CYBER HERO

Group 5369 - No iDeas
Lin Jing Ya
Zeng JingJie, Jackie

CYBER HERO **1**

1. Project Files **4**
1.1 Poster 4
1.2 Video 4
1.3 Game Files 4

2. Project Overview **5**
2.1 Team Name 5
2.2 Proposed Level of Achievement 5
2.3 Motivation 5
2.4 Aim & Objectives 5
2.5 User Stories 5
2.6 Project Scope 6

3. Game Features **7**
3.1 User Interface 7
3.2 Player Movement 7
3.3 Variety of Maps 7
3.4 Health System 7
3.5 Enemies 8
3.6 Player Animation 8
3.7 Level Progress Status 8
3.8 Educational Feature 8
3.9 Player/Level Progress 9
3.10 Interactive Audio and Visual Cues 9

4. Implementation **9**
4.1 Tech Stack 9
4.1.1 Unity Engine 9
4.1.2 Unity Asset Store 9
4.1.3 C# and Microsoft Visual Studio 2022 10
4.1.4 Plastic SCM 10
4.2 Debugging 10
4.3 Software Engineering Practices 11
4.3.1 Version Control 11
4.3.2 Coding Practices 11
4.3.3 Game Design 11

5. Development Timeline **12**

6. Mock Up **13**

7. Challenges & Tradeoffs: **14**
7.1 Map Design 14

7.2 Potions 14
7.3 Enemies 14

8. Testing **15**
8.1 Unit Testing 15
8.2 User Testing 16

9. Project Log **17**

10. Technical Proof **21**

1. Project Files

1.1 Poster

1.2 Video

Our gameplay demonstration and technical proof video can be found here:
https://drive.google.com/file/d/1qhSBaXADkuLAe3NX6X6myfgFOOvCDxPY/view?usp=sharing

1.3 Game Files

The game can be downloaded at the following link:
https://drive.google.com/file/d/14bCKfJqPojRJxVUVpsURsLU2nTvYh5tw/view?usp=sharing

Simply extract the files from the compressed zip file, and double click on the Cyber Hero.exe file to start playing!

2. Project Overview

2.1 Team Name

No iDeas

2.2 Proposed Level of Achievement

Project Gemini

2.3 Motivation

Interested in game development, always wanted to try it out for ourselves. We felt that the disconnect between coding assignments in our modules and real-world coding applications is quite large, so we wanted to experience making code that works in a real-world application like a game. We are interested in learning new techniques and new ways to design code, as well as new algorithms and logic designs for our code, so that we can make full use of our summer break equipping ourselves with the skills for future semesters.

2.4 Aim & Objectives

To develop an immersive, story-driven game based in realism and real-world designs of PC parts as well as real-world functionalities of computer viruses.

- Create a fun and interesting game
- Raise awareness on computer vulnerabilities and its impacts
- Spread basic knowledge on computer security, viruses and malware

2.5 User Stories

1. As a gamer who often plays video games, I often feel that there are too many multiplayer games and sometimes I just want to enjoy a nice single-player experience

2. As a fan of retro-style games, I want to have a modern game that still evokes the look and feel of retro 8-bit style games

3. As an enjoyer of puzzle games and adventure games, I want to play a game that offers me both puzzles and adventures in a new and cool setting

4. As a computing student, there are very few games that are relatable to me, and I want to play a game that makes use of my knowledge in computing while still being able to have fun.

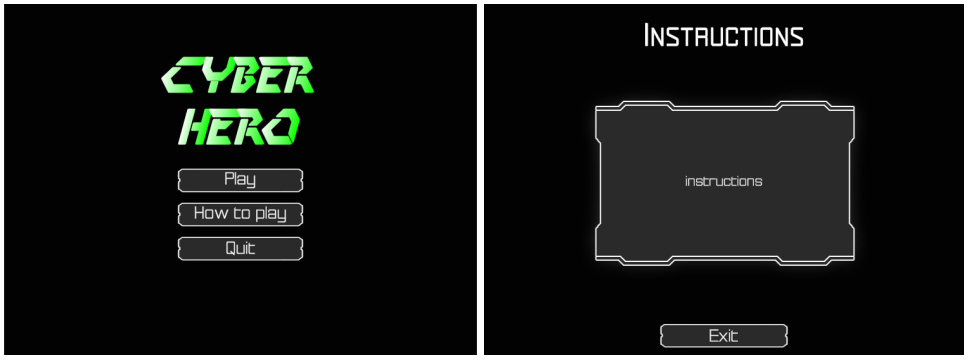
2.6 Project Scope

The project is focused on developing a 2D side scroller adventure game with the aim of defeating all enemies to eradicate a computer virus.

3. Game Features

3.1 User Interface

Shown below is the main menu page where the title of the game is displayed. Players can choose to either start playing, read the instructions or quit the game by clicking one of the three buttons. The instructions page is displayed upon clicking the ‘how to play’ button.

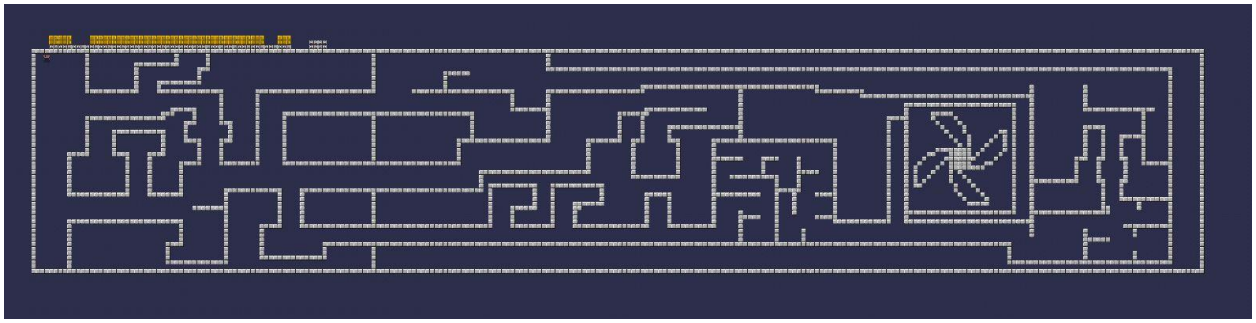


3.2 Player Movement

The player character is equipped with four-directional movement which is suitable for the level design, as opposed to a 2D platformer-style game. The movement in this game is very intuitive, and is a top-down style arcade game.

3.3 Variety of Maps

To enhance player enjoyment we came up with a few different maps all of which were designed to imitate that of the various PC parts such that our game is inline with the storyline of the game.



E.g. GPU level of the game, with added realism design features such as the cooling fan and the PCIe connectors

3.4 Health System

A health system is necessary in most games and for the game, player health is displayed in the form of a health bar which depletes as the player comes into contact with the enemies. There are also in-game objects such as potions which replenishes player health.

3.5 Enemies

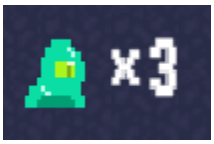
The game has three types of enemies. The first would be a simple slime enemy which has horizontal movement and changes its direction of movement on collision with the map walls. Colliding into it deals damage on the player. The second type would be the dragon. This type of enemy chases after the player character when the player is within a certain distance of it. Failure to run away from it also deals damage. Lastly, there is the eye and interacting with the sprite triggers a simple multiple choice quiz on computer security. Answering the question correctly destroys the enemy and damage is dealt for wrong answers.

3.6 Player Animation

For better realism, we have also animated our player and enemy characters. The player has 2 states of animation, namely the idle state and the running state. Animation has also been added to the enemies and information crystals for a better game experience.

3.7 Level Progress Status

In order for players to keep track of their progress in the specific level, we have implemented an enemy left count on the top right hand corner of the game. This is so that the player knows that all enemies in the level have been destroyed and they can move to the next.



3.8 Educational Feature

Players will be introduced to some basic knowledge on computer security and PCs when interacting with an in-game object in the form of a crystal. They would be tested on these information when they come into contact with the eye enemy where players would have to answer the question correctly to destroy the enemy, making learning an essential part of the game. The game also features level design that is loosely based on the design of computer components in the real world, to add a layer of realism.

3.9 Player/Level Progress

One of the key features that would be vital for the gameplay experience would be retaining level progress and player progress as they change levels. We used an in-built Unity tool, PlayerPrefs to achieve the desired effect, by saving the current health of the player and number of levels completed. Once a level is completed and all enemies are defeated, the level counter will increment and the status of that level will be set to completed. Once all levels are completed and the counter reaches the set number that is equal to the total number of playable levels, the game will be finished and trigger the “Game Complete” screen.

3.10 Interactive Audio and Visual Cues

One of the features that we implemented to improve the player experience was specific audio cues to alert the player of game events. Upon taking damage from the slime and dragon enemies, the player will hear a specific audio cue paired with a brief moment of camera shake to alert them of taking damage. When the player encounters a quiz enemy, an incorrect answer will play a specific audio cue, and the correct answer will also play a specific audio cue. Finally, when the player is damaged and collects a potion, there will also be a specific audio cue to alert the player that they have been healed. These audio and visual cues will help keep the player engaged and improve their experience by making it easier to keep track of what is happening.

4. Implementation

4.1 Tech Stack

- Unity Engine
- Unity Asset Store
- C# & Microsoft Visual Studio 2022
- Plastic SCM

4.1.1 Unity Engine

We are using Unity Engine as our primary game editing software, specifically editor version 2021.3.3f1. We chose this engine and editor as it has long term support (LTS) from Unity, and provides us with all the tools we need to implement our planned features in one convenient, user-friendly package.

4.1.2 Unity Asset Store

The Unity Asset Store is one of the features included with using the Unity Engine. Specifically, having access to a wide repository of ready-made sprites, environment blocks, and animations saved us a large amount of time that would have been otherwise spent designing these from

scratch. Our project makes use of several free-to-use, licence-free asset packages freely available from the Unity Asset Store, combined and enhanced to create a polished game interface.

4.1.3 C# and Microsoft Visual Studio 2022

C# is the primary scripting language used by the Unity Engine, and our game has required extensive use of scripts to map player movement, player behaviour, enemy behaviour, as well as gameplay elements like the 2-way portals, the health bar, potions, and also UI elements like the main menu, pause menu, and the “Game Over” screen. With the need for extensive scripting in C#, we would turn to Microsoft Visual Studio 2022 as our go-to IDE. Visual Studio 2022 can be modified with an install package specifically designed for C# scripting in Unity Engine, and within Unity’s project settings, Visual Studio can be selected as the default IDE to modify and create scripts. Double clicking on the script in the Assets window automatically launches Visual Studio, loading the script for us to edit. This level of integration between the two softwares was also one of the key reasons for choosing Microsoft Visual Studio 2022

4.1.4 Plastic SCM

Early on in the development process, we encountered issues with collaborative work, as the game files quickly became very large and thus impractical to be transferred back and forth between machines for separate editing. Our first avenue was to use Github, but again, as development progressed, the game files grew in size very rapidly, and very early on the size of the game rapidly outstripped Github’s upload limits, even using Github Desktop. Google Drive was also out of the question, and early searches yielded results that pointed to Unity’s own cloud-sharing system, which we quickly realised was already defunct and not usable.

Finally we discovered that Unity had moved to using Plastic SCM for collaborative workspaces, and that all we needed to do was to install the Plastic SCM plugin in our Unity editor. Once that was done, there was now a handy little window in our Unity editors, allowing us to check in our updates to a shared workspace and update our workspace with changesets made by other users with the click of a button. This meant we could both work on the project on separate machines, sometimes simultaneously, and even move the project to other machines with great ease. Plastic SCM also tracks the changesets implemented with each cycle of checking in and updating the workspace, down to deletion of a single file from a game object, which combined with its easy-to-use interface and integration with Unity Engine, allows for vastly superior version control and easy management of the workspace.

4.2 Debugging

Debugging was done with the use of Unity’s inbuilt function “Debug.Log()”. The function allowed the printing of messages containing certain game objects to find out information about the current state of the game object. We used it in areas such as finding out the player’s health as the game runs such that we could track damage taken and healing aspects of the game.

4.3 Software Engineering Practices

4.3.1 Version Control

We employed the use of PlasticSCM to enable version control. Upon implementing changes in our individual workspaces, the workspace would be considered “checked-out”, and once we were done with our changes we would use Unity’s in-built PlasticSCM window to check-in our changes, and the other user would then update their workspace to match the “checked-in” workspace in the PlasticSCM cloud.

4.3.2 Coding Practices

Names of functions that we wrote are able to clearly state the task of the function and standard naming practices were used. We avoided deep-nesting and included comments on areas to improve on.

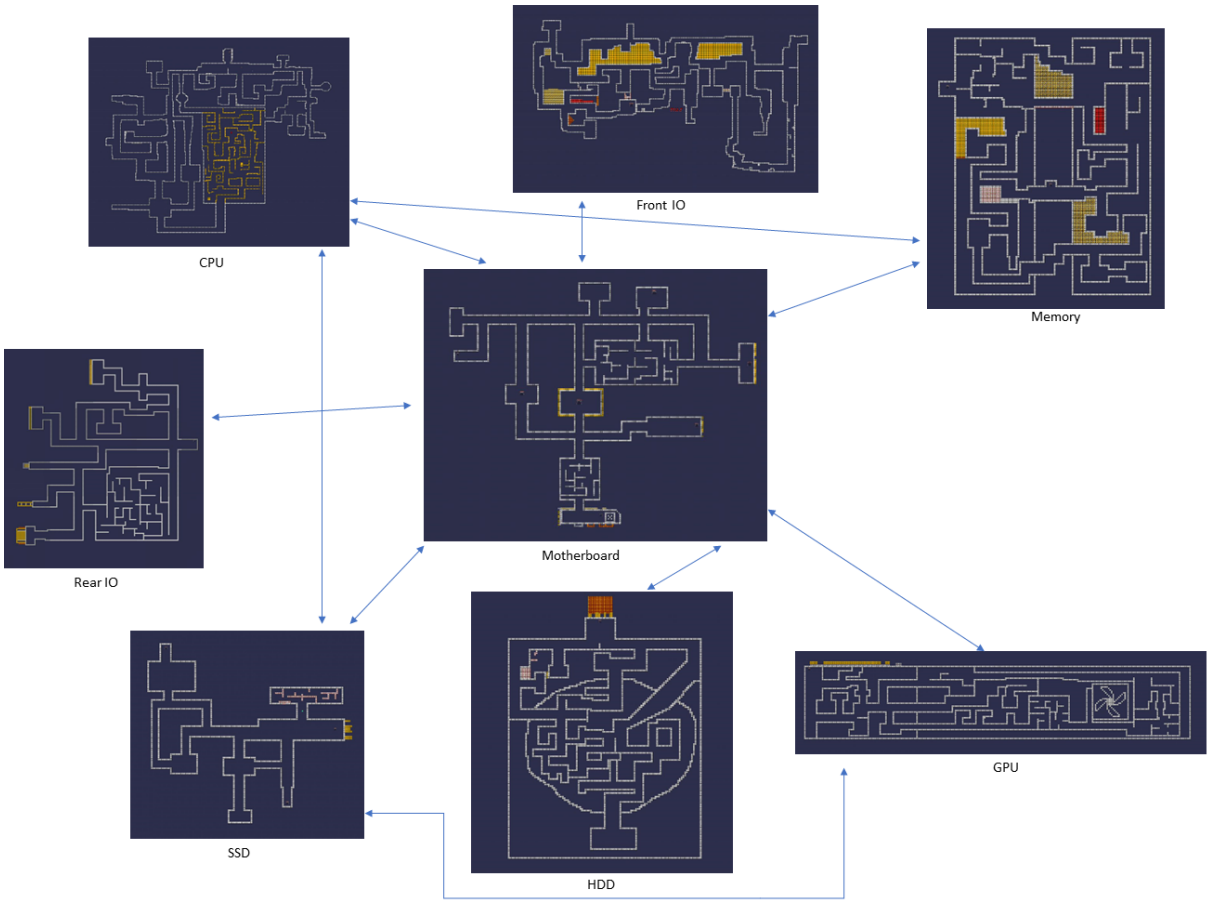
4.3.3 Game Design

Designing the game’s enemies necessitated the use of prefabs, a Unity feature, and a blank level upon which to test various features. Prefabs are a form of asset in Unity, and can be created simply by dragging an object, which could be anything from a GameObject to a sprite with code, into a folder in the game files. Following which, updating one particular instance of the prefab would give an option to “Override all”, which would push out those changes to all existing prefabs of the same type. Having a blank level with which to test these prefabs also proved indispensable, as the blank level allowed us a sandbox environment which mimicked the characteristics and interactions present in our other levels, so we could test features like the enemies and the portals between levels in a safe environment with no fear of causing catastrophic bugs. Once the object was verified to be working as intended in the blank level, we would simply update the prefab to override all other instances, pushing out the changes to all those similar instances. Having the object saved as a prefab also allowed us to simply drag and drop the object, with all its attributes, from the folder into the game scene.

5. Development Timeline

Date	Plan
2nd week of May	Finalise pitch and ideas for Orbital for Liftoff
3rd week of May	Brainstorming ideas and initial design concepts
4th week of May	1st Unity workshop and initial test build
5th week of May	2nd Unity workshop, design of map and levels in the game, initial test of player movement and interactions
1st week of June	Creating new levels + animating character
2nd week of June	Implementation of enemies' movement and skills
3rd week of June	Implementation of health system
4th week of June	Implementation of in-game items
5th week of June	Implementations of animations and effects
1st week of July	Improving enemy movement and refining levels
2nd week of July	Pushout prototype and implement PlayerPrefs
3rd week of July	Testing and debugging
4th week of July onwards	Collecting feedback, fixing bugs and final refinements

6. Mock Up



Legend: bidirectional blue arrows represent ability of the player to travel between levels in both directions

The level design is intended to mimic the look of PC components in the real world, with added complexities and designs to enhance the player's experience. Each level has portals which give the player the ability to travel back and forth between levels as they play, they are also designed to mimic the real-world flow of data between components in the PC.

7. Challenges & Tradeoffs:

7.1 Map Design

Map design was implemented using tilemaps and the tile palette tool to “paint” in each block of the map. This method was chosen as it allowed us to have manual control over the style of the map, as well as access to the 2D Tilemap Collider function, allowing the wall to have collision physics with our player and the enemies. Using tilemaps to design the map also allows us to update and change the map very rapidly by simply deleting the blocks we want to remove and adding new paths by drawing new blocks in. The trade-off of this feature was that it was very tedious having to draw in each block one by one, causing map design to take up a prolonged period of time vastly beyond what was initially projected. The other trade-off is that this causes our game files to become very bulky, and is not as memory-efficient as other methods, due to the game engine needing to render every single block independently upon loading a level. Whether or not this is an issue faced by other methods of map design is currently unclear, but we have observed longer loading times on less powerful computers as more levels are designed and added.

7.2 Potions

The previous implementation of the potion was for the potion to be destroyed upon contact with a damaged player, healing the player. However this solution was inefficient as the engine would need to render and create a brand new potion every time the level is reset. The solution was to update the script such that the potion is not destroyed, but rather set to inactive upon contact, which would conserve resources and increase the efficiency of the game.

7.3 Enemies

The enemies in our game previously also faced the same limitations as mentioned above for the potions, being destroyed and then re-rendered each time the level is reloaded. This again was very inefficient and caused a lot of unnecessary load on the system, so we implemented the same change as we did for the potions.

8. Testing

8.1 Unit Testing

S/N	Unit		Test Input	Expected Outcome	Remarks
1	Player Movement		Arrows/ WASD keys	Player moves in direction of individual arrow keys	Works as expected
2	Enemies movement	Simple	Visual confirmation	Moves in a straight line and changes direction upon collision with wall	Works as expected
		Homing	Move player near and away from enemies for visual confirmation	Chases after player when the player is within a certain distance from the enemy, return to original position when distance apart is larger than same specific distance	
		Flee		Runs in opposite direction of player when the player is within a certain distance from the enemy, return to original position when distance apart is larger than same specific distance	
3	Collision Detection		Intentionally move player to collide with these objects	For enemies: damage dealt and health increases For quiz enemies: quiz menu pops up For crystals: information menu pops up For potions: health increases	Works as expected
4	Player Health	Enemy	Intentionally allow enemy to collide with player	Damage dealt and health bar decreases by one bar per collision/ wrong answer	Works as expected
		Quiz Enemy	Intentionally answer incorrectly		
		Potion	Intentionally consume potion both when health	When full: no changes to health bar and potion is not consumes When half-full: health bar	

			bar is full and half-full	increases by one bar and potion object is destroyed	
5	Player Death		Intentionally allow damga to be taken until health bar is empty	Gameover menu pops up	Works as expected
6	Quiz Enemy Death		Answer quiz correctly	Enemy object is destroyed	Works as expected
7	Teleportation Doors		Make player go through doors	Should bring player to other levels	Works as expected
8	UI Elements		Manually test all buttons and visual confirmation	- All buttons should work as intended - Health bars should correspond to health amount and change with damage or healing - UI navigation using mouse should work - Minimap should follow player around	Works as expected

8.2 User Testing

User testing was done by distributing our game to playtesters. They were then asked to fill up a survey with regards to feedback on the various features, bugs encountered and overall game experience.

	Feedback	Follow-up
1	Fleeing enemy crashes into wall and stops	Used a different way of calculating its destination coordinates to improve movement
2	Loss of player control before player death	Edited script to fix problem
3	Player is unable to move if homing enemy pins it to wall	Decreased mass of homing enemy sprite such that it is unable to push player
4	Text not staying in boxes for UI aspects	Manually edited UI for text to fit, checked all relevant objects for similar problems

9. Project Log

S/N	Task	Date	Hours Spent		Remarks
			Jackie	Jing Ya	
Liftoff (09/05/2022 -16/05/2022)					
1	Proposal Revision	09/05/2022	1	1	Revised proposal after feedback from meeting with advisor
2	Liftoff Submission	10/05/2022	2	1	Worked on project poster and video pitch
Milestone 1 (17/05/2022 - 30/05/2022)					
3	Unity	17/05/2022 - 20/05/2022	2	2	Downloaded required software and initial experimentation with tutorial videos
4	Unity Workshop Part 1	21/05/2022	2	2	Mission Control Workshop
5	Unity	21/05/2022 - 26/05/2022	6	6	1) Self-learning and experimentation with the different features in unity 2) Completed simple game from workshop
6	Physical Team Meeting	27/05/2022	3	3	1) Discussed game details and initial map designs 2) Choosing suitable assets 3) Player Movement + scripting
7	Unity Workshop Part 2	28/05/2022	2	2	Mission Control Workshop
8	Programming at Home	28/05/2022 - 29/05/2022	4	0	First level map design and tile palette
			0	4	User interface + scripting
9	Online Team Meeting	29/05/2022	2	2	Worked on milestone 1 deliverables - ReadMe and

					Project Log
10	Milestone 1 Submission	29/05/2022	1	1	Worked on video and poster
11	Plastic SCM	29/05/2022	1	1	Figured out how to use Plastic SCM and uploaded our project for ease of collaboration
Milestone 2 (31/05/2022 - 27/06/2022)					
12	Programming + Self Learning at Home	31/05/2022 - 06/06/2022	4	4	Refining first level and watching tutorial videos on features that we want to implement
13	Online Team Meeting	07/06/2022	1	1	Discussion of game details and splitting of workload
14	Programming at Home	08/06/2022 - 24/06/2022	25	0	Map design for 8 new levels
		09/06/2022	2	0	Scripting and debugging 2-way portal in between levels
		23/06/2022	5	0	Player animation - idle and movement
		08/06/2022 - 09/06/2022	0	5	Pause, “Game Over” menu and level progress bar
		10/06/2022 - 13/06/2022	0	10	Player health system - able to take damage and heal with displayed health bar + scripting
		14/06/2022 - 20.06.2022	0	12	Variation of enemies with different movements and behaviors + scripting
		21/06/2022 - 23/06/2022	0	7	Quiz and learning feature on player interaction with in-game objects + scripting
15	Physical Team Meeting	24/06/2022	3	3	1) Debugging and refining game 2) Discussion on game direction

					3) Working on milestone 2 deliverables
16	Online Team Meeting	26/06/2022	2	2	Working on milestone 2 deliverables (ReadMe)
17	Milestone 2 Submission	27/06/2022	5	4	1) Creating poster and video content 2) Detailing project log
18	Online Team Meeting	27/06/2022	2	2	Working on milestone 2 deliverables + submission
Milestone 3 (28/06/2022 - 25/07/2022)					
	Programming at Home	28/06/2022 - 02/07/2022	4	4	Populating all levels with sufficient enemies
	Online Team Meeting	03/07/2022	2	2	Discussion of game details and splitting of workload
	Self Learning at Home	04/07/2022 - 11/07/2022	8	8	Learning ways to implement new features online
	Programming at Home	04/07/2022 - 11/07/2022	3	0	Background Audio
			5	0	Interactive Audio - eg. sound cues for the quiz enemies and contact enemies
			4	4	Debugging
			5	5	Animations
			0	7	Minimap and instructions manual
	Online Team Meeting	11/07/2022	2	2	Discussion of game details and progress checking
	Programming at Home	12/07/2022 - 14/07/2022	5	5	Creating quiz questions and fun facts
			1	1	Filling in questions for all levels
			6	6	Debugging
	Online Team Meeting	15/07/2022	2	2	Discussion of game details

					and exploring export options
	Prototype 1	16/07/2022	1	1	Create feedback form and export game
	Programming at Home	16/07/2022 - 21/07/2022	5	0	Camera Shake
			10	0	PlayerPrefs - scripting tool to allow for player progress like health, spawn, and level progress to be retained if the player changes levels
			0	7	Fixing and improving user interface
			0	6	Improving enemy movement
	Milestone 3 Submission	21/07/2022	0	3	Working on readme and project log
	Online Team Meeting	22/07/2022	1	1	Discussion of game details and progress checking
	Programming at Home	23/07/2022	1	1	Debugging
	Online Team Meeting	24/07/2022	3	3	Working on milestone 3 deliverables and debugging
	Milestone 3 Submission	25/07/2022	2	2	Working on milestone 3 deliverables
	Total Hours		145	145	-

10. Technical Proof

Fig. 1: Screenshot of the main menu

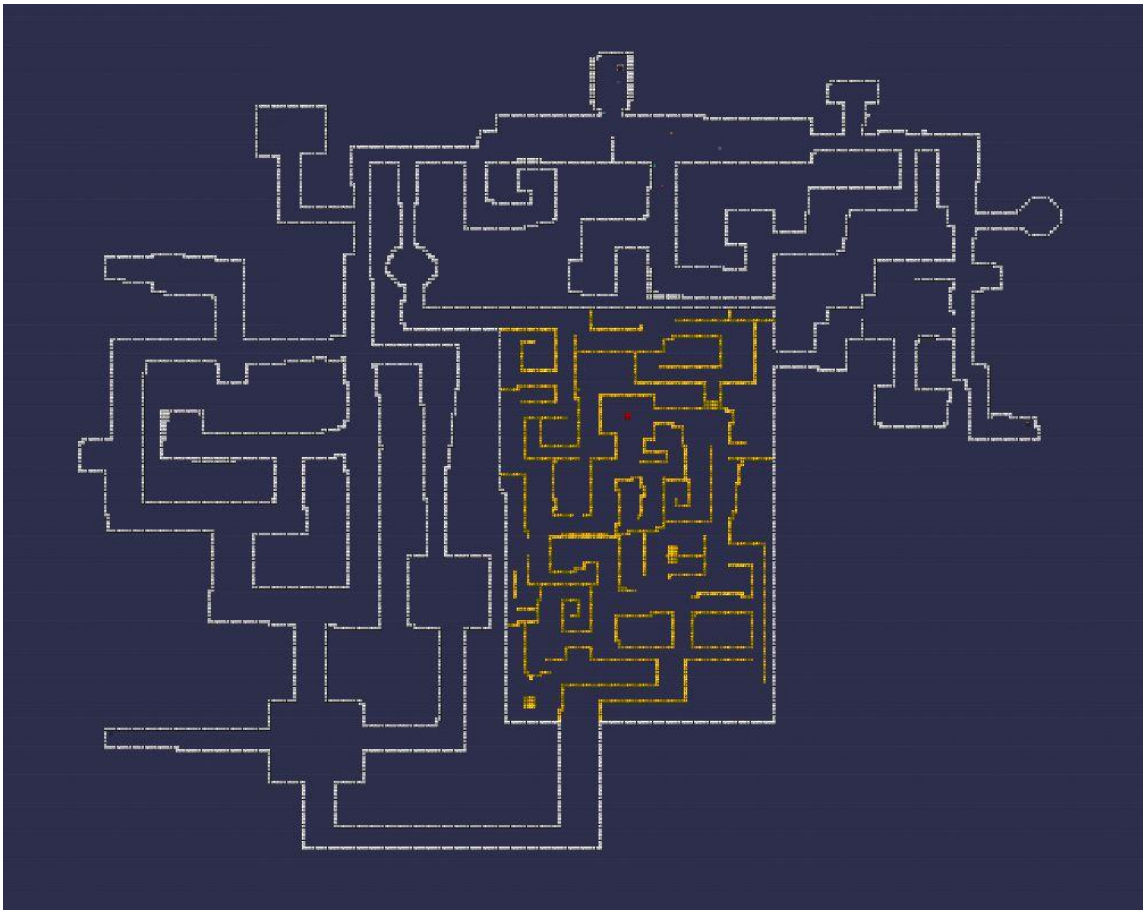


Fig. 2.1: Screenshot of the CPU map level

Fig. 2.2: Screenshot of the Front IO map level

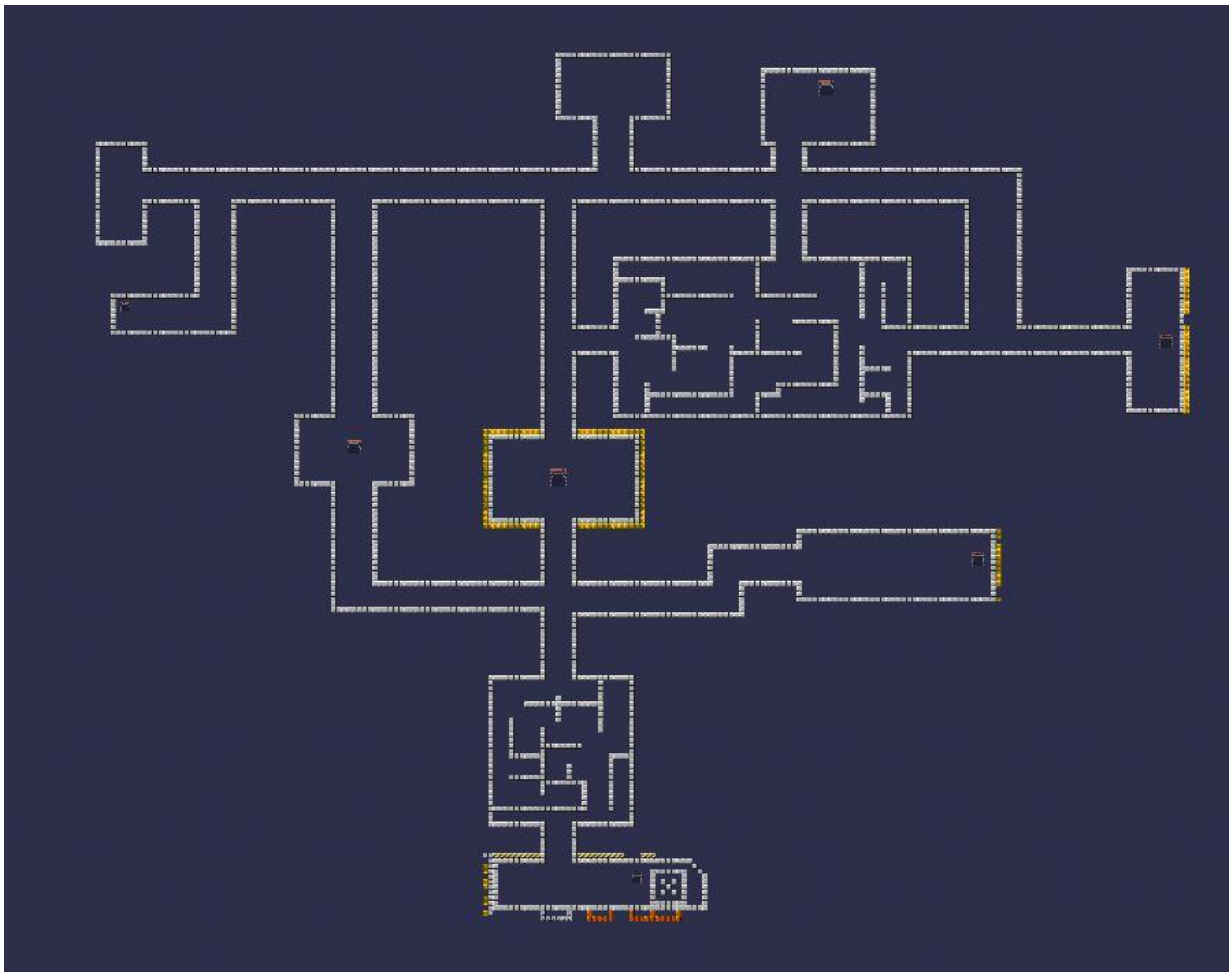


Fig 2.3: Screenshot of the Motherboard map level

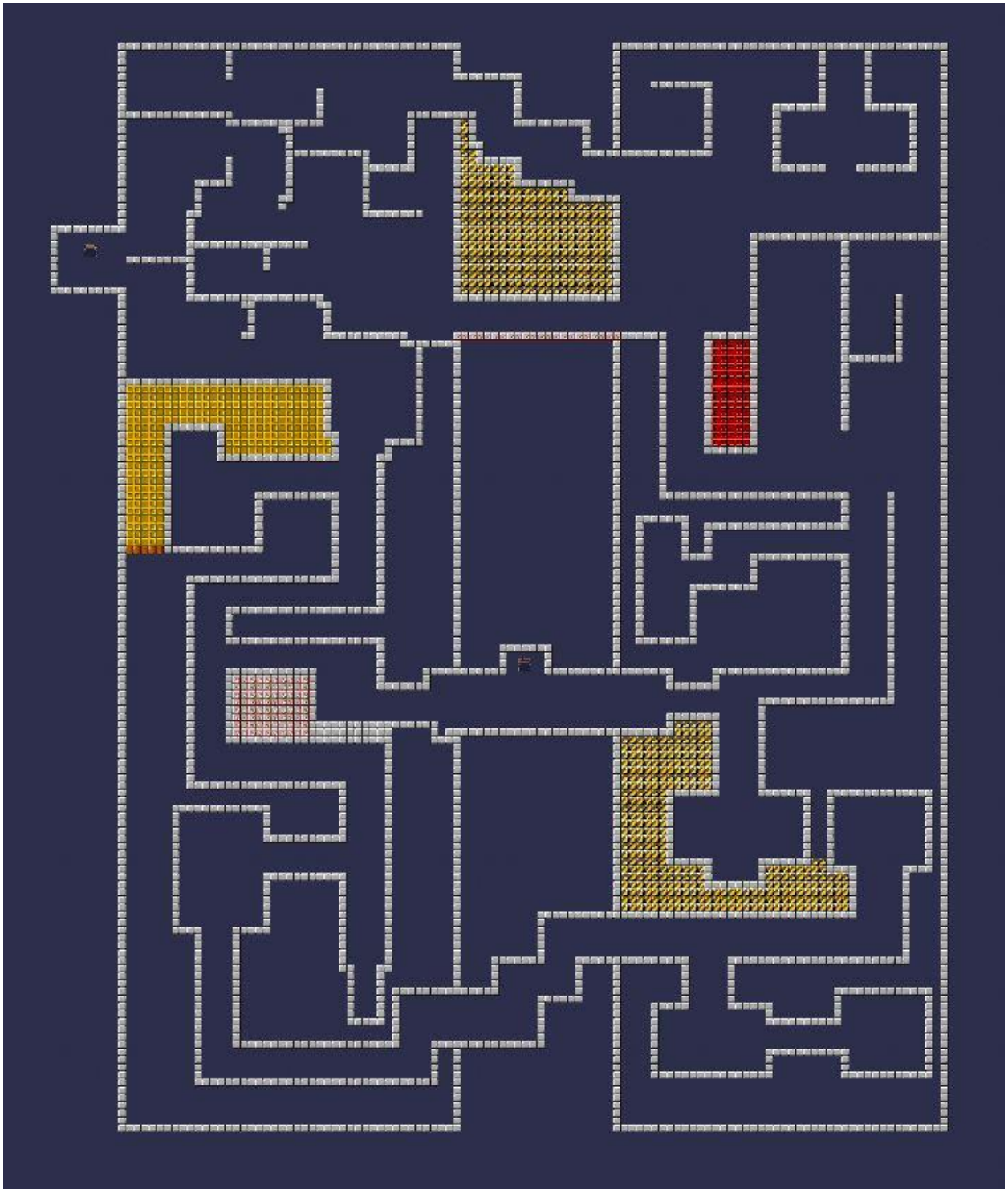


Fig 2.4: Screenshot of the Memory map level

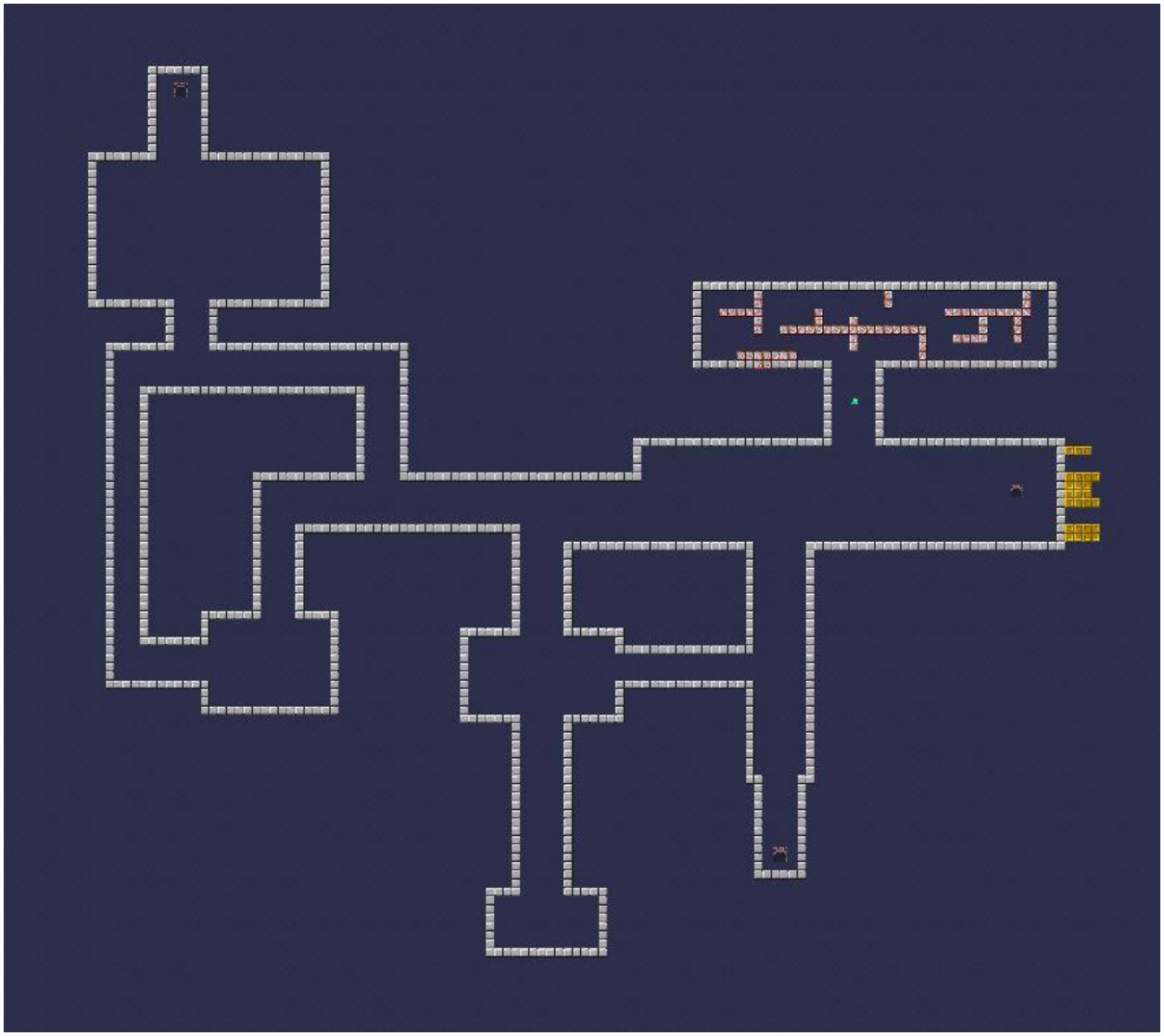


Fig 2.5: Screenshot of the SSD map level

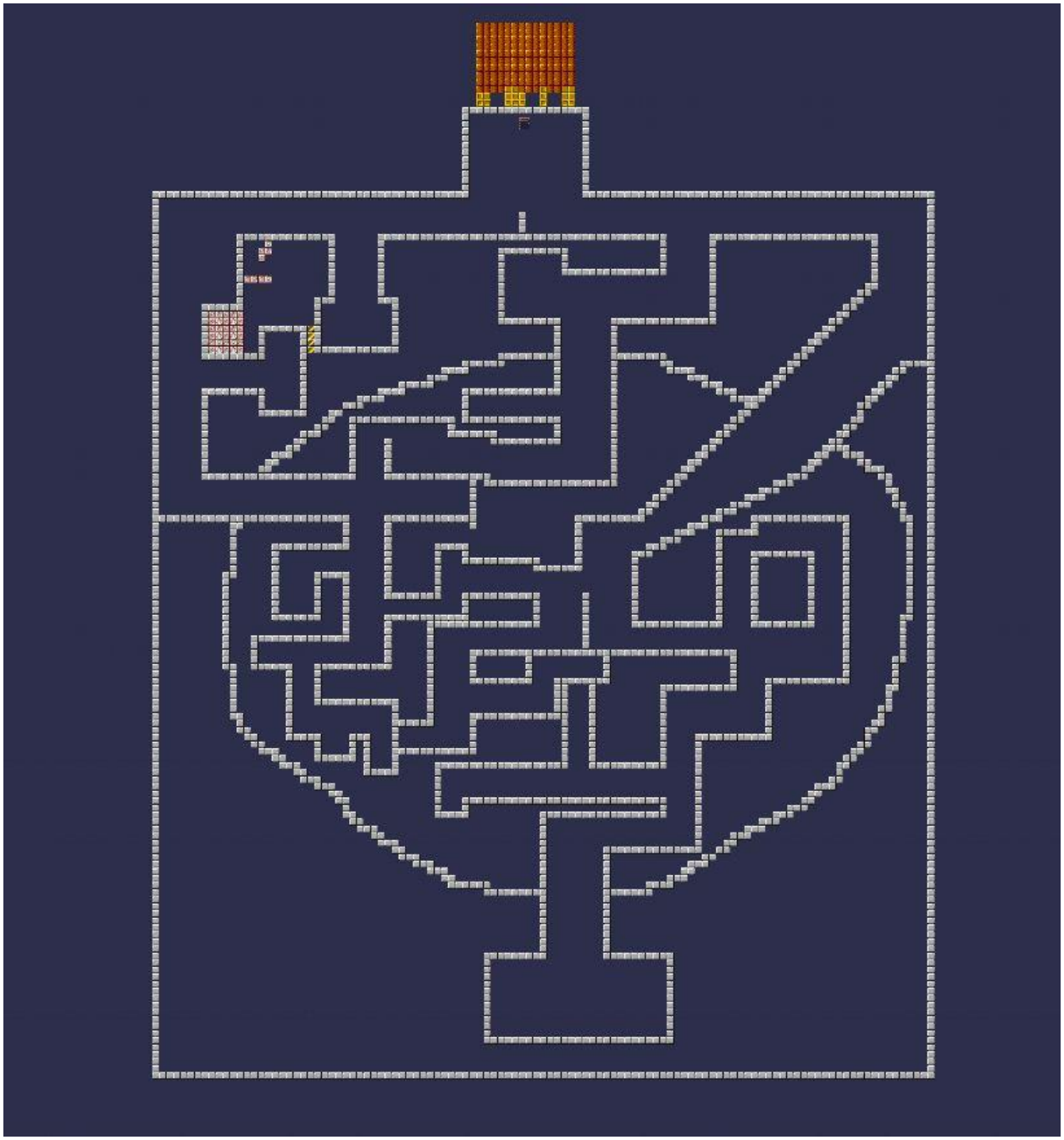


Fig 2.6: Screenshot of the HDD map level

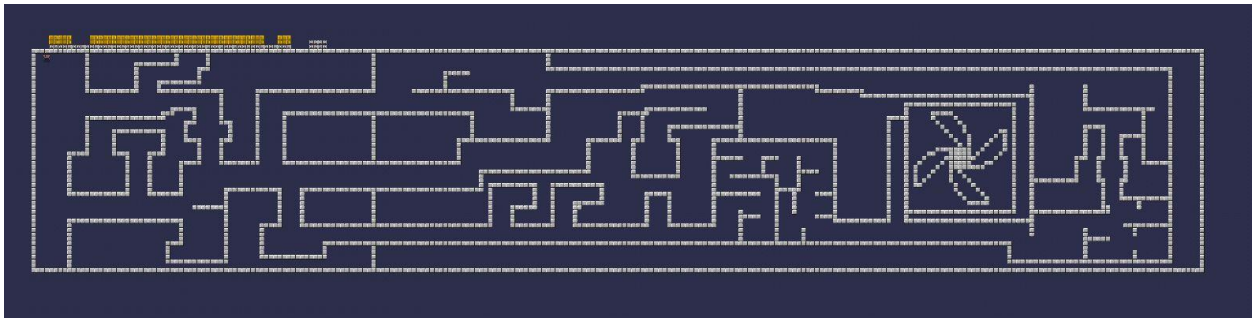


Fig 2.7: Screenshot of the GPU map level

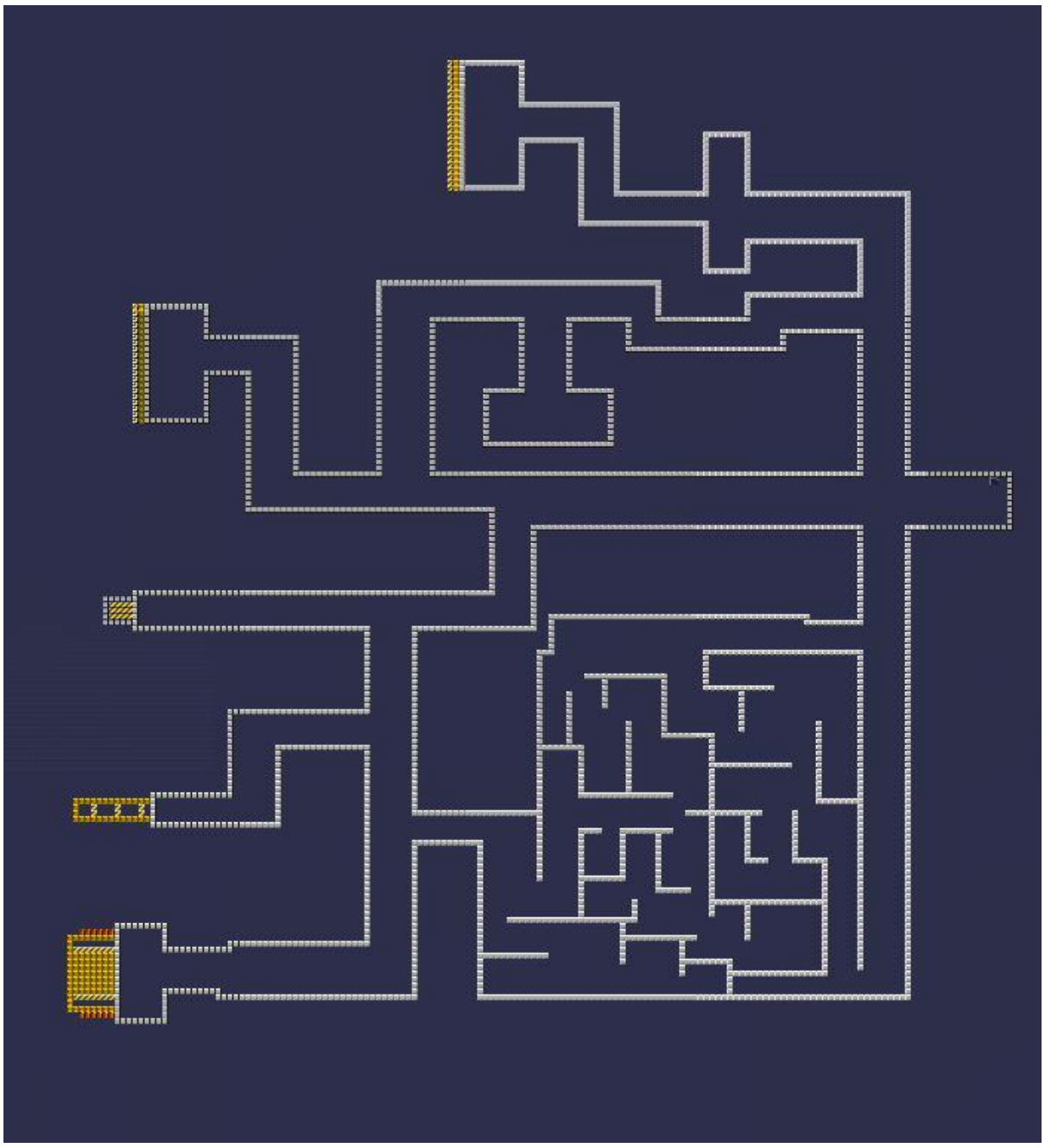


Fig 2.8: Screenshot of the Rear IO map level

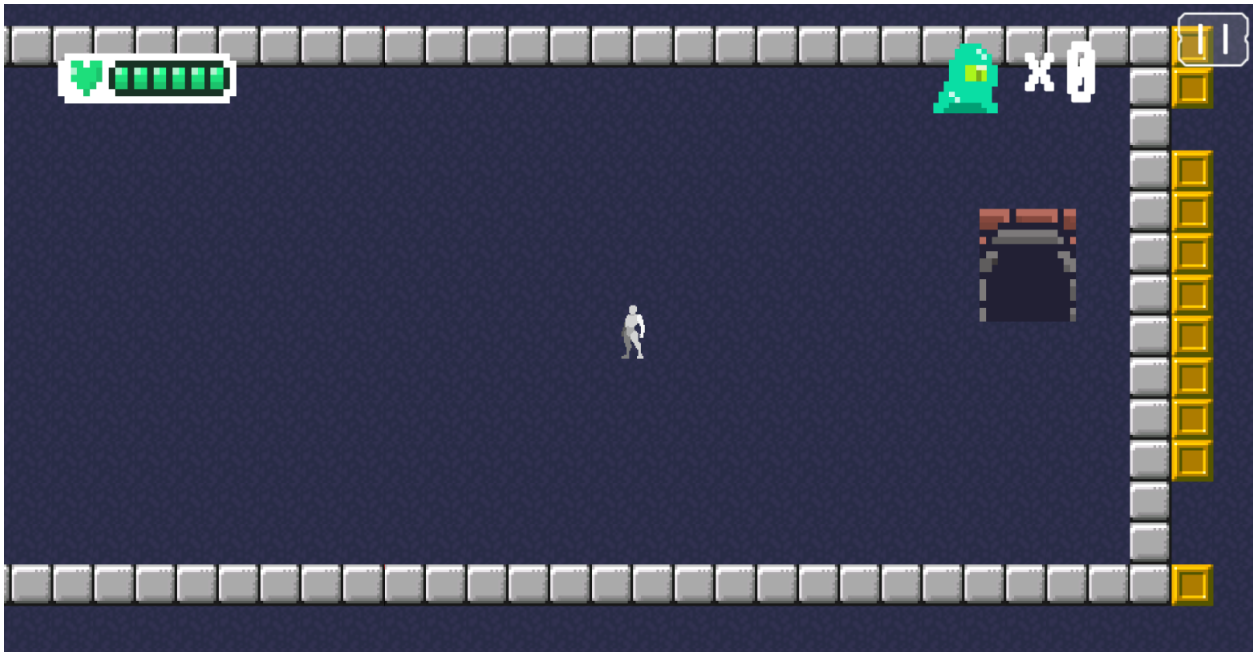


Fig 3.1: Sample screenshot of level with UI including health bar, enemy count and pause button

Fig 3.2: Sample screenshot of level with Pause menu UI overlay

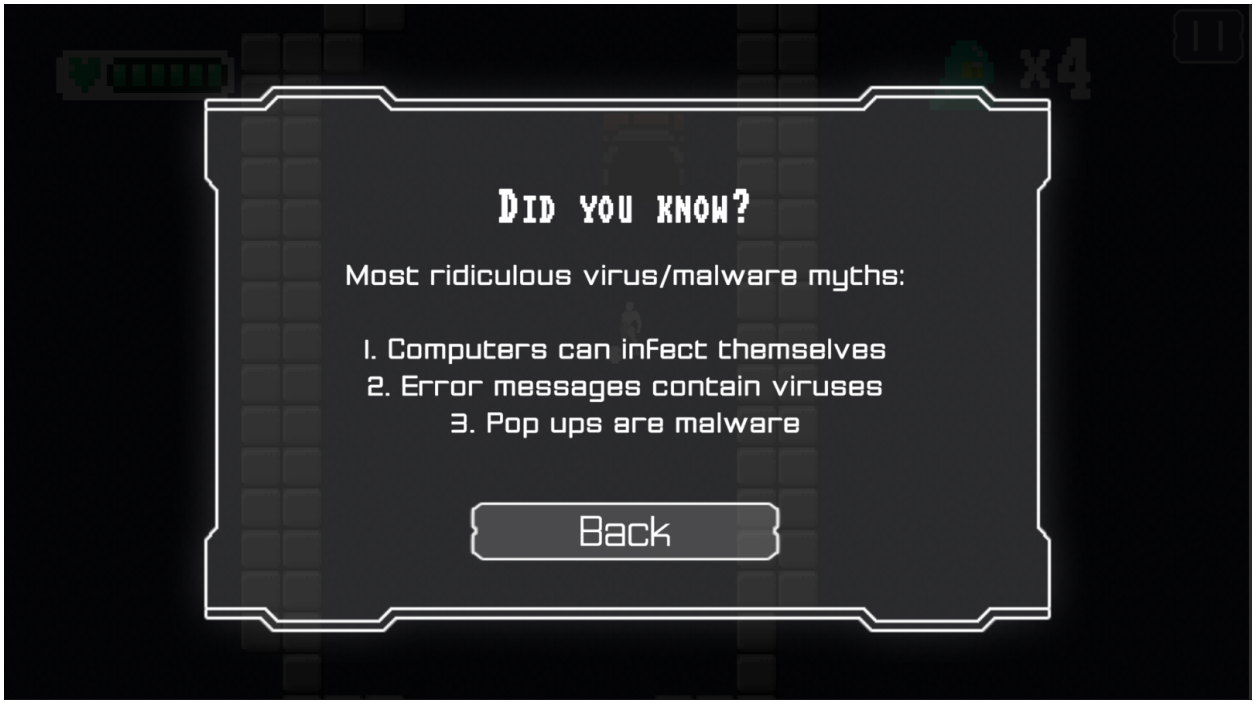


Fig 3.3: Sample screenshot of fun fact box triggered by the player colliding with a crystal

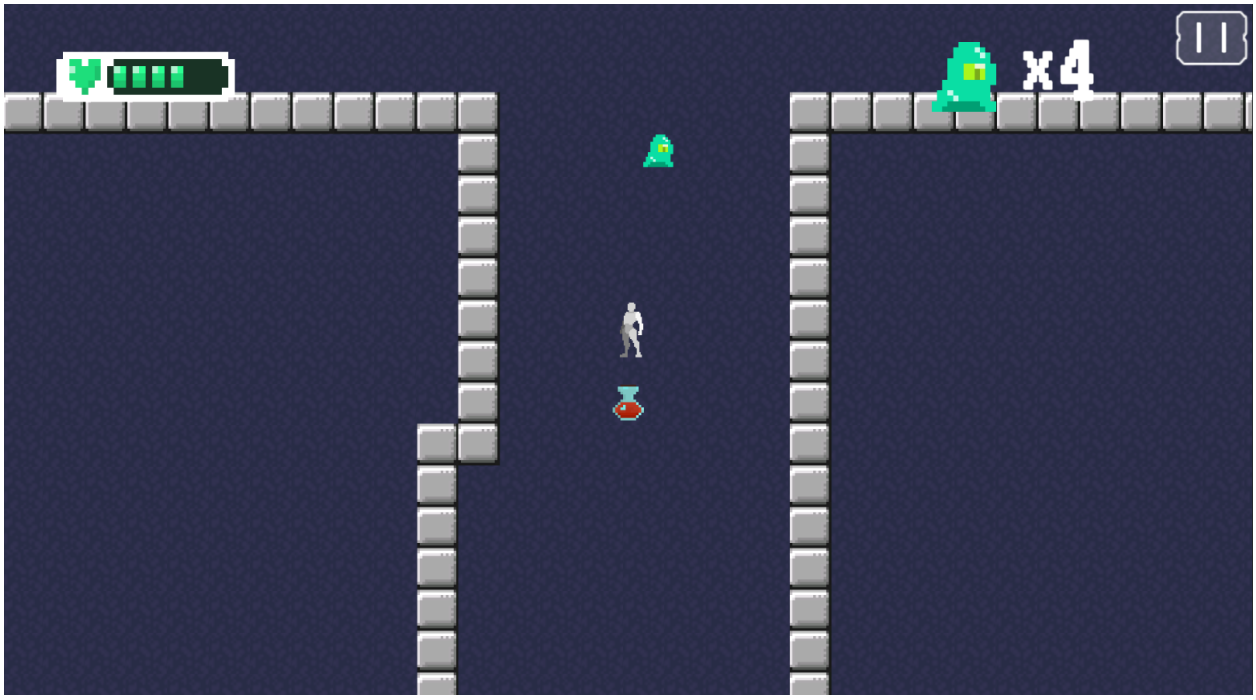


Fig 3.4.1: Sample screenshot of player with reduced health after contact with enemy

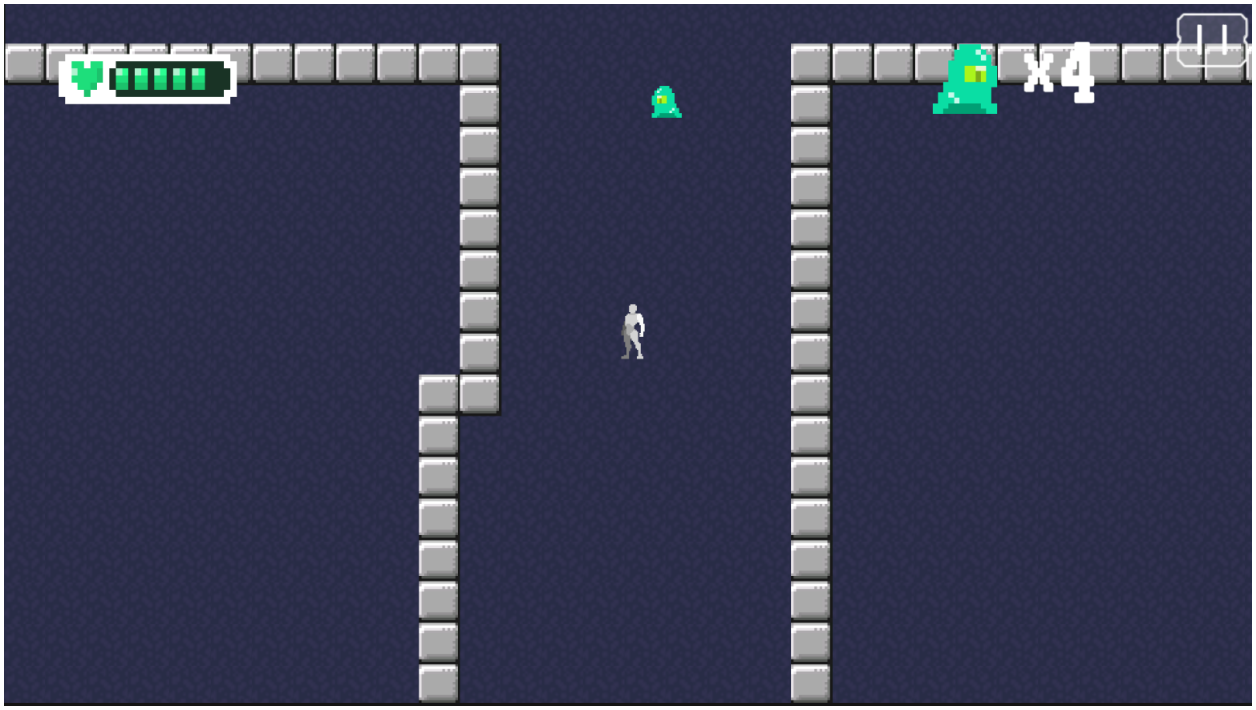


Fig 3.4.2: Sample screenshot of player with health regained after collecting a potion, the potion is also no longer visible in the scene after the player has collected it

Fig 3.5: Screenshot of "Game Over" overlay, triggered when player health reaches 0

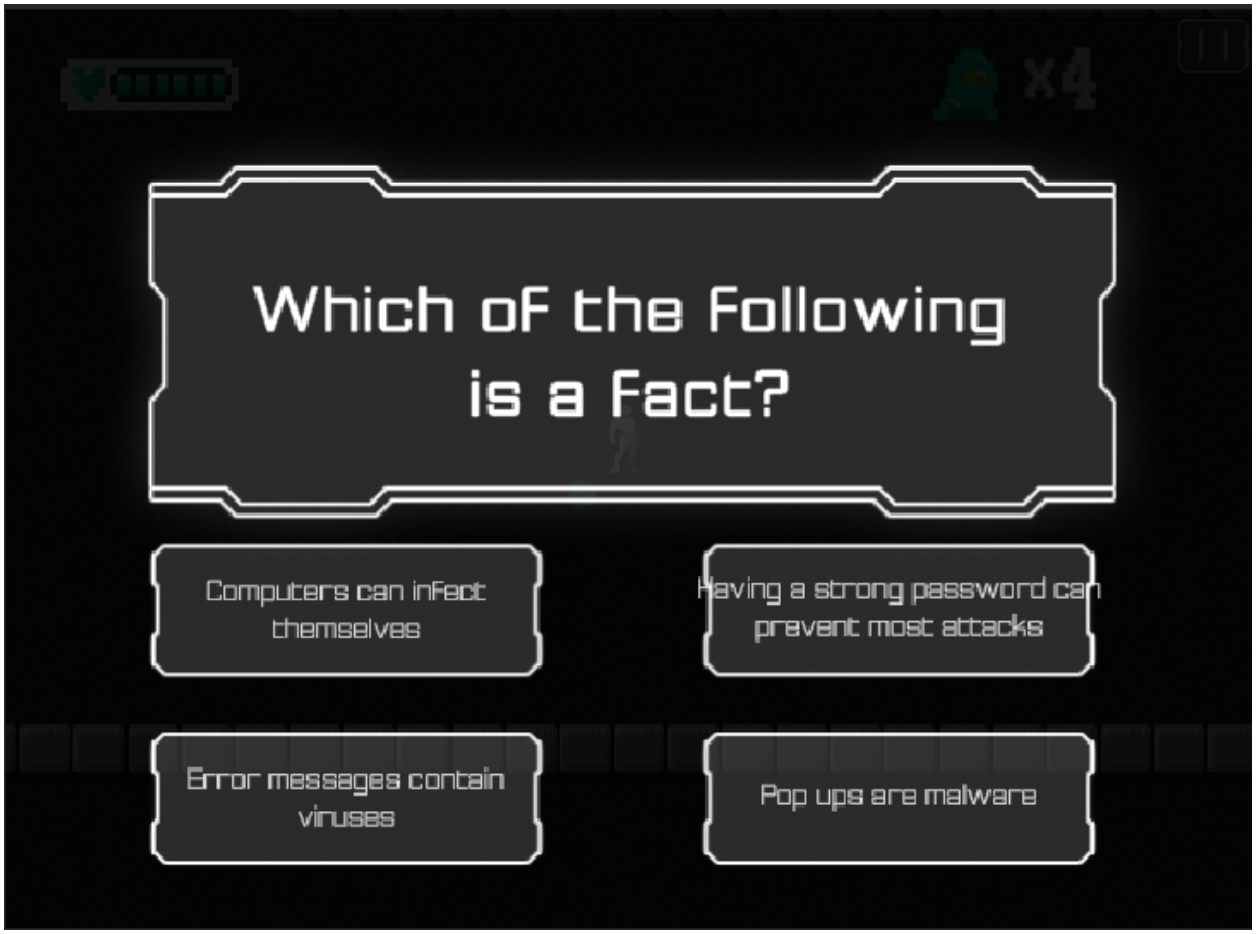


Fig 3.6: Sample screenshot of overlay brought up when player comes into contact with quiz enemy

Fig 3.7: Screenshot of portal door, which allows players to jump between levels. The portal has 2-way travel, allowing the player to travel back and forth between levels, always spawning below the door upon level change

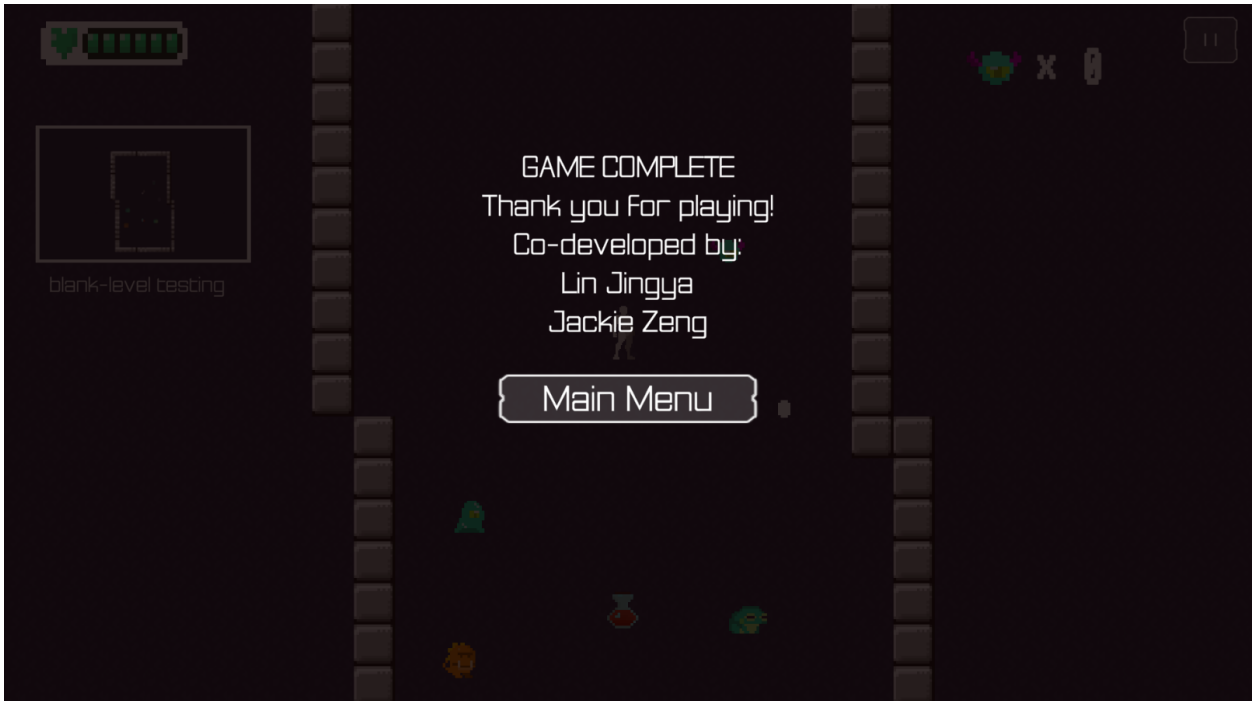


Fig. 3.8: Screenshot of the Game Complete Screen, which will appear when all levels are completed