

DM2198 Game Development Project

Academic Year 2018/2019 Semester 1

Final Report

Team Number / Name:

Team 1

Project Title:

The Great Escapade

Members:

Kendrick Sim

Lim Yan Quan

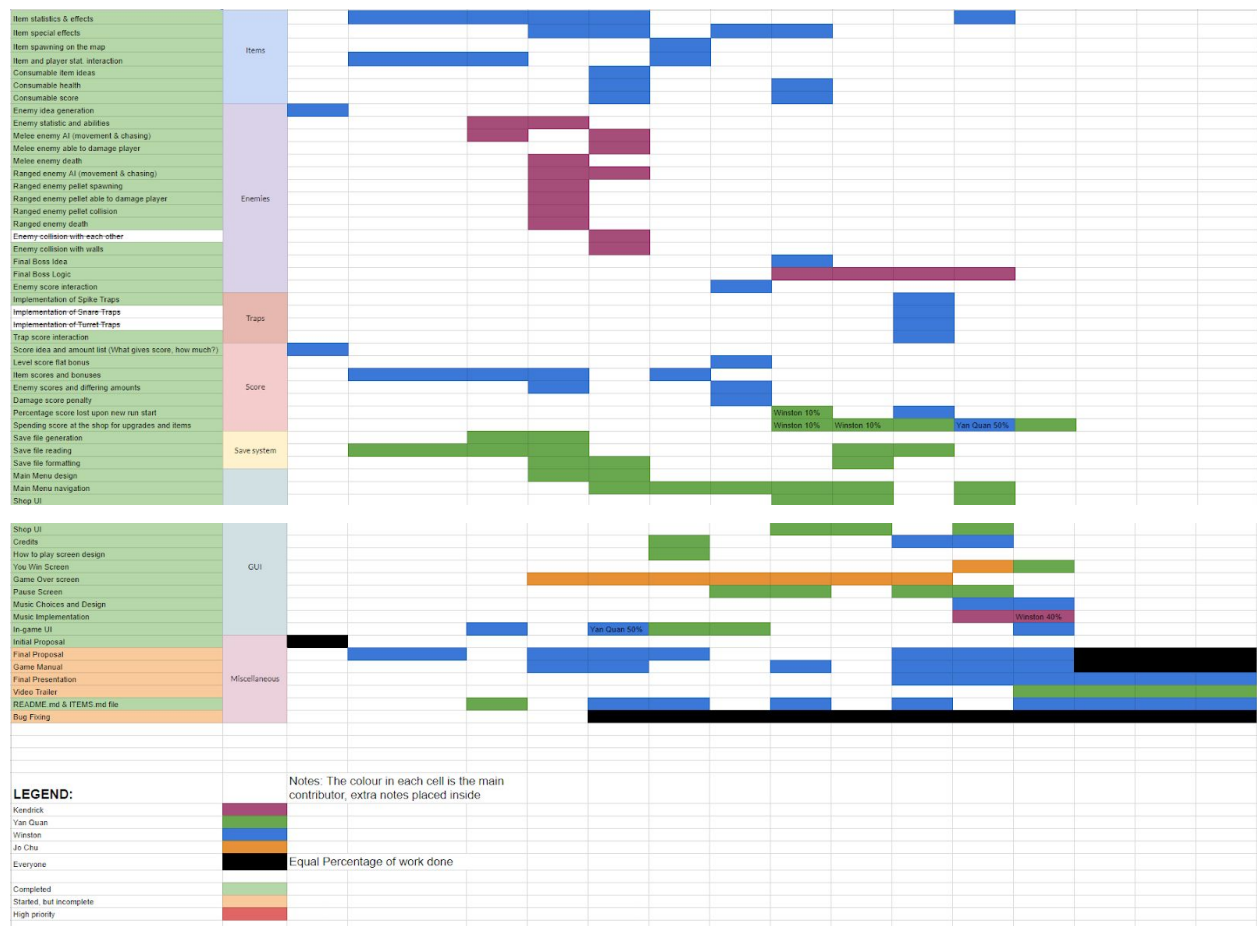
Winston Ngoui

Pi Jo Chu

This is a **Roguelike** game that implements the core elements of many roguelike games such as randomized item spawning, random map generation and the permadeath mechanic. The game is set in a tower dungeon where the player was captured by an evil overlord and held captive in the top floor of the heavily guarded tower. The player's objective is to climb down the tower, floor by floor, to reach the bottom and escape from the tower, whilst battling the many minions stationed within the tower who are tasked with preventing the player from escaping. This game takes heavy influence from The Binding of Isaac, Crypt of the Necrodancer, Risk of Rain, Rogue Legacy and Spelunky. The player starts off with the same amount of health, damage and movement speed every run. They can collect items that randomly spawn throughout each floor to boost their stats and give them different fighting abilities to help them reach the bottom floor.

Gantt Chart:

PROJECT SCHEDULE		WEEK 0							WEEK 1							WEEK 2							WEEK 3			
		Monday - Friday		Monday 13-8		Tuesday 14-6	Wednesday 15-6	Thursday 16-8	Friday 17-8	Monday 20-8	Tuesday 21-8	Wednesday 22-8	Thursday 23-8	Friday 24-8	Monday 27-8	Tuesday 28-8	Wednesday 29-8	Thursday 30-8								
Feature List	Category																									
Door Spawning Algorithm	Level Generation																									
Room Spawning Algorithm																										
Room Content Spawning																										
Generation of Exit room & Spawning room																										
Generation of mandatory routes to exit																										
Single room rendering		Yan Quan (35%), Wuxian (10%)																								
Double size of room render																										
Mipmap Generation																										
Fog of War																										
Preset room design for each level		Yan Quan 40% Yan Quan 40%																								
Preset room implementation																										
Player Death																										
Map scrolling effect when moving to next room																										
Level Reset for next level																										
8-direction player movement	Movement																									
Player collision with walls																										
Player collision with enemies																										
No collision doors																										
Closed doors when in battle																										
8-direction player shooting	Attacking																									
Pellet spawning																										
Pellet sprite movement																										
Pellet collision with walls																										
Pellet collision with enemies																										
Pellet timer and eraseal																										
Player melee damage																										
Player health	Player Stats																									
Player max health																										
Player damage																										
Player range																										
Item idea generation																										
Item statistics & effects																										
Item special effects																										



Week 1 Start of Studio Project (Completion of core gameplay mechanics):

13/8: - Complete Level Generation (Kendrick)

- Start on Score System & Save System (Yan Quan)
- Start of Player Stats. & Items (Winston)

14/8: - Start on Enemy AI (Movement & Melee Attacking) (Kendrick)

- Continue on Score System & Save System (Yan Quan)
- Continue on Player Stats. & Items (Item stat. increase) (Winston)

15/8: - Complete basic Enemy AI (Movement & Ranged attacking)

- Complete Save System & File I/O (Yan Quan)
- Continue on Player Stats. & Items (Item upgrades and interaction with score) (Winston)
- Start on Player death logic (Jo Chu)

16/8: - Complete item spawning mechanics from preset room generation (Kendrick)

- Complete melee enemy AI and ranged enemy AI (Kendrick)
- Start on Level 1 (Yan Quan)
- Continue on save system and start on main menu (Yan Quan)
- Complete Player Stats. & Item implementations (Winston)
- Continue on Player death logic (Jo Chu)

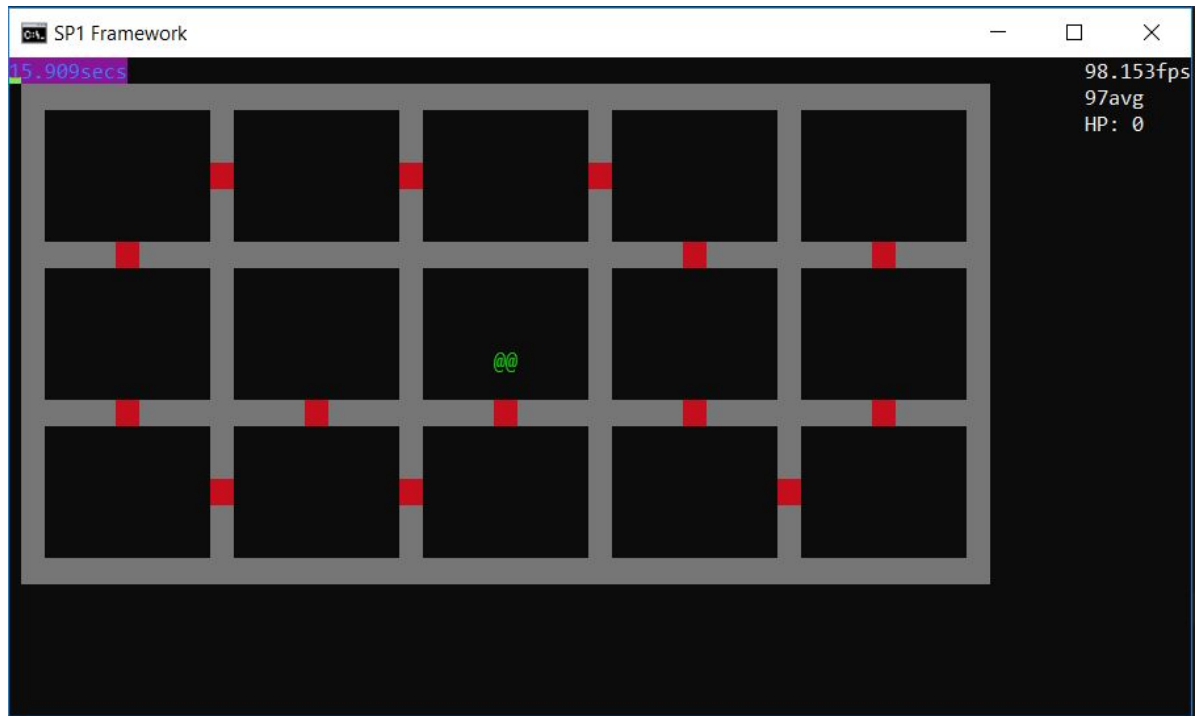
17/8: - Complete Minimap and Fog of War (Kendrick)

- Start on Player scoring with other in-game mechanics (e.g items, enemies) (Winston)

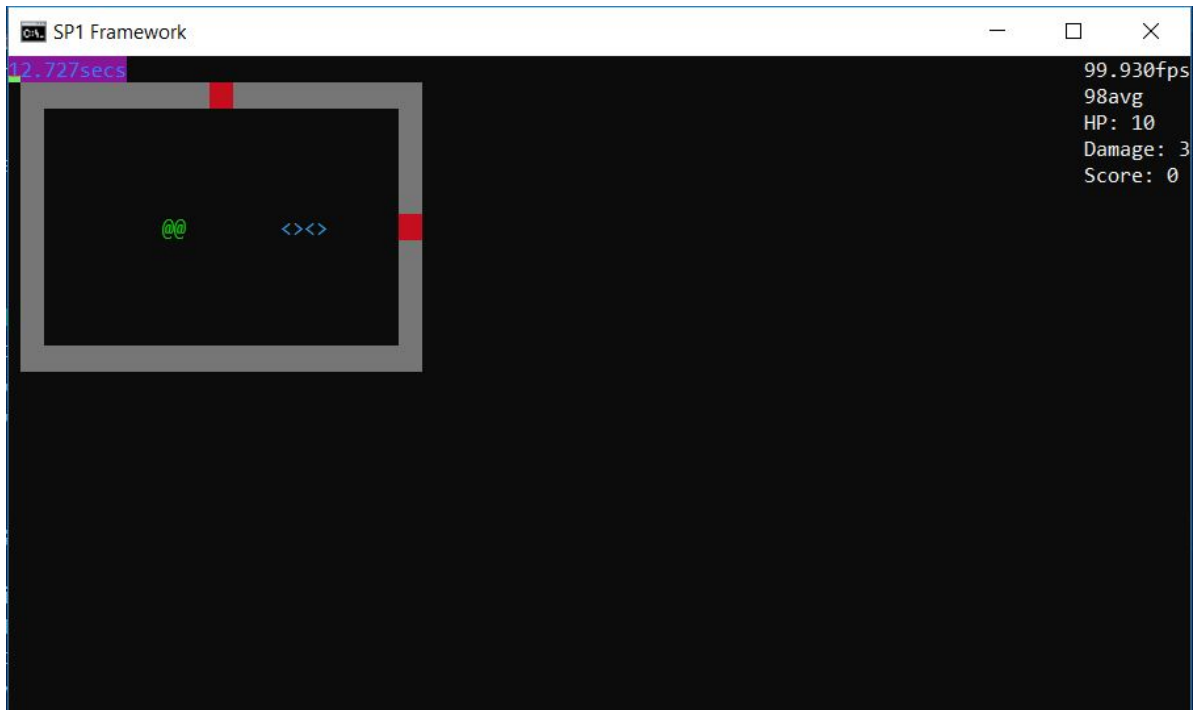
- Continue on Player death logic (Jo Chu)
- Continue on Main menu navigation and logic (Yan Quan)
- 20/8: - Major Bug fixing with current build.(Kendrick)
 - Complete scrolling effect between rooms. (Kendrick)
 - Complete Player scoring with other in-game mechanics (e.g items, enemies) (Winston)
 - Continue on Main menu navigation and logic (Yan Quan)
 - Continue on Player death logic (Jo Chu)
- 21/8: - Start on Final Boss Logic (Kendrick)
 - Continue on Main menu navigation and logic (Yan Quan)
 - Start on Shop UI and Logic (Yan Quan & Winston)
 - Yan Quan: Shop UI and coding of shop navigation
 - Winston: Reference to item level variables in player.h & item.h, help in coding of shop navigation
 - Complete Player death logic (Jo Chu)
 - Continue on Levels 4 and 5 room designs (Winston)
- 22/8: -Continue on Final Boss Logic (Kendrick)
 - Complete Main menu navigation and logic (Yan Quan)
 - Complete Shop UI and Logic (Yan Quan & Winston)
 - Yan Quan: Shop UI and coding of shop navigation
 - Winston: Reference to item level variables in player.h & item.h, help in coding of shop navigation
 - Complete Level 4 and 5 room designs (Winston)
- 23/8: - Continue on Final Boss Logic (Kendrick)
 - Polish and beautify Main Menu and Sub-menus (Yan Quan and Winston)
 - Yan Quan: Polish up shop UI and credits
 - Winston: Main Menu and How to Play
 - Balance room presets and continue on misc. Documents (Final report, presentation slides, game manual) (Winston)
 - Complete adding Spike Traps (Winston)
- 24/8: - Complete Final Boss Logic (Kendrick)
 - Improve user experience by moving UI closer to map (Winston)
 - Beautify Shop UI and optimize item upgrades (Yan Quan)
 - Start on "You Win" Screen (Jo Chu)
 - Fix bugs through play-testing (Everyone)
 - Polish up implemented mechanics (Everyone)
- 27/8: - Complete addition of background music into the game (Kendrick & Winston)
 - Music implementation logic (Kendrick)
 - Music choice and implementation into the game using Kendrick's logic (Winston)
 - Complete "You Win" Screen (Yan Quan)
 - Bugfix and playtest (Everyone)
- 28/8: - Playtest and fix any bugs found (Everyone)
 - Start on presentation rehearsal (Everyone)
 - Polish up miscellaneous documents (Everyone)

- 29/8: - Playtest and fix any bugs found (Everyone)
- Start on presentation rehearsal (Everyone)
 - Polish up miscellaneous documents (Everyone)

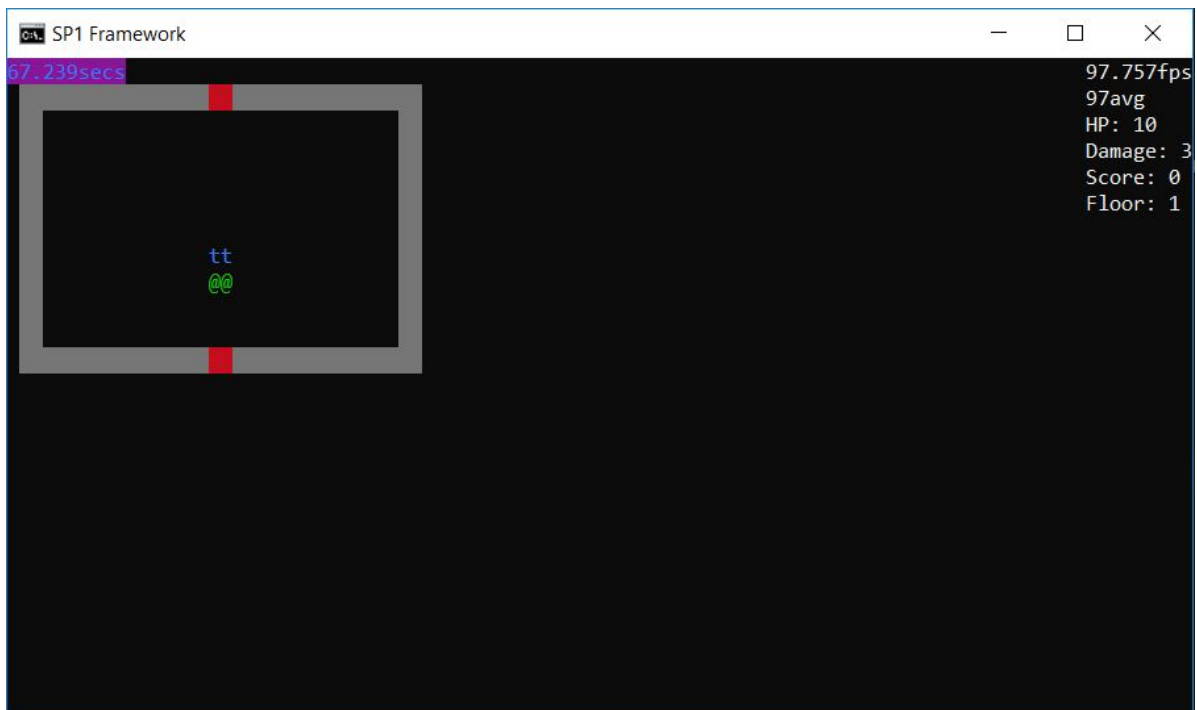
Screenshots



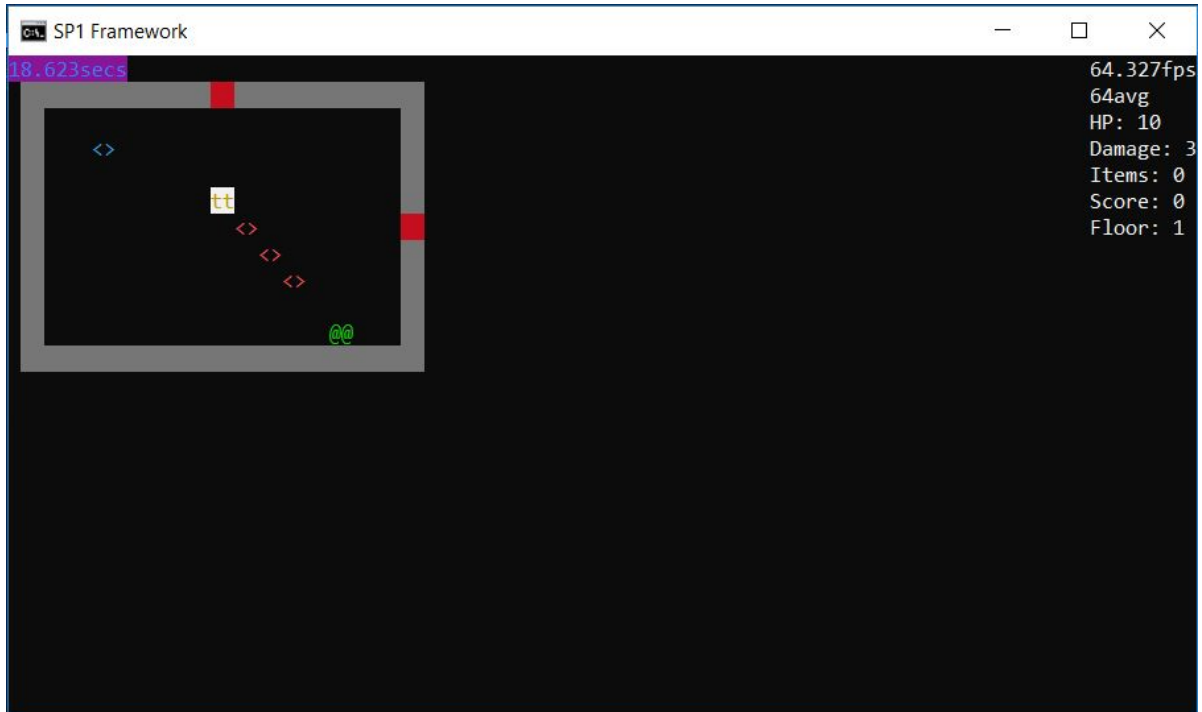
(WIP 1: First iteration of the game, with the entire map being shown in the console at once. Map generation, movement and shooting already implemented.)



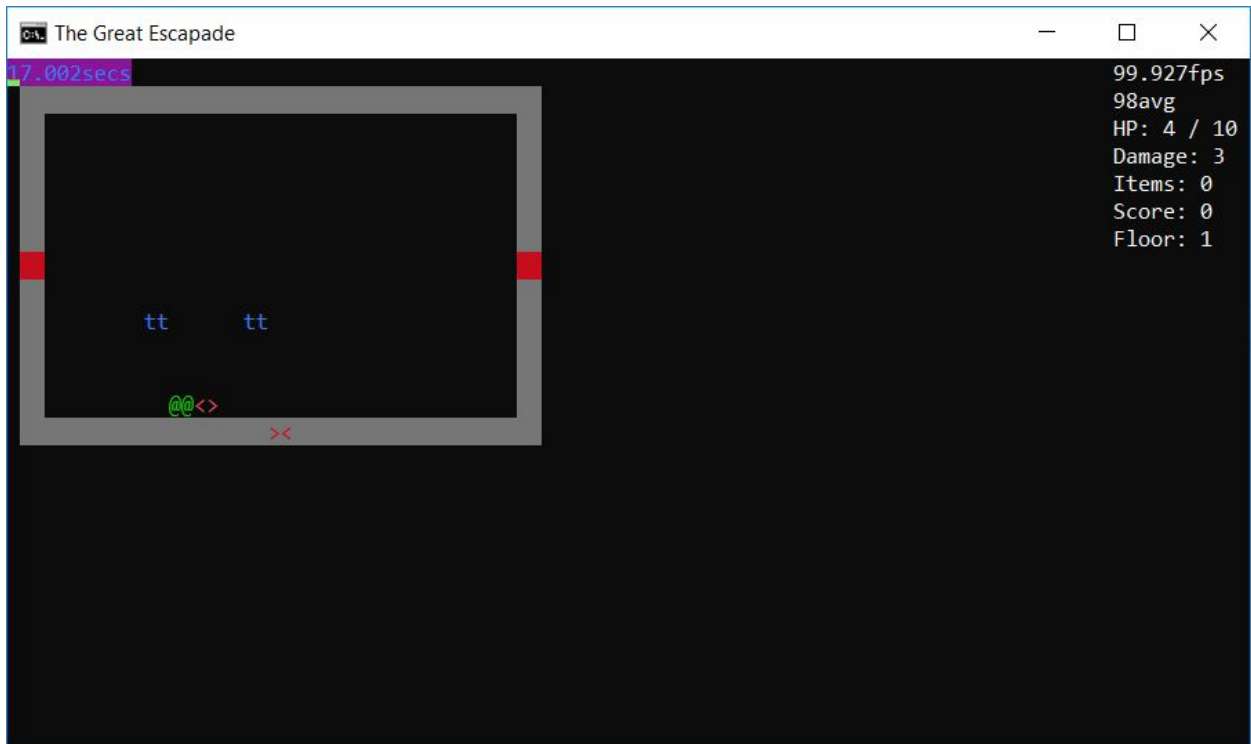
(WIP 2: Second iteration of the game. Only 1 room is shown on the console, but the entire map is already pre-generated. Changed map rendering in console from previous iteration)



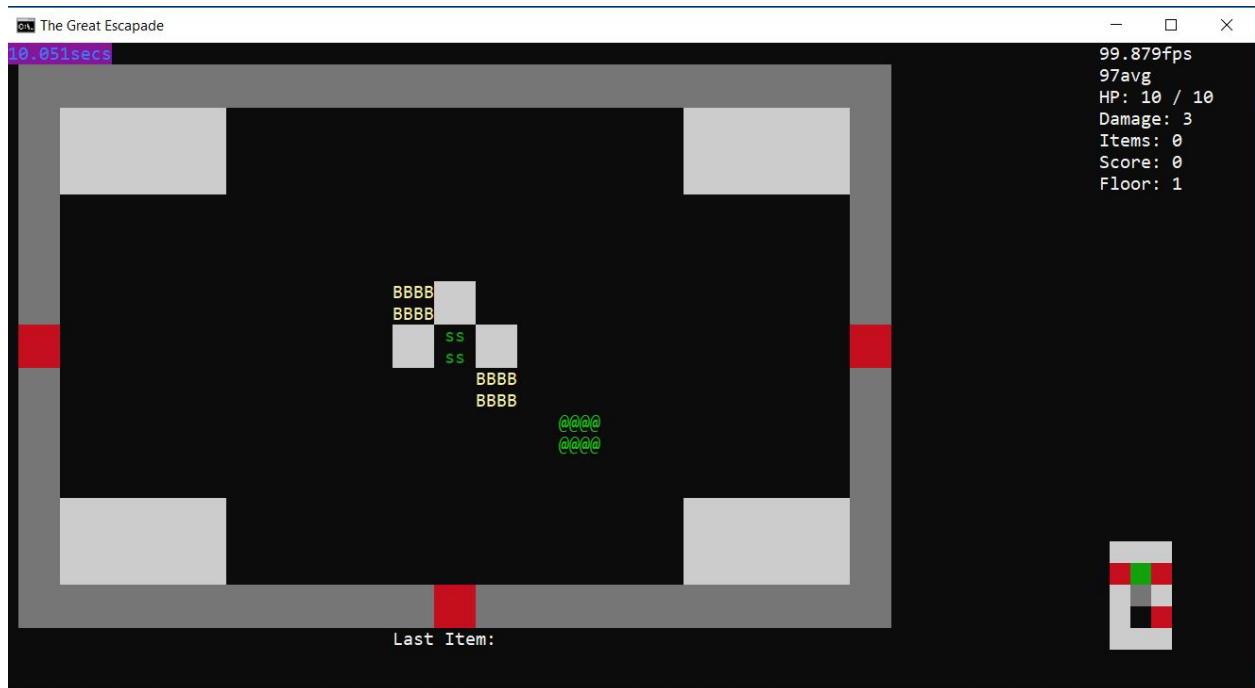
(WIP 3: Third iteration of the game. Basic enemy AI for both melee and ranged implemented (Enemy chasing player).)



(WIP 4: Third iteration of the game. Ranged enemies now shoot at the player when player is within its 8 angle vision. Enemies flash white when shooting as an indication)



(WIP 5: Fourth iteration of the game. Ranged enemy and Melee enemy AI are completely fleshed out and working)



(WIP 6: Fifth iteration of the game. Added mini-mapping, updated UI, added map scrolling effect, blown up map to double its previous size, implemented items and consumables, added scoring, added main menu and navigation)



(WIP 7: Sixth iteration of the game. Added a final boss that has 2 phases in melee and ranged, able to dash quickly towards the player. Shop UI beautified and optimised)

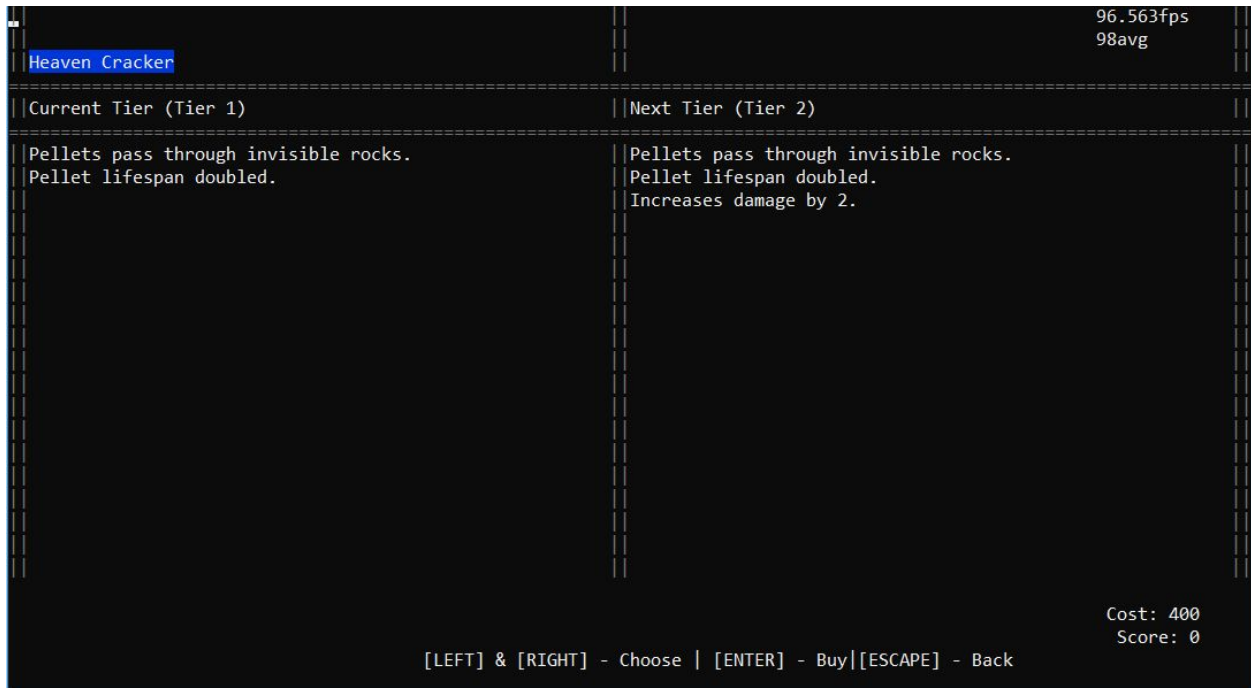


(Seventh and Final iteration (content freeze): Improved user experience, border of screen flashes red when hit and remains dark red when low on HP. Moved UI closer to map and finalised game over screen)

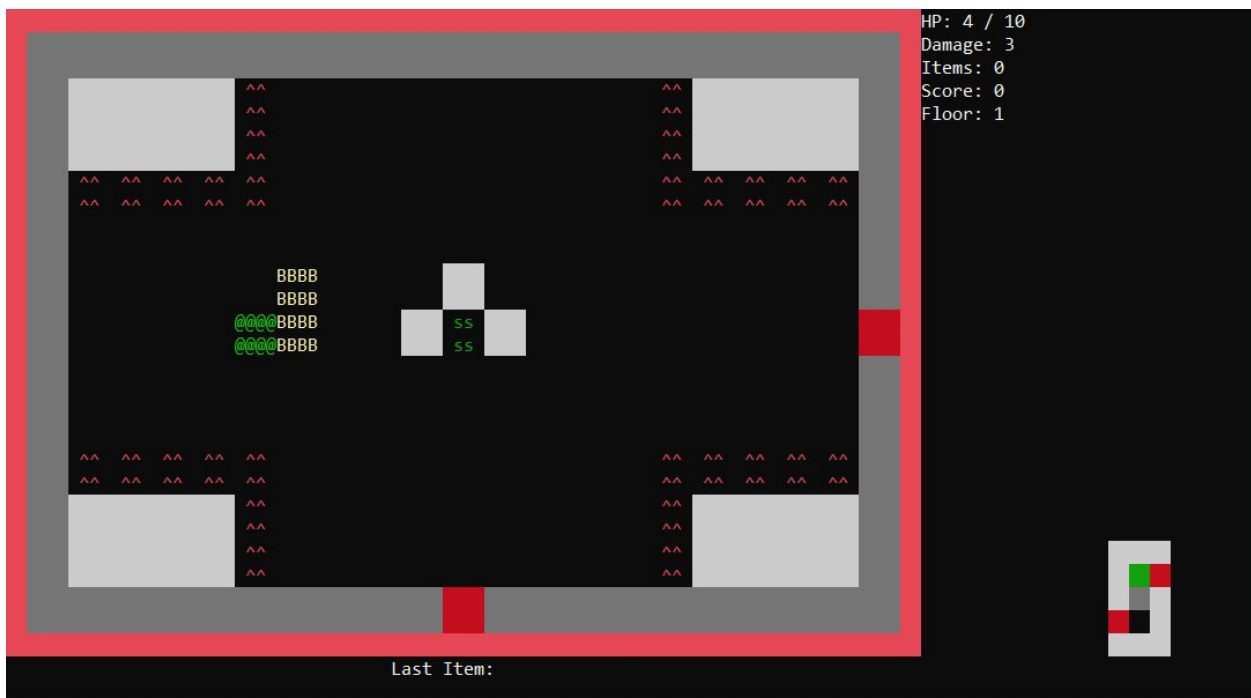
Gameplay Images



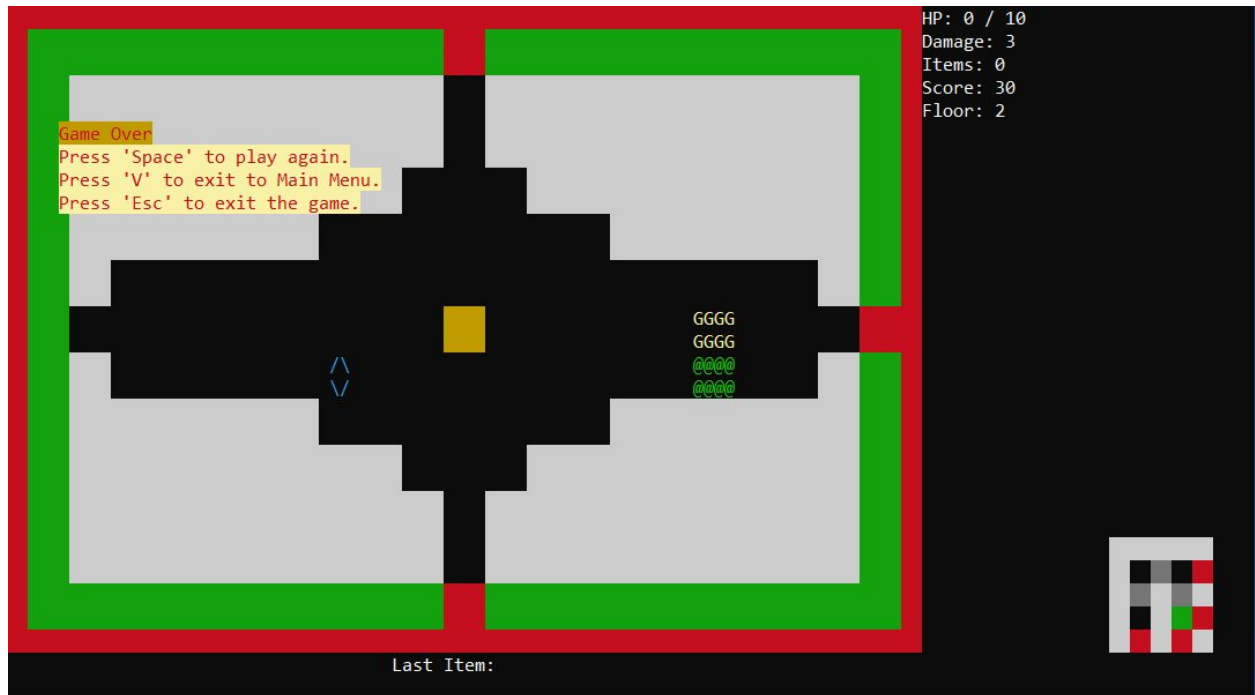
(Main Menu, player can choose between one of the options to proceed.)



(Shop, where players can use score earned through playing to upgrade items to the next tier)



(Example of a level 1 room, red border means player got hit, ss is a score consumable, bandit enemies chases down the player)



(Example of player dying on level 2, player gets to choose between quick restarting, quitting to menu or quitting to desktop, item room here is guarded by a guardian)



(Final boss fight with the evil overlord, has 2 attack phases of ranged and melee. Fires many pellets randomly in multiple directions in ranged, charges towards the player and bounces off walls in melee)

Description of Features

Preset Generation: The map is pre-generated when the player enters every floor, but only 1 room is rendered at a time in the console. Each room has a size of 11 tiles x 19 tiles, with a certain amount of enemies spawning around the room. Item rooms spawn as a different type of room, housing 1 item with Guardians spawning in item rooms. Each floor also has a different colour and theme to allow the player to know which floor they are on. Floor 1 has a grey border and is the tutorial level, Floor 2 has a green border and is very claustrophobic, Floor 3 has a blue border and has many invisible walls, Floor 4 has an orange border and has a lot of enemies, and finally Floor 5 has a purple border and is a combination of all the previous levels, incorporating invisible walls, many enemies and a sense of closed spaces.

Movement: The player is able to move in realtime in all 8 directions (N, NE, E, SE, S, SW, W, NW) and use **WASD** to move. To move diagonally, the player has to input two directions at the same time, as long as the inputs are not in opposite directions. (Example: W and A at the same time to move NW). The player can move 1 tile every 0.2 seconds, and inputs can be held down without the need of constantly pressing the button to move.

Combat: The player is able to shoot projectiles at a rate of 1 pellet every 0.35 seconds, pellets travel until they hit a wall, an enemy or until their lifespan runs out (default 2.5 seconds) and can be fired in all 8 directions as well. Players use the **arrow keys** as inputs for shooting. To shoot diagonally, the player has to input two directions at the same time, as long as the inputs are not in opposite directions. (Example: Up and Left at the same time to shoot NW). Enemies have a short delay and an indication where the enemy's background will flash white before attacking. The border will flash red when the player is hit, and will remain dark red if the player is low on HP.

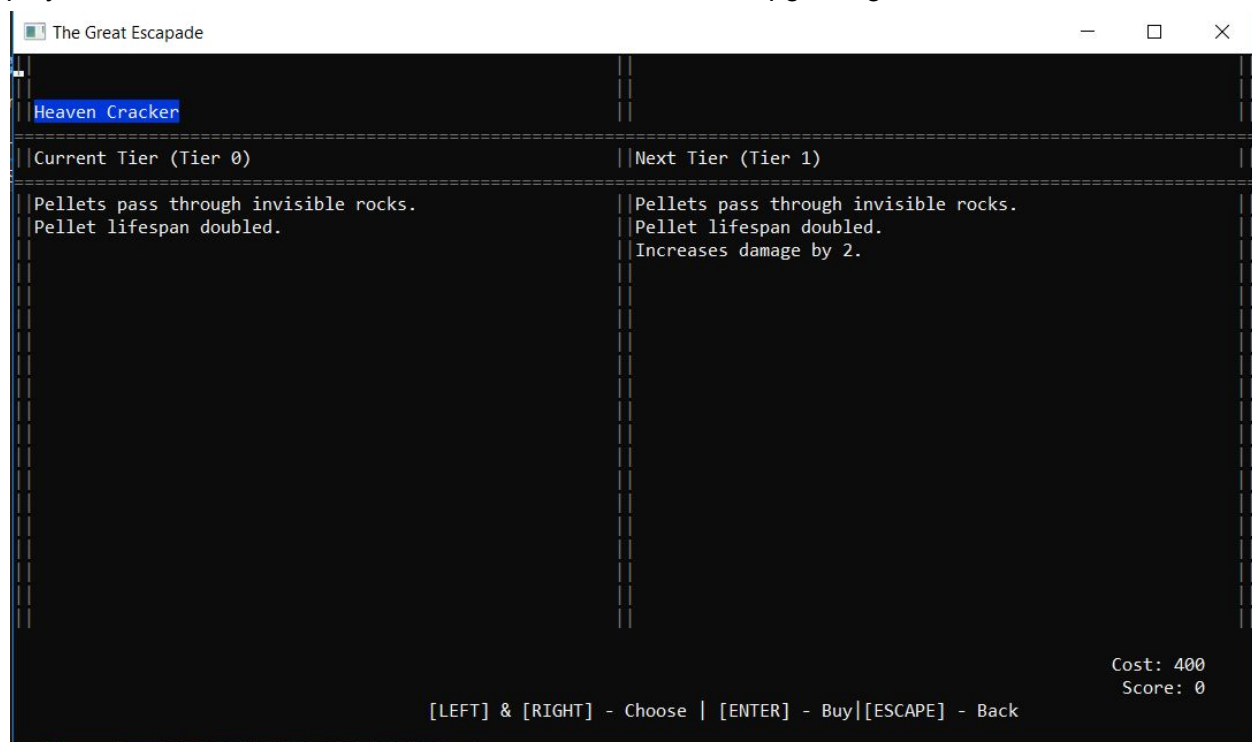
Player Statistics (Health and Damage): The player starts off with 10 HP and 3 Damage. HP and Damage are modified with certain items, such as the **Health Potion** increasing the player's health by 5 HP and the **Enchanted Sword** that increases the player's damage by 2 damage and health by 3 HP. The player loses health when hit by an enemy, and the amount of health lost depends on the enemy type, such as the **Knight** which damages the player by 5 HP and the **Bandit** which damages the player by 3 HP.

Death: The player will die when their HP drops to 0 or less, and cause a game over. Dying will not cause the player to lose their progress in their save file in terms of item upgrades and accumulated score, but will cause the player to lose all their items in their current run. Dying will not cause a loss in score, but when the player starts a new run, 90% of all remaining score is lost.

Unit Collision: The player and enemies in the game have collision enabled, meaning they can't pass through walls nor through the player, and vice-versa. Unit collision also applies to pellets shot from the player and enemies, which causes the pellet to disappear when it hits a wall or invisible stone. However, if the player has the item Heaven Cracker, they will be able to shoot through invisible stone.

Score System: The game has a score system that rewards the player for eliminating enemies (depends on enemy type) and clearing floors, while punishing the player if they get hit by enemies (-5 points per hp lost). The score gained by the player at the end of each run is saved into an external save file and accumulated, which can be used for upgrading items in the game for better stats. Score is not lost when the player dies, but 90% of the accumulated score is lost when the player starts a new run. As such, the player should try to spend as much of their accumulated score buying item upgrades before starting a new run. The score shown in-game is the player's score in the current run, while the global score shown in the menu is the player's accumulated score after each run and is used for shopping.

Shop System: The game also has a shop, which allows the player to spend their global score to purchase item upgrades that increase each item's effectiveness, to give the player a sense of progression and accomplishment. The shop makes use of the item struct in item.h to give the player some information on each item tier and the cost of upgrading each item.

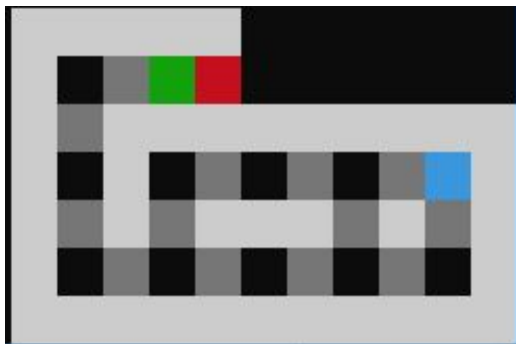


Save System: At the end of each run, the score gained is saved in a text file, which is retrieved when the game is launched the next time. Item upgrades are also saved in the save file, so the player will keep everything they upgraded from the last time they played. This makes use of file I/O, where the program will read a text file containing information about the save such as the global score and the levels of each item.

Room Design: Each room has a different design that aims to encourage variation in how the player approaches the room's enemies. Rooms usually contain stone and invisible stone that act as cover, and/or as a trap to increase the difficulty of the room. Invisible stones have no




indication of being there, but rooms containing invisible stones are guaranteed to be symmetrical, so players can use half of the room with visible stone as a reference. Rooms with no visible stone at all require the player to feel their way around. They can shoot at invisible stone to understand where they are, as pellets get destroyed by invisible stone. Rooms can also contain spike traps, which deal 4 HP of damage to the player. Within the program, spike traps are referenced as a “^”, regular stone as a ‘#’ and invisible stone as a ‘!’.

Minimap & Fog of War: On the bottom-right side of the console, a minimap of the entire floor is displayed that helps to inform the player of their current location within the floor. The minimap updates in realtime as the player explores the map. The room the player is currently in is displayed as green, while the exit room is displayed as blue. Rooms that have yet to be explored have their connecting doors displayed as red.



Consumables: Consumable items spawn in preset locations within different rooms throughout the level, and are meant to aid the player in their escape. There are 4 consumables: Minor Health Potion, Greater Health Potion, Small Medal, and Large Medal. Potions give the player some amount of health back, not exceeding their maximum health, and medals give the player an amounts of score.

Consumable List

	Name	Represented character	Sprite	Effect
1.	Minor Health Pack	‘1’		Restores 5 health to the player, not exceeding maximum HP. Cannot be picked up when at full HP.
2.	Greater Health Pack	‘2’		Restores 10 health to the player, not exceeding maximum HP. Cannot be picked up when at full HP.
3.	Small Medal	‘3’		Player gains 20 score.




4.	Large Medal	'4'		Player gains 50 score.
----	-------------	-----	---	------------------------




Enemies: Enemies spawn throughout the level in rooms and are the main obstacle to the player. There are 2 types of enemies that spawn - ranged and melee.

Enemies in the game have their own AI, depending if they are melee or ranged. Melee AI tries to move next to the player and attack, while ranged AI keeps a distance from the player and shoots when the enemy can directly shoot at the player. Enemies have a visual indication right before they attack, giving the player some time to dodge incoming attacks.

The Evil Overlord Final Boss has his own unique AI, incorporating both ranged and melee elements. In Ranged form, he fires multiple pellets rapidly in-line or diagonally, chosen at random. He also moves randomly but cannot move and shoot at the same time. In Melee form, instead of moving towards the player, he will charge directly at the player quickly and will bounce off the player if it hits the player, dealing damage on contact. He has friction that decelerates him when he charges and will bounce off walls if he collides with them. He will switch between the two forms after a period of time, and will have a clear visual indication when doing so by flashing a different colour before changing. When the boss dies, it will flash white for a duration, before spawning the exit stairs in the middle of the room.

Enemy List

	Name	HP	Damage	Speed (Delay between movement in seconds)	Attack Length	Attack Threshold	Represented character	Sprite	Abilities
1.	Bandit	12	3	0.3	0.25	0.1	'b'		Melee attacker, same speed as the default speed of the player
2.	Mage	9	2	0.4	0.3	0.1	'm'		Ranged attacker, can either attack or move at a time
3.	Knight	25	5	0.2	0.3	0.1	'k'		Melee attacker, slightly faster than the default speed of the player.

4	Enhanced Sorcerer	15	2	0.35	0.4	0.1	'e'		Ranged attacker, attack and move at the same time
5	Guardian	35	5	0.18	0.3	0.1	'g'		Melee attacker, faster than the default speed of the player, a lot of HP. Only spawns in item rooms.
6	Evil Overlord	1000	4/5	0.35	0.5	0.3	-		Can switch between melee and ranged. Charges towards the player, bounces off walls with friction to slow down in melee and shoots multiple pellets in ranged.

Items: An Item spawns in all item rooms, with 1 guaranteed on each floor. There are 7 items in the game, where each have unique effects and stat upgrades. Item rooms are almost always guarded by one or more Guardian enemies, and sometimes by mages.

Item List

	Name	Cost to upgrade	Effect	Upgrade Level 1	Upgrade Level 2	Upgrade Level 3
1.	Heaven Cracker	400 (+150 per tier)	Player's projectiles pass through invisible stone. Pellet lifespan doubled	Damage increased by 2.	Damage increased by 3.	Damage increased by 4.
2.	Enchanted Sword	600 (+150 per tier)	Damage increase by 2, health increased by 3.	Damage increase by 3, health increased by 4.	Damage increase by 4, health increased by 5.	Damage increase by 5, health increased by 6.
3.	Health Potion	500 (+200 per tier)	Increases the player's HP by 2	Health increase by 4.	Health increase by 6.	Health increase by 8.
4.	Glass	500 (+150)	All enemies deals	All enemies	All enemies	All enemies

	Canon	per tier)	2 more damage. The player deals 4 more damage	deals 3 more damage. The player deals 5 more damage	deals 4 more damage. The player deals 6 more damage	deals 5 more damage. The player deals 7 more damage
5.	Magic Potion	500 (+150 per tier)	Increases attack speed & projectile velocity by 20%	Attack speed & projectile velocity increase by 30%	Attack speed & projectile velocity increase by 40%	Attack speed & projectile velocity increase by 50%
6.	BONUS!	300 (+200 per tier)	Multiplies the player's score by 1.5x	Multiplies the player's score by 2x.	Multiplies the player's score by 2.5x.	Multiplies the player's score by 3x.
7.	Blue Feather	500 (+150 per tier)	Increases the player's movement speed by 20%	Movement speed increases by 30%	Movement speed increases by 40%	Movement speed increases by 50%

Knowledge Applied

We chose our game mechanics in line of the **Survival** core mechanic that we learnt from Principles of Game Design, where the gameplay that resulted would have a **smooth flow** where the game is not too difficult and not too easy. There needs to be a good balance between **risk and reward** for there to be motivation to play our game. An example of this in practice is balancing the melee class with more damage and health. Additionally, enemies have a visual indication before attacking, which is especially true with the final boss, where he flashes before changing forms and before attacking. Our game is geared more towards the **Exploration** side of **Fun**, where the map is unknown and the player has to explore each room without any knowledge of what might be inside them. This factor of not knowing what is going to happen adds another layer of fun into the game. **Feedback** is clear to the player as a red border is shown to alert the player when hit or low on health. The player is also rewarded with stat improvements whenever they obtain an item (for example, the health potion which increases the player's health by 5.), thus having a clear feedback of the player's actions helps to make the game more immersive. We also applied what we have learnt in making the game manual for the game, ensuring that we effectively conveyed information to the player for a more immersive experience and providing useful information if the player reads the manual.

As for C++ programming, we frequently used **Structs** to store information. For example, we used structs to store the properties of a particular pellet - what direction is it going in, whether it is friendly or not, the next time it should update its location, etc., and the use of structs in items, where item statistic boosts are stored as structs as well for more organised coding. **Functions** were frequently used as well to execute similar processes at the same time and for

organisation. For example, we used many functions for map generation, where some functions are in charge of generating the doors of the map, some are used to guarantee a pathway to the exit, and others to set the boundaries of the map. **Pointers** were consistently used throughout the entire project in function parameters and for using variables in other .cpp files. **File I/O** was used in the save system and music. The save system used external text files that stored information as integers. The program would read and write in these text files whenever an event happens, such as when an item is bought or when score is gained after a run. Music files were loaded and played with the help of file I/O as well. **Loops** were used in situations where repeating a piece of code was needed, such as checking for unit collision and generating a floor. **Vectors** were used when storing multiple similar pieces of information into a single place, such as for having all the unique items into a single vector, and having pellets shot stored in a vector temporarily. **Array of characters** were sometimes used in place of strings to enable for easier character manipulation.

Linear Algebra was used during the development process as well, especially in the final boss AI. We used trigonometry to calculate the angle that the boss charges towards the player, direction vectors to calculate the distance moved in the X and Y coordinates, and physics to calculate the velocity decrease and distance moved during a charge. Simple indices were used in the calculation of score, such as when starting a new run, 90% is lost.

Finally, we applied what we have learnt in **Communication Skills** for our final presentation on 30th August Thursday. We made sure to maintain eye contact and project our voice to enable our content to be heard loud and clear. The presentation slides were also made in line with the guidelines taught in comm skills, with easy to understand content without too many words and with the use of visuals. We tried to refer to our scripts as little as possible and rehearsed multiple times before the actual presentation to make sure we were ready.

Problems Faced

Room Size: Our original design called for rooms sized 3 tiles x 5 tiles. Once we created the layout, however, we found that this size would severely limit the maneuverability of the players and the enemies, resulting in decreased ability to dodge projectiles and everything feeling cramped. The solution was to increase the size of the rooms, which presented another problem: there is insufficient space on the console to display the entire level. The solution to this problem was to only render the room that the player is currently in, which also allows us more flexibility to changing the map layout.

GitHub Merging: When working on the project, we frequently committed and pushed out updates for others to continue working on the latest version of the code. However, when we push out code that had changes in the same place as another person's changes, it caused GitHub to duplicate existing code with extra useless text, causing the program to break as Visual Studios could not read the useless text. We fixed the issue by manually removing the

useless text in the filters file and all other files, and making sure we coordinate our changes better to prevent such issues from happening in the future.

Unknown Heap Corruption: When implementing the scroll effect of the map between rooms, we encountered a mysterious heap corruption issue that caused the program to crash randomly when the minimap is about to be filled on a level. The heap corruption caused memory leaks that also made the program run slower than usual. After multiple attempts at fixing it and asking for different sources of help (other groups, supervisors), the issue still remained. However, for an unknown reason, the issue suddenly resolved itself and the program ran smoothly and without crashing again. We tested many times (at least 15 times) after the issue supposedly “self-resolved” to make sure that it was really resolved.

Disparity in Coding Ability: Group members have different coding abilities, some of us are much more experienced with coding, which causes that person to be better at coding than others, while some of us are still learning and grasping C++ coding. This difference in ability causes an issue when it comes to task distribution, as we cannot give too difficult of tasks to the weaker coders and give all the tasks to the stronger coders. To solve this, we had to strike a balance, and give the relatively easier tasks to the weaker coders and learn from the stronger coders.

Inability to implement sound mixing: After extensive searching in the internet and asking around from other groups, we were unable to implement more than 1 sound playing simultaneously and settled on only playing background music. We used Josh’s music library (from group 4) to implement the background music. However, this process took some time as we faced difficulty in file directory manipulation.

What we have learnt: We learnt how to implement sound into a console game like this, using an external music library. We also learnt that any problem can be solved by going through a step-by-step process in debugging and looking through relevant variables and functions, if there were any issues that we are not familiar with, we can always Google to find out more. We also learnt a lot about how to use GitHub efficiently, using git.ignore and proper merging techniques to collaborate with group mates better.

Future Enhancements and Improvements

Addition of more than 1 sound playing at a given time: As of now, we can only play either music or sound effect at any given time, making use of Josh’s music library (from group 4) to read a file and play it. So, a future improvement that we could have done is to try to mix sounds so that more than 1 sound can be mixed and played simultaneously.

Addition of more items and item unlocks: Currently, there are 7 items in the game, and each item are upgradable to level 4 (3 upgrades per item). A future improvement that could make this game even more of a roguelike game is to implement more items and possible item unlocks in the shop, using score.

New Enemy AI to change pace of game: We have 2 AIs implemented, a melee AI and a ranged AI. We can make the game even more fun and dynamic to play by implementing even more different types of AI into the game, such as a long-ranged sniper type of AI and a teleporting AI. This would cause the player to be more weary of their movements as compared to now.

Different room sizes and shapes: Rooms are of the standard 11 x 19 size, with different preset room designs. To make the game stand out more with different gameplay, we could implement rooms that are half the size for example, to make the player think fast and act quickly, and maybe rooms that are diamond shaped to impair movement.