UNIVERSITY POLITEHNICA OF BUCHAREST
FACULTY OF AUTOMATIC CONTROL AND COMPUTERS
COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

DIPLOMA THESIS

Godchosen
A 2D mythological adventure game

Marian-Paul Lupuleasa

Thesis advisors:

Prof. dr. ing. Costin Anton Boiangiu
As. drd. ing. Giorgiana Violeta Vlăsceanu
Drd. ing. Nicolae Tarbă

BUCHAREST
2023

CONTENTS

SINOPSIS

Această teză prezintă motivația și detaliile implementării unui joc video de aventură 2D, dezvoltat în motorul de joc Unity, precum și rezultatele obținute, dar și posibile îmbunătățiri ulterioare care pot ridica jocul la un nivel superior.

Motivația principală a acestei teze este crearea unui joc video care îmbină aspecte și funcționalități ale altor jocuri, atât clasice, cât și de ultimă generație, pentru a crea o experiență proaspătă și unică pentru jucător, precum și un punct de plecare pentru o carieră de dezvoltare de jocuri video.

Soluția propusă a fost dezvoltată folosind motorul de joc Unity și limbajul de programare C#. Rezultatul este o experiență captivantă, în stil pixel art 2D, cu o poveste plasată în mitologia greacă, susținută de un gameplay plin de acțiune care combină mecanismele multiple atât ale clasicilor, cât și ale jocurilor recente foarte lăudate, pentru a reînvia nostalgia jocurilor din copilărie.

Cuvinte cheie: joc video, 2D, Unity, C#, pixel art, mitologie greacă

ABSTRACT

This thesis presents the motivation and implementation details of a 2D adventure video game, developed in the Unity game engine, as well as the results obtained, and also possible further improvements that can elevate the game to a higher level.

The main motivation of this thesis is creating a video game that blends aspects and functionalities of other games, both cult classics and state of art, to create a fresh and unique experience for the player, as well as a starting point for a video game development career.

The proposed solution was developed using the Unity game engine and the C# programming language. The result is an immersive, 2D pixel art style, story driven experience set in the Greek mythology, backed up by an action packed gameplay that combines multiple mechanics of both cult classics and highly praised newcomers, to revive the nostalgia of childhood's games.

Keywords: video game, 2D, Unity, C#, pixel art, Greek mythology

1 INTRODUCTION

This chapter introduces and defines the solution and its purpose while also explaining some of the terms that will be used to describe the game and its implementation throughout the whole paper.

1.1 Solution Description

The proposed solution is a two-dimensional (2D) video game developed in the Unity game engine, with a side-scrolling perspective, that combines elements from both the platformer and the tower defense sub-genres in order to deliver an epic journey into the mythological lands of Ancient Greece.

1.2 Motivation

The video game industry has grown tremendously over the years and is currently more popular than ever. It includes hundreds of types of games and thousands of products that represent a form of entertainment for a large part of the planet's population. In this ocean of products, there is a game for every person, regardless of age or background [0].

Personally, as a soon to be graduate that searches for their purpose in this world, the video game industry has been an important part of my life since an early age and is becoming more and more significant as I came to the realisation that I want to do more than just play video games, I want to be a part of creating them. As a result, I have taken this opportunity to develop my first video game in a personal style, by taking inspiration from some of my favourite games of all time and combining different aspects and elements to create an original product that reflects my passion and commitment to this craft.

1.3 Related Terms

Unity [1] - a game engine developed by Unity Technologies and released in 2005. It is mainly used to create 2D and 3D video games written in the C# or JavaScript programming languages.

2D video game - A type of video game that uses parallel projection to represent 3D bodies in a two-dimensional space from a certain perspective, based on the game genre and what it is trying to convey. Thanks to their graphical simplicity, 2D games require fewer assets and can focus on creating large and complex worlds and mechanics, making it easier to design models and animations.

Perspective - The way to present the game's world using cameras situated at different angles. A 2D video game has one of the following 3 perspectives: top-down (view from above), side-scrolling (view from the side) and 2.5D (diagonal view, between 2D and 3D).

Side-scrolling - A type of 2D video game perspective that positions the camera on the side of the game space, giving the player a horizontal view of the scene. This point of view can be used to create horizontal layers that confer the game a sense of height and is mainly used for 2D platformers and shooter games.

Platformer - A sub-genre of the action game genre in which the main objective is to move the playable character from one point to another, while navigating platforms, fighting or avoiding enemies and collecting different objects that help the player in their adventure. Platformer games are usually developed in a 2D side-scrolling perspective, making the movement and the combat easier to observe and interact with.

Examples of platformer games: **Super Mario Bros [2]**, **Spelunker [3]**

Tower defense - A sub-genre of the strategy game genre with the main goal of having the player defend an area from waves of enemies, by utilizing strategic techniques to stop their advance and defeat them before reaching the defended territory.

Examples of tower defense games: **Bloons Tower Defense [4]**, **Plants vs. Zombies [5]**

2 STATE OF THE ART

This chapter describes the video games that represent the main inspiration for the development of the solution, be it theme, visuals, mechanics or gameplay.

To create a cohesive story and a unique experience, the game draws its inspiration from products of numerous genres, released during the entire timeline of video games' existence, ranging from 80's classics, all the way to modern day innovators.

2.1 Theme and Visuals - Ancient Greece

2.1.1 Hades

Hades [6] is a 2020 roguelike game developed by Supergiant Games. The game is set in the Underworld, ancient Greece's realm of the dead, and follows Zagreus, the eponymous main antagonist's son, who is trying to escape hell and reach the mortal world, in order to meet his mother. Being a roguelike game, the main focus of this game is the constant attempt of the protagonist to escape the Underworld, each failure resulting into starting over from the beginning. However, after each attempt, the player is able to buy upgrades to make them stronger until they are able to defeat their father, Hades, and reach the end.

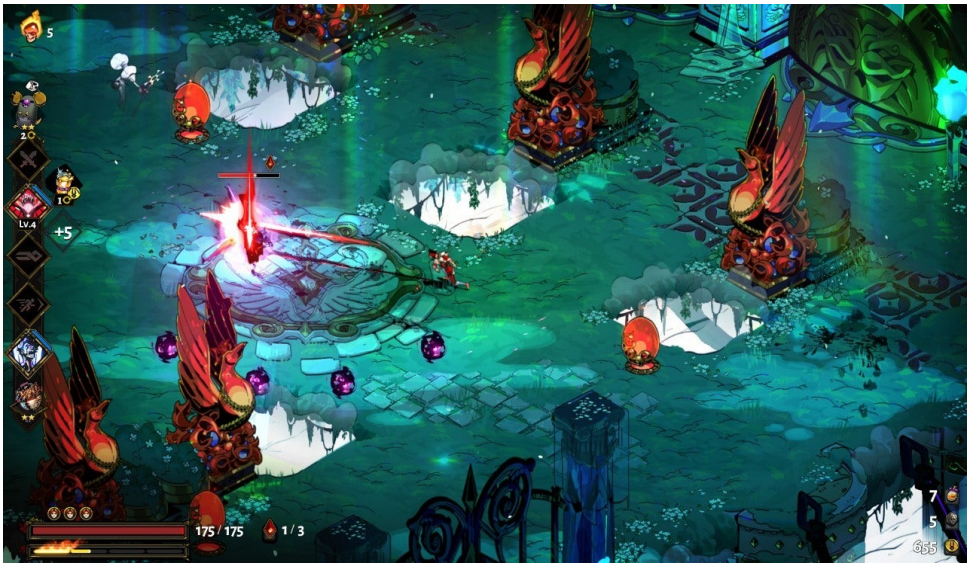

Figure 1: Hades Gameplay Scene

2.1.2 Immortal Fenyx Rising

Immortal Fenyx Rising [7] is a 2020 3D third-person action adventure game developed by Ubisoft Quebec. The game is set in ancient Greece and follows the eponymous protagonist, Fenyx, as he tries to save the world from destruction by battling mythological creatures, such as the gorgons, the Minotaur, the Lernaean Hydra and, finally, the titan Typhon, while being aided by a number of Greek gods, who lend him their powers.

Figure 2: Immortals Fenyx Rising Visual Scene

2.1.3 Inspiration

Both games are set in ancient Greece and are an important influence for the game's main setting, while providing inspiration for the main character, the enemies and the environment. In addition, the game's defining mechanic, the weapon-swapping mechanic, which allows the player to switch between different weapons to interact with the environment, has its roots in Hades' weapon system, which allows the player to start an attempt by picking a weapon of their choice, each with a different playstyle.

Therefore, the solution alternates between three weapons, a sword, a bow and a shield, in order to offer a deeper experience while playing. Thematically, each weapon is infused with the power of a god in order to convey the feeling that the player is channeling their power and is getting aided in their adventure by higher forces.

2.2 Perspective - Side-scrolling

2.2.1 Cuphead

Cuphead [8] is a 2017 2D run and gun game, developed and published by Studio MDHR, in which the player controls the eponymous character or his brother, Mugman, and is tasked with defeating several unique bosses in order to collect their souls and free himself from the devil's grasp. Playable both in solo and multiplayer mode, the game combines platformer and shooting elements to create unique encounters, enforced by colorful retro-style graphics and music.

Figure 3: Cuphead Gameplay Scene

Figure 4: Cuphead Gameplay Scene

2.2.2 Hollow Knight

Hollow Knight [9] is a 2017 side-scrolling game, developed and published by Team Cherry, in which the player controls an insectoid character, named the Knight, who tries to put an end to a plague that has affected their kingdom by exploring the realm and defeating infected enemies and the source of the plague itself. Packed with unique boss encounters, deep exploration and complex combat, Hollow knight is one of the most praised side-scrolling games of all time, delivering a well rounded platformer from both a narrative and a mechanical aspect.

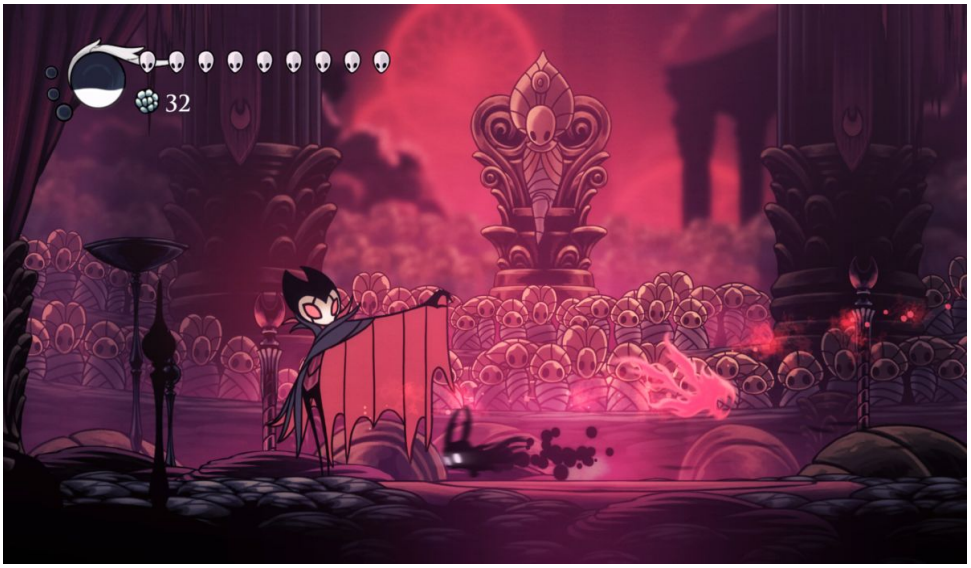

Figure 5: Hollow Knight Gameplay Scene

Figure 6: Hollow Knight Gameplay Scene

2.3 Gameplay and Mechanics - Platformer sub-genre

2.3.1 Spelunker

Spelunker [3] is a 1983 2D platformer game, developed by Timothy G. Martin of Micro-Graphic Image, in which the player controls an adventurer who explores a big cave divided into 6 levels, with the aim to discover a long-forgotten treasure, while avoiding dangerous enemies and traps that intend to stop the player from reaching the end of the game. Being one of the oldest platformers in the video game industry, it has become a genre defining cult classic and the main inspiration for every future platformer type game.

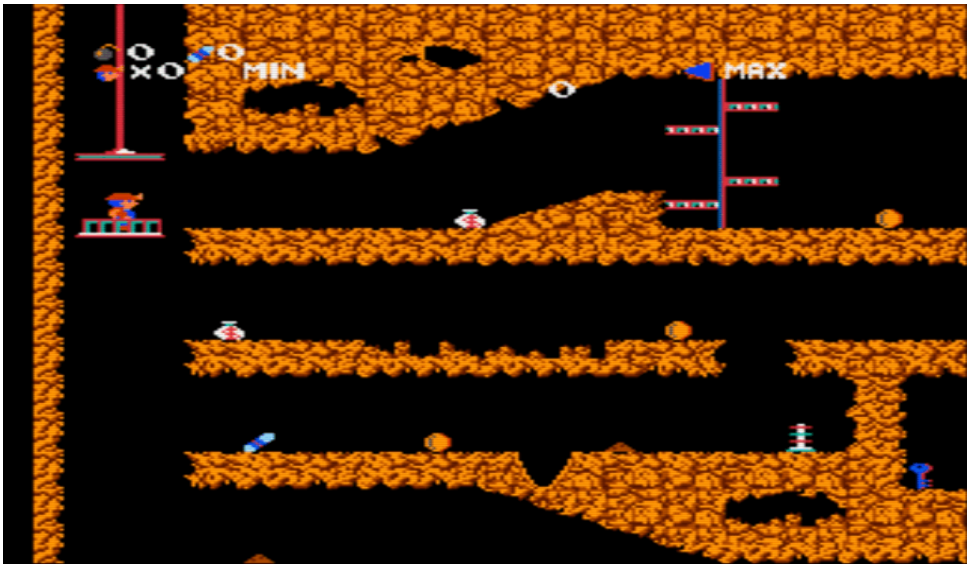

Figure 7: Spelunker Gameplay Scene

Figure 8: Spelunker Gameplay Scene

2.3.2 Spelunky

Spelunky [10] is a 2008 2D platformer game, developed by Mossmouth, LLC, that draws heavily from Spelunker but improves its gameplay on every level, both visually and mechanically by providing more environments, more enemies, more traps and more collectibles, but also countless new interactions between these mechanics, thus creating a fresh and unique experience.

Figure 9: Spelunky Gameplay Scene

2.3.3 Inspiration

Both games' main mechanics include: dangerous enemies that must be defeated, deadly traps that must be avoided and collectable items that can aid the player in their journey. The proposed game incorporates these mechanics in a similar way, but adds depth to the gameplay by providing unique ways to interact with them using one of the weapons with specific uses, while also setting a different theme and story, surrounding Greek mythology and the countless gods and mythological creatures that are a part of it.

2.4 Gameplay and Mechanics - Tower defense sub-genre

2.4.1 Plants vs Zombies

Plants vs Zombies [5] is a 2009 tower defense game, developed by PopCap Games, in which the player is tasked with defending themselves and their home from a horde of flesh eating zombies by using unique species of plants, each with their unique ability, to kill the invading army of enemies.

Figure 10: Plants vs. Zombies Gameplay Scene

Figure 11: Plants vs. Zombies Gameplay Scene

2.4.2 Inspiration

The mechanic that inspired the solution is the presence of lanes, each representing a path the zombies can follow towards the house that the player has to defend by placing plants along its entire length. This system allows for a more controlled gameflow, as opposed to a big area where the enemies come in a chaotic way, overwhelming the player.

Therefore, the second part of the game represents a tower defense level in which the player will move between five lanes in order to protect the environment's structural integrity from the wrath of the enemy forces. This mechanic also inspired the final part of the game, in which the player has to escape the temple by navigating between lanes to avoid getting hit by falling debris.

2.5 Related Work Conclusions

Each of the games presented above stands out from others of their genre, be it by having a unique setting, a compelling story, an original weaving of mechanics or simply by bringing genre defining elements to the table, becoming a primary source of inspiration for future game releases.

The solution combines different game genres, thematic styles and mechanics, in order to craft an original experience, with a fun and easy to follow story, that spawns across unique levels, while also offering a rewarding playtime and a sense of achievement after completing the adventure.

3 PROPOSED SOLUTION

This chapter describes the solution as a whole product, from a **narrative**, **architectural** and **functional** standpoint.

3.1 Description

The solution is a 2D video-game that combines platformer and tower defense elements to deliver an easy to follow story-driven experience set in the mythological world of Ancient Greece. The main focus of the game is retrieving an ancient artifact while navigating a perilous temple, filled with dangerous enemies and deadly traps, with the help of god-infused weapons, each with its different uses in combat and exploration.

3.2 Narrative

"The player is tasked by the gods with recovering a powerful artifact from the depths of an ancient (and dangerous) temple. After wandering through the temple and getting a taste of its deadly traps and menacing serpent-like guardians, the player finds a pair of beautiful wings that gives their wearer the ability to fly. Suddenly, waves of enemies swarm the temple in an attempt to stop the player from escaping with the relic they were guarding, led by none other than Medusa, the Gorgon. After a fierce battle during which the temple starts collapsing, the player proceeds to escape by flying upwards through a crack that leads to the surface, leaving the ruins of the temple behind."

3.3 Architecture

The game is composed of multiple scenes that are represented by menus and levels. The general game flow and the transitions between these scenes is done using screens and buttons. The existing scenes and the connections between them are presented in the following figure:

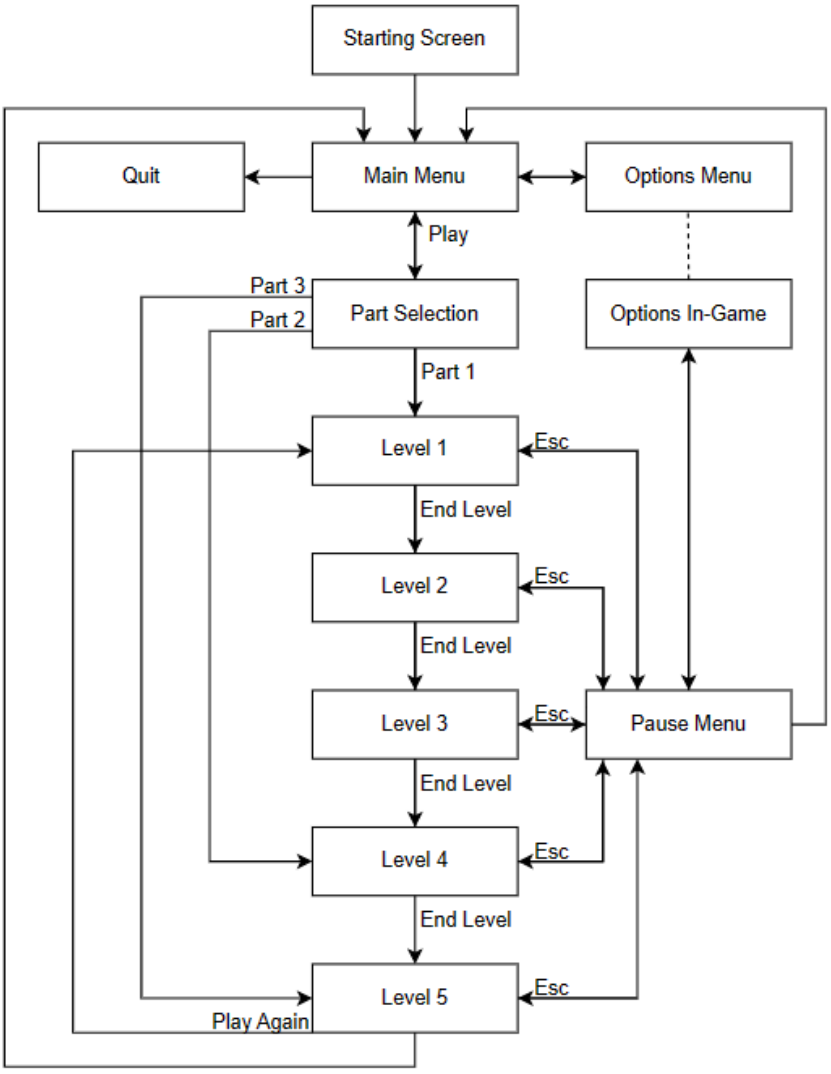

Figure 12: Game Flow Diagram

Although the **Pause Menu** is not a scene, but a screen, it is the main way the transition between scenes is made while inside the playable part of the game.

3.3.1 Menus and Screens

The game has several menus and screens that allow the players to navigate through the game and its levels, while having access to sound options, as well as pausing or restarting the current game session.

Starting Screen - The default scene of the game, presenting the name and the aesthetic of the game.

Figure 13: Starting Screen Scene

Main Menu - The main scene of the menu, allowing the player to play or quit the game, as well as access to the options menu.

Figure 14: Main Menu Scene

Options - Split in two parts, one scene accessible from the main menu, outside the level scenes, and one screen accessible from the pause menu, inside the level scenes. Both menus are synchronized and offer the same functionality, but in different segments of the game.

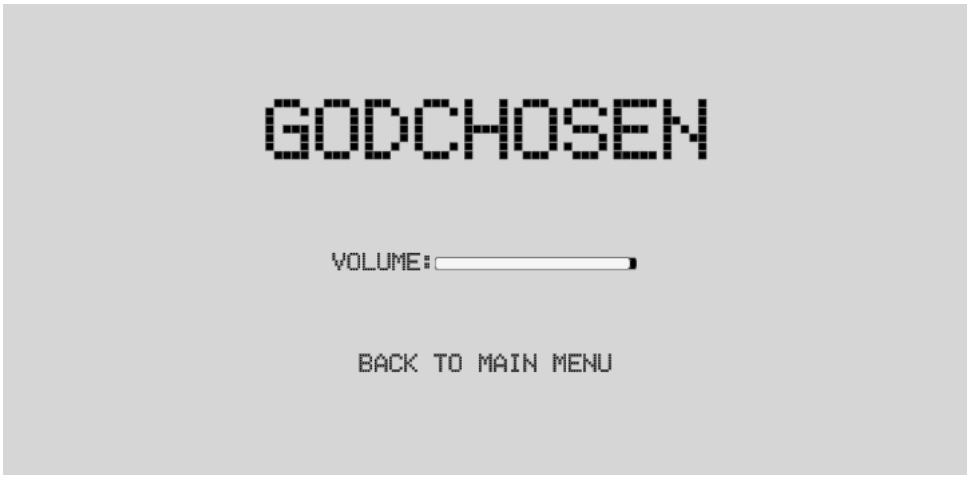

Figure 15: Options Scene

Figure 16: Options Screen

Part Selection - The scene that allows the player to choose the part of the game they wish to play.

Figure 17: Part Selection Scene

Pause Menu - The screen that activates when the player pauses the game, allowing them to restart the current part and access the options screen or main menu.

Figure 18: Pause Menu Screen

Win Screen - The screen that activates when the player finishes the final part of the game, allowing them to restart the game or navigate to the main menu.

Figure 19: Win Screen

Lose Screen - The screen that activates when the player loses all their lives, allowing them to restart the game of navigate to the main menu.

Figure 20: Lose Screen

3.3.2 Game Parts

The game is split into **three** parts, each with their own story segments, mechanics and visuals.

Part I

The first part consists of **three** platforming levels and represents the biggest part of the game. During this part, the player will learn the mechanics of the game and will battle enemies, avoid traps and reach the sought after artifact.

Level 1 - The first level of the first part has a simple platformer layout and introduces the movement and the attack mechanics to the player.

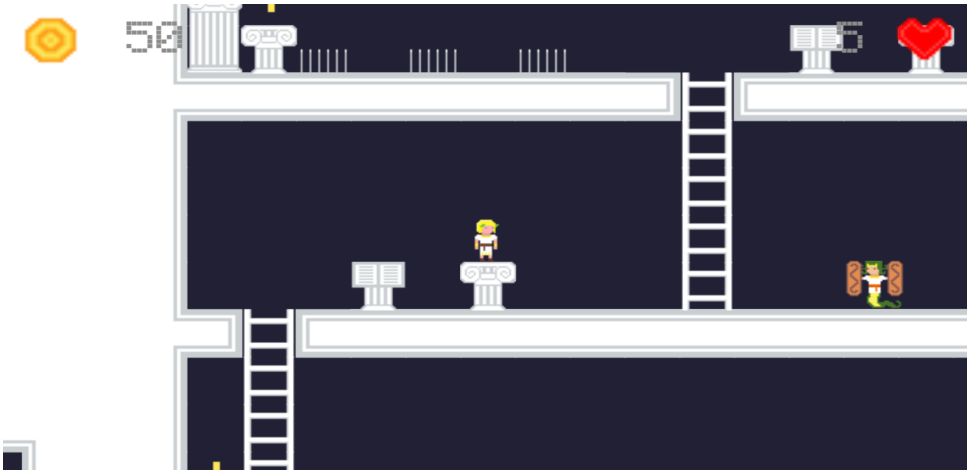

Figure 21: Level 1

Level 2 - The second level of the first part is bigger and more complex than the first one, and introduces more complex mechanics and interactions with the different enemy types.

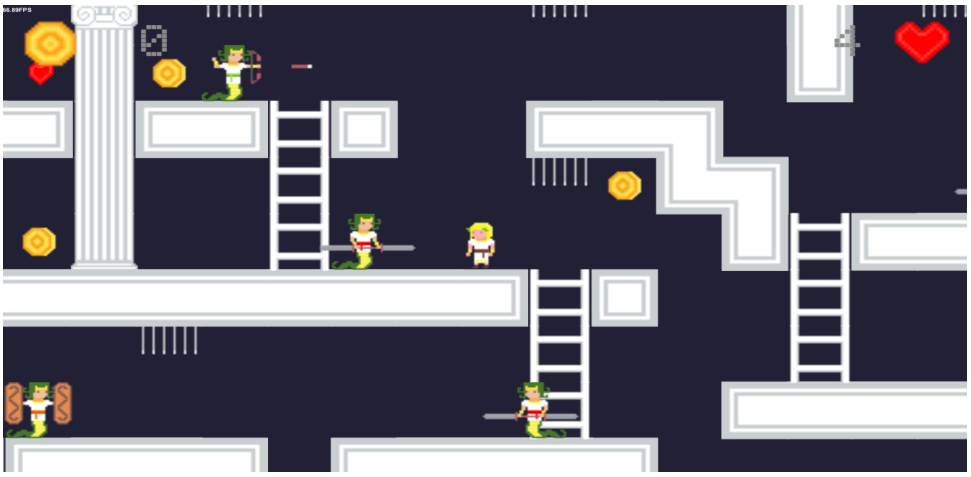

Figure 22: Level 2

Level 3 - The third level of the first part is the biggest one and combines all the mechanics available during the platforming part to force the player to use their creativity and intelligence to navigate the plethora of traps and enemies in order to recover the artifact.

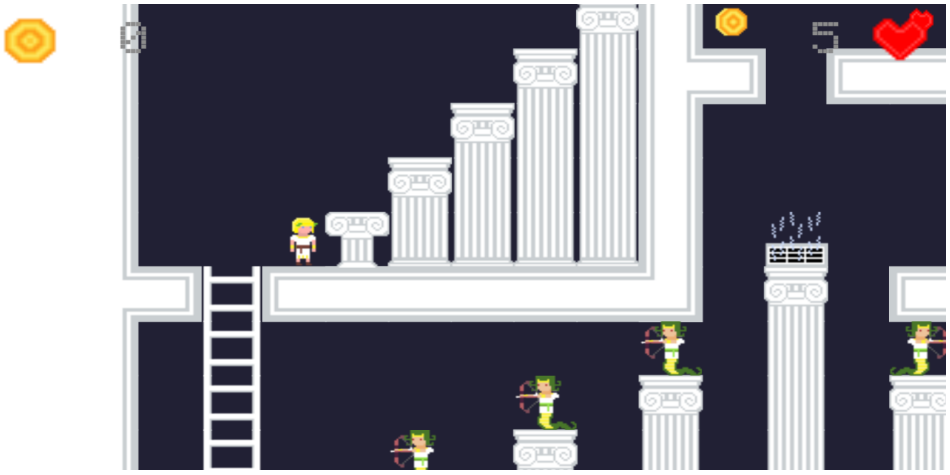

Figure 23: Level 3

Part II

The second part consists of a tower defense level in which the player has to defeat waves of enemies that threaten the stability of the temple by navigating between five horizontal lanes, each with its own enemies, traps and obstacles, culminating with a mighty boss fight that will consequently trigger the temple's collapse.

Level 4 - The only level of the second part is a platformer level that requires the player to defend the temple from waves of enemies, and finally from Medusa herself, by navigating between lanes while blocking damaging projectiles and eliminating all the enemies that fly towards the player.

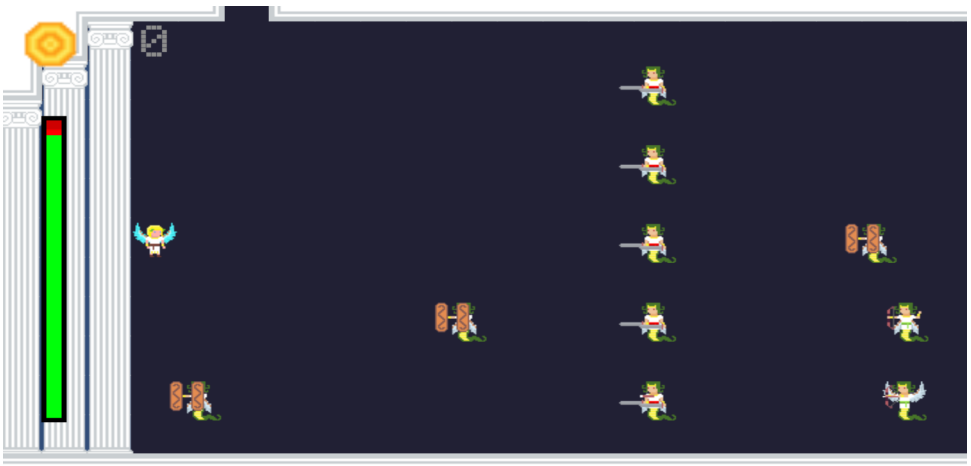

Figure 24: Level 4 - Waves Phase

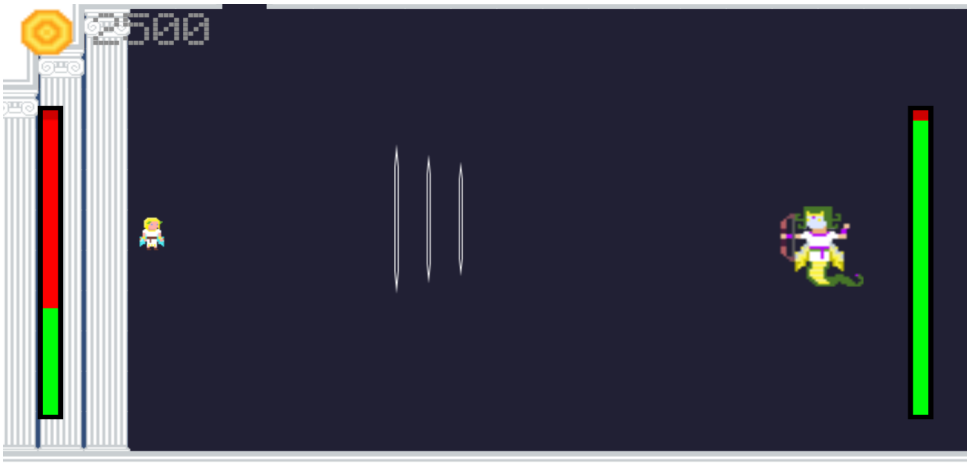

Figure 25: Level 4 - Boss Phase

Part III

The third and final part consists of a non-combat timed escape from the collapsing temple, combining elements from the first two parts, requiring the player to dodge projectiles and falling terrain while flying between vertical lanes in an attempt to reach the surface with the help of the recovered relic.

Level 5 - The only level of the third part follows the player in their attempt to escape the collapsing temple, by flying towards the surface, while avoiding all the obstacles they encounter.

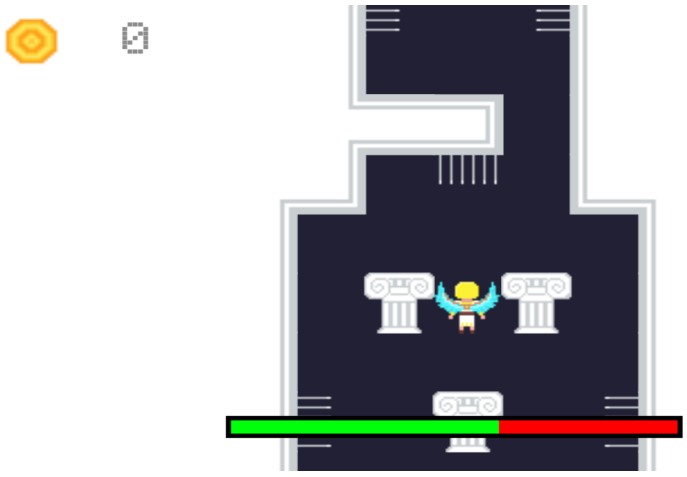

Figure 26: Level 5

3.4 Functionality

3.4.1 Player

The player's movement is influenced by his position in the game world and interaction with the game objects. The player can run on the ground, jump, climb ladders, bounce on vents and even fly, in later parts of the game.

The player has access to **three** weapons, each with different uses, both in combat and in exploration:

Ares's Sword is a melee offensive weapon used to defeat enemies that get too close.

Artemis's Bow is a ranged offensive weapon used to defeat distant enemies.

Athena's Shield is a defensive weapon used to block incoming projectiles.

Each of the three weapons has its strengths and weaknesses, being stronger against a certain type of enemy and weaker against another type of enemy:

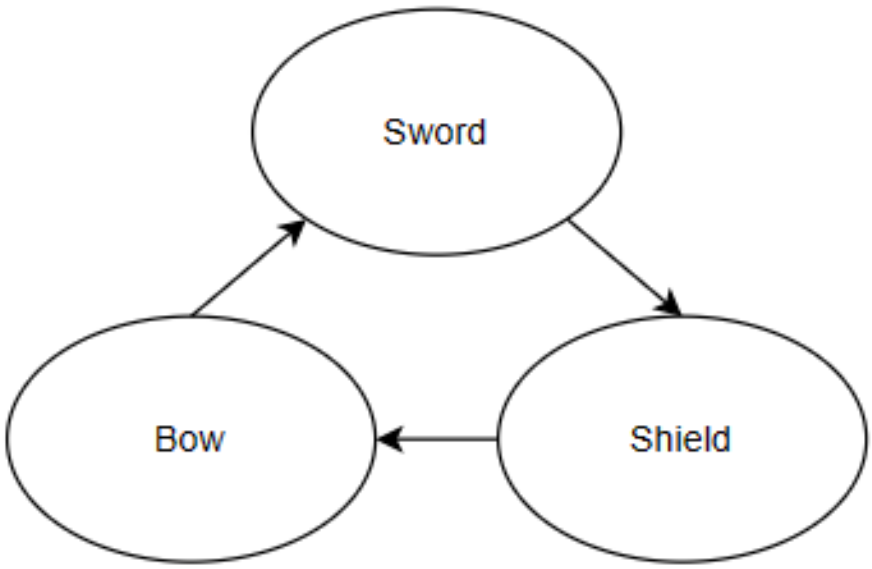

Figure 27: Weapon vulnerabilities

The sword can damage shielded targets, but can't reach distant targets that use bows.

The shield can block arrow projectiles, but can't block sword attacks.

The bow can damage targets that use swords before they get in range to attack, but the arrow projectiles are blocked by shielded targets.

3.4.2 Enemies

There are **three** types of regular enemies that mirror the player's weapons, each with a ground and a flying variation, depending on the part of the game where they are present. There is also a boss enemy that has special movement and attack patterns.

The Sword Guardian is a melee offensive enemy that carries a double bladed sword.

The Bow Guardian is a ranged offensive enemy that constantly shoots ranged projectiles in a certain direction.

The Shield Guardian is a melee defensive enemy that protects other enemies from the player's projectiles.

Medusa, the Gorgon is a boss type enemy that appears during the flying tower defense level and has access to an arsenal of petrifying attacks.

3.4.3 Interactables

While exploring, the player will find diverse objects that they can interact with, simply by walking through them.

Tutorial Books are interactables that teach the player how to play the game.

Coins are interactables that increase the player's score points.

Hearts are interactables that increase the player's health points.

3.4.4 Score

The score is the main way the player can see their progress. Killing enemies and collecting coins award score points, determining the player to take their time in exploring the world in order to obtain a highscore.

3.4.5 Lives

The player starts with **five** lives that can be increased by collecting hearts.

When being hit by an enemy, the player loses one life. However, if the player is hit by a trap, they lose one life and are teleported back to the starting point of the current level, keeping the previous state of the level. This means that defeated enemies and interactables don't respawn.

When reaching zero lives, the game is over and the player has to start over.

4 IMPLEMENTATION

This chapter describes the implementation details of the game's components, offering detailed descriptions of the assets, mechanics and interactions present throughout the game.

4.1 Assets

The solution was implemented using the Unity[1] game engine and the C# programming language. Unity provides useful tools to create 2D games, such as: a physics engine, sprite animation and audio playback. The assets used in developing the game vary from visual elements (sprites, tiles and animations) to game logic elements (prefabs, levels and scripts) and work together to create gameplay mechanics, interactions between game objects and visual effects in order to deliver a well-balanced and functioning game.

4.1.1 Scenes

Scenes [11] represent parts of the game that have different purposes. Scenes can be used to create levels or menus and are usually connected by pressing buttons or triggering a certain condition inside a level. Each scene consists of multiple game objects, each with their own components.

4.1.2 Game Objects and Components

Game objects [12] are the base entities used in Unity to create characters, interactable objects or scenery. A Game Object acts as the interface for anything that resides in the game world, each having their own purpose in the game space.

Every game object consists of multiple components that work together to ensure the object's functionality. For example, in order to make a simple playable character, a game object with the following components is needed: a mandatory **Transform** component, that determines the position, rotation and scale of the object, a **Sprite Renderer** that visually represents the character, a **body** that can interact with other objects, a **Player Input** that allows the character to be controlled by the player and one or multiple **scripts** that dictate how the character behaves in the game world.

4.1.3 Scripts

Scripts are programmable files that dictate the game logic and how the objects interact with each other in the game world. They are used to implement different behaviours inside the game. Examples of behaviours that the scripts can implement: movement, interaction between objects, navigation between scenes.

4.1.4 Sprites

Sprites [13] are 2D graphical objects that are used to represent the game world, including characters, interactable objects, terrain and background. All the sprites used in the game were made by myself. These sprites were saved as PNG files and imported directly into the game engine.

4.1.5 Animations

Animations [14] are visual representations of a sprite's movement. They are used to make characters and objects dynamic and bring the game to life. In Unity, an animation is a group of sprites attributed to a game object. When an animation is triggered by a certain condition, the engine loops through the sprites and creates the impression of movement. Examples of animations: walking, jumping, attacking, etc.

4.1.6 Audio Clips

Sounds are an important part of any game because they imbue the game with life. Background music builds the atmosphere of the game while diverse sounds can make certain moments, like hitting an enemy, parrying an attack or picking up a treasure, feel impactful. In Unity, **Audio Clips** are files that can be attached to game objects to make them produce the desired sounds in specific occurrences.

4.2 Scenes

The game consists of **nine** scenes, split into both menus and levels. The transition between these scenes is made by using buttons, intermediary screens and certain game objects.

4.2.1 Menus

Out of the **nine** game scenes, **four** of them are menus, allowing the player to begin their playthrough, change their settings and quit the game.

Starting Screen is the first scene and the start of the game, having the role to introduce the player to the atmosphere of the game.

Press here to play button - loads the **Main Menu** scene.

Main Menu is the second scene of the game and is the menu from which the player can start/exit the game or access the options menu.

Play button - loads the **Part Selection** scene.

Options button - loads the **Options** scene.

Quit button - exits the game.

Options is the third scene of the game and is the menu from which the player can modify the sound of the game.

Volume slider - modifies the volume of the entire game.

Back to Main Menu button - loads the **Main Menu** scene.

Part Selection is the fourth scene of the game and is the menu from which the player can select which of the **three** parts they can play.

Part I - Explore the temple button - loads the **Level 1** scene.

Part II - Protect the temple button - loads the **Level 4** scene.

Part III - Escape the temple button - loads the **Level 5** scene.

Back to Main Menu button - loads the **Main Menu** scene.

4.2.2 Levels

The other **five** scenes are represented by levels grouped into **three** parts, with each part containing at least one level.

Part I - Explore the temple

This part consists of **three** platforming levels with increasing difficulty and number of mechanics:

Level 1 is the first level of the first part has a simple platformer layout and introduces the movement and the attack mechanics to the player. To finish the level, the player must find the marked exit and walk through it.

Exit Game Object - loads the **Level 2** scene.

Level 2 is the second level of the first part is bigger and more complex than the first one, introducing more mechanics and interactions with the different enemy types. To finish the level, the player must find the marked exit and walk through it.

Exit Game Object - loads the **Level 3** scene.

Level 3 is the third level of the first part is the biggest one and combines all the mechanics available during the platforming part to force the player to use their creativity and intelligence to navigate the plethora of traps and enemies in order to recover the artifact. To finish the level, the player must find the artifact and walk through it.

Wings Game Object - loads the **Level 4** scene.

When the player presses the **Esc** Key, the game will be paused and the **Pause Menu** screen will be opened.

Continue Button - unpauses the game and closes the **Pause Menu** screen.

Restart Button - loads the **Level 1** scene, restarting the current game session.

Options Button - opens the **Options** screen, which has the same functionality as the **Options** scene.

Main Menu Button - loads the **Main Menu** scene, restarting the current game session.

Losing all lives will pause the game and open the **Lose screen**:

Play again? Button - loads the **Level 1** scene, restarting the current game session.

Main Menu Button - loads the **Main Menu** scene, restarting the current game session.

Part II - Protect the temple

This part consists of one tower defense levels with a boss fight at the end:

Level 4 is the only level of the second part and is a platformer level that requires the player to defend the temple from waves of enemies, and finally from Medusa herself, by navigating between lanes while blocking damaging projectiles and eliminating all the enemies that fly towards the player. The player's lives are replaced by a health bar that depletes when the temple's walls are hit by enemies of their projectiles. The boss also has a health bar that must be totally depleted in order to finish the level.

Pressing the **Esc** Key will pause the game and open the **Pause Menu** screen:

Continue Button - unpauses the game and closes the **Pause Menu** screen.

Restart Button - loads the **Level 4** scene, restarting the current game session.

Options Button - opens the **Options** screen, which has the same functionality as the **Options** scene.

Main Menu Button - loads the **Main Menu** scene, restarting the current game session.

When the temple's health bar is empty, the game will be paused and the **Lose screen** will be displayed:

Play again? Button - loads the **Level 4** scene, restarting the current game session.

Main Menu Button - loads the **Main Menu** scene, restarting the current game session.

Part III - Escape the temple

This part consists of one timed escape level and represents the end of the game.

Level 5 is the only level of the third part follows the player in their attempt to escape the collapsing temple, by flying towards the surface, while avoiding obstacles and falling terrain. The player's health is represented by a health bar that depletes every time the player is hit by an obstacle.

Pressing the **Esc** Key will pause the game and open the **Pause Menu** screen:

Continue Button - unpauses the game and closes the **Pause Menu** screen.

Restart Button - loads the **Level 5** scene, restarting the current game session.

Options Button - opens the **Options** screen, which has the same functionality as the **Options** scene.

Main Menu Button - loads the **Main Menu** scene, restarting the current game session.

When the player's health bar is empty, the game will be paused and the **Lose screen** will be displayed:

Play again? Button - loads the **Level 5** scene, restarting the current game session.

Main Menu Button - loads the **Main Menu** scene, restarting the current game session.

When the player reaches the surface, the game will be paused and the **Win screen** will be displayed:

Play again? Button - loads the **Level 1** scene, restarting the current game session.

Main Menu Button - loads the **Main Menu** scene, restarting the current game session.

4.3 Game Objects

Every game scene has a set of mandatory game objects that create the game space and its functioning logic. Each of these objects consists of more or less components with different purposes and some of them even have other children game objects linked to them.

4.3.1 Game Session

The **Game Session** object is used to keep track of changes made to the game space throughout the entire game, even between levels and menus. This game object ensures continuity between scenes and memorises important information such as the lives and score of the player, while also managing the User Interface (UI) of the game.

4.3.2 Tilemap Grid

Tiles [15] are a special type of sprites that are used to represent the background and terrain of the game world. Their main characteristic is that they are usually static and can't be destroyed, being used to create uninteractable game objects. The tile sprites are grouped in a **Tilemap** for easier utilisation and management.

The **Tilemap Grid** object contains multiple Tilemap children, each representing a layer of the game world's tilemap.

Background Tilemap - The background layer that creates scenery behind the game world. It has a Collider component to lock the camera view inside its area.

Platform Tilemap - The terrain layer on which the player and the enemies move freely throughout the game world. It has a Collider component that prevents the player, the enemies and all the projectiles from passing through.

Ladder Tilemap - The layer that the player uses to move between different part of the game scene using vertical ladders that have a Collider which allow the player to move up and down, without gravity, to access new areas of the level.

Traps Tilemap - The trap layer, that uses a Collider to damage the player on contact, resulting in a life loss.

Bouncing Tilemap - The bouncing layer that consists of vents that lift the player upwards to access shortcuts and otherwise unreachable zones.

4.3.3 Player

The Player object is used to represent the playable character controlled by the player during their playthrough. As any other game object, the Player consists of multiple components:

Component Name	**Description**
Transform	Defines the position, rotation and scale of the player
Sprite Renderer	Controls the sprite's appearance and renders it into the game world
Rigidbody 2D	Enables Unity's physics engine, activating the object's mass, gravity, friction and other properties
Capsule Collider 2D	Represents the player's body, enabling collision with walls, enemies, projectiles and interactable objects
Box Collider 2D	Represents the player's feet, enabling collision with the ground and the ladder, used for jumping and climbing
Player Input	Used to create input actions and link them to a certain key or button
Player Script	Used to manage the player's input actions, animations, movements and attacks
Children Objects	The Player object also has multiple children objects that define the player's weapons and their corresponding Transform components

Table 1: Components of the Player game object

4.3.4 Cameras

A **camera** [16] is a device through which the player views the game world. A scene can have multiple cameras but only one camera can be active at any time. Usually, each camera is used for a different purpose and changing cameras can be used to create transition effects.

During the first part of the game, a **State Driven Camera**is used to change between multiple state based cameras. Depending on the player's state (Running, Climbing or Idling), the camera corresponding to that state will activate.

During the third part of the game, a camera is used to mimic following the player. However, when the player hits an obstacle and remains behind, the camera keeps going up, giving the sense of a timed escape.

4.3.5 Scene Persist

The **Scene Persist** object contains a script that allows the scene to keep its current state when the scene is reloaded for the player, this being triggered by the player losing a life because of a trap. This object also has multiple children, represented by the game objects that should keep their state when current scene is reloaded, these being the enemies and the objects that the player interacts with. If the player triggers a trap, the scene is reloaded but the defeated enemies and the objects the player has already interacted with don't reset to their initial state. However, if the player loses all their lives or returns to the main menu, the whole game session will be reloaded, with all the game objects returning back to their original states.

4.4 Scripts

4.4.1 Player Script

The player has a script with different functionalities for each game part. With the game being split into three parts with their own game style, each part has different movement mechanics and animations, requiring slightly modified methods.

Movement Inputs

OnMove - The player's movement input is saved in a variable that is used by other methods to move the character in the game world.

OnJump - The player jumps with the help of the Space key. During the first part, the player can shortly increase their Y position coordinate to jump over obstacles and to reach new parts of the game world. However, the existence of gravity pulls the player to their original Y value after a short time. This action can only be completed while on the ground.

States

Idle - The default state of the player, when no input is provided, corresponding to the Idling animation. This state is present in all three parts of the game but with different animations, with the first part happening on the ground, while the second and third are happening in the air.

Climb - The state in which the player is continuously moving along the ladders as a result of the OnMove input on the Y axis (up-down), only during the first part of the game.

Die - The state in which the player has taken damage and is unable to take actions until the game scene is reset. This state is only present in the first part of the game.

Run - The state in which the player is moving on the X axis (left-right), as a result of the OnMove input, only during the first part of the game.

VerticalFly - The state in which the player is moving on the Y axis (up-down), as a result of pressing the W and S keys, only during the second part of the game.

HorizontalFly - The state in which the player is moving on the X axis (left-right), as a result of pressing the A and D keys, only during the third part of the game.

Weapon Inputs

OnSwordAttack (Left Mouse Button) - Instantiates a sword game object in front of the player that damages all enemies that it comes in contact with, triggering the SwordAttacking animation. After the animation time is over, the animation is deactivated, the instance created is destroyed and the player can trigger another input attack action.

OnBowAttack (Right Mouse Button) - Instantiates an arrow game object that flies in the direction the player is facing, triggering the BowAttacking animation. The instance is destroyed after colliding with another object, damaging any enemy hit. After the animation time is over, the animation is deactivated and the player can trigger another input attack action.

OnShieldParry (C) - Instantiates a shield game object in front of the player, parrying all incoming enemy attacks and projectiles that it comes in contact with, triggering the ShieldParrying animation. After the animation time is over, the animation is deactivated, the instance created is destroyed and the player can trigger another input attack action.

More details about the player's weapons and their mechanics can be seen in the table below:

Weapon Name	**Ares' Sword**	**Artemis' Bow**	**Athena's Shield**
Weapon Type	Melee Offensive	Ranged Offensive	Melee Defensive
Keybind	Left Mouse Button	Right Mouse Button	C
Upside	Effective against shielded enemies	Long range	Blocks enemy projectiles
Downside	Short range	Ineffective against shielded enemies	Ineffective in combat

Table 2: Player's weapons

Player Instances

Instances are game objects that are created only when certain conditions are met and are not present in the game space from the start.

A **sword** is instanced every time the player presses the Left Mouse Button. If the instance collides with an enemy, it damages them, plays a sound and destroys itself.

An **arrow** is instanced every time the player presses the Right Mouse Button. If an allied instance collides with an enemy, it damages them and a sound is played. On any type of collision, the arrow instance in destroyed.

A **shield** is instanced every time the player presses the C key. If an enemy projectile collies with the shield, that projectile is destroyed.

4.4.2 Enemy Scripts

The regular enemies have 2 scripts, one dictating their movement, and the other, their attack pattern. Enemies are present only in the first 2 parts of the game and function differently based on the parts they are present in. However, the boss enemy is more complex and has a different script compared to the regular enemies.

EnemyMovement

During the first part of the game, the enemy moves continuously along a piece of ground until they reach a wall or an edge, immediately turning around and continuing along the same path. This script is only used by the Sword Guardians that patrol certain areas of the game scene.

During the second part of the game, the Sword and Shield Guardians move all the way towards the players, while the Bow Guardian only moves until it is in position to start shooting.

EnemyAttack

During both of the combat parts of the game, the Bow Guardian constantly shoots arrows in the direction they are facing, at a set interval. However, in the second part, the Bow Guardian doesn't start attacking until it reaches a fixed position.

More details about the enemies and their mechanics can be seen in the table below:

Enemy Name	**Sword Guardian**	**Bow Guardian**	**Shield Guardian**
Weapon Type	Melee Offensive Weapon	Ranged Offensive Weapon	Melee Defensive Weapon
Ground Movement	Moves along the ground	Immobile	Immobile
Flying Movement	Moves until it reaches the player	Moves until it is in position to shoot	Moves until it reaches the player
Effect	Damages on collision	Shoots arrows	Immune to arrows
EnemyMovement Script	Ground and Flying	Flying	Flying
EnemyAttack Script	-	Ground and Flying	-
Strength	Shield	Sword	Bow
Weakness	Bow	Shield	Sword
Score Points	100	100	100

Table 3: Regular Enemies

Boss

The boss enemy spawns during the second part of the game after defeating all the enemies that are attacking the temple. As soon as it spawns, the boss starts teleporting randomly on one of the five lanes and shoots an arrow projectile towards the player. After repeating this attack pattern five times, the boss teleports in the middle lane and unleashes three shockwaves that stun the player if it hits them. While stunned, the player can't move, attack or parry. However, if the player successfully parries the shockwaves, the boss is stunned instead, resulting in an opportunity for the player to unleash several attacks and severely damage the boss.

More details about the boss and its mechanics can be seen in the table below:

Boss Name	**Medusa, the Gorgon**
Weapon Type	Ranged Offensive Weapon
Movement	Teleports on a random lane before attacking
Attack	Shoots an arrow on the current lane
Special Attack	Every five attacks, moves to the middle lane and unleashes three shockwaves that stun the player is not parried. If parried, Medusa is stunned instead.
Score Points	1000

Table 4: Boss Enemy

Enemy Instances

An **arrow** is instanced every time the enemy Arrow Guardians or the Boss attacks. If an enemy arrow instance hits the player, the player loses a life or the temple's health is lowered. However, if the instance hits the player's shield, the player doesn't take any damage. On any type of collision, the arrow instance in destroyed.

A **shockwave** is instanced every time the Boss uses its special attack. If the player is hit by a shockwave, they are stunned for a few seconds. However, if the player parries the shockwaves with their shield, the boss is stunned for a few seconds instead.

4.4.3 Interactables Scripts

CoinPickUp - The script that describes the coin interactable. When the player walks through a coin, a sound is played, the player gains points and the coin instance in destroyed.

HeartPickUp - The script that describes the heart interactable. When the player walks through a heart, a sound is played, the player gains a life and the heart instance in destroyed.

4.4.4 Game Logic Scripts

Besides the scripts that are linked to game objects that the player can interact with, there are several scripts that dictate the game logic and how it all flows together.

GameSession

This is the script connected to the **Game Session** game object, and represents the main script of the game. It enables the game session to persist when losing a life because of a trap, resulting in a game scene reload and it only resets when the player has lost all lives, resulting in a game session reload. It also holds information about the User Interface and its components, like total score, lives and health bars, while also providing methods to add and subtract from their values.

ScenePersist

If the **GameSession** script is connected to the whole game, the **ScenePersist** script is connected to the current game scene, saving the current level state when the player loses a life by triggering a trap. When the game session is reloaded, the GameSession script calls the method from the ScenePersist to also reset all the game scenes to their original state.

4.5 Sprites and Animations

Sprites and animations are the components that imbue life to any game and are the first things that one notices when looking at a video game for the first time. If the functionalities are the body of the game, then the sprites and animations are the soul.

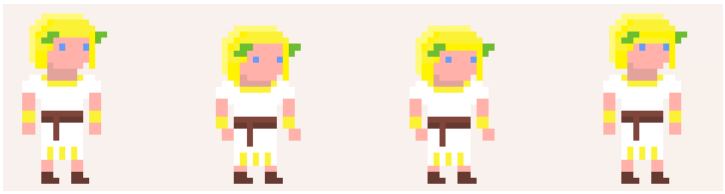

Figure 28: Player Idling Animation

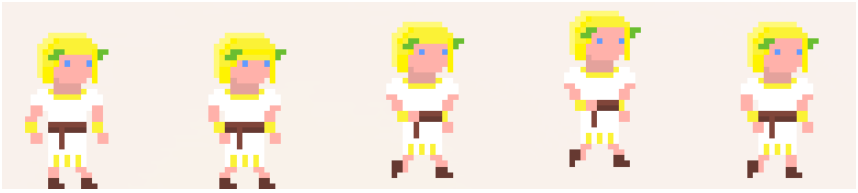

Figure 29: Player Running Animation (Part 1)

Figure 30: Player Vertical Flying Animation (Part 2)

Figure 31: Player Horizontal Flying Animation (Part 3)

Figure 32: Player Climbing Animation

Figure 33: Player Sword Attacking Animation

Figure 34: Player Bow Attacking Animation

Figure 35: Player Shield Defending Animation

Figure 36: Player Flying Sword Attacking Animation

Figure 37: Player Flying Bow Attacking Animation

Figure 38: Player Flying Shield Parrying Animation

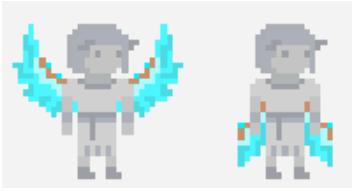

Figure 39: Petrified Animation

Figure 40: Sword Guardian Idling Animation

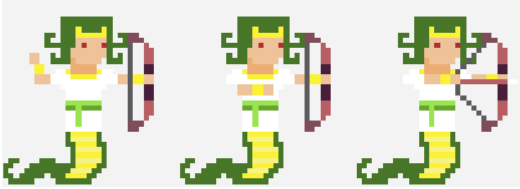

Figure 41: Bow Guardian Attacking Animation

Figure 42: Shield Guardian Idling Animation

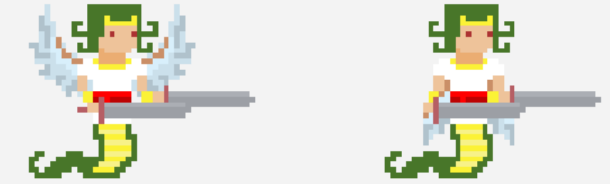

Figure 43: Flying Sword Guardian Idling Animation

Figure 44: Flying Bow Guardian Attacking Animation

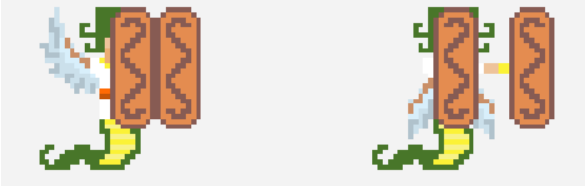

Figure 45: Flying Enemy Shield Parrying Animation

Figure 46: Medusa Attacking Animation

Figure 47: Medusa Stone Gazing Animation

Figure 48: Coin Idling Animation

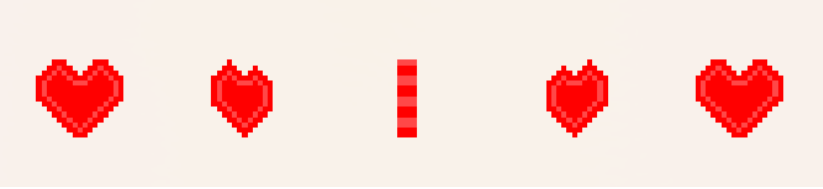

Figure 49: Heart Idling Animation

5 TESTING AND EVALUATION

This chapter presents the equipment used to test the performance and the duration of the game, as well as the results of testing all the game's mechanics.

The game has been tested on a laptop with the following specifications:

CPU - Intel Core i7-9750H CPU @ 2.60GHz

RAM - 24GB of RAM

GPU - NVIDIA GeForce GTX 1650 with 16GB VRAM

Resolution - 1920x1080 pixels

The performance is consistent at a steady 48 frames per second during the entire playthrough. This is a good result as the game doesn't require many frames per second to be optimally played due to its simple mechanics that allow a bigger response time.

The game took around 30 minutes to complete, including collecting every coin and defeating every enemy encountered. This time may be lower or higher, based on the player's speed and skill at avoiding taking damage and being hit by traps.

During the testing period, which included multiple game completions, all transitions and functionalities of the game have been tested with no bugs encountered. The tested functionalities can be seen in the following tables:

Scene/Screen Name	Button	Description	Status
Starting Screen	**Press here to play**	Loads the Main Menu scene	Working
Main Menu	**Options**	Loads the Options scene	Working
Main Menu	**Play**	Loads the Part Selection scene	Working
Main Menu	**Quit**	Exits the game	Working
Options	**Back to Main Menu**	Loads the Main Menu scene	Working
Part Selection	**Back to Main Menu**	Loads the Main Menu scene	Working
Part Selection	**Part 1**	Loads the Level 1 scene	Working
Part Selection	**Part 2**	Loads the Level 4 scene	Working
Part Selection	**Part 3**	Loads the Level 5 scene	Working

Table 5: Main Menu Transitions Testing

Scene/Screen Name	Button	Description	Status
Pause Menu	**Continue**	Unpauses the game	Working
Pause Menu	**Options**	Loads the in-game Options screen	Working
Pause Menu	**Restart**	Reloads the first scene of the current part	Working
Pause Menu	**Main Menu**	Loads the Main Menu scene	Working
Options	**Back**	Returns to the Pause Menu screen	Working
Lose Screen	**Play Again?**	Reloads the first scene of the current part	Working
Lose Screen	**Main Menu**	Loads the Main Menu scene	Working
Win Screen	**Play Again?**	Reloads the Level 1 scene	Working
Win Screen	**Main Menu**	Loads the Main Menu scene	Working

Table 6: In-Game Transitions Testing

Name	Description	Status
Run	Pressing the A/D key moves the player in the chosen direction	Working
Jump	Pressing the Space key launches the player upwards for a short period of time	Working
Climb	Pressing the W/S key while on a ladder moves the player in the chosen direction	Working
Sword Attack	Pressing the Left Mouse Button triggers an attack in the direction the player is facing, destroying all enemies hit	Working
Bow Attack	Pressing the Right Mouse Button launches an arrow in the direction the player is facing, destroying the first non-shielded enemy hit	Working
Shield Parry	Pressing the C key creates a shield that blocks all incoming projectiles for the duration	Working
Die	Touching a trap makes the player lose a life and be transported at the start of the current level	Working
Vertical Fly	Pressing the W/S key moves the player one lane upwards/downwards	Working
Horizontal Fly	Pressing the A/D key moves the player one lane to the left/right	Working

Table 7: Player Functionality Testing

Name	Description	Status
(Flying) Sword Guardian Movement	Moves along a certain path	Working
(Flying) Bow Guardian Attack	Periodically shoots arrows in the facing direction	Working
(Flying) Shield Guardian Parry	Blocks all incoming player arrows	Working
Flying Bow Guardian Movement	Moves to a certain position and then stops	Working
Flying Shield Guardian Movement	Moves to a certain position and then stops	Working
Medusa Movement	Moves randomly on one of the 5 lanes	Working
Medusa Bow Attack	After moving, shoots an arrow on the current lane	Working
Medusa Stone Gaze Attack	Teleports to the middle lane and shoots 3 shockwave projectiles that stun the player, if they hit the player, or stun Medusa, if parried by the player	Working

Table 8: Enemy Functionality Testing

From	To	Trigger	Status
Level 1	**Level 2**	Going through the tunnel at the end of the level	Working
Level 2	**Level 3**	Going through the tunnel at the end of the level	Working
Level 3	**Level 4**	Picking up the wings	Working
Level 4	**Level 5**	Defeating Medusa	Working

Table 9: Level Transitions Testing

6 CONCLUSIONS

This chapter presents the conclusions drawn from developing the solution and some personal impressions regarding the entire process.

The proposed solution is my first attempt at creating a fully functional video game and, personally, it was a huge success. When I started developing this project, I had limited video game development knowledge. This, combined with never working in the Unity game engine before, lead to a long learning process, filled with trial and error. However, seeing the final solution with my own eyes, made it all worth it.

Although the game looks simple, both visually and mechanically, the entire development process exceeded 100 hours and was filled with deep research, countless adjustments, bug fixes and even art lessons. The result is a video game that tells a story through countless sprites, animations and mechanics, a story of a young man who wants to change the world with his craft.

Despite the long development process, the game is relatively simple and can be improved on many levels, by adding additional functionalities and more complex visuals.

7 FUTURE DEVELOPMENT

This final chapter presents ideas of further improvements that can be brought to the game but, due to their complexity and time constraints, didn't make it into this version of the game.

The game's atmosphere can be improved, both in the graphical and the sound departments, by creating more detailed sprites and animations, most notably by increasing their number of pixels and making use of more fitting sound for each interaction, making the game more immersive.

With the main mechanics of the game already done, the game's scalability is not something hard to achieve. Bigger and more complex levels, new weapon and enemy types and new mechanics are all possible by adding on top of the existing project. New levels can have improved mechanics, like water breathing and exploring dark zones that obscure vision, as well as exciting interactable objects like levers that unlock hidden areas and empowering potions.

A highscore system would have added a sense of replayability to the game by making the player want to beat their own highscore and become better. However, due to breaking the game in parts, this feature was hard to implement and could only work for each individual part.

Another feature that could improve the gameplay is the minimap mechanic that shows the player the whole layout of the level. Considering the levels' simple and intuitive design, the absence of this feature doesn't take away from the game. However, a more complex game could make use of such a simple, yet effective, feature to help the player easily explore every part of the game.

Although a bit more complex, an upgrade system can increase the depth of the game by adding progression to the player's character.

8 REFERENCES

[0] The Video Game Industry: Formation, Present State, and Future, Edited by Peter Zackariasson, Timothy Wilson, 2012

[1] Unity, URL: https://unity.com/, Accessed 20-05-2023

[2] Super Mario Bros., URL: https://supermario-game.com/, Accessed 21-05-2023

[3] Spelunker, URL: https://www.spelunker-hd.com/, Accessed 21-05-2023

[4] Bloon Tower Defense, URL: https://ninjakiwi.com/Games/Tower-Defense/Bloons-Tower-Defense.html, Accessed 21-05-2023

[5] Plants vs. Zombies, URL: https://www.ea.com/ea-studios/popcap/plants-vs-zombies, Accessed 21-05-2023

[6] Hades, URL: https://www.supergiantgames.com/games/hades/, Accessed 24-05-2023

[7] Immortal Fenyx Rising, URL: https://www.ubisoft.com/en-gb/game/immortals-fenyx-rising, Accessed 24-05-2023

[8] Cuphead, URL: https://www.cupheadgame.com/, Accessed 26-05-2023

[9] Hollow Knight, URL: https://www.hollowknight.com/, Accessed 26-05-2023

[10] Spelunky, URL: https://spelunkyworld.com/original.html, Accessed 28-05-2023

[11] Unity Scene, URL: https://docs.unity3d.com/ScriptReference/SceneManagement.Scene.html, Accessed 5-06-2023

[12] Unity Game Object, URL: https://docs.unity3d.com/ScriptReference/GameObject.html, Accessed 5-06-2023

[13] Unity Sprite, URL: https://docs.unity3d.com/ScriptReference/Sprite.html, Accessed 5-06-2023

[14] Unity Animation, URL: https://docs.unity3d.com/ScriptReference/Animation.html, Accessed 5-06-2023

[15] Unity Tile, URL: https://docs.unity3d.com/ScriptReference/Tilemaps.Tile.html, Accessed 5-06-2023

[16] Unity Camera, URL: https://docs.unity3d.com/ScriptReference/Camera.html, Accessed 5-06-2023