University of Plymouth

School of Engineering,

Computing, and Mathematics

COMP3000

Computing Project

2024/2025

Timeline Takedown

Morgan Tomos Hodge

10779528

BSc (Hons) Computing and Games Development

# Acknowledgements

I would like to thank Anthony Edwards, he has been incredibly helpful this year going out of his way to help out and genuinely caring about issues I encountered or questions I have.

**Abstract**

This report covers the development of Timeline Takedown, a first-person, wave - based survival shooter developed using the Unity game engine (Unity Technologies, 2024). The project takes inspiration from classic round-based survival modes, such as those found in Call of Duty Zombies. The motivation behind this project stems from my personal experience growing up with games in this genre. In recent years, I have observed a noticeable decline in the quality and innovation of similar games, which led me to explore creating my own version - tailored to deliver a fun, replayable experience for my friends, family, and broader audiences who enjoy this style of gameplay.

This report begins with an exploration of existing games within the round-based survival shooter genre, examining what made them successful and identifying the key developers and titles that influenced the field. Following this, I will outline the objectives and deliverables of the project I created, along with the method of approach taken to meet these goals.

The main body of this report focuses on the development and implementation of the game. I will be discussing the project management techniques I used to maintain a steady and efficient workflow throughout the project. I will also be breaking down the development sprints and discussing what was achieved at each phase, and when they took place. Additionally, user testing will be covered detailing the tools and methods used to gather feedback and the resulting changes made to the game.

The final section of the report will feature an end of project review, where I will reflect on what was achieved throughout this project, and the quality of the work achieved. This will be followed by a reflection, evaluating what went well, what could have been improved, and how this experience will affect how I work on projects in the future.

# **Contents**

Contents

[Acknowledgements 2](#_Toc196670999)

[**Abstract** 3](#_Toc196671000)

[**Contents** 4](#_Toc196671001)

[**Word Count**: 10301 6](#_Toc196671002)

[**Code Links**: https://github.com/Mdot5596/Timeline-Takedown 6](#_Toc196671003)

[**1 - Introduction** 7](#_Toc196671004)

[1.1 Overview 7](#_Toc196671005)

[1.2 Audience 7](#_Toc196671006)

[1.3 Purpose 8](#_Toc196671007)

[**2 - Background, objectives & deliverables** 9](#_Toc196671008)

[2.1 Project Background 9](#_Toc196671009)

[2.2 Market / Competitors 9](#_Toc196671010)

[2.3 Objectives & Deliverables 11](#_Toc196671011)

[**3 - Ludology** 11](#_Toc196671012)

[3.1 Introduction 11](#_Toc196671013)

[3.2 Key Examples 12](#_Toc196671014)

[3.3 Application 13](#_Toc196671015)

[**4 - Method of Approach** 13](#_Toc196671016)

[4.1 Methodologies 13](#_Toc196671017)

[4.2 Limitations 14](#_Toc196671018)

[4.3 Alternatives 15](#_Toc196671019)

[4.4 Technologies 15](#_Toc196671020)

[4.5 Project Management Approach 17](#_Toc196671021)

[4.6 Planning and Structuring the Project 17](#_Toc196671022)

[4.7 Visal Planning Tools 18](#_Toc196671023)

[4.8 Tracking Progress & Reflecting 18](#_Toc196671024)

[**5 – Legal, Social, Ethical and Professional issues** 18](#_Toc196671025)

[5.1 Legal 19](#_Toc196671026)

[5.2 Social 19](#_Toc196671027)

[5.3 Ethical 20](#_Toc196671028)

[5.4 Professional 21](#_Toc196671029)

[**6 - Project Management Tools** 22](#_Toc196671030)

[6.1 Project Management 22](#_Toc196671031)

[6.2 Version Control 22](#_Toc196671032)

[6.3 Meetings 23](#_Toc196671033)

[6.4 Development Logs 23](#_Toc196671034)

[6.5 Testing Methods 23](#_Toc196671035)

[**7 – Implementation** 25](#_Toc196671036)

[7.1 Project Setup and Design - Sprint 1 25](#_Toc196671037)

[7.1.1 Overview 25](#_Toc196671038)

[7.1.2 Sprint Tasks 25](#_Toc196671039)

[Summary Of Actions 25](#_Toc196671040)

[7.2 Research and Game states - Sprint 2 26](#_Toc196671041)

[7.2.1 Overview 26](#_Toc196671042)

[7.2.2 Sprint Tasks 27](#_Toc196671043)

[7.2.3 Summary of Actions 27](#_Toc196671044)

[7.3 Assets and Online Resources - Sprint 3 28](#_Toc196671045)

[7.3.1 Overview 28](#_Toc196671046)

[7.3.2 Sprint Tasks 28](#_Toc196671047)

[7.3.3 Summary of Actions 29](#_Toc196671048)

[7.4 Enemy & Main Game Loop - Sprint 4 30](#_Toc196671049)

[7.4.1 Overview 30](#_Toc196671050)

[7.4.2 Sprint Tasks 30](#_Toc196671051)

[7.4.3 Summary of Actions 30](#_Toc196671052)

[7.5 Minimum Viable Product - Sprint 5 32](#_Toc196671053)

[7.5.1 Overview 32](#_Toc196671054)

[7.5.2 Sprint Tasks 32](#_Toc196671055)

[7.5.3 Summary of Actions 33](#_Toc196671056)

[7.6 Animation and Aesthetics - Sprint 6 35](#_Toc196671057)

[7.6.1 Overview 35](#_Toc196671058)

[7.6.2 Sprint Tasks 35](#_Toc196671059)

[7.6.3 Summary of Actions 36](#_Toc196671060)

[7.7 Poster & Level 2 - Sprint 7 37](#_Toc196671061)

[7.7.1 Overview 37](#_Toc196671062)

[7.7.2 Sprint Tasks 37](#_Toc196671063)

[7.7.3 Summary of Actions 38](#_Toc196671064)

[7.8- Zombie Behaviour & Public Playtest - Sprint 8 38](#_Toc196671065)

[7.8.1 Overview 38](#_Toc196671066)

[7.8.2 Sprint Tasks 39](#_Toc196671067)

[7.8.3 Summary of Actions 39](#_Toc196671068)

[7.8- Level 2 Sprint 8 40](#_Toc196671069)

[7.8.1 Overview 40](#_Toc196671070)

[7.8.2 Sprint Tasks 41](#_Toc196671071)

[7.8.3 Summary of Actions 41](#_Toc196671072)

[7.9- Minimum Awesome Product - Sprint 9None of this is perfect 42](#_Toc196671073)

[7.9.1 Overview 42](#_Toc196671074)

[7. 9.2 Sprint Tasks 42](#_Toc196671075)

[7.9.3 Summary of Actions 43](#_Toc196671076)

[**8 - End-project report** 44](#_Toc196671077)

[**9 - Postmortem** 45](#_Toc196671078)

[9.1 Reflection 45](#_Toc196671079)

[9.2 Data 45](#_Toc196671080)

[9.3 Conclusion 45](#_Toc196671081)

[**10 - References** 47](#_Toc196671082)

# **Word Count**: 10494

# **Code Links**: https://github.com/Mdot5596/Timeline-Takedown

# **1 - Introduction**

## 1.1 Overview

Timeline Takedown is a video game developed for PC, with an initial release planned for Itch.io, and potential future release on Steam. The game was created using Unity Editor version 2022.3.18f1, with all scripting written in C#. Inkscape was used to design the game’s user interface elements.

The gameplay takes place across two distinct levels, each featuring a unique theme to provide variety and maintain player engagement. In each level, the player must survive five escalating waves of enemies before facing a final boss encounter in round five. Defeating the boss rewards the player with a timepiece part, which is required to progress to the next level. This core gameplay loop is designed to offer a structured yet intense survival experience that encourages strategic play and replayability.

## 1.2 Audience

Timeline Takedown is designed to fall under the PEGI 12 age rating, as it features “violence in a fantasy setting or non-realistic violence” (PEGI, 2017). The visual style and gameplay avoid graphic content, making it suitable for younger audiences compared to more intense survival shooters. Despite its simplified tone, the game retains the core appeal of the round-based survival genre, offering an engaging and fast-paced experience that is accessible to a wide range of players.

The game fits within the round-based survival shooter market, alongside titles such as Call of Duty Zombies (Treyarch, 2008), Killing Floor (Tripwire Interactive, 2009), and similar wave-based experiences. However, unlike these examples which typically target an audience of males aged 18 to 34 (Anon, 2023), often due to more mature themes and realistic violence, Timeline Takedown is intended to reach a slightly broader and younger demographic. Its simpler mechanics and less intense tone are designed to be more approachable, reducing the learning curve while maintaining the genre’s core excitement.

## 1.3 Purpose

The primary goal of Timeline Takedown was to develop a game that could be enjoyed by a wide range of players, particularly those who enjoy round based survival games. This project’s objective was to create something unique within the genre, blending traditional survival mechanics with engaging map exploration and narrative elements. The game was designed for enjoyment rather than financial gains as I wanted to produce a game that I would personally find fun and rewarding to play.

Timeline Takedown is about players surviving waves of enemies while exploring different timelines through well-constructed level design. The game incorporates storytelling elements, providing context about the story throughout the introduction and concluding with an end game sequence. While survival is at the heart of the game, the inclusion of narrative sections adds that layer of immersion, allowing players to have a sense of the story behind the gameplay.

# **2 - Background, objectives & deliverables**

## 2.1 Project Background

Timeline Takedown is a round-based survival shooter, these types of games have become a widely recognised and an engaging subgenre within the wider category of first-person shooters. These games typically challenge the player to survive increasingly difficult waves of enemies, the player tends to play strategically, use map knowledge to their advantage, and ration their resources in order to survive. One of the most iconic examples in this genre is the Call of Duty Zombies mode that was originally introduced as a bonus reward for completing the main games campaign in Call of Duty: World at War (Treyarch, 2008). This example was one of the first instances that popularised the round-based survival genre that included fast paced combat, puzzle like map design, hidden easter eggs, and wave-based progression.

This genre is known for its high replayability value, and its ability to create intense moments of gameplay. Players are placed in intricately crafted level, facing off waves of AI enemies that increase in both number and difficulty with each round.

This project explores the design and development of a custom round – based survival game that is created in Unity. Timeline Takedown and my thought process behind certain mechanics in this game take direct inspiration from the mechanics and pacing of Call of Duty Zombies. The goal is to develop a game that captures the intensity and satisfaction of surviving waves of enemies, while also exploring exciting locations through custom made levels and enemies.

As part of this project two fully playable maps have been created, each offering unique layout, theme, weapons, and enemies. These maps aim to test different survival strategies and enhance repalyabilty as they all have completely different layouts, for example one is inside a tight cornered hospital and another is outside in the sandy climate of Egypt. This project also investigates how wave escalation, enemy AI , and environmental design contribute to overall player engagement in round-based survival games.

## 2.2 Market / Competitors

Waved-based gameplay is a core mechanic used in many successful titles. The following table compares several games that utilise this system, highlighting their unique features, supported platforms, and player feedback to help inform future design considerations.

|  |  |  |  |
| --- | --- | --- | --- |
| Game Title | Features | Platform | Reviews (Customer Feedback) |
| Call Of Duty Zombies | * “Easter Eggs” – Objectives hidden in maps requiring puzzle solving * Custom maps * Perk system * Progressive weapon upgrades * Large AAA team behind development | PC, Xbox, PlayStation, limited Nintendo products | Generally positive – praised for addictive gameplay, creativity, and replay value. However, more recent titles have shown a lack of player engagement due to poor choices by the developers such as replacing beloved voice actors with AI and using AI generated art (Yin-Poole, 2025). |
| Killing Floor | * Co – op survival horror * Unique perk system * Boss fights * Variety of enemies and weapons | PC, Xbox, PlayStation | Positive – loved for teamwork focus, fun to play with friends. But some say it can feel repetitive. |
| Left 4 dead | * AI Director dynamically changes the gameplay * Co-op story mode * Different infected types * Fast-paced wave combat | PC, Xbox | Very Positive – acclaimed for co-op fun, replayability, and modding support. |
| Sker Ritual | * Round based survival horror * Co-op (1-4 Players) * Supernatural enemies * Weapon upgrades and unique abilities * “Easter Eggs” – Objectives hidden in maps requiring | PC, PlayStation, Xbox | Mostly Positive – praised for atmosphere and gameplay loop, though some desire more content. |
| Deep Rock Galactic | * Co-op wave survival * Objective based missions * Class system * Procedurally generated caves | PC, Xbox, PlayStation | Very Positive – praised for teamwork mechanics, humour, and replayability. |

*Figure 1: Competitor Analysis*

## 2.3 Objectives & Deliverables

A Game Design Document (GDD) was created for Timeline Takedown and can be found in Appendix 1. During the development of this document, a list of key deliverables was outlined and organised using a priority-based system. This helped structure the development process by identifying which features and assets were essential for the core gameplay experience, and which could be considered lower priority or stretch goals, depending on the time available.

Must Have :

o Functional wave-based combat system with enemies and bosses

o Historical themed levels with unique enemies and environments

o Basic UI – Health display, ammo, wave counter, etc)

o Player movement and combat mechanics( movement, shooting, enemy targeting)

o Basic level progression with the time machine parts and boss fights

Should Have :

o Multiple difficulty settings( easy/normal/hard)

o Weapon variety (different weapons)

o Power Ups

o Sound Design that matches the map theme

Could Have:

o 2 player CO-OP

o Boss-Specific mechanics that vary depending on location (special attacks)

o Cosmetic customisation for player model and weapons

o Player level progression

Want to Have:

o More than 3 or levels, possibly up to 10

o Online support for friends to play over the internet

o Hidden challenges/missions (easter-eggs)

Although not being stated in the GDD directly, having an itch.io page built for the game and uploading a working .exe on the itch page was also a “must have” requirement.

# **3 - Ludology**

## 3.1 Introduction

Round based survival games focus on the player surviving waves of enemies. Each game interprets this genre differently. Some challenge the player to simply survive as long as possible with no objective beyond that, while others combine survival with challenges or objectives to progress through.

These types of games remain highly engaging due to their replayability. Even though the core gameplay remains the same each time, the players experiences is different with each game. In round-based survival games, there are many different aspects a player can choose to focus on, such as how long they can survive, how many kills they can achieve with a certain weapon, or how quickly they can complete a challenge. This variety of self-imposed goals is what keeps players coming back, providing a fresh experience with each playthrough.

## 3.2 Key Examples

#### 3.2.1 Call of Duty225

First introduced in Call of Duty: World at War (Treyarch, 2008), Zombies mode became an instant fan favourite. This game mode started off as a side project that was made by a few bored game developers in their free time, this was then implemented as a bonus reward for completing the main game’s campaign. *This was a secret game. A mini-mode. An Easter Egg - whatever you want to call it - crafted from the passion and creativity of a team with little free time* – (Andy Hartup, 2016) What began as an experiment soon evolved into one of the major reasons players purchased Call of Duty titles each year.

In this game mode players are placed in a unique map where endless waves of zombies grow progressively stronger, faster, and more aggressive with each round. There are weapon upgrades, map unlocks, buildable items, and quests (known as easter eggs) that players can complete within different maps.

What makes Call of Duty’s Zombies survival system effective is the tight balance between player power and enemy difficulty. Players must constantly manage resources like ammunition and points while making strategic decisions about when to unlock new areas or upgrade their existing equipment. For example, a player must weigh the risks of spending points on the randomised “Mystery Box” and potentially receiving an awful weapon, against purchasing a wall buy weapon that is a guaranteed weapon of the players choice. These choices that make the player think quick under pressure and cause constant tension to help the game maintain a strong engagement loop.

#### 3.2.2 Killing Floor 2

Killing floor 2 (Tripwire Interactive, 2016) is a wave-based survival experience where players go up against waves of increasingly dangerous mutant creatures that results in a boss fight at the end. Unlike call of duty zombies, killing floor 2 introduces a class-based system where players can pick specific roles and weapon loadouts like medic and support. These roles alter the playstyle of the entire game as each class will have separate benefits and drawbacks, encouraging the player to take their time to plan out how they will play.

The wave system works by scaling not just the enemy numbers but also the spawn rates, damage output, and the type of bosses that appear. Between waves players can buy weapons and upgrades using in came currency earnt the within the rounds. What makes killing floor 2 fun is the pacing: fast , brutal combat combined with short preparation periods that keeps the players on edge. However, a weakness in this game is that without a full team solo players often find higher difficulties punishing as there is an ongoing issue with enemy scaling.

## 3.3 Application

Timeline Takedown took inspiration from mechanics that were present within this review. I wanted to capture the same sense of increasing pressure over time, as well as give players time to breathe and plan between the rounds.

In Timeline Takedown the wave system scales progressively, each new wave increases the number of enemies spawned, and the final wave includes a boss enemy. By controlling the enemy numbers and behaviour I was able to create a natural difficulty curve without needing overly complex systems.

At this stage in development, direct rewards between rounds were limited, but powerups were placed within the environment to help the player recover before the next wave began. Additionally, a short break was implemented between waves, where enemy spawning pauses for a few seconds. This system allows the player time to reload safely, reposition, and search the map for powerups. These breathing periods were important for maintaining the games pacing and giving the player a sense of preparation before the next wave, this feature was inspired by the grace period that Killing Floor 2 gives players.

In the future expansions of the project, I aim to add additional rewards such as a weapon upgrade system like the “Pack A Punch” system from Call of Duty. This allows players to invest points or resources to significantly upgrade their weapons. This would give players more meaningful choices and strategic options as the difficulty increases.

# **4 - Method of Approach**

## 4.1 Methodologies

Throughout the development of Timeline Takedown, I followed a flexible and iterative approach that allowed me to respond to challenges and make improvements as the project evolved. While not following a strict software development methodology, my workflow was heavily inspired by Agile principles, particularly the focus on breaking the project down into smaller tasks and continuously building on each version of the game.

I chose Agile because it was the best fit for the nature of game development, where creativity and adaptability contribute significant importance to the development of the game. This methodology allowed for significant creative freedom, enabling me to implement changes as I worked, and structure my workflow into manageable chunks. To support this approach, I used Trello (Trello, 2025) as my task management tool. Trello enabled me to visually organise the different aspects of the project, such as sprint planning, deadlines, feature tracking, and bug lists.

I used a Kanban-style task management system through Trello to keep track of my backlog, active tasks, and completed work. This helped me stay organised and prioritise tasks based on what was most important or time-sensitive at each stage.

I didn’t follow traditional test-driven development at the start of the project, but I regularly played through the game during development to identify bugs, balancing issues, and areas for improvement. I also made use of runtime parameter adjustment in Unity (e.g., for enemy speed, wave timing, etc.), which helped me iterate quickly without rebuilding the game constantly.

Alongside this, I recorded development logs and kept notes throughout the project to reflect on progress and decisions. These proved useful for tracking what had been achieved, what needed reworking, and where time was being spent.

## 4.2 Limitations

While Agile proved to be a highly effective approach for my project, it was not without its limitations. One challenge I encountered was related to long-term planning and how quickly priorities could shift mid development. For example, Level 2 was originally intended to be an underwater stage, but after struggling with the complexity of implementing swimming mechanics, which would have required a significant amount of time and technical problem-solving, I made the decision to pivot. Instead, I designed an abandoned hospital level, which allowed me to focus on gameplay rather than mechanics that were beyond the scope of a solo project.

This shift is a good example of Agiles flexibility but also highlights a downside: some ideas had to be cut or changed quickly, and not all time spent on the original concept translated into the final product. In terms of task prioritisation, I did well in focusing on core mechanics first before diving into level design. However, one thing I would do differently is prioritising animations earlier in development. This aspect of the game took longer than expected, and I continued to run into issues with animation syncing and quality even late into the project timeline. If I had addressed those earlier, it could have smoothed out a lot of later stages and testing.

Despite these limitations, the benefits of using an Agile-inspired workflow - particularly the ability to pivot quickly and iterate based on feasibility were crucial in managing the unpredictable nature of game development.

### 4.3 Alternatives

When researching different development methodologies, I found several alternatives that had potential to be the right fit such as Waterfall, Scrum, and Lean Development. Each approach had its own strengths that could have aligned with certain aspects of Timeline Takedown.

Waterfall was considered due to its structured, sequential nature with clearly defined stages such as planning, design, implementation, testing and deployment. This method can be beneficial for projects with fixed requirements and a predictable outcome. However, game development is rarely that linear. Creative features, gameplay mechanics, and balancing often need ongoing adjustment, and Waterfall’s rigidity could have slowed down the process or required major overhauls after key stages were completed.

Scrum, a more structured subset of Agile, was also a strong contender. Its emphasis on regular sprints, stand-ups , and reviews helps push for consistent progress and team accountability. If this was a team project than Scrum might have been ideal as it provides a clear communication framework and keeps all members busy through frequent check ins. However, as a solo developer, maintaining all the formal Scrum ceremonies would have added unnecessary overhead without much added value.

While each one had their own potentials I ultimately chose to stick with a flexible Agile-inspired approach. It gave me the best of all worlds: structure without rigidity, room for creativity, and the ability to adapt quickly to changes or new ideas. My familiarity with Agile also meant I could dive into development without a steep learning curve.

On top of that, industry professionals frequently recommend Agile for game development due to its iterative structure, feedback-driven loops, and suitability for handling the unpredictable nature of game design and mechanics. For instance, Clinton Keith, a veteran game developer and author of Agile Game Development with Scrum, advocates for Agile methodologies, stating that they can help teams deliver games more efficiently, rapidly, and cost-effectively, while also enhancing the development experience for team members (Clinton Keith, 2020) . This made Agile the most natural and effective choice for me.

## 4.4 Technologies

In order to bring Timeline Takedown to life, I relied on a range of technologies that supported different aspects of the game’s development process, from planning and design to version control. Below is a breakdown of the key software and tools used throughout the project, along with the reasoning behind each choice. These technologies were selected based on their compatibility with my workflow, accessibility, and suitability for solo development.

* **Unity Engine**

The game was developed in Unity Engine (version 2022.3.18f1), which was chosen due to its flexibility, familiarity, and strong support for 3D games development. Unity also offers a large range of built in systems to make development slightly easier, such as physics, animations, UI.

* **Programming Language: C#**

All gameplay scripting was done in C# as its Unity’s primary supported language, and the most commonly used within Unity. It was used to implement core systems such as enemy AI, wave spawner mechanics, player interactions, and UI functionality.

* **UI Design: Inkscape**

Inkscape was used to create the majority of the UI seen in this game, drafts were created inside of Inkscape and then iterated into the final version, as seen in Appendix 2. Inkscape was chosen as I have had a lot of prior experience with this software, and I knew what both I and the software is capable of achieving. It is also free and offers a range of useful features such as image size scaling, easy exporting to any file type, and a lot more.

* **Development Logs (Devlogs)**

Development logs – or commonly known as Devlogs, were recorded at the end of every 2 sprints. They were used as a method to track progress, record key milestones, and upload onto YouTube for viewers to watch. Links to each devlog can be found in Appendix 3.

* **Version Control**

Version Control was managed by using GitHub and GitHub Desktop, allowing for daily commits and backup, and allowing for easy tracking during development as I also added a description to most commits. GitHub desktop was mostly used due to its simplicity and user-friendly interface.

* **Map Designs**

Initial map designs were sketched on paper to plan out the layout, player flow, and enemy spawn points before being built in Unity. These paper designs were then refined using the online tool Dungeon Scrawl (app.dungeonscrawl.com, n.d.), which allowed for a more visual and structured digital representation of each level. These map designs can be seen in Appendix 4.

Each tool and technology listed above played a vital role in shaping Timeline Takedown. Choosing tools that I was familiar with such as Inkscape and GitHub Desktop – while still pushing myself to explore new techniques such as dungeon scrawl and paper-based designs – allowed me to work efficiently as a solo developer. I considered including a full breakdown of hardware and software used but I have decided to instead include that in the appendix to keep the main section focused on the tools directly involved in development, if you wish to see the full breakdown of tools, please refer to APENDIX 5. Overall, the combination of these technologies supported a smooth workflow and allowed me to bring the vision for the game to life.

## 4.5 Project Management Approach

Throughout the development of Timeline Takedown, I used a flexible and iterative project management approach inspired by the Agile methodology. While I did not follow a strict formal Agile framework such as Scrum, I applied many of its core principles such as breaking the work into smaller manageable tasks, working in sprints, and frequently reflecting on progress throughout the use of development logs and meetings. This approach suited the nature of game development and gave me the creative freedom to adapt, iterate and evolve the game design as needed.

### 4.6 Planning and Structuring the Project

The project began with the creation of a comprehensive Game Design Document (GDD), which acted as the foundation for the games vision. This document included early design ideas, gameplay mechanics, art style reference, and technical requirements. I also researched into other games that I took inspiration from and discussed what I would like to implement from these games into mine, such as round based mechanics from call of duty zombies, and inspiration on how The Simpsons game did level and character themes. The full GDD can be found in Appendix 1.

Development was divided into bi-weekly sprints, each with focused goals such as implementing a specific feature, completing level layout, and fixing bugs. I used Trello to organise and manage these sprints through a Kanban style board, with columns for “To Do”, “In Progress”, “Completed”, and additional columns for side notes such as “Useful Resources” and “Important Dates”. This helped me prioritise tasks and track development progress visually.

Image of Trello

Figure 2: Trello

As a method of monitoring progress and staying accountable - and a requirement of this project, I attended bi-weekly meetings with my supervisor. At these meetings I presented updates, discussed issues I was facing, and received feedback. These sessions were essential for keeping the project on track as in some situations such as when I had an issue with the animation and the models’ feet clipping under the ground , I was given a resource that would prove to fix this problem. Without these meetings the project would have moved a lot slower as I would have got stuck on issues and development would have slowed down as a result.

### 4.7 Visal Planning Tools

During the early stages, I created paper-based diagrams for both map layouts and UI mock-ups, allowing for me to experiment freely with ideas before committing to digital formats. These designs were later recreated using Dungeon Scrawl – an online tool for map creation - to create clean, digital versions of the level layouts. This tool made it easier to visualise flow, structure and enemy spawn point placement before building the levels inside Unity. If you would like to view this paper-based diagrams, please refer to Appendix 6.

### 4.8 Tracking Progress & Reflecting

At the end of each sprint, I recorded and uploaded Development logs to YouTube. These served as development checkpoints and allowed me to reflect on progress, document challenges, explain key decisions and talk about what I will be working on for the upcoming sprint. Not only were they useful for personal tracking, but they also allowed for public visibility on the projects evolution and can be referred back to through my career in games development. A link to the Devlog playlist that includes all the videos created can be found in Appendix 3

I maintained a balance between feature development, testing, and visual polish by continually adjusting the Trello board and reviewing my GDD and other design documents I created. For example, when I realised the underwater level was too complex due to the technical demands of swimming mechanics and animation, I changed to a hospital themed level, which allowed me to reuse existing gameplay systems more effectively. This flexibility was one of the major strengths of my Agile workflow.

# **5 – Legal, Social, Ethical and Professional issues**

Throughout the development of this project, I had to constantly consider legal, social, ethical and professional issues to make sure that I was developing my project in a professional and responsible manner.

## 5.1 Legal

Legally, the most important consideration for me was checking that the assets I used in my game were safe and compliant as almost all the game would be made from online resources, except the building models. When building my asset list as found in Appendix 7, I made sure that all assets had either public domain licenses or clear usage rights. This applied not just to 3D models but also tools, audio clips, and animations. Since I used a wide variety of audio and animations throughout the project, I made sure to source them from reliable websites like Maximo and royalty free audio sites.

I also experimented with AI generated artwork for loading screens and various UI elements using DeepAI. However, before doing this I reviewed the licensing terms which confirmed that the content could be used commercially and legally within my project.

## 5.2 Social

Socially I was aware that including war themes and enemies into Timeline Takedown could be a sensitive area. A well-known example is Call of Duty World at War, which originally included Nazi memorabilia like swastikas. Due to public backlash and the risk of causing offense, these were later removed and replaced in the new remastered version of the game Zombie Chronicles with other images as shown in figure 3. Keeping this in mind, I was careful when sourcing World War 2 era solider assets for level 2. I made sure the models I used didn’t feature any explicit or offensive symbols. This made sure that I maintained a respectful and inclusive approach while still delivering the intended theme.





*Figure 3: Changes between same maps after receiving backlash for nazi symbols*

## 5.3 Ethical

Ethically, I ensured all playtesting remained completely anonymous to conform to the University’s Ethics policy, this can be viewed in Appendix 8. No personal data was asked for or collected from testers. One exception was when I used an image of a classmate for playtesting for promotional purposes however, he did give me consent for this. To still respect his privacy and keep him anonymous, his face was not visible, and I did not name him.

Additionally, I took care not to use any plagiarised material within this project. All assets, code, audio, and animations were sourced from legitimate platforms with appropriate licensing, and I avoided using any work from other developers without clear permission or usage rights.

## 5.4 Professional

Professionalism was consistently maintained throughout the development of this project. Every scheduled supervisor meeting was attended, and a good amount of work was completed on a weekly basis throughout the entirety of this project, as shown in my GitHub Graph and commit logs in figure 4. Supervisor feedback was actively listened to and implemented, for example suggestions such as using AI generated artwork for early-stage prototyping was acted upon. This demonstrated strong collaboration skills and the ability to adapt based on professional advice.

A screenshot of a calendar

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 4: Github Graph and commit logs*

I demonstrated further professionalism throughout, with careful management of project documentation. I frequently committed to GitHub with detailed commit descriptions to keep track the changes made to the project, making sure that version control was properly upheld.

Detailed devlogs were also recorded after each sprint, documenting progress, issues, and next steps. These practices not only maintained high professional standards but also ensured the project remained well organised throughout its development lifecycle.

# **6 - Project Management Tools**

## 6.1 Project Management

The main method of project management was throughout the use of Trello, this tracked all the sprints within the creation of Timeline Takedown. I followed the Kanban workflow with separate columns for to do, doing, and done I feel like more in this gap can be done.

It included other information than just sprints records, such as links to resources, dates, useful information and more as show in this screenshot. Full Trello board images can be found in Appendix 9.

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 6: Trello Board*

## 6.2 Version Control

GitHub Desktop was used as the version control tool during this project. The desktop client was chosen due to its familiarity, and intuitive user interface which made it quicker and more accessible to manage commits compared to using GitHub through the command line.

Because I committed frequently as I was aiming to log every change made during development to efficiently track progress – using the command line version would have been far more time consuming as it would have required manually typing commands for each commit. With GitHub Desktop, this process was streamlined to just a few clicks.

## 6.3 Meetings

Meetings were used as an important project management tool throughout the development of Timeline Takedown. They provided structured checkpoints where the supervisor could track the overall progress and offer feedback or new recommendations to improve the quality of the project.

These meetings were held every other week in the Smeaton building at the University of Plymouth, typically lasting between 10-15- minutes.

## 6.4 Development Logs

During each sprint, a development log (Devlog) was recorded. These videos served as a project management tool to track progress, document any issues encountered during the sprint, and outline plans and ideas for the upcoming sprint. This approach was highly efficient as the recordings could be reviewed at a later date to quickly recap key developments, challenges, and decisions made throughout the project’s lifecycle. All devlogs can be found at Appendix 3.

## 6.5 Testing Methods

Testing was conducted using three different approaches throughout development. Firstly, self-testing took place during feature implementation. As soon as a new mechanic or functionality was added, I would immediately test it to ensure it worked as intended. This allowed for rapid feedback and adjustments while still in the development phase. For example, when setting the movement speed for enemies, I would quickly playtest to check if the speed felt appropriate. If it did not meet expectations, I would adjust the value and retest until satisfied. The same approach was used when deciding on enemy spawn locations – if the flow of enemies through the map felt unbalanced or unnatural, I could easily reposition the spawn points and test again in real time.

Sprint testing was conducted at the end of each two-week sprint, these sessions focused on the features that had been developed during that sprint. I referred to my Trello board, which listed each completed task, and systematically tested each one. Features would be ticked off the board once they were fully tested and verified. This ensured that no new functionality was forgotten or broken before recording the developer log for that sprint.

Lastly, usability testing was carried out to gather external user feedback. I hosted multiple in person playtesting sessions where volunteers could play the game either at my computer or in the Smeaton building at the University of Plymouth. This method was important as testers often noticed issues I had overlooked. For example, one tester discovered a floating model left above the map that I had completely forgotten about. All usability tests were conducted anonymously, with no personal data collected, in accordance with university guidelines.

In conclusion, testing was a crucial part of the project’s success. It helped uncover bugs, refine gameplay mechanics, and gather real human feedback, which led to important improvements. Without consistent and varied testing methods, many issues would have gone unnoticed, and the overall quality of the game would have been significantly lower.

See Appendix 10 for more information on the testing process, including examples of the testing process such as questions asked, and feedback received.

# **7 – Implementation**

## 7.1 Project Setup and Design - Sprint 1

### 7.1.1 Overview

This was the first sprint of the project, after attending the required seminar, I chose the type of software project I wanted to develop – that being a game and selected an assistant supervisor to support the project. The choice of creating a game came easy to me as I aspire to work in the games development industry after graduating, with this project complete it would stand as a solid portfolio piece I can show to employers and display on my portfolio website (morganhodge.co.uk). This project would also strengthen my skills as a game developer, through project management , good use of version control, and 3D Unity skills.

During this first sprint I focused on outlining the initial concept, goals, and setting up my workspace. Following guidance from the seminar, I began drafting the Game Design Document(GDD), which would act as a foundational reference throughout the development process.

### 7.1.2 Sprint Tasks

* Set Up Work Environment
  + Create Github Repository
  + Set up Github Desktop
  + Create new Trello board and give access to supervisor
  + Gather all required information such as briefs and required documentation and add them to the repository
* Level Design (Paper Based)
  + Level 1
  + Level 2
* Basic Player Movement
* First Person Camera Movement (Mouse)
* First Person Walking Movement (Keyboard)
* Test Scene Created

### Summary Of Actions

The player movement created for this first sprint was required as I wouldn’t be able to proceed with the development or test anything I would be implementing if I could not move around the scene.

The code implementation shown in Figure 7 demonstrates how player movement was set up. One of the strengths of this system is its flexibility, key parameters such as movement speed, jump force, and jump cooldown are easily adjustable. This allowed for efficient playtesting and balancing throughout development, without the need to rewrite core logic.

A screen shot of a computer program

AI-generated content may be incorrect. A screenshot of a computer

AI-generated content may be incorrect.

*Figure 7: Player Movement Code*

Additionally, the paper-based level designs were created in this sprint but were not displayed until the Devlog 1.

## 7.2 Research and Game states - Sprint 2

### 7.2.1 Overview

This second sprint mostly consisted of setting up game states such as a Start Screen, Pause menu and settings tab.

Research was also conducted in this sprint, exploring art styles I could use as well as potential assets for the enemy type – this was documented within the Devlog 1.

During my research into art styles and visual design within video games, I was recommended several books that would be relevant to game development. As a fan of the franchise they are based on, I was eager to purchase these books and learn from them. The books I purchased during this sprint and studied included:

* The Art of Fallout 4 – A detailed look at the environment and character design choices made in fallout 4 , this was useful reference material for post-apocalyptic and sci-fi aesthetics.
* Elden Ring Art Book Vol1 – This book offered insight to the world building , creature design and atmospheric art direction that was used in Elden Ring, this helped me understand how to setup an atmosphere in a level using certain colours and audio.
* Blood Sweat and Pixels by Jason Schreier – This is not an art book but was highly recommended within the game’s development community. This was an insightful read as it provided a look on how games are really constructed and the struggles and triumphs of game development across multiple different studios. This was a strong motivational resource during my own development.

### 7.2.2 Sprint Tasks

* Devlog 1 created – <https://www.youtube.com/watch?v=Su4Demj-MFw>
* Paper Based Concept Art
* Created basic UI on Inkscape
* Game states created Menu-Start-Pause
* Movement tweaking

### 7.2.3 Summary of Actions

Concept art was sketched up, this was the result:

A paper with drawings and words

AI-generated content may be incorrect.

*Figure 8: Concept Art*

The UI elements were initially created using Inkscape. I approached this stage knowing that these assets would just act as placeholder visuals, rather than the final design. The goal at this stage was to have functional UI in place for testing gameplay mechanics and user interactions. An example of this early placeholder Art can be found below in figure 9.

A screenshot of a video game

AI-generated content may be incorrect.

*Figure 9: Basic UI created in Inkscape*

When researching into how to create the different game states (such as main menu, pause, game over) I thought I would have to create separate Unity scene for each state and manage them using scene indexes in the build settings. However, through further research and advice from online resources, I learnt that using separate canvas objects with a single scene was a much more efficient approach. This method allowed for better performance, 0 loading time, easier UI management and smoother transitions between game states.

## 7.3 Assets and Online Resources - Sprint 3

### 7.3.1 Overview

This sprint primarily focused on researching suitable online assets and exploring AI tools to generate temporary placeholder art. I aimed to find assets that matched the intended visual style and tone of the game while also making sure they were suitable for implementation into Unity.

During this stage, I created a test scene where I imported and arranged some of the selected assets as this allowed me to visually evaluate how well they fit and the overall aesthetic.

### 7.3.2 Sprint Tasks

* Devlog 2 created – https://www.youtube.com/watch?v=RXlF7QE14os
* Created Document of assets
* Created a test level and imported certain assets
* Used tools to generate UI and Loading Screens
* Started to create a block out level for game mechanics to be tested

### 7.3.3 Summary of Actions

When creating the document of assets I would be potentially using, I focused on finding free, copy safe assets that could be used within the game. To keep track of the resources I found I created a word document compiling the names and download links of each asset. A copy of this assets list can be found in Appendix 7.

During a recent supervisor meeting I was recommended an AI tool called DeepAI, which is capable of generating images for free. Following this advice, I used DeepAI to produce placeholder art assets for the game. Since developing fine visuals was not a top priority at this stage, using AI generated assets allowed me to maintain visual consistency while focusing on gameplay functionality.

Lastly, I downloaded a selection of the most appropriate models from the asset list I had compiled. These assets were then imported into a test scene in Unity, allowing me to evaluate their visual compatibility with the games intended style and theme. This hands-on assessment provided immediate visual feedback, helping me decide whether each asset was suitable for continued use or needed to be replaced or modified.

A screenshot of a computer game

AI-generated content may be incorrect.

*Figure 10 : A screenshot from my devlog displaying the models imported*

## 7.4 Enemy & Main Game Loop - Sprint 4

### 7.4.1 Overview

At this stage of development I was working on the games functionality you could now end the game by picking up “end game” part. Enemies are now implemented, they have the ability to locate and follow the player, and adjustable parameters like speed. Additionally, the player now has a health system, allowing them to take damage, die, and regenerate health.

### 7.4.2 Sprint Tasks

* Devlog 3 – https://www.youtube.com/watch?v=vxEHmGPZ-y4
* Health Functionality
* Enemy Controller
* Level can be completed
* UI
* Map Design on Dungeon Scrawl

### 7.4.3 Summary of Actions

To enable enemy pathfinding and movement I had to first bake a NavMesh onto the ground surface within the scene. This allowed the enemies to navigate the environment using Unity’s built in NavMesh system.

I then create a script that uses Unity’s NavMeshAgent component to control the enemy’s movement. The script constantly updates the enemy’s destination to follow the player’s position. To avoid potential runtime errors, I added a check that checks to see if the enemy is still alive, that the NavMeshAgent is still active, and that it remains on a valid section of the NavMesh before attempting to move.

A screen shot of a computer program

AI-generated content may be incorrect.

*Figure 11: Enemy Controller Script*

During this sprint, the players health system was implemented, allowing the player to take damage and heal. If the players health bar reaches zero, the game ends. However, enemies are not yet capable of dealing damage. To test this functionality of the health system, I created a test object that when interacted with would damage and heal the player. This was used as a temporary method to test the health mechanics until enemy interactions could be fully integrated.

Lastly, I transferred the paper-based level designs to digital format using a tool called Dungeon Scrawl, this was recommended by my supervisor. This was beneficial as it provided a clearer visual representation of the levels, and this was crucial when trying to translate designs into Unity.

A white paper with writing on it

AI-generated content may be incorrect.A screenshot of a game

AI-generated content may be incorrect.

*Figure 12: Transformation of paper design(top) to digital(bottom)*

## 7.5 Minimum Viable Product - Sprint 5

### 7.5.1 Overview

At this stage, the game had reached its Minimum Viable Product (MVP) phase, meaning it had the core functionality required to be considered a working project. The main game loop was fully implemented, including the wave manager, which handles enemy spawning and progression. Projectile mechanics were set up, allowing the player to attack enemies. Enemies were now able to both take and deal damage and were set to de-spawn upon death as animations had not been implemented yet. Additionally, the wave system was functioning correctly, with new waves only starting after the appropriate number of enemies had been defeated in the current round.

### 7.5.2 Sprint Tasks

* Early Stage of Wave manager
* Projectiles
* Enemy Spawn Points
* Enemies can take damage
* Start of Power Up implementation

### 7.5.3 Summary of Actions

The wave manager script is a central part of the games round based survival system, it controls the flow of enemy waves, tracking the progress of the round through the use of Unity Text Mesh Pro UI, and manages the boss spawning.

The script begins by checking the current wave number and uses a predefined array called enemiesPerWave to determine how many enemies should be spawned for each wave. If the wave is marked as a boss wave, a boss enemy is also spawned. To prevent duplicate spawns, a bossSpawned flag ensures the boss is only spawned once per wave.

Each time an enemy is defeated, the method EnemyDefeated() is called which increments the kill count, but this may be removed by the final release as this is mostly used as a debugging tool and not a final game feature. Also, the number of enemies remaining in the current wave is decreased and the UI is updated to reflect this.

A screen shot of a computer program

AI-generated content may be incorrect.

*Figure 13: Enemy Defeated Method*

When all enemies in a wave are defeated, the game waits a few seconds using WaitForNextWave() before transitioning to the next wave using the StartNewWave() func. Once the final wave is complete, the script plays a victory audio que and stops any more waves being spawned.

A screen shot of a computer program

AI-generated content may be incorrect.

*Figure 14*

The enemy spawner is responsible for spawning enemies and bosses during gameplay, based on wave progression that is integrated within the wave manager. It allows me to set spawn points and enemy prefabs through the Unity inspector, allowing me to have full control and flexibility over how and where the enemies appear.

When a new wave starts the StartWave() method is called, which begins a coroutine SpawnWave() that instantiates a set number of enemies at randomised spawn points. A delay between spawns is needed or multiple enemies can spawn at the same spawn point , causing issues such as overlapping or leading to the enemies damaging each other.

A screen shot of a computer program

AI-generated content may be incorrect.

*Figure 15: Enemy Spawner Script*

Whenever an enemy is defeated, EnemyDefeated() is called, which passes that information back to the wave manager to update the wave progression and necessary UI elements.

## 7.6 Animation and Aesthetics - Sprint 6

### 7.6.1 Overview

This sprint was dedicated mostly to the visual elements of level 1, as of last sprint I had completed the base game loop, now it was time to start making the game look aesthetically pleasing. Towards the end of the second week of the sprint a private play test was conducted, this was mostly for feedback on the new visuals.

### 7.6.2 Sprint Tasks

* Devlog 4 - https://www.youtube.com/watch?v=lJwd-J0-sTs
* Enemy Animations
* Working on Level 1 Aesthetics/map layout
* Ambience sound script
* PowerUps

### 7.6.3 Summary of Actions

#### Animation implementation

At this stage the enemies had no walking or death animation, this caused them to appear as if they were hovering towards the player rather than moving realistically. As I had no prior experience with Unity’s Animator tool, this challenge proved to be a big learning curve for me. However, after watching several online tutorials and receiving guidance from lecturer Tyler Cheng, I was able to successfully integrate animations into the enemy model. This greatly improved the games visual flow and made enemy movement feel more natural and immersive.

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 16: Enemy Animator*

#### Ambience sound

When conducting research for this project, I learned that a lot of games in this category utilise ambience sound. What this means is that when the player moves to different locations the audio changes seamlessly with them, it can add a more immersive feel to the game.

As a developer who is creating a game within the same category, I wanted to introduce this mechanic into my game. I did this by researching online into how you can create ambience sound in your game as I knew it was more advanced than simply adding an audio source into the game scene.

What I had to do was create a script that continuously updates the position of an ambient sound source based on the players location. Specifically, the script calculates the closest point on a defined collider area to the players position using Collider.ClosestPoint(). This value is then used to reposition the sound emitting GameObject to that point in every frame. This makes the audio appear to be coming from the environment rather than following the player directly. I could add multiple box colliders on the scene, each acting as their own sound area, allowing for smooth audio transitions when going to different locations such as inside the pyramid has an eerier audio compared to the Egyptian music outside.

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 17: Ambience Sound in the Unity inspector*

A screen shot of a computer program

AI-generated content may be incorrect.

*Figure 18: Ambience sound script*

## 7.7 Poster & Level 2 - Sprint 7

### 7.7.1 Overview

This sprint primarily focused on creating the poster for the game, I wanted to create something that I can use online to promote the game, so I spent a lot of time on it. Also, during this sprint, I worked on polishing parts of level 1 such as creating different weapons, as well as bullets. Level 2 was also planned.

### 7.7.2 Sprint Tasks

* Poster
* Level 2 Design
* Gamestates
* Polishing of Level 1

### 7.7.3 Summary of Actions

#### Poster

All versions of the poster can be found in appendix 11.

#### Changes for level 2

Level 2 was originally planned to be an underwater shipwreck environment featuring swimming mechanics. However, during this sprint I came to the realisation that this was not a feasible idea as I had a limited timeframe. After conducting extensive research into swimming mechanics and the complexity involved in it, I decided that it was best I shifted the direction for level 2.

As a result of this I decided upon a new concept, an abandoned World War 2 hospital overrun by dead soldiers. This setting still preserved the eerie tone I was aiming for, while still allowing me to reuse existing movement and mechanics without the need for complex underwater systems. A level layout was created using Dungeon scrawl to help plan for the upcoming sprint.

A diagram of a hospital

AI-generated content may be incorrect.

*Figure 19: Level 2 Design*

## 7.8- Zombie Behaviour & Public Playtest - Sprint 8

### 7.8.1 Overview

This sprint marked the start of the development of Level 2. With the level design already created in the previous sprint, I began blocking out the environment within a separate scene in Unity. This sprint also featured the first public playtest, which extended beyond the designated private testing with selected individuals. Finally, work began on implementing enemy attack behaviour.

## 7.8.2 Sprint Tasks

* Devlog 5 - https://youtu.be/2hVWVUD5KXo
* Level 2 Blockout
* Level 2 Enemy Animations (running)
* Level 2 Zombie Attack Script
* Public Playtest

### 7.8.3 Summary of Actions

#### Level 2 Enemy Attack

When designing the enemy behaviour for this level, I decided that a close-range melee attack would be more fitting than projectiles, given the zombie themed enemies and the confined indoor environment. The enemies were intended to swipe or bite the player when within range, aligning with traditional zombie behaviour in games. Although the attack animations had not yet been implemented at this stage, the foundational attack script was developed, attached to the enemy models and fully working.

A screen shot of a computer program

AI-generated content may be incorrect.

*Figure 20: Zombie Attack Script*

This script allows the enemy to deal melee damage to the player when they are within range. It uses a trigger collider to detect when the player is close and checks a timer to ensure the enemy only attacks at set intervals.

#### Public Playtest

The first public playtest was conducted during this sprint, prior to this all playtests had been undertaken within an organised environment I created, with play testers that I handpicked.

If you wish to read more about this public playtest than I speak about this more in REFRENCE TO HERE. But the overall outcome of this was that I needed to make the game harder and there were a few minor visual bugs. I swiftly made changes to the game upon this feedback, and I would say that the playtest was successful.

A person sitting at a desk using a computer

AI-generated content may be incorrect.

*Figure 21: Undergoing a playtest*

## 7.8- Level 2 Sprint 8

### 7.8.1 Overview

This sprint was mostly spent building level 2, I began by sourcing appropriate assets online that complied with LSEP guidelines, checking they could be freely used in my project. With those assets gathered I transformed the level from a basic blockout into a fully fleshed out abandoned hospital. For this I used ProBuilder within Unity as I encountered issues with RealtimeCSG in the previous level and found ProBuilder to be more reliable.

### 7.8.2 Sprint Tasks

* Devlog 6 - https://youtu.be/nWcjUrmOtSQ
* Building Level 2
* Enemy Attack animations
* L2 Boss
* Level 2 Baking NavMesh and making sure the wave manager and other scripts work
* Started Itchio page
* Found assets

### 7.8.3 Summary of Actions

#### Level 2

This sprint was primarily focused on completing Level 2 to ensure I had a Minimum Awesome Product (MAP) by the end of the sprint. The entire level, consisting of multiple rooms, was created using various asset packs and audio sources, which are detailed in the asset list located in Appendix 7.

During this sprint attack animations were added to the enemies that were only triggered during attacks. Additionally, a boss was introduced which spawns in during round 5 and drops the timepiece part once being defeated.

Once the level design was complete the NavMesh was baked onto the ground, and all necessary game scripts were integrated onto the scene to ensure the game loop functioned as intended.

#### Itch

As the project was nearing completion, I created an itch.io page to showcase the game. This page was a platform to share progress and attract players to save my game so that when it releases in the future, they will be notified instantly.

Content was starting to be created for the page, including artwork, screenshots, and a description.

A screenshot of a video game

AI-generated content may be incorrect.

*Figure 22 : early version of itch page*

## 7.9- Minimum Awesome Product - Sprint 9

### 7.9.1 Overview

Now approaching the end of this project, the focus had shifted primarily to polishing the game. With the main game loop completed and all core functionalities implemented, the remaining time is being used to tidy up the game and reach a professional level.

During this sprint, the final required functionality, power ups, were added. This completed the gameplay features that were initially outlined during the Game Design Document.

In addition to gameplay features, time was also allocated to polishing the games storytelling elements and refining the game states. This included a intro audio sequence and outro, as well as a short pop up at the start that informs the player on the rules.

### 7. 9.2 Sprint Tasks

* Devlog 7 - https://youtu.be/-r5FLIDuMCo
* Polishing everything (floors , walls, textures)
* PowerUps fully implemented
* New UI created
* Testing
* Voice overs for story telling

### 7.9.3 Summary of Actions

#### Story Telling

During this final sprint, I reflected on the original goals outlined in the GDD and realised that the game lacked sufficient lore and storytelling elements. In the GDD, I had planned to include voice acting and narrative moments to enhance the players immersion and provide deeper context to the story’s world building.

To address this, I implemented two key voiceover elements during this sprint:

* Start Screen Voice Over: When the player first loads into the main menu, a voice over now plays, delivering the background story of the game and setting the tone for the players experience.
* Endgame Voice Over: Upon completing the game, a second voice over concludes the storyline, providing close and an ending narrative arc for players who complete the game. There is also a teaser to potential new updates and maps as it ends with the words “Everything seems normal… for now”.

These additions bring the game closer to the original vision by building atmosphere and making the world feel more alive.

#### Powerups

During this sprint, multiple power ups were added to give players temporary advantages during the tougher waves. The power ups implemented include:

1. Max Health: Instantly restores the players health to full
2. Double speed: Doubles the speed the player normally moves at for 20 seconds
3. Invincibility: Grants the player with invincibility for 30 seconds

To ensure players new when a powerup was active, a visual feedback system was introduced. Unity’s canvas elements and images were used for this. The image remains on display until the powerup has ran out of time.

A screenshot of a video game

AI-generated content may be incorrect.

*Figure 23: Invincibility Powerup active*

# **8 - End-project report**

Timeline Takedown was a successful project, it serves as a strong portfolio piece and a project that has helped to build my skills as a game developer. As discussed in 2.3 objectives & deliverables in this report , there were a set list of key deliverables that were created for this project. All the must have features were successfully implemented and multiple “should have” features were also implemented.

One of the strengths of this project was the level design, both levels were created from scratch, this involved original planning with paper-based level maps, that were then transformed into digital maps. These designs were then implemented into unity and created using Realtime CSG and Probuilder, all the assets were carefully hand picked and arranged across the map to build upon the theme. However, there was a planned number of at least 3 maps by the time of completion and there had only been 2 completed by the end, failing to meet one of the “want to have” requirements.

Another strength was the wavemanager system, once built it was simple to implement into the second level, it was just a matter of dragging game objects and prefabs to set up. If I was to continue development on this game than I would be able to set this system up in future levels with ease.

Unfortunately, I failed to meet the objectives of the “could have” and “want to have” section. I spoke about implementing mechanics such as online multiplayer, missions, cosmetics and more, but due to time and technical constraints I did not reach these goals.

In conclusion, I achieved the base goals set out at the start of the project and I am pleased with how the final product turned out. A full game was created, featuring a complete main game loop and a clear objective for the player to accomplish. All core requirements were met, and had there been a few additional weeks available, Level 3 would have been completed as its design stage was already finished.

One of my key takeaways from this project is the importance of effective prioritisation. There were instances where I spent too much time focusing on certain aspects, such as animation and troubleshooting NavMesh issues, which then reduced the time available for other sections. In future projects, I would place a greater emphasis on better time management and ensuring that all features are developed in a more balanced and structured way.

# **9 - Postmortem**

## 9.1 Reflection

During this project almost all assets in the game were not created by me. I see this as an imperfection within the project as it feels less personal knowing that I have not contributed to that much of the games modelling and asset making. Going forward I would like to start replacing assets with assets that I have made. Time constraints were stopping me from doing this during this project as if I was to have created all assets for the game, I would have never got the game done in time.

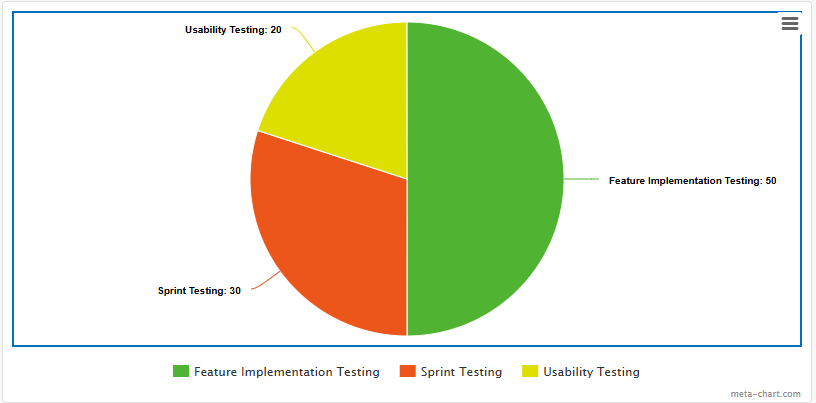
Additionally, I would like to release Timeline Takedown to Steam in the future, I believe once there are more levels in the game, this would be worthy of a steam release. For the time being, Itchio is more acceptable as its free for developers to upload projects to, and free for game enthusiasts to download games.

## 9.2 Data

A chart with colorful rectangular objects

AI-generated content may be incorrect.

*Figure 24: Sprint Schedule for Timeline Takedown*



*Figure 24: Testing Types*

## 9.3 Conclusion

In conclusion Timeline Takedown was a success, the game had a working wave system where the enemies would spawn in at set spawn locations, and the waves would cycle as intended. Even though there are not as many levels as I initially planned for, the game is in a good place now where it has a complete game loop and can easily be worked upon in the future to create new content such as new levels, enemies, and weapons.

# **10 - References**

Unity Technologies, 2024. *Unity Game Engine (Version 2022.3)* [software]. Available at: <https://unity.com/> [Accessed 13 April 2025].

PEGI (2017). *PEGI age ratings | Pegi Public Site*. [online] Pegi.info. Available at: <https://pegi.info/page/pegi-age-ratings> [Accessed 13 April 2025].

Treyarch, 2008. *Call of Duty: World at War* [video game]. Activision.

‌Tripwire Interactive, 2009. Killing Floor [video game]. Tripwire Interactive.

Anon, (2023). *Unveiling the Demographics of Call of Duty’s Target Audience – Openr*. [online] Available at: <https://openr.co/unveiling-the-demographics-of-call-of-dutys-target-audience/> [Accessed 13 April 2025].

Yin-Poole, W. (2025). *Activision Finally Admits It Uses Generative AI for Some Call of Duty: Black Ops 6 Assets After Backlash Following ‘AI Slop’ Zombie Santa Loading Screen - IGN*. [online] IGN. Available at: <https://www.ign.com/articles/activision-finally-admits-it-uses-generative-ai-for-some-call-of-duty-black-ops-6-assets-after-backlash-following-ai-slop-zombie-santa-loading-screen> (Accessed 15 April 2025).

Andy Hartup, (2016). *Call of Duty Zombies - the accidental creation of a glorious gaming monster*. [online] gamesradar. Available at: <https://www.gamesradar.com/call-of-duty-zombies-the-accidental-creation-of-a-glorious-gaming-monster/> (Accessed 20 April 2025).

Tripwire Interactive, 2016. Killing Floor 2 [video game]. Tripwire Interactive. Available at: <https://tripwireinteractive.com/#/games> [Accessed 15 April 2025].

‌

Trello, 2025. Timeline Takedown Development Board. [online] Available at:

[Accessed 17 April 2025].

Clinton Keith, (2020). *Agile Game Development: Build, Play, Repeat, 2nd Edition*. [online] O’Reilly Online Learning. Available at: https://www.oreilly.com/library/view/agile-game-development/9780136204831/ [Accessed 17 Apr. 2025].

‌app.dungeonscrawl.com. (n.d.). *Dungeon Scrawl*. [online] Available at: https://app.dungeonscrawl.com

## **11 - Appendices**