SCC 210 Group Project Design Report

Group C6

Table of Contents

1. Introduction
    - Introduction and influences 2
    - Initial Ideas 2
2. Game Design
    - Chosen Idea 4
    - Features and Gameplay 4
    - Game Design Principles 5
    - Game Rules 8
3. Software Engineering
    - Functional and Non-Functional Requirements 9
    - Use Cases 10
    - Acceptance Tests 13
4. Implementation Plans
    - Task List 15
    - Risk Assessment 16
    - Milestones 17
    - Activity Network 18
    - Gantt Chart 19
5. Appendix
    - Sources 20
    - Game Sketches 20
    - Kamek's Tower Sketches 21

Introduction

Introduction and Influences

During the initial stages of development, we began to brainstorm several unique ideas each taking inspiration from vastly different genres and games (see appendix – section 2 for some original sketches). A key theme however between all the ideas was the style of graphics within the games. Most, if not all, our ideas revolved around the idea of using 16-bit style graphics and using a tile set to render the environments within our game. This led us down the route of 2-D top down game like "Pokémon Gold" (figure 1.1) or a side scroller similar to

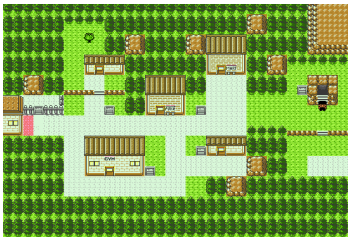


Figure 1.1 – Pokémon Gold (1999)

Figure 1.2 – Super Mario Bros. (1985)

"Super Mario Bros." (figure 1.2). Based off these classic titles we came up with a number of varying quality ideas for our group project.

Initial Ideas

The first idea we came up with was a simple top down RPG (Role playing game) similar to games such as the original "Legend of Zelda" (figure 1.3). In this game the player controls the hero and navigates around a large world defeating various small monsters on the way and larger dungeon bosses within special temples each with unique mechanics. The game would feature 16-bit style graphics and the player would have a choice of different playable characters with unique abilities (Sword and shield or magic etc).

Figure 1.3 – The Legend of Zelda (1986)

Another idea that followed was a classic 2-D beat-em-up game akin to "Streets of Rage" (figure 1.4) and early fighting games. The game is based on a series of side-scrolling levels filled with enemies with the player moving left to right defeating them as they go to reach the end of each level. Each level has an increasing number of enemies with new moves for the player to beat in order to reach the goal.

Figure 1.4 – Streets of Rage (1991)

An alternative idea we had was an arcade style puzzle platformer. In this game, the player has simple controls like move and jump but can also manipulate the levels themselves to solve platforming puzzles using various transformations on the stage, like rotate and stretch. The game is a series of progressively harder and harder puzzles each requiring more complex manipulation of the level itself to beat.

One other proposed idea was that of a 'roguelike' style game much like "Rogue Legacy" (figure 1.5). This game is based on the idea of gradual progression through repeated play. Each time the player starts they continue with any skills, upgrades and weapons from previous attempts. The player isn't expected to beat the game quickly, rather fail and get stronger over time to increase the chance of victory with each attempt. In our game, the player controls a hero climbing a randomly generated tower filled with enemies with unique themes on specific floors. The ultimate goal is to reach the top of the tower and defeat the final boss. As the player climbs and defeats enemies they unlock new skills and weapons to aid them in future attempts.

Figure 1.5 – Rogue Legacy (2013)

Another less serious idea that was put forward was QWOPTOPUS, a game based on the browser game "QWOP" (figure 1.6) in which a player has control of the upper and lower halves of a runner's leg's and aims to try and run as far as possible. In our game, QWOPTOPUS, the player would control each tentacle of an octopus and attempt to navigate through environments filled with fragile items. The player would receive a higher score by completing levels quickly and with minimal damage to the environment.

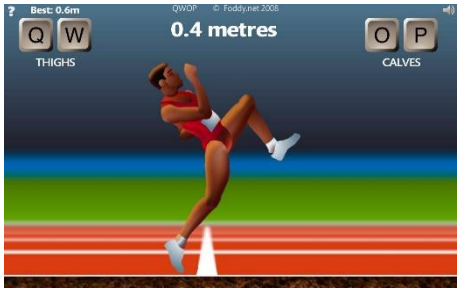


Figure 1.6 – QWOP (2008)

Figure 1.7 – Super Mario Kart (1992)

The final game idea we came up with was a top down racing game in which the player races around various crazy courses littered with obstacles to avoid in a similar style to the older "Mario Kart" games (figure 1.7). Racing against computer opponents the player must drive skilfully while using a variety of different items to help them and hinder their opponents.

Game Design

Chosen Idea – Kamek's Tower

After weighing up the various game ideas we had we decided to opt for the fourth idea, the 'rogue-like' tower climbing game, which later became 'Kamek's Tower'. We considered both QWOPTOPUS and the arcade-style platformer too open-ended, and thought they would not drive the player to complete the game, given how they had less clear goals. Whilst the rpg-style game would allow us to design a large open world we felt that the gameplay offered would be too simple and would limit our design potential. However, 'Kamek's Tower' offered us the chance to try and design unique gameplay opportunities based on the theme of each floor creating novel experiences for the player. Whilst developing it may be a challenge we figured this idea would produce the best final product even in an incomplete state.

Backstory –

The player plays as a noble hero set out on a quest to slay the evil necromancer, Kamek, who has been terrorizing the land. They travel across the land to his tower to slay him, only to find the tower is far taller than the hero could have ever imagined. Within, the hero will encounter strange magical rooms filled with beasts, demons, and the necromancer's failed experiments. After finally climbing through rooms frozen with mystical ice, coated in dense spider webs, and a burning inferno the hero reaches the top and must face the evil necromancer Kamek in a final battle to decide the fate of the land. This progression can be seen in the storyboarding section of appendix, section 3.

Features and gameplay

- The game will provide keyboard controls for both movement and attacking with short response time so the game performs in real time.
- The game will be played from a 2-D side on perspective, where the player is at the centre of the frame and the environment surrounding them.
- The game will feature 10 unique floor themes each with their own mechanic and enemy types. Some of these can be seen in the appendix, section 3.
    1. Base tower theme – Castle like environment with simple enemies such as bats and skeletons.
    2. Overgrown – Huge vines have taken over these floors, watch out for carnivorous plants and deadly insects.
    3. The Alchemist's Lab – Strange experiments roam free with magic elixirs and wonderful concoctions leaking everywhere.
    4. Underwater – The floors have been flooded; swim through making sure to keep getting air bubbles to stay alive whilst fending off deadly sharks.
    5. Slime – Viscous green slime coats the walls and floor oozing together forming large slime monsters that split apart into smaller enemies on death.

6. Glacial – A cold bitter wind has frozen the tower; razor-sharp icicles line the ceiling and dangerous ice patches have made the floor slippery.
7. Spiders Nest – Taking up residence near the top the tower the spider queen has weaved her web throughout the floors leaving her hatchlings waiting to strike.
8. Blazing Inferno – The burning heat of dragon's breath has incinerated the floors leaving lethal pools of lava and fire demons running amok.
9. The Void – Kamek's powerful magic has torn apart the fabric of space and time itself, exposing the void beneath, and affecting gravity! Falling into the void is a fate worse than death, so the player must be careful.
10. Throne Room – Residing at the top of the tower the decisive battle with Kamek awaits in the corrupted throne room.

- At the beginning of the game the player starts with weak equipment and unlocks better gear from killing floor bosses or randomly acquiring them from monster drops. The quality of the weapons is based on how high the player is, and the difficulty of the monster killed.
- Floors higher in the tower feature tougher enemies with more health that deal more damage as well as having more complex floor mechanics to deal with.
- A variety of different weapons will be available each with their own advantages and disadvantages. For example, a spear has a longer reach than a sword but deals less damage.
- Collision between the player and enemies/projectiles serves as the main survival condition, the player must avoid enemies and defeat them to progress.
- Upon death the hero is expelled from the tower falling to their death, leaving their previous equipment ready for the next new adventurer to take.

Game Design Principles

Whilst designing any game it is important to focus on key principles to design the game for its key purpose: fun. This can be broken down into six main principles: pursuing and achieving goals – how to keep the player interested; interactivity – making sure the player isn't just an observer; feedback – how the player gets information from the game; gameplay variety – preventing the game from becoming boring by constantly engaging the player; gameplay fairness and consistency – making the game predicable for the player; and avoiding repetition – keeping the game interesting by avoiding monotonous repetition.

Pursuing and Achieving Goals

Within the game there are several different goals for the player to pursue. Ultimately the long-term goal is defeat the final boss, the evil necromancer Kamek, however along the way there are a multitude of other goals the player may implicitly pursue. Floor bosses provide the player with medium-term goals to try and achieve, these may start off as unreachable

but over time as the player gets stronger these become possible and after finally beating it provides the player with a sense of satisfaction as well as rewards in the form of new weapons. Shorter term goals might include things such as unlocking a new weapon from a random drop or reaching a higher floor than a previous attempt. Each of these goals provide the player with something to aim for, to encourage continued play, in addition to the progression within the game itself.

Interactivity

Having positive interactions with the game is crucial for the player to maintain interest in the game itself. Intuitive controls that are mapped tightly to actions within the game help remove the frustration caused during the initial learning process of playing the game for the first few runs. Once these controls are mastered the player no longer focuses on what buttons to press but rather on the patterns of enemies to maximise their chance of victory. Therefore, the controls will be simple and intuitive, WASD keys for movement and arrow keys for directional attacks.

Aside from factors like the control scheme the gameplay itself must allow the player to feel like the decisions they make are impactful and have a real effect on the game. Having hidden rooms within our game allows for this as the player is rewarded for performing special actions or noticing a secret indicator. Providing a noticeable bounty incentivises the player to search thoroughly and makes their actions have a clear payoff.

Feedback

Whilst playing the game the player shouldn't struggle to get feedback from the game itself. Information should be obviously visible on the user interface that is displayed the player during the game. The players current health as well as other stats such as weapon damage and current floor number will always be displayed on the screen to help the player make decisions based on their current situation. In addition, upon hitting an enemy or getting hit damage numbers will be displayed close the area of impact to focus attention on it and making it evident to the player immediately which entity has taken damage. Localised health bars above enemies further improve clarity whilst also enabling the player to make strategic decisions such as focusing down the weaker enemies first before tackling stronger, beefier enemies.

Gameplay Variety

Variety in gameplay must also be taken into consideration when designing any game. Offering the player multiple strategies and methods of achieving goals within the game helps keeps gameplay varied. To achieve this variety in our game we chose to have different types of weapons available to the player; daggers offer a short range but balance this with the ability to strike quicker than most other weapons; ranged weapons like crossbows offer the player to deal damage from a safe distance however reloading leaves the player open to attack. With unique benefits and drawbacks weapon choice provides varied situations for the player throughout the run, they may struggle on certain floors using a dagger however this may provide incredibly beneficial later in the game. This ability to mix and match

weapons to suit the situation as well as to the players strengths helps provide the variety necessary for fun gameplay.

Gameplay is also varied due to the nature of our level design – floors remain consistent within their theme, but after defeating the respective boss and moving on, the theme switches up, meaning gameplay will too.

Gameplay Consistency and Fairness

Gameplay consistency and fairness is imperative to keeping any game compelling and fun to play. Random losses and unusual events must be mitigated to prevent player frustration due to the iterative essence of the core gameplay. In our game every enemy has a consistent base health and damage they deal which is scaled up as the player gets higher in the tower. This means the difficulty should gradually ramp up at roughly the same rate as the players skill. Pairing this with the clear information displayed to the player from the UI the player should know exactly what is happening always throughout the game.

Avoiding Repetition

The game being from the 'rogue-like' genre of games makes it inherently repetitive, the core gameplay is based around gradual progression through multiple attempts at the game, for the game to remain fun this repetition must be abstracted away from the player. In games like "Binding of Isaac" (figure 2.1) this repetition is counteracted by having many randomly chosen items within the game each run. Each item modifies the player in a separate way, some altering the players stats others completely modifying how the player deals damage. This combination of random items and potential synergies makes each run feel unique even though the core gameplay each time remains the same. This allows the player to master the core gameplay mechanics and focus less on the ingrained repetition.

Figure 2.1 – A selection of items available in "The Binding of Isaac" (2011)

Within our game we will be using a similar strategy in an attempt to eliminate the feeling of repetition, randomising the floors and enemies within them. This means that each time the player starts a new run the game will play slightly differently, not only does this help keep the game feeling fresh but it also introduces some difficulty variation. Certain runs will randomly be harder than others creating new challenges for the player and forcing them to adapt to trickier situations.

Game Rules

To maintain consistency and fairness in the game a core set of rules is crucial. These rules can be split into three distinct categories - Constitutive rules, abstract core mathematical rules that define the essential game logic; Operational rules, rules the player follows the shape the gameplay itself; Implicit rules, 'unwritten' rules of behaviour that go unstated within the game.

Constitutive Rules

- C.1. The player's movement is restricted by the walls within the game indicated by a solid outline within the environment.
- C.2. Upon contact with an enemy or enemy projectile the player will take damage relative to the strength of that enemy.
- C.3. Before each run of the game the player can choose their weapon those that they have collected previously.
- C.4. The player has 100 hit points.
- C.5. Once the player has left the current floor they are unable to return to it.
- C.6. If an entity (player or enemy) loses all the hit points they die and are removed from the game.
- C.7. The player has a constant movement speed.
- C.8. The player can jump a set maximum distance horizontally and vertically.
- C.9. Each weapon has a consistent damage and speed it can be swung/shot with.

Operational Rules

- O.1. To advance up a floor in the tower the player must defeat all enemies on the current floor.
- O.2. After beating all floors in a set floor theme, the player must defeat that theme's boss to advance into the new set of floors.
- O.3. The player can only have one weapon equipped, which is chosen before the player starts a run.
- O.4. Hidden rooms can be accessed by performing special actions such as pressing a hidden button on the current floor.
- O.5. Upon death the player is expelled from the tower, falling to their death, and leaving their equipment for the new adventurer to collect.
- O.6. Each time a player enters the tower the floors are randomised.

Implicit Rules

- I.1. The player mustn't have unlimited health.
- I.2. The player can't beat the game without first defeating every other floor.
- I.3. Enemies deal a consistent set amount of damage.
- I.4. All entities (players and enemies) are killable.

Software Engineering

Functional and Non-Functional Requirements

In conjunction with the game principles and rules there are certain functional and non-functional requirements that must be met in to order to ensure the game is produced properly. Functional requirements define critical things the game must do, whereas non-functional requirements are less critical, however must be testable.

Functional Requirements

- The system must allow one user to play the game.
- The system must listen to some input device to interpret the user's actions.
    - The system must allow the user to control their character using a keyboard.
- The system must allow the user to save and load their game states.
- The game must challenge the user to overcome obstacles before they progress.
- The game must be beatable - by defeating the final boss (Kamek).
- The game must allow the user to acquire more weapons, as to give the game variety.
- The game must have sound effects.
- The game must have a variety room themes, each set followed by their respective themed boss.
- The game must have no game-breaking (either crashing or cheating) bugs.
- The game must be played from a 2-D side on perspective
- The game shall have a title screen with buttons on to start the game.
- The player shall have to kill all enemies on the current floor to progress.

Non-Functional Requirements

- The game shall have a failure rate of no more than 5%.
- The game shall start within 3 seconds.
- The game shall be positively reviewed - at least an average of 75/100.
- The game shall have a maximum input response time of 20ms.
- The game shall run at rate of at least 30fps.
- The game shall run on standard hardware (lab computers).
- The game's code shall be maintained using appropriate version control (github).

Use cases

Below is a use case diagram for our game (Figure 3.1) showing basic use cases of the player from the main menu of the game. In addition to the diagram there are also several more detailed use case tables (Figures 3.2, 3.3, 3.4) describing in more depth the key actors and success conditions for these use cases.

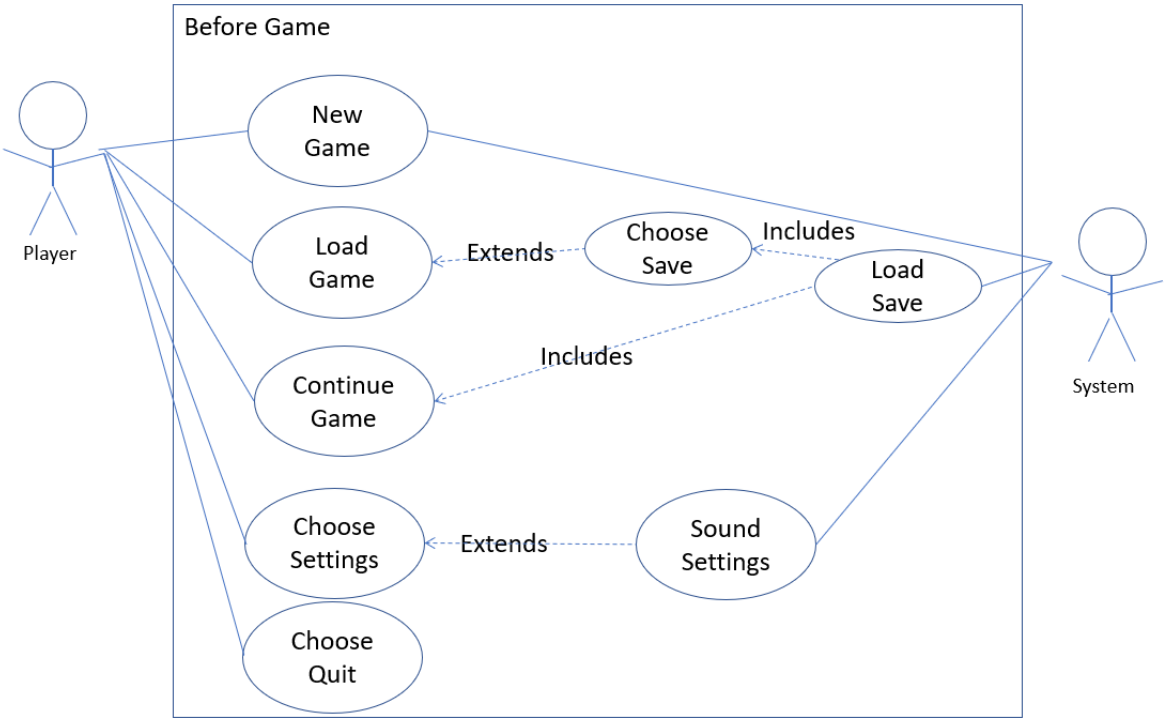


Figure 3.1 Use Case Diagram

Use Case Name:	**New Game**
Scope:	'Kamek's Tower'
Primary Actor:	Player
Secondary Actor:	System
Summary:	The player creates a new game save and begins the game.
Preconditions:	The player has started the game and is on the title menu screen.
Main Success Scenario:	1. The player clicks the New Game button 2. Player enters name for new save 3. Player confirms the name 4. Game begins
Alternatives:	2.a. Player doesn't enter a name; default name is used
Exceptions:	3.a. Player doesn't confirm the new save so it is never created.
Post-Conditions:	The player has successfully created a new save and has started the game.

Figure 3.2 Use case table for the 'New Game' use case

Use Case Name:	**Change Settings**
Scope:	'Kamek's Tower'
Primary Actor:	Player
Secondary Actor:	System
Summary:	The player accesses the settings menu and alters them.
Preconditions:	The player has launched the game and is on the main menu or has paused the game and is on the pause menu.
Main Success Scenario:	1. The player clicks the settings button in the menu. 2. Player selects a setting to alter. 3. Player alters the setting. 4. Player confirms the setting change. 5. Player leaves settings menu.
Alternatives:	3.a. Player doesn't alter setting, skip this.
Exceptions:	4.a. Player doesn't confirm the changes, settings stay unchanged.
Post-Conditions:	The player has modified the game settings and exited the settings menu.

Figure 3.3 Use case table for the 'Change Settings' use case

Use Case Name:	**Continue**
Scope:	'Kamek's Tower'
Primary Actor:	Player
Secondary Actor:	System
Summary:	The player continues from a previous game state.
Preconditions:	The player has started a game previously and is on the title menu screen.
Main Success Scenario:	1. The player clicks the continue button 2. Player selects which save to continue from. 3. Player confirms the selection. 4. Game resumes from the previous state of that save.
Alternatives:	2.a. Player has only one save, selected automatically, skip.
Exceptions:	3.a. Player doesn't confirm so never continues.
Post-Conditions:	The player has successfully resumed a previous game state of a given state.

Figure 3.4 Use case table for the 'Change Settings' use case

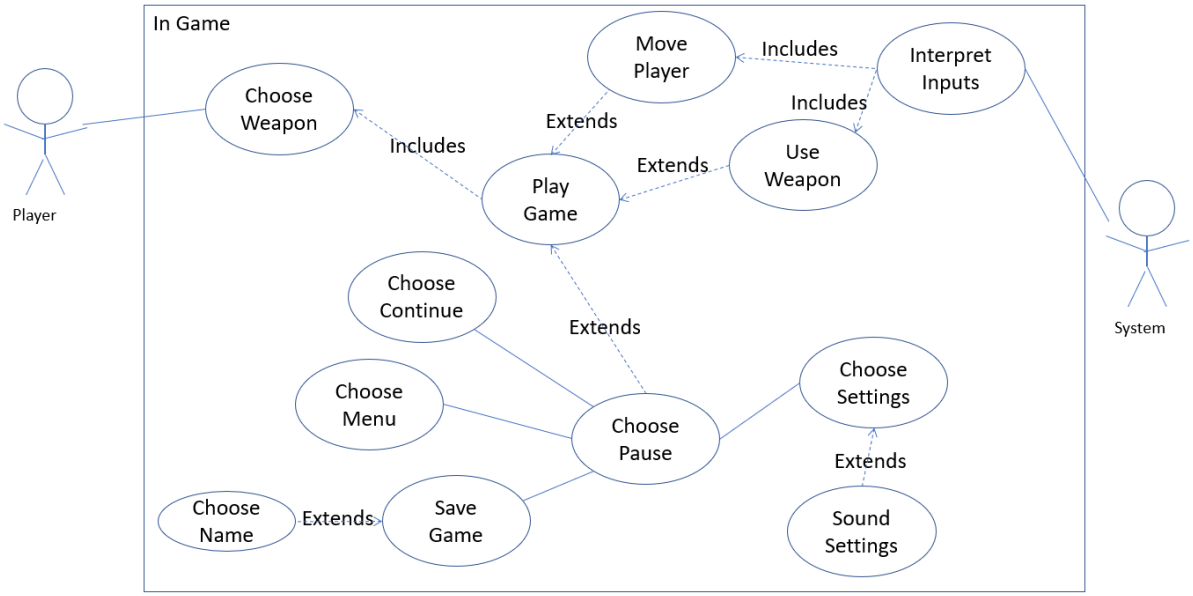


Figure 3.5 In Game Use Case Diagram

Use Case Name:	**In Game**
Scope:	'Kamek's Tower'
Primary Actor:	Player
Secondary Actor:	System
Summary:	The player plays the game.
Preconditions:	The player starts and plays the game. The user also is provided a pause menu and in-game save functionality.
Main Success Scenario:	1. The player chooses weapon and starts game 2. The player plays the game using movement and attack controls 3. The player enters the pause menu 4. The player saves the game 5. The player chooses the save name 6. The player confirms the save name 7. The player then exits back to the main menu
Alternatives:	4.a The player enters the setting menu and alters the sound settings. 5.a The player does not enter a name, so a default name is used.
Exceptions:	6.a The does not confirm the new save so it is never created
Post-Conditions:	The player has played the game, saved and returned to the main menu

Figure 3.6 Use Case table for the in game use case

Acceptance Test Cases

Testing is an important part of an iterative design process, to help with testing the game correctly meets the requirements we generated many acceptance tests. Each individual test case has an expected input and the desired output, if this output is met correctly the test case passes and can be checked off in the table.

ID	Description	Input	Expected Output	P/F
1	Game has a response time of 20ms	Arrow keys/WASD	Action is completed within 20ms	
2	Player can attack in 8 directions	Combination of arrow keys	All 8 inputs result in the character attacking in the correct direction	
3	Player can move left and right	A/D	Player moves left, and right	
4	Player can drop through thin platforms and stairs	S on a thin platform, stairs	Player drops down	
5	Player can jump	W/Space	Player jumps	
6	Player cannot move through walls	Any movement option near a wall	Player does not pass through wall	
7	Defeating a boss allows for movement to another level	Defeat a boss, enter portal	Travels to next level	
8	Defeating the final boss will complete the game	Defeat the final boss	Recognize achievement, eventually lead back to main menu	
9	Each level has at least one valid path of completion	Complete many levels	Not all paths to bosses were entirely blocked	
10	Reducing the health of a player to 0 or lower ends the game	Take large amounts of damage	Player dies and restarts	
11	Reducing the health of an enemy to 0 removes that instance of the enemy	Damage an enemy until its health is reduced to 0	Enemy dies; cannot damage player any further, cannot be seen	

ID	Description	Input	Expected Output	P/F
12	Picking up a weapon makes it available to the character on subsequent runs	Pick up a weapon, then die, begin a new run	Weapon is available to equip	
13	Pause menu halts game state	Pause key	Movement of all entities stop, state based actions such as damage from lava do not continue	
14	Available weapons persist between game instances	Check weapons, close game, reopen game and check weapons again	Weapons remain the same	
15	Sound effects are audible	Regular gameplay	Sound effects are audible	
16	Check levels are generated pseudo-randomly	Play multiple runs	Levels should be identical only very infrequently or not at all	
17	Weapons have different attributes	Use a variety of weapons	Weapons deal varying damage, at various distances, with varying effects	
18	Game state can be saved and resumed from after completing a boss	Defeat a boss, close game, reopen and resume game	Game continues at the start of a floor	
19	Game consistently updates the screen at rate of 30 fps or higher on lab machines	Play game as normal on a lab computer	Framerate is consistently 30 fps or higher	
20	Users on average rate the game 75/100 or higher	Let multiple users play the game and request a rating when they are finished	Users rate the game 75/100 or higher on average	

Implementation Plans

Task List

Shown below (Figure 4.1) is the list of tasks for the first nine weeks of designing the game. The table is broken down into weeks as well as individual tasks key to designing the game. The column on the far right shows which member of the group completed the task, shown by a “C” and green highlight.

Weeks	Task	Status					
		Adam	Tom	Jacob	Rebekah	Jiahao	James
1-2	Come up with game ideas	C	C	C	C	C	C
	Create task list	C					
	Sketch ideas	C	C			C	C
	Make notes for report	C	C				
3-4	Choose idea	C	C	C	C	C	C
	Name idea		C				
	Sketch out game features	C	C				C
	Develop gameplay ideas	C	C	C	C	C	C
	Make notes for report			C			
5-6	Describe software requirements	C	C	C	C	C	C
	Produce game principles and rules	C	C	C	C	C	C
	Make notes for report		C		C		
7-8	Risk assessment				C		
	Milestones					C	
	Compile software requirements	C	C				
	Deliverables					C	
	Use cases						C
	Activity network			C			
	Gantt chart	C					
	Test Cases			C			
	Finish Report		C				

Figure 4.1 Task List for the first 9 weeks.

Risk Assessment

As the project spans over an extended period, it is necessary to produce a risk assessment for potential problems which the group could come across. This analysis will aid the group in how deal with recovery from these potential problems.

This assessment below (Figure 4.2) details how to compensate for the potential risks what we may encounter during the project.

Num.	Risk	Description	Likelihood	Impact	Contingency Plan
1.	Team member is unable to contribute.	A member of the team is unable to participate and contribute to the work either due to health reasons or lack of motivation.	Medium	Low	The delegation of tasks will need to be updated so that the team member in question's work can be completed.
2.	Programming work is lost	All programming work made thus far can no longer be accessed and the project will need to be started from the beginning.	Low	High	There will be multiple backups of the work made by each member of the team, both locally and in a shared environment (e.g. github)
3	Documentation work is lost	All documentation work for the project is lost and will need to be redone.	Low	High	There will need to be backups of the documentational work, keep both on personal locations and in a shared environment.
4	Group disagreement	Members of the team disagree over certain aspects of the project	Medium	Medium	A group vote could be conducted to solve the contrasting ideas of the disagreeing group members
5	Contrasting coding styles	Group members may have different methods of coding certain aspects of the project which may lead to other members not being able to understand their code	Medium	Medium	The group should decide on a set style before implementation begins. Detailed comments should be included within each member's work so that the other members can understand what is occurring.

6	Team member fails to meet a deadline	A team member may get behind on their work contribution and therefore doesn't meet a set deadline	Medium	Low	The work load would need to be redistributed to aid the lacking team member

Figure 4.2 Risk Assessment table

Milestones

Throughout the implementation of the game there are many crucial milestones that highlight core functionalities of the game. Within the table below (Figure 4.2) these milestones are listed along with any dependencies and the expected start and completion dates.

#	Description	Internal Dependencies	Acceptance Test	Start Date	End Date	Duration
1	Code design			Week 9	Week 10	1 weeks
2	Implementation of Interfaces	1		Week 10	Week 11	2 weeks
3	Add input from users		Test Case 1-6	Week 10	Week 11	2 weeks
4	Art development			Week 10	Week 16	6 weeks
5	Menu (Health bar, Time, Marks and etc)	2	Test Case 10, Test Case 13, Test Case 14	Week 12	Week 13	2 weeks
6	Weapons	2	Test Case 12,17	Week 12	Week 13	2 weeks
7	Collision (sword, daggers, shield, spear, range references, fists)	2,3,6		Week 12	Week 13	2 weeks
8	Room themes			Week 13	Week 13	1 week
9	Enemies	2	Test Case7,11	Week 13	Week 14	2 weeks
10	Valid path completing game	1,3,6,7,9	Test Case 9			
11	Sound effects	1, 6, 7, 9	Test Case 15	Week 13	Week 14	2 weeks
12	Levels of game	8	Test Case 7,8,16	Week 13	Week 14	2 weeks
13	Save and load game	1-12	Test Case 18	Week 13	Week 15	3 weeks

14	Testing	1-13	Test Case 1-18	Week 13	Week 16	3 weeks
15	Fully runnable	1-14		Week 16	Week 16	1 weeks

Figure 4.3 Milestones of the project

Activity Network

Below is activity network (Figure 4.4) showing the different key stages in the development of our project. Tasks are allocated sufficient time so that the group can complete them all in addition to the report. A total of 22 weeks is allocated for planning and design, 40 for implementation, and 8 for testing, with the 20 remaining weeks reserved for the report. Between the 6 group members this should be easily achievable in the given time frame.

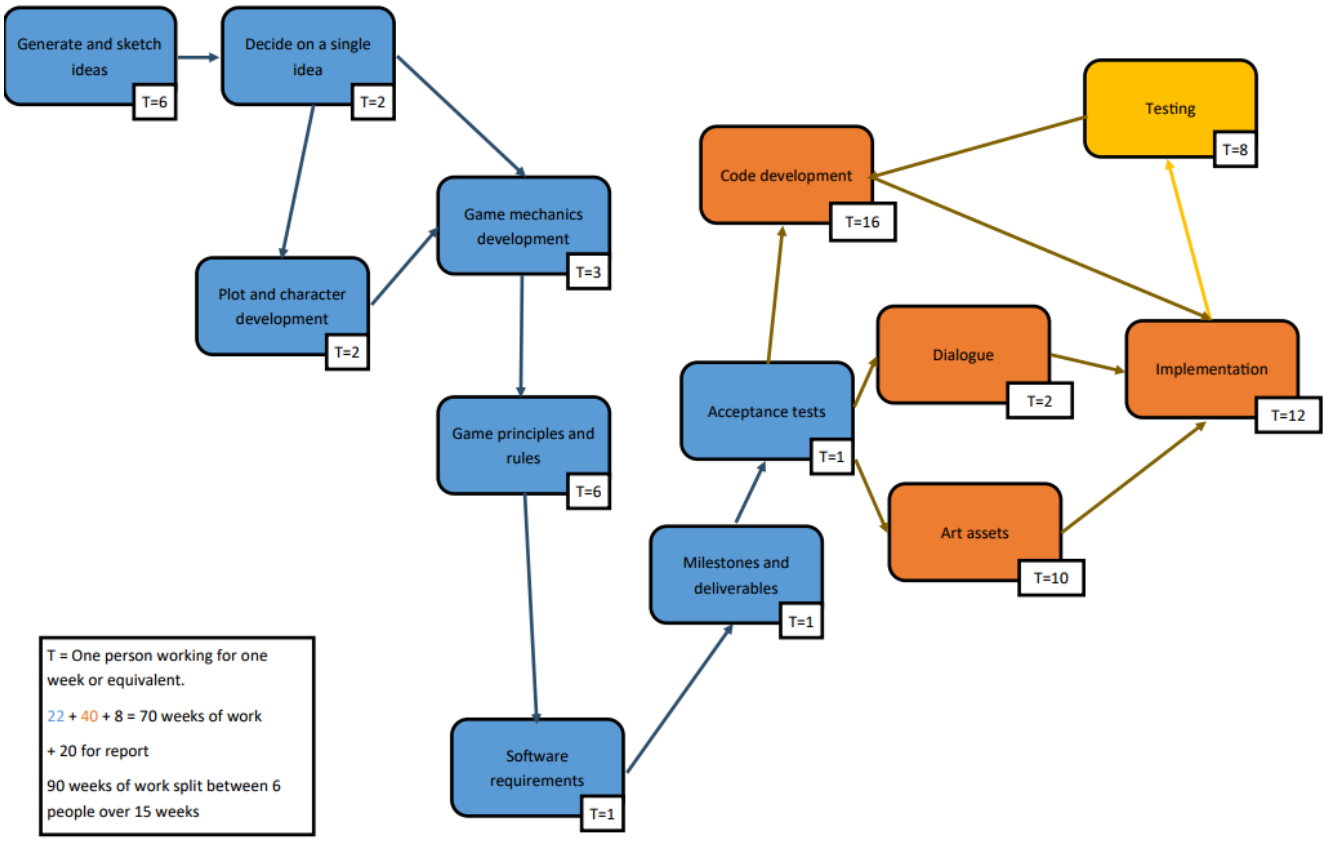
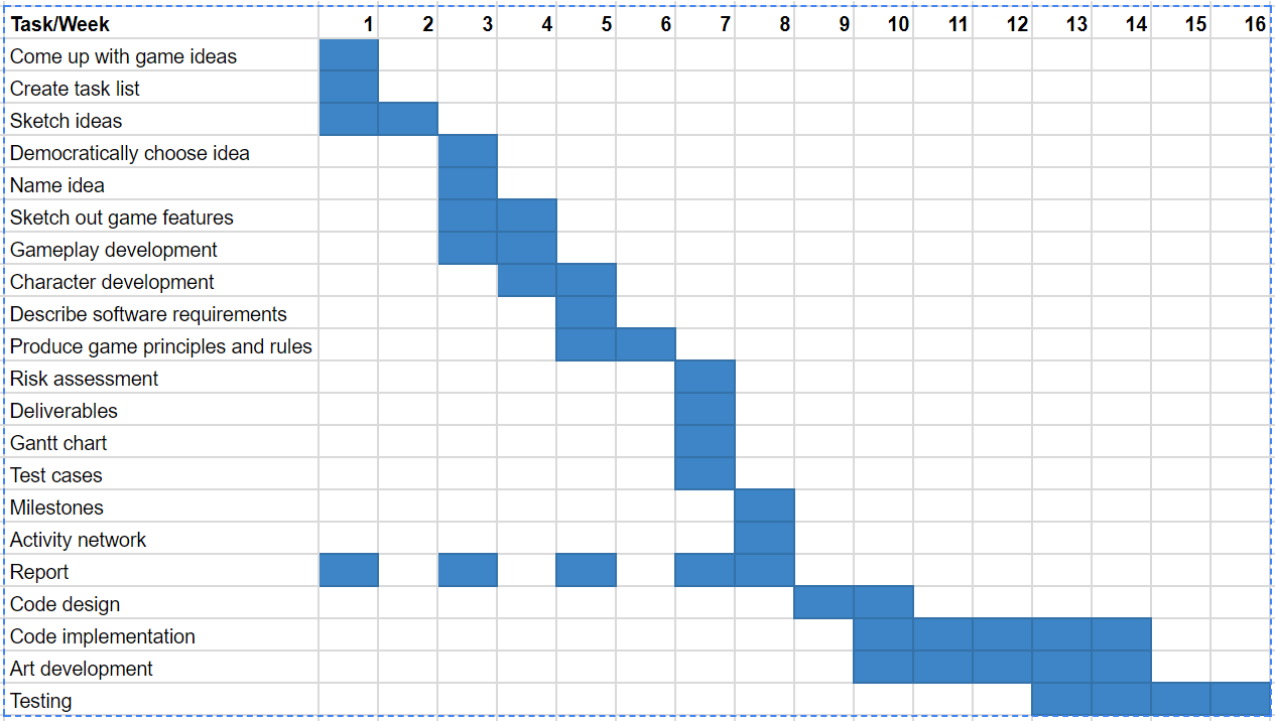


Figure 4.4 Activity Network of the project

Gantt Chart

Another chart used to help plan out the implementation of the game was a Gantt Chart. This chart shows key tasks within the project and the weeks we aim to complete them in. Certain tasks such as the art development and code implementation can be done simultaneously as they don't directly affect each other.

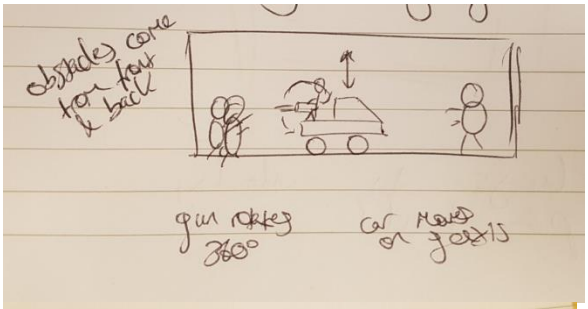


Appendix

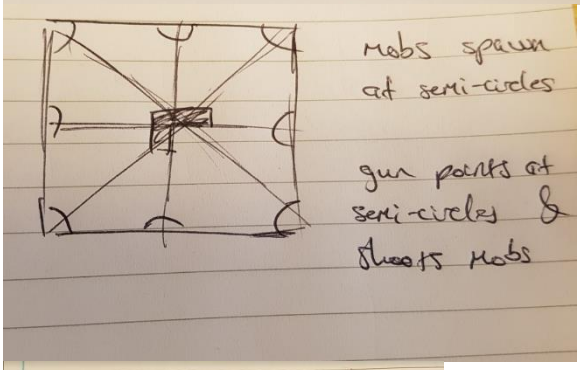
Sources

Figure 1.1 https://nicoblog.org/wp-content/uploads/2017/06/Pokemon-GSC-Johto-AzaleaTown.png
Figure 1.2 https://headsup.boyslife.org/files/2017/05/super_mario_bros.0.png
Figure 1.3 https://shockingvideogamesecrets.files.wordpress.com/2011/12/the-legend-of-zelda-nes.jpg
Figure 1.4 https://farm1.staticflickr.com/405/19106581654_a0141c0110.jpg
Figure 1.5 https://cdn.vox-cdn.com/uploads/chorus_asset/file/2755014/RogueLegacy_2013-06-18_20-20-33-581.1372810664.jpg
Figure 1.6 http://www.foddy.net/wp/wp-content/uploads/2010/10/qwopss.jpg
Figure 1.8 http://download.gamezone.com/uploads/image/data/1164544/Super_Mario_Kart_-_SNES_-_1.jpg
Figure 2.1 http://nintendotoday.com/wp-content/uploads/2017/04/binding-of-isaac-items.png

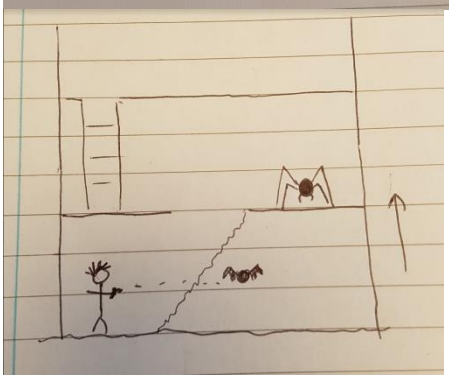
Game Sketches



An idea for a side-scrolling zombie survival game.

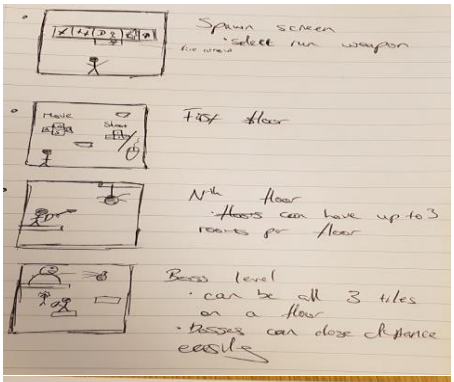


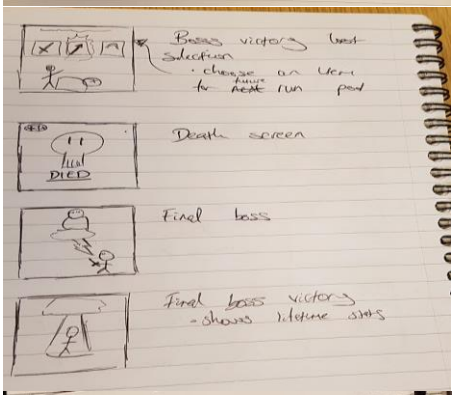
An idea or an arcade shoot-em-up style game.



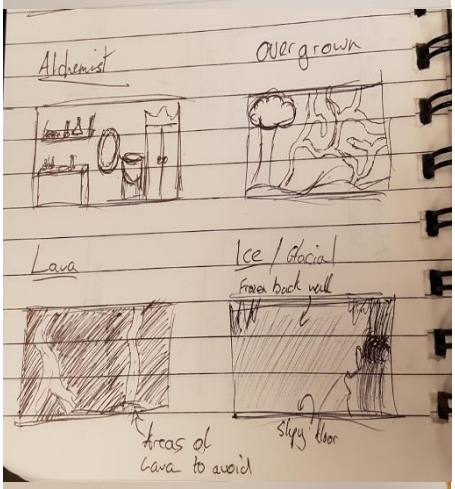
An idea for a shoot-em-up roguelike, which would evolve to become our ‘Kamek’s Tower’.

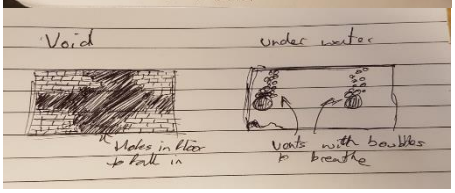
Kamek's Tower Sketches





A storyboarding of how the game could start, progress, and end.





Original sketches expanding on ideas for every theme we had come up with at the time of designing