# **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.