SE 3XA3: Software Requirements Specification Mario Level X

Group 210, Group 210
Ahmad Gharib and ghariba
Edward Liu and liuz150
Connor Czarnuch and czarnucc

February 10, 2020

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	**Version**	**Notes**
2020-02-09	1.0	SRS Version 0

This document describes the requirements for ”Mario Level X” The template for the Software Requirements Specification (SRS) is a subset of the Volere template. If you make further modifications to the template, you should explicity state what modifications were made.

1 Project Drivers

1.1 The Purpose of the Project

The purpose of this project is to modify the original game, ”Mario level 1”, to allow players to design their own levels and play it. The original code does not separate game functionality from level layout, so items such as checkpoints, monsters, and obstacles are hard coded and cannot be changed. We would like to further modularize the code such that the level layout data becomes a parameter, which can be imported from an external source, such as a file or database. By separating game functionality and level layout, we can give players control of the level layout aspect of the game to create their own customized Mario level.

1.2 The Stakeholders

1.2.1 The Client

- Invigilators of 3XA3

1.2.2 The Customers

- Anyone that will play Mario level x.

1.2.3 Other Stakeholders

- Mario fans
- Original developer of Mario level 1

1.3 Mandated Constraints

Project is constrained by time since it must be fully completed by April 6. The game is also constrained by the device it is running on. The device requirements required to run the new game should be similar to that of the original game.

1.4 Naming Conventions and Terminology

- PYGAME: Python library used for game development.
- GAME ITEM: Refers to items in Mario. (blocks, mushrooms, monsters, pipes, etc)
- LEVEL DATA: A JSON file containing custom level data.

1.5 Relevant Facts and Assumptions

It is assumed that the user has basic knowledge computer games, specifically Mario games. Users should understand the basic objective in Mario in order to play the game. Users should also have knowledge of what each "game item" does in order to design levels effectively.

2 Functional Requirements

2.1 The Scope of the Work and the Product

Deliverables

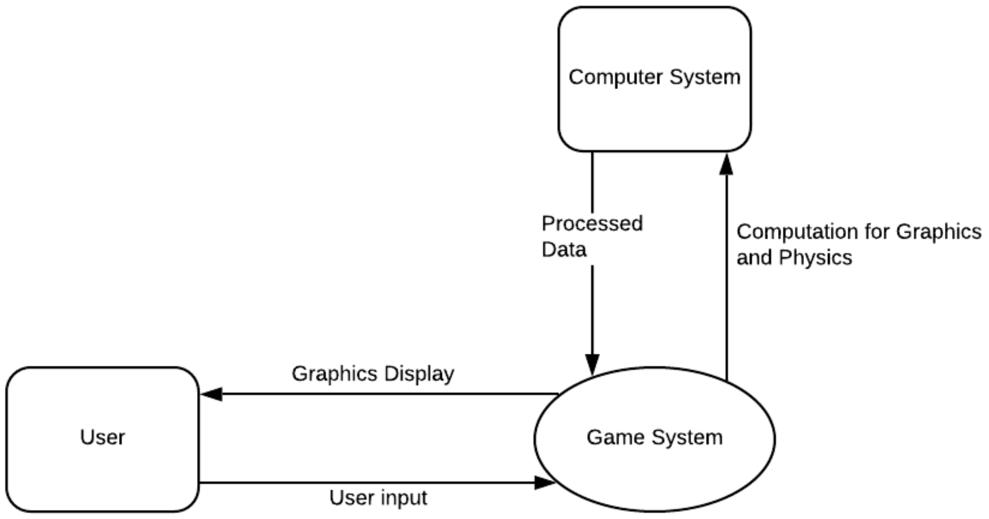
- Final software product
- All required documentation

Deadlines

- Problem Statement - January 24
- Development Plan - January 31
- Requirements Document (Revision 0) - February 7
- Proof of Concept - February 14
- Test Plan (Revision 0) - February 28

- Design and Document (Revision 0) - February 28
- Demonstration (Revision 0) - March 16
- Final Demonstration (Revision 1) - March 30
- Final Documentation (Revision 1) - April 6

2.1.1 The Context of the Work



2.1.2 Work Partitioning

Work will be split into small tasks that developers can choose to work on. Each task will be short and will be worked on in separate branches in the git repository. After the task work in the branch is done, the branch is merged to master. All tasks will be worked on until the project is complete. This will be following the scrum methodology.

2.1.3 Individual Product Use Cases

Use Case: User wants to creates new level.
Trigger: User selects create new level from menu.
Pre-condition: User must be on menu screen.

Outcomes: Game displays create level UI.

Use Case: User wants to exports level data.
Trigger: User clicks export level button for the level.
Pre-condition: User must have a level selected.
Outcomes: A file will be created and saved in a predefined location for the user to retrieve.

Use Case: User wants to saves custom level.
Trigger: User clicks save level in level editor.
Pre-condition: User must be editing a level.
Outcomes: A file with a level data will be stored in a predefined folder that will contain all the users levels.

Use Case: User wants to play a level.
Trigger: User clicks on level they want to play.
Pre-condition: User must be in the select level screen.
Outcomes: Game will render the user selected level with its corresponding level data file.

2.2 Functional Requirements

- Requirement Number: FR1
  Users must have a GUI to design levels.
  Rationale: The user should not be exposed to the raw level layout data. The user does not need to know how the level data is stored.

- Requirement Number FR2
  Users can play through their custom levels.
  Rationale: The premise of the game is to allow users to both create levels and play them.

- Requirement Number FR3
  Created level data must be saved on non-volatile memory.
  Rationale: User's level should be preserved regardless if game or device is shut down.

- Requirement Number FR4
  Users can create multiple levels

Rationale: Improve experience by allowing users to create multiple levels.

- Requirement Number FR5
  Users can name levels
  Rationale: Allow users to identify each level.

- Requirement Number FR6
  Levels should be both shareable
  Rationale: Users should be able to both export and import levels. Users can play levels that have been designed by other players.

3 Non-functional Requirements

3.1 Look and Feel Requirements

- Requirement NF1: The initial menu will have GUI buttons for selecting single and two player modes, along with a button for selecting edit mode and custom levels.
  Rationale: This will be used to start the game in single player, or edit mode.
  Originator: Connor Czarnuch
  Fit Criterion: The single player button will change the program to the level 1 state. The edit button will change the program to the edit state. The choose custom level button will allow the user to select a custom level map.

- Requirement NF2: The design of the original game will be preserved in all aspects of the GUI and gameplay.
  Rationale: Since the game is a remake of the original game, Super Mario Bros., the design language and design of certain aspects should match the original as close as possible.
  Originator: Connor Czarnuch
  Fit Criterion: If the game resembles the original with some added features.

- Requirement NF3: Edit mode will allow the player to move using the same mechanism as playing the level 1 stage. Each click will change the identity of the block at that point.
  Rationale: The edit mode needs to be familiar and allow the user to use it without a tutorial. Moving around the same way and registering mouse clicks allows the player to learn the interface more easily.
  Originator: Connor Czarnuch
  Fit Criterion: The game should allow the player to move and change blocks in edit mode.

3.2 Usability and Humanity Requirements

Requirement NF4: Any user should be able to easily select any of the menu options and play the game without the need of a tutorial. Interface should be user centered. Rationale: Without a way of introducing a tutorial, the game should be easily playable by players of all ages.
Originator: Connor Czarnuch
Fit Criterion: A user of basic skill should be able to play the game and edit a level.

Requirement NF5: The program should be easily playable via an executable on windows operating systems. Playing on other operating systems require use of the python interpreter via a command line or IDE.
Rationale: Any basic user should be able to easily install and play the game.
Originator: Connor Czarnuch
Fit Criterion: The game is playable via the command line and on windows systems via the executable.

Requirement NF6: All of the menu items should be displayed in Canadian English.
Rationale: The program is localized for the Canadian market so the spelling and language should match.
Originator: Connor Czarnuch
Fit Criterion: All grammar and spelling is correct.

3.3 Performance Requirements

Requirement NF7: The application should be responsive when a user requests an action from the GUI. Rationale: To avoid double inputs and user error, the GUI should be responsive to show the user that the requested action is being processed.
Originator: Connor Czarnuch
Fit Criterion: All user actions with reference to the GUI should activate within 50 milliseconds.

Requirement NF8: The application should be responsive when a user is playing the game and triggering movement keys. The movement of the character on screen should match what is being inputted by the user.
Rationale: The user needs to not feel frustrated when playing the game, and by having the least input lag will reduce the frustration the user feels.
Originator: Connor Czarnuch
Fit Criterion: All movement actions should show up on screen within 15 milliseconds of a user pressing a key.

Requirement NF9: The application should not quit unexpectedly.
Rationale: The user needs to know if a fatal error caused the game to crash so that the developers can improve the program to avoid and similar problems.
Originator: Connor Czarnuch
Fit Criterion: The application does not quit unexpectedly, instead offers an error message before quitting.

Requirement NF10: The application should handle all user inputs that affect the game.
Rationale: If a user input is unhandled and affects the gameplay, then there is no documentation providing information on what happened.
Originator: Connor Czarnuch
Fit Criterion: Any unhandled user inputs should not affect the game.

3.4 Operational and Environmental Requirements

This program does not have any environmental or operational impact.

3.5 Maintainability and Support Requirements

Requirement NF11: The source code to this program should be visible to the public.
Rationale: Open source code allows contributors to improve the program and to more easily maintain the program.
Originator: Connor Czarnuch
Fit Criterion: Source code is available in a public repository.

Requirement NF12: The program will be programmed in a cross platform library using packages that are available on all platforms and easily downloaded or included in an executable.
Rationale: This allows the developers to develop the same codebase for any platform.
Originator: Connor Czarnuch
Fit Criterion: The program is written in Python 3.

3.6 Security Requirements

Requirement NF13: The program will not transmit any user data.
Rationale: For the privacy of the user, the program will not store or share any user data.
Originator: Connor Czarnuch
Fit Criterion: No user data is stored or transmitted.

3.7 Cultural Requirements

Requirement NF14: The program should not contain any imagery that could offend any culture.
Rationale: To enhance user experience, the program should avoid using any imagery that could cause a user to feel dissatisfied with the program.
Originator: Connor Czarnuch
Fit Criterion: No culturally sensitive imagery is shown.

3.8 Legal Requirements

Requirement NF15: The software should be distributed under the MIT permissive licence.

Rationale: To keep our program open the MIT licence is used to share software with a limited restriction.
Originator: Connor Czarnuch
Fit Criterion: Distributed with the MIT licence.

3.9 Health and Safety Requirements

Requirement NF16: The program should not contain any effects on screen that could induce seizures for any user or any epileptic user.
Rationale: To avoid any health problems with the users, the program should not contain any effects that could cause damage.
Originator: Connor Czarnuch
Fit Criterion: There are no bright flashing effects.

This section is not in the original Volere template, but health and safety are issues that should be considered for every engineering project.

4 Project Issues

4.1 Open Issues

Modularized Map Creation

- An existing issue that we intend to solve via our modularized map creation is to remove the hard-coded nature of the existing project. Currently, the map creation is not modular and all the entities and collisions are hard-coded to the exact pixel of where they should spawn, meaning a complete overhaul of this implementation is required to allow the user to modify the map.

Understanding the code-base of the existing project

- While the original project does contain some internal documentation in the form of comments, it is not extensively documented. Therefore it is important to analyse the existing project in-depth in order to understand what design decisions were made during the programming process, and how we can improve on these implementations.

4.2 Off-the-Shelf Solutions

Super Mario Maker is a product that very closely resembles the end-goal of our project. Our aim is to provide the user the ability to create any level with their creativity being the only limiting factor as to what level they can create. In Super Mario Maker, a similar premise exists where the user can create a level and share it with other players to play. The difference between our game and Super Mario Maker is our game is local, and allows for real-time level modification while you're still playing the level you wish to change.

4.3 New Problems

Allowing players to modify the level in real time, even when they are still playing poses a few potential challenges. One of these potential issues is that the game engine being used, PyGame, may have difficulties creating and deleting collisions during the game, and may cause unexpected engine bugs. Another issue would be how the game will handle new edge-cases that didn't exist in the original game, such as what if the user puts a block on top of another block. This edge-case can have varying results depending on implementation, so it is important for us as developers to consider all of these and ensure that our new features are bug-free.

4.4 Tasks

The tasks are outlined when the project was introduced in class as a list of deliverables. These include: Problem Statement, Development Plan, Requirements Document (Revision 0), Proof of Concept, Test Plan (Revision 0), Design and Document (Revision 0), Demonstration (Revision 0), Final Demonstration (Revision 1), and the Final Documentation (Revision 1). This outlines all the required tasks to be completed by the time of project submission.

4.5 Migration to the New Product

Features will be added over time, starting with the highest priority to the lowest priority. After each feature is completely implemented, test cases are to be run to ensure that it works as intended before the next feature is added.

The product will be tested on Windows, Linux, and MacOS to ensure it is working as intended on all platforms.

4.6 Risks

It is still uncertain as to whether or not the scope of our project is too large to be implemented in a single semester. It is also unclear as to if we can test for every single edge-case in the allotted time as games tend to have a tendency of containing many bugs even after extensive testing.

4.7 Costs

None.

4.8 User Documentation and Training

The game will come with a document with instructions on all the controls for the user.

4.9 Waiting Room

Improve on player controls and GUI.

4.10 Ideas for Solutions

Implement thorough documentation and planned improvements.