Computer Games Development SE607
Software Functional Specification
Year IV

[Seán Whelan]
[C00250016]

[26-04-2023]

DECLARATION

Work submitted for assessment which does not include this declaration will not be assessed.

- I declare that all material in this submission e.g. thesis/essay/project/assignment is entirely my/our own work except where duly acknowledged.
- I have cited the sources of all quotations, paraphrases, summaries of information, tables, diagrams or other material; including software and other electronic media in which intellectual property rights may reside.
- I have provided a complete bibliography of all works and sources used in the preparation of this submission.
- I understand that failure to comply with the Institute's regulations governing plagiarism constitutes a serious offence.

Student Name: (Printed) SEÁN WHELAN

Student Number(s): C00250016

Signature(s): Seán Whelan

Date: 26 April 2023

Contents

Acknowledgements

I would like to thank the following people who assisted in completing this project.

My project supervisor Martin Harrigan.

All of the lecturers that I have had throughout my 4 years at SETU, they have helped develop my programming and game development skills allowing me to finish this project and many more.

Introduction:

SlayerMaker2D is a no code game editor primarily targeting users who have no programming or game development experience that would like to create 2d top-down shootemup games. The scope and objective of "SlayerMaker2D" would be find a middle ground between allowing users to create a diverse line-up of games while keeping the project lacking in bloat of options, to allow it to be easier to use.

Functional requirements

1. SlayerMaker should allow the user to choose from various game options, including name, size, background and between 3 game type templates.
2. SlayerMaker should allow the user to create game objects and place them anywhere within the playable area. These game objects should have all their textures, functionality etc already attached.
3. The game objects collectively along with game options chosen to make up a game. Using these the player should be able to create a level from scratch with ease.

4. The game the player creates should be testable with the click of a button. This should allow the user to test the game and make note of any changes they feel necessary. Then with the click of a button they should be able to make those changes.
5. When the user is happy with their game creation, with the click of a button the user should be able to save the game objects and game options to a file.
6. When the user returns to the main screen the user should be able to find the final version of their game. They should be able to, with the click of a button be able to, with the click of a button play the game.
7. SlayerMaker should provide a user-friendly interface for doing all of this to ensure the process of creating, testing, editing, and playing a game is intuitive.
8. With the click of a button the user should be able to save the game data of a game they think is good to a database.

Performance requirements

1. SlayerMaker should allow the user to navigate through the different screens of the game with minimal to no delay.
2. The user should be able to add as many game objects to their game as they see fit without minimal or no delay.
3. The user should be able to transition from editing their new game to testing it to back to editing with minimal or no delay. This should even be the case if there is a lot of game objects.
4. The editor should use system resources such as CPU, memory, and graphics card efficiently, without causing the system to slow down or crash.
5. The games created using the editor should perform well. The collisions should register, the user should not be dying because of delay/ lag despite their being a lot of game objects.
6. The editor should work with every PC that is compatible.
7.

User interface

1. SlayerMaker should have a clean, intuitive, and visually appealing design that makes it easy for users to navigate and use.
2. SlayerMaker should have a menu, toolbar and choicebar system that allows you to quickly and easily build the game the user wants without being overwhelmed by an overbearing number of tools and choices.
3. The User Interface should make it very easy for the user to click and place any objects, it should be clear how this works.
4. When the user advances through the creation process the choicebar should be contextual to where they are
5. The user interface should allow the user to scroll the camera in all 4 cardinal directions and allow zooming.
6. A rubber should allow the player to undo any object that they have added that they feel needs to be removed.

7. The editor should adapt to various sizes of screen and resolution and still be usable, within a given, fair margin.

Assumptions and constraints

Assumptions:

1. The user has little to no Game Development, coding, or any other experience relevant to game editors.
2. The user has a Personal Computer which is running a recent version of windows.
3. SlayerMaker has access on this PC to necessary resources such as memory and storage to create and save their games as well as CPU and GPU to run the application and view the window.
4. I assume that the users of SlayerMaker are proficient in English, as this is the language used in the editor, mainly for buttons and menu navigation.
5. I assume that the development of SlayerMaker will take a certain amount of time and that the resources allocated will be sufficient to complete the project.

Constraints:

1. I may have technical constraints, such as the hardware limitations of the target platform, which may limit the features and functionality of your no code game editor.
2. I have a limited budget for the development of SlayerMaker, which may limit the resources available for development, testing, and support.
3. I have a fixed timeline for the development of your SlayerMaker, which may limit the amount of time available for development, testing, and documentation.
4. SlayerMaker may be dependent on external libraries or frameworks, which may constrain the development and functionality of your software.
5. SlayerMaker may be subject to legal and regulatory requirements, which may constrain the development and distribution of your software.
6.

Interfaces

1. SlayerMaker will require a GUI to enable users to interact with the software. The GUI should be intuitive, easy to navigate, and provide all necessary functionality for game creation, editing and playing.
2. SlayerMaker will need to support user input, such as keyboard and mouse input, to enable users to interact with the software and create their games.
3. SlayerMaker will need to interface with SFML to provide game development functionality to the user.
4. SlayerMaker will need to interface with a database to upload game data to and download game data from a database.

Data requirements

1. Game object data: Your no code game editor will need to store data about game objects, such as their position, size, sprite, behavior, and any associated scripts or events.

2. SlayerMaker will need to store game data about each game created. This includes game options like background and game type as well as the game objects that were used to create each game.
3. SlayerMaker will need to store data about game assets, including as images, sounds, music, game data files and fonts.

Error handling

1. SlayerMaker should provide clear and concise error messages to the user when an error occurs, indicating the source of the error and any relevant details.
2. SlayerMaker should be designed to recover gracefully from errors, ensuring that the user's work is not lost, and that the software remains stable and functional.
3. SlayerMaker should validate user input to prevent errors from occurring in the first place. This can include checking for invalid data types, out-of-range values, and other potential input errors.
4. SlayerMaker should handle exceptions, such as memory allocation errors or other runtime errors, in a manner that ensures the software remains stable and functional.

Feature List

Just a simple list of features still to be implemented by 17/01/2023.

Wall Placement and wall Object rewrite

Player, simple movement, animated sprite

Test level function

Scene Navigation

sf::View control. Zooming / Panning etc

- Saving placements to a text document
- Level Creation / scene to choose level
- Load placements from a text document

One enemy which seeks player

Enemy attack

Multiple enemies

Animation class, modular can animate any sprite sheet with given parameters.

Creating Executable with necessary assets for specific level

Prebuilt or asset manager for textures.

Prebuilt or asset manager for Sound, effects etc.

Customisation, 3 wall types, 3 ground types, 3 player types, 3 enemy types.