

**National College of Ireland**

2020/2021

Shane Mulrooney, x19454016, x19454016@student.ncirl.ie

Sean Fulton, x19518013, x19518013@student.ncirl.ie

Cree Gunning, x19302733, x19302733@student.ncirl.ie

Plague Panic

Technical Report

**Table of Contents**

[Executive Summary 1](#_Toc70170926)

[1.0 Introduction 2](#_Toc70170927)

[1.1. Background 2](#_Toc70170928)

[1.2. Aims 2](#_Toc70170929)

[1.3. Technology 2](#_Toc70170930)

[2.0 System 3](#_Toc70170931)

[2.1. Requirements 3](#_Toc70170932)

[2.1.1. Functional Requirements 3](#_Toc70170933)

[2.1.1.1. Use Case Diagram 3](#_Toc70170934)

[2.1.1.2. Requirement 1: Start Game 3](#_Toc70170935)

[2.1.1.3. Description & Priority 3](#_Toc70170936)

[2.1.1.4. Use Case: Start Game 3](#_Toc70170937)

[2.1.1.5. Requirement 2: End Game 4](#_Toc70170938)

[2.1.1.6. Description & Priority 4](#_Toc70170939)

[2.1.1.7. Use Case: End Game 4](#_Toc70170940)

[2.1.2. Non-Functional Requirements 5](#_Toc70170941)

[2.2. Design & Architecture 5](#_Toc70170942)

[2.3. Implementation 6](#_Toc70170943)

[2.4. Graphical User Interface (GUI) 7](#_Toc70170944)

[2.5. Testing 9](#_Toc70170945)

[3.0 Conclusions 11](#_Toc70170946)

[4.0 Further Development 11](#_Toc70170947)

[5.0 Appendices 12](#_Toc70170948)

[5.1. Project Plan 12](#_Toc70170949)

[5.1. Collaboration Summary 12](#_Toc70170950)

# Executive Summary

The idea for this project stemmed from our combined interest in games and we sought out to create something that is both challenging and fun for the player. Using the beginner-friendly game engine Godot, our goal was to combine multiple elements from various genres and games to establish a unique playing experience.

This report will describe in detail, the use cases, functional and non-functional requirements, implementation, and testing that were committed during the planning and development phases of our project. We will discuss the reasons for developing our game and the advantages and disadvantages that go along with developing the type of game we chose to create. Our game was developed for Windows machines with the potential to be ported to different devices in the future.

# **Introduction**

## Background

Being gamers ourselves, we wanted a challenge to develop a game of our own and found our Team Project module to be a great opportunity to do so. Small development teams who make intricate and complex games has always fascinated us, so we sought to understand the difficulties and complexities of developing a game with a small group of people.

## Aims

The main goal for our project is to develop a game that is both fully functional and a fun experience for the player. Combining different elements from various genres of games has allowed us to create something uniquely fun for players who like to challenge themselves.

With the limited time we had to complete this project, we decided to create a game that does not necessarily have a main objective, but to instead challenge the player to better their score every time they play.

## Technology

At the beginning of the development process for our game, we set out to use any sufficient game development tool that would allow us to achieve the game we had envisioned. This led us to Pygame. Pygame is a game development module, as part of the Python programming language. Pygame games are developed without a UI, so the entire game is created using code. We ran into trouble early on with Pygame, finding it increasingly difficult to visualise the development of our game. At this point we decided that a game engine would be preferable for our development process. This is when we chose Godot as our main development technology.

Godot is a free to use game development engine which makes use of scenes and nodes, which act as classes, in a hierarchal, tree-like structure. Godot’s UI is easy to learn and great for beginners of game development. The scripting language is GDScript, which is very similar to Python, with some minor differences. All of these factors made choosing Godot as our main development engine very easy for us.

Godot allowed us to achieve elements of our game that we initially thought would be extremely difficult or even impossible to implement with our experience with game development, however, with sufficient research and perseverance we were able to achieve the goals we stated in the planning phase of our game.

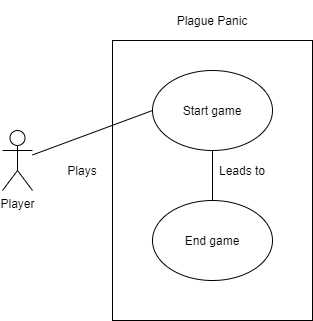
# **System**

## Requirements

## Functional Requirements

1. Functional Requirement: Start Game
2. Functional Requirement: End Game

## Use Case Diagram



## Requirement 1: Start Game

## Description & Priority

This requirement is of highest priority as it grants the player the ability to access the functions of the game.

## Use Case: Start Game

**Scope**

The scope of this use case is to allow the Player to start the game and access all the functions the game has to offer.

**Description**

This use case describes the Player starting the game.

**Flow Description**

**Precondition**

The system is awaiting to be launched by the user.

**Activation**

This use case starts when the Player runs the application.

**Main flow**

1. The Player runs Plague Panic application.
2. The Player selects Start Game from the Main Menu.
3. The Player plays the game.

**Alternate flow**

1. : Player quits game from Main Menu.
2. The Player opens Plague Panic.
3. The Player selects Exit Game from the Main Menu.
4. The game closes.

**Termination**

Upon termination this Use Case presents the End Game requirement.

**Post condition**

The Player has the ability to play the game.

## Requirement 2: End Game

## Description & Priority

This requirement enables the game to end when the character the Player chooses to exit the game.

## Use Case: End Game

**Scope**

The scope of this use case is to enable the game application to close when the Player decides to quit the game.

**Description**

This use case describes the Player exiting the game.

**Flow Description**

**Precondition**

The game is running.

**Activation**

This use case starts when the Player chooses the quit game option from the main menu or game over screen.

**Main flow**

1. The game is running.
2. The Player’s character dies.
3. The Player exits the game.

**Alternate flow**

1. : The Player quits the game.
2. The game is running.
3. The Player exits the game from main menu.

**Termination**

Upon termination the game application closes and the system ends.

**Post condition**

The system is finished.

## Non-Functional Requirements

**Performance Requirement**

The game should run at a consistent framerate of 60 frames per second to allow for a hight performing game experience for the player.

**Robustness Requirement**

At no point should the game crash or become unavailable for the user to play.

**Availability Requirement**

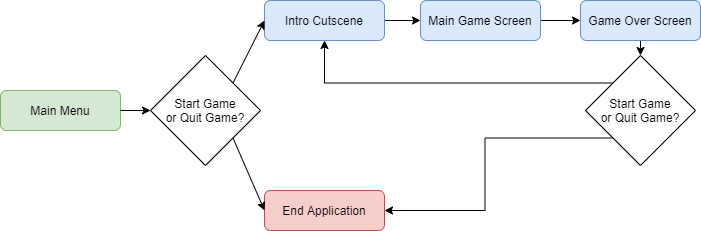
The game should be available to any player to play at any time as long as the game is downloaded to their machine.

**Maintainability**

The game should be designed in a way that allows for easy future updates and expansions to content within the game.

## Design & Architecture

This Flowchart represents an overview of our application ‘Plague Panic’

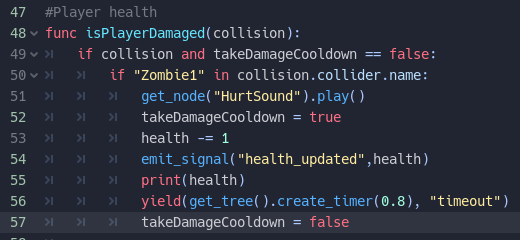


Plague Panic’s architecture was created to allow the user upon, executing the application, to be presented with the Main Menu be prompted with a decision to either quit the application or progress on to the next screen by pressing “Start Game” if the user chooses the “Start Game” option they will be shown the “Intro Cutscene” of the game. Upon the completion of the cutscene, the user will be greeted with the “Main Game Screen” of Plague Panic where the user must survive as long as they can without being defeated. When defeated, the user will be presented with the “Game Over Screen” which prompts another decision to the user to either restart at the Main Level Scene (skipping the Intro Cutscene) or quitting the application.

## Implementation

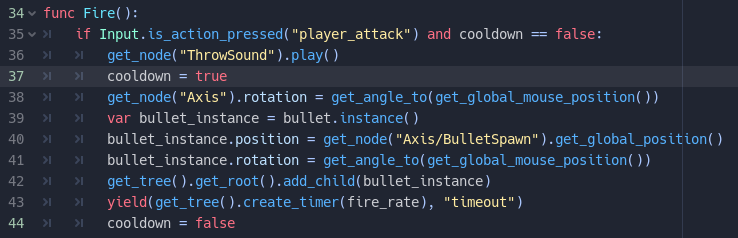
**Player Health**

When a player comes into contact with an enemy within the game, the player’s health should decrease and be displayed for the user. To do this, a function called isPlayerDamaged() is implemented that checks if the player sprite has collided with an enemy sprite and in turn, decreases the player’s health value. This is represented with the following snippet of code:



**Player Attack**

The player has the ability to a fire a weapon at the enemies in the game. To achieve this we created a function called Fire(). Fire() creates an instance of a bullet scene every time a particular button is pressed by the user (in this case, the “player\_attack” button which is mapped to left mouse button). This is represented with the following snippet of code:



**New Round**

Every time the player defeats the required number of enemies in a round, the following round will begin and in turn, the round number increases and the number of enemies that spawn increases also. This is represented by the newRound() function shown below:



## Graphical User Interface (GUI)

**Main Menu Screen**

****

This is the first screen the user will see when the application has run. This menu gives the player 2 options, to either start the game or to quit the game.

**Intro Cutscene Screen**

****

This screen appears after the user selects the Start Game option from the main menus screen. A short story is given to the player for some context to the game followed by the title of the game.

**Main Game Screen**

****

This is the main game screen where the player plays the game. This screen appears directly after the intro cutscene has run. This is where the player controls the player. There are 2 notable UI elements on this screen. Up the top left side of the screen there is information displaying the current round number, which will increase if the player defeats a certain number of enemies. The bottom left of the screen displays the players health bar which deteriorates when the player comes into contact with an enemy.

**Game Over Screen**

****

This is the screen that will appear when the player’s health has fully depleted. Displayed on this screen is the information of which round number the player had made it to with 2 options to either play the game again from the main game screen or to quit the application.

## Testing

**Player Movement**

Expected Result: User presses any of the directional keys (WASD), the Player character moves in the respective direction.

Actual Result: The user pressed the WASD keys and the Player character moved in the respective direction.

Test Result: **Pass**

**Player Shooting**

Expected Result: When the user clicks the left mouse button a bullet should propel from the Player character and travel towards the direction of the mouse.

Actual Result: The user clicked the left mouse button and the bullet continuously travelled in opposite direction to the mouse position.

Test Result: **Fail**

**Player Shooting**

Expected Result: When the user clicks the left mouse button a bullet should propel from the Player character and travel towards the direction of the mouse.

Actual Result: The user clicked the left mouse button and the bullet travelled from the Player character’ sprite towards the position of the mouse.

Test Result: **Pass**

**Enemy Spawns**

Expected Result: When each round starts a certain number of zombies should spawn (Depending on the current round number.) in the set spawn points around the map.

Actual Result: The main level scene had loaded, and the correct number of zombies spawned in their respective spawn points.

Test Result: **Pass**

**Round System**

Expected Result: When the player character defeats the required number of zombies to progress to next round the current round will be updated to the next round.

Actual Result: The player defeated the last zombie during the current round, the game progressed to the next round spawning more zombies for the player character to defeat.

Test Result: **Pass**

**Start Game Button (Main Menu)**

Expected Result: Once the user clicks the Start Game button on the main menu of the application the application will change to the main level scene.

Actual Result: The user clicks the Start Game button on the main menu and the application changes to the main level scene.

Test Result: **Pass**

**Quit Game Button (Main Menu)**

Expected Result: Once the user clicks the Quit Game button the application window will close.

Actual Result: The user clicks the Quit Game button and the application closes.

Test Result: **Pass**

**Play Again Button (Game Over Menu)**

Expected Result: When the user presses the Play Again button is pressed the level scene restarts.

Actual Result: The user pressed the Play Again button and the intro cutscene played again.

Test Result: **Fail**

**Play Again Button (Game Over Menu)**

Expected Result: When the user presses the Play Again button is pressed the level scene restarts.

Actual Result: The user pressed the Play Again button and the game restarted by loading the level scene.

Test Result: **Pass**

# **Conclusions**

The advantage of this game is having a unique gaming experience that combines multiple elements from different genres of games. The premise of a game that does not necessarily give the player an end goal but to instead allow the player to attempt to beat their score every time they play, creates a different element of challenge with every play through. This decreases the likelihood of the game becoming stale to the player.

A disadvantage of this game in its current state is that it has only been developed for Windows machines in mind. This restricts a lot of players who may be more familiar with gaming on their mobile device or home consoles etc.

With a lot more time and effort, the potential this game is huge. More weapons and enemies at the player’s disposal will create a more varied experience for the player while also introducing different elements of challenge. A co-operative multiplayer experience could establish a whole different experience for the player by allowing friends to join together and try to beat their score together as a team.

# **Further Development**

Given more time to work on this project there are numerous ideas we have which would improve the overall quality of our game. The ideas we have in mind are as follows:

**Additional Weapons:** We would like to give the player the option of having different weapons at their disposal.

**Enemy Variations:** To counter the additional weapons, we would also different variations of enemies to the game to add a more challenging aspect to the game.

**Co-operative Multiplayer:** We feel co-op multiplayer would be an excellent addition to our game. Giving players the option to play our game with their friends would be a difficult but rewarding challenge.

# **Appendices**

## Project Plan

**Preparatory Phase – 24th January – 1st February**

This phase of our project plan allowed us to come up with a basic idea for our project and decide on the technologies we would use to complete it.

**Research Phase – 1st February – 14th February**

This phase allowed us to get an understanding of Godot and GDScript, the engine and language we chose to develop our game. During this phase, we each separately researched different elements of the engine and language that we could later combine to begin the development process.

**Development Phase – 15th February – 15th April**

This was the phase in which the vast majority of the development of our game took place. During this phase weekly Microsoft Teams meetings were conducted to ensure adequate work was being completed by each team member and that deadlines were being met for various elements of the project.

**Finalising and Testing Phase – 16th April – 25th April**

This was arguably the most important phase of our project as it allowed us time to tweak any bugs and polish any elements of the game that required finish touches.

## Collaboration Summary

Our main tool for collaborative work which we used was GitHub Desktop. This application allowed us to push and pull the work we had completed individually without losing track of who did what. Some of the Git Log will be displayed below:

**Player Character Git Log**

