SE 3XA3: Software Requirements Specification
Sketchy Super Mario Bros

Team 1, Wario's Miners
Jason Nam and namy2
Kristine Uchendu and uchenduc
Rylan Sykes and sykesr

February 11, 2022

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	Version	Notes
Feb. 11th 2022	1.0	Kristine's initial notes have been added.
Feb. 11th 2022	1.1	The group has come together to add all of their parts.

1 Project Drivers

1.1 The Purpose of the Project

Initially released in 1985, Super Mario Brothers is a platform game that was originally developed for Nintendo arcades. The game is often named as one of the greatest video games of all time and has sold over 50 million copies worldwide. In the game, the player controls Mario (or potentially his brother Luigi, if in multiplayer mode) on his journey through the Mushroom Kingdom to save Princess Toadstool from Bowser. Mario and/or Luigi have to overcome hazards, such as living mushrooms (called Goombas) and pits, with the help of power-up items like the Super Mushroom, the Fire Flower and the Starman. There is currently no official desktop version of the original 16-bit game, but there is an open-source desktop game emulating it. The purpose of the Sketchy Super Mario Brothers project is as follows: we aspire to improve upon the existing 16-bit Super Mario Brothers for Desktop Game, to create a user-friendly and enjoyable game experience for players.

1.2 The Stakeholders

1.2.1 The Client

The clients for this project are the instructor, Dr. Bokhari, and the teaching assistants (Stephanie Koehl, Abdul Rab Mohammad, Oluwaseun Owojaiyo, and Veerash Palanichamy). All information about the project is received from them. This information includes deadlines for key milestones, best practices for documentation, feedback, and more.

1.2.2 The Customers

The targeted customers for the improved desktop version of Super Mario Bros., are fans of any of the many variations of the game, as well as people who are interested in learning how to play the game. The customers are also people that would be interested in downloading the game on their desktop and playing the game on their computer.

1.2.3 Other Stakeholders

Because the basis of this project is improving upon a medium-sized open source project, another group of stakeholders for this project are developers who decide to contribute to this project in the future. For future developers the documentation needs to be detailed, and the code needs to be well-structured. Another potential stakeholder is Nintendo, the gaming company that created the original game. We do not own the rights to characters or the idea of the game so monetizing it is not an option, and we must avoid breaking any and all copyright laws.

1.3 Mandated Constraints

- The project is due on April 12th.
- The TAs need to be able to run the project on their computer.
- Our project must be based on the chose open open-source project (mario-game).
- There is no budget for this project so any software that needs to be bought cannot be considered.

1.4 Naming Conventions and Terminology

- SSMB - Sketchy Super Mario Bros
- GUI - Graphical User Interface
- UI - User Interface
- UX - User Experience
- FR - Functional Requirement
- NF - Non-functional Requirement

- UC - Use Case
- BE - Business Event
- TA - Teaching assistant

1.5 Relevant Facts and Assumptions

1.5.1 Facts

- The original project has approximately 1600 lines of code
- The original open-source projects has not implemented certain features of the original game.

1.5.2 Assumptions

- The user has PC
- The user's PC can run Java
- The user has Gradle installed on their computer

2 Functional Requirements

2.1 The Scope of the Work and the Product

The scope of the project is to design and implement a clone of the 16-bit Mario game originally created for the SNES game console. This is achieved through the use of various modern libraries which enable development on multiple platforms concurrently. The project will be developed through a series of deliverables spanning 4 months which include: design documents, requirements specification, testing plan, and proof of concepts. The purpose of the project will be to allow gamers to play the original game on modern hardware despite not having the original game console.

2.1.1 The Context of the Work

Refer to the context diagram below.

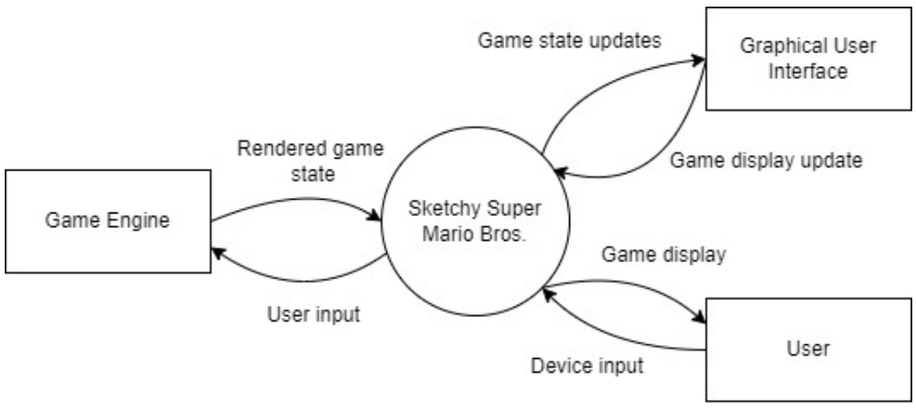

Figure 1: Context Diagram

2.1.2 Work Partitioning

Business Event	Event Name	Input and Output	Summary
BE1	User enters game	Game display (out)	Game will display the initial state (common amongst all starts)
BE2	User exits game	Game display (out)	Game will exit, game state or display will not persist
BE3	User inputs movement action	User input (in) Game display (out)	The GUI will display the corresponding action on the users device
BE4	Mario contacts enemy	Game state (in) Game display (out)	An attack from Mario (if done correctly) will kill an enemy
BE5	Enemy contacts Mario	Game state (in) Game display (out)	Any contact from Mario (to enemy) that is non-lethal will result in Mario losing health points
BE6	Mario takes damage	Game state (in) Game display (out)	Mario will lose health accordingly and will be visibly displayed to the user
BE7	Mario is eliminated	Game state (in) Game display (out)	Mario's health points will reach 0 and game will display death messages
BE8	Enemy is eliminated	Game state (in) Game display (out)	Enemy will be cleared from game GUI and can no longer attack Mario
BE9	Mario contacts powerup	Game state (in) Game display (out)	Mario will visually and functionally react to the power-up obtained
BE10	Mario falls off of platform	Game display (out)	Game will display death message and restart
BE11	Mario contacts brick from below	Game state (in) Game display (out)	Game will reward player based on Mario's state (power-up vs. no power-up)
BE12	Mario contacts finish pole	Game display (out)	Game is complete and game displays win screen, then restarts

2.1.3 Individual Product Use Cases

The individual product use cases are equivalent to the events defined in the work partitioning.

Use Case	Use Case Name	Summary
UC1	User enters game	Game will display the initial state (common amongst all starts)
UC2	User exits game	Game will exit, game state or display will not persist
UC3	User inputs movement action	The GUI will display the corresponding action on the users device
UC4	Mario contacts enemy	An attack from Mario (if done correctly) will kill an enemy
UC5	Enemy contacts Mario	Any contact from Mario (to enemy) that is non-lethal will result in Mario losing health points
UC6	Mario takes damage	Mario will lose health accordingly and will be visibly displayed to the user
UC7	Mario is eliminated	Mario's health points will reach 0 and game will display death messages
UC8	Enemy is eliminated	Enemy will be cleared from game GUI and can no longer attack Mario
UC9	Mario contacts powerup	Mario will visually and functionally react to the power-up obtained
UC10	Mario falls off of platform	Game will display death message and restart

UC11	Mario contacts brick from below	Game will reward player based on Mario's state (power-up vs. no power-up)
UC12	Mario contacts finish pole	Game is complete and game displays win screen, then restarts

2.2 Functional Requirements

FR	**Requirement**	**Rationale**	**Fit Criterion**	**Use Case**	**Created**
FR1	The user must be able to start a new game	A new game must be started in order to play	A new game is able to start	UC1	2022-02-11
FR2	The user must be able to exit the game	The game must be able to exit for users to stop playing	The game is able to terminate	UC2	2022-02-11
FR3	The user must be able to move Mario to the right and left	Mario must move left and right as the game is a side scroller	Mario is able to move right and left depending on user input	UC3	2022-02-11
FR4	The user must be able to move Mario up (jump)	There are platforms on higher levels that require jumping	Mario is able to jump	UC3	2022-02-11
FR5	Mario, enemies, and power-ups in the game must be subject to gravity	Said objects should be able to jump and fall down, as well as descend from platforms	Said objects descend when mid-air	UC3	2022-02-11
FR6	Mario must be able to attack enemies	Mario can eliminate enemies for path-clearing as well as rewards	Mario can land on enemies to eliminate them	UC4, UC8	2022-02-11
FR7	Mario must be able to take damage from the environment and enemies	User must face challenges when clearing the level	Mario is able to lose health points	UC5, UC6	2022-02-11
FR8	Mario must be able to be eliminated	Game needs a way for the user to fail the level	Mario reaches 0 health points	UC7	2022-02-11
FR9	User must be visibly notified when Mario loses health points	User should be aware that they are losing health points	Animation/re-color displays when Mario loses health points	UC6	2022-02-11
FR10	Game must display a 'Game Over' screen when Mario is eliminated	User should be aware that they have lost	Message is displayed and game is restarted	UC7	2022-02-11
FR11	User must be visibly notified when enemy loses health points	User should be aware that enemy has lost health points	Animation/re-color displays on enemy	UC4, UC8	2022-02-11
FR12	Enemies must be removed from display when eliminated	Indicates to user that enemy is eliminated	Enemy no longer appears on GUI and is unable to interact with Mario	UC8	2022-02-11

FR13	Mario must be able to consume power-ups	User must have a way to clear level easier	Mario visibly consumes power-up and respective ability is visibly and functionally applied to Mario	UC9	2022-02-11
FR14	Mario must be able to break bricks	Allows user to obtain coins	Coins will appear from bricks when Mario jumps and makes contact from brick from below until broken	UC11	2022-02-11
FR15	User must be able to finish level	Allows user to complete game	Win screen must be displayed when Mario makes contact with Finish Pole	UC12	2022-02-11

3 Non-functional Requirements

3.1 Look and Feel Requirements

NF1 Description: The system shall display intuitive GUI.

Rationale: To allow users to get familiar with the interface quickly.

Originator: Jason Nam

Fit criterion: The users are able to select stages and set preferences within the options window within 1 minute of system use.

Priority: High

History: Created February 11, 2022

NF2 Description: The system shall look similar to the original Super Mario Bros 16 bit game.

Rationale: To ensure that users who play this game will identify it as or as similar to the original Super Mario Bros 16 bit game.

Originator: Jason Nam

Fit criterion: By first playing this system, 80 percent of users will immediately identify the system as a Super Mario Bros game.

Priority: Medium

History: Created February 11, 2022

NF3 Description: The system shall appear nostalgic.

Rationale: To ensure users who have enjoyed the original Super Mario Bros 16 bit will thoroughly enjoy this system.

Originator: Jason Nam

Fit criterion: The users who have memories of Super Mario Bros will stay on the system for more than 5 minutes.

Priority: Medium

History: Created February 11, 2022

3.2 Usability and Humanity Requirements

NF4 Description: The system shall be used by users with no knowledge about the system mechanics.

Rationale: To ensure that users who play this game for the first time will not struggle.

Originator: Jason Nam

Fit criterion: 50 percent of first time users will be able to pass the first stage.

Priority: Medium

History: Created February 11, 2022

NF5 Description: The system shall be used by users with a simple understanding of the English language.

Rationale: To ensure that users will not require vast english vocabulary to understand system mechanics.

Originator: Jason Nam

Fit criterion: 30 percent of users will have a different native language other than english.

Priority: Low

History: Created February 11, 2022

NF6 Description: The system shall use symbols and terms that are easy to understand.

Rationale: To avoid forcing users to learn new symbols and terms that can hinder their satisfaction with the system.

Originator: Jason Nam

Fit criterion: 90 percent of users will subconsciously identify symbols and terms used in the system.

Priority: Medium

History: Created February 11, 2022

NF7 Description: The system shall be made usable by partially sighted users.

Rationale: To ensure users with partially disabled sights are able to enjoy the system without serious impairment.

Originator: Jason Nam

Fit criterion: Less than 20 percent of users with partially sighted disabilities will express severe discomfort with the system.

Priority: Medium

History: Created February 11, 2022

3.3 Performance Requirements

NF8 Description: The system shall be reliable with little to no failures.

Rationale: To ensure users can run the system without system failure.

Originator: Jason Nam

Fit criterion: The system will fail no less than 3 times in a day.

Priority: High

History: Created February 11, 2022

NF9 Description: The system shall be available.

Rationale: To ensure users can install the system.

Originator: Jason Nam

Fit criterion: The system will not be available to install no less than 5 times a year.

Priority: Low

History: Created February 11, 2022

NF10 Description: The system shall be responsive to user input and perform requested action if required criteria are met.

Rationale: To ensure user satisfaction is not disturbed by latency and wrong responses.

Originator: Jason Nam

Fit criterion: After user input, the system will respond within TIME_RESPONSE with the requested action and response.

Priority: High

History: Created February 11, 2022

3.4 Operational and Environmental Requirements

NF11 Description: The system shall be portable on Linux systems.

Rationale: To allow users on Linux systems to run this system.

Originator: Jason Nam

Fit criterion: The system is Linux compatible.

Priority: Low

History: Created February 11, 2022

NF12 Description: The system shall be portable on Windows systems.

Rationale: To allow users on Windows systems to run this system.

Originator: Jason Nam

Fit criterion: The system is Windows system compatible.

Priority: High

History: Created February 11, 2022

NF13 Description: The system shall be programmed in a language that will be operable on future operating platforms.

Rationale: To ensure the system will be usable after future updates.

Originator: Jason Nam

Fit criterion: The system is written in a programming language that is cross-platform.

Priority: High

History: Created February 11, 2022

3.5 Maintainability and Support Requirements

NF14 Description: The system shall be easy to maintain.

Rationale: To ensure the maintainability of the system will be simple and costless.

Originator: Jason Nam

Fit criterion: The system takes less than 100 lines of code modification to maintain.

Priority: Medium

History: Created February 11, 2022

3.6 Security Requirements

NF15 Description: The system shall not collect and upload user data without authorization.

Rationale: To protect privacy of users and their personal information.

Originator: Jason Nam

Fit criterion: The system does not have any function that can collect and upload user data.

Priority: Low

History: Created February 11, 2022

3.7 Cultural Requirements

NF16 Description: The system shall not contain any imagery or text that could be regarded as hostile or offensive towards users and their cultures.

Rationale: Users will be unsatisfied due to the hostile or offensive context the game produces.

Originator: Jason Nam

Fit criterion: System does not contain any offensive cultural imagery or text.

Priority: Medium

History: Created February 11, 2022

3.8 Health and Safety Requirements

NF17 Description: The system shall avoid causing epileptic seizures to users, and avoid impairing user visions.

Rationale: To ensure the system does not cause epileptic seizures to users.

Originator: Jason Nam

Fit criterion: Less than 1 percent of users will express epileptic symptoms with the system.

Priority: High

History: Created February 11, 2022

4 Project Issues

4.1 Open Issues

A predominant issue is that the version of the libraries and package used in the original game are old and not compatible with newer versions of other software required to run the project. This prevents running the game and workarounds are still being looked into.

4.2 Off-the-Shelf Solutions

There are other open source 16-bit Mario clones existing on GitHub (and probably elsewhere).

4.3 New Problems

There are no new problems that are expected to arise.

4.4 Tasks

Refer to the Gantt chart below.

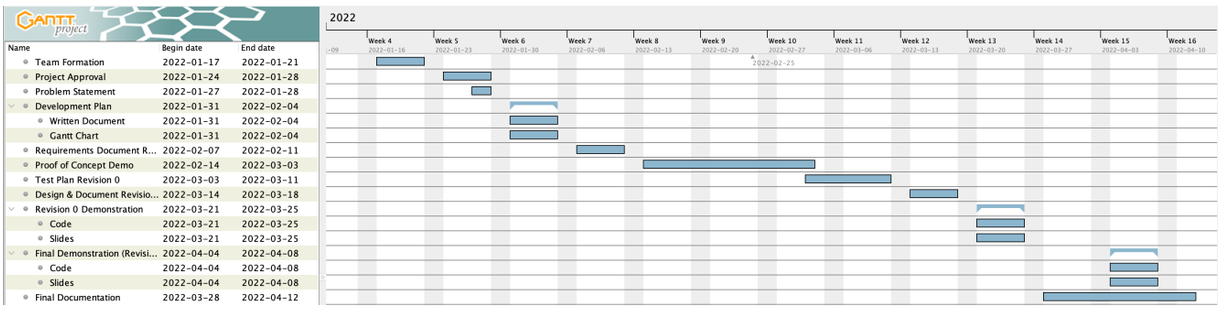

Figure 2: Gantt Chart.

4.5 Migration to the New Product

The new product will support newer versions of the original libraries that were built using the project.

4.6 Risks

The system is built on outdated versions of certain software, such as Gradle. This will hinder maintainability of the system. This will also hinder the ability to perform version control on the program.

4.7 Costs

There are no known costs for this project.

4.8 User Documentation and Training

- **Download Instructions:** The first set of documentation needed for the user is a document explaining how the game is to be downloaded and how users must go about configuring and running the game.
- **Walk-Through Video:** If a player has never played the game before than providing a walk through video should help a new player understand the game's objectives.
- **Help Drop-down:** During game play, if a player forgets what the controls are, then they should be able to hover over a help icon that outlines what keys to uses on your keyboard, as well as what those keys do in the game.

4.9 Waiting Room

- Create additional levels that matches levels from original game
- Create a special final level in which the objective is to defeat the final boss.

4.10 Ideas for Solutions

- Utilize object oriented features of Java and graphics library libgdx and reference material from the original game

5 Appendix

5.1 Symbolic Parameters

- TIME_RESPONSE - A time constant quantifying the amount of time it takes the system to respond to a user input.