Fightn’ 5

Justin Gewecke

CST-451 Capstone Project Requirements Document

Grand Canyon University

Instructor: Professor Amr Elchouemi

Revision: 1.0

Date: 10/2/2022

**ABSTRACT**

This project’s goal is to make a fun video game that can be made within the timespan of a month. The video game will only be available on Windows machines; the application window will function like other applications, such as Spotify, Chrome, and Paint in how you can close, minimize, and expand the application. The game will be accessible on most (if not all) windows configurations.

This project will be made using the Unity Game Engine. Essentially, this engine will create the visual GUI for my game that users can interact with. However, this GUI is a blank canvas that requires coding to add any functionality to it. To accomplish this, I will utilize C# scripts that can interact with that visual GUI. All the game data and logic will be stored in these C# files. Otherwise, no other external tools or frameworks will be used. As an alternative solution Unity, MonoGame is also a possible solution. Instead of being an engine, MonoGame is a C# framework that make it easy to make a Windows application—but it requires a lot more user intervention to get something out of it.

The game will consist of a main menu, which includes a settings menu, and a start/exit button. When starting the game, the user can fight 50 battles in an increasing difficulty curve. Once the user beats the last battle, they win. Each battle is turn-based, so the player takes a turn once the enemy finishes. As the player plays the game, they may also save/load their data to continue playing later.

|  |
| --- |
| History and Signoff Sheet |

**Change Record**

|  |  |  |
| --- | --- | --- |
| **Date** | **Author** | **Revision Notes** |
| 9/25/2022 | Justin Gewecke | Initial draft for review/discussion |
| 10/2/2022 | Justin Gewecke | Initial project requirements outlined |
|  |  |  |

|  |
| --- |
| **Overall Instructor Feedback/Comments** |

|  |
| --- |
| **Overall Instructor Feedback/Comments** |

**Integrated Instructor Feedback into Project Documentation**

Yes  No

**TABLE OF CONTENTS**

Functional Requirements 4

Non-Functional Requirements 5

Technical Requirements 6

Logical System Design 7

User Interface Design 8

Reports Design 9

Functional Requirements

**Use Cases**

|  |  |  |
| --- | --- | --- |
| **Use Case or User Story** | **Approval Date** | **Justification** |
| **As a user, I want to be able to adjust the volume of the game.** |  | **Make the game more comfortable for the user's ears** |
| **As a user, I want to be able to mute the game.** |  | **This will allow the user to listen to music or podcasts during the gaming session.** |
| **As a user, I want to easily see the progress of my game.** |  | **Make the situation of the game obvious. The user can know how close they are to finishing the game.** |
| **As a user, I want to be able to upgrade my units.** |  | **This will provide the user with a sense of progress.** |
| **As a user, I want to be able to save/load my game’s progress.** |  | **This will allow the user to continue their session at different times even when closing the game.** |
| **As a user, I want to see the enemy’s info during battle.** |  | **This will inform the user of the situation.** |
| **As a user, I want to see my info during battle.** |  | **This will inform the user of the situation.** |
| **As a user, I want to see my actions.** |  | **This will show what the user can do during a move.** |
| **As a user, I want my enemy to do actions by themselves.** |  | **This will allow the user to play the game without a second person.** |
| **As a user, I want to experience defeat when my team is lacking.** |  | **This will show the user that they need to progress further.** |
| **As a user, I want to experience victory.** |  | **This will allow the user to feel good about progressing.** |

***NOTE: If necessary, you may add subsections to those listed in order to match the requirements in the assignment description. Do not remove any top-level sections (those that are listed in the Table of Contents). As required by your project, you may add additional top-level sections (please update the Table of Contents).***

Non-Functional Requirements

**Use Cases**

|  |  |  |
| --- | --- | --- |
| **Use Case or User Story** | **Approval Date** | **Justification** |
| **As a user, I want my experience to not feel slow/sluggish when interacting with the application.** |  | **This will ensure a smooth game experience.** |
| **As a user, I want to play this game on my PC.** |  | **This limits the platform in which this game can be played.** |
| **As a user, I should quickly be able to load/save my game progress.** |  | **This will make loading/saving games a fast and easy experience.** |
| **As a user, I want the game to startup as quickly as possible.** |  | **This will provide the user more time to play the game.** |
| **As a user, I want an easy way to install the game.** |  | **This will make it easy for the user to install and play the game.** |

Technical Requirements

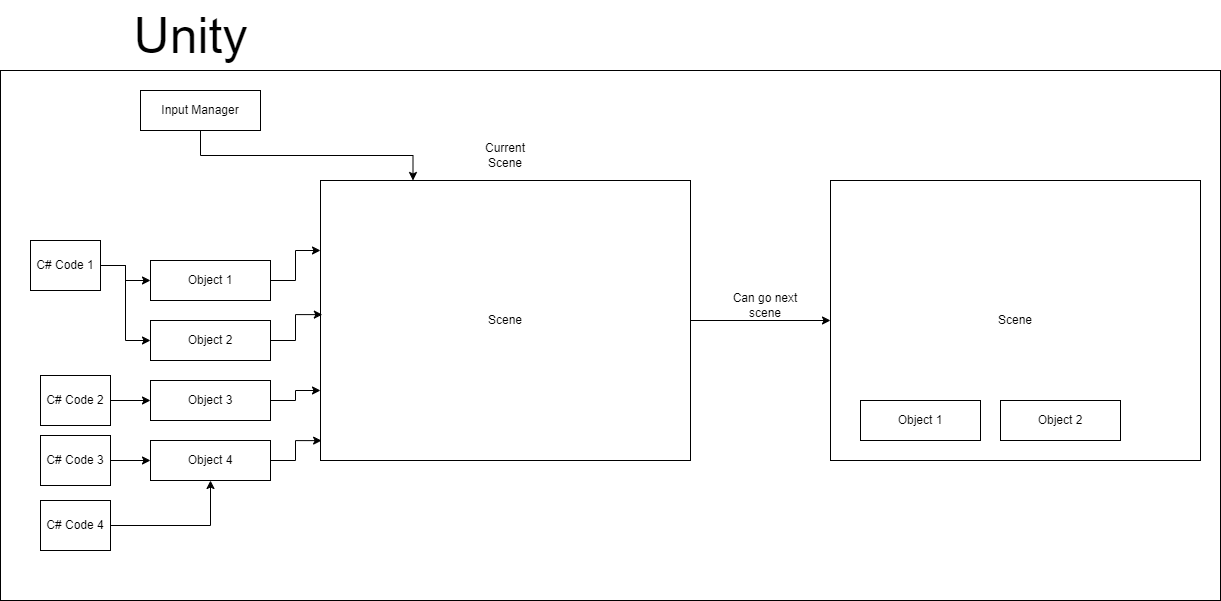
**Use Cases**

The first tool of choice is the Unity Game Engine. This tool handles all the visual aspects of the application. This allows me to focus on the data, and programming. The engine utilizes C#, so all the game logic would be in this language. No other tools/frameworks would be used with tool; if needed, C# libraries/frameworks could be used if found to be needed.

As an alternative to Unity, there is the C# MonoGame framework which is an open-source revival of the XNA framework which was made by Microsoft. Like Unity, this framework handles the backend rendering, which is a massive time saver. Compared to Unity, this framework does not handle some GUI and user interaction nuances, which would require more effort towards this area of the application. Using MonoGame, I would likely not need any our libraries. Though, I am inexperienced with this framework so this may change.

|  |  |  |
| --- | --- | --- |
| **Technology or Tool** | **Approval Date** | **Justification** |
| **Unity Game Engine** |  | **This will allow me to work a lot faster since I won’t have to focus on the game rendering and GUI. Instead, I can focus on the game data and programming logic.** |
| **MonoGame** |  | **This will allow me to create my game with more flexibility since I will be controlling more elements of the User/user interaction. This will add a lot of development time needed for this application, however.** |

Logical System Design



To summarize the Unity architecture, it compiles everything the user sees into “Scenes.” For instance, the settings menu, battle menu, and main screen can be separated into different scenes. The scene for the settings menu would likely contain an object that allows the user to mute or change the volume of the game. There can be multiple objects inside of a scene. To provide functionality and other changes to the scene, C# can be added to objects. Multiple C# scripts can be added to an object. As a result, you can have multiple objects and C# scripts inside of a scene. Using C#, it is possible to change to another “scene” which can have its own game objects and logic.

User Interface Design

Diagram

Description automatically generated

Reports Design

This system will not provide any reports. This game is intended to be an interactive experience that reacts to the user’s input in real time. The game does not process any information to form a report for the user to digest. The application can generate a save file for the user, but this is not intended to be read by the user. If this game had online capabilities, it could track game time and the user’s interaction with the application. However, this game is fully offline, meaning there is no reason for any reporting.