Technical Design Document (TDD)

Contents

[CRC cards 2](#_Toc97905322)

[Architecture 6](#_Toc97905323)

[Technology 6](#_Toc97905324)

[Sprint 1 Report 13](#_Toc97905325)

[Sprint 2 Report 15](#_Toc97905326)

[Sprint 3 Report 17](#_Toc97905327)

# 

# CRC cards

Graphical user interface, text

Description automatically generated with medium confidence

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Text

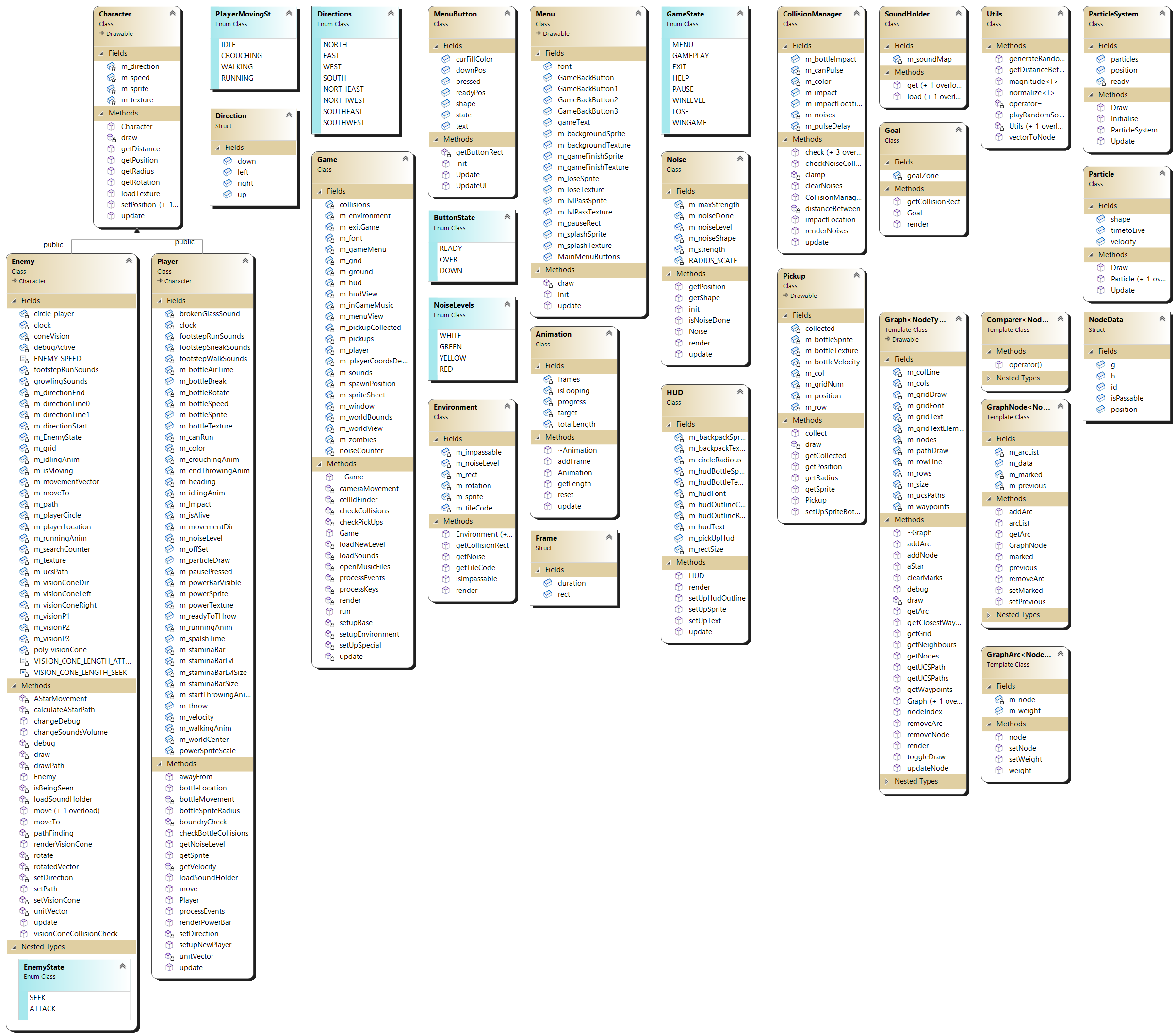
Description automatically generated

Graphical user interface, text, application

Description automatically generated

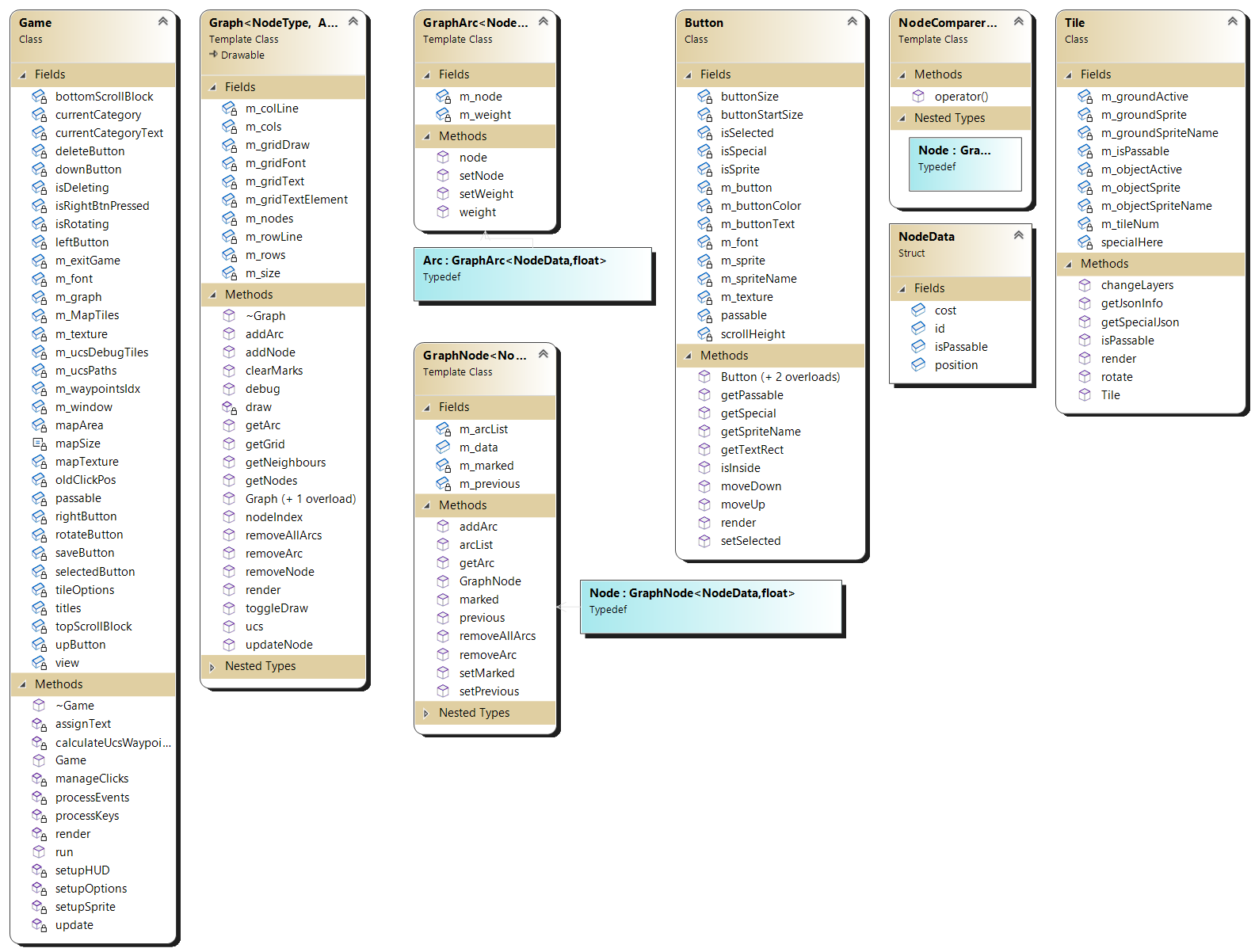
# Architecture

## Game Class Diagram



**Version 3**

## Level Builder Class Diagram



**Version 1**

# Technology

**Research**

* **JSON for Modern C++**

We needed a library to parse and create JSON files to store our level data (i.e.: tiles position, pre-computed UCS Pathfinding, …). We tried to find the best and most easy-to-use one and we found this one, JSON for Modern C++, made by nlohmann which is the most famous JSON library used for C++.

* **tinyc2.h:**

I choose to use tiny2c because it was a pre-made library and it enabled us to easily detect a collision between enemy vision cone and player (circle). Cause we originally did a point to triangle collision check for it and it was not efficient.

**Installation**

* **JSON for Modern C++**

The library is integrated with a single header file which is called json.hpp and contains all the library. This is the simplest way to integrate this library.

Here is the link to the documentation and download:

* <https://json.nlohmann.me/>
* <https://github.com/nlohmann/json>
* **tinyc2.h**

The library contains creation for different shapes like polygons, circles, lines, rectangles, etc.

Then you can use the functions to detect the collisions between them. Everything is in a single header file which makes it easy to implement and use.

link: https://github.com/RandyGaul/tinyheaders/tree/master/examples\_tinygl\_and\_tinyc2

**Technical achievement**

* Adrien:

Because I am new in this course and did not have any previous experience with SFML, I think that all the project was really challenging for me, as I had to learn a totally new library and technologies and be productive at the same time. Even though I already have experience programming and developing software.

However, if I had to choose a really challenging part, it probably was the pathfinding algorithm and implementation, the majority of my time for the third sprint has been used for this.

This code snippet taken from the Level Builder shows how the UCS Pathfinding is pre-computed after a click on the Save Button.

It is necessary to update the “passable” property of each tile to properly calculate the pathfinding.

The resulted paths are then exported to the JSON file.

Text

Description automatically generated

The part which calculated the waypoints is also interesting and allows to have any size of grid without breaking the waypoint positions which will always be centered at the same place (depending on the size of the grid of course).

Text

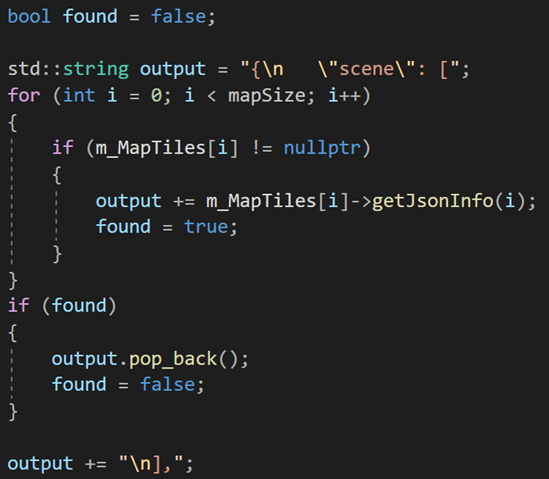
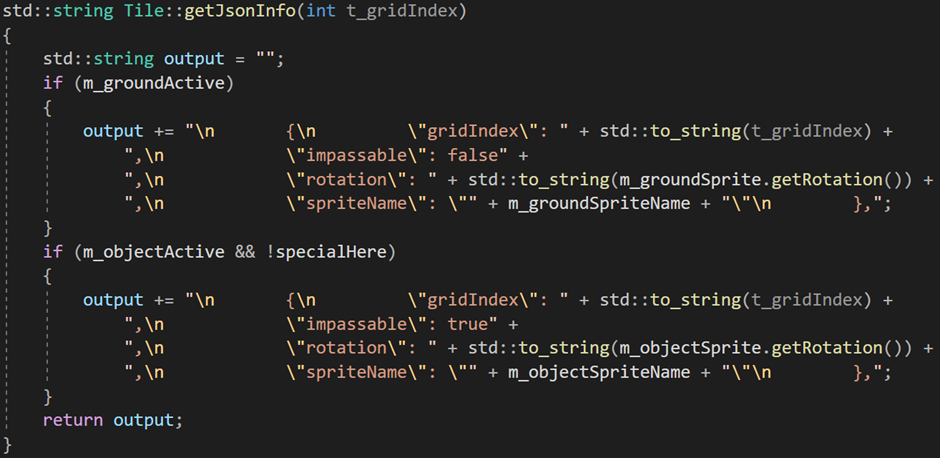
Description automatically generated

I am also really happy with my work to move the zombie following the calculated pathfinding with A\* and UCS. Here is the code snippet of the “moveTo” method which is used to move the zombie from one point to another using only the pathfinding.

Text

Description automatically generated A screenshot of a computer

Description automatically generated with medium confidence

* Eoin: I was in charge of making the level builder, this required managing a lot of data and organizing it as well as giving clear feedback to the player as well as the best way to output that data to the JSON file, It took me a bit but once I had my code layout planned out it all seemed to fall into place
* 
* The above snippet is the basics of how I exported to the JSON file
* 
* The above snippet was how I output tiles information, separating the ground layer and object layer for each tile
* Masih:

Although most of the projects we did up until now were in SFML, I can say I learned a lot during this project. The use of JSON files and reading data from it was very interesting and fun. I would say the most challenging part for me was the vision cone and the collision check I ended up using an external library for collision detection.

Code snippet is as follow:







# Sprint 1 Report

**Summary of planned work**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Features and tasks** | **Time  Estimate** | **Time Actual** | **Team Member** | **Complete** |
| **Feature 1: Moving Player** | | | | |
| Task 1: Render a rectangle | 1 hr | 2hr | Masih | Yes |
| Task 2: Setup input handler | 30 min | 30 min | Masih | Yes |
| Task 3: Player movement | 1hr | 2hr | Adrien | Yes |
| Task 4: Improve movement | 1 hr | 1 hr | Eoin | Yes |
| **Feature 2: Enemy** | | | | |
| Task 1: Render the Enemy | 1 hr | 1hr | Masih | Yes |
| Task 2: Setup Collision with Player | 1 hr | 1 hr | Eoin | Yes |
| Task 3: creating enemy vision cone | 2 hr | 4hr | Masih | Yes |
| Task 4: detecting collision between player and enemy vision cone | 2 hr | 3hr | Masih | Yes |
| Task 5: Improve Movement | 30 min | 1 hr | Eoin | Yes |
| **Feature 3: Environment** | | | | |
| Task 1: Render the object | 30 min | 30 min | Eoin | Yes |
| Task 2: Setup Collision with player and enemies | 1 hr | 1 hr | Eoin | Yes |
| Task 3: Setup Constructor for the level loader | 30 min | 30 min | Eoin | Yes |
| **Feature 4: Noise** | | | | |
| Task 1: Draw the Noise | 1 hr | 1 hr | Masih | Yes |
| Task 2: Allow Noise to be added | 1 hr | 1 hr | Eoin | Yes |
| Task 3: Make the Noise vary based on the noise level | 1 hr | 2 hr | Eoin | Yes |
| Task 4: Setup a radius that detects nearby zombies based on the noise level | 1 hr | 2 hr | Eoin | Yes |
| **Feature 5: Pickups** | | | | |
| Task 1: Draw Pickups on the screen | 30 min | 30 min | Eoin | Yes |
| Task 2: Setup Collision with Player | 1 hr | 1 hr | Eoin | Yes |
| Task 3: adding the sprite of pickups | 30 min | 1hr | Masih | Yes |
| Task 4: adding the key input for throwing the pickups | 30 min | 1hr | Masih | Yes |
| Task 5: adding the power bar | 30 min | 30 min | Masih | Yes |
| **Feature 6: Enemy Vision Cone** | | | | |
| Task 1: setting up the cone | 1 hr | 2 hr | Masih | Yes |
| Task 2: cone state changing | 30 min | 1 hr | Masih | Yes |
| Task 3: cone and player collision | 1 hr | 3 hr | Masih | Yes |
| **Feature 7: View/Camera** | | | | |
| Task 1: setting up the view | 2hr | 3hr | Adrien | Yes |
| Task 2: render element to the view | 30min | 1hr | Adrien | Yes |
| Task 3: implement sf::Drawable to all the game objects | 30 min | 30 min | Adrien | Yes |
| **Feature 8: Animation** | | | | |
| Task 1: create the Animation class | 1hr | 3hr | Adrien | Yes |
| Task 2: scale all the sprites | 3h | 3hr | Adrien | Yes |
| Task 3: create sprite sheets | 30min | 1hr | Adrien | Yes |
| Task 4: create the animations | 1hr | 1hr | Adrien | Yes |
| **Feature 10: Menu & UI** | | | | |
| Task 1: setting up the menu buttons | 20 min | 20 min | Masih | Yes |
| Task 2: creating and setting up the menu | 1 hr | 1hr | Masih | Yes |
| Task 3: adding visual improvements | 30 min | 30 min | Masih | Yes |

# Sprint 2 Report

**Summary of planned work**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Features and tasks** | | **Time  Estimate** | | **Time Actual** | **Team Member** | | **Complete** |
| **Feature 1: Moving Player** | | | | | | | |
| Task 1: fix player state | | 30min | | 2hr | Adrien | | Yes |
| **Feature 8: Animation** | | | | | | | |
| Task 1: split the throwing animation into two | | 1hr | | 1hr | Adrien | | Yes |
| **Feature 9: Level Builder** | | | | | | | |
| Task 1: basic script to generate the environment | | 1hr | | 2hr | Adrien | | Yes |
| Task 2: Draw Layout | | 1 hr | | 1 hr | Eoin | | Yes |
| Task 3: Make Button Object | | 30 mins | | 30 mins | Eoin | | Yes |
| Task 4: Generate and Draw Buttons | | 30 mins | | 30 mins | Eoin | | Yes |
| Task 5: Setup Menu Interaction | | 30 mins | | 30 mins | Eoin | | Yes |
| Task 6: Setup Scrolling on menu | | 1 hr | | 3 hrs | Eoin | | Yes |
| Task 7: setup interaction with map | | 1 hr | | 2 hrs | Eoin | | Yes |
| **Feature 10: Menu & UI** | | | | | | | |
| Task1: Setting up the HUD | 1hr | | 1hr | | | Masih | Yes |
| Task2: Finding and Setting Sprites | 2hr | | 2hr | | | Masih | Yes |
| Task3: interaction with pickups | 1hr | | 1hr | | | Masih | Yes |
| **Feature 11: Sound** | | | | | | | |
| Task 1: Setup SoundHolder | 2 hr | | 3hr | | | Adrien | Yes |
| Task 2: Find sounds | 4hr | | 4hr | | | Adrien | Yes |
| Task 3: Edit sounds | 1hr | | 2hr | | | Adrien | Yes |
| Task 4: Add and sync footsteps sound with animation | 1hr | | 1hr | | | Adrien | Yes |
| Task 5: update footstep sounds volume depending on the distance to player | 1hr | | 2hr | | | Adrien |  |
| Task 6: Add ambient sounds/music | 3hr | | 1hr | | | Adrien/Masih | No |

# Sprint 3 Report

**Summary of planned work**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Features and tasks** | **Time  Estimate** | **Time Actual** | **Team Member** | **Complete** |
| **Feature 2: Enemy** | | | | |
| Task 1: A\* pathfinding implemented | 2 hr  30 min | 2 hr  1 hr | Masih  Adrien (help for debugging) | Yes |
| Task 2: UCS pathfinding implemented | 3 hr | 4 hr | Adrien | Yes |
| Task 3: Enemy reacts to noise | 1 hr | 2 hr | Adrien | Yes |
| Task 4: Path is drawn in Debug mode | 1 hr | 1 hr | Eoin | Yes |
| Task 5: Impassable objects are highlighted in Debug mode | 1 hr | 1 hr | Eoin | Yes |
| Task 6: movement according to the calculated pathfinding (A\* and UCS) | 4 hr | 3 hr | Adrien | Yes |
| **Feature 4: Noise** | | | | |
| Task 1: Ground varies in Noise | 1 hr | 1 hr | Eoin | Yes |
| Task 2: Noise varies according to the player speed (linked to the ground type as well) | 2 hr | 2 hr | Adrien | Yes |
| **Feature 5: Pickups** | | | | |
| Task 1: power bar | 2 hr | 2 hr | Masih | Yes |
| Task 2: pick up a throw | 2 hr | 5 hr | Masih | Yes |
| Task 3: bottle break (particle system) | 2 hr | 2 hr | Masih | Yes |
| Task 4: noise creation | 1 hr | 1 hr | Eoin/Masih | Yes |
| **Feature 9: Level Builder** | | | | |
| Task 1: Allow Player to rotate objects | 2 hr | 2 hr | Eoin | Yes |
| Task 2: Allow player to delete objects | 30 min | 30 min | Eoin | Yes |
| Task 3: Add special objects | 30 min | 30 min | Eoin | Yes |
| Task 4: Add parsing of special objects in-game | 1 hr | 1 hr | Eoin | Yes |
| Task 5: Exporting level information to a JSON file | 1 hr | 1 hr | Eoin | Yes |
| Task 6: Check Whether level already exists | 30 min | 30 min | Eoin | Yes |
| Task 7: Adapt Level Builder window to screen size | 2 hr | 3 hr | Adrien | Yes |
| Task 8: Map scrolling with mouse | 3 hr | 3 hr | Adrien | Yes |
| Task 9: UCS waypoint automatic calculation | 1 hr | 2 hr | Adrien | Yes |
| Task 10: pre-calculation of UCS Pathfinding on save | 2 hr | 3 hr | Adrien | Yes |
| **Feature 10: Menu & UI** | | | | |
| Task 1: adding pause menu | 2 hr | 3 hr | Masih | Yes |
| Task 2: adding help menu | 3 hr | 5 hr | Masih | Yes |
| Task 3: adding load screen | 30 mins | 30 min | Eoin | Yes |
| Task 4: Dynamically load levels | 1 hr | 1 hr | Eoin | Yes |
| Task 5: When the last level is complete trigger win | 30 min | 30 min | Eoin | Yes |
| Task 6: implementing background screen to different game states | 2 hr | 2 hr | Masih | Yes |
| **Feature 11: Sound** | | | | |
| Task 1: add growling sounds for zombies | 30 min | 30 min | Adrien | Yes |
| Task 2: add bottle break sound effect | 30 min | 30 min | Adrien | Yes |
| Task 3: add music (menu and game ambient music) | 1 hr | 1 hr | Adrien | Yes |