Technical Design Document – Outline

# Title Page

# Document History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author(s) | Changes |
| 1 | March 5th, 2024 | Hadlee Bullerwell |  |
|  |  |  |  |
|  |  |  |  |

# Table of Contents

Table of Contents

[1 Title Page 1](#_Toc160728056)

[2 Document History 1](#_Toc160728057)

[3 Table of Contents 1](#_Toc160728058)

[4 Game Summary 1](#_Toc160728059)

[5 Development Environment 1](#_Toc160728060)

[5.1 Development Hardware 1](#_Toc160728061)

[5.2 Programming Languages 2](#_Toc160728062)

[5.3 Development Tools 2](#_Toc160728063)

[5.4 External Code 2](#_Toc160728064)

[5.5 Game Engine 2](#_Toc160728065)

[6 Architectural Analysis 2](#_Toc160728066)

[6.1 Classes 2](#_Toc160728067)

[6.2 Behavioral Analysis 3](#_Toc160728068)

[6.3 Game Loop 3](#_Toc160728069)

[7 Technical Risks 4](#_Toc160728070)

# Game Summary

“Magic Cat-Cademy” invites players into the mesmerizing land of Felandia where magic and danger coexist. Embark on a heroic quest alongside Lucy, a gray and black tabby cat, to rescue her feline companions from the clutches of the menacing hellhounds. As you navigate through the levels, Lucy’s magical abilities become the key to overcoming hardships, unveiling a captivating adventure, and forging new friendships. This game blends the charm of a magical world and strategic gameplay, creating an immersive experience that draws in players of any age.

Gear up for an exciting adventure with Lucy, where every step of the journey unravels the power of her magical abilities. “Magic Cat-Cademy” promises an engaging adventure where players will venture through the land of Felandia and rescue it from the devious hellhounds. Prepare for an immersive experience that draws you in and captivates you as each moment unfolds a new chapter for the land of Felandia.

# Development Environment

## Development Hardware

List the hardware resources (including operating system) needed for game development.

* Windows 11
* 16GB Memory
* Lots of hard drive space

## Programming Languages

Specify the primary programming language(s), including version (e.g. C++03), to be used for development. Also specify any scripting languages that are used (if applicable).

C++ - Version 23

JSON – Most recent version

## Development Tools

List the software needed for development, such as IDEs, compilers, debuggers, version control software, graphical editors, sound editors, etc.

Visual Studio – Main IDE for coding

Visual Studio Code – Used to edit JSON files

Aseprite – Created animation and sprites

Free Texture Packer – Created spritesheets

## External Code

List the libraries to be used, including the URLs of the official library site, and the version number of the library used in this project.

SFML v2.60 - <https://www.sfml-dev.org/>

## Game Engine

Describe the game engine to be used (or implemented), as well as modifications that need to be done to it.

GEX Game Engine – Developed in Prog1266. An entity-component-system-based game engine.

The only modifications I have added so far, are adding new components.

# Architectural Analysis

## Classes

Describe the classes that will have to be implemented. For each class, provide:

* Its responsibilities
* How it collaborates with other classes

|  |  |  |
| --- | --- | --- |
| Class | Responsibilities | Collaborations |
|  |  |  |
|  |  |  |
|  |  |  |

Present class diagrams that show the relationships between classes. Show only the most essential attributes and methods for each class.

## Behavioral Analysis

Present statecharts, flow charts (activity diagrams), sequence diagrams, etc. that model complicated behavior. If your game has actors that implement a state machine, this would be the section where you’d present the statechart.

## Game Loop

Describe, in order, the sequence of activities that happen during each game loop. You must document this even if you’ll be using the “Clown Cannon” game engine.

|  |  |
| --- | --- |
| Steps - | What Happens - |
| Update entity manager | If any entities were destroyed it destroys them, if any entities were added it adds them. |
| Move the world view | The game is a side scroller, so this moves the world view to the left. |
| Moves all objects | This allows object movement, so when I press a key per say, my character will move. |
| Checks collisions | This consistently checks if any characters/obstacles collide. |
| Updates animations | Updates the animations of all characters/obstacles. |
| Checks lifespans | Checks the lifespans to see if anything needs to be removed. |
| Keeps players in bounds | Ensures the player cannot go out of the world bounds |
| Check the players states | Checks the players state, such as for changing animation if they are jumping/walking, etc. |
| Draws lives | Continuously draws the lives, updating when one is lost. |

# Technical Risks

List all technical risks that could make it difficult or impossible to complete the game. Examples:

* Uncertainty on how to implement a certain feature
* Uncertainty on if a certain feature can be executed fast enough in real time
* First time using a certain library

|  |  |  |
| --- | --- | --- |
| Risk | Severity | Mitigation (what is to be done to eliminate or minimize this risk) |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |