|  |
| --- |
| Logo, company name  Description automatically generated**Yr 12 ATAR Computer science**  **2020 assessment task 2** |

**Task:** Developing an Interactive Video Game using Python and Pygame

**Type:** Project

**Due Date: 20th April 2024**

**Weighting:** 20%

**Marks:** 50

**Overview**

The Year 12 Computer Science ATAR project challenges students to design and implement an interactive video game using Python and the Pygame library. This project aims to assess students' understanding and application of various computer science concepts, including control structures, data types, modular coding, operators, data structures, development processes, good programming practices, algorithm representation, code structure, error detection and correction, and object-oriented programming.

**Project Requirements:**

1. Game Concept and Design (8%)
   * Students are expected to create a comprehensive game concept with a detailed description of the game's objective, rules, and mechanics. Design documentation should include visual aids such as sketches and diagrams, providing a clear understanding of the game's structure.
2. Program Control Structures (10%)
   * Students should implement appropriate control structures, such as loops and conditionals, to effectively manage the flow of the game. Code should be well-organized, promoting clarity and a logical structure that enhances the user experience.
3. Data Types and Variables (8%)
   * Utilize a variety of data types (integers, strings, lists, dictionaries) based on the requirements of the game. Variables should be aptly named and used strategically for storing and managing game-related information.
4. Modular Coding (10%)
   * Code should be organized into functions and classes, each serving specific tasks within the game. Functions should have clear input parameters and return values, enhancing code readability and maintainability.
5. Operators (7%)
   * Demonstrate proficiency in using arithmetic, relational, and logical operators within the game. Show how these operators contribute to the game's functionality and logic.
6. Data Structures (7%)
   * Efficiently employ arrays and dictionaries to store and manipulate game data. Clearly demonstrate how these data structures enhance the performance and functionality of the game.
7. Development Processes and Good Programming Practices (15%)
   * Implement good programming practices, including input validation, clear mainline, one task per subroutine, use of stubs, and version control. Address social and ethical issues related to game development, showcasing an awareness of responsible coding practices.
8. Algorithm Representation and Code Structure (7%)
   * Provide pseudocode or flowcharts illustrating key algorithms used in the game. Ensure that the actual code aligns with the proposed design specifications, demonstrating a seamless translation from design to implementation.
9. Error Detection and Correction (10%)
   * Implement robust error handling mechanisms for syntax, logic, and runtime errors. Use techniques such as try-catch blocks for runtime errors, input validation for user errors, and thorough testing to identify and correct errors. Demonstrate a systematic approach to handling and rectifying issues.
10. Object-Oriented Programming Concepts (8%)
    * Implement classes and objects to represent game entities. Utilise attributes, methods, abstraction, instantiation, and inheritance where applicable. Showcase a clear understanding of object-oriented programming principles and their application in the game.

**Submission Guidelines:**

* Students are required to submit a well-documented Python codebase along with any supplementary documents related to the design and planning of the game.
* Submissions should be made through the designated platform by the specified deadline.

**Marking Criteria:**

Each category in the marking key corresponds to a percentage of the total grade, and the overall assessment will be based on the comprehensive evaluation of the project in accordance with the specified criteria.

**Important Dates:**

* Project Start Date: 8th February 2024
* Project Submission Deadline: 20th April 2024

**GAME DEV DOCUMENT**

# **Introduction**

“Forestfall Inferno’s Descent” is a rogue-lite, indie platformer game inspired by the videogames, “Dead Cells” developed by Motion Twin, “Vaegrant” developed by DaFluffyPotato, as well as “Super Mario” by Nintendo. It combines the elements of exploration, “choose your own pace” combat, progression mechanics, physics mechanics, and character development/growth (will be introduced in later stages of the game). The game was developed to challenge the player utilising the harsh environment/level generation and the difficulty of each level (mechanics will be applied in near-future updates).

# **Gameplay Overview**

## Core Mechanics

* **Combat:** The player engages in fast-paced combat using a variety of weapons, spells, as well as perks of the player to defeat the enemies and complete each level of the game.
* **Exploration:** The game alpha edition includes 5 levels (1 village level, 2 main gameplay rooms, 2 level transition/save rooms where the player can save their game progress, unfortunately will not be included in this version of the game). The player completes each level by traveling through each room until they’ve reached the end of the map to win the game (will be introduced in later stages of the game).
* **Survival:** Forestfall incorporates many puzzles and scenarios as well as numerous aggressive enemies that require the player to survive each, otherwise they will be forced back to the beginning of the room.
* **Progression/Player Growth:** The player has a perk/level up system, that acquired points (by defeating enemies) can be spent on to unlock magic abilities/spells, and improve the character’s statistics (health, stamina, speed, etc…).

## Level Generation

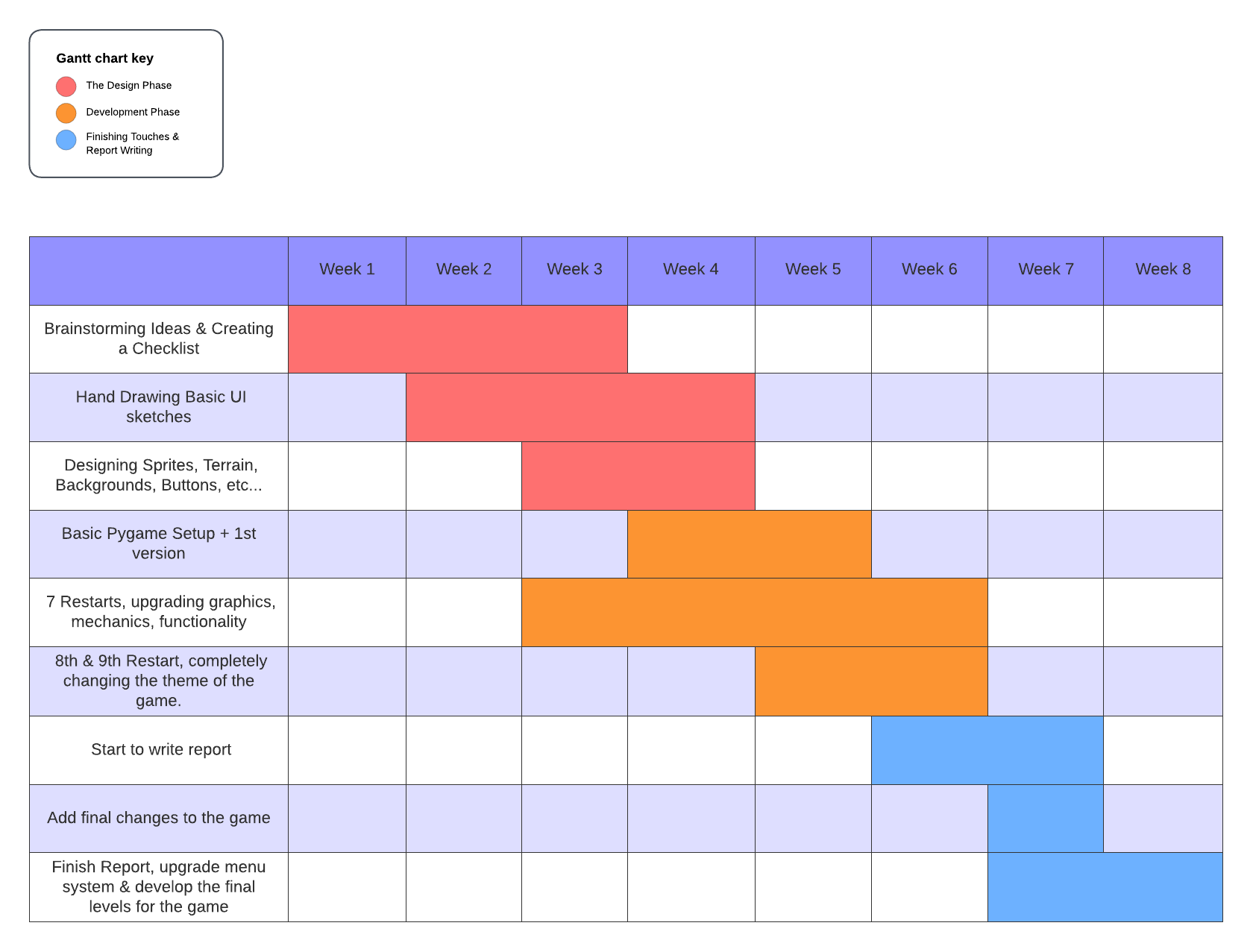
* **Level Generation & Creation:** The levels are generated in a specific set order that increase in difficulty. Each level difficulty set (easy, medium, hard) is generated in a set of three rooms. The level difficulty groups are separated by a rest room where the game can be saved.
* **Loot, Weapons, & Points Generation:** Loot and weapons are assigned to specific treasure rooms that are accessed through a gate/portal in at least one room of each difficulty. The statistics of the weapons improve as the difficulty increases. The points gained by defeating enemies are set/assigned to specific enemies, e.g. Goblin = 10 points, Phantom = 45 points.

# **Important Note**

For the game, some of the sprites for example the enemy sprite, is not my own. For that sprite in particular I acquired it from DafluffyPotato’s Ninja Game tutorial and from itch.io, as allowed by the teacher. Further, the framework (descriptions of the steps required to develop the code, not the code itself) for the physics engine, animating the player, the level editor, loading the player and the enemies, were all provided by the Ninja Game tutorial, as well as numerous devlogs made by him. Currently the game, I have used “Decayed into Darkness” theme song ‘Genshin Impact’ temporarily as it suits the theme of the game. Once I have time, I will compose my own music pieces for the game. The music used for the village level I composed using BandLabs. **The pause menu for the game is mysteriously not functioning properly, however the pause function itself is working.**

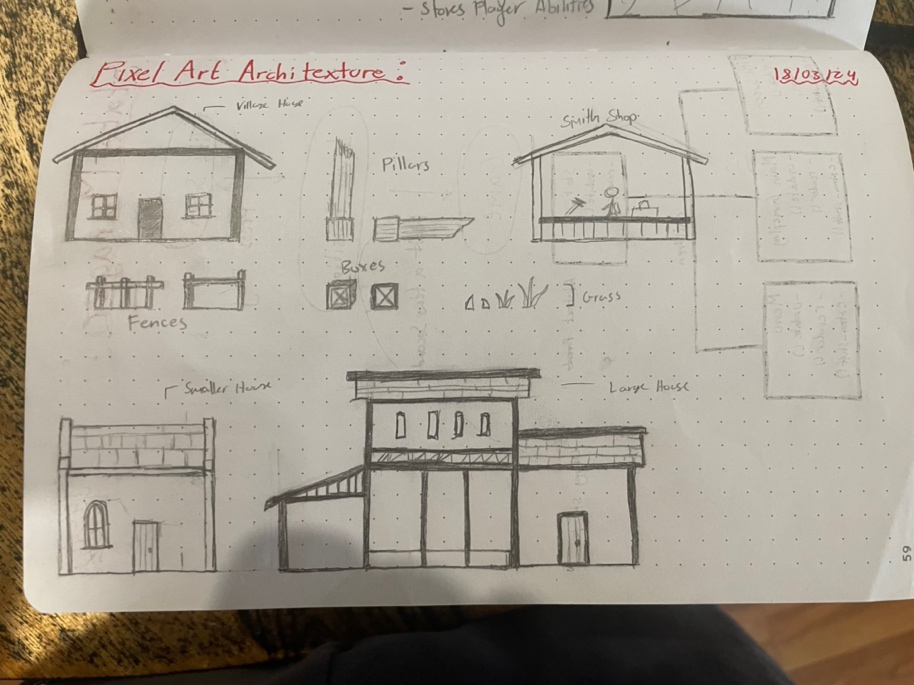
# **Project timeline/Planning**

To efficiently organise the aspects of the game that I needed to complete within the two-month period; I created a ‘Gantt Chart’ in the first week to visually represent and organise the essential milestones I required to complete at a specific point in the development of the game. The ‘Gantt Chart’ I created is this:

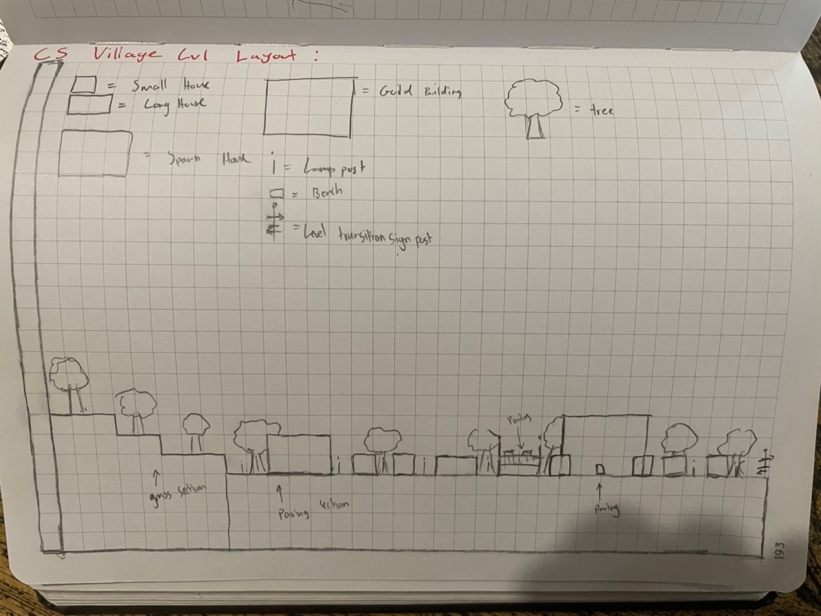
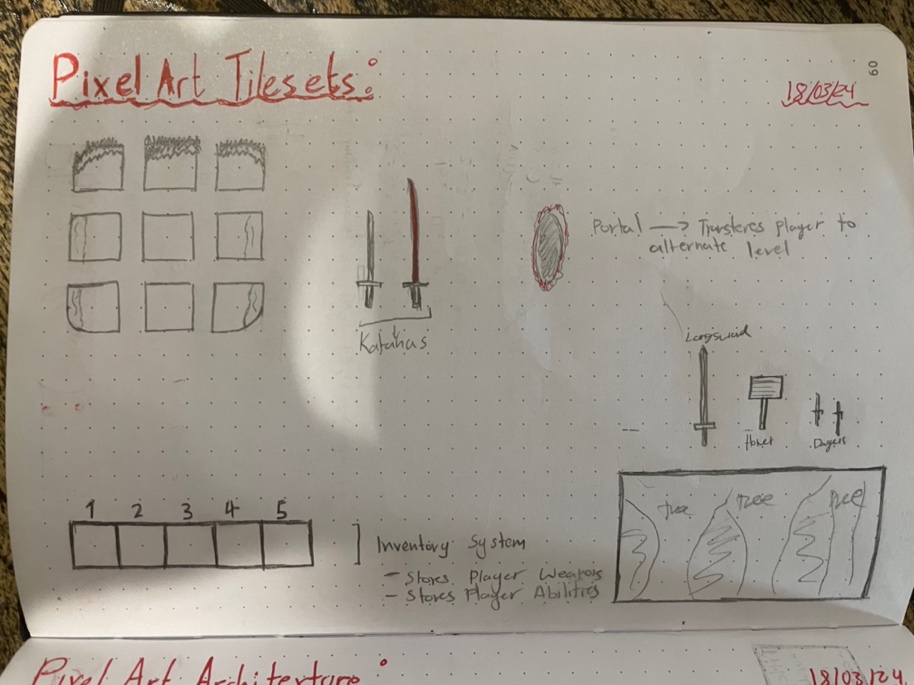


# **Game Design & Development**

I had to take many steps to develop, design, and implement my desired ideas into the game. Such steps involved sketching the graphical aspects, levels, terrain, architecture, characters, and enemies by hand in my book, then converting them to digital images.



Here are some of the sketches I drew during the design phase of the project.



# **Storyline**

“Forestfall” begins on the overworld, where there are forests, temples and jungles. As the player progresses and leaves the overworld, the player travels to the underground caverns, where there are caves, abandoned mining shafts, and dungeons, all set under the “medium” difficulty. The player then travels to the “Underworld” where there are fire pits, hell dungeons, lava rivers. The underworld level was developed and designed based on Dante’s “Inferno” which depicts the circles of “Hell”. Each level room has enemies that operate based on the difficulty providing a challenge to the player.

The overall storyline of the game is that the protagonist (the main character), is an adventurer (there are different characters) that is handed a guild quest that instructs him to travel to the depths of Hell and defeat the Devil himself. The quest was assigned to him/her by the King as a royal decree. If the player fails to defeat the Devil, the nation is engulfed in darkness and the Game Ends (game over). The player is recommended to bypass all floors and levels as quickly as they can (no formal time system has been developed).

# **Characters**

* **The Main Player:** The user can select one of three characters; A Samurai, Assassin, Mage, or a Knight. All the selectable characters can utilise special perks and spells.
* **Passive NPCs’:** There are many passive NPC’s that spawn throughout the levels, mainly present in the introduction level, as it is set in a Castle Town/City. Each NPC has dialogue mechanics that the player can interact and communicate with.
* **The Enemies:** Each of the levels has a set type of enemies that challenge the player. Level 1 (Easy): Goblins, Trolls, & 1x Phantom Knight. Level 2 (Medium): Phantom Knights, Skeletons, 3x Undead Berserker. Level 3 (Hard): Demons, Tortured Souls, Phantoms, Fire Mages.
* **The Final Boss:** The Devil (Satan) + Demon Minions

# **Controls**

**Movement:** W, S = Look Up or Down. A, D = Move Left or Right. Space = Jump. Left Shift = Dash/Dodge.

**Combat:** Left Mouse Click = Weapon Combat Moves. Right Mouse Click = Spell Use/Activation.

**Interaction:** I = Open Inventory/Upgrades Menu. E = Talk/Interact with Objects, Items, NPC’s. Escape = Pause Menu/Pause the Game.

**The Mouse is used to interact with buttons, and other main aspects of playing the game.**

# **Future Updates**

In the near future, for later updates, I will develop a further range of both passive NPCs and enemies as well as implementing a solid storyline, new levels that follow the storyline. The levels I will be adding are, a jungle, desert, dungeon, a difficult hell inferno level, as well as a boss room. Alongside developing the new rooms, I will create a game saving function that is only accessible during the rest rooms between levels. I am planning on creating DLC packs that add additional weapons, levels, enemies, and potentially characters. Each DLC pack will have a unique aspect that is not accessible if the user is playing the base game.

# **Bibliography/Citations**

DafluffyPotato (2023) *Pygame platformer tutorial - full course*, *YouTube*. Available at: https://www.youtube.com/watch?v=2gABYM5M0ww&t=3959s (Accessed: 20 April 2024).

DafluffyPotato (2020) *Collisions/tiles/physics - pygame tutorial: Making a platformer EP. 3*, *YouTube*. Available at: https://www.youtube.com/watch?v=abH2MSBdnWc&t=976s (Accessed: 20 April 2024).

DafluffyPotato (2020b) *Framework and A level editor! - python/pygame devlog #1*, *YouTube*. Available at: https://www.youtube.com/watch?v=8SA5mOk7i1s&t=173s (Accessed: 20 April 2024).

DafluffyPotato (2020c) *Water, wind, and grass - python/pygame devlog #2*, *YouTube*. Available at: https://www.youtube.com/watch?v=QXVu6xhcpVQ&t=545s (Accessed: 20 April 2024).

DafluffyPotato (2020b) *Dynamic leaves, particles, and Parallax - Python/pygame devlog #3*, *YouTube*. Available at: https://www.youtube.com/watch?v=7A7yeIm\_278 (Accessed: 20 April 2024).

DafluffyPotato (2020d) *Screen shake - pygame tutorial*, *YouTube*. Available at: https://www.youtube.com/watch?v=3nhQLJq0Lwk (Accessed: 20 April 2024).

N/A (2022) *How to make a menu screen in PYGAME!*, *YouTube*. Available at: https://www.youtube.com/watch?v=GMBqjxcKogA (Accessed: 20 April 2024).

DafluffyPotato (2020d) *Menus - pygame tutorial*, *YouTube*. Available at: https://www.youtube.com/watch?v=0RryiSjpJn0&t=382s (Accessed: 20 April 2024).

ClearCode (2022) *Creating a mario maker style game in Python*, *YouTube*. Available at: https://www.youtube.com/watch?v=qYomF9p\_SYM&t=22898s (Accessed: 20 April 2024).

CodersLegacy (2023) *Creating a menu system in Pygame (the easy way)*, *YouTube*. Available at: https://www.youtube.com/watch?v=bHGsbKTg8po&t=243s (Accessed: 20 April 2024).

N/A (no date) *Tile set and BG pack - 16x16 tilesets by Biganon 1*, *itch.io*. Available at: https://biganon-1.itch.io/tile-set-and-bg-pack?download (Accessed: 20 April 2024).