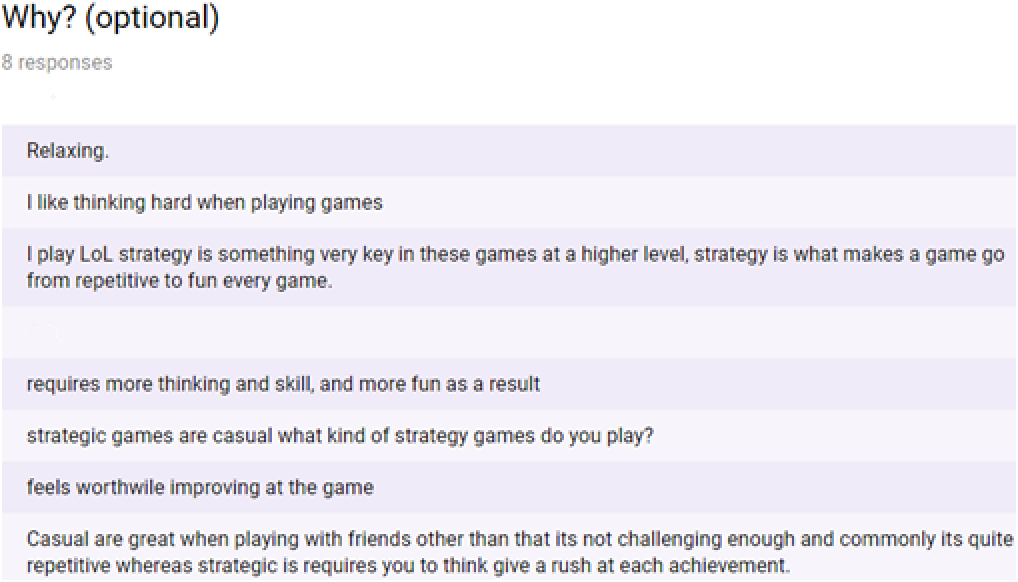
Exactly half of my sample clearly have a strong attraction to video games spending more than 5 hours a week gaming. This is also exaggerated by the fact that most of the subjects are also studying and preparing for university at the time this survey was published. There is also an equal proportion of responses that spend much less time per week gaming (<5 hours) this means my game will have to cater to many different experience levels. I will include a tutorial that beginners can use to learn the game mechanics. I will also make the tutorial optional so as not to annoy more experienced players. I will allow for control remapping since this is desirable among competitive gamers.



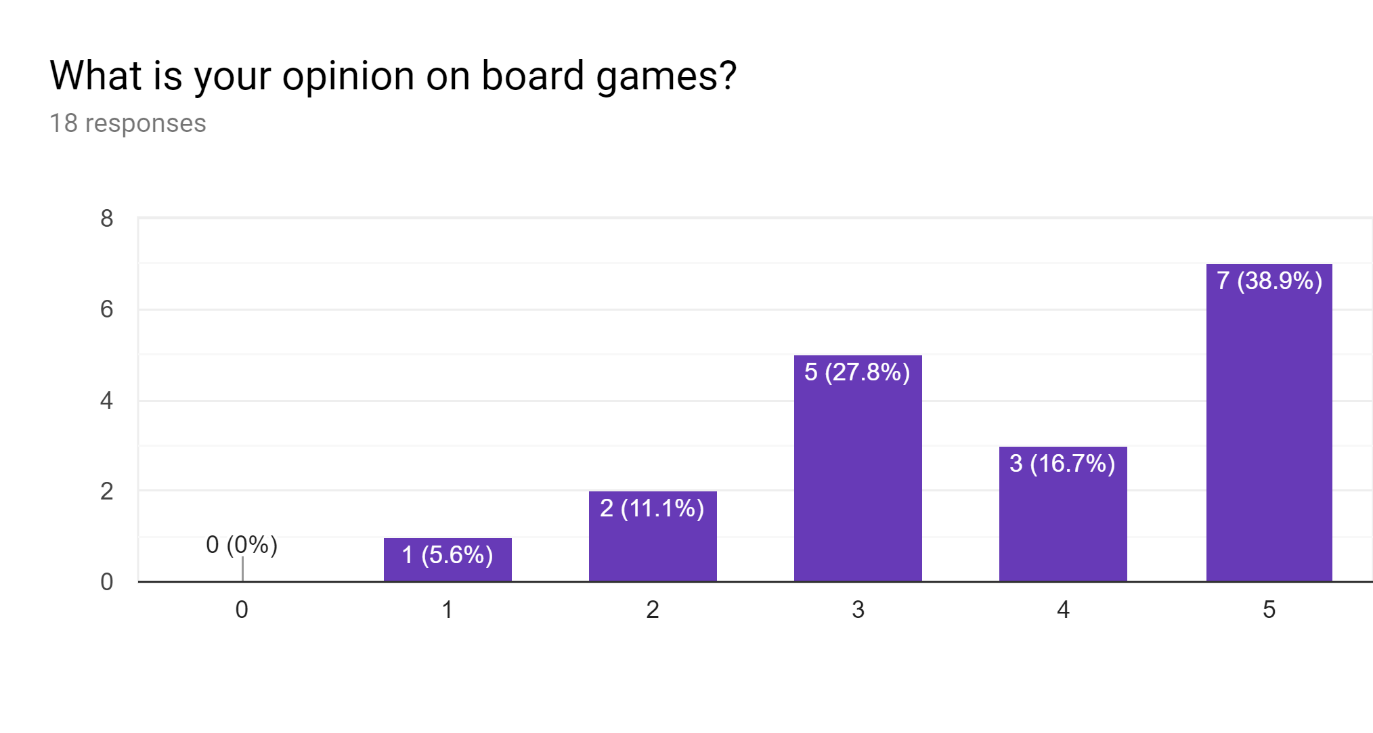
The overwhelming majority of subjects play games on the PC platform so my game will be designed for PC gaming (keyboard/ mouse input etc)



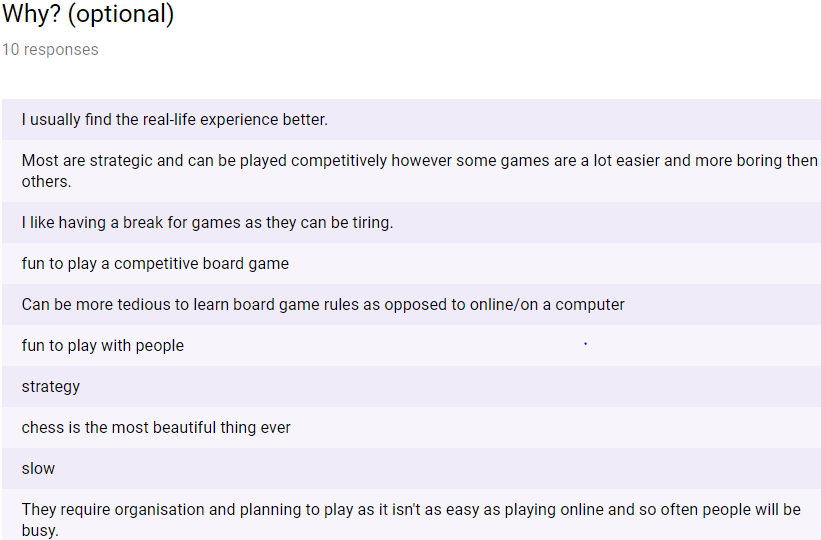
There is a near 50/50 split in responses to this question with the majority leaning in favour of more strategic games. 44.4% of responses say they prefer casual games. Some responses outline why below:



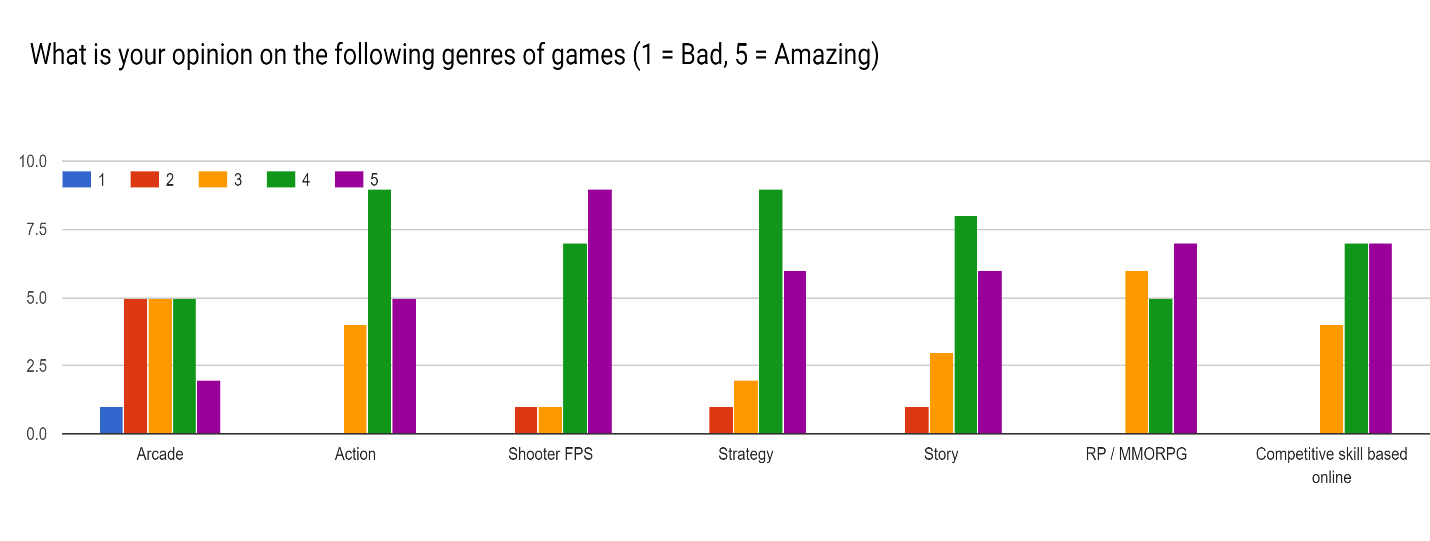
Due to the nature of the game (strategic) it will be hard to accommodate for casual players however I can still have different levels of difficulty in the game to make it easier for beginners and casual players and make the game harder for more advanced players. Another suggestion is the inclusion of achievements. Though this isn’t necessary it will add a challenge aspect to the game therefore I will try to incorporate this.



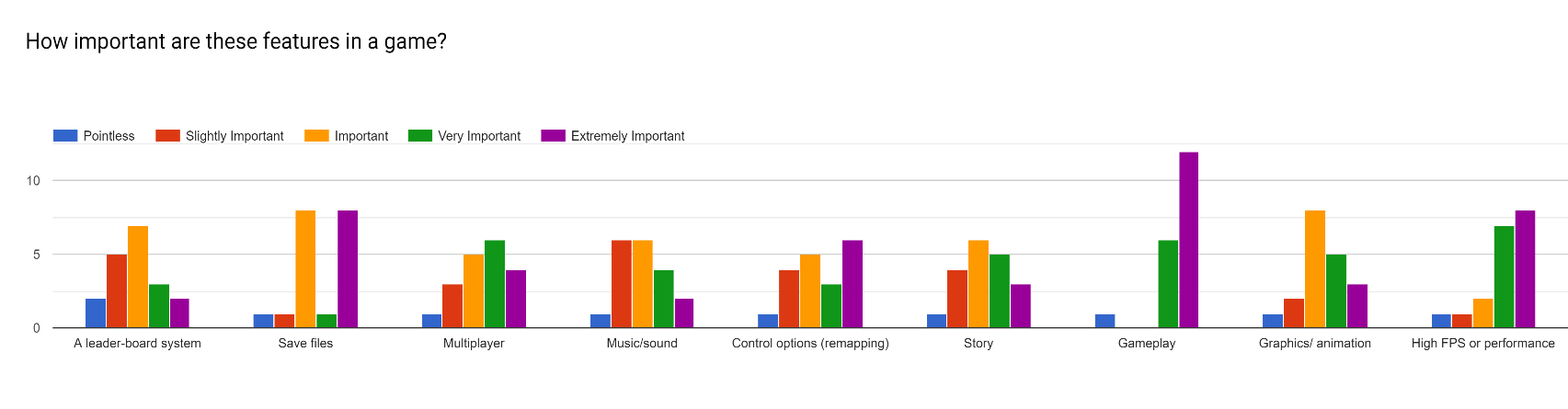
Since the nature of my project is similar to a board game and can be designed in a board game format, I thought it would be appropriate to ask about the user’s opinion on board games (on a scale from 0 to 5 or horrible - amazing). Most responses indicate a neutral to positive opinion on board games which means I will make the game a 2D top down game with characters clearly indicated. 83.3% of responses had a neutral to positive opinion so this is appropriate.



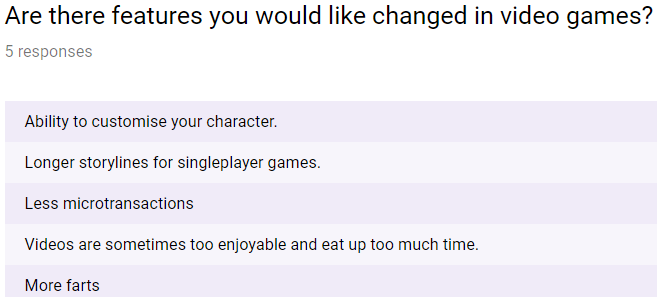
Within the verbal responses the difference between casual and competitive gamers is clear. Some gamers don’t want to put in the effort of learning the rules of a board game since it “Can be tedious to learn rules as opposed to on a computer”. One response in particular states that board games require setup which is time consuming. Fortunately, I can easily accommodate for this since a computer game will automatically setup my game. Another says they “like having a break”. This indicates to me that I should include save files and a pause option to allow players to rest in between play sessions.

****

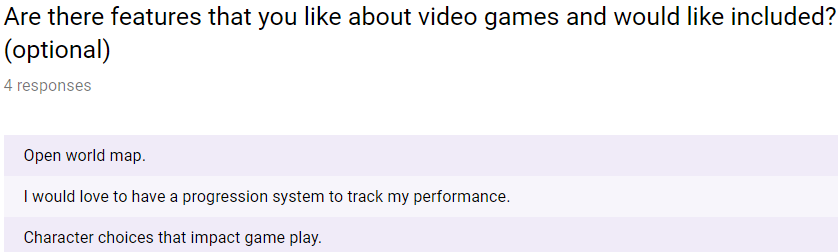
There is a consensus of positive opinion on these genres: action, strategy, story, RP/ MMORPG and competitive skill based online. There is a more median distribution of opinions on arcade games. This is a mainly desirable outcome since users have overwhelmingly positive opinion on competitive skill based and strategy games. This is the genre of game that I will be creating. Responses also express interest in massively multiplayer online and competitive online. Due to my time constraints there may not be enough time to create an online multiplayer game however there is a possibility of some ranking system be it online or arcade style. Furthermore, networking over LAN may be a possibility and should be taken into consideration.



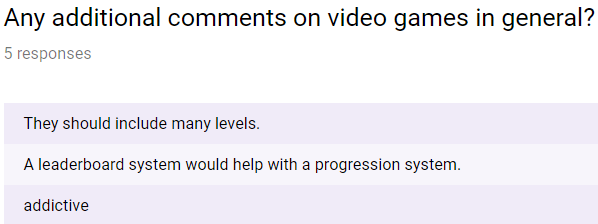
Responses to this question stress the importance of gameplay listing it as the most important aspect of a game. Save files were second most important which may be since players may want to be able to stop playing and do something else without compromising progress. This means having save files will be very important and gameplay should work and be efficient with no glitches. High FPS/ performance is also listed as a close third in important game aspects, though this may not be hard to do, having the game be runnable on most reasonable PC’s is very important and a must have if gameplay isn’t going to be compromised. Most other design aspects where considered just important as opposed to slightly or extremely important. This means that they should have some aspects such as controls remapping and a leader board system, however these are not considered to be as essential as the previously mentioned design aspects. Animation would be desirable, and I should add at least smooth sprite movement. The game could have a story however these isn’t really something that should have too much time devoted to it.



Subjects were asked if there are any other features or issues with video games they would like changed. This was asked to prevent me from adding features or omitting features that end users may deem to be important. Character customisation is an interesting design choice and the game could have this implemented. Other than this, responses were not relevant to my project except for perhaps not using videos for tutorials and have interactive tutorials.



One user who was asked this question responded with a good suggestion to have an open world map. Though this is not possible within the constraints of my own abilities and time, I can have the game be procedurally generated to compensate for the endless exploration and endless possibilities aspect of open world games which may be the reason the user requested this.



Another user asked for many levels and with procedural generation this can easily be achieved. A progression system was also requested, and this should be incorporated in the game in some way.

## Programming languages

### Python:

Python is a multi-paradigm programming language that can be imperative, procedural, object orientated and functional. However, strictly speaking python is mainly used for object oriented and imperative programming. The imperative programming paradigm is one where the program is written as a series of commands or calls to function that a program completes sequentially. Object oriented programming paradigm focus on what can be called “data structures” (but not really) known as objects. These objects have attributes which are essentially variables that are specific to that object and its class. The objects belong to classes that denote the nature of these objects and the different methods that can be used on them. They provide a framework for creation of objects. Methods are functions that are specific to an object and can be performed on an object of a class.

Python is very useful for rapid code development and has a very high amount of online support and resources that can be used to troubleshoot common problems and learn how to create programs. Furthermore, python has support for a library (or module) known as Pygame which provides much of the framework for game creation such as key press detection and surface (or canvas) creation. Pygame uses a programming paradigm known as event driven programming. In event driven programming programs work by having a main loop. During this loop the program will check for events. The events may come from the computer such as a certain time or random number or an event will more likely be user input such as mouse position change. Once the code detects an event the program will use selection statements to decide what to do in case off certain events. For example, if the escape key is pressed the program may cease running. This is particularly useful for games since the game needs to draw onto a canvas each frame and what it draws will depend on user input. I am already fluent in this programming language.

However, there is one major flaw with python, the program is very implicit. One example would be the fact that lists are passed by reference. This will cause problems when trying to troubleshoot code that doesn’t seem to work for any apparent reason. Furthermore, python has no built in option to restrict the type of variables (such as integer, float, string etc) and has no option to choose whether arguments are passed by reference or by value. There is also no option of making class attributes and methods private, public or static in python. This may lead to code that is unsustainable in the long run with many issues (when working in dynamic teams).

### JavaScript:

JavaScript is a programming language commonly used for web development. It mainly supports OOP and has support for procedural programming. This means that the program has built in support for surface/ canvas creation. It also has built in functionality for in-built buttons/ input boxes (from HTML). The programming language has built in integration with HTML and CSS which makes functionality such as login systems and multi-platform games much easier to create. Which brings us onto the next point, JavaScript is supported on virtually all browser environments and , due to the nature of the way the code is compiled (into intermediate code that is run on a virtual machine in the browser), it is also supported on all devices. This would make cross compatibility a given between devices which is a very big positive. Unlike python, JavaScript has private and public class attributes and does have type declaration for variables (which is useful for creating code in the long run as stated before). There is also a vast amount of online support for JavaScript as it is one of the more common programming languages. One interesting framework for JavaScript is Node.JS. This framework is highly efficient due to its usage of the chrome V8 engine (which compiles JS code directly into machine code) which has regular updates increasing efficiency (due to heavy Google investment). Node.JS is also open source so it would be free to use and has many different modes that can be bespoke to my problem. Also, code for the server back end and website front end is very easy to manage since they both use JavaScript.

However, JavaScript doesn’t have options for passing variables by reference or by value which may cause some issues since different copies of the same type of data (for example objects) will update with the original piece of data creating hard to find problems in code. Furthermore, code development in JavaScript is time consuming, especially for myself since I mainly have fluency in python. This unfortunately makes this language an unlikely option due to my time constraints (outlined in project scope (check contents).

## Prototyping

During the investigation phase of my project, I have created a prototype of my game that is entirely incomplete but has some features that will be in the game. I decided to use python to create the project since I already have extensive knowledge of the language and due to my time constraints, I cannot learn Node.JS in time (outlined in constraints and limitations).

I used pythons Pygame library since it provides functionality for creating canvases, surfaces and sprites etc. To program using the Pygame module the main approach (and the approach I will be taking) is to use event driven programming (this is explained in the programming language section under python).

I have created some flowcharts that describe how to solve some of the different problems before reaching the final solution. One such problem is maze generation.

I need to create a maze generation algorithm. The rules are as follows;

* The maze must have at least one path from each node to every other node (no nodes should be blocked off)
* Nodes can have walls in between each other facing North, West, East or South (NWSE compass)
* The maze must have loops (unlike a perfect maze which is why recursive backtracking isn’t used)

### Maze Generation Algorithm (verbal):

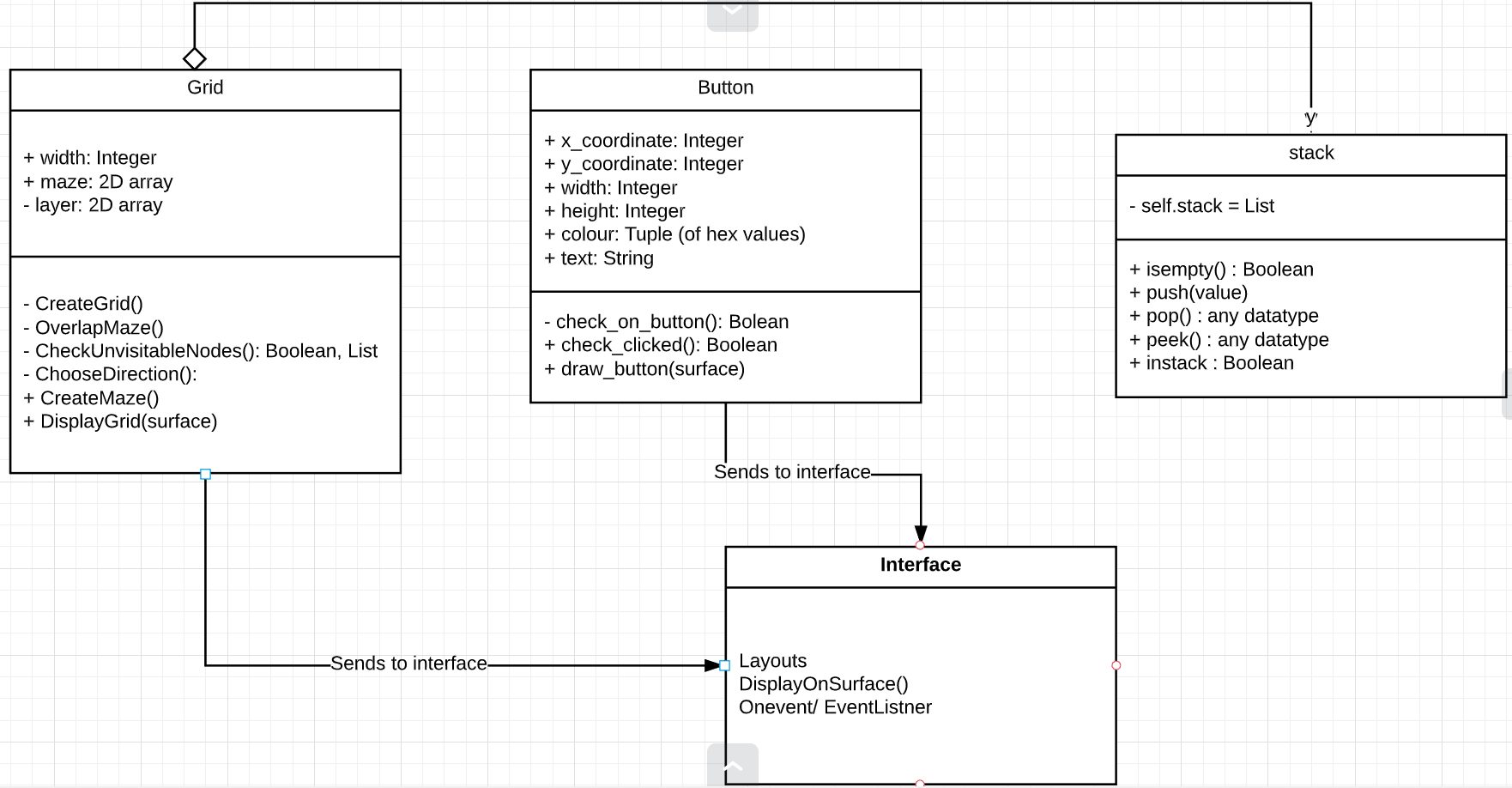
* Create a function that creates a grid of specified size (2D array)
* Create a function that will procedurally generate paths through that grid
* Create the stack data structure

How it will work:

* Grid (2d array) will be initialised using a for loop. The maze will contain an array of (1, 1, 1, 1) at every index representing NWSE walls (1 indicates a wall and a 0 indicates an open path). Let grid be called maze of width n
* Generate paths by creating a layer the same way the maze was created (stored in separate variable)
* Loop while there are unreachable positions in the maze i.e. (1, 1, 1, 1)
* Choose a random start position = current position and a fixed end position within the maze
* Loop until current position is the same as exit position.
* Calculate directions that you can take from current position (omit paths out of the maze and visited locations (using history stack))
* Randomly choose from these directions and change current position to the position corresponding to that direction
* “Break” the walls between the past node and the new node by changing numbers within the array from 0 to 1
* Add past node position (index format) to a history stack (called history from now on)
* If there are no directions to travel in follow the dead-end routine as outlined below
* While there are no possible directions to take do the following
* Calculate directions that you can take from current position (omit paths out of the maze and visited locations (using history stack))
* If there is no possible direction, then pop the last position of the stack History.
* Replace walls of last position (based on the direction you have travelled to get to last position) for example if you returned south replace (1,1,0,1) with (1,1,1,1) for current position and (0,1,1,1) with (1,1,1,1) for last position
* Change current position to last position
* Repeat the process until a valid direction to travel is found
* Once the current position is the same as the exit position overlap the current layer with the current state of the maze in a similar process as a bitwise AND operation (creating a mask)
* Repeat process above until there are no unvisitable nodes
* You now have a “maze” (more of a dungeon)

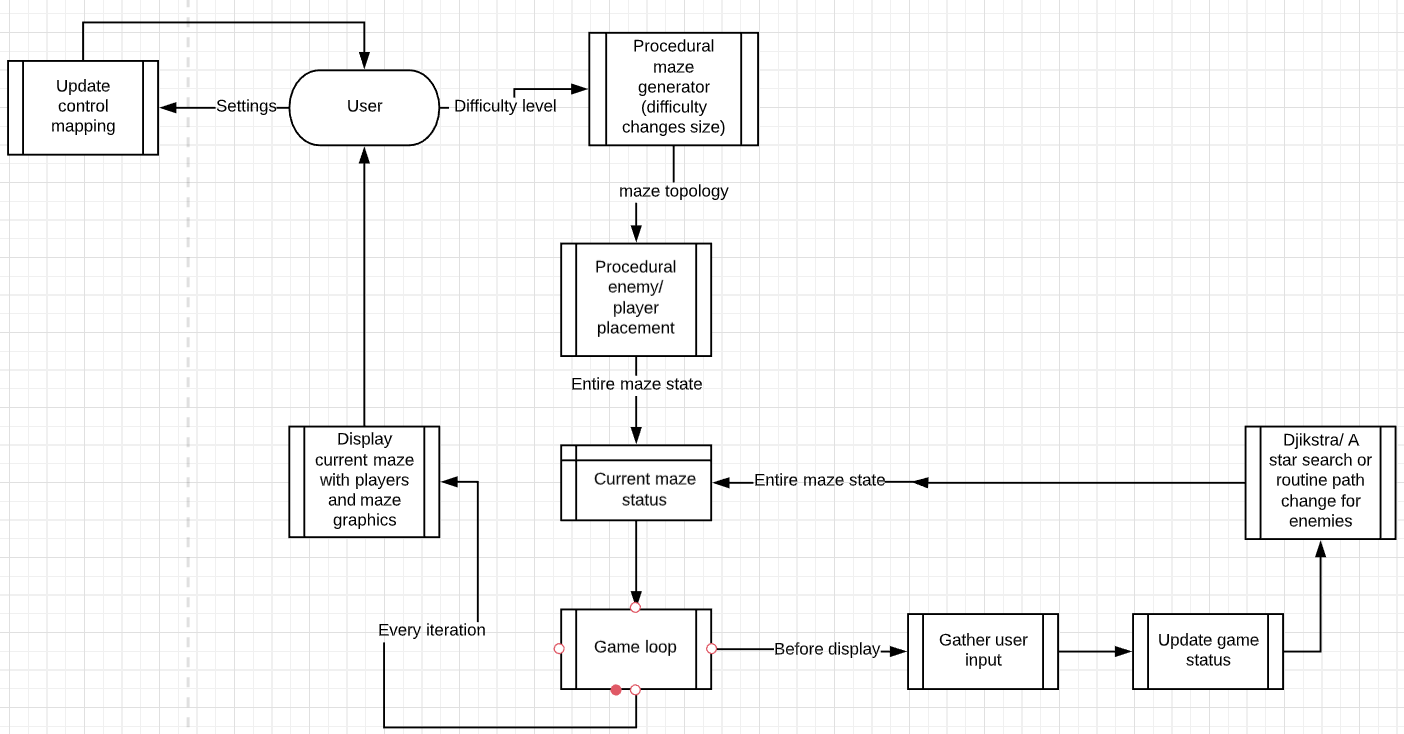
For this to work a stack is required therefore I have a class diagram

### Class diagram:

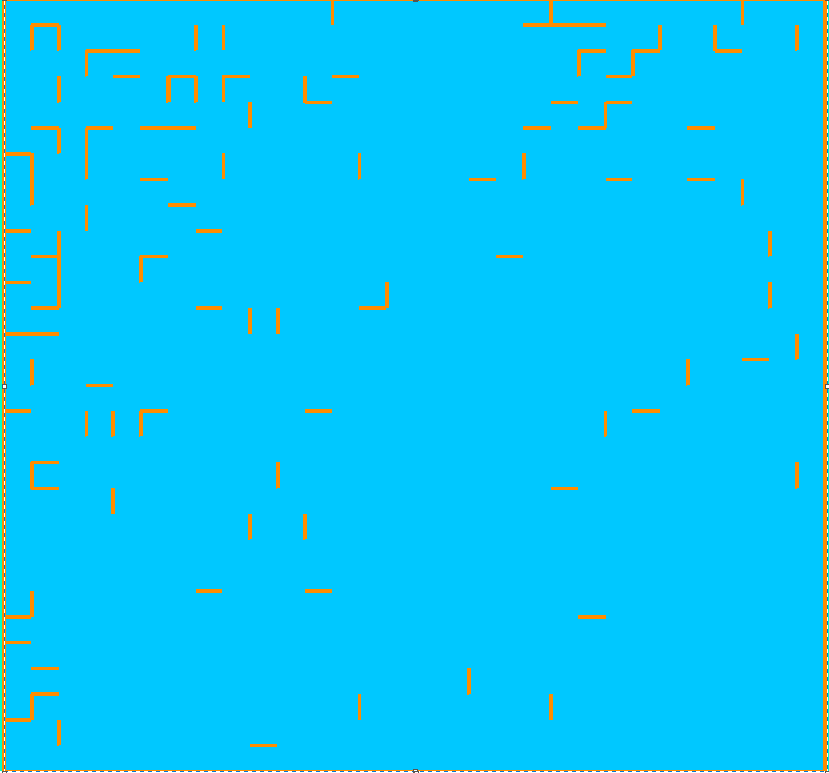


This is quite simple and works much like any other stack although you can check if a value is in the stack and you can see the data item at the top of the stack.

### Data Flow Diagrams



### Prototype showcase

Below is a screenshot of my procedural maze generation software (grid width of 30) for my prototype. A video that showcases the prototype at different maze sizes (as well as button showcase) is found via the link. The window is also resizable however this is hard to capture on video. 

<https://youtu.be/nBSNnPqGkBs>

## Objectives / Constraints

My main constraint will be time. There may not be enough time to add all the features and functionality I may want to add. However, there should be enough time to complete the core functionality of the project. There are however limitations with Python such as the lack of cross platform availability or the implicitness of variable types and protected variables (outlined in programming language analysis).

Through communication with potential end users, and through investigation of similar projects I have created the following objectives:

### Main objectives

* A user interface (menu system and inventory / options / game control system)
* Turn based movement that works as intended
* Clear indication of game objectives / how to win and a win scenario in every level
* Working HUD
* Enemy AI that works and doesn’t make the game impossible to beat
* Multiple difficulty levels or some form of challenge the longer a game goes on
* Procedural map generation and enemy / player placement algorithms
* Low hardware requirements (Intel Pentium g4560) high performance
* No game breaking bugs / software issues
* Playable on at the very least a Windows 10 PC

### Optional Objectives

* Clear user interface
* Remappable controls
* Simple non-intrusive HUD (takes up small proportion of screen / concise)
* Tile weightings to increase strategic play
* Enemy alertness level to increase difficulty level as you play
* Smooth animation/ pleasing graphics
* Resizable window with scalability
* Cheating prevention
* Fog of war
* Save files / pause functionality
* Accessibility settings such as font resizing and colour changing
* Music and volume settings to go with that
* Leader board system / progression system