ÉCOLE POUR L'INFORMATIQUE ET LES TECHNIQUES AVANCÉES

UNDERGRADUATE 1^{st} Year SEM. 2

Project Q
FAUST

POLLO Y PAPA

DAVID CALDERÓN, MAXIME BARDOUIL, RAJ MAHAJAN,
WADHAH OULED AMEUR

IAN TERNIER

APRIL-2022

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

Faust, is a rogue-like game influenced by Dead Cells and Hades, which presents the story of a man driven with the obsession of gaining omnipotence. Faust is a 2-dimensional game created by Pollo-Y-Papas, with a pixel art style and progressively increasing enemy difficulty. While rogue-like games are single players, Faust will have a multiplayer option for the players to choose.

In this report we will explore the 2nd stage of the creation process of the game, this includes the difficulties we experienced, what we learnt, as well as what needs to be done for the next defense.

Figure I: *Faust logo*

Chapter 2

MODIFICATION TO THE BOOK OF SPECIFICATIONS

During the process of making the game, some assigned roles were changed. The following table shows what each person worked on.

This section states the distribution of the major elements that will be worked on for the Game. The word “Head” implies that the person is in charge of the task while, “Vice-Head” is someone that will be helping the person complete it.

	David Calderon	**Maxime Bardouil**	**Raj Mahajan**	**Wadhah Ouled-Ameur**
Project Manager			X	
Character, Stage Design (NPC, Bosses)	Head		Vice-Head	
Physics of the Game	Head		Vice-Head	
Map Generation AI		Vice-Head		Head
Animation	Vice-Head		Head	
Network		Head		Vice-Head

Chapter 3

CURRENT PROGRESS

Presentation	Tasks
25 - 29 April, 2022	1. User Interface a. Main Menu b. In-Game UI 2. Basic Map Generation a. Function Program but not necessarily optimized 3. Character Animations a. Movement and action animations of characters 4. Character Design a. All character, NPCs and weapons are designed 5. Intelligent AI a. Enemy Pathing and Attacking 6. Network a. Finish All Research b. Start implementation 7. Operational Website a. Website Interface b. Summary of the Game c. Track of Progress

3.1 User Interface : Maxime François-Bardouil

As mentioned in the report for the last defense, a User Interface or UI has been created. This includes a Start Menu, a Death Screen and In-game Overlay.

As seen in Figure II, the Start Menu includes options such as Start Game, Quitting it and lastly Options which will later be used to customize audio and other in-game options. Secondly, there is a Death Screen (Figure III) that allows the user to either return back to the Main Menu or to start another instance of the game.

Figure II: *Start Menu*

Figure III: *Death Screen*

Lastly, there is the in-game overlay, which consists of the Stage the player is in, the Return Menu as well as the Health for the Player and the Enemies. Finally, in-game there are newly created buttons, that allow the player to host a room for other players to join and play in.

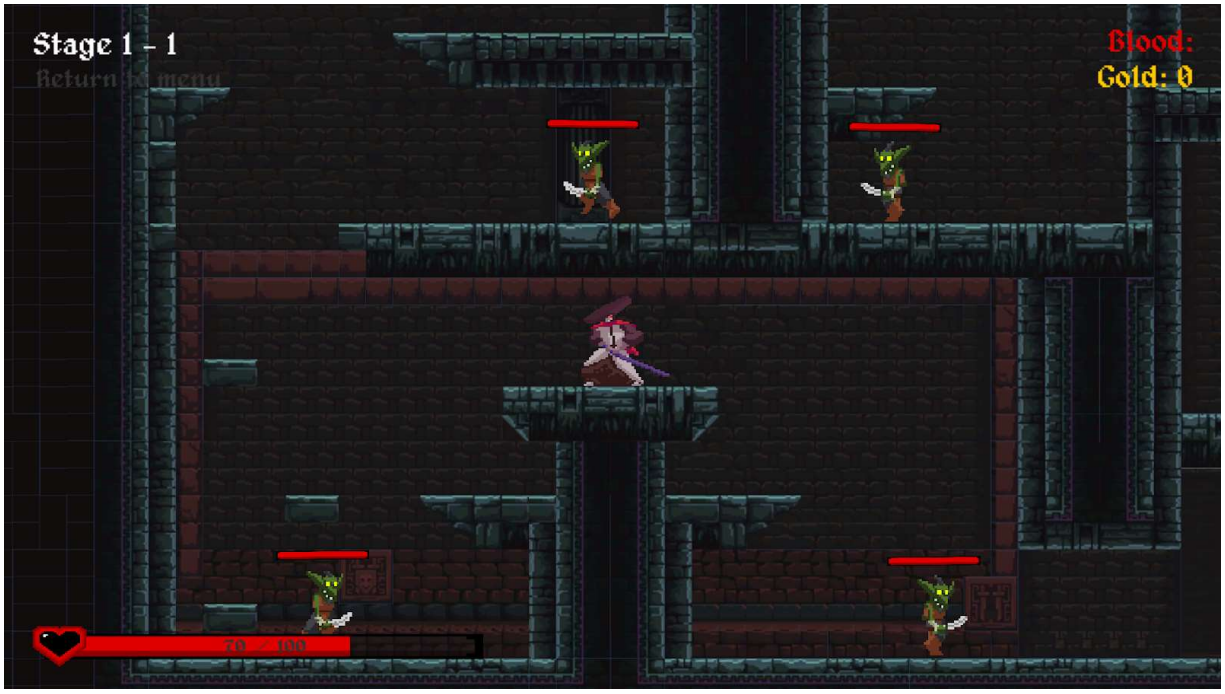

Figure IV: *In-Game Overlay*

3.2 Map Generation and Creation : Wadhah Ouled Ameur

As a reminder, In Faust, the player must complete 4 Levels each consisting of several randomly ordered rooms to complete the game. Because of this, the Mapmaking process went through many changes.

First, we Had to find a collection of tilesets that all share a similar art style so that the player's experience remained seamless and convincing. and for this, we decided to go with Szadi Art's pixel fantasy set on the unity asset store.

After deciding on a tileset it was time to decide on the actual structure of the map itself. In the beginning, each level was going to be made of about Seven or Eight small rooms, but that idea was scrapped since the player would find themselves to be switching rooms very often which could get annoying.

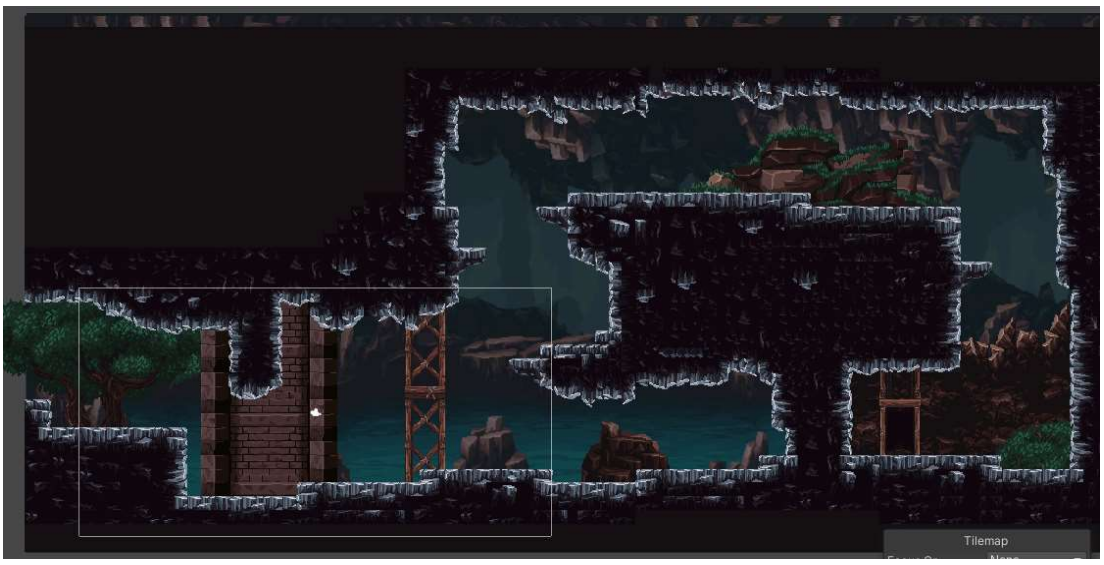

Figure V: *Small Rooms Design*

So instead Each level is now made of Four Medium-sized rooms and a boss room at the end of the level.

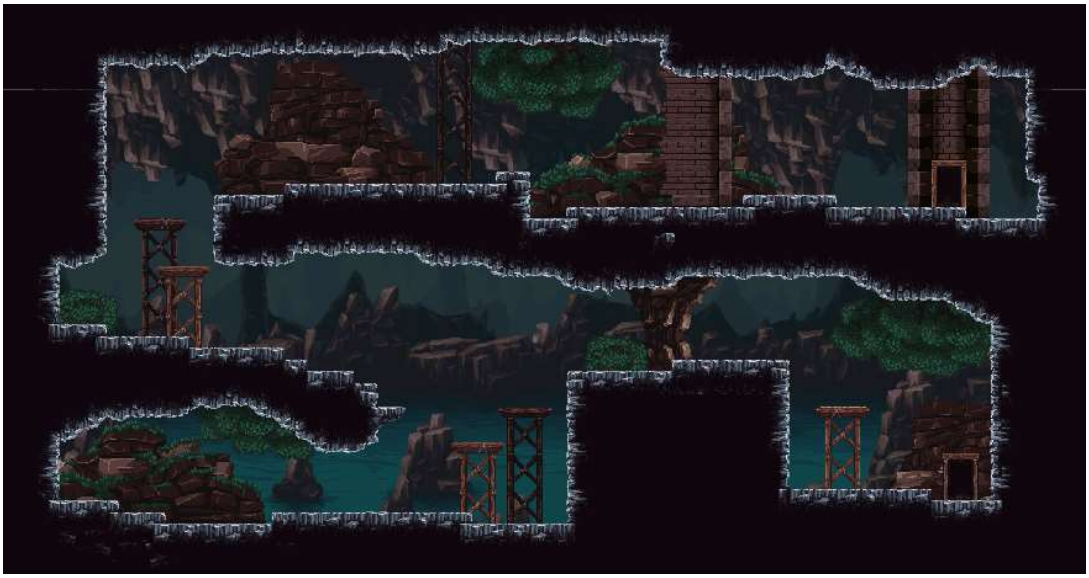

Figure VI: *Example of a medium room*

Currently, the Four levels are going to be:

Level 1: The Cave

Level 2: The Deep forest

Level 3: The Snowy mountain

Level 4: Mephistopheles Castle

For the Time Being The only Level that is 100% complete is The Cave. Where every complete Level Would Consists of Eight Total rooms so that four of them can be selected at random at each run. it could also be important to note that all the levels were created in the same way as the test map that was presented in the first presentation.

After creating all the rooms for the Cave Level Came the Time To add Different features to the level so that the players could interact with them. one of these features is one-way platforms.

One way platforms allow the player to pass through them from one direction but stand on them from the other. To create these we used the "Platform Effector 2D" So that when the player presses "Space" or the jump button They would be able to pass through the bottom of the platform but land on the top and when they press the "S" The player will fall through the platform.

Figure VII: *Platform Mechanism*

Afterwards Came the Implementation of the Doors so that you can pass from one room to the other. Since Each room is made on its respective Scene When the game Starts we create a random list made of a random combination of 4 rooms and a boss room. Afterwards, we made it to where if the player Collides with the Door, the player would be transported to the next Room in the list.

3.3 Character Animation & Design : Raj Mahajan & David Calderon

We created a few more enemies as well as NPCs that we will be using for the game. These sections will explore the characteristics of these characters and how they were made.

3.3.1 Creation of NPC

An NPC is a Non-playable Character, meaning these are characters that cannot be played but also have the connotation of being neutral towards the player. This means that enemies cannot be classified as NPCs.

Firstly, an NPC was created to do this, an asset of a flying eye was found, once that was done with the addition of an indicator and frame an NPC was created. A NPC works with 2 rules. Once the player gets in a certain range of the NPC, a small purple indicator will be visible to the player as in Figure VIII.

This gives the player the option to click enter, in which case the player will initiate a conversation, which will allow them to see a text in a white frame (Figure IX). NPCs will be used to provide buffs to the character using the in-game currency.

Figure VIII: *NPC with an Indicator*

Figure IX: *NPC with an Indicator & the Frame*

3.3.2 Enemies

Figure X: *Additional Enemies*

Once that was done, it was time to make some more enemies. Making enemies consists of making animation for their actions such as running, hitting, death and so on. Secondly, an animator was just in place for enemies such as the ones above in Figure X. Figure XI shows the animator which links all the animations of each enemy together and allows for transition from one animation to another.

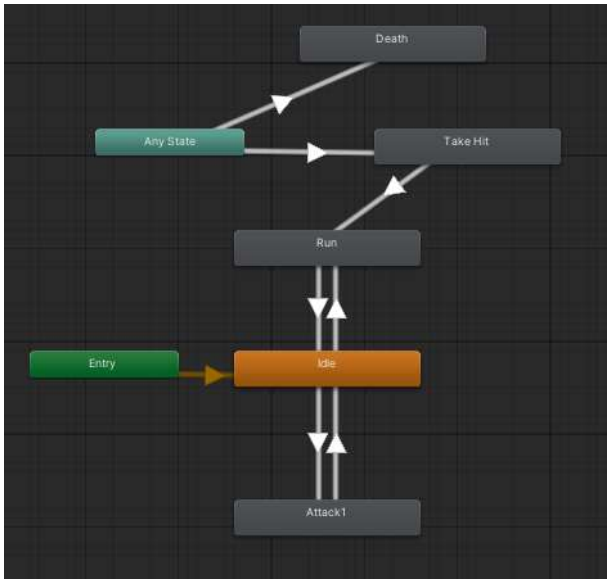

Figure XI: *Animator for enemies*

3.3.3 Bosses

An essential part of any Rouge-like is the variety of the enemies the player can fight with, and after working on some different designs with a static basic attack, we wanted to implement something somewhat more challenging.

Figure XII: *First boss*

The boss had to have a different feel of what the game was so far in order to change the pace and defy the player's expectation, making it really feel like a challenge, so we went for a mix of ranged and close combat.

It was important for us to make the boss a lot more dynamic and "fast", so the first part we handled was creating a melee attack. That in itself was not incredibly difficult because so far that was our enemies main way of attacking the player, but it became a lot more challenging once we were trying to code the ranged attack.

Figure XIII: *First boss ranged attack*

The challenge came from the fact that we needed to look for very specific methods to try and figure out the relative position of the player from the boss, and based on that criteria chose exactly what to do. Overall it all came down to finding better ways to use triggers in the animations as well as implementing a randomizer to pick which attack to queue.

3.4 Enemy AI : David-Ortiz Calderon

For this version of the project, our biggest improvement in the enemy AI comes from the "brain" we used for the boss. By some experimentation, we found a way to check for the player's position and follow it. Furthermore, we used the built-in animator feature in unity to simulate the logic of a machine state diagram, which is just a way to fully explain the options the boss could take on each step of the way.

This is an incredibly powerful feature, as we can now define in a couple of scripts inside the animator, a great number of behaviors an enemy could take on depending on its current state, any state.

3.5 Network : Maxime François-Bardouil

Before starting to implement a multiplayer functionality into our game, we researched quite a bit about the different options that the unity asset store provides and ended up looking mostly at two of the most used networking assets for unity, Proton and Mirror.

3.5.1 Unity Asset: Mirror

Mirror is a unity asset that focuses on providing projects with the ability to create a LAN room hosted on one of the player's computer. The advantages of mirror are that it is completely free, that it allows for a very large number of players to connect to the same room, and that it can support large scale projects.

Figure XIV: *Mirror Logo*

To implement Mirror, we started by creating an object that would take care of transmitting the data from player to player by creating an empty object and adding the Network Manager components (Figure XV). After this, we had to make some changes to our previous progress to adapt it to the mirror environment. These changes included giving all current objects a network identity component so that the network would recognize them (figure XVI), but also deleting the player's character and making it the server player, meaning that when a player joins the server, it will automatically appear with this character.

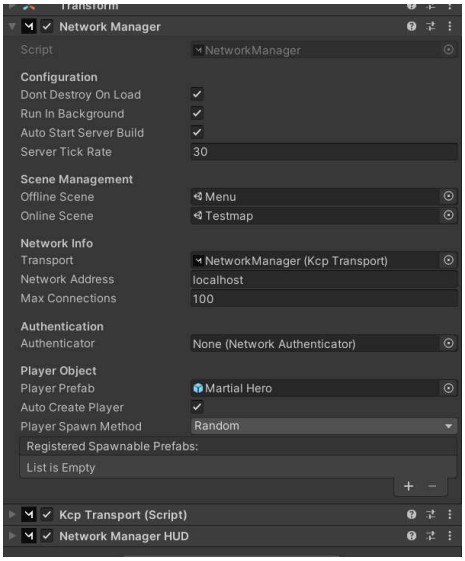

Figure XV: *Network Manager Object*

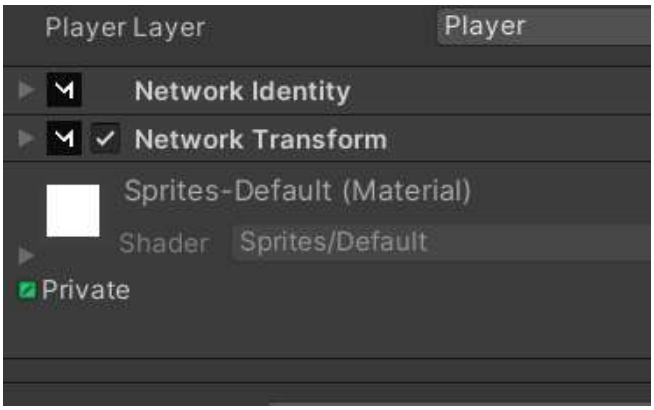

Figure XVI: *Network Identity Components*

Some changes also had to be made in the existing C# scripts so that the enemies and HUD would be recognized by the server and that the player actions were synchronized for all players.

3.5.1 Unity Asset: Photon

The Photon unity asset, also known as PUN 2 (Photon Unity Networking) is another networking asset that offers something a little different from mirror. Its main advantage is that the asset allows us to host our game through a server, meaning that players can play together from across the globe. Photon is also known for being very user friendly in the implementation and usage of the asset. Unfortunately, some of the features are limited due to the asset having a paid version.

Figure XVII: *Photon logo*

To implement Photon, the first step was to create buttons that allowed the player to Create or Join servers using an input field that will be used as the room address. TO create the buttons, we made a C# script that contained a CreateRoom, JoinRoom, and LeaveRoom function, which were linked to each button. These buttons have been added to the map screen for the testing but will be moved to the main menu after the full implementation of the networking (Figure XVIII).

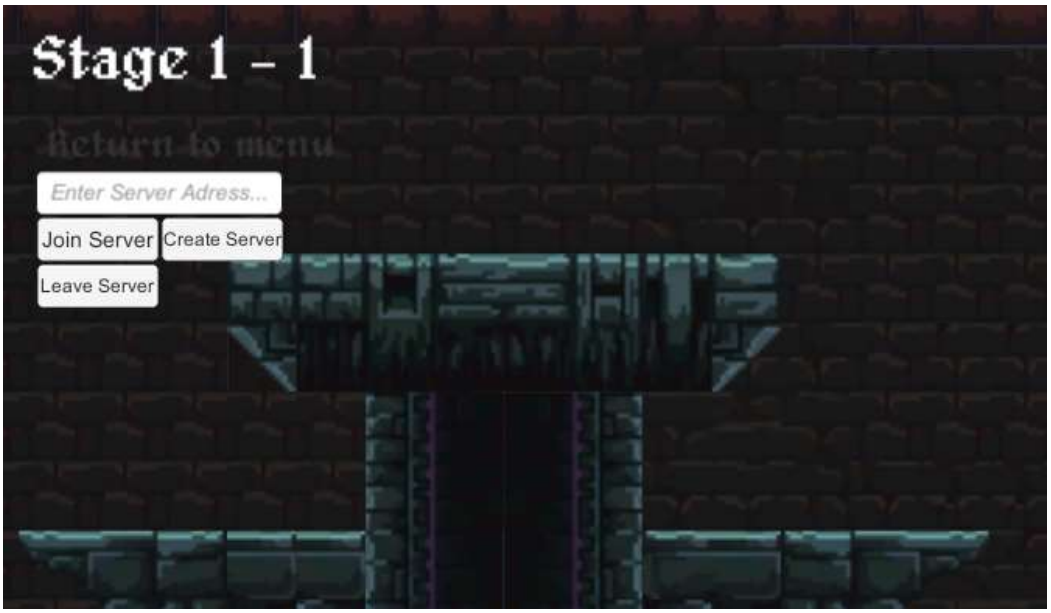

Figure XVIII: *Photon Network Buttons*

After making sure that the buttons worked, we proceeded to remove the camera from the map and added it to the player preset directly. This allowed us to easily hide the camera for all the players other than the local player.

The Photon Networking asset is the one we decided to continue the project with because it gave us more freedom for where we wanted to take the project in the future.

Our next goal for the networking is to synchronize the view and actions of all players through photon and manage to get a fully working multiplayer testmap. Once this has been achieved, we will be able to fully implement the multiplayer into our main project.

3.6 Website : Raj Mahajan Aka Mali

We created our website using Wix, an online service which provides the ability to create a website for free. On this website, we decided to include our logo, a quick summary of the storyline, our current progress, as well as download buttons for the past project reports.

Website link:

https://polloyfaust.wixsite.com/website

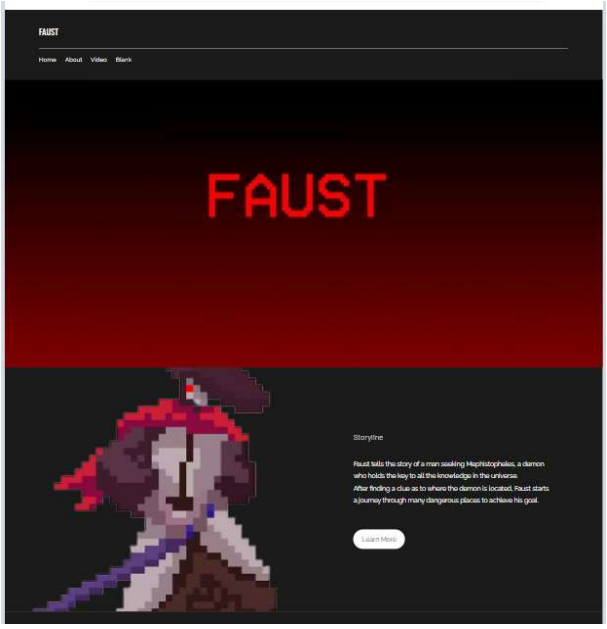

Figure XIX: *Faust Website*

Chapter 4

CHALLENGES

This section explores the difficulties faced by the Pollo Y Papa group while developing the game. This includes the topics we most struggled at as well as the problems we encountered.

4.1 Unity Collaboration

At the start of the development of the project, we used Unity Collaborate, this in itself took a few weeks to get used to. For example, on many occasions we encountered problems while syncing which led to days worth of work to be overwritten by the Unity collab feature. However, once we finally understood how to use Unity Collaborate, the project was migrated to a new way of collaboration called Plastic SCM. We were forced to use Plastic SCM because the previous solution (Unity Collab) was disabled on the project.

Therefore, we had to familiarize ourselves with the new application, and while we tried, we were not able to make it work. Due to this we were forced to use the “EXPORT” and “IMPORT” options in Unity Project. This meant we would each work on out section and once finished would import the specific files that were edited and send them to the others in the team to import.

While this worked it was not the most optimal thing to do, therefore for the next defense we will try our best to learn Plastic SCM.

4.2 Finding assets

Finding the perfect assets has been a concern of the team since the start of the project. This is because the game, Faust, requires assets that fit the style we are going for, which is a 2D dark pixel art. While we do own assets for maps we are incredibly limited, and as we develop more the game we are finding it difficult to find assets that will satisfy the team. Secondly, the prices of the assets also have to be taken into account. Since most assets that fit our style cost around 10 - 15 Euros each, it becomes expensive and not worthy enough to buy them. Lastly, there exists limited enemy assets that we own, hence we have to look to buy some.

Figure XX: *Possible Enemy Asset*

4.3 Network

The main challenge we faced while working on the network was understanding how the networking assets worked and what were the differences between them. It took a long time to understand how the building C# functions behaved and when they had to be used since the documentation regarding some of the assets was limited or outdated.

It was also very hard to debug due to it having a lot of issues related to our previous code which was not made with multiplayer features in mind. The testing also required two instances of the game being open at the same time, leading to it being much longer than if we could just test through unity editor (we had to launch an instance through the editor and build & run a second one for every test).

Chapter 5

FUTURE PROSPECTS

The following table presents the tasks needed to be completed for the next defenses.

Presentation	Tasks
6 - 17 June, 2022	1. Installation Manual and Operating Manual 2. Map Generation a. Fully operational and optimized map generation AI 3. Network a. Be able to host 2 player online game 4. Save File a. Progress save after game exit 5. Audio a. Ambient Sounds b. In-Game Music c. Sound Effects 6. Finish website a. Downloadable versions of the game (with and without audio) b. Timetable of the game progress

The following section explains additional features or changes we would like to implement.

5.1 Maps

When it comes to the Maps, The First thing we need to do is complete all the other levels. Afterwards, we are going to implement functional spike traps that would damage the player on contact.

We also like to make it so that the player can only move on to the next room if they cleared the current one of all enemies so that the player cant just ignore the enemies and run past them to the door.

With that would have achieved all the initial plans we had for the maps, But We would also like to experiment with other things.

For example, as it currently stands, The rooms of each level are separated with each one being on its own scene. to make the game more engaging we are going to play around with the idea of putting all the rooms on the same scene and connecting them with hallways instead, to give the impression of a bigger map.

Another thing we might add if we are done with the maps early is more rooms for each level so that we can create bigger levels and add more variety.

5.2 Enemy AI

As of right now, our enemies can detect the player if they are directly in front of them, and will ignore it the second they get out of sight. Clearly, this is not the best way to handle detection, so for the future, and thanks to the things we learned while coding the AI for the first boss, we could cast a circle from the enemies position and if the player is within a certain range, only then they could follow it.

The trickiest part about this implementation is that the player can move and jump around, so this must be taken into account going into the future.

5.3 New Character

While the current main character is a good basic character, there is still room for improvement for a better main character who fits the dark style we are looking for. For example, Figure XXI is an image of a 2D Dark Knight asset that fits the theme. Secondly, this asset has a multitude of animations and attacks such as sliding and dodging which were one of the improvements we have looking at for our current player.

Getting this character will allow us to make the game fit well together as well as allow the player to a variety of attacks, with its abundance of animations.

Figure XXI: *Dark Knight*