



Shrine Seeker

Ben Sharkey

N00212320

Supervisor: Naoise Collins

Second Reader: Catherine Noonan

Year 4 2024/25

DL836 BSc (Hons) in Creative Computing

Abstract

The aim of this project was to create a 16-bit adventure game with a heavy focus on evoking a sense of nostalgia in the player for the classic titles of the 1990s. The aim with the game was to create an experience that represents this era through the use of pixelated graphics, chiptune sounds and simple, straightforward gameplay. The rationale for this project in particular stems from the endurance of these retro titles and aims to provide an insight as to why, after all these years, they are still some of the most enjoyed video games in the world.

The purpose of the game is to allow the players to explore the feeling of nostalgia as they navigate a temple, armed with nothing but a sword. This simple yet intuitive concept places emphasis on the core gameplay loop present in those retro titles and reinforces the nostalgic experience with use of sensory immersion.

The development process can be broken down into a few key steps. First, pixel sprites were gathered that accurately replicated the visual style of 16-bit games. Second, chiptune sounds were gathered to complement the visuals. Third, the core game mechanics and logic, including combat, player movement and enemy AI were implemented. Finally, these three key components were tied together in order to create the project about to be presented.

Acknowledgements

First and foremost, I would like to thank my supervisor Naoise Collins for the invaluable guidance and assistance throughout the development project. Furthermore, I would like to thank all of my friends for their support. Last, but certainly not least, I would like to give special thanks to my family, as without their unwavering support none of this would have been possible.

The incorporation of material without formal and proper acknowledgement (even with no deliberate intent to cheat) can constitute plagiarism.

If you have received significant help with a solution from one or more colleagues, you should document this in your submitted work and if you have any doubt as to what level of discussion/collaboration is acceptable, you should consult your lecturer or the Course Director.

WARNING: Take care when discarding program listings lest they be copied by someone else, which may well bring you under suspicion. Do not leave copies of your own files on a hard disk where they can be accessed by others. Be aware that removable media, used to transfer work, may also be removed and/or copied by others if left unattended.

Plagiarism is considered to be an act of fraudulence and an offence against Institute discipline.

Alleged plagiarism will be investigated and dealt with appropriately by the Institute. Please refer to the Institute Handbook for further details of penalties.

The following is an extract from the B.Sc. in Creative Computing (Hons) course handbook. Please read carefully and sign the declaration below

Collusion may be defined as more than one person working on an individual assessment. This would include jointly developed solutions as well as one individual giving a solution to another who then makes some changes and hands it up as their own work.

DECLARATION:

I am aware of the Institute's policy on plagiarism and certify that this thesis is my own work.

Student : BEN SHARKEY

Signed

A handwritten signature in black ink that reads "Ben Sharkey". The signature is written in a cursive, slightly slanted style.

Failure to complete and submit this form may lead to an investigation into your work.

Table of Contents

1 Introduction	1
2 Research	3
Chapter 1: Introduction	4
Chapter 2: The 8-bit and 16-bit Eras	5
Chapter 3: Impact on Modern Gaming	7
Chapter 4: Case Studies	8
3 Requirements	11
3.1 Introduction	11
3.2 Requirements gathering	12
3.2.1 Similar applications	12
3.2.2 Interviews	13
3.3 Requirements modelling	17
3.3.1 Personas	17
3.3.2 Functional requirements	18
3.3.3 Non-functional requirements	18
3.3.4 Use Case Diagrams	19
3.4 Feasibility	19
3.5 Conclusion	20
4 Design	22
4.1 Introduction	22
4.2 Program Design	22
4.2.1 Gameplay Loop	23
4.2.2 Object Management	23
4.2.3 Scene Management	23
4.2.4 Input System	23
4.2.5 Physics & Collision Detection	24
4.2.6 Game State Management	24
4.2.7 Enemy AI	24
4.2.8 Level Design Integration	24
4.3 User interface design	24
4.3.1 Visual Presentation	25
4.3.2 Heads-Up Display	25
4.3.3 Input Feedback	25
4.3.4 Menus	25
4.3.5 World Interaction	26
4.3.6 Storyboard	26

4.3.7 Level Design	26
4.3.8 Environment	26
4.4 Conclusion	28
5 Implementation	29
5.1 Introduction	29
5.2 Scrum Methodology	30
5.3 Development environment	31
5.4 Sprint 1 - Research & Concept	31
5.4.1 Goal	31
5.4.2 Item 1 - Research	31
5.4.3 Item 2 - Concept	32
5.5 Sprint 2 - Feasibility & Requirements	32
5.5.1 Goal	32
5.5.2 Item 1 - Core Engine Proficiency	32
5.5.3 Item 2 - Asset Creation	33
5.5.4 Item 3 - Core Mechanics	33
5.5.5 Item 4 - Level Design and Integration	35
5.5.6 Item 5 - Performance Optimization	36
5.5.7 Item 6 - Time Commitment	37
5.6 Sprint 3 - Key Mechanic	37
5.6.1 Goal	37
5.6.2 Item 6 - Key Mechanic Brainstorming	37
5.7 Sprint 4 - Prototype	39
5.7.1 Goal	39
5.7.2 Item 1 - Player Movement	39
Item 2 - Player Attack	41
5.7.3 Item 3 - Damage and Knockback	42
5.8 Sprint 5 - Level Design	43
5.8.1 Goal	43
5.8.2 Item 1 - Visual Design	43
5.8.3 Item 2 - Map Design	44
5.9 Sprint 6 - Mechanic Design	45
5.9.1 Goal	45
5.9.2 Item 1 - Chest	45
5.9.3 Item 2 - Door	47
5.9.4 Item 3 - Enemy Room	49
5.10 Sprint 7 - Audio Design	53
5.10.1 Goal	53
5.10.2 Item 1 - Audio Design	53
5.11 Sprint 8 - Finalization	54

5.12 Conclusion	54
6 Testing	55
6.1 Introduction	55
6.2 Functional Testing	55
6.2.1 Combat Mechanics	55
6.2.2 World Interaction	56
6.2.3 Game Progression/State	57
6.2.4 Discussion of Functional Testing Results	58
6.3 User Testing	59
6.3.1 Private Testing	59
6.3.2 Public Testing	59
6.4 Conclusion	60
7 Project Management	62
7.1 Introduction	62
7.2 Project Phases	62
7.2.1 Proposal	62
7.2.2 Requirements	63
7.2.3 Design	64
7.2.4 Implementation	64
7.2.5 Testing	65
7.3 SCRUM Methodology	65
7.4 Project Management Tools	66
7.4.1 Trello	66
7.4.2 GitHub	66
7.5 Reflection	67
7.5.1 Your views on the project	67
7.5.2 Completing a large software development project	68
7.5.3 Working with a supervisor	68
7.5.4 Technical skills	68
7.5.5 Further competencies and skills	68
7.6 Conclusion	69
8 Conclusion	70
References	71

1 Introduction

The aim of this project was to create a 16-bit style adventure game, designed around the feeling of nostalgia evoked by the classic titles of the 1990s. The application area for this particular project lies within independent game development with the target audience being players with a deep appreciation of both the aesthetic and mechanics offered by retro titles. By replicating these properties, this project aims to provide a connection to the past of sorts, allowing players to revisit or even discover anew the charm of 16-bit games.

The development process for this project involved the use of a wide range of software to achieve the desired finish. Unity served as the core game engine, providing a solid foundation for the physics, rendering and overall game logic required. Visual Studio Code was used for creating the C# scripts. Piskel was used for editing sprites and making sure they looked the best they possibly could in order to accurately reflect the retro look required for the project scope. Finally, an online library called Epidemic Sounds was used to source the chiptune music and sound effects that were used to enhance the auditory experience offered by the game.

The project was managed through Trello, which provided a straightforward and easy to read solution for managing the tasks and deadlines through the development process. Github was used for version control, ensuring the code was kept clean and accessible as well as making bugs easier to track down and fix. These tools proved to be essential in the making the project as they assisted in maintaining a clean and efficient workflow throughout.

The requirements of the game were focused around delivering an experience that reflected those classic 16-bit titles. This included engaging combat, a visually appealing retro art style and intuitive controls for the player. The design phase included creating both the layout for the temple and the sprites for the characters while also maintaining a consistent art style and aesthetic. The implementation phase included the implementation of both the audio and art assets as well as the creation and optimization of core game logic.

Testing was one of the most integral parts of the development process. In order to more accurately gather user feedback and verify how playable the game was, the project was showcased at Comic Con in Dublin on March 15th and 16th 2025. This presented an opportunity to gather data, observe how players interacted with the game and identify bugs present in the build. In the end, hundreds of people were able to play the game and provide instant, accurate feedback that proved to be invaluable in the run to the final product. The insights and experience gained from this testing environment contributed significantly to the overall polish and playability of the final build, helping validate the aim and design choices behind the project.

2 Research

The research phase of this project was heavily based on the research paper we submitted as part of the Research & Analytics module in Semester 1. The paper in question is based around the impact of both 8-Bit and 16-Bit games on the modern title. The paper is as follows.

Table of Contents

Chapter 1: Introduction	3
1.1 Background:	3
1.2 Research Question:	3
1.3 Research Objectives:	3
1.4 Research Methodology:	4
1.5 Report Structure:	4
Chapter 2: The 8-bit and 16-bit Eras	4
2.1 Technological Advancements:	4
2.2 Pioneering Games and Consoles:	5
2.3 Game Design and Mechanics:	5
Chapter 3: Impact on Modern Gaming	6
3.1 Game Design and Mechanics:	6
3.2 Storytelling and Narrative:	6
3.3 Player Experience and Nostalgia:	7
Chapter 4: Case Studies	7
4.1 Games:	7
4.2 Analysis:	7
4.3 Discussion:	8
Chapter 5: Community	8
5.1 Development of Gaming Communities:	8
5.2 Speedrunning:	9
5.3 Retro Gaming Competitions:	9
5.4 Role of Gaming Conventions:	9
Chapter 6: Conclusion	10
6.1 Summary of Findings:	10
6.2 Contribution to Knowledge:	10
6.3 Limitations and Future Research:	10
6.4 Final Thoughts:	10
References	11

Chapter 1: Introduction

1.1 Background:

Both the 8-bit and 16-bit eras are some of the most important and vital in the history of video games. Often referred to as the 'Golden Age of Gaming', these eras marked a pivotal shift in the evolution of video games. Spanning the late 1970s to the mid-late 1990s, these eras were characterised by their significant leaps in technology and innovation in the video game space. They also hold the origin stories of many of the industry's biggest franchises that continue to influence the industry today.

The Nintendo Entertainment System and the Super Nintendo Entertainment System, most commonly known as the NES and SNES respectively, were responsible for bringing gaming to a worldwide audience.

1.2 Research Question:

In what way have the 8-bit and 16-bit eras of video games affected the development of modern gaming, specifically in terms of mechanics, storytelling and especially player experience?

1.3 Research Objectives:

In relation to the research questions, the following concepts will be explored;

1. Analysis of the technological advancements that both defined the 8-bit and 16-bit eras and furthered their impact on game design.
2. A study of the core mechanics and game design techniques present during these eras such as role-playing, platforming and puzzle-solving.
3. A breakdown of the influence that 8-bit and 16-bit games have had on both modern game mechanics and design, especially the use of pixel art and retro aesthetics.

4. An exploration of the evolution of storytelling in video games, from the limited yet creative methods used in early games to the more in depth and gripping narratives associated with modern titles.
5. An assessment of how 8-bit and 16-bit titles have impacted player experience and a brief look into the concept of nostalgia.

1.4 Research Methodology:

This research will be conducted with a mixed-methods approach in mind, aiming to combine both quantitative and qualitative methods to create a deeper understanding of the topic. These methods include literature review, case study analysis and a survey.

1.5 Report Structure:

The following paper will be broken down into a handful of key chapters. Chapter 2 will contain an overview of the 8-bit and 16-bit eras including advancements in technology, core game design principles and the most influential games of these eras. Chapter 3 will further explore the impact of these eras of gaming as we know it today with a focus on storytelling, game mechanics and of course the player experience. Chapter 4 will cover a handful of case studies regarding specific games that demonstrate the influence of 8-bit and 16-bit games of design. Finally, Chapter 5 will summarise all of the previous findings and provide an overall conclusion to the research report.

Chapter 2: The 8-bit and 16-bit Eras

2.1 Technological Advancements:

The 8-bit and 16-bit eras are most known for the leaps they created in the technology used to create games. The technological breakthroughs, such as the creation of more powerful processors and rapidly improving graphics chips, allowed for the creation of more in depth and immersive games.

The 8-bit era was one of the earlier forms of gaming, meaning it had very limited features even compared to the 16-bit era. During this era games had a very limited colour palette that often resulted in very simple pixelated graphics. As well as this the sound capabilities were limited to very simple sound effects and was home to a unique genre of music known as chiptune music, [1] “a type of electronic music that utilises the sound chips found in vintage arcade machines, computers, and video game consoles popular in the 1980s.” The

games in this era were also built on straightforward mechanics such as running, jumping and attacking.

The 16-bit era had a lot more going for it than its counterpart, boasting enhanced graphics and sound. The increased processing power present in this era allowed for the games to have more detailed sprites, wider colour palettes and more smooth animation. Advanced sound chips also made higher quality sound effects and music possible in these games, creating a more immersive auditory experience for the player. Game developers also had more freedom during this era, experimenting with more complex gameplay mechanics such as challenging platforming, puzzle solving and role-playing elements.

2.2 Pioneering Games and Consoles:

These two eras are home to some of the most widely recognized gaming systems in history. The 8-bit era had both the Nintendo Entertainment System and the Sega Master System while the 16-bit era was home to the Super Nintendo Entertainment System and the Sega Genesis/Mega Drive.

Throughout these two eras the aforementioned consoles became home to a lot of franchises that would go on to shape modern gaming as we know it. Among these were titles such as Super Mario Bros, Legend of Zelda, Sonic the Hedgehog, Final Fantasy, Streets of Rage and Mortal Kombat.

2.3 Game Design and Mechanics:

The 8-bit 16-bit eras laid the foundation for many of the core game design principles and mechanics that are still used in games today.

1. Platforming: Games such as Super Mario Bros. and Sonic the Hedgehog popularised the platforming genre to a wider audience by emphasising the use of timing, precise jumps and level design.
2. Role-Playing Games: Also known as RPGs, titles such as Final Fantasy and Dragon Quest introduced more complex storylines, turn-based combat systems and character development that had not been seen in many games before this.
3. Puzzle-Solving: Games such as Metroid and The Legend of Zelda presented an environment to players in which thinking outside the box and solving puzzles was key to progressing through the game.

Chapter 3: Impact on Modern Gaming

3.1 Game Design and Mechanics:

The design principles used during the 8-bit and 16-bit eras are still seen in the games that we have today.

As mentioned previously, platformers such as Super Mario Bros. and Sonic the Hedgehog introduced running and precise jumping. But these games also introduced the obstacle avoidance mechanics that have shaped the way countless modern platforming games work. Early RPGs then established the foundations of character progression, skill trees and the turn-based combat systems which continue to be the centre of modern RPGs today. Finally, games such as The Legend of Zelda and Metroid popularised the implementation of environmental puzzles with item-based solutions as well as exploration mechanics that continue to form the core of most modern day adventure games.

Additionally, the pixelated art style and retro aesthetic of these early games has experienced a boom in popularity as of late. Modern day indie developers often pay homage to these classic games by using pixel art and chiptune music to create a nostalgic experience.

3.2 Storytelling and Narrative:

Early games often relied on very limited storytelling, however they introduced narrative elements that have evolved over time into the complex narratives we see today in modern titles. These developments can be split up into three key categories;

1. **Character Development:** Iconic characters such as Mario, Link and Sonic established their backstories and a strong sense of identity in their earlier appearances which made room for more nuanced character development in their more recent games.
2. **Worldbuilding:** Games such as The Legend of Zelda and Final Fantasy created these immersive worlds that were packed full of lore and history. This would go on to inspire many of the open-world games we see today.
3. **Nonlinear Storytelling:** Although early games typically followed a more linear narrative, modern games began to experiment with broader nonlinear narratives with player choice. These decisions were inspired by those early linear titles such as The Legend of Zelda: A Link to the Past.

3.3 Player Experience and Nostalgia:

The 8-bit and 16-bit eras had a big influence on both player expectations and preferences, influencing the modern games we know today as a result. These influences can be split up into 3 primary categories;

1. **Nostalgia:** Most gamers today grew up playing the classic 8-bit and 16-bit games of yesterday and developers leverage this sense of nostalgia in order to appeal to this demographic of players.
2. **Player Agency:** Early games gave players freedom and the ability to explore, which inspired modern games to provide players with more of the same.
3. **Immersive Experiences:** Although early games were limited by the technology of the time, they prioritized immersive experiences which led to the development of more realistic game worlds full of detail and lore.

Chapter 4: Case Studies

4.1 Games:

For this section of the paper, we will examine two modern games that really show the influence that 8-bit and 16-bit games have had on modern titles:

Hollow Knight: A Metroidvania-style game that is most known for its focus on exploration, challenging combat, and an atmospheric world full of character.

Undertale: A role-playing game that features a unique one-of-a-kind narrative, player choice, and an innovative battle system.

4.2 Analysis:

Hollow Knight:

Game Mechanics: Hollow Knight focuses on precise platforming, independent exploration, and a combat system reminiscent of classic Metroidvania games.

Visual Style: The game's pixel-based art style and limited color palette hold the same aesthetics as the 8-bit and 16-bit era games we all know and love which gives the game a nostalgic atmosphere.

Sound Design: The game features a chiptune soundtrack and the sound effects present make the game feel like a retro title, enhancing the overall gaming experience for players who are chasing a retro experience.

Narrative Structure: While the narrative is more subtle and open ended than other games of its type, it encourages the player to explore independently and make discoveries by themselves without guidance, similar to classic adventure games.

Undertale:

Game Mechanics: Undertale's innovative battle system gives players the choice to either kill or spare their enemies which expands on the traditional concept of an RPG game.

Visual Style: The game's pixel-based art style and simple character designs pay homage to the classic RPGs present on the likes of the SNES.

Sound Design: The game's chiptune soundtrack and retro sound effects create a nostalgic and magical atmosphere, providing a very relaxing experience for players making their way through the game.

Narrative Structure: The nonlinear narrative offers a unique gaming experience that is reminiscent of the classic adventure games we all know and love. This is further expanded with its aforementioned kill/spare system, forcing the player to form their own moral compass in order to progress through the story and reach one of the multiple endings that the game offers.

4.3 Discussion:

Both Hollow Knight and Undertale are perfect examples of how modern game developers draw inspiration from both the 8-bit and 16-bit eras in order to build innovative and engaging experiences. By utilizing classic game mechanics, art styles and sound design, these games appeal to both older gamers seeking a sense of nostalgia and to the new generation of players only just beginning their journey into gaming.

These games also showcase the enduring impact and appeal of pixel art and retro titles. By embracing this era of gaming history developers can create visually distinct and memorable games that stand out in the diluted modern market we find ourselves with today.

Writing about the influence of 8-bit and 16-bit games on the modern market directly effected the development of this project and provided many invaluable insights. By examining the mechanics, player experiences and style of storytelling present in these games, the paper provided many guidelines for replicating the appeal of those classic titles.

In particular, the analysis of core gameplay principles and both pixel art and chiptune audio provided a new perspective and inspiration for the design and general aesthetic of the game. As well as this, exploring the concept of nostalgia and the overall importance of player agency shaped the game's emphasis on both exploration and immersion. Games such as Hollow Knight and Undertale provided invaluable information on retro aesthetics when combined with modern gameplay which reinforced the project's goal of creating a nostalgic yet unique experience for the player. In the end, this research served as an extremely valuable resource that ensured the game effectively captured the essence of games from the 16-bit era while maintaining a modern feel, appealing to all players both modern and retro.

3 Requirements

3.1 Introduction

The purpose of the requirements phase is to allow for developers to work out what the application should be able to do. It is important to understand what the users would like the application to do rather than the developer deciding what is required.

You can write a bit about your project area. Each paragraph has a blank line between it and the previous paragraph

The requirements phase for this project was focused around realizing the core functionalities and user expectations behind a 16-bit retro game. This stage was paramount in ensuring the final build aligned with the project's aim of evoking a sense of nostalgia in users searching for those classic action-adventure games of the 1990s. The project's application area called for a very specific set of requirements tailored around replicating both the core gameplay and aesthetic of that era.

Understanding the wants of the user was also invaluable in shaping an authentic 16-Bit experience. This involved identifying key elements that resonated with long time fans of the genre while ensuring the game remained accessible for those unfamiliar with the genre as a whole. The core requirement/goal was to create a game that had a genuine 'retro' feel, achieved through simple, engaging gameplay, charming pixel graphics and upbeat chiptune audio.

To be more specific, user expectations were centered around intuitive controls, satisfying mechanics and both a visually immersive and consistent environment. Players wished for a straightforward experience that allowed them to explore a temple like environment, battle enemies using a simple sword as their weapon and progress through the game without needing to comprehend convoluted narratives or mind-bending mechanics.

In order to ensure the requirements were accurately reflecting user preferences, feedback from players was considered throughout each stage of the development progress. The user testing performed at Comic Con provided insights into player expectations that proved to be invaluable in the extended development of the project. This real-world style of testing helped refine the final game and ensured that it both achieved and provided the intended nostalgic experience.

3.2 Requirements gathering

3.2.1 Similar applications

1. The Legend of Zelda: A Link to the Past (SNES)

Description:

- A Link to the Past is an action-adventure game that features puzzles, combat and exploration. Players take control of a character named Link as he travels between the worlds of light and darkness in order to rescue Princess Zelda and defeat the evil known as Ganon.
- Link to the Past is mostly known for its imaginative level design.

Advantages:

- Incredible level design featuring intricate dungeons and an actively changing environment.
- First of its kind item mechanics and puzzles.
- Interesting narrative and memorable characters.
- Accessible difficulty

Disadvantages:

- Some of the puzzles can be obscure and often vague, potentially leading to the player becoming frustrated.
- The narrative has not aged well, although strong for its time it is now quite simple by modern standards.

2. Castlevania: Symphony of the Night (PlayStation)

Description:

- Symphony of the Night is an action role-playing platformer featuring combat mechanics and exploration. Players control a character by the name of Alucard as he explores Dracula's castle, gaining abilities and items along the way.
- This game is known for its non-linear exploration, giving players more freedom over how they experience the game.

Advantages:

- Non-linear exploration with an open, analogous castle.
- In depth RPG elements with both consistent character progression and item collection mechanics.
- Beautiful pixel art and an atmospheric soundtrack to compliment it.
- An endless amount of replayability.

Disadvantages:

- Certain players may find the open ended nature of the castle tedious.
- The difficulty can suddenly spike in certain areas of the map, providing an inconsistent experience and expectation.

3. Super Metroid (SNES)

Description:

- Super Metroid for the SNES is an action-adventure platformer featuring both exploration and combat elements. Players control a character by the name of Samus Aran as she explores the planet of Zebes.
- This game is known for its immersive atmosphere and its open ended non-linear exploration.

Advantages:

- Atmospheric environments and immersive world design.
- Simple controls and fluid character movement.
- Places emphasis on exploration and independent discovery, enhancing the experience of the player.
- Well designed levels.

Disadvantages:

- The game can be cryptic at times, providing limited guidance for progression.
- Some players may find the difficulty of the game frustrating.

3.2.2 Interviews

Conduct interviews with 3 or 4 users to find out what the important features for them for the app are. There may be various issues that arise in multiple interviews. These can be grouped together into a number of themes.

As part of the research for the project a series of interviews were conducted in order to gather more information regarding user expectations. The interview format is as follows;

1. What would you say are your favourite aspects of 16-bit video games?
2. What are some common frustrations would you say you experience when playing these types of games?
3. What features would you say are most important to you when playing a 16-bit adventure game?
4. In your opinion, how important are the visuals and sound design to your overall enjoyment of retro games?
5. What level of difficulty would you say you find the most enjoyable when playing a video game?
6. Any other comments or suggestions?

Based on notes taken during the interviews, here are four of the best interviews. All interviews were done anonymously.

User 1:

1. What would you say are your favorite aspects of 16-bit video games?

"I love exploring and the sense of independence. Finding those hidden areas and secret items is really satisfying."

2. What are some common frustrations would you say you experience when playing these types of games?

"Sometimes the controls can feel quite clunky, or the puzzles can be too obscure and difficult to figure out."

3. What features would you say are most important to you when playing a 16-bit adventure game?

"Exploration and combat are the most important to me. I like a nice balance between the two, makes the game feel more full."

4. In your opinion, how important are the visuals and sound design to your overall enjoyment of retro games?

"The art and sounds are crucial! They set the tone and make the game immersive. Without them the game would feel a lot different."

5. What level of difficulty would you say you find the most enjoyable when playing a video game?

"I think a moderate challenge is the best for me. I don't like it to be too easy or too frustrating."

6. Any other comments or suggestions?

"A map system would be really helpful, especially given that it's set in a temple. People could get lost!"

User 2:

1. What would you say are your favorite aspects of 16-bit video games?

"For me that nostalgic feeling and the simple gameplay loops are what I enjoy most."

2. What are some common frustrations would you say you experience when playing these types of games?

"Repetitive combat is a big one, or having to backtrack a lot around the map can get boring quickly."

3. What features would you say are most important to you when playing a 16-bit adventure game?

"Fast paced combat and a good story are essential for me. I like to feel invested in the world the game is creating."

4. In your opinion, how important are the visuals and sound design to your overall enjoyment of retro games?

"The design of the pixel art and chiptune music are a huge part of the appeal of retro games as a whole. They need to be done right"

5. What level of difficulty would you say you find the most enjoyable when playing a video game?

"I prefer a game that's challenging, but fair. I find easier games quite boring, I like that feeling of achievement when you beat a powerful boss!"

6. Any other comments or suggestions?

"Make sure the controls are responsive. They're super important!"

User 3:

1. What would you say are your favourite aspects of 16-bit video games?

"I quite enjoy the challenge and satisfaction of overcoming difficult enemies or puzzles."

2. What are some common frustrations would you say you experience when playing these types of games?

"The level design can be confusing sometimes and makes it easy to get lost."

3. What features would you say are most important to you when playing a 16-bit adventure game?

"I love exploring the world and trying to learn all of the lore behind it"

4. In your opinion, how important are the visuals and sound design to your overall enjoyment of retro games?

"Well they create the atmosphere of the whole game in my opinion, so I'd say they're pretty important!."

5. What level of difficulty would you say you find the most enjoyable when playing a video game?

"I like a game that has a progressing difficulty, starting easy and getting harder."

6. Any other comments or suggestions?

"I would love to see a nice variety of both enemies and environments."

User 4:

1. What would you say are your favorite aspects of 16-bit video games?

"That sense of adventure is probably my favorite thing about 16-bit games, nothing beats the old Zelda games!."

2. What are some common frustrations would you say you experience when playing these types of games?

"Sometimes I find the combat is too simple, or that the enemy AI is poor."

3. What features would you say are most important to you when playing a 16-bit adventure game?

"The combat must be fun. If it's not, then it drags down the whole game."

4. In your opinion, how important are the visuals and sound design to your overall enjoyment of retro games?

"The art and music in a game are extremely important! It's what keeps me coming back to retro games."

5. What level of difficulty would you say you find the most enjoyable when playing a video game?

"I prefer an easier, more casual gaming experience."

6. Any other comments or suggestions?

"I would like the game to have a unique feel, and not just be a carbon copy of other games."

3.3 Requirements modelling

3.3.1 Personas

Around the theme of the project there are two distinct personas that the game aims to please: People who are interested in exploring the feeling of nostalgia, and people who are rediscovering it through games that remind them of their childhoods.

Andre is the first of these two personas. A student that just about missed out on that 90s era of games who now wishes to explore it and see what he's been missing.

The persona card for Andre Silva is designed with a modern, clean aesthetic. It features a circular profile picture of a smiling young man with short dark hair, wearing a denim jacket over a white shirt. The card is divided into several sections: a header with the name 'Andre Silva' and the role 'Student'; a 'BIO' section with a short paragraph about his background; a 'GOALS' section detailing his interest in retro gaming; a 'MOTIVATION' section explaining his desire to explore new genres; a 'PERSONALITY' section with two horizontal progress bars for 'Thoughtful' and 'Resourceful'; and a 'SKILLS' section with two rows of five circular indicators for 'Problem Solving' and 'Writing'. The card also includes three small white buttons labeled 'Smart', 'Honest', and 'Loyal' at the bottom left. The background of the card is a light gray with a subtle geometric pattern, and the overall color scheme is dark blue and white.

BIO

Andre is a college student studying technology. Following the outbreak of COVID, Andre decided it was about time he delved into retro gaming, just as so many other people did during that time.

GOALS

Andre's goal here is to explore the feeling of nostalgia, and discover what people are always ranting and raving about when it comes to these retro games.

MOTIVATION

Spurred on by the boredom of lockdown, Andre had plenty of time on his hands to explore new genres.

PERSONALITY

Thoughtful

Resourceful

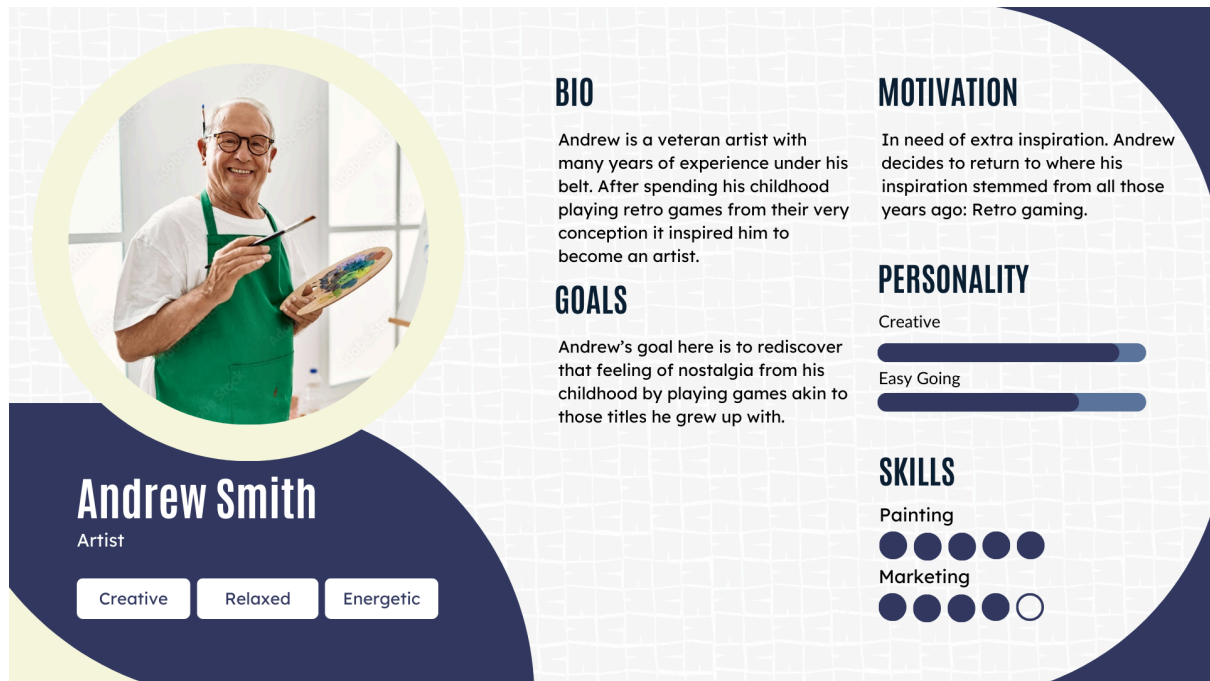
SKILLS

Problem Solving

Writing

Smart Honest Loyal

Andrew is the second of the two personas. A veteran artist who grew up playing retro games from their very conception who wishes to return to them and rediscover that feeling of nostalgia.



3.3.2 Functional requirements

1. The game should allow players to explore the temple in an engaging way, with some hidden areas and secrets to discover.
2. The combat system should feel responsive and fluid.
3. The graphics and soundtrack should be of a good quality to further the atmosphere of the game.
4. The controls for the game should be simple and easy to understand.
5. The difficulty curve throughout the game should remain fair, but challenging.
6. The level design should be designed to minimize frustration or confusion.
7. The game should offer a unique twist, and avoid feeling generic.
8. Items that the player discovers should have a clear purpose and be easy to understand.

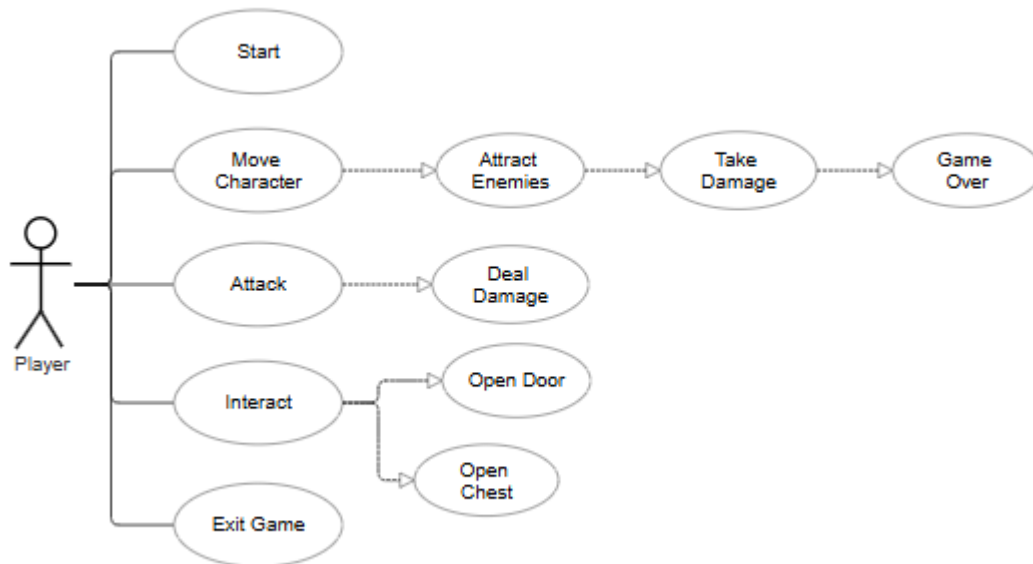
3.3.3 Non-functional requirements

1. The game should have an easy to understand interface and feature intuitive controls in order to minimize the learning curve players experience.
2. The game should maintain a consistent frame rate of at least 30 FPS in order to ensure the gameplay is smooth and responsive.
3. The code base itself should be well-organized and documented to allow bug fixes to be executed swiftly.

4. The game should be accessible for players at all skill levels.

3.3.4 Use Case Diagrams

This is a use case diagram depicting the standard experience a player will have when playing the game.



3.4 Feasibility

The primary development engine chosen to create this project is Unity, a very popular game engine known for its easy to use feature set and cross-platform capabilities. The scripting capabilities and editor software offered by Unity allow for a tighter, streamlined workflow when developing 2D games. This makes it the perfect engine to create the desired retro gameplay and general aesthetic needed for this project.

Visual Studio code was utilized for all of the scripting throughout the project due to it's lightweight nature and ability to seamlessly integrate itself with the Unity engine. The visual assets were created using a software by the name of Piskel, a free to use sprite suite typically used to both create and animate high quality 2D sprites. The audio used throughout the game was sourced from an online library called Epidemic Sound, a royalty-free music library. The reasoning behind this choice was to ensure the audio design was of a professional grade and was without any licensing complications.

Version control for this project was managed through GitHub, which provided a reliable and, most importantly, secure platform for both storing and tracking the changes being made to the code at each stage. Project management was done through Trello, which offered a

flexible and easy to understand solution for managing deadlines, tracking progress and prioritizing certain tasks.

In terms of the technical feasibility, all the aforementioned applications present little to no compatibility issues. Unity's compatibility with Visual Studio Code is something that has already long been established within the industry. The files that Piskel outputs are fully compatible with Unity's formatting from the get-go which eliminates any potential issues that could happen with asset integration. Finally, the files present in the Epidemic Sound library are all ready to use with Unity's audio system right out of the box, allowing instant integration without having to compress files or sacrifice quality.

With that being said, potential challenges do exist in the form of optimization. While Unity is fully capable of producing competent 2D titles, careful optimization of game logic and visual assets is needed in order to maintain a consistent framerate, especially on lower end systems. The number of on-screen sprites has to be monitored as well as the number of calculations that are being performed at each frame.

Another potential issue lies within the implementation of enemy AI and collision detection. While Unity does provide tools to make this easier for the developer, extensive testing and scripting may be required in order to achieve the ideal level of complexity and responsiveness. Making sure that the game feels challenging yet remains fair to the player is a delicate balancing act and a key part of delivering an enjoyable experience.

The use of Epidemic Sound for all of the audio assets does mitigate licensing concerns, but it can become limiting when trying to keep all sounds within the same aesthetic. The audio design must enhance the experience for the player, not take away from it.

Overall, the technologies chosen to develop the game provide a solid foundation in the creation of a 16-Bit title. With the right planning, optimization and testing, these challenges can be effectively addressed and make for a high quality and, better yet, an enjoyable gaming experience for the player.

3.5 Conclusion

Following this chapter, we've established the requirements and technological framework needed for the development of the project. Through user interviews and detailed analysis, a

clear set of functional requirements was defined that prioritizes core gameplay elements such as engaging combat, open ended exploration and an immersive auditory experience. The game must offer a sense of achievement when exploring, deliver enjoyable combat and provide both a visually and aurally satisfying atmosphere for the player. Easy to understand controls, a balanced difficulty curve and a clear level design are all essential pieces of the project. They serve as the foundation for the design and implementation of the game and ensures that the final build aligns with both the expectations of the user and scope of the project.

A handful of non-functional requirements were also established in order to ensure the quality of the project. A user-friendly interface, well-organized code and a consistent frame rate are all things that would assist in heightening the level of quality the final build possesses. As well as this, accessibility was touched on, bringing the idea that the game must be playable by players of all skill levels. These non-functional requirements are crucial in ensuring the success of the game as they all directly impact and alter the experience the player has when playing.

The analysis of the project's feasibility demonstrated how viable the project was from a technical standpoint, outlining the applications chosen as well as the potential challenges they may produce. Unity, Visual Studio Code, Piskel and Epidemic Sound were all selected specifically for their suitability, efficiency and, most importantly, their compatibility with the concept. Although potential issues such as optimization and complex enemy AI implementation were identified, they were deemed to be manageable as long as the development process was met with careful planning and in depth testing. GitHub and Trello were chosen for version control and project management respectively in order to further support the feasibility of the project, ensuring that the development process was structured correctly and managed closely.

In short, this chapter has laid the foundation for the development of the game, focusing on user-centric functional requirements, non-functional considerations and a sound feasibility assessment. The requirements and planned technologies established provide a great foundation for both the design and implementation phases and ensure the creation of a high quality and nostalgic 16-bit title.

4 Design

4.1 Introduction

This chapter will detail the design of the project, building upon both the functional and non-functional requirements specified in Chapter 3. The primary purpose of the design phase is to turn those requirements into a fleshed out blueprint that will be used to create the game's structure and general player interaction. The overall aim is to ensure that the final game meets the identified needs and goals in an effective manner, particularly when it comes to evoking that desired feeling of nostalgia in the player as they progress through the game.

The design of the game is divided into two key areas:

- **Program Design:** This surrounds the underlying structure and systems that govern how the game functions including the core design of the game. Examples of this could be player movement, enemy AI, level progression and combat. It will also address how the game states are managed as well as the overall architecture of the game's logic.
- **User Interface/Player Experience Design:** This area focuses on how the player themselves interact with the world and its systems. This includes the responsiveness of the controls, the way information is presented visually, and the overall flow of the player's experience. This aspect of development is crucial in achieving the aforementioned needs and goals.

The application for this project is a top-down 2D adventure game set within a temple. The player controls an adventurer armed with nothing but a sword as they navigate the environment, uncover secrets and battle enemies. The game places heavy emphasis on both exploration and combat, giving the player freedom to progress however they please while giving them a satisfying combat experience akin to those classic games from the 90s. This chapter will outline how those core elements are both structured and presented to the player in order to create an engaging and cohesive experience that meets the requirements of the project.

4.2 Program Design

The program design section focuses on providing the game's systems and logic a structure that is built around maintaining an efficient coding environment. The aim is to use this structure to create a modular and easy to understand architecture that simplifies the development process and allows for easy modifications.

4.2.1 Gameplay Loop

The game will follow a fairly simple gameplay loop, consisting of the following stages;

- **Initialization:** Setting up the initial game states and core systems as well as loading the various assets such as the sprites and audio.
- **Input Handling:** Processing input provided by the player through either a keyboard or controller in order to produce the intended actions.
- **Update:** Updating the game state based on the input provided by the player, time progression and general game logic such as the physics and enemy AI. This stage is where the majority of the game's calculations and logic execution will occur.
- **Rendering:** Visualizing the current game state on the screen including characters, UI elements and the backgrounds.

4.2.2 Object Management

All entities within the world will be managed using a system that allows for an efficient creation, updating and destruction process. This system involves component-based architecture that uses reusable behaviours such as `SpriteRenderer` components as well as a focus on OOP scripting. This approach to object management promotes modularity and reduces the chance of code duplication throughout the development process.

4.2.3 Scene Management

The different areas/screens in the game such as the rooms and menus will all be organized into scenes. A scene management system will be used to handle both the loading and unloading of scenes while transitioning to the next state. This system ensures a smoother game flow for the player.

4.2.4 Input System

A dedicated input system will be implemented to handle the specifics of input devices. This system will map physical inputs to in-game actions, making it easier to add support for different input devices in the future without having to make any significant changes to the code.

4.2.5 Physics & Collision Detection

A 2D collision detection system will be implemented to handle interactions that occur between game objects. This system will determine when objects overlap or collide and will trigger an appropriate response such as damage or preventing movement.

4.2.6 Game State Management

A system will be put in place to track each game state as well as which one is active at any given time including player health, inventory, death conditions and overall progression. This system will allow the game to be more flexible and elevate the player experience during gameplay.

4.2.7 Enemy AI

Enemy behaviour will be implemented through dedicated AI systems. This system involves defining different enemy types with their own specific movement patterns, attacks and reactions to the player. This AI should be designed to provide a challenging yet fair experience for the player.

4.2.8 Level Design Integration

The level design system will be integrated into the game through a system that both loads and renders the environment. The environment itself will be built using Unity's Tilemap Editor. The system will also handle object collisions through the use of a layer system within Unity itself.

By structuring the game in such a way, the development process will be much more manageable and will allow for an easier debugging experience as well as allowing easier implementation of new features. This overall design is built to prioritize the maintainability and clarity of the codebase throughout the development process.

4.3 User interface design

This section describes how the interface is designed. The section will differ depending on whether an app or a game is being developed.

This section will outline how the game's interface is designed, focusing on how information is presented to the player and how they will interact with the world. Given the nature of the game, the interface will prioritize direct interaction with the world and will aim for a clean, uncluttered experience.

4.3.1 Visual Presentation

The game's world is the primary visual interface that the player will see, rendered in a 2D pixel art style. Key information will be integrated into this render where possible in order to minimize the possibility of breaking immersion for the player, examples for this include:

- **Player Character:** A defined sprite with an animation tree that conveys actions such as moving, attacking, and taking damage in a clear and easy to understand manner.
- **Enemies:** Distinct sprites with animation trees that indicate their current state and attack animations.
- **Environment:** A tile-based world that is visually consistent with the desired 16-bit aesthetic, providing both clear pathways and appropriate obstacles.

4.3.2 Heads-Up Display

Essential information will be displayed in real time through an unintrusive HUD. Examples for this include:

- **Player Health:** A simple visual representation of the player's health in the form of a heart counter in the top left corner of the screen. It should be positioned in a way that is easily visible but not in a way that affects the player's experience while playing.
- **Item Display:** If additional items are added beyond just keys, it may be worth adding a small section in the corner of the screen that shows what items the player holds and how many of each they possess.

4.3.3 Input Feedback

Player input should be responsive and easy to understand. Visual feedback will be provided for actions such as enemies flashing upon being hit and various animations such as opening a chest or swinging the sword.

4.3.4 Menus

The menus throughout the game will be kept simple and remain in line with the retro aesthetic that the game is aiming for. This will be done through the use of pixelated fonts and chiptune music in the background as well as 16-bit style selection noises when the player hits a button in the menus.

4.3.5 World Interaction

All interaction with the world such as opening doors or interacting with objects will be context-sensitive and visually indicated in a clear way. For example, a subtle context clue prompt could pop up over the character's head when standing near an interactable object, or the player sprite will change when performing certain actions.

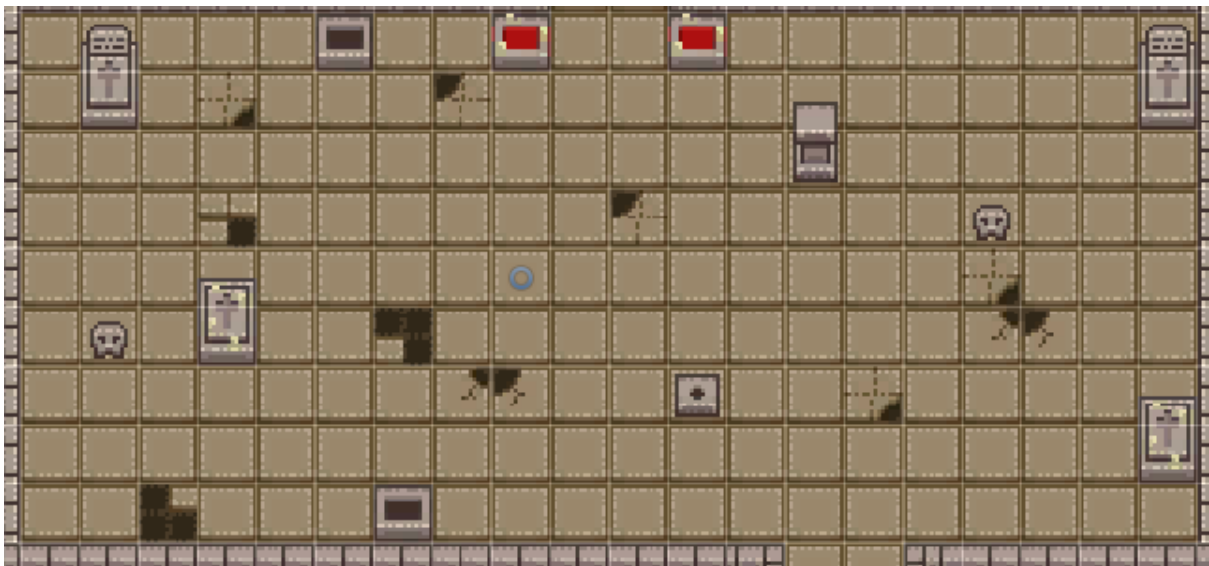
Overall, the goal of the UI's design is to be informative and functional without distracting the player or taking away from their overall experience with the game. Clarity is key when designing the user interface, ensuring that the focus remains on both the action and exploration in the world.

4.3.6 Storyboard

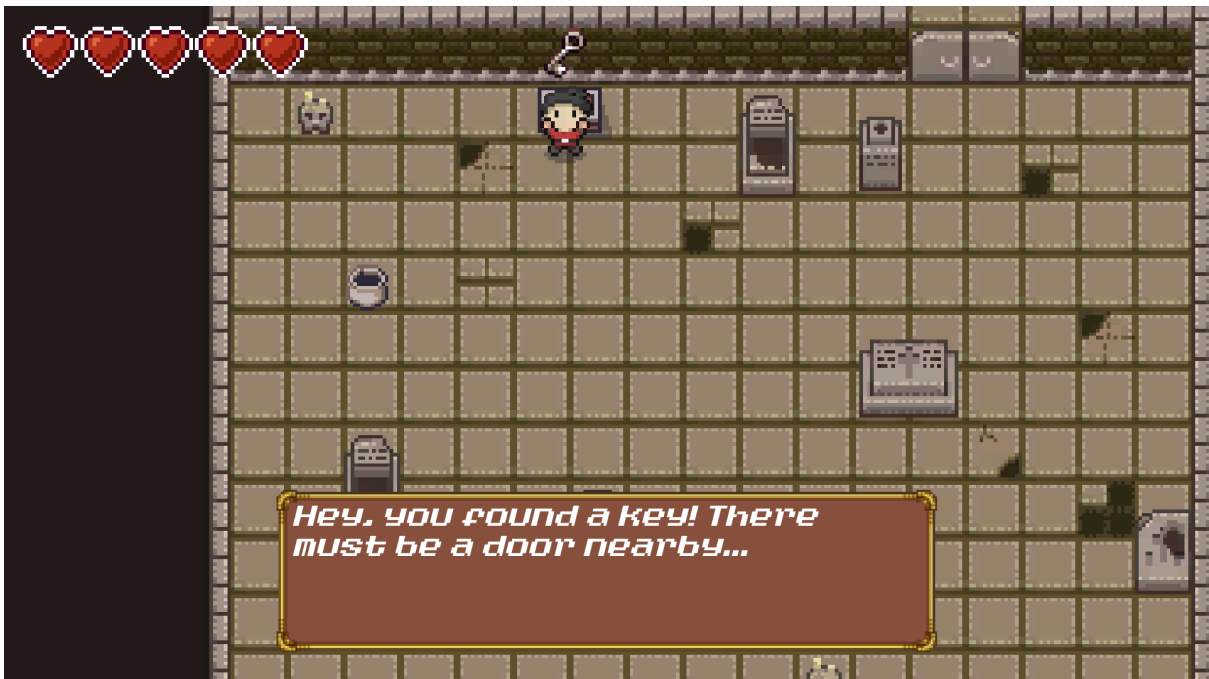
Spawn:



Level Design:



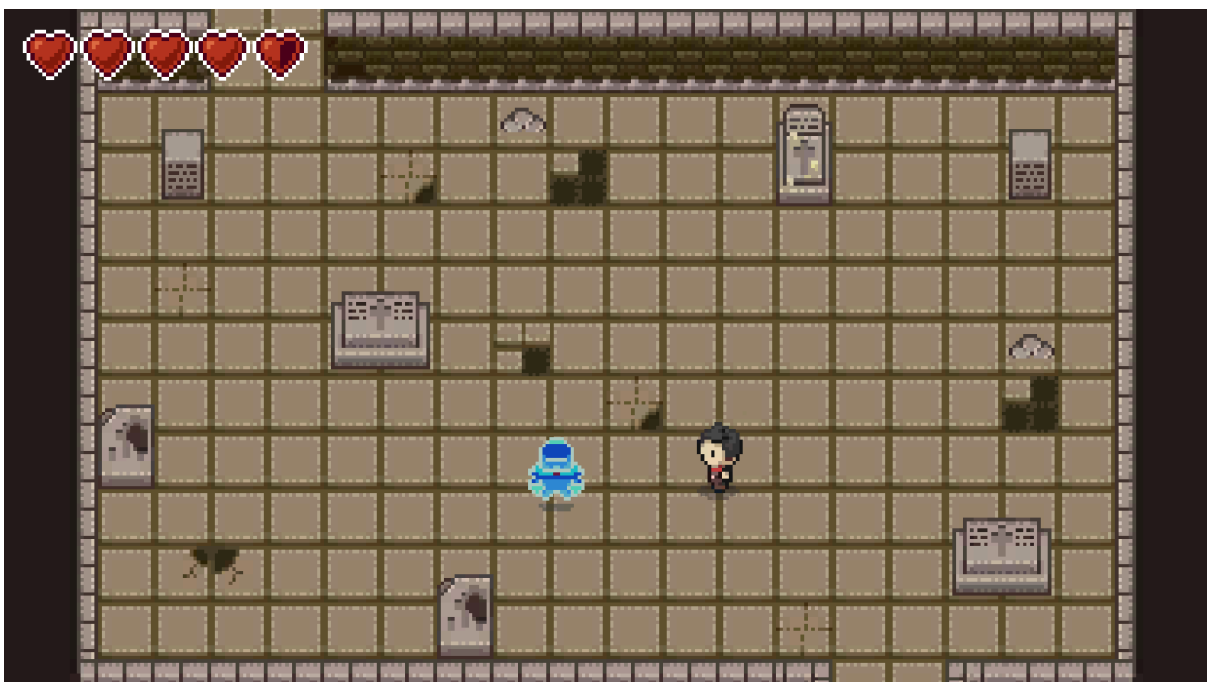
Open Chest:



Unlock Door:



Enemy Combat



Enemy Room:



4.3.7 Level Design

The game isn't exactly split up into 'levels', instead, it is one big level that the player can explore. The room transitions are handled through the Cinemachine plugin for Unity however the entire temple has been fit into a single scene. This has been done to remove load times and to ensure the flow of the game is the best it possibly can be for the player.

4.3.8 Environment

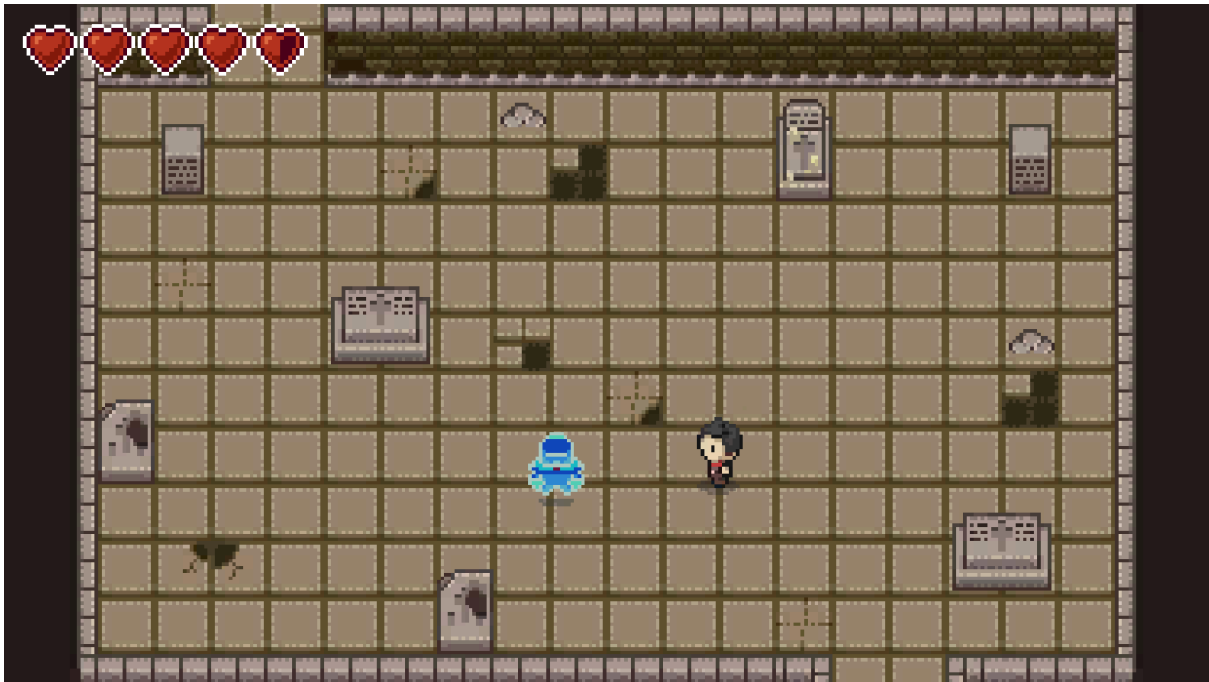
The game puts the player in a mysterious temple that they must explore. The aesthetic of the temple is made to look as if it's been abandoned for some time now.



The player can interact with a few objects including a chest that they find a key in. They can then use this key to escape from the spawn room.



Upon escaping from the spawn room the player will find themselves face to face with multiple paladin-like enemies wearing blackened suits of armor. This is the theme for the rest of the game as the player will move through the rest of the temple fighting these entities in an attempt to survive the onslaught.



4.4 Conclusion

This chapter detailed the design framework for the project, translating the aforementioned requirements into a blueprint for use during development. The design is primarily segmented into Program Design and Player Experience, expanding on both the technical architecture and the way that the player interacts with the world.

The Program Design section outlines the core systems that will be used to drive the game including the gameplay loop, a component based object management system, a scene manager, a dedicated input system, a 2D collision detection system, a game state manager, enemy AI and the integration of level design. This structured approach ensures a modular, maintainable, and efficient codebase throughout the development process.

The section on Player Experience aims to lay out a structure that aims to prioritize a clean and immersive experience. Key information will be integrated directly into the world with a minimal HUD that displays essential elements such as player health and held items. Controls will be responsive and boast clear visual feedback with simple, retro style menus and context-sensitive world interactions that are all designed to enhance player experience, not take away from it. The game features an interconnected temple in a single scene in order to eliminate load times and improve the flow of the game. The gameplay loop involves unlocking the door to escape the first room using a key found in a chest and then navigating the rest of the temple while battling any enemies that the player crosses paths with.

In short, this chapter laid out the design plan that expands on both the technical and experiential aspects of the project. The outlined structure and considerations for the user interface aim to create an engaging and cohesive experience that both understands and meets the requirements of the project and delivers that desired sense of nostalgia for the player.

5 Implementation

5.1 Introduction

This chapter details the implementation process for the project that will be used to bring the design plan outline in the previous chapter to life. The game has been developed with the use of the following key technologies:

- **Unity:** Unity is a game engine known for its 2D capabilities, visual development environment, and C# scripting. It provides the foundation for the structure, physics, collision detection, and asset management within the game. The integrated tilemap editor was crucial in creating the environment that the player moves around, and its animation system allowed for the creation of dynamic character and enemy sprites.
- **Visual Studio Code:** Visual Studio Code was used as the primary IDE for scripting in C#. Its debugging tools, lightweight nature, and seamless integration with Unity allowed for an efficient and organized development process for all of the game's logic including the enemy AI, player controls and the management of game states.
- **Piskel:** Piskel is a free online sprite editor that was used in the creation of some of the visual assets present in the game. As a software that boasts both an intuitive interface and specialized tools it allowed for the creation of multiple authentic looking sprites that matched the desired aesthetic for the game.
- **Epidemic Sound:** Epidemic Sound is an online library of royalty-free music and sound effects that fueled the game's audio design. The selection of chiptune audio offered was crucial in recreating that feeling of nostalgia and aided in immersing the player into the world.

The application for this project is a top-down 2D adventure game set within a temple. The player is put in control of a lone adventurer armed with nothing but a sword as they navigate through the temple battling enemies.

5.2 Scrum Methodology

As a solo developer, the use of SCRUM methodology provided an invaluable foundation and structure throughout the implementation of the game. By adopting the core principles associated with the methodology, the project was able to benefit from a clear roadmap throughout its eight distinct sprints. In this case, the product backlog served as a dynamic list of assets and game features that were constantly being refined based on the development progress and various insights that emerged throughout. Planning sprints at the

beginning of every two week cycle allowed for focus on a smaller array of tasks, ensuring a more concentrated development experience.

Daily self-reflection became a crucial practice throughout the development cycle in order to maintain momentum and identify potential roadblocks early on. By reviewing the previous day's work and outlining the tasks for the day ahead, it created a sense of accountability and accomplishment. Further reviews at the end of each sprint provided an opportunity to assess the implemented features and ensure they were in line with the project's scope and requirements. These reviews were crucial in identifying what areas were in need of improvement.

The sprint retrospectives were very beneficial in optimizing the workflow throughout the development process. Reflecting on both the successes and challenges of each individual sprint allowed for continuous improvement in both time management and overall development strategies. For example, challenges encountered during the Feasibility sprint directly formed the planning and execution of the sprints that came after. This led to a much more streamlined process.

The structured nature of the SCRUM methodology in this solo project allowed for a more organized and adaptable development process. The sprints being divided up into two week long segments ensured a more focused style of development and allowed for a more concentrated level of refinement based on the ongoing testing stages. For example, personal playtesting that was conducted following the Character Design sprint formed the eventual movement and animations implemented in the Mechanic Design sprint.

In short, the use of SCRUM methodology in a solo setting provided the project with the structure and feedback loops needed to ensure that the game met the project scope and made for a focused and refined development process from start to finish.

5.3 Development environment

Visual Studio Code is the primary IDE utilized during the development of the project. VS Code's lightweight nature, combined with its C# support and pre existing integration into Unity made it the ideal choice for scripting. Its debugging tools and seamless integration with version control tools significantly streamlined the coding process. The ability to navigate through project files with ease, refactor code and identify errors within scripts contributed to a more organized and efficient workflow throughout the development process.

Git was employed for use as the version control system for the project, with GitHub serving as the remote repository. The use of Git ensured that the tracking of all code changes and storage of the project in a secure location was made simple and seamless. Regular patch-note style commits were made to document the evolution of the project throughout the various sprints with descriptive comments ensuring that the implemented changes were easy to follow and understand. GitHub served as the central remote location for the project files. Using Git for version control proved to be invaluable in maintaining the integrity of the code and managing the development history of the project.

5.4 Sprint 1 - Research & Concept

5.4.1 Goal

The first sprint was focused around the following goals;

- Research similar games
- Identify a solid concept for what the project should be

5.4.2 Item 1 - Research

In order to gather information, a research paper was written in Term 1 during the Research & Analytics module. This was an extremely important phase as it ensured that the project wouldn't be developed blindly or without foundation. It also allowed for both a deeper understanding and easier implementation of the common niches found in most games.

This section was code free as it was before Unity was even opened for the first time, but it provided a lot more clarity on the technical scope of the project which will be touched upon more in the following sprint, 'Feasibility'.

5.4.3 Item 2 - Concept

Following the research phase the concept was then ironed out. The loose concept for the game was fueled by a past idea for a passion project and was finally expanded upon through the use of the research. Nostalgia was to be a big focus for the project, as the nature of the game would make it extremely easy to invoke those feelings in the player. As a result, the aforementioned research paper spoke a lot on nostalgia as an emotion and went on to form the foundation for the overall scope and goals of the project.

The concept began as a simple and loose idea focused around recreating a very popular Zelda title, 'Minish Cap', and eventually evolved into what the project would become: A 2D adventure game focused on invoking that feeling of nostalgia in the player.

5.5 Sprint 2 - Feasibility & Requirements

5.5.1 Goal

The goal of the second sprint was as follows;

- Analyse the feasibility of each area of the game
- Break that feasibility down into easy to read steps

5.5.2 Item 1 - Core Engine Proficiency

Originally, Godot was to be considered the primary game engine for developing the game. Upon delving deeper into the available libraries and general technologies available however, it was decided that Unity would instead serve as the core engine. Unity as an engine is the more feasible choice for the game given its strong 2D development capabilities.

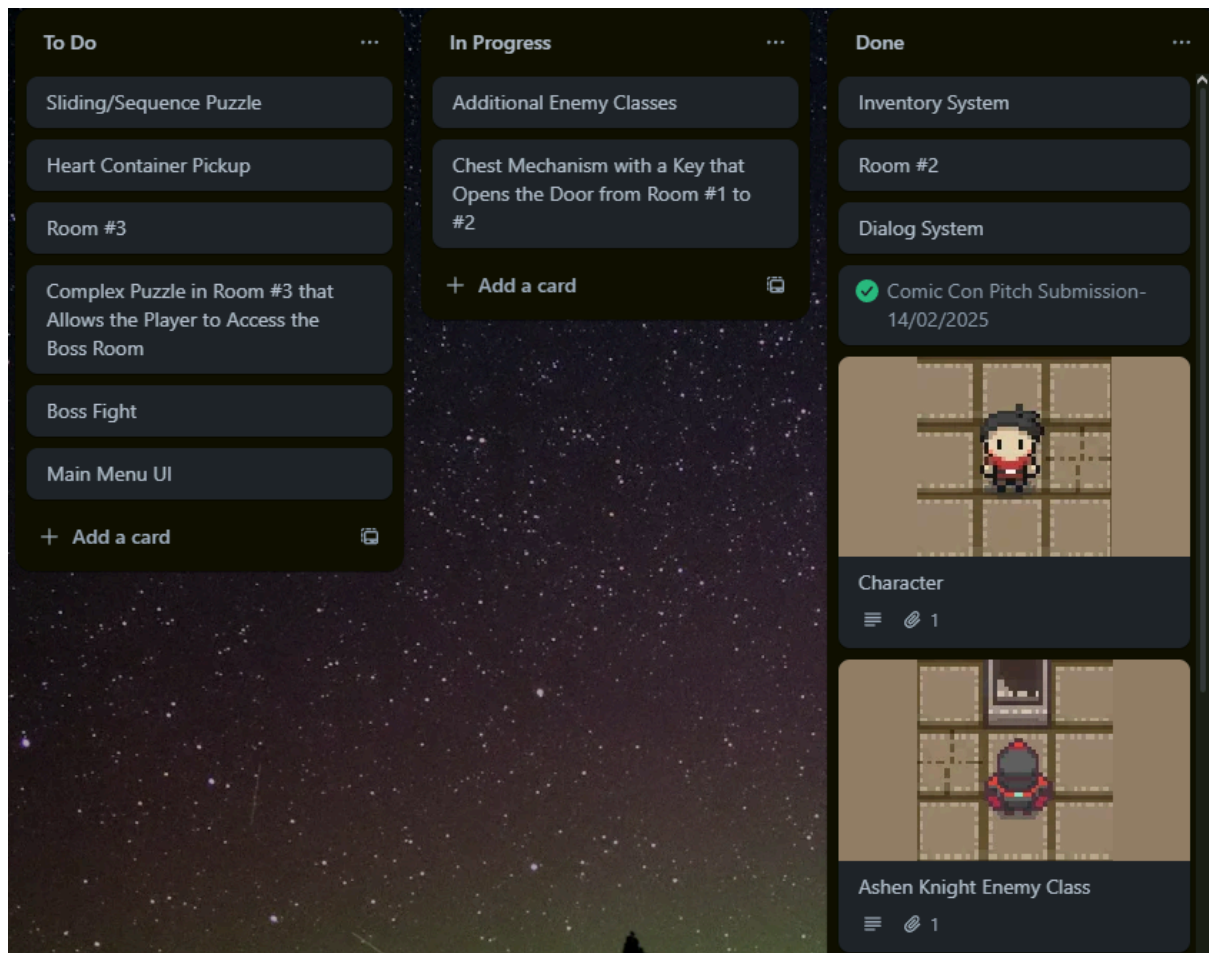
	Unity	VS	Godot
Parameters	Unity		Godot
Definition	It is a cross-platform game engine used to develop video games.		It is a free, cross-platform, game engine for developing video games released under the MIT license.
Compatibility	Unity supports macOS, Windows, Linux, CentOS, and Ubuntu.		Godot supports windows, Free BSD, macOS, Open BSD.
Launched By	It was launched by Unity technologies in 2004.		It was launched by Juan Linietsky and Ariel Manzur in 2014.
Supported Languages	Unity natively supports C#, along with other alternative languages like Javascript.		Easy learning curve
Popularity	More Popular		Less Popular
Asset Store	Extensive asset store		Assets marketplace is smaller
Graphics Quality	Better graphics quality		Poor graphics quality

5.5.3 Item 2 - Asset Creation

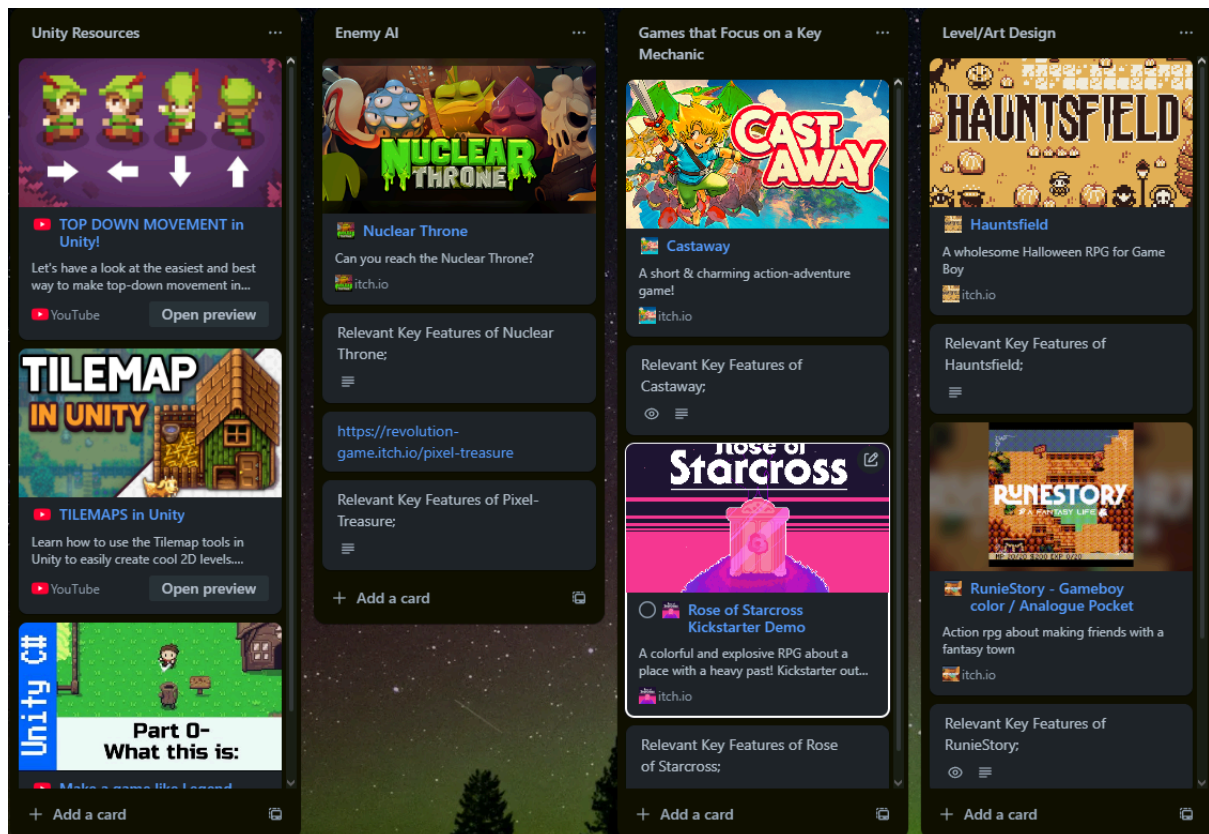
The choice to use Piskel and Epidemic Sound for asset and audio design respectively was a choice fueled by each software's accessible and manageable tool set fit for a solo developer. As well as this, their pre-existing compatibility with Unity made it a no-brainer as the connection minimized technical hurdles.

5.5.4 Item 3 - Core Mechanics

The next step was to ensure that the planned mechanics such as player movement, enemy AI and combat were actually feasible under the timeframe. This is where the likes of Trello came in to organize the tasks ahead and track them to ensure completion. Keeping Unity's scripting capabilities and established game development principles in mind, it was deduced that the feature list would be feasible.



A big use that Trello had during the early days of the project's development was noting down games that were focused on certain things as well as tutorials that may become useful down the line. These focuses included things such as Enemy AI, Level and Art Design as well as games that focused on a Key Mechanic of sorts, although more on those mechanics in Sprint 3.



This is where Trello really proved its merit, as without it the research phase would've been a million times more difficult and nowhere near as organized.

5.5.5 Item 4 - Level Design and Integration

The next step was figuring out what scale level would be feasible under the timeframe. It needed to be something that the player could explore but not so big that it would sacrifice quality under the time constraints. It was ultimately decided that the map would be a medium size consisting of a few rooms that the player could navigate between while playing.



5.5.6 Item 5 - Performance Optimization

Optimization was an ongoing focus throughout the entire development process. Achieving a playable and enjoyable frame rate with the intended feature list is only achievable with careful asset management and efficient scripting practices. If the game cannot achieve an ideal frame rate it will directly impact the player experience which is a critical aspect of the final product given the focus on invoking that sense of nostalgia in the player.

5.5.7 Item 6 - Time Commitment

Utilizing SCRUM methodology and dividing the game into eight two-week long sprints will prove invaluable to the development process. Given the scope of the project, allocating sufficient time to each key area is extremely important as lacking on even one of them can cause a domino effect to the other areas. Upon analyzing and bringing the scope closer to its final form, it was deduced that the eight sprints should be more than enough to cover the requirements and feature list.

Considering the solo nature of the project, time management becomes even more important to the success of the development process. By focusing on the core mechanics as well as implementing a single, interconnected level, it will ensure that the design will be manageable within the capacity of a solo developer.

5.6 Sprint 3 - Key Mechanic

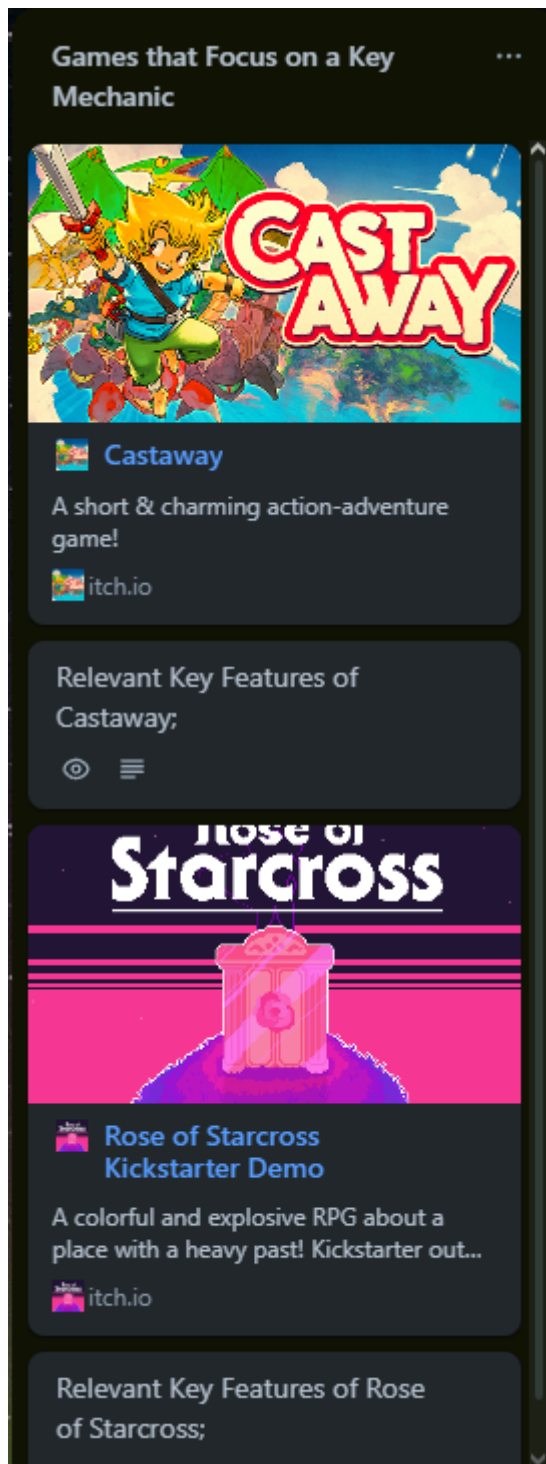
5.6.1 Goal

The goal of the third sprint was to figure out how to make the game unique through the implementation of a key mechanic.

5.6.2 Item 6 - Key Mechanic Brainstorming

During the initial research phase for this sprint there were many different mechanics considered. Among these were an Undertale style combat minigame system, a parry system and charged sword attacks. The minigame system was deemed unsuitable as it ran too far out of the project scope and would stick out. The parry system was a mechanic that was heavily considered however given the timeframe the project had to work under it didn't seem feasible. There is a good chance that if implemented it would come out rushed and end up being frustrating for the player, hindering the player experience and breaking immersion if it wasn't a smooth or reliable feature during combat.

Trello was a big help in brainstorming and recording all of these ideas as it served as a board where games that served as inspiration could be pinned and saved for further analysis.




The interactive process here involved multiple periods of brainstorming and prototyping in the game engine itself. Several mechanics were implemented briefly and tested, with each one being evaluated thoroughly for how well it would integrate with the gameplay loop and what it would bring to the player experience. It also involved a critical assessment of how these potential mechanics would interact with existing enemy designs and the overall exploration of the world.

5.7 Sprint 4 - Prototype

5.7.1 Goal

After spending the first few sprints gathering research and finalizing the overall scope of the project it was finally time to start the development process. The goal of this sprint was to create a loose grey box prototype.

The overall aim here was to create a simple prototype in which the player could move around a blank world. Below is a short video that shows the final prototype that this sprint produced.

 Major_Project_Tech_Demo_-_SampleScene_-_Windows_Mac_Linux_-_Unity_2021.3....

The player model was the focus of this prototype. The idea here was that if the character was developed before anything else it could be dropped into any environment and would mean that features could be developed and implemented with the specific player model in mind instead of just a greybox placeholder.

Following the design of the player sprite the next step was to begin scripting.

5.7.2 Item 1 - Player Movement

At the start of the script, the foundation for a simple state machine was created. This will be used to manage all functionality for the player and streamline it in future scripts.

```
16 references
10  public enum PlayerState
11  {
12      | 3 references
13      | 2 references
14      | 4 references
15      | 3 references
16      | 3 references
    walk,
    attack,
    interact,
    stagger,
    idle
```

Within the Fixed Update, raw values are pulled and read from Horizontal and Vertical axes.

```
58 |         positionChange.x = Input.GetAxisRaw("Horizontal");
59 |         positionChange.y = Input.GetAxisRaw("Vertical");
```

If movement input is detected and the player is in either the walk or idle state, 'UpdateAnimationAndMove()' is called.

```
66 |         else if (currentState == PlayerState.walk || currentState == PlayerState.idle)
67 |         {
68 |             UpdateAnimationAndMove();
69 |         }
70 |
71 |     }
```

UpdateAnimationAndMove() updates the parameters required by the animator based on the input provided and then calls 'MoveCharacter()'.

```
113 |     void UpdateAnimationAndMove()
114 |     {
115 |         if (positionChange != Vector3.zero)
116 |         {
117 |             MoveCharacter();
118 |             animator.SetFloat("moveX", positionChange.x);
119 |             animator.SetFloat("moveY", positionChange.y);
120 |             animator.SetBool("moving", true);
121 |         } else {
122 |             animator.SetBool("moving", false);
123 |         }
124 |     }
```

MoveCharacter() simply normalizes the vector and utilizes the attached Rigidbody2D to move the character model.

```
126 |     void MoveCharacter()
127 |     {
128 |         positionChange.Normalize();
129 |         playerRigidBody.MovePosition(transform.position + positionChange * speed * Time.deltaTime);
130 |     }
```

Item 2 - Player Attack

Now that the movement functionality had been set up, the next step was to begin implementing some attack functionality for the player. The goal for this was to enable the player to start up an attack animation and play a corresponding sound effect to go with it.

Within the Fixed Update, there is a check for an 'attack' input. This will also only trigger as long as the player is not already attacking or in a stagger state.

```
61 |         if (Input.GetButton("attack") && currentState != PlayerState.attack && currentState != PlayerState.stagger)
62 |         {
63 |             StartCoroutine(AttackCo());
64 |         }
```

AttackCo() is a coroutine that performs the following in sequence:

- Sets the 'attacking' boolean to true. This is a boolean that triggers an animation state in the blend tree present in the Unity editor.
- Sets the current player state to 'attacking'.
- Plays the appropriate audio alongside the attack if assigned in the editor.
- Pauses for a single frame using 'yield return null'
- Resets the attacking boolean back to false.
- Pauses for a further 0.3 seconds to prevent the player from attacking non stop.
- Resets the player state back to walk as long as the player is not interacting with anything.

```
73 | private IEnumerator AttackCo()
74 | {
75 |     animator.SetBool("attacking", true);
76 |     currentState = PlayerState.attack;
77 |
78 |     if (swordSwing != null)
79 |     {
80 |         swordSwing.Play();
81 |     }
82 |
83 |     yield return null;
84 |     animator.SetBool("attacking", false);
85 |     yield return new WaitForSeconds(0.3f);
86 |
87 |     if(currentState != PlayerState.interact)
88 |     {
89 |         currentState = PlayerState.walk;
90 |     }
91 | }
```

5.7.3 Item 3 - Damage and Knockback

Now that basic attack functionality has been implemented, damage is next. This next part of the script is set up to handle the player taking damage, reduction of their health, triggering both visual and audio based feedback using a signal system and applying a brief knockback effect. As well as this, simple game over functionality must be implemented for when the player's health hits zero. The first piece added to the script is the Knock() function.

- The Knock() function takes two floats, knockTime and damage, as parameters.
- The damage float is used to reduce currentHealth.RuntimeValue.
- playerHealthSignal is then raised, this is the signal that updates the heart count visible to the player in the UI.
- If the player still has health, the KnockCo() coroutine is started
- If the player's health reaches zero, the game object is deactivated

```
132     public void Knock(float knockTime, float damage)
133     {
134         currentHealth.RuntimeValue -= damage;
135         playerHealthSignal.Raise();
136         if(currentHealth.RuntimeValue > 0)
137         {
138             StartCoroutine(KnockCo(knockTime));
139         }else{
140             this.gameObject.SetActive(false);
141         }
142     }
143 }
```

KnockCo() is a coroutine that raises the playerHit signal, a signal that handles the visual and audio feedback for taking damage. If the playerRigidBody exists, it will pause based on the value of knockTime and then set the velocity of the rigid body to zero before setting the current player state to idle.

```
145     private IEnumerator KnockCo(float knockTime)
146     {
147         playerHit.Raise();
148
149         if(playerRigidBody != null)
150         {
151             yield return new WaitForSeconds(knockTime);
152             playerRigidBody.velocity = Vector2.zero;
153             currentState = PlayerState.idle;
154             playerRigidBody.velocity = Vector2.zero;
155         }
156     }
```

5.8 Sprint 5 - Level Design

5.8.1 Goal

Following the creation of a character model with both movement and attack, the next step is to create a level for them to roam.

5.8.2 Item 1 - Visual Design

As mentioned previously, the game takes place in an interconnected temple. It was extremely important that the correct assets were chosen here to align with the game's desired retro feel. Many of the game's assets were sourced from a paid package created by a developer known as ElvGames, an extremely talented 2D artist in the indie game space.

<https://www.gamedevmarket.net/asset/crypt-16x16-pixelart-tileset-rogue-adventure>



The reasoning behind this choice stems from the requirements of the game. These assets directly align with the desired retro feel of the game as well as processing the ability to invoke a sense of nostalgia in the player.

This is another time when the use of Trello became particularly useful. Having a board where games with a focus on art direction could be pinned and analysed further became invaluable to making a final decision on assets as it provided a much simpler and streamlined way to gather inspiration.

5.8.3 Item 2 - Map Design

A big part of designing the world itself was ensuring that it was balanced in terms of scale. The level had to be big enough to make the player feel like they had something to explore, but not so big that they felt as if they were doing more walking than actual playing.

As touched on briefly, the decision was made to have the map consist of a handful of medium size rooms as seen below.



5.9 Sprint 6 - Mechanic Design

5.9.1 Goal

Now that the character model and level had been implemented it was time to focus on some mechanics so that the player wouldn't just be wandering around an empty world.

5.9.2 Item 1 - Chest

The treasure chest located in the first room primarily relies on a single script. The first step was to make sure the chest could detect when the player is in range and could allow interaction via a certain input in order to both open the chest and retrieve the contents within.

The Update() function checks if the 'Interact' button has been pressed. As well as this it checks if the playerInRange boolean is set to true. This boolean is from a secondary script called Interactable which is a simple script that is attached to all interactable objects that handles things such as player range and the context clue signals.

```
25     void Update()
26     {
27         if(Input.GetButtonDown("Interact") && playerInRange)
28         {
29             if(!isOpen)
30             {
31                 OpenChest();
32             } else{
33                 ChestAlreadyOpen();
34             }
35         }
36     }
```

If the chest is not already open, OpenChest() is called, but if the chest is already open then ChestOpenAlready() is called. OpenChest() displays the description of the item in the chest on screen for the player to see and adds the item to the player's inventory. As well as this, a signal triggers for the character to visually turn around and raise the item to the camera, plays the appropriate sound effect and marks the chest as open. Here is a quick breakdown of this line by line.

Activates the 'dialogBox' game object that is used to display text on screen.

```
40     |         |     dialogBox.SetActive(true);
```

Sets the text for this dialog box to match the item's description of the chest's contents which is an 'Item' object.

```
41 | | dialogText.text = contents.itemDescription;
```

Calls the AddItem() function to the playerInventory, which is an 'Inventory' object, to add the contents of the chest to the player's inventory.

```
42 | | playerInventory.AddItem(contents);
```

Sets the current item in the player's inventory to the item that was just acquired.

```
43 | | playerInventory.currentItem = contents;
```

Triggers the raiseItem signal. This is the signal that tells the character model to turn around and raise the acquired item up to the camera, a classic animation in this genre of game.

```
44 | | raiseItem.Raise();
```

Triggers the context signal. This tells the game that the chest is no longer interactable and as a result should not provide a context clue if the player is within range anymore.

```
45 | | context.Raise();
```

Sets the isOpen boolean to true. This is done to prevent the chest from being reopened a second time.

```
46 | | isOpen = true;
```

This sets the 'opened' boolean on the chest's Animator component to true, which triggers the opening animation.

```
47 | | anim.SetBool("opened", true);
```


Finally, if assigned, the appropriate sound effect will play alongside the opening animation.

```
49         if (openChest != null)
50         {
51             openChest.Play();
52         }
```

As a whole, that is the OpenChest() function line by line. Handling a chest that has already been opened through the ChestAlreadyOpen() function is a very simple piece of code that is split into only two lines. The first line deactivates the dialog box on screen, and the second line triggers the raiseItem signal for the second time. This causes the character model to lower the item and return to its original visual state.

```
55     public void ChestAlreadyOpen()
56     {
57         dialogBox.SetActive(false);
58         raiseItem.Raise();
59     }
```

5.9.3 Item 2 - Door

In order to use the key retrieved from the chest on the door to escape from the first room, the Door script checks if the interact button has been pressed while the player is in range. As well as this, it checks if the door is of the 'key' type, a type that is defined in an enum at the top of the script.

```
5     public enum DoorType
6     {
7         1 reference
7         key,
8         0 references
8         enemy,
9         0 references
9         button
10    }
```

```

26         if (Input.GetButtonDown("Interact"))
27         {
28             if(playerInRange && thisDoorType == DoorType.key)
29             {
30                 if(playerInventory.numberOfKeys > 0)
31                 {
32                     if (doorOpen != null)
33                     {
34                         doorOpen.Play();
35                     }
36                     playerInventory.numberOfKeys--;
37                     Open();
38                 }
39             }
40         }

```

If the conditions are met for the if statement, it checks if the player has a key in their inventory.

```

if(playerInventory.numberOfKeys > 0)

```

If the player has a key, it will then play the appropriate sound effect if assigned, then decrease the player's key count and call the Open() function to open the door.

```

30         if(playerInventory.numberOfKeys > 0)
31         {
32             if (doorOpen != null)
33             {
34                 doorOpen.Play();
35             }
36             playerInventory.numberOfKeys--;
37             Open();
38         }

```

The Open() function opens the door by disabling both the sprite and its assigned collider before setting the open boolean to true.

```
43     public void Open()
44     {
45         doorSprite.enabled = false;
46         open = true;
47         physicsCollider.enabled = false;
48     }
```

In turn, the Close() function does the exact same thing but in reverse, turning everything back on and setting the open boolean to false.

```
50     public void Close()
51     {
52         doorSprite.enabled = true;
53         open = false;
54         physicsCollider.enabled = true;
55     }
```

5.9.4 Item 3 - Enemy Room

The enemy room is a mechanic that has been in countless games. Upon the player entering the room, the doors close behind them and only reopen when all enemies have been defeated.

Using OnTriggerEnter2D, the coroutine ActivateAfterDelay() is run following a short delay after the player has entered the trigger collider of the room.

```
41     public override void OnTriggerEnter2D(Collider2D other)
42     {
43         if (other.CompareTag("Player") && !other.isTrigger)
44         {
45             StartCoroutine(ActivateAfterDelay());
46         }
47     }
48 }
```

ActivateAfterDelay() activates the virtual camera for the room before pausing for the specified time.

```
52         virtualCamera.SetActive(true);  
53         yield return new WaitForSeconds(activationDelay);
```

It then iterates through the array of enemies and pots, activating their game objects in the hierarchy. CloseDoors() is then called to close all of the doors that are connected to the room.

```
56         for (int i = 0; i < enemies.Length; i++)  
57         {  
58             ChangeActivation(enemies[i], true);  
59         }  
60         for (int i = 0; i < pots.Length; i++)  
61         {  
62             ChangeActivation(pots[i], true);  
63         }  
64         CloseDoors();
```

EnemiesActive() is a function that both counts and returns the amount of enemies currently active in the hierarchy from the enemies array. It loops through the array and adds up all active enemies in the activeEnemies int. It will then return the final count once it reaches the end of the array.

```
20     public int EnemiesActive()  
21     {  
22         int activeEnemies = 0;  
23         for (int i = 0; i < enemies.Length; i++)  
24         {  
25             if (enemies[i].gameObject.activeInHierarchy)  
26             {  
27                 activeEnemies++;  
28             }  
29         }  
30         return activeEnemies;  
31     }
```

CheckEnemies() is a function that is used to verify if the enemy count has reached zero. This is the function used to determine when the doors should reopen. It starts by calling EnemiesActive() to get the current count of enemies and calls OpenDoors() if the number returned is zero.

```
33     public void CheckEnemies()
34     {
35         if (EnemiesActive() == 0)
36         {
37             OpenDoors();
38         }
39     }
```

CloseDoors() is the function used to close the doors when the player enters the room. It loops through a 'doors' array and calls Close() for each one, playing the appropriate sound effect alongside it if it's been assigned.

```
87     public void CloseDoors()
88     {
89         for(int i = 0; i < doors.Length; i ++)
90         {
91             doors[i].Close();
92         }
93         if (doorClose != null)
94         {
95             doorClose.Play();
96         }
97         Debug.Log("Close Doors");
98     }
```

OpenDoors() does the exact same thing but in reverse, looping through the array and opening all of the doors.

```

100     public void OpenDoors()
101     {
102         for (int i = 0; i < doors.Length; i++)
103         {
104             doors[i].Open();
105         }
106         if (doorOpen != null)
107         {
108             doorOpen.Play();
109         }
110         Debug.Log("Open Doors");
111     }

```

When the player exits the room it is handled with an OnTriggerExit2D function. This does the exact same thing as the OnTriggerEnter2D function but in reverse, setting all enemies and pots to false and turning off the virtual camera assigned to the room.

```

67     public override void OnTriggerExit2D(Collider2D other)
68     {
69         if (other.CompareTag("Player") && !other.isTrigger)
70         {
71             for (int i = 0; i < enemies.Length; i++)
72             {
73                 ChangeActivation(enemies[i], false);
74             }
75             for (int i = 0; i < pots.Length; i++)
76             {
77                 ChangeActivation(pots[i], false);
78             }
79             virtualCamera.SetActive(false);
80         }
81     }
82 }

```

5.10 Sprint 7 - Audio Design

5.10.1 Goal

The goal for this sprint was to pick out appropriate sound effects and background music for the game. This is extremely important as having a chiptune style soundtrack that matches the tone and visuals of the game is what will tie it all together and really give it that nostalgic feeling.

5.10.2 Item 1 - Audio Design

As mentioned multiple times throughout this report, Epidemic Sound served as the library from which all of the sound effects for the game would be sourced. Below is an example of the sound effects that Epidemic Sound offers.



It took a while to pick out the sound effects and background music due to it having to be something that really fit the tone. As well as this, there were a lot of sound effects needed including attack sounds, item pickup, both dealing and receiving damage among many others. In the end however the chosen audio ended up working extremely well and improved player experience and general game immersion massively.

The audio was all implemented using Unity's audio manager system. The sound effects were then called upon throughout scripts as shown in Sprint 6. An example of this can be seen below.

```
93 |         if (doorClose != null)
94 |         {
95 |             doorClose.Play();
96 |         }
```

Overall, although challenging, the audio design process of the game was extremely rewarding and vastly improved the quality of the game.

6 Testing

6.1 Introduction

The following chapter details the testing processes utilized in order to ensure the functionality, quality and overall positive player experience of the game. This phase was crucial in both identifying and addressing bugs and validating that the game met the requirements outlined earlier in this report, especially the idea of invoking a sense of nostalgia while providing an engaging gameplay experience for the player. There were three primary rounds of testing: functional testing, private user testing, and public user testing. These were all invaluable in gathering comprehensive feedback and helping to ensure that the final product was polished.

6.2 Functional Testing

6.2.1 Combat Mechanics

Test No	Description of test case	Input	Expected Output	Actual Output	Comment
01	Hit Detection (Player to Enemy)	Attack button when facing enemy	Enemy takes damage and gets knocked back	Enemy took damage and got knocked back successfully	
02	Hit Detection (Enemy to Player)	Player makes contact with the enemy.	Player takes damage and gets knocked back	Player took damage and got knocked back successfully	
03	Hit Detection (Enemy to Enemy)	Guide enemies to make contact with each other	Enemies pass through each other without doing any damage.	Initially, enemies did do damage to each other and often triggered a glitch in their transform properties that would send them	This was fixed by adding in a layer system that kept them from interacting with each other.

				shooting across the map.	
04	Damage Logic	Trigger damage both ways	The correct amount of damage gets applied upon contact	Both the player and enemy took the correct amount of damage	

6.2.2 World Interaction

Test No	Description of test case	Input	Expected Output	Actual Output	Comment
01	Chest Interaction	Player presses the interact button when in range of the chest	Chest opens with an animation and the character turns around to the camera holding the retrieved item	Initially, there was a bug that prevented the player from interacting with anything else in the world after opening the chest.	This was simply a sequence error in the script and was fixed relatively quickly.
02	Key Use	Player presses the interact button when in range of a door after opening the chest	The door should be unlocked and opened following the interaction. This also shows that the inventory system is functioning .	Door opened and allowed the player to pass through without issue.	
03	Object Interaction	Player presses the	A text box should pop up on	Initially, the text popup would	This was discovered to be a

		interact key when in range of an interactable object such as a gravestone or sign.	screen that details the writing on the sign or gravestone . The player should be able to remove this from the screen as well by pressing the interact button a second time.	remain stuck on the screen even if the player pressed the interact key a second time.	simple scripting error in the function that handled the dialog box and was fixed without much issue.
--	--	--	---	---	--

6.2.3 Game Progression/State

Test No	Description of test case	Input	Expected Output	Actual Output	Comment
01	Game Over	Player receives damage equivalent to their health	The player model should disappear and signal the game over state.	The player model disappeared and forced the player to restart the game.	
02	Enemy Room Functionality	The player enters the enemy room	The doors should close behind the player following a short delay.	Initially there was a bug where the doors would close before the player even made it into the room.	This was the first bug that presented significant difficulty in the development process. This bug was fixed with the addition of the delay

					timer that forced the doors to wait a few seconds before closing following the player's entry into the room.
03	Enemy Room Functionality Cont.	The player defeats all enemies in the room.	The doors should reopen following the death of all enemies.	Initially, there was an issue where the doors would not reopen, locking the player in the room.	This was an extremely difficult fix as it required rewriting a lot of the room's scripting and completely overhauling how it detected the enemies. An array system had to be implemented in which the script would check if the enemy objects in the array were still active in the hierarchy in order to deduce whether or not the doors should be opened.

6.2.4 Discussion of Functional Testing Results

The functional testing phase ended up being invaluable to the development process of the game. By testing features as they were added there were a lot of bugs discovered and fixed in a relatively short amount of time. This allowed the game to be more polished at each stage of development instead of having a lot of issues build up at the end. Some bugs, such as ones present in the enemy room, were extremely difficult fixes and had they not been discovered at the relatively early stage that they were, there's a high chance that they would still be outstanding even now in the final weeks.

6.3 User Testing

User Testing was split into two distinct rounds.

- Private Testing
- Public Testing

6.3.1 Private Testing

Private Testing was the first round of user testing done for the game. This round of testing was performed by friends, family and fellow students. This was done early on in development, around February, and as a result became invaluable in forming the foundation of the game. It was during this round that the interviews detailed in 3.2.2 were held.

This round primarily yielded a lot of quality of life related feedback. Testers observed that the movement felt a bit slippery at first and that the collision registration felt inaccurate, causing players to get stuck on the corners of various objects.

6.3.2 Public Testing

Public Testing was the final round of user testing conducted on the game and it took place on quite a grand stage: Comic Con Dublin 2025. The game was showcased alongside many extremely talented indie developers to tens of thousands of people over the course of a weekend in March. This provided a rare opportunity for a project to have an abundance of testers and helped greatly in identifying the areas that the game was struggling in but also excelling in.

The game received a lot of positive praise at the event, with testers speaking highly of its nostalgic feeling and fitting soundtrack. Testers also fell in love with the visual style, complementing the implementation of the visual assets and various smaller details such as the fonts and item sprites utilized.

Feedback was also given by both Black Shamrock, Larian Studios and 2K, three of the biggest game development studios in the world. They observed that although the game was lacking content in its early stage, it had a lot of potential:

2K Notes:

Love the visuals, very reminiscent of legend of zelda. The presentation was the perfect length. The game itself has a huge amount of potential, but I think 2 rooms with a little combat is just below the threshold of a full vertical slice. Would have liked to have seen more.

Black Shamrock Notes:

Solid foundation for a great game. Its quite light on the content side, but I highly recommend this be worked on more and resubmitted in august. If there was just a little more content at the same level of quality of what we saw in the presentation, this could be an easy inclusion.

Beyond this, there were a few key bugs discovered throughout the event. Testers noticed that when resetting the game the variables would not reset with it and so the attack powers of the enemies would double each, causing players to die in one hit after only the third reset. As well as this, enemy collisions were not always fluid with testers being unable to attack enemies if they were positioned in a specific way to the player or sometimes even being clipped through the map boundaries. It was the discovery of bugs such as these that really hammered home the reasoning behind doing the public user testing as they are bugs that likely wouldn't have been discovered had the project only gone through private testing.

6.4 Conclusion

This chapter has gone into detail about the three distinct rounds of testing conducted throughout the development process of the game. The first stage, functional testing, involved examining gameplay mechanics as they were implemented. This round involved testing of the combat system to ensure accurate hit detection and damage application, the functionality of world interactions such as opening doors and chests and finally, the core game progression and state management such as the intended behaviours of specific areas such as the enemy rooms. This first phase proved to be invaluable in both identifying and fixing a multitude of bugs early on in the development process which prevented them from building up and becoming a bigger problem toward the end of the project.

Around halfway through development the project began user testing, which was divided into two rounds. The first of these two rounds was focused around private testing with friends, family and fellow students. This took place at the start of February and identified key quality-of-life issues such as the player movement not being up to par and some inaccuracies with collision detection that led to the character model becoming stuck in the environment.

The final round of user testing took place during a public showcase at Comic Con Dublin 2025. This event was incredible for the project as it provided a massive opportunity to expose the game to a wide and unbiased audience of testers. This audience included thousands of attendees as well as multiple veteran game studios. The game received praise for the atmosphere it creates and the soundtrack. As well as this, testers really appreciated the visual style and attention to detail that the game displayed in its art and UI elements. This large scale testing also dug up some critical bugs that had not been found during the limited private user and functional testing. This included scaling issues upon resetting, ghost attacks and map clipping. The discovery of these bugs proved the importance that this testing had, as this level of user feedback in the later stages of development provided insights that likely would have gone unseen otherwise.

Overall, the testing phase was a success and proved to be extremely valuable to the development of the project.

7 Project Management

7.1 Introduction

This chapter goes into detail on the strategies and tools that were utilized in order to manage the development cycle of the project. The chapter will be split up into multiple sections, starting with the initial concept stage followed by the gathering of requirements, the overall design, the implementation phase, and finally, the testing methods used at each stage of the project's life. As well as the breakdown of each stage, the chapter will also cover the aforementioned tools used to manage these stages including Trello and GitHub. The overall aim of this chapter is to provide an overview of the project's management and highlight both the successes and pitfalls of the methods used.

7.2 Project Phases

In this section, describe each of the following project phases. Explain any issues which arose for each of the phases.

7.2.1 Proposal

The proposal phase involved deciding on the concept for the game and fleshing it out into an actual proposal that could be used as the grounds for a project. The stage was relatively easy compared to the rest due to the concept being based on a passion project. This made it simple to flesh it out and decide on the specific details that would give it enough weight to be used for the project.

Trello was extremely useful during this phase as it served as a space to store similar games for inspiration in an easy to read format. Using the notes feature on Trello made it a lot easier to pick these games apart for their key features. Below is an example of these notes, based around a small Indie game called Nuclear Throne.



Description

[Edit](#)

Nuclear Throne is Vlambeers latest action roguelike-like about mutants that spend their workdays trying to fight for the throne in a post-apocalyptic world. The radioactive waste in the world allows mutants to get ahead by mutating new limbs on the fly and the abundant availability of powerful weaponry makes the quest to become the ruler of the Wasteland one fraught with peril. All of this is really just an excuse for us to make a fun action game.

Nuclear Throne is a great example of how an Enemy's AI should behave. Roguelike games typically tend to have amazing AI systems due to it being the heart of the gameplay loop. Roguelike games have exploded in popularity the last few years with games such as Hades becoming overnight powerhouses in the gaming market. As a result of this there's been a lot of development in the way of Enemy AI systems and it would be well worth looking into this side of the industry to get resources and tutorials on how to develop a well rounded combat system for the player. The Roguelike genre would also be an amazing place to look if I want to implement any sort of powerups or items into the game as they're all rich with interesting stat-affecting collectables.

7.2.2 Requirements

The requirements phase was quite simple from the get go. The desire was to create a 2D game that invoked the same feelings of nostalgia as the early Gameboy titles from the 90's and early 2000's. Through the use of personas and research performed in the Research & Analytics module, the aforementioned list of both functional and non functional requirements were settled upon.

Functional Requirements:

1. The game should allow players to explore the temple in an engaging way, with some hidden areas and secrets to discover.
2. The combat system should feel responsive and fluid.
3. The graphics and soundtrack should be of a good quality to further the atmosphere of the game.
4. The controls for the game should be simple and easy to understand.

5. The difficulty curve throughout the game should remain fair, but challenging.
6. The level design should be designed to minimize frustration or confusion.
7. The game should offer a unique twist, and avoid feeling generic.
8. Items that the player discovers should have a clear purpose and be easy to understand.

Non-Functional Requirements:

1. The game should have an easy to understand interface and feature intuitive controls in order to minimize the learning curve players experience.
2. The game should maintain a consistent frame rate of at least 30 FPS in order to ensure the gameplay is smooth and responsive.
3. The code base itself should be well-organized and documented to allow bug fixes to be executed swiftly.
4. The game should be accessible for players at all skill levels.

A number of interviews were also conducted during this phase that fed into the final list of requirements. By interviewing people who would usually play these games it helps form a much clearer picture of player expectations and in turn leads to a better, more polished and user centric final product.

7.2.3 Design

The requirements for the project were a heavy influence on the design choices made in this phase. The assets, audio and general feel of the project had to be centered around delivering an experience akin to that of the games that inspired the concept. As part of the final non-functional requirement, 'The game should be accessible for players at all skill levels.', it was also important that the asset chosen be clearly defined and easy on the eyes. Regardless of what difficulty the game was developed around, if the assets were hard on the eyes it would naturally increase the difficulty and reduce player experience entirely which is something that directly goes against the project scope.

7.2.4 Implementation

The implementation phase did present a few obstacles. As covered earlier on in the report, there were multiple major bugs that were uncovered during development that took quite a while to fix. Outside of these bugs however, implementing everything together proved to be very fun. With the use of Unity's tilemap system and audio manager, the implementation of assets was very simple and streamlined. The blend tree systems in Unity's animation suite also proved to be very easy to use, tying the character and enemy models together and drastically improving the look of the game.

GitHub proved to be invaluable during this phase as well, providing a way to manage and track the game's code at each state and provide a streamlined level of version control for the project. The simplicity of GitHub's interface was also a massive help, minimizing stress during this phase.

7.2.5 Testing

The testing phase was quite broad, as it began very early on in the project's development cycle with functional testing. Functional testing was conducted throughout the entire development cycle as features and mechanics were tested as they were implemented to ensure that they were functional and worked well with each other. Aside from this, there were also two different rounds of user testing conducted in order to maximize the amount of testing performed on the game. Private user testing primarily took place in February and consisted of friends, family and fellow students testing the game and providing feedback and gameplay features and bugs. The final round of user testing was especially surreal as it took place during a showcase at Comic Con Dublin 2025. Exposing the game to thousands of testers in this public setting was instrumental in uncovering a lot of key bugs as well as confirming what areas the game was excelling in.

7.3 SCRUM Methodology

The use of sprints during the implementation of the project provided a more focused approach to developing all of the different elements required by the game. Although the time centric nature of the sprints did force a sense of urgency, the rigid two week duration did also present some obstacles. These obstacles were primarily with the more complex tasks that simply couldn't be completed in a two week period. Despite this, the structure of the sprints was extremely effective in breaking the large number of tasks into various, manageable categories and was instrumental in maintaining momentum throughout development.

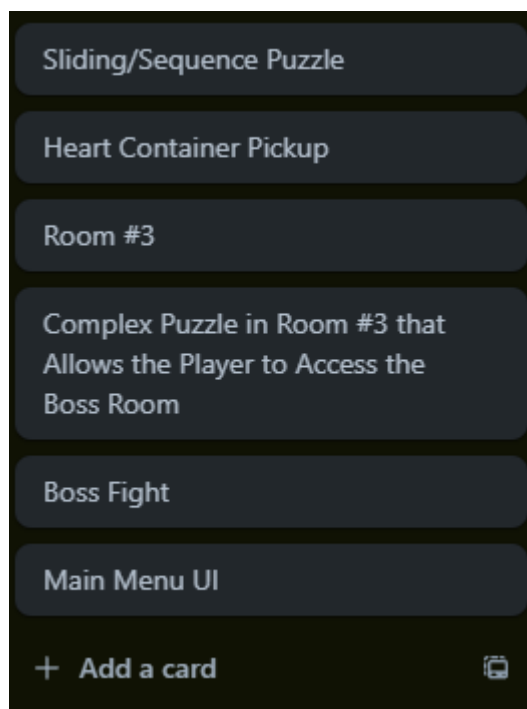
SCRUM methodology as a whole ended up being largely beneficial as well. The product backlog created a roadmap of sorts, and the aforementioned sprint structure ensured more focused periods of development. Frequent self-reflection aided in the tracking of project progress, while the sprint reviews ensured that the development principles and overall project improved from sprint to sprint. The nature of SCRUM overall provided a sense of always having a milestone ahead to reach for which was incredibly motivating throughout the development process. Ultimately, the structure and adaptability that the framework of SCRUM provided proved to be crucial in guiding the development of the game and put a major focus on organization and continuous improvement throughout.

7.4 Project Management Tools

7.4.1 Trello

Trello by nature is a web-based application typically used to manage project progression using a flashcard format. These 'flashcards' are typically split up into lists that represent different categories of research or tasks of a certain status such as being in progress or completed. Each of these cards can contain various bits of information such as notes on the task or a due date.

During the project Trello served as the primary tool for organizing tasks throughout each of the sprints. At the beginning of each sprint the Trello board was loaded with various tasks that corresponded to that sprint. An example of this can be seen below.
















Having this visual representation of project progress provided a clear and easy to read overview that made it much easier to identify weaker areas of the project during development. It was also particularly useful when tackling larger tasks, providing the ability to break them down into smaller, bite-sized steps.

7.4.2 GitHub

GitHub by nature is a web-based platform based around the version control system, Git. Git is a version control system used to track changes made to files over time, allowing developers to easily revert to previous versions and sort their tasks into different 'branches'. GitHub is an extension for Git that provides the developer with a remote repository for storing the code and various files for the

project on the cloud along with an array of tools for both managing and collaborating on the project if needed.

For this project, GitHub was used as the primary repository for storing the project files. The ability to consistently commit code to the repository as features were implemented ensured that a log was kept of the project's history. This log also made it simple to rollback to a previous version if needed when bugs arose, making diagnosing and fixing these bugs much easier in the long run.

Comic Con  Poggs5401 • 8 days ago
Enemy Rooms Cont.  Poggs5401 • last month
Bug Fixes #2  Poggs5401 • last month
Expanded Audio Design  Poggs5401 • last month
Enemy Rooms, Background Music, Hearts, Room Boundaries  Poggs5401 • last month
Locked Door Functionality  Poggs5401 • last month
Cinemachine & Enemy Boundaries  Poggs5401 • 2 months ago
Death Animations  Poggs5401 • 2 months ago
Patrol Functionality  Poggs5401 • 2 months ago
Bug Fixes #1  Poggs5401 • 2 months ago
Treasure Chest Functionality  Poggs5401 • 2 months ago
New Font Added  Poggs5401 • 2 months ago
Dialog System, Treasure Chest & Inventory System Foundations  Poggs5401 • 2 months ago

7.5 Reflection

7.5.1 Your views on the project

As a whole I believe the project went quite well. There were some issues in terms of time constraints that resulted in there being less features in the final product than what was originally pitched. There were many things that had to be scaled back including the implementation of puzzles and more in depth combat mechanics as well as a boss enemy variant. This was due to there being a lot of complications during the implementation of the enemy rooms in the build up to the Comic Con showcase. It was due to these complications that the decision was made to pivot to a more straightforward combat demo that then ended up being the final product. I feel that this was less than ideal given the original scope. The project as a whole faced a lot of scaling back due to time. The original concept would have seen the game be an open world RPG style experience with multiple hours of gameplay. Upon the timeline being released this was quickly scaled back to include only a

temple and instead be made up of rooms of varying puzzles and enemy types with the game ending with a challenging boss fight. This was then scaled back even further following the complications faced during the implementation of the enemy room. Although this was very unfortunate I feel as if the project still went well.

7.5.2 Completing a large software development project

I learned a lot about time management from this project. Completing a large software project as a solo developer in a relatively short amount of time was a massive learning experience that forced me to work outside of my comfort zone, manage a large multitude of tasks simultaneously and learn how to tie every aspect of the project together into one finished product.

7.5.3 Working with a supervisor

Working with a dedicated project supervisor was a first for me. Working with Naoise helped massively throughout the project as his support helped me stay on the right track. Weekly meetings ensured that I never strayed too far from the original scope and that I always had plenty of feedback to work with. Another massive perk I found to having a supervisor was the experience that they bring to the table. Some tasks can't simply be looked up online and learned in an afternoon from scratch, so having a supervisor with years worth of experience in the game development industry proved to be invaluable in completing some tasks as I was able to get quick answers to my questions and concerns while implementing various mechanics.

7.5.4 Technical skills

I feel as if I learned a lot from a technical standpoint throughout the development of this project. It was my first time working on a solo project to such a scale and my first time working with many 2D features in Unity such as the lighting systems. I learned a lot more about how to control game objects in the hierarchy, using them to control enemies spawning in. As well as this, I was able to learn a lot more about the use of audio managers for sound effects and blend trees for four-way directional animations. All of these things were extremely important in achieving the eventual final product and I feel as if I have become a much better and more competent developer as a result.

7.5.5 Further competencies and skills

In terms of skills that will help me going forward as a developer in the workplace I feel as if I've learned a lot. Learning how to handle larger scale software projects and using tools such as Trello and GitHub to manage it has improved my skills as a developer as a whole, and I feel as if it has prepared me even more to break into the workplace. On top of this, I've now

learned a lot more about how important it is to organize research and tasks into an easy to read format, as if you fail to keep these things organized it causes the entire project to suffer.

7.6 Conclusion

Write a couple of paragraphs summing up the chapter. Explain what area your project is about. Describe what the chapter has discussed.

The chapter provided an overview on the methods and tools used throughout the development process. Included in this overview was the progression through the various key phases from the initial proposal and gathering of requirements to the design phase where the focus was centered around aligning the game's aesthetic with its requirements and goals. The implementation phase followed and with it came challenges in the form of complex bugs. Despite these setbacks however the use of Unity's library of features and GitHub's version control capabilities made the development process a lot less stressful. The testing phase was conducted throughout all of these previous phases with extensive testing performed with the addition of every new feature. The testing phase culminated with a public showcase of the game at Comic Con Dublin 2025. This public showcase was invaluable in exposing large bugs that would've been extremely difficult to discover during private testing.

This chapter also covered the use of SCRUM methodology throughout the project and the various benefits it provided during development. By breaking the project up into several two week long sprints it made both managing larger tasks and maintaining momentum a lot easier. While the sprint system did pose a few challenges in terms of time management, it still proved to be beneficial overall in streamlining the workflow of the project. The use of Trello also helped a lot in managing larger tasks and visualizing ideas for the project by providing a space where all research could be pinned and saved for inspiration. In short, this chapter has gone into detail on the methods used to manage the project, highlighting both the successes and shortcomings of each method.

8 Conclusion

In conclusion, this project initially stemmed from a passion project built upon the desire to explore nostalgia as an emotion and eventually evolved into a top-down 2D game. The overall aim was to provide the player with an engaging experience that evoked this feeling of nostalgia through the use of simple gameplay mechanics and a charming visual style.

The project utilized a range of different technologies within Unity. The tilemap editor was used to create the world that the player finds themselves in and the audio manager was used to give the project more charm and weight. As well as this, Unity's animation suite was key in streamlining the implementation of assets, especially through the use of blend trees. GitHub was then utilized as a remote way of storing the project files and having a level of version control in order to simplify both tracking down and fixing the various bugs that surfaced throughout development.

The project had a very structured approach, starting with research into both the target audience and overall genre that the game was aimed at. This built into the design phase as the research conducted assisted in capturing that desired nostalgic aesthetic and ensured accessibility. Although challenging, the implementation phase produced a functional game that showcased the desired core mechanics. Testing was also conducted throughout every stage of the project's life and was split up into three phases: functional testing, private user testing, and public user testing. All three of these stages played a massive part in both identifying and rectifying issues that may have remained unseen otherwise.

Overall, the project delivered a game that was both playable and that met the aims and originally set out. Project management, through the use of Trello and SCRUM methodology, was crucial in organizing the workload, tracking tasks, and maintaining momentum throughout the development cycle. Key takeaways include the importance of in depth testing, the various challenges that come with solo development, and the value of structure in the methods used to manage the project.

In terms of further development, adding in additional enemy types and expanding the world beyond the temple would make for an even more enjoyable experience. As well as this, the addition of both puzzles and perhaps a dialog/quest system would really expand the possibilities of the game.

References

The Department of Technology and Psychology in IADT uses APA referencing style.

Use alphabetical order for your references.

This site gives details about how to cite websites using APA:

<https://www.wikihow.com/Cite-a-Website-in-APA>

The following is a useful site for creating citations for APA for websites.

<http://www.citationmachine.net/apa/cite-a-website>

You can also use the Referencing tab within Microsoft Word to enter reference information manually. Word then creates an APA style reference.

Munjal, S. (2022, April 12). Unity vs Godot: Major differences you must know. Java

Assignment Help. <https://www.javaassignmenthelp.com/blog/unity-vs-godot/>

Allen, D. (2023). 16-Bit Games We Still Enjoy Today. [online] TheGamer. Available at:

<https://www.thegamer.com/16-bit-era-games-snes-genesis-still-hold-up/> [Accessed 22 Nov. 2024].

Bowden, T. (2024). Pixels to Reality: The Evolution of Video Game Art. [online] RMCAD.

Available at: <https://www.rmcad.edu/blog/the-evolution-of-video-game-art/> [Accessed 27 Nov. 2024].

English, K. and D.I.D Electrical (2023). An Exhaustive History of Eight Generations of Video Game Consoles: 1967 to 2018. [online] DID Electrical. Available at:

<https://www.did.ie/blogs/gadgets/an-exhaustive-history-of-eight-generations-of-video-game-consoles-1967-to-2018> [Accessed 27 Nov. 2024].

Gamestate (2023). Evolution of Video Game Graphics. [online] Gamestate. Available at:

<https://gamestate.com/blogs/news/the-evolution-of-video-game-graphics-from-8-bit-to-hd-and-vr?srsId=AfmBOoqC1fsM5bcvtVUrQFOjz9F-CXsRg7adwWKP-0MCRH2jxmBtCOPa>

[Accessed 22 Nov. 2024].

Gordon, W. (2021). What's the Best Way to Play Retro Games? [online] Wired. Available at: <https://www.wired.com/story/best-way-to-play-retro-games-virtual-console-emulator/> [Accessed 27 Nov. 2024].

Hernández, M. (2024). History of eSports: How Did Video Game Competitions begin? [online] Telefónica. Available at: <https://www.telefonica.com/en/communication-room/blog/history-of-esports-how-did-video-game-competitions-begin/> [Accessed 6 Dec. 2024].

Hurley, R. (2024). Retro-futurism: How the past influences visual design in video games. [online] Available at: https://illustro-iadt.figshare.com/articles/thesis/Retro-futurism_How_the_past_influences_visual_design_in_video_games/25407934?file=45037816 [Accessed 22 Nov. 2024].

Irfandi (2023). The Nostalgia Effect: How Retro Games Influence Modern Gaming. [online] Medium. Available at: https://medium.com/@dq_irfandi/the-nostalgia-effect-how-retro-games-influence-modern-gaming-8925be77694e [Accessed 22 Nov. 2024].

Jabr, F. (2024). John A. Long - Publications List. Publicationslist.org, 14(6). Jack, P. (2023). Why Emulators Are Important to the Future of Games Preservation. [online] CBR. Available at: <https://www.cbr.com/retro-games-need-emulation-for-games-preservation/> [Accessed 27 Nov. 2024].

Jouanna Bondakji (2023). 8-Bit Games We Still Enjoy Today. [online] TheGamer. Available at: <https://www.thegamer.com/8-bit-era-games-nes-master-system-still-hold-up/> [Accessed 22 Nov. 2024].

Kent, S.L. (2001). The ultimate history of video games. New York: Random House International ; London. Mahardy, M. (2014). 9 Game Franchises That Returned from the Dead - IGN. [online] IGN. Available at: <https://www.ign.com/articles/2014/05/28/9-game-franchises-that-returned-from-the-dead> [Accessed 27 Nov. 2024].

Mason, G. (2016). Nine Ways the the 8-bit Era Made Gaming What It Is Today. Eurogamer.net. [online] 15 May. Available at: <https://www.eurogamer.net/nine-ways-the-the-8-bit-era-made-gaming-what-it-is-today> [Accessed 27 Nov. 2024].

Matthew, P. (2024). The Rise of Gaming Conventions and Their Impact on the Industry. [online] Medium. Available at: <https://medium.com/@pierce.matthew/the-rise-of-gaming-conventions-and-their-impact-on-the-industry-dc4a06ee99c1> [Accessed 6 Dec. 2024].

Mubarak (2024). What Do 8-Bit and 16-Bit Actually Mean? A Simple Guide to Classic Gaming Terms. [online] Retro News. Available at: <https://www.retronews.com/what-do-8-bit-and-16-bit-actually-mean-a-simple-guide-to-classic-gaming-terms/> [Accessed 22 Nov. 2024].

Parker, B. (2024). The Rise of Speedrunning: Gaming's Fastest Phenomenon. [online] Medium. Available at: <https://medium.com/@bparks918/the-rise-of-speedrunning-gamings-fastest-phenomenon-283da6436fae> [Accessed 6 Dec. 2024].

Potila, T. (2023). Soundtrap | What Is Chiptune And How To Make Chiptune Beats. [online] Soundtrap. Available at: <https://www.soundtrap.com/content/blog/how-to-make-chiptune-beats>. Rajpurohit, P. (2024).

Evolution of Video Game Graphics - Then Vs Now. [online] 300Mind Blog. Available at: <https://300mind.studio/blog/the-evolution-of-video-game-graphics/> [Accessed 27 Nov. 2024].

Scott Daniel (2024). The Rise of Retro Gaming: Nostalgia in the Digital Age. [online] Medium. Available at: https://medium.com/@scottdaniel_20892/the-rise-of-retro-gaming-nostalgia-in-the-digital-age-ff5c90ddde85 [Accessed 27 Nov. 2024].

Steinberg, N. (2018). The 15 Greatest Video Game Reboots That Breathed New Life into a Franchise. [online] HowStuffWorks. Available at: <https://electronics.howstuffworks.com/greatest-video-game-reboots-that-breathed-new-life-into-a-franchise.htm> [Accessed 27 Nov. 2024].

Thompson, M. (2024). From 8-bit to Hyper-Realism: The Evolution of Video Game Graphics | Gaming. [online] Landofgeek.com. Available at: <https://www.landofgeek.com/posts/evolution-video-game-graphics-8-bit-hyper-realism> [Accessed 27 Nov. 2024].

Vento, J. (2024). How Old-School Games Are Inspiring a New Generation of Indie Developers. [online] Medium. Available at: <https://medium.com/@DarkRa/how-old-school-games-are-inspiring-a-new-generation-of-indie-developers-5dedb17ca3e1> [Accessed 27 Nov. 2024].

Wells, S. (2024). The Evolution and Impact of Gaming Consoles: A Comprehensive History. [online] The Lifestyle Daily. Available at: <https://www.lifestyledaily.co.uk/article/2024/08/14/evolution-and-impact-gaming-consoles-comprehensive-history> [Accessed 27 Nov. 2024].

Wulf, T., Bowman, N.D., Rieger, D., Velez, J.A. and Breuer, J. (2018). Running Head: Video Game Nostalgia and Retro Gaming. Media and Communication, [online] 6(2), pp.60–68. Available at: <https://www.cogitatiopress.com/mediaandcommunication/article/view/1317/781> [Accessed 27 Nov. 2024].

Yoon, S. (2024). Gaming Culture: A New Language for the Digital Age. [online] Forbes. Available at: <https://www.forbes.com/sites/forbesbooksauthors/2024/05/14/gaming-culture-a-new-language-for-the-digital-age/> [Accessed 6 Dec. 2024].